TRACTABLE STRUCTURED PREDICTION USING THE PERMUTOHEDRAL LATTICE

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES OF ETH ZURICH
(DR. SC. ETH ZURICH)

presented by
MARTIN KIEFEL

Diploma in Computer Science, Karlsruhe Institute of Technology

born on 23.05.1984

citizen of Germany

accepted on the recommendation of
Prof. Dr. Thomas Hofmann, examiner
Prof. Dr. Bernt Schiele, co-examiner
Dr. Peter Gehler, co-examiner

2017
ABSTRACT

Computer vision reasons about the semantic content of visual data. This typically amounts to prediction tasks that infer multiple unknown objects from an image observation. For instance, the task of semantic segmentation assigns a label such as “street” or “car” to a pixel in a picture to describe its semantic property. Under certain conditions, it makes sense to leverage the dependencies between the output variables and predict the labels all at the same time. This usually is called structured prediction and this thesis considers two such tasks: human pose estimation and semantic segmentation.

In general there exists a modeling trade-off between being able to encode prior knowledge about the problem and to allow for tractable inference that derives the best prediction in a suitable amount of time. Furthermore, there is the problem of finding an estimator to set free model parameters based on training data. This work proposes a set of highly-connected graphical models and discusses their application to human pose estimation and semantic segmentation. These models exhibit a high level of flexibility to represent prior information about the correlation of output variables, e.g., the configuration of body parts, or label relations of nearby pixels. Tractable inference is possible in these models by using a variational approximation to the underlying probability distribution. Previous work has observed that a bilateral convolution in the permutohedral lattice accelerates the computation. This work extends this idea in the context of the aforementioned tasks. We put inference first and train model parameters end-to-end by directly optimizing a surrogate loss. More specifically:

- We present the Fields of Parts (FoP) model for human pose estimation. The binary Conditional Markov Random Field (CRF) is inspired by the idea of Pictorial Structures (PS) and similarly utilizes local appearance and spatial configuration of adjacent body parts. It opens up novel ways to encode color and segmentation information into the pairwise energy terms of the model and to modify the model structure to encode more statistical dependencies between body parts while still retaining fast inference times.

- Training of semantic segmentation models relies on pixelwise image annotations that—compared to other computer vision tasks—is avail-
able only in limited amounts. We leverage 3-dimensional annotations of point cloud data and transfer the label information to 400k image frames of street scenes using the dense graphical model inference presented in this work.

• We extend the permutohedral convolution to almost arbitrary convolutional kernels. Instead of using the permutohedral lattice to approximate the de facto standard use-case—the Gaussian bilateral blur—we take the viewpoint of defining the convolution in the lattice itself. This allows to learn dense pairwise potential functions in a CRF and has implications for CNN architectures and low-level image processing.
Das Forschungsfeld des Maschinellen Sehens behandelt semantische Eigen-
schaften in visuellen Daten. Dies beinhaltet typischerweise das Klassifizieren
von einer größeren Menge von unbekannten Objekten basierend auf der Be-
obachtung eines Bildes. Beispielsweise weist semantische Segmentierung je-
dem Pixel in einem Bild eine Klasse wie “Straße” oder “Auto” zu, um die
semantische Bedeutung festzulegen. Unter bestimmten Voraussetzungen ist
es vorteilhaft die Abhängigkeiten und Struktur der Ausgabevariablen auszu-
utzen und eine größere Anzahl von ihnen zusammen zu klassifizieren, was
als Struktur-Klassifikation oder structured prediction bezeichnet wird. Diese
Arbeit befasst sich mit zwei Instanzen dieses Vorgehens: Menschlicher Posen-
schätzung und semantischer Segmentierung.

Im Allgemeinen ist bei der Modellierung einer Problemstellung zwischen
Flexibilität, die es erlaubt Expertenwissen abzubilden, und effizienter Inferenz
abzuwägen. Zusätzlich stellt sich die Frage nach der Lernbarkeit von freien
Modellparametern. Diese Arbeit führt eine Klasse von dicht vernetzten grafi-
schen Modellen ein und diskutiert ihre Anwendung im Bereich der mensch-
lichen Posenschätzung und semantischen Segmentierung. Diese Modelle wei-
sen einen hohen Grad an Flexibilität auf, um Vorwissen über die Korrelation
von Ausgabevariablen, wie der Konfiguration von Gliedmaßen oder die Ähn-
llichkeit der Klassen benachbarter Pixel, auszunutzen. Effiziente Inferenz wird
durch eine Methode der Variationenrechnung möglich, die eine Approxima-
tion an die zugrundeliegende Wahrscheinlichkeitsverteilung bildet. Bisherige
Veröffentlichungen beschreiben Wege um die Berechnung mit der Hilfe einer
bilateralen Faltung auf dem Permutaedrischen Gitter zu beschleunigen. Die-
se Arbeit erweitert diese Idee in den zuvor genannten Anwendungsbereichen.
Der Inferenzalgorithmus spielt dabei eine zentrale Rolle und freie Modellpa-
rameter werden direkt über die Optimierung eines Fehlermaßes von Daten
gelernt. Konkret stellen wir folgende Modelle in dieser Arbeit vor:

- Für die menschliche Posenschätzung führen wir das Fields of Parts (FoP)
  Modell ein. Es ist ein binäres Conditional Markov Random Field (CRF),
  das auf der Pictorial Structures (PS) Idee fußt, und nutzt ebenfalls vi-
suelle Beschaffenheit und räumliche Konfiguration angrenzender Kör-
perteile. Es ermöglicht neue Arten Farb- und Segmentierungsinforma-
tionen in den paarweisen Energietermen des Modells zu verwenden und erlaubt es die Modellstruktur zu erweitern, um weitere statistische Abhängigkeiten zwischen den Körperteilen auszudrücken.

- Das Training von Segmentierungsmodellen ist auf annotierte Pixeldaten angewiesen, die im Vergleich zu anderen Anwendungsbereichen nur beschränkt verfügbar sind. Wir nutzen 3-dimensionale Punktwolken mit Klasseninformationen und übertragen diese in 400k Bilder mit Straßenszenen, wobei wir die vorgestellten Inferenzmethoden für stark verbundene grafische Modelle verwenden.

The following publications are included in parts or in an extended version in this thesis:


- Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger (2016). “Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer.” In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.


Furthermore, the following publications were part of my PhD research, are however outside of the scope of this work and are not covered in the text:


I would like to thank my advisor Peter Gehler for guiding me through the process of writing a thesis! I have certainly learned a lot from Peter and that does not only include computer vision and machine learning but also the fine techniques of mountain biking and latte art. It has been a pleasure to walk into his office and leave totally excited to try new things. Thank you Peter!

Both Michael Black and Bernhard Schölkopf have created an amazing environment to think and work. I am very grateful that I have been part of the empirical inference and perceiving systems labs of the Max Planck Institute in Tübingen. Thank you for creating places to meet so many smart people.

I would like to thank Thomas Hofmann and Bernt Schiele for agreeing to review my thesis and being a member of my thesis committee.

I would also like to thank Carsten Rother who supported me in my early PhD days. It has been amazing to work with Sebastian Nowozin and Jeremy Jancsary during my internship at Microsoft Research in Cambridge. It was a pleasure to work with such remarkable researchers.

I am also very grateful to have worked with Andreas Geiger and briefly join him on his quest for autonomous driving.

I would like to thank Philipp Hennig: he has changed the way I think about probabilistic reasoning. The countless discussions that we had about a myriad of different things will be something I will never forget. Philipp has never stopped to encourage me to continue my research!

Also, I would like to thank all members of the empirical inference and perceiving systems labs for making working there so enjoyable. I already miss the conversations at tea-talks, group presentations or the coffee machine.

Juli, Stephan, Kim, Sebastian, Bryan, Rebecca, Arne and many other friends helped to keep me sane and always joined for some great time in beer gardens or at a “Daddeldienstag”.

I am also deeply indebted to my family: Pia, Muriel, Christian, Peter, Teresa, Samson, Layla, Kilian, Moritz. Your support has made this thesis possible.

Above all, I would like to thank Nicki who has walked with me all the time. It is incredible that you have never stopped believing in me. I am so excited to see all the other adventures to come!
## CONTENTS

1 **INTRODUCTION** 1  
1.1 Making Structured Predictions 1  
1.2 Thesis Aims 3  
  1.2.1 Fields of Parts 4  
  1.2.2 Semantic Instance Annotation 5  
  1.2.3 Permutohedral Convolution 6  

2 **STRUCTURED PREDICTION** 9  
2.1 Setup 9  
  2.1.1 Loss Functions 10  
  2.1.2 Risk 11  
  2.1.3 Decomposing Loss Functions 12  
  2.1.4 Bayesian Decision Theory 13  
2.2 Graphical Models 15  
  2.2.1 Semantics of Factor Graphs 16  
  2.2.2 Representation of Arbitrary Distributions 17  
  2.2.3 The Energy Perspective 18  
  2.2.4 Markov Random Field 19  
2.3 Structured Inference 22  
  2.3.1 Sum-Product Algorithm 22  
  2.3.2 Max-Sum Algorithm 25  
  2.3.3 Mean Field Inference 26  
  2.3.4 Approximate Inference in Dense Graphs 29  
2.4 Learning 32  
  2.4.1 Probabilistic Parameter Learning 33  
  2.4.2 Expected Loss Minimization 34  

3 **HUMAN POSE ESTIMATION WITH FIELDS OF PARTS** 39  
3.1 Introduction 39  
3.2 Related Work 42  
3.3 Fields of Parts 44  
  3.3.1 Model 44  
  3.3.2 Discussion 47  
  3.3.3 Comparison to Pictorial Structures 51
## Contents

3.4 Learning and Inference 53
  3.4.1 Inference 53
  3.4.2 Parameter Estimation 56
3.5 Experiments 60
  3.5.1 Comparison to Pictorial Structures 61
  3.5.2 Comparison with State-of-the-Art 61
3.6 Future Work 64
3.7 Conclusion 65

4 Semantic Annotation of Street Scenes by 3D to 2D Label Transfer 67
  4.1 Related Work 69
    4.1.1 Methods 69
    4.1.2 Datasets 71
  4.2 Method 72
    4.2.1 Data Collection 72
    4.2.2 Annotation 73
    4.2.3 Model 74
    4.2.4 Learning and Inference 80
  4.3 Experimental Evaluation 81
    4.3.1 Quantitative Evaluation 82
    4.3.2 Qualitative Evaluation 88
  4.4 Conclusion 89

5 Permutohedral Convolutional Neural Networks 91
  5.1 Introduction 91
  5.2 Related Work 93
    5.2.1 Image Adaptive Filtering 93
    5.2.2 Dense Conditional Markov Random Field (CRF) 94
    5.2.3 Convolutional Neural Networks (CNNs) 95
  5.3 Learning Sparse High Dimensional Filters 95
    5.3.1 Permutohedral Lattice Convolutions 97
    5.3.2 Learning Permutohedral Filters 100
  5.4 Bilateral Filter Applications 101
    5.4.1 Image Denoising 102
    5.4.2 3-Dimensional Mesh Denoising 103
    5.4.3 Joint Bilateral Upsampling 105
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Human Pose Estimation</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Semantic Segmentation</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Example Factor Graph</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Single Node Factor Graph</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Image Denoising</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Example Markov Random Field</td>
<td>20</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Root Node Perspective in a Tree-Structured Graph</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Factor Node Perspective in a Tree-Structured Graph</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Dense Pairwise Graph</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Fields of Parts Model</td>
<td>41</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Image-Dependent Pairwise Potentials</td>
<td>49</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Qualitative Human Segmentation Results</td>
<td>50</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Segmentation Based Pairwise Potentials</td>
<td>51</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Loopy Extensions to the Fields of Parts Model</td>
<td>52</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Fields of Parts Inference Stack</td>
<td>53</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Qualitative Results for Leeds Sport Poses Dataset</td>
<td>57</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>Comparison to Yang and Ramanan (2013)</td>
<td>59</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Fields of Parts-Segmentation</td>
<td>62</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Dataset Annotation Trade-Off</td>
<td>68</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>3- to 2-Dimensional Label Transfer</td>
<td>70</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Label Transfer Model</td>
<td>74</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Geometric Unary Potentials</td>
<td>77</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Fold Potentials</td>
<td>79</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Performance with Respect to Estimated Pixels</td>
<td>83</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Qualitative Results</td>
<td>86</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Schematic of the Permutohedral Convolution</td>
<td>96</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Visualization of the Permutohedral Lattice</td>
<td>98</td>
</tr>
</tbody>
</table>
Figure 5.3  Illustration of a 1-Dimensional Permutohedral Lattice Construction 99
Figure 5.4  4-Dimensional Isomap Features for 3-Dimensional Human Bodies 103
Figure 5.5  Sample Denoising Result 105
Figure 5.6  Guided Upsampling 107
Figure 5.7  Guided Depth Upsampling 109
Figure 5.8  Semantic Segmentation Results 112
Figure 5.9  Material Segmentation Results 114
Figure 5.10  Sample Assamese Character Images 118

LIST OF TABLES

Table 3.1  Leeds Sport Poses Results for Fields of Parts in Average Precision of Key-Points 63
Table 3.2  Leeds Sport Poses Results for Fields of Parts in Percentage of Correct Parts 63
Table 4.1  Comparison to Label Transfer Baselines on Semantic Segmentation Task 84
Table 4.2  Ablation Study on Semantic Segmentation Task 85
Table 4.3  Ablation Study on Instance Segmentation Task 88
Table 5.1  Peak Signal-to-Noise Ratio Results of a Denoising Task 102
Table 5.2  Body Denoising 104
Table 5.3  Joint Bilateral Upsampling 106
Table 5.4  Color Upsampling with Different Train and Test Upsampling Factors 108
Table 5.5  Learned Potentials for Semantic Segmentation 110
Table 5.6  Learned Potentials for Material Segmentation 113
Table 5.7  Runtime Comparison: Bilateral Convolution Layer vs. Spatial Convolution 116
Table 5.8  Classification Accuracy on MNIST 117
Table 5.9 Results on Assamese Characters 119

**LIST OF ALGORITHMS**

<table>
<thead>
<tr>
<th>Algorithm 2.1</th>
<th>Mean Field Update</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.2</td>
<td>Dense Mean Field Update</td>
<td>32</td>
</tr>
<tr>
<td>Algorithm 3.1</td>
<td>Mean Field Update in the Fields of Parts Model</td>
<td>55</td>
</tr>
</tbody>
</table>
INTRODUCTION

1.1 MAKING STRUCTURED PREDICTIONS

Teaching computers or machines to see or rather connecting an image with its semantic properties is one of the core questions of computer vision. Usually, computer vision is part of a bigger agent system that navigates the world and takes actions based on its inputs. Thus, finding objects in a pictured scene, predicting physical properties of single pixels or understanding the 3-dimensional scene layout are examples of problems computer vision is solving. This work primarily focuses on tasks with output values with an internal structure and calls these special tasks structured prediction problems.

Instead of deriving a fixed rule from experts’ knowledge to convert input images to output predictions, this thesis describes learning systems that come up with rules driven by data. Hence, the goal of structured output learning is to find a function

\[ h : \mathcal{X} \rightarrow \mathcal{Y} \]

that takes an image observation from the space \( \mathcal{X} \) as input and produces a label from \( \mathcal{Y} \). The label space might grow exponentially fast in some problem specific number, e.g., the pixel count in an image. In contrast to regular multiclass classification that maps each image to an element from \( \mathcal{Y} \), structured prediction extends this idea by relating the elements from the label space to one another. Understanding closeness of labels—without a formal definition of what closeness means here—allows us to train a classifier from data sets which are much smaller than the number of possible labels in \( \mathcal{Y} \). In general, the output space \( \mathcal{Y} \) can be anything from a set of graphs, strings or other structured objects. Furthermore, the presented applications in this work feature parts of the underlying structure, which take labels from a per-part label set \( \mathcal{L} \) and compose the classification vectors from \( \mathcal{Y} := \mathcal{L}^d \).

The special case of a decomposing label space is a central object of this work. In the case of limited training data or if the evaluation method does not decompose in an additive way over this vector structure of \( \mathcal{Y} = \mathcal{L}^d \) it can be advantageous for \( h \) to predict a consistent full vector from \( \mathcal{Y} \) instead of an
Figure 1.1: Human pose estimation from a single monocular image. An example input image from Ramanan (2006) can be seen in (a) with its ground-truth annotation in (b). Key-points on the joints of the person depict shape and articulation of the pose. In this dataset the annotations include locations for the top of the head, neck, shoulders, elbows, wrists, upper-legs, knees, and ankles. Lines connect the different key-points to create a stick-man for a better visualization on the gray-scale image on the right.

Human pose estimation and semantic segmentation are two examples of successful structured prediction applications in computer vision and are a focus of this work.

Predicting the location of human body key-points for human pose estimation like the top of the head or elbows from single images (e.g., see fig. 1.1) is a challenging inference problem and serves as a crucial prerequisite step to high level vision applications, e.g., action recognition. The complexity is due to the weak visual appearance of arms and legs, which sometimes only cover a small area in an image or are occluded by the torso or other objects in a scene. Furthermore, the projection of the body from a 3-dimensional space down to the 2-dimensional image plane foreshortens arms and legs when they are pointing towards the camera. Instead of predicting the location of each body key-point individually we can model the relation between them and predict the location for all at the same time. This allows us to encode additional domain knowl-
edge. For example, we expect that the head of a person does not appear in an arbitrary distance from the person’s neck.

A common loss that measures the performance of inference algorithms for human pose estimation is Average Precision of Key-Points (APK). It counts the number of correctly predicted locations of body parts. Even though this loss decomposes over the structure of the problem—it is a sum iterating over the key-points—joint modeling leads to an improved performance in the case of limited training data.

Semantic segmentation considers the task of assigning a semantic label, e.g., “street”, “car”, to each pixel in a given image as shown in fig. 1.2. Annotation of a full image by humans can take a considerable amount of time and other means of making more information available to the learner can lead to better performance and compensates the missing training data. Similar to the parts of human pose estimation, nearby pixel exhibit a strong correlation and often share the same label. Structured prediction exploits this by predicting all labels of an image at the same time by finding a label set that satisfies both local image evidence and an overall smoothness.

Graphical models (e.g., Koller and Friedman (2009)) provide a common language to describe probabilistic models and inference and provide a key tool in this work. In particular, Conditional Markov Random Fields (CRFs) have been successfully applied to computer vision problems. In their factor graph interpretation, graphical models connect nodes with factors to model the probabilistic dependencies between random variables. The inference process usually results in message passing algorithms between nodes and becomes more involved if loops in the graph exist. Approximate inference solves this issue, but by its nature changes the model class in a way that is hard to quantify.

Additionally to the choice of the model class and inference algorithm, there is a set of free parameters that are not known beforehand. In the context of this work we assume that there is a set of training labels to fit these parameters. In the light of approximate inference it is not trivial to set those free model parameters from training data.

1.2 Thesis Aims

The goal of this thesis is to develop efficient inference algorithms for the problems of human pose estimation and semantic segmentation—two structured prediction problems. This work is inspired by the literature for dense factor
(a) Input (b) Ground Truth

Figure 1.2: Semantic segmentation assigns a semantic label to the pixels in the image. The picture in (a) from Everingham et al. (2010) shows the two classes “bird” and “background” that have been annotated by humans, which is shown in (b). There is an additional margin between the classes (white) where prediction performance is not measured.

graphs (Krähenbühl and Koltun, 2011), which makes extensive use of a permutohedral lattice approximation to Gaussian blurs.

1.2.1 Fields of Parts

The recent advances for dense CRF inference (Krähenbühl and Koltun, 2011) allows more model flexibility for human pose estimation. The contribution of chapter 3 is two-fold:

On the one side, the thesis transfers the possibilities of encoding prior knowledge of segmentation problems to physical constraints in human pose estimation models. The idea here is to encode the presence and absence of each body part as a binary variable for every possible location, orientation, or scaling of a part. This leads to a model with a large number of variables that describe a state with a binary label, unlike the prominent Pictorial Structures (PS) model that represents each part with a single random variable with a large state space. We call the sets of variables that each describe the state of a part as Fields of Parts (FoP). Moreover, the model treats background and foreground, i. e., presence or absence of a body part, symmetrically through the binary label space. This leads to a novel and more complete interpretation of the image.

On the other hand inference plays an important role. The focus of this work is an algorithm that takes advantage of the acceleration technique proposed by Krähenbühl and Koltun (2011). We extend their technique to cover a mean field message passing mechanism between body part fields and inside
body part fields. Vineet, Sheasby, et al. (2013) present a similar idea, but do not generalize it to part fields. Furthermore, the thesis draws a connection to marginal based learning techniques (Domke, 2011; Domke, 2013). Domke (2013) advocates that learning should both take the desired loss function as well as the approximate nature of the inference procedure into account. He calls it back-mean-field learning, which is also mentioned in Krähenbühl and Koltun (2013). We use this type of unrolled inference to train a classifier for losses used in human pose estimation and close a loop between the approximate inference technique and parameter learning.

1.2.2 Semantic Instance Annotation

The recent learning approaches for image classification, object detection or image segmentation of Krizhevsky, Sutskever, and Hinton (2012), Zhu et al. (2015), and Long, Shelhamer, and Darrell (2015) show impressive results and rely on high-capacity deep learning architectures with a large quantity of free parameters. Large training sets with image counts in the millions like ImageNet (Russakovsky et al., 2015) for image classification make it possible to set these weights. The power of these models to digest these large datasets also raises the question how to get other task specific datasets. The common approach to build a ground truth is to ask human annotators to give the right answer. While this is still possible for non-structured task like image classification, the method does not scale in the same way for structured prediction tasks. These tasks are more time-consuming and labor intensive; in particular image segmentation requires pixel-level answers, which can take around 60 minutes per image depending on the annotation method.

There are other means to collect large amounts of training data: In this work we build a new dataset for image segmentation with both per-pixel object class and instance-level annotations for street scenes by first solving a simpler proxy task. We leverage labeled 3-dimensional point cloud data that has been recorded in suburban street scenes. This has at least three advantages over the direct approach that targets the individual image pixels in the 2-dimensional projection: First, especially for the case of video data we can connect a single 3-dimensional model with multiple image frames. Thus, it quickly pays off to annotate a single 3-dimensional scene and transfer that information, rather than handling the images individually. Second, a 3-dimensional seed scene provides consistency between the image frames. In particular in the case of instance labels. Furthermore, there is consistency between the image frames
and the point cloud. Third, occlusion handling, which is often hard to solve in 2 dimensions, becomes easier for mostly-convex objects in 3 dimensions.

The transfer reasoning of the 3-dimensional scene annotation towards 2-dimensional pixel labels uses the findings of the FoP model that is presented in chapter 3. Our model connects each of the 3-dimensional observations with every pixel in the image frames to infer the correct labeling. We also show how to include latent properties about the scene into the model’s reasoning.

We summarize the contributions of chapter 4:

- We present a novel geo-registered dataset of suburban scenes recorded by a moving platform. The dataset comprises over 400k images and over 100k laser scans, and we provide semantic 3-dimensional annotations for all static scene elements.

- We propose a method which is able to transfer these labels from 3 dimensions into 2 dimensions, yielding pixelwise semantic instance annotations. We demonstrate the potential of our approach in ablation studies and with respect to several 2-dimensional and 3-dimensional baselines.

1.2.3 Permutohedral Convolution

Both previous applications in human pose estimation and semantic segmentation rely on Gaussian convolutions that are defined on the permutohedral lattice. This thesis extends the convolution operation to general parameterized kernels in chapter 5.

Krähenbühl and Koltun (2011) propose an acceleration technique for mean field approximations in the case of CRF models with Gaussian edge potentials. The permutohedral bilateral blur (Adams, Baek, and Davis, 2010) speeds up the computation of the messages in the approximative message passing scheme. This thesis augments the class of densely connected graphical models with tractable inference with highly parameterized pairwise potential functions.

The result has consequences for Convolutional Neural Networks (CNNs) as well. The non-linear classifiers improved previous state-of-the-art considerably on computer vision tasks, e.g., object classification (Krizhevsky, Sutskever, and Hinton, 2012), and consist of smaller max-pooling, convolution, or soft-max operations, among others. A convolutional layer of a CNN is almost exclusively associated with a spatial convolution in the image domain; apart from the notable exception of Bruna et al. (2013). This means that the
image signal is sampled at regular grid locations. The novel approach of this work takes a signal-theoretic viewpoint and generalizes the convolutional operation to sparse input data that is sampled at irregular locations and potentially lives in a high-dimensional space. In the most general form the discrete version of the convolution computes

\[ v_j' = \sum_{j=1}^{n} k(z_i - z_j) v_i \quad \forall j, \]

where it convolves the input signal \( v_i \) with a filter kernel \( k \) to reach the result \( v_j' \). This work only considers stationary kernels and therefore the kernel weight only depends on the difference of the two feature vectors \( z_i \) and \( z_j \). This form includes the spatial convolution as a special case.

Another area that profits from developments of high-dimensional filtering operations is the original use-case of the bilateral filter, i.e., as an edge preserving smoothness operation without a specific kernel in mind. Nevertheless, the de facto standard application is a Gaussian bilateral filter. A more general but not less efficient bilateral filtering operation with a learnable filter has also consequences for low-level vision applications, e.g., image denoising. In particular this work builds on and is a direct generalization of the original blur (Aurich and Wuele, 1995; Smith and Brady, 1997; Tomasi and Roberto, 1998).

We summarize the consequences of a generalized permutohedral convolution as follows:

- An application to bilateral filtering makes learned filter operations accessible to low-level vision tasks.

- The richer kernel class inside the permutohedral convolution is a direct generalization of the result from Krähenbühl and Koltun (2011) from Gaussian edge potentials to highly parameterized pairwise potentials.

- A generalization of the spatial convolution layer in CNNs extends modeling flexibility and allows them to process sparse features with image adaptive receptive fields.
STRUCTURED PREDICTION

This chapter introduces main ideas from literature to set the stage for the contributions of this work. We divide this chapter into four parts. The first part covers fundamental ideas about learning systems and in particular shows the connection of a task loss function to structured prediction. The rest of the chapter turns to the steps to solve a problem: modeling the problem with domain knowledge, running inference to find answers and learning to fit free parameters of a model from data.

2.1 SETUP

One of the central questions in computer vision is the inference of semantic properties from images. This is true for the tasks of human pose estimation and semantic segmentation; both share the property that a prediction exhibits a vector structure in contrast to a single classification. Human pose prediction locates every part of the human body from a single image and semantic segmentation assigns a semantic label to every pixel in the image.

Figure 1.2 contains an example ground-truth annotation for a semantic segmentation problem. Zooming in on the pixels reveals that the annotation features a dependency: nearby pixels are likely to share the same label. In other words, there is structure in the output space of the appearing labels. One of the main themes of this work is to develop methods that make use of this knowledge to either regularize the solution or solve for consistent structures. The next paragraphs elaborate when it actually makes sense to use this structure and define further machine learning concepts.

More formally: The input space $\mathcal{X}$ collects all possible input images and the set $\mathcal{Y}$ gathers the task specific multidimensional labels. Structured output learning answers the question on how to find a good predictor function

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$  \hspace{1cm} (2.1)

from a set of possible hypothesis $\mathcal{H}$. The set $\mathcal{Y}$ is able to cover both discrete and continuous label spaces and we call these setups classification or regression respectively. However, for the rest of this text the output space $\mathcal{Y}$ does only represent discrete and finite label spaces unless stated otherwise.
In this work each hypothesis is based on a scoring function $g$ that compares all possible outcomes from $Y$ for a given input $X$ in $h$

$$h(x) = \arg\max_y g(x, y),$$

(2.2)

to find the best answer $y$ that achieves the highest score.

Up to now there is no way to distinguish the predictors in $\mathcal{H}$, or put differently, no obvious solution how to set the scoring function $g$ apart from using possible domain knowledge about the problem. The domain knowledge prefers certain hypotheses over others without having seen any samples from the joint distribution over $X$ and $Y$. This distribution is not revealed directly and there is only access to a sample of i.i.d. image and label pairs $(x_i, y_i)$, $i \in \{1, \ldots, n\}$. The data builds the basis for selecting a good predictor from the hypothesis space. The hope is that a predictor that works well on training data will also work well on unseen samples from the joint distribution, i.e., it generalizes to the full distribution.

The exponentially large label space $Y$ renders a regular multi-label prediction approach intractable. Depending on the learning objective it treats each of the labels from $Y$ as a separate problem, which requires large training sets to distinguish all the possible outcomes. Structure output learning ties a connection between the outcomes and intuitively learns multiple labels per training sample. It is the loss function that defines which structures of the predictions are important and that relates elements in $Y$.

2.1.1 Loss Functions

The loss function is part of a prediction task and it evaluates the performance of the final predictor

$$\Delta(\hat{y}, y^*) : Y \times Y \to \mathbb{R}^+.$$  

(2.3)

It assigns a non-negative loss value to the prediction $\hat{y} = h(x), x \in X$, in the case that the true underlying label is $y^* \in Y$. The lowest possible loss of 0 is at least reached if both arguments are equal to $y^*$. The zero-one loss and the Hamming loss are two widely applied examples.

**zero-one loss**

**ZERO-ONE LOSS:** A common loss in classification is the zero-one loss

---

1 For theoretical results on learnable problems it is often require that the loss is bounded. See Vapnik (1995) for more details.
\[ \Delta_{0/1}(\hat{y}, y^*) := 1 - 1 \left[ \left[ \hat{y} \right]_j = [y^*]_j, \forall j \in \{0, \ldots, d\} \right], \tag{2.4} \]
which incurs a cost of 1 every time there is at least one component of the label vector that does not agree with the ground-truth annotation. Thus, only a perfect match results in the loss of 0.

The indicator function \( \mathbb{1} \) takes the value 1 if the Boolean argument evaluates to \( True \). Also, eq. (2.4) makes use of \( [\cdot]_j \) which looks up the \( j \)-th component of the vector it encloses.

**Hamming Loss:** In semantic segmentation and other structured prediction tasks, however, it is likely that an error in a small number of elements would go unnoticed by a human observer. So rather than expecting the predictor to perfectly forecast the full vector, the task’s loss measures the percentage of correctly predicted labels of \( \hat{y} \) for a given ground-truth annotation \( y \) instead. This common error measure is also known as the Hamming loss

\[ \Delta_{\text{Hamming Loss}}(\hat{y}, y^*) := 1 - \frac{1}{d} \sum_{j=0}^{d} \mathbb{1} \left[ \left[ \hat{y} \right]_j = [y^*]_j \right]. \tag{2.5} \]

### 2.1.2 Risk

In general, a predictor should work well on the joint distribution over images and labels and thus this turns the loss value itself into a random variable \( \Delta(h(X), Y) \). The expectation of this random variable is the risk of a predictor \( h \), i.e., the expectation over the loss computed by \( h \),

\[ \mathcal{R}_P(h) := \mathbb{E}_{(X, Y) \sim P} \left[ \Delta(h(X), Y) \right]. \tag{2.6} \]

The unobserved joint probability distribution \( P \) makes it impossible to evaluate this quantity directly. The empirical risk takes the training samples as input and is an approximation to the true risk in eq. (2.6)

\[ \hat{\mathcal{R}}_P(h) := \frac{1}{n} \sum_{i=0}^{n} \Delta(h(x_i), y_i). \tag{2.7} \]

In general selecting the hypothesis that performs best with respect to eq. (2.7) does not lead to the best hypothesis from \( \mathcal{H} \). An over-fitting learner that memorizes all training examples performs perfectly with respect to eq. (2.7) but is likely to fail on unseen examples from the data distribution. Statistical
structured prediction learning theory (e.g., Vapnik (1995)) describes prerequisites for a generalizing learner where the ability to be successful depends on the chosen hypothesis class $\mathcal{H}$.

### 2.1.3 Decomposing Loss Functions

In the previous definitions the output structure of the problem did not play a prominent role. This section tries to convey some intuition for when structured learners will yield good predictors that take dependencies in the output vectors into account. Losses that decompose in an additive way over the output variables build cases where structured prediction methods are not strictly necessary. Put differently, if the evaluation metric is not considering dependencies between the output variables—apart from summing up the losses—then it will not be necessary for the predictor to consider dependencies between the variables during the inference process. These special losses can be formally described with

$$\Delta(\hat{y}, y^*) = \sum_{j=0}^{d} \Delta_j(\hat{y}_j, y^*_j). \tag{2.8}$$

We abbreviate $\hat{y}_j := [\hat{y}]_j$ and $y^*_j := [y^*]_j$ when focusing on a single element of $\hat{y}$. The risk of a given predictor $h$ with respect to the true underlying probability distribution $P(Y \mid X)$ with a Probability Mass Function (PMF) $p(y \mid x)$ reveals,

$$R_P(h) = \mathbb{E}_{(X,Y) \sim P} \left[ \sum_{j=0}^{d} \Delta_j([h(X)]_j, [Y]_j) \right]$$

$$= \sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \sum_{j=0}^{d} \Delta_j([h(x)]_j, y_j) \tag{2.9}$$

$$= \sum_{j=0}^{d} \sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \Delta_j([h(x)]_j, y_j).$$
Separating the sum over $y$ into $y_j$ and all other variables $[y]_{\setminus j}$ and rearranging the terms yields

$$\mathbb{R}_P(h) = \sum_{j=0}^{d} \sum_{x \in \mathcal{X}} \sum_{y_j \in [y]_j} p(x) p(y_j | x) \Delta_j ([h(x)]_j, y_j)$$

$$\sum_{[y]_{\setminus j} \in [y]_{\setminus j}} p([y]_{\setminus j} | y_j, x)$$

$$= \sum_{j=0}^{d} \sum_{x \in \mathcal{X}} \sum_{y_j \in [y]_j} p(x, y_j) \Delta_j ([h(x)]_j, y_j)$$

$$= \sum_{j=0}^{d} \mathbb{E}_{(x, [y]_j) \sim p} \left[ \Delta_j ([h(X)]_j, [Y]_j) \right].$$

Thus, the risk $\mathbb{R}_P(h)$ of the structured predictor decomposes over the problem structure in the same way as the loss does. As all terms from the sum in the last line are independent from each other, the inference can treat them separately and a non-structured method can substitute the best performing structured prediction method while achieving the same or a better performance.

However, even though losses in computer vision often decompose over the element structure of the label space it can make sense to apply structured prediction. All the above arguments hold only for the full probability distribution and not for finite-sample scenarios of applications. Taking the output space dependencies into account might act as a model regularization.

2.1.4 Bayesian Decision Theory

The inference algorithms of this work make use of probability theory, which—founded on desiderata—suffices to describe our beliefs about the state of the world (Jaynes and Bretthorst, 2003). Bayesian decision theory tells us how to turn a belief $Q$ into predictors that choose a concrete label given some input. Our best guess $h_Q(x)$ picks the label that incurs the lowest expected conditional loss

$$h_{Q,\Delta}(x) = \arg\min_{\hat{y}} \mathbb{E}_{Y \sim Q(\cdot | x=x)} [\Delta(Y, \hat{y})]$$

$$= \arg\min_{\hat{y}} \sum_{y \in \mathcal{Y}} \Delta(\hat{y}, y) q(y | x).$$

(2.11)
Depending on the loss function, the answer may vary and the derivation of the best prediction for the zero-one loss in eq. (2.4) and the Hamming loss in eq. (2.5) result in two commonly used inference methods.

**Zero-One Loss:** Inserting the zero-one loss into eq. (2.11) yields

\[
h_{Q,\Delta_{0/1}}(x) = \arg\min_{\hat{y}} \mathbb{E}_{Y \sim Q(Y \mid x=x)}[\Delta_{0/1}(Y, \hat{y})] \\
= \arg\min_{\hat{y}} \sum_{y} \Delta_{0/1}(\hat{y}, y) \ q(y \mid x) \\
= \arg\min_{\hat{y}} q(\hat{y} \mid x).
\]

The best forecast that a predictor can make is the Maximum a-Posteriori (MAP) state; the state with the highest probability according to the belief. Section 2.3.2 presents the max-sum algorithm that finds this state for the class of graphical models.

**Hamming Loss/Decomposing Losses:** With reference to the Hamming loss or losses that decompose along the elements of the prediction vector (cf. eq. (2.8)), Bayesian decision theory gives a different recommendation compared to the zero-one loss. Considering the \( j \)th-entry of the predictor,

\[
\begin{align*}
[h_{Q}(x)]_j &= \left[ \arg\min_{\hat{y}} \mathbb{E}_{Y \sim Q(Y \mid x=x)}[\Delta(\hat{y}, Y)] \right]_j \\
&= \left[ \arg\min_{\hat{y}} \sum_{y \in Y} \Delta(\hat{y}, y) \ q(y \mid x) \right]_j \\
&= \left[ \arg\min_{\hat{y}} \sum_{y \in Y} \sum_{j=0}^{d} \Delta_j(\hat{y}, y_j) \ q(y \mid x) \right]_j,
\end{align*}
\]
and splitting the summation of $\mathbf{y}$ similar to eq. (2.9) into $y_j$ and all other variables $[\mathbf{y}]_\setminus j$ leads to

$$
\left[ f_Q(x) \right]_j = \left[ \arg\min_{\mathbf{y}} \sum_{j=0}^{d} \sum_{y_j \in [Y]_j} \Delta_j(\hat{y}_j, y_j) \ q(y_j | x) \right]_j = \left[ \arg\min_{\mathbf{y}} \sum_{j=0}^{d} \mathbb{E}_{[Y]_j \sim Q(\cdot | x=x)} \left[ \Delta_j(\hat{y}_j, [Y]_j) \right] \right]_j = \arg\min_{\hat{y}_j} \mathbb{E}_{[Y]_j \sim Q(\cdot | x=x)} \left[ \Delta_j(\hat{y}_j, [Y]_j) \right].
$$

(2.14)

Thus, the marginal of the $j$th-element of the posterior distribution over $Y$ suffices to find the prediction that incurs the lowest expected loss. Furthermore, the optimal prediction based on two different distributions $Q$ and $\overline{Q}$ is the same as long as they share the same marginals in $[\hat{y}]_j$. This is true even if $Q$ features an inter-dependence structure between its variables and $\overline{Q}$ fully factorizes over all variables with a PMF

$$
q(y_j | x) = \prod_{j=1}^{d} q_j(y_j | x).
$$

(2.15)

The sum-product algorithm in section 2.3.1 computes the marginals for distributions in graphical models that exhibit a tree structure and mean field inference, which is subject to section 2.3.3, extends this insight to more general distributions with the help of approximation techniques.

## 2.2 Graphical Models

In their most general form, probability distributions are not restricted in any way how they describe inter-dependencies between random variables. However, inferring the state of a system, let alone describing the interplay between the variables becomes intractable due to the combinatorial explosion of the state space. Graphical models are a tool to specify inter-dependencies in data and they are briefly introduced in this section. Bishop (2006), Koller and Friedman (2009), and Barber (2012) are references for an in-depth discussion.

Graphical models represent the distributions with graphs $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, which consist of a set of vertexes $\mathcal{V}$ and edges $\mathcal{E}$ between vertexes. They form an expressive language for inference algorithms that operate locally on the graph,
structured prediction

Figure 2.1: The pictured factor graph is made up by the variable nodes $Y_1, \ldots, Y_5$ and the factor nodes $f_1, \ldots, f_4$, that are drawn as circles and filled squares respectively. The bipartite graph has only connections between factor nodes and variable nodes.

passing messages between nodes. The graphical representation relates graph theoretic properties, e.g., existence of loops, with algebraic characteristics of the multivariate distribution.

This work focuses on a specific type of graphical models: factor graphs. Circular variable nodes and factor nodes, which are recognizable by squares, build the set of vertexes for these undirected graphs. For an example graph see fig. 2.1. Directed and undirected graphical models are two more examples of graphical formalism expressing distributions but do not play an important role in this work. Bishop (2006) and Koller and Friedman (2009) contain more details about them.

The next two sections address the translation between the two worlds: First, section 2.2.1 discusses how the functional form of a PMF can be read off from a graph. Then, section 2.2.2 shows whether and how a factor graph decomposes a given probability distribution.

2.2.1 Semantics of Factor Graphs

Every factor graph corresponds to a Gibbs distribution, which has a PMF $p(y)$ of the form

$$p(y) := \frac{1}{Z} \prod_{k=1}^{m} f_k(s_k)$$

$$Z := \sum_{x \in \mathcal{X}} p(y).$$

The PMF is a product of factors $f_k$, $k \in \{1, \ldots, m\}$, which are non-negative factor functions. A scope of a factor is the set of arguments $s_k \subseteq \{y_1, \ldots, y_d\}$ to
2.2 Graphical Models

2.2.1 Representation of Joint Distributions

It is true that the class of factor graphs is covering the full domain of probability distributions. Thus, there is no probability distribution that cannot be represented by a factor graph (see fig. 2.2). Every factor graph is a representative for a set of probability distributions that follow a specific algebraic shape; precisely those that can be written in the graph’s Gibbs distribution. Is there an easier way to identify the distributions than by inspecting their algebraic...
form? The Hammersley-Clifford theorem (see Koller and Friedman (2009, Theorem 4.8) for a proof) characterizes all distributions that are covered by the graph’s interpretation.

Given a positive distribution $P$ over the random variables $Y$ and a factor graph $G$ with the same variable nodes $Y$, if $P$ shares all conditional independences that the graph $G$ induces, then $P$ is a Gibbs distribution of $G$ and factorizes over its structure.

This theorem is one of the main motivations for graphical models. It gives the specific conditions for a probability distribution so it can be written in the shape of eq. (2.16). It allows us to think in terms of factor graphs to model our domain knowledge and rest assured that the graph will cover the true underlying distribution if it features the induced conditional independence properties by the graph.

### 2.2.3 The Energy Perspective

Factor graphs provide a tool to describe PMFs in a flexible way. But they do not restrict the functional form of the involved factors. This section presents a different perspective to factors and introduces a parameterized form of probabilistic models, which later is trained from data.

As a notational convenience it is advantageous to define factors as an argument to the exponential function in the log-space

$$f(s) = \exp(-\phi(s)),$$

(2.18)

which additionally ensures positivity of the factor. The function $\phi$ is called the energy or potential function of a factor $f$ for reasons that are beyond this
Figure 2.3: Image denoising infers the original image (a) from Everingham et al. (2010) from its noisy observation (b). The chance that a pixel flips its label is set to a probability of 0.1 in this example.

monograph. But both styles of writing a factor are equivalent and the Gibbs distribution from eq. (2.16) changes to

$$p(y) = \exp\left(-\sum_{k=0}^{m} \phi_k(s_k) - \ln(Z)\right)$$

$$\propto \exp\left(-\sum_{k=0}^{m} \phi_k(s_k)\right),$$

where the last line establishes the equality only up to an unknown scalar.

The derived class of log-linear models introduces free parameters as weights $w$ of feature functions $\phi$. This changes the PMF from eq. (2.19) to

$$p(y \mid w) \propto \exp\left(-\sum_{k=0}^{m} w_k \phi_k(s_k)\right).$$

2.2.4 Markov Random Field

Graphical models are generic in nature and are not aimed at a specific computer vision or low-level vision task. However, with their repetitive construction and their common 2-dimensional layout, Markov Random Fields (MRFs) gather a class of graphical models which have an arguable success in computer vision. Figure 2.4 depicts a graphical model that shares those properties and is discussed in more depth in the next paragraphs as a modeling example.

The Ising model expresses interactions of binary random variables and
Figure 2.4: A Markov Random Field (MRF) is a special graphical model with a repetitive structure. The pictured instance connects all variables \( Y \) with a 4-connected graph and binds the variables to corresponding variables from \( X \). Observing \( X \) turns the probability distribution of interest into a conditional distribution which lends its name to the emerging Conditional Markov Random Field (CRF).

Therefore is well suited for the low-level vision task of image denoising. It reasons about the original and unobserved image from a noisy picture observation. Since the observation turns the probability distribution underlying the graph into a conditional expression it make sense to talk about a Conditional Markov Random Field (CRF).

We assume a per-pixel noise that flips the binary state of pixels in the original image with a probability \(< 0.5\). Figure 2.3 shows an example image and its corrupted version. The reconstruction approach leverages the knowledge that pixels that are direct neighbors usually share the same label in the original image. All other information about the corrected version in the variables \( Y \) comes from the noisy and observed input variables \( X \).

For each pixel location \( j \) there is a cost of \( u \) for a disagreeing label \( y_j \) from the noisy observation \( x_j \), which leads to the following pixel-wise potential

\[
\phi_j(y | x) = \begin{cases} 
0, & \text{if } x = y \\
u, & \text{otherwise}
\end{cases}, \quad \forall j \in \{0, \ldots, d\}, u \geq 0. \tag{2.21}
\]

Right now, the model’s best guess is to stick with the noisy label, which is the most probable solution when neighboring pixels are not considered.
To achieve smoothing, the model considers the surrounding of a pixel $j$ by assigning a cost if pixels do not agree on a label. In this example the direct influence of one pixel is only restricted to its neighbors in a 4-connected graph, cf. fig. 2.4,

$$
\phi_{j,j'}(y, y') = \begin{cases} 
0, & \text{if } x = x', \forall j, j' \in \{0, \ldots, d\}, v \geq 0. \\
v, & \text{otherwise}
\end{cases}
$$

(2.22)

Combining the potentials from eq. (2.21) and eq. (2.22) leads to the following energy function:

$$
E(x | y) = \sum_{j=1}^{d} \phi_j(y_j | x_j) + \sum_{j=1}^{d} \sum_{j'=1}^{d} \phi_{j,j'}(y_j, y_{j'}). 
$$

(2.23)

Also, a pixel's output label variable $Y$ is independent of any other output label $X'$ given its immediate 4 neighbors. This follows from the discussion in section 2.2.1 about independence properties of the probability distributions that are factorized by the graph. This strong assumption is likely violated for real images and section 2.3.4 suggests a more flexible model with more connections to solve this problem.

When talking about the different potentials of a model, the literature usually groups the energy functions based on the number of connected output variables. Unary potential and pairwise potentials connect to a single or a pair of unobserved variables respectively. In the above example all $\phi_j$ are unary potentials and describe a single $y_j$ in the output space. All $\phi_{j,j'}$ are pairwise potentials and they characterize the energy states of the two output variables $y_j$ and $y_{j'}$.

Although, only a single output variable is in the scope of a unary potential, it may include the full set of input or observed variables. More precisely, even though a single output variable parametrizes a unary potential, e.g., the semantic label of a single pixel, or the value of a reconstructed pixel in the Ising model, it can take a full peak at the observed image. In the above example this flexibility is not used and in general the region of influence for one output pixel variable is restricted to a context around the location of that pixel. A similar reasoning applies to pairwise potentials, too.
2.3 Structured Inference

Factor graphs provide a powerful tool to model distributions of data through their link of algebraic properties and the structure of the graph. This connection extends to inference algorithms, too, and especially for tree-structured graphs marginal inference (eq. (2.14)) and MAP inference (eq. (2.12)) the computation is both efficient and exact.

2.3.1 Sum-Product Algorithm

Marginal inference integrates out all nuisance variables and it is part of the optimal answer for loss functions that decompose over the variables of the distribution (cf. eq. (2.14)).

The sum-product algorithm computes the marginals through a message passing scheme between the nodes of the graph and is briefly introduced in this section. Its generic nature makes it applicable to a wide range of problems and it was presented in the seminal work of Lauritzen and Spiegelhalter (1988).

All steps of this algorithm act locally on parts of the graph while traversing over its tree-structure to compute the marginal of \( Y^j \) of a multivariate probability distribution of \( Y \)

\[
p(y_j) = \sum_{[y] \backslash j} p(y). \tag{2.24}
\]

As described in section 2.2 the PMF of the graph is a Gibbs distribution from eq. (2.16) and plugging it into eq. (2.24) yields

\[
p(y_j) = \sum_{[y] \backslash j} \prod_{k=1}^{m} f_k(s_k), \tag{2.25}
\]

where \([y] \backslash j\) denotes the vector with all variables except \( y_j \). However, this entity is hard to compute as it involves a sum over all combinations of the values of the random variables in \([y] \backslash j\).

The graphical structure of the distribution allows us to find a tree with the random variable \( Y_j \) at its root. Then, the random variables are partitioned into \( c \) disjoint sets \( T_{f_1}, T_{f_2}, \ldots, T_{f_c} \) where each set corresponds to one branch leading to a factor \( f_1, f_2, \ldots, f_c \) from the root node. Similar to the graphical interpretation (cf. fig. 2.5) this induces a similar grouping of the factor terms
in eq. (2.25). Each branch group contains a factor \( f_l \) at the top of the branch and all factors further down the branch from \( T_{f_l} \),

\[
p(y_j) = \sum_{[y] \setminus j} \prod_{k=1}^m f_k(s_k)
= \sum_{[y] \setminus j} \left( \prod_{l \in N_Y(j)} f_l(s_l) \prod_{k \in T_{f_l}} f_k(s_k) \right)
= \prod_{l \in N_f(j)} \sum_{T_{f_l}} \left( f_l(s_l) \prod_{k \in T_{f_l}} f_k(s_k) \right)
= \prod_{l=1}^c m_{f_l \rightarrow Y_j}(y_j).
\]

In step (\(*\)), we use the disjoint nature of the sets \( T_l \), which follows from the tree structure of the graph. Thus, the sets \( T_l \) partition the domain of the sum.

We call \( m_{f_l \rightarrow Y_j} \) a message from factor \( f_l \) to variable \( Y_j \) with \( l \in \{1, \ldots, c\} \) and it takes as input the state \( y_j \) of the receiving variable. The collection of all messages that arrive at variable \( y_j \) is sufficient to compute the marginal of the random variable \( Y_j \)

\[
p(y_j) = \prod_{l=1}^c m_{f_l \rightarrow Y_j}(y_j).
\]
Figure 2.6: All variables \( T_{f_1} \) that appear in the branch with the factor \( f_1 \) at its top can be split up into the disjoint variable sets \( T_{Y_1}, T_{Y_2}, \ldots, T_{Y_{c'}} \). Every variable \( l' \) sends a message \( m_{Y_{l'}, \rightarrow f_1} \) to the factor \( f_1 \).

The next step simplifies the messages \( m_{f_1 \rightarrow y_j} \) by turning to the remainder of the branch \( f_1 \) where \( T_{f_1} \) contains all the \( c' \)-many associated variables \( Y_1, Y_2, \ldots, Y_{c'} \) (cf. fig. 2.6). Given that a restricted scope \( S_{l'}^j \) contains all random variables of the factor \( f_1 \) without the variable \( Y_j \), then all sums that are not related to factor \( f_1 \) can be pushed further down the tree.

\[
m_{f_1 \rightarrow Y_1}(y_j) = \sum_{t_{f_1}} \left( f_1(s_t) \prod_{k \in M_{f_1}(1)} f_k(s_k) \right)
= \sum_{s_t^j} \left( f_1(y_j, s_t^j) \prod_{l' \in N_{f_1}(1) \setminus \hat{j}} \sum_{t_{Y_{l'}}} \prod_{k \in M_{Y_{l'}}} f_k(s_k) \right)
\]

(2.28)

Hence, the node \( f_1 \) gathers all incoming messages \( m_{Y_{l'}, \rightarrow f_1}(y_{l'}) \) and computes their product. By contrast to the previous variable node in eq. (2.26), the factor node evaluates the sum over all sending random variables; \( Y_1, Y_2, \ldots, Y_{c'} \) in eq. (2.28).

Any of the lower levels of the tree behaves in the same way: variable nodes multiply all incoming messages—ignoring the outgoing direction—and send
the result to the next higher factor node; factor nodes compute the product of the incoming messages and marginalize out all of the sending variables in the sense of eq. (2.28).

Once the lowest level of the tree is reached, a variable node sends a constant 1-function. A factor on the other hand sends its own definition to the parent variable node.

Pushing down the summation of variables as far down in the tree as possible reduces the complexity of the initial marginalization from eq. (2.25), which contains a sum over the entire set of random variables. Although the mathematical recursive derivation seems tedious, the resulting algorithm can be read off the graph rather mechanically. The following equation summarizes the two types of messages from a node or from a factor

\[
m_{y_i \rightarrow f_k}(y_i) = \prod_{l \in N_Y(i) \setminus k} m_{f_l \rightarrow y_i}(y_i),
\]

\[
m_{f_k \rightarrow y_i}(y_i) = \sum_{s_k^i} f_k(y_i, s_k^i) \prod_{l \in N_f(k) \setminus i} m_{y_l \rightarrow f_k}(y_l). \tag{2.29}
\]

Although the sum-product algorithm provides a powerful and generic tool to run inference in graphical models, two restrictions are worth mentioning. First, as presented here it is only possible to run and achieve exact results in tree-structured graphs. For any graph that contains loops, other methods need to be devised to compute marginals. Second, the marginalization operation inside a factor might still be infeasible for factor nodes with a high connectivity. Then the computation involves a sum over all possible combinations of the involved variables.

### 2.3.2 Max-Sum Algorithm

For the zero-one loss in eq. (2.12) the optimal prediction is the state of the random variables that has the highest probability, i.e., the MAP state,

\[
\arg\max_y p(y) = \arg\max_y \prod_{k=1}^m f_k(s_k). \tag{2.30}
\]

The reasoning from the last section 2.3.1 can be similarly applied to the max operator to push the maximization operations as far down as possible in the tree. Commonly, a definition of the algorithm in log-domain avoids multi-
applications of small probabilities. The corresponding inference operations for
variables and nodes then look like
\[
m_{Y_i \to f_k}(y_i) = \sum_{l \in \mathcal{N}(i) \setminus k} m_{l \to Y_i}(y_i),
\]
\[
m_{f_k \to Y_i}(y_i) = \max_{s_k^i} \ln(f_k(y_i, s_k^i)) + \sum_{l \in \mathcal{N}(k) \setminus i} m_{Y_l \to f_k}(y_l). \tag{2.31}
\]

2.3.3 Mean Field Inference

The previous sections examined a class of inference algorithms that rely on
the tree structure of a graphical model. In general, graphs with loops require ap-
proximate inference techniques to solve the inference problem efficiently. Al-
though, there exists an extension of the sum-product algorithm to graphs with
loops called junction-tree algorithm; for graphs with large cliques the computa-
tions become involved. The marginalization inside the connecting factors
on the edge of a clique require the enumeration of all states of all random vari-
ables inside the clique. A number that grows exponentially in the number of
nodes. This section introduces the mean field algorithm, which is a method
that restricts the class of probability distributions to approximate the model
distribution. The allowed distributions are simple enough in their structure
to allow for efficient subsequent reasoning.

The mean field algorithm finds the distribution \( Q \) that is as close as possible
to the model distribution \( P \) in the sense of the Kullback-Leibler divergence.
The divergence between two distributions \( Q \) and \( P \) is defined by
\[
\mathcal{D}(Q \| P) = \sum_y q(y) \ln \frac{q(y)}{p(y)}. \tag{2.32}
\]

The idea of the algorithm is to restrict the distribution \( Q \) to a fully factorized
distribution
\[
q(y) = \prod_{i=0}^{d} q_i(y_i). \tag{2.33}
\]

Thus, the mean field approximation is a structural approximation that solves
a variational problem to simplify the inference. The next paragraphs describe
how the closest distribution \( Q \) can be interactively found given a fixed model
distribution \( P \).
2.3 Structured Inference

The mean field algorithm is the starting point for inference algorithms for graphical models with dense connections, which are presented later in chapter 3, chapter 4, and chapter 5. Unlike simpler models, e.g., a MRF with a 4-connected neighborhood in fig. 2.4, dense models include a pairwise potential between any two variables of the distribution.

Notably, the minimization of the KL-divergence $D(Q \parallel P)$ turns into a functional that is not dependent on the exact normalization constant $Z$ of $P$, in the case of a fixed $P$,

\[
D(Q \parallel P) = \sum_y q(y) \ln \frac{q(y)}{p(y)} = \sum_y q(y) \ln q(y) - \sum_y q(y) \ln p(y)
\]

\[
= -H_{Y \sim Q}(Y) - \sum_y q(y) \ln \left( \frac{1}{Z} \prod_{k=1}^{m} f_k(s_k) \right)
\]

\[
= -H_{Y \sim Q}(Y) - \sum_y q(y) \left( \sum_{k=1}^{m} \ln(f_k(s_k)) - \ln Z \right)
\]

\[
= -H_{Y \sim Q}(Y) - \mathbb{E}_{Y \sim Q} \left[ \sum_{k=1}^{m} \ln(f_k(S_k)) \right] + \ln Z
\]

\[
= -H_{Y \sim Q}(Y) + \mathbb{E}_{Y \sim Q} \left[ \sum_{k=1}^{m} \phi_k(S_k) \right] + \ln Z.
\]

Equation (2.34) uses the definition of the entropy $H_{Y \sim Q}$ of a random variable $Y$ that is distributed according to a multivariate distribution $Q$

\[
H_{Y \sim Q}(Y) := \mathbb{E}_{Y \sim Q} \left[ \ln q(Y) \right].
\]

As a consequence, maximizing the functional

\[
\text{maximize } F[P, Q] := H_{Y \sim Q}(Y) - \mathbb{E}_{Y \sim Q} \left[ \sum_{k=1}^{m} \phi_k(S_k) \right]
\]

subject to \( \sum_{y_i} q_i(y_i) = 1 \quad \forall i \in \{1, \ldots, d\} \)

leads to a minimization of the KL-divergence from eq. (2.32). For a component-wise optimization we consider those terms from eq. (2.36) that depend on the component $q_i(y_i)$ of the distribution

\[
F_i[P, Q] = H_{Y_i \sim Q_i}(Y_i) - \sum_{k=1}^{m} \mathbb{E}_{Y \sim Q} \left[ \phi_k(S_k) \right].
\]
This uses the additivity of information of factorizing distributions:

$$H_{Y \sim Q}(Y) = \sum_{i=0}^{d} H_{Y_i \sim Q_i}(Y_i).$$

(2.38)

The Lagrange functional for $q_i(y_i)$ can express the constraints of the original problem from eq. (2.36) given all other marginal distributions $q_j(y_j)$ for $j \neq i$ by

$$L_i[Q] = H_{Y_i \sim Q_i}(Y_i) - \sum_{k=1}^{m} \mathbb{E}_{Y \sim Q}[\phi_k(S_k)] + \lambda \left( \sum_{y_i} q_i(y_i) - 1 \right).$$

(2.39)

To compute the partial derivative of $L_i[Q]$ with respect to $q_i(y_i)$, first we inspect the derivative of the expectation in eq. (2.39). For a fixed term of the sum, the partial derivative will be 0 if $Y_i$ is not in the scope $S$. For the case where $Y_i$ is in the scope $S$ it makes sense to split up the scope $S$ into $Y_i$ and all other variables from the set $S \setminus i$

$$\frac{\partial}{\partial q_i(y_i)} \mathbb{E}_{Y \sim Q}[\phi(S \setminus i, Y_i)]$$

$$= \frac{\partial}{\partial q_i(y_i)} \sum_{y} \prod_{j=1}^{d} q_j(y_j) \phi(s \setminus i, y_i)$$

$$= \frac{\partial}{\partial q_i(y_i)} \sum_{s \setminus i} \sum_{y_i} \prod_{j:Y_j \in S} q_j(y_j) \phi(s \setminus i, y_i)$$

$$= \sum_{s \setminus i} \prod_{j:Y_j \in S \setminus i} q_j(y_j) \phi(s \setminus i, y_i)$$

$$= \mathbb{E}_{S \setminus i \sim Q} [\phi(S \setminus i, y_i)].$$

(2.40)

Combining the derivative from eq. (2.40) with the derivatives of the other terms in eq. (2.39) leads to the full derivative that vanishes at the local extremum

$$\frac{\partial L_i[Q]}{\partial q_i(y_i)} = - \ln(q_i(y_i)) - 1 + \lambda$$

$$- \sum_{k:Y_i \in S_k} \mathbb{E}_{S_k \setminus i \sim Q}[\phi_k(S_k \setminus i, y_i)]$$

$$= 0.$$
Algorithm 2.1: Mean Field Update

1: \( q_i(l) \leftarrow \text{normalize}(\exp(-\sum_{k : Y_i = S_k} \phi_k(l))), \forall i \)
2: for n iterations do
3: for all variables \( i \) do
4: \( \triangleright \text{Compute all message from factor } k \text{ to variable } i. \)
5: \( m_{k \rightarrow i}(l) \leftarrow \mathbb{E}_{S_k \sim Q}[\phi_k(l, S_k \setminus i)], \forall k : Y_i \in S_k \)
6: \( \triangleright \text{Aggregate all messages and normalize in } q(y_i) \)
7: \( q_i(l) \leftarrow \text{normalize}(\exp(-\sum_{k : Y_i \in S_k} m_{k \rightarrow i}(l))) \)
8: end for
9: end for

Thus, isolating \( \ln(q_i(y_i)) \) on one side leads to

\[
q_i(y_i) \propto \exp\left(-\sum_{k : Y_i \in S_k} \mathbb{E}_{S_k \sim Q}[\phi_k(S_k \setminus i, y_i)]\right). \tag{2.42}
\]

The normalization constant subsumes all terms other than the sum.

Given all marginals \( q_j(y_j), j \neq i, \) of the distribution \( Q, \) the functional from eq. (2.37) is concave in \( q_i(y_i). \) Thus, the stationary condition in eq. (2.42) describes a local maximum. However, the full approximating distribution \( Q \) might not be a local maximum of the full optimization problem from eq. (2.36); even if all its single variable distributions \( Q_i(Y_i) \) fulfill eq. (2.42). The factorizing distribution can be a local minimum, a saddle point or a local maximum.

There is a mean field optimization algorithm 2.1 that iteratively reduces the objective value from eq. (2.36). The update scheme initializes the distribution \( Q \) and updates each of the marginal distributions \( Q_i(Y_i) \) according to eq. (2.42), which decreases the functional at every step. The underlying KL-divergence is bounded by 0 which proves the convergence.

2.3.4 Approximate Inference in Dense Graphs

The previous section 2.3.3 discusses the mean field algorithm, which makes a structural assumption to be able to compute marginal distributions. As shown in eq. (2.42), all factors that include a variable in their scope are combined to produce an updated belief, which seems to rule out models with a high number of factors. However, Krähenbühl and Koltun (2011) observed that the mean field approximation for dense graphs with Gaussian edge po-
potentials can be evaluated efficiently by using acceleration techniques from computer graphics.

Furthermore, the authors argue in their work that it is important for a CRF to recover more of the statistics of natural images between output variables. Therefore, Krähenbühl and Koltun (2011) propose a graphical model that includes a random variable with a label space $\mathcal{L}$ for every pixel and pairwise Gaussian connections between every pair. An illustration of a dense graphical model is depicted in fig. 2.7.

We show later in eq. (2.45), that in order to be able to use fast algorithms to compute the mean field update steps, the pairwise potentials need to be restricted to Gaussian edge potentials. Thus, the energy between two variable $Y_i$ and $Y_j$ is defined by

$$\Phi_{i,j}(y_i, y_j) = \mu(y_i, y_j) \exp\left(-\frac{||z_i - z_j||^2}{2}\right),$$  \hspace{1cm} (2.43)$$

where $z_i$ and $z_j$ denote the features for $Y_i$ and $Y_j$ respectively. Possible features are the pixels’ location in the image. But more general information about the pixels, e.g., their color, is possible, too. Features add the flexibility to the model which variables are more likely to share their label and which do not. In section 2.2.4, the denoising example included a similar argument where a 4-connected neighborhood around a pixel incurs a low energy cost if all involved variables share the same label. To achieve a similar effect for the pairwise...
2.3 Structured Inference

Connection in eq. (2.43) the features $z_i$ and $z_j$ can be set to the pixels’ location in the image. Still, the region of influence is not restricted to the 4-connected neighborhood, but connects all parts of the image. The squared exponential function weights the influence of one pixel onto another pixel by their distance in feature space. The possibility to include color information in the features $z_i$ enables models that encourage label sharing between pixels if they are close in location and color space.

The compatibility function $\mu$ in eq. (2.43) adjusts the influencing strength of one label on another label and is a matrix of size $|\mathcal{L}| \times |\mathcal{L}|$. For instance, its role in semantic segmentation is to fix the different energy costs of neighboring pixels. Probably, it is less likely that a “bird” pixel appears in the direct vicinity of a “water” pixel than in the neighborhood of a “sky” pixel and $\mu$ can adapt to that by causing a higher energy cost.

Plugging the pairwise potential from eq. (2.43) into eq. (2.42) yields

$$q_i(y_i) \propto \exp\left(- \sum_{k:Y_i \in \mathcal{S}_k} \mathbb{E}_{S_k^i \sim Q} [\Phi_k(y_i, S_k^i)]\right)$$

$$= \exp\left(-\Phi_i(y_i) - \sum_{j=1, j \neq i}^d \mathbb{E}_{Y_j \sim Q} [\Phi_{i,j}(y_i, Y_j)]\right)$$

$$= \exp\left(-\Phi_i(y_i) - \sum_{j=1, j \neq i}^d \sum_{y_j} \Phi_{i,j}(y_i, y_j) q_j(y_j)\right).$$

(2.44)

The key observation of Krähenbühl and Koltun (2011) is that all variables share the same label space $\mathcal{L}$ so that the second sum becomes independent of the first sum’s index $j$. After changing the order the two sums we find

$$q_i(x_i) \propto \exp(-\Phi_i(x_i) - \sum_{l \in \mathcal{L}} \sum_{j=1, j \neq i}^d \Phi_{i,j}(x_i, l) q_j(l))$$

$$= \exp(-\Phi_i(x_i) - \sum_{l \in \mathcal{L}} \mu(x_i, l) \sum_{j=1, j \neq i}^d \exp\left(-\frac{\|z_i - z_j\|^2}{2}\right) q_j(l)).$$

(2.45)

Taking a closer look at the last line of eq. (2.45) the computationally expensive part can be identified: all $|\mathcal{L}| \cdot d$ entries in $q$ require $(|\mathcal{L}| \cdot d)^2$ steps, as
Algorithm 2.2: Dense Mean Field Update

1: \( q_i(l) \leftarrow \text{normalize}(\exp(-\phi_i(l))) \)
2: for n iterations do
3:     for all variables \( i \) do
4:         \( m_{s \rightarrow i}(l) \leftarrow \sum_{j=1}^{d} \exp(-0.5 \cdot \|z_i - z_j\|_2^2) \ q_j(l) \)
5:     \( \triangleright \text{Compatibility transform and accumulation of messages.} \)
6:     \( q_i(l) \leftarrow \text{normalize}(\exp(-\phi_i(l) - \sum_{l' \in L} \mu(l, l') m_{s \rightarrow i}(l'))) \)
7: end for
8: end for

every entry in \( q \) needs to take all other entries into account. Krähenbühl and Koltun (2011) observed that the convolutional step

\[
\sum_{j=1, j \neq i}^{d} q_j(l) \exp\left(-\frac{\|z_i - z_j\|_2^2}{2}\right) \tag{2.46}
\]

is well studied in computer graphics. There, bilateral filtering (Aurich and Weule, 1995; Smith and Brady, 1997; Tomasi and Roberto, 1998) implements a blur on images that does not smear color over edges. The precise details of the used algorithm, i.e., the Gaussian permutohedral blur, are the subject of chapter 5. The advantage of this approximation algorithm is its linear runtime in the number of elements in the sum, which reduces the computational burden of line 5 in algorithm 2.2 to a runtime in \( \mathcal{O}(|L| \cdot d) \). There is no hope for an asymptotically faster algorithm, since all \( |L| \cdot d \) variables need to be considered during the computation.

This work extends the idea from Krähenbühl and Koltun (2011) to allow for fields of variables in the case of human pose estimation in chapter 3 and sparse 3-dimensional laser scan data for label transfer in chapter 4. Furthermore, the presented operations can be understood as part of a layered neural network architecture, which is the central topic of chapter 5.

2.4 Learning

We now turn to fitting free parameter in the model from data. Throughout all this work we assume that all model parameters are concatenated in \( w \).

There are at least two approaches to learning structured models (Nowozin and Lampert, 2011). The first group of objective functions is motivated prob-
abilistically and employs estimators that are minimizing a divergence between the sample and the model distribution. All these methods assume that an intermediate probabilistic output of the system is used to drive another decision process in the end. As mentioned in section 2.1.4, this could be done by predicting the state with the lowest expected loss. The second group of objectives is directly optimizing the empirical risk that is tightly coupled to the loss that will be used during test time. The next two sections talk in greater detail about the two approaches.

2.4.1 Probabilistic Parameter Learning

The probabilistically motivated methods fit parameters so that the learned distribution is a close match to the observed statistics of the data distribution. In this section the closeness is defined again in terms of the KL-divergence from eq. (2.32). Minimizing the mean divergence between the true data PMF \( p(y | x) \) and the model’s estimate \( q(y | x, w) \) over the input space \( \mathcal{X} \) for a conditional probability model, e.g., a CRF, is equivalent to maximizing the likelihood of the data under the model distribution (e.g., see Nowozin and Lampert (2011))

\[
\argmin_w \mathbb{E}_{X \sim p} \left[ D(p(\cdot | X) \| Q(\cdot | X, w)) \right] = \argmin_w \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y | x) \ln \frac{p(y | x)}{q(y | x, w)} = \argmax_w \mathbb{E}_{(X,Y) \sim p} \left[ \ln q(Y | X, w) \right] \approx \argmax_w \sum_{i=1}^n \ln q(y_i | x_i, w). \tag{2.47}
\]

This comes with the appealing property that the estimator is converging to the true underlying parameters \( w^* \) in the limit of infinite data. However, this requires that the family of model distribution \( q \) contains the true model.

The occurrence of the normalized distribution \( q \) in the objective value in eq. (2.47) renders the estimator intractable in complex computer vision models. Computing the partition function \( Z \) of \( q \) (cf. eq. (2.16)) in general requires summing over all possible output labels in \( \mathcal{Y} \) that often has an exponential size in \( d \), where \( d \) is the dimension of the prediction vector. Nowozin and Lampert (2011) contains an overview of the proposed alternatives and approximations.
However, this work takes a different approach and is inspired by the work of Domke (2013). In chapters 3 to 5 the mean field algorithm from section 2.3.3 runs the inference and it approximates the model by a fully factorized distribution. The univariate logistic loss is one possible scoring function to fit the marginals as well as possible.

\[
\arg\max_\theta \sum_{j=1}^d \sum_{i=1}^n \ln q([y_i]_j | x_i, w). \tag{2.48}
\]

Rather than fitting all dependencies of the distribution, only the marginals of the model distribution might be of relevance to take decisions for some tasks. When measuring the KL-divergence between the model and true the data marginals (e.g., Domke (2013)) the univariate loss emerges

\[
\begin{align*}
&= \arg\min_{\theta} \mathbb{E}_{X \sim P} \left[ \sum_{j=1}^d D \left( P([\cdot]_j | X) \parallel Q([\cdot]_j | X, \theta) \right) \right] \\
&= \arg\min_w \sum_{x \in X} p(x) \sum_{j=1}^d \sum_{[y_j] \in [y]} p([y]_j | x) \ln \frac{p([y]_j | x)}{q([y]_j | x, w)} \\
&= \arg\max_w \sum_{j=1}^d \mathbb{E} \left( X, [y]_j \sim P \right) \ln q([y]_j | X, w) \\
&\approx \arg\max_w \sum_{j=1}^d \sum_{i=1}^n \ln q([y_i]_j | x_i, w).
\end{align*}
\tag{2.49}
\]

### 2.4.2 Expected Loss Minimization

The previous training method fits an intermediate belief about the output label and at test-time takes a decision based on decision theory (cf. section 2.1.4). A different school of thought uses the test-time loss function to train a classifier that minimizes the empirical risk from eq. (2.7),

\[
\arg\min_h \mathbb{E}_X \left[ r(h) + \frac{1}{N} \sum_{i=1}^n \Delta(h(x_i), y_i) \right]. \tag{2.50}
\]

As mentioned in the related discussion, the minimization of the risk in eq. (2.6) is not possible, because the analytic form of the data distribution is not observed. However, statistical learning theory (Vapnik, 1995) characterizes
hypothesis classes that can be learned efficiently in this framework. A regularizer $r(h)$ is the counterweight to the data information and strives for a simpler hypothesis.

Margin-based classifiers like the Support Vector Machine (SVM) are an example of predictors that are trained with empirical risk minimization. For binary classification, i.e., $y = \{-1, 1\}$, the training objective maximizes the margin of the positive and negative class to the separating hyperplane with

$$\begin{align*}
\text{minimize}_{w, \xi} & \quad \|w\|^2 + \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad y_i w^T x_i \geq 1 - \xi_i, \quad \forall i \in \{1, \ldots, n\}, \\
& \quad \xi_i \geq 0, \quad \forall i \in \{1, \ldots, n\}. 
\end{align*} \tag{2.51}$$

A weight vector $w$ determines the plane by being perpendicular to it. Thus, all positive samples compute a positive dot product with $w$ and lie on one side of the hyperplane; all negative samples result in a negative dot product with $w$ and lie on the other side of the hyperplane. The final scoring function in the sense of eq. (2.2) for a SVM at test time can be described with

$$g(x, y) := y w^T x. \quad \tag{2.52}$$

The formulation in eq. (2.51) describes a soft-margin, where training samples may be misclassified by the trained predictor if the regularizer out-weights the data information or in cases with a restricted hypothesis class.

The optimization problem in eq. (2.51) can be rewritten in its unconstrained for with

$$\begin{align*}
\text{minimize}_{w, \xi} & \quad \|w\|^2 + \sum_{i=1}^{n} \ell_{\text{hinge}}(y_i, w^T x_i) \\
\ell_{\text{hinge}}(y, t) := & \max(0, 1 - yt).
\end{align*} \tag{2.53}$$

The zero-one loss function $\Delta_{0/1}$ from eq. (2.4), that is usually associated with a binary classification problem, cannot be used with eq. (2.51) when gradient based methods are employed for minimization. The constant values of the loss function lead to gradients that vanish at all locations of the objective function. A proxy loss function, which acts as a substitute and upper bounds $\Delta_{0/1}$, allows to successfully train the parameters. The Hinge loss $\ell_{\text{hinge}}$ is an example of an employed proxy loss.
An extension of the SVM objective from eq. (2.51) to structured problems was proposed by Taskar, Guestrin, and Daphne Koller (2003) and Tschantzaridis et al. (2005):

\[
\begin{align*}
\text{minimize}_{w, \xi} & \quad \|w\|^2_2 + \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad g(x_i, y_i, w) - g(x_i, \hat{y}_i, w) \geq \Delta(\hat{y}_i, y_i) - \xi_i, \quad (2.54) \\
& \quad \forall i \in \{1, \ldots, n\}, \forall \hat{y}_i \in \mathcal{Y}, \\
& \quad \xi_i \geq 0, \quad \forall i \in \{1, \ldots, n\}.
\end{align*}
\]

The structured SVM trains a scoring function \(g\) that maximizes a margin of at least \(\Delta\) between the score of a ground truth annotation \(y_i\) for a training sample \(x_i\) and every other possible label \(\hat{y}_i\) from \(\mathcal{Y}\). The margin, which is set to 1 for the SVM, is rescaled by the loss \(\Delta\) that a solution \(\hat{y}\) would incur instead of the annotation \(y\). Thus, two solutions that are “close” with respect to this loss may be similarly scored by \(g\). The loss function relates the different elements from \(\mathcal{Y}\) through this training process and usually simplifies the learning problem by using less training data.

The optimization problem in eq. (2.54) can be transformed into an unconstrained problem

\[
\begin{align*}
\text{minimize}_{w, \xi} & \quad \|w\|^2_2 + \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad \max_\hat{y} (g(x_i, \hat{y}, w) + \Delta(\hat{y}, y_i)) - g(x_i, y_i, w) \leq \xi_i, \quad (2.55) \\
& \quad \forall i \in \{1, \ldots, n\}, \\
& \quad \xi_i \geq 0, \quad \forall i \in \{1, \ldots, n\}.
\end{align*}
\]

by changing the constraints that hold for all output labels to a single constraint that only considers the maximally offending solution, i.e.,

\[
\text{argmax}_\hat{y} g(x_i, \hat{y}, w) + \Delta(\hat{y}, y_i). \quad (2.56)
\]

Instead of enforcing a margin for all possible labels, the constraint finds the label that leads to the largest margin violation. The maximization of this constraint resembles the original inference problem in eq. (2.2) that is run during test time

\[
\text{argmax}_\hat{y} g(x_i, \hat{y}, w). \quad (2.57)
\]
Hence, if an efficient inference algorithm exist that can find the answer to both eq. (2.57) and eq. (2.56), we can solve the optimization problem in eq. (2.55). At every step of the optimization algorithm—when the objective value and its gradients are computed—we need to solve a loss-augmented inference problem.

The first constraint of the optimization problem in eq. (2.55) enforces a direct inequality for the slack variables $\xi_i$ and therefore can be plugged into the objective function. Taking the positivity constraint for $\xi_i$ and the monotonicity of $\ell$ into account, we find that

$$\min_{\mathbf{w}} \|\mathbf{w}\|_2^2 + \sum_{i=1}^n \max(0, \delta_i)$$

where $\delta_i := \max_{\hat{y}} (g(x_i, \hat{y}, \mathbf{w}) + \Delta(\hat{y}, y_i)) - g(x_i, y_i, \mathbf{w}).$ (2.58)

This optimization objective appears again in chapter 3 to learn the parameters for an inference algorithm for human pose estimation.
HUMAN POSE ESTIMATION WITH FIELDS OF PARTS

3.1 INTRODUCTION

In this work we consider the challenging problem of human pose estimation from a single image. This task serves as a crucial prerequisite step to high level vision applications, for example human action recognition (Jhuang et al., 2013), and natural human computer interfaces (J. Shotton et al., 2011). Therefore, it is among the most studied problems in the field of computer vision.

The main difficulty of pose estimation is the weak local appearance evidence for every single body part. While heads nowadays can reliably be detected, localization of general body parts such as arms, legs, or hands remain challenging. The following factors complicate detection: fore-shortening and self-occlusion of parts; different clothing and light environments lead to variability in appearance; some parts might cover only a small area which makes it hard to encode them robustly.

Consequently, the predominant method for this problem are approaches that model both appearance and part configuration jointly. This idea of combining part appearance evidence with spatial configuration for part relations dates back to Fischler and Elschlager (1973) and was popularized as a CRF model by Felzenszwalb and Huttenlocher (2005). The CRF approach of Felzenszwalb and Huttenlocher elegantly expresses pose estimation in a statistical structured prediction problem and introduces with the distance transform an efficient exact inference technique. This model serves as a basis for variants and thus resulted in significant empirical improvements on increasingly challenging datasets (Ramanan, 2006; Ferrari, Marin, and Zisserman, 2008; Johnson and Everingham, 2010).

Most work focuses on the main dimensions of the pose estimation problem: use of discriminative appearance information (Sapp, Jordan, and Taskar, 2010; Pishchulin et al., 2013a; Pishchulin et al., 2013b; Yang and Ramanan, 2011; Yang and Ramanan, 2013; Eichner and Ferrari, 2012; Eichner and Ferrari, 2009) and stronger models for the spatial body configuration (Sapp, Weiss, and Taskar, 2011; M. Sun et al., 2012; Pishchulin et al., 2013a).
Examples of better appearance models are the local image conditioned features used in Sapp, Jordan, and Taskar (2010), the use of mid-level representations via Poselets (Gkioxari et al., 2013; Bourdev et al., 2010; Pishchulin et al., 2013a), or semantic segmentation information to include background evidence (Eichner and Ferrari, 2012; Vineet, Sheasby, et al., 2013; Ladicky, Torr, and Zisserman, 2013; Bray, Kohli, and Torr, 2006).

The spatial graphical model of Felzenszwalb and Huttenlocher (2005) is a tree, a limitation that obviously does not reflect probabilistic dependencies in the human body, e.g., color relation between left and right limbs. M. Sun et al. (2012) address this by introducing loops into the graph and Dantone et al. (2013) and A. Jain, Tompson, Andriluka, Taylor, and Christoph Bregler (2013) regress onto part positions directly.

Another dimension of interest is inference efficiency: while performing well, richer appearance features typically require more computations and therefore are slow. The same is true for the probabilistic graph and giving up the tree structure usually results in more involved inference techniques. Sapp, Toshev, and Taskar (2010) and Sapp, Jordan, and Taskar (2010) propose cascading models or a coarse-to-fine search for pose estimation to speed up inference and enable the use of richer appearance features.

This work proposes the Fields of Parts (FoP) model; a re-formulation of the human pose estimation problem. The FoP model offers a different view on all three dimensions—appearance, structure, and inference. It is inspired by the PS model, but has different semantics which lead to interesting modeling possibilities. The main idea of this model is simple: a binary random variable models the presence or absence of a body part at every possible location, orientation, and scale of a body part. This results in a large number of variables, seemingly complicating the matter.

This work shows that this model is tractable and presents a way to perform efficient marginal inference and more importantly, that this re-parametrization offers new and interesting modeling possibilities. In particular, it allows to carry over ideas from semantic segmentation. We achieve this without the need to explicitly include a segmentation layer or rely on a pose estimation pipeline as a preprocessing step in order to generate body part proposals. The FoP model provides a full interpretation of the image: the presence of a body is explained at every position simultaneously while by an argument of symmetry including evidence from the background without the need for explicit segmentation variables. Also, the graph topography is flexible. We are not bound to a tree structure with restricted potentials in order to use the
The marginal inference technique that this work proposes, namely mean field, is approximate. However, we reason that this is not a severe limitation. We account for the approximation already during training time using back-mean-field learning (Domke, 2011; Domke, 2013).

The inference complexity depends only linearly on any important dimension of the model: number of part-connections, number of feature dimensions, and size of the image. Furthermore it is amendable to parallelization.

The FoP model builds upon advances from three separate domains: efficient inference for segmentation (Krähenbühl and Koltun, 2011), parameter estimation with approximate inference (Domke, 2011; Domke, 2013), and expressive
PS models (Yang and Ramanan, 2013). We report on modeling, technical, and experimental contributions:

- A reformulation of the human pose estimation problem. This opens up new modeling flexibility and provides a new viewpoint on this well-studied problem (model in section 3.3.1, discussion in section 3.3.2).

- An generalization of the inference from Krähenbühl and Koltun (2011) as described in section 2.3.4. This makes it possible to use efficient mean field inference in the FoP formulation (section 3.4.1).

- A new estimator tailored to pose prediction using a binary CRF formulation. (section 3.4.2).

- Experimentally, we demonstrate that the FoP model with the same set of parameters as Yang and Ramanan (2013) achieves a performance increase of 6.0% on the Leeds Sport Poses (LSP) dataset (Johnson and Everingham, 2010), novel variants improve this even further (section 3.5).

3.2 RELATED WORK

We adapt the part based formulation from Yang and Ramanan (2013) since it offers a good trade-off between flexibility and efficiency. The authors propose to model a body as a collection of body joints, with each body joint being represented as a point in the 2-dimensional plane for its position, and a multinomial type variable that accounts for appearance variations. For the FoP we enumerate all states and model each one with a binary random variable. A different way to model body part appearance is by a representation as boxes with a center, orientation and scale, e. g., Andriluka, Roth, and Schiele (2009). The model in Pishchulin et al. (2013b) combines both the body part and body joint representations into a single joint model. The authors report improved performance, however their proposed method has a runtime of minutes per image.

Other approaches introduce more connections in the factor graph to account for the dependencies of body parts not reflected in a tree structure. One such example is M. Sun et al. (2012) that combines a densely connected model with efficient branch and bound inference.

PS models can be understood as body pose detectors that only model the foreground object while handling background information indirectly. Ferrari,
Marin, and Zisserman (2008) and Sapp, Jordan, and Taskar (2010) use segmentation information within their pose estimation model; this typically complicates the inference process. Therefore, these methods either use sequential algorithms (Ferrari, Marin, and Zisserman, 2008) or CRF inference methods with elaborate search based methods (Sapp, Toshev, and Taskar, 2010).

Another way to include background evidence is to explicitly include a separate segmentation layer (Ladicky, Torr, and Zisserman, 2013; Bray, Kohli, and Torr, 2006; Vineet, Sheasby, et al., 2013; H. Wang and Koller, 2011). Most of these works following this choice have in common that they rely on a separate pose estimation algorithm, e. g., Andriluka, Roth, and Schiele (2009) and Yang and Ramanan (2011), to retrieve a number of candidate poses. Based on these proposals a CRF structure is then instantiated with factors for segmentation and selector variables for the proposals. Additional proposed CRF layers include foreground/background segmentation (H. Wang and Koller, 2011), body part segmentation (Ladicky, Torr, and Zisserman, 2013) or a combination with stereo estimation (Vineet, Sheasby, et al., 2013). Finally, Eichner and Ferrari (2012) exploit commonalities in the background appearance within a dataset by fitting a separate color likelihood term to an estimate of the background area.

Recently, there has been a shift towards models based on CNNs and deeper architectures. They share the switch to learned features rather than relying on hand-crafted inputs, i. e., Histogram of Oriented Gradients (HOG)-features that are used in this work. This leads to an improved performance compared to the model from this work, but requires a larger training set to fit the increased number of parameters. Toshev and Szegedy (2014) formulate the pose prediction problem as a regression and directly predict the location of the joints as coordinates in the image. However, the majority of recent work also follows the parametrization advocated in this thesis and build on top of a per-pixel/location output of their unary classifiers. This is in contrast to the PS model that introduces a single random variable per part. Apart from the notable exception of decision trees in Ramakrishna et al. (2014), top performing methods use exclusively CNNs at least as their unary terms. These initial beliefs are either directly combined with more contextual information as in Fan et al. (2015) or serve as an input to a supplementary CRF reasoning that is either fixed (A. Jain, Tompson, Andriluka, Taylor, and Christopher Bregler, 2014) or learned, e. g., in X. Chen and Yuille (2014). The models from Tompson, A. Jain, et al. (2014), Tompson, Goroshin, et al. (2015), and Wei et al. (2016)
all can be understood as unrolled message passing that is also employed in the FoP model that is subject of section 3.3.1.

For inference and learning this work builds upon the advances from Krähenbühl and Koltun (2011) that it generalizes. The authors show that mean field inference in densely connected models with Gaussian pairwise potentials reduces to an application of bilateral filtering as seen in section 2.3.4. The other connection that we draw is to marginal based learning techniques advocated in Domke (2011) and Domke (2013). He argues that learning should both take the desired loss function as well as the approximate nature of the inference procedure into account. Our model implements this by using back-mean-field learning, also mentioned in Krähenbühl and Koltun (2013).

3.3 Fields of Parts

The flexible body part model of Yang and Ramanan (2013) serves as the starting point for our derivation. Yang and Ramanan (2013) propose to model each body part \( p \) as a random variable \( \tilde{Y}_p \) with three vector elements for the 2-dimensional position in the image \( I \) and a latent type variable \( T \in \{1, \ldots, K\} \). The idea of introducing \( T \) is to capture appearance differences of a part due to fore-shortening, rotation, etc., while at the same time increasing the flexibility of the body configuration. We gather all possible states of \( \tilde{Y}_p \) in a set \( \mathcal{L}_p \); the entire body is then represented as the concatenation \( \tilde{Y} = (\tilde{Y}^1, \ldots, \tilde{Y}^p) \).

The PS model defines a Gibbs distribution \( P(\tilde{Y} | I, w) \) according to eq. (2.16), where \( w \) denotes the collection of all model parameters.

This work proposes a different kind of parametrization. Section 3.3.1 introduces the model; the gained flexibility that it offers is the topic of section 3.3.2. The technical contributions on inference in section 3.4.1 and learning in section 3.4.2 that enable the use of this parametrization are the topic of section 3.4.

3.3.1 Model

We parametrize the problem in the following way: for every part \( p \) and every possible state in \( \mathcal{L}_p \) we introduce a binary random variable \( Y^p_i, i = 1, \ldots, |\mathcal{L}_p| \). Each such variable represents the presence \( Y^p_i = 1 \) and absence \( Y^p_i = 0 \) of a part at its location, type, and scale in the image. We refer to the collection of variables for a part \( Y^p = \{Y^p_i\}_{i=1, \ldots, |\mathcal{L}_p|} \) as a field. With \( Y \) we denote the collection of all variables for all parts. The total number of variables per part
p is $|\mathcal{L}^p|$, the total number for all parts $S = \sum_p |\mathcal{L}^p|$ and thus the state space of $Y$ is of size $2^S$.

We introduce a set of variables $Y$ on different scales of the image. This ensures that the model can cope with different zoom levels. The inference step treats all levels independently without any connections between the variables. However, we do not use a super- or subscript to denote the larger state space not to clutter the notation.

Next, we discuss how to connect the variables in a meaningful way.

### 3.3.1.1 Energy

Given an image $I$ and model parameters $w$, we write the energy of a Gibbs distribution $P(Y | I, w)$ as the sum of unary and pairwise terms as described in section 2.2.3

$$E(y | I, w) = \sum_{p=1}^{P} \sum_{i=1}^{|\mathcal{L}^p|} \phi_i^p(y_i^p | I, w)$$

$$+ \sum_{p \sim p'} \sum_{i=1}^{|\mathcal{L}^p|} \sum_{j=1}^{|\mathcal{L}^{p'}|} \phi_{i,j}^{p,p'}(y_i^p, y_j^{p'} | I, w). \quad (3.1)$$

Note, that the neighborhood relation is defined between different fields $p \sim p'$, e.g., wrist and elbow. For any two fields in this relation, a separate factor node connects each pair of random variables $(Y_i^p, Y_j^{p'})$. We illustrate the resulting cyclic CRF graph in fig. 3.1 for the case of kinematic chain connections in $p \sim p'$ and six body parts.

### 3.3.1.2 Unary Factors

Local appearance of body parts is captured through the unary factors $\phi_i^p$ for state $i$ and part $p$. In the simplest case this might be a log-linear model from eq. (2.20)

$$\phi_i^p(y_i^p | I, w) = \langle w_i^{p, \text{unary}}, \Phi_i(I) \rangle. \quad (3.2)$$

Concretely, we use exactly the same factors as Yang and Ramanan (2013) in order to make the models comparable: HOG features in $\Phi(I)$ (Dalal and Triggs, 2005) and a linear filter $w_i^{p, \text{unary}}$ of size $5 \times 5$ at different scales of the image. However, any other unary classifier—in particular a deep architecture—can be used in-lieu of the weighted HOG descriptors.
3.3.1.3 Pairwise Factors

The important piece of the FoP model are the pairwise connections. Their form needs to fulfill two requirements: encode a meaningful spatial configuration between neighboring fields, and allow for efficient approximate inference. We are inspired by the observation of Krähenbühl and Koltun (2011). In their work they show that mean field inference in densely connected models with Gaussian pairwise potentials can be implemented as a bilateral filtering (section 2.3.4). Since highly optimized algorithms exist (Adams, Baek, and Davis, 2010) for this operation, the approximate inference is efficient. The pairwise terms in the FoP model have the following form

$$\Phi_{i,j}^{p,p'}(y_i^p, y_j^{p'} | I, w) = - \sum_m \mu_m(y_i^p, y_j^{p'})$$

$$k_m^{p,p'}(z_m(i, p; I, w), z_m(j, p'; I, w); w),$$

$$k_m^{p,p'}(z, z'; w) = \exp\left(-\frac{1}{2}(z - z' - z_m^{p,p'})^T (\Sigma_m^{p,p'})^{-1}(z - z' - z_m^{p,p'}))\right).$$

The key observation is that eq. (3.3) allows to encode the same spatial relation between body part variables $Y_i^p$ and $Y_j^{p'}$, as the PS model does for $\tilde{Y}_i^p$ and $\tilde{Y}_j^{p'}$. This potential is a linear combination of Gaussian kernels $k_m$ weighted by a compatibility matrix $\mu$ of size $2 \times 2$. Over the course of this document we introduce three different types of these kernels.

To encode the same spatial relationship as PS models with a kernel $k_{\text{PS}}^{p,p'}$ we use the 2-dimensional positions of a state $i$ as feature $z_{\text{PS}}(i, p; I)$. Consider two variables $Y_i^p$ and $Y_j^{p'}$, and their 2-dimensional image positions: the two states with maximal influence on each other regarding the kernel function are those whose 2-dimensional position are offset by exactly $z_m^{p,p'}$. In this case the value of the exponential kernel $k_{\text{PS}}^{p,p'}$ in eq. (3.4) reaches its maximum as the difference of its argument vanishes. Naturally, the value decreases exponentially depending on the distance and the variance $\Sigma_{\text{PS}}^{p,p'}$. The symmetric compatibility function $\mu_{\text{PS}}$ weights the kernel evaluation based on the labels of $Y_i^p$ and $Y_j^{p'}$. Usually, $\mu_{\text{PS}}$ is a positive definite matrix and its purpose is to encourage the variable label pair $Y_i^p$ and $Y_j^{p'}$ to agree. A disagreeing label set would incur a higher energy cost.

Note that a state $i$ also includes the type/mixture component $T$. For every
part there are as many random variables at the same 2-dimensional location as there are mixture components $K$ in the model. For every type/type pair we use a different offset and variance. Again, to enable comparison we implement the choice made in Yang and Ramanan (2013), where the offset only depends on one of the two types. In Yang and Ramanan (2013) the child type determines the offset and variance. In summary the FoP model is able to represent the same kind of flexible body part configurations. A minor difference is that here, we use Gaussian potentials, whereas in the PS model the spatial term is log-linear (cf. eq. (2.20)).

3.3.2 Discussion

The parametrization of the FoP model allows to carry over ideas from semantic segmentation into the pose estimation problem.

In the previous section the Gaussian pairwise terms only use positional information. In fact any features $z(i, p; I) \in \mathbb{R}^{k}$ can model the influence of two states on each other. For example, we can use color by appending RGB values to the image locations, resulting in a bilateral kernel. This is in contrast to PS models (Felzenszwalb and Huttenlocher, 2005; Yang and Ramanan, 2013; Andriluka, Roth, and Schiele, 2009) where extra local image evidence can not easily be included. The reason is inference time; in order to use the distance transform, the features have to lie on a grid, and for example RGB values do not necessarily follow this constraint. Without this restricted form of the features, the general sum product algorithm scales quadratically in the number of states.

We exploit this new possibility in three different ways: including color information, using foreground/background segmentation of a person and connecting the CRF more densely.

3.3.2.1 FoP-Bilateral

Additionally to the between-fields connections $p \sim p'$ for $p \neq p'$, we also connect the variables within a single field $p$ using an additional pairwise term in eq. (3.3) for $\phi_{i,j}^{p,p}(y_{i}^{p}, y_{j}^{p} | I, w)$:

$$
\mu_{\text{bilateral}}(y_{i}^{p}, y_{j}^{p}) k_{p}^{\text{bilateral}}(z_{\text{bilateral}}(i, p; I), z_{\text{bilateral}}(j, p; I); w).
$$

(3.5)
We set

\[ \mathbf{\mu}_{\text{bilateral}} := \begin{pmatrix} w_{\text{bilateral}} & 0 \\ 0 & 0 \end{pmatrix} \]  \tag{3.6} 

and use the 2-dimensional position and RGB color features in a 3 × 3 neighborhood around the position of \( i \) for \( z_{\text{bilateral}}(i, p; 1) \). Thus, the kernel weight is highest for variables \( Y_i^p \) and \( Y_j^p \) that are near each other in image-space \textit{and} similar in color-space.

The term in eq. (3.5) prefers if variables that share their surrounding color and are reasonably close to also share the background label. This is interesting for images where color provides a strong cue for background, e.g., with large portions of the picture covered by sky or a football player standing on a patch of grass. The term connects all variables in the image but weights the influence exponentially with the distance of the variables. The color that enables the term does not need to be known before test-time and is not part of the unary terms of the model. Thus, cross-talking between variables in a part field can happen if the dominant color of the background changes from image to image. A visualization of the influence of a state variable on the rest of its field can be seen in fig. 3.2. The compatibility function \( \mathbf{\mu}_{\text{bilateral}} \) in eq. (3.5) only removes cost from the overall configuration energy for the background class; a variable with strong evidence for a foreground element does not lead to variables to agree on the foreground label in the same part field. We expect that this would harm, e.g., nearby parts of the upper body that often share the same clothing and color. In effect this propagates background information in the image over the random variables and it is the same type of a bilateral kernel as used in segmentation methods (Krähenbühl and Koltun, 2011; Vineet, Sheasby, et al., 2013). Here, it aids prediction of body parts without explicitly reasoning about segmentation.

In our experiments we set the covariance matrix of the kernel in eq. (3.5) with its parameters from eq. (3.4) to a diagonal matrix

\[ \Sigma_{\text{bilateral}}^p = \begin{pmatrix} 25 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0.01 \end{pmatrix} \]  \tag{3.7} 

and the offsets \( z_{\text{bilateral}}^p \) to 0.
3.3 Fields of Parts

Figure 3.2: For an input image (a) from the test set of the LSP dataset (Johnson and Everingham, 2010), (b) shows the values of the bilateral kernel $k_{bilateral}^P(z, \cdot)$ for a state variable $z$ marked with a cross from the upper right region in the image. The kernel weights all connection between every pair of variables and only propagates the background label 0. It relies on the distance in spatial domain and in color domain. Thus, the influence of the selected variable from a grass patch does not bleed into the body variables of the person.

3.3.2.2 FoP-Segmentation

As a second modification to the model in eq. (3.3) we include segmentation predictions as extra image evidence into the set of pairwise terms. The decision tree implementation of Nowozin, Rother, et al. (2011) and its features are used to train a person/background classifier on the training images. From ground truth bounding box annotations we construct 0/1 segmentation masks for training (cf. fig. 3.3). The final decision tree yields a score in $d_{u,v} \in [0,1]$ for every position $(u,v)$ in the image, namely, the fraction of person-pixels in the corresponding leaf. We then append this score to the spatial features to all states at their positions. This results in a bilateral kernel and allows for propagation of information that is different inside or outside of the predicted segmentation.

We define the additional kernel for eq. (3.3) by

$$\mu_{seg}(y^P_i, y^P_j) \ k_{seg}^P(z_{seg}(i, p; I), z_{seg}(j, p; I); w),$$

(3.8)
Figure 3.3: We generate a coarse bounding box annotation around sticks from the pose annotations for a training image (a). The trained unary decision tree model from Nowozin, Rother, et al. (2011) outputs a distribution for every pixel in the test image (b). Bright green is used if all samples in a leaf of the tree agree on the foreground label, whereas bright blue depicts full certainty of the background label. The smaller insets show the respective input images.

with

\[ \mu_{\text{seg}} := \begin{pmatrix} w_{\text{seg}} & 0 \\ 0 & w'_{\text{seg}} \end{pmatrix}. \tag{3.9} \]

The offsets \( \mu_{\text{seg}}^{p,p'} \) are equal to the offsets from the base kernel \( \mu_{P_{\text{seg}}^{p,p'}} \) appended with a zero offset for the segmentation feature. We set the covariance to

\[ \Sigma_{\text{seg}}^{p,p} = \begin{pmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}. \tag{3.10} \]

Figure 3.4 shows qualitatively how state variables are influencing neighboring fields in this setting.

3.3.2.3 FoP-Loopy

The CRF of the FoP model is a loopy graph already. The upcoming section 3.4 shows that the inference complexity depends only linearly on the number of
Figure 3.4: A test image (a) with its segmentation output from the decision tree. Two more visualizations show the kernel strength of the segmentation term from eq. (3.8) on the head field with respect to the neck field. Colors that tend more towards red depict larger values. We select two examples (b) inside and (c) outside the main segmentation area of the body to demonstrate that information hardly crosses segmentation boundaries.

field-field connections in $p \sim p'$. This allows us to extend the relation $\sim$, which is inspired by a kinematic chain, and connect the fields more densely with only a modest increase in computational complexity. In this variant ($FoP$-$Loopy$) we introduce 10 more connections between parts that contain spatial information about each other, e.g., left and right hip. Figure 3.5 shows the changed model topology. Back-fitting them to the regular $PS$ model from Yang and Ramanan (2013) would destroy the tree structure that is a prerequisite for their inference algorithm.

3.3.3 Comparison to Pictorial Structures

There are two main differences between the $FoP$ model and the $PS$ model concerning the semantic of their outputs. $PS$ models explain the foreground; they represent a conditional distribution $P(\tilde{Y} | I, w)$ over all possible body configurations. In contrast the $FoP$ model explains the entire image $P(Y | I, w)$, i.e., foreground and background at every location. Hence, the $FoP$ model is not just a relaxation of the $PS$ model in the sense that we allow multiple detections for one part. Due to an argument of label symmetry, the model contains non-
trivial background-background terms that do not exist for the PS model. This is in spirit of works that combine segmentation information into the pose estimation problem (Bray, Kohli, and Torr, 2006; Eichner and Ferrari, 2012; Vineet, Sheasby, et al., 2013; Ladicky, Torr, and Zisserman, 2013) but with the crucial difference that the FoP model is designed for pose estimation. It does not require a separate algorithm to generate part proposals and the model does not contain an explicit segmentation layer.

Second, consider the case of multiple persons or no persons in an image. What would the optimal distribution be? With no person in the image the best a PS model can do is to achieve a uniform distribution over the body poses, because it has no notion of absent body parts. In the case of multiple persons the distribution becomes multi-modal. Consequently, the probability mass has to be distributed over different persons and thus the scores will have to decrease. A similar effect will happen if the image size is increased. This can be undesirable depending on the application, the score/probability
of a body pose should not depend on the number of people in the image or its size. Therefore a detection step is a crucial prerequisite for the PS model. The FoP model solves this by explaining that state of every image location separately. This allows it to have multiple or no locations with detections and to become independent of the image size. However, the FoP model does not add an interpretation which part detection actually belongs to a single body.

### 3.4 Learning and Inference

In this section we present the technical extension of Krähenbühl and Koltun (2011) and develop novel ideas based on section 2.3.4 that enable efficient inference (section 3.4.1). We then present an estimator tailored to the pose prediction problem with this binary CRF (section 3.4.2).

#### 3.4.1 Inference

Exact inference in the FoP model is unfortunately prohibitive due to the loopy structure of the factor graph. We resort to approximate inference, and in particular to a mean field approximation. As described in section 2.3.3, with mean field the intractable distribution is replaced by a factorizing approximation $Q$, usually by the product of its marginal PMFs $q(x \mid I, w) = \prod_i q_i(x_i \mid I, w)$, that are then fit to yield a low KL divergence with the target distribution. Every binary state variable $X^p_i$ gets its approximating probability distribution
Note that by finding the factorizing distribution $Q$ we gain all included state marginals of $X^p_i$. Thus, questions that are hard to answer with respect to the original probability distribution become easy with the approximation.

Krähenbühl and Koltun (2011) have shown that the mean field update equations in discrete CRF models with Gaussian pairwise potentials can be implemented by means of bilateral filtering. Similar to the discussion in section 2.3.4 the potentials from eq. (3.1) plugged into eq. (2.42) lead to

$$q_i^p(y_i^p \mid I, w) \propto \exp(-\phi_i^p(y_i^p \mid I, w))$$

$$- \sum_{p \sim p'} \sum_{j=1}^{\left|\mathcal{L}^p\right|} \mathbb{E}_{y_j^{p'} \sim Q_j^{p'}} \left[ \phi_i^{p',p}(y_i^p, y_j^{p'} \mid I, w) \right]$$

$$= \exp(-\phi_i^p(y_i^p \mid I, w))$$

$$- \sum_{p \sim p'} \sum_{j=1}^{\left|\mathcal{L}^p\right|} \sum_{y_j^{p'}} \phi_i^{p,p'}(y_i^p, y_j^{p'} \mid I, w) q_j^{p'}(y_j^{p'}) \quad (3.11)$$

Once again the key to acceleration is the observation that all variables share the same label space, i.e., $\{0, 1\}$ and the innermost sum can be pulled out

$$q_i^p(y_i^p \mid I, w) \propto \exp(-\phi_i^p(y_i^p \mid I, w))$$

$$- \sum_{p \sim p'} \sum_{l'} \sum_{j=1}^{\left|\mathcal{L}^p\right|} \phi_i^{p,p'}(y_i^p, l' \mid I, w) q_j^{p'}(l') \quad (3.12)$$

Thus, in the FoP model the mean field update equations can be derived to

$$q(y_i^p \mid I, w) \propto \exp(-\phi_i^p(y_i^p \mid I, w)) - \sum_{p \sim p'} \sum_{l'} \sum_{m} \mu_m(y_i^p, l')$$

$$\sum_{j=1}^{\left|\mathcal{L}^p\right|} k_{m}^{p,p'}(z_m(i, p; I, w), z_m(j, p'; I, w); w) q_j^{p'}(l' \mid I, w) \quad (3.13)$$

This generalizes the results of Krähenbühl and Koltun (2011) where there is no part connection relationship $p \sim p'$. In the update step eq. (3.13) we can exploit the underlying structure of the factor graph to perform bilateral filtering of the two affected neighboring fields. There are two filtering operations—from $p$ to $p'$ and back—for every field connection $p \sim p'$. The full update algorithm is described in algorithm 3.1.
Algorithm 3.1: Mean Field Update in the Fields of Parts Model

\[ q_i^p(l) \leftarrow \text{normalize}(-\phi_i^p(l)) \]

\[ \text{for } n \text{ iterations do} \]

\[ \triangleright \text{Initialize all messages.} \]

\[ \tilde{q}_i^p(l) \leftarrow 0, \quad \forall p, i \]

\[ \text{for } p, p' \sim p \text{ do} \]

\[ \triangleright \text{Message passing from part } p' \text{ to } p. \]

\[ m_{s \to i, p}(l) \leftarrow \sum_{j=1}^{\mathcal{L} \setminus p'} k_{m}^{p, p'}(z_m(i, p), z_m(j, p')) q_j^{p'}(l), \quad \forall i \]

\[ \triangleright \text{Compatibility transform and accumulation of messages.} \]

\[ \tilde{q}_i^p(l) \leftarrow \tilde{q}_i^p(l) + \sum_{l'} \sum_{m} \mu_m(l, l') m_{s \to i, p}(l') \quad \forall i \]

\[ \quad \text{end for} \]

\[ q_i^p(l) \leftarrow \text{normalize}(\exp(-\phi_i^p(l) - \tilde{q}_i^p(l))), \quad \forall p, i \]

\[ \text{end for} \]

As noted by Krähenbühl and Koltun (2011) this block update scheme is not guaranteed to converge. In practice we have not seen any convergence problems for our model.

To come by the expensive operation of calculating the message from one part field \( p \) to another part field \( p' \), we also make use of an acceleration technique of the permutohedral lattice (Adams, Baek, and Davis, 2010). This reduces the computational cost to be linear in the number of states of the two involved fields in contrast to the quadratic cost in the number of states in a naive implementation. We loosen the probabilistic interpretation of the mean field update and allow the compatibility matrix \( \mu^{p, p'} \) to differ for the messages passed from \( p \) to \( p' \) and vice versa.

For images that contain a single person only we report, for each field separately, the state that is most probable to be of value 1,

\[ \hat{i}^p = \arg\max_{i \in \mathcal{L}^p} q_i^p(1 | I, \mathbf{w}). \quad \tag{3.14} \]

Nevertheless, there is no reason not to use a different prediction rule, e.g., in the case of multiple persons in one image. The complexity of the inference algorithm scales favorably, namely linear in every dimension: number of mean field iterations, number of Gaussian kernels \( m \), linear in the dimension of pairwise features \( k \), linear in the number of part-part connections \( p \sim p' \). Furthermore the model is amendable to easy parallelization, e.g., by calculating the messages sent by the part fields in parallel. In our current CPU implementa-
tion the model requires about 6s for inference on a single level in an image of size 100 × 200.

3.4.2 Parameter Estimation

Part annotations are available as 2-dimensional positions \((u, v)\) of the separate body parts which we translate into the binary CRF formulation. Using \(K\) types for part \(p\), the FoP model contains \(K\) random variables that represent the position \((u, v)\), one for each type. It is desirable to find parameters \(w\) that yield a high probability for at least one of those variables being in state 1. Here we construct a max-margin objective in eq. (2.54) that is tailored to pose estimation: the predicted state \(\hat{\mathbf{p}}\) in eq. (3.14) should be at the correct image position. There is no loss for background states in pose estimation, and thus they are not included in the objective.

3.4.2.1 Prediction Loss

We measure the performance of body pose models using loss functions that ideally represent the desired output of the systems. For the parametrization of body parts as 2-dimensional positions the Average Precision of Key-Points (APK) measure is natural, Yang and Ramanan (2013) refer to it as the “golden standard”. A correct prediction falls inside a small region of the annotated point. To be precise, for a part at the annotated location \(i^*\), the loss for a prediction \(\hat{i}\) is defined to be

\[
\Delta^p(i^*, \hat{i}) = 1[\|i^* - \hat{i}\| > \alpha \max(h, w)] ,
\]

where \(1\) is the indicator function. The loss depends on the size of the object that we are looking for (namely height \(h\) and width \(w\)) and a threshold \(\alpha\) to restrict the region where we count a part as detected. Yang and Ramanan (2013) choose \(\alpha\) to be equal to 0.1 on full body pose estimation tasks.

3.4.2.2 Objective Function:

We use a structured maximum-margin estimator (Tsochantaridis et al., 2005) as in eq. (2.54) to encourage the model to fit parameters that lead to a low loss
Figure 3.7: Qualitative results for the LSP dataset: each pair of images is showing an input image from the test set in black and white overlaid with the FoP predictions. The left side of each pair shows the marginal prediction where probabilities per location closer to 1 are depicted with more opaque colors. The right side of each pair includes the stick-man representation of the prediction and for each stick-man joint we pick the location with the highest probability of a present part. All parts in a group share the same color, i.e., head in orange, left arm in purple, right arm in light blue, left leg in red and right leg in dark blue.
\( \Delta^p \). Similar to the loss we decompose the optimization problem along the parts

\[
\begin{align*}
\text{minimize}_{w, \xi, \xi^p \geq 0} & \quad \sum_p \xi^p + C\|w\|^2_2 \\
\text{subject to} & \quad s^p_{i^*} - s^p_i \geq \Delta^p(i^*, i) - \xi^p \quad \forall p, \forall i \in Y^p \\
& \quad s^p_i := \sigma^{-1}(q^p_i(1 \mid w)) = -\ln \left( \frac{1}{q^p_i(1 \mid w)} - 1 \right).
\end{align*}
\]

Equation (3.16) demands a margin of \( \Delta^p(i^*, i) \) between the inverse-sigmoid-score of the annotated state \( i^* \) and every other state \( i \). Maximizing a margin between the values \( s_i \) and \( s_j \) increases the margin between the log scores for both state \( 1 \) at the ground truth location and the state \( 0 \) at the offending location

\[
\begin{align*}
s_i - s_j &= \sigma^{-1}(q_i(1)) - \sigma^{-1}(q_j(1)) \\
&= -\ln \left( \frac{1}{q_i(1)} - 1 \right) + \ln \left( \frac{1}{q_j(1)} - 1 \right) \\
&= \ln \left( \frac{q_i(1)}{q_j(1)} \right) - \ln \left( \frac{1 - q_i(1)}{1 - q_j(1)} \right) \\
&= (\ln(q_i(0)) + \ln(q_j(0))) + (\ln(q_j(1)) + \ln(q_i(1))).
\end{align*}
\]

The slack variable \( \xi^p \) tracks the constraint violation and turns the optimization problem into a soft-margin objective (c.f. eq. (2.51)) with the regularizer \( C\|w\|^2_2 \). To prevent over-fitting to training data of our experiments we set the regularization weight \( C \) to 0.001 and did not change this value over the course of the experiments.

3.4.2.3 **Optimization**

We can rewrite eq. (3.16) similar to eq. (2.58) equivalently as an unconstrained optimization problem

\[
\begin{align*}
\text{minimize}_w & \quad \sum_p \max(0, -s^p_{i^*} + \max_{i \in Y^p}(s_i + \Delta^p(i^*, i))) + C\|w\|^2_2.
\end{align*}
\]

Every evaluation of the unconstrained objective contains solutions to a loss-augmented inference problem of the APK loss. This problem decomposes
Figure 3.8: Visual comparison of the prediction of the model by Yang and Ramanan (2013) (left column) and marginal distributions and stick-man predictions of the FoP base model (middle and right column). The color coding follows the description in fig. 3.7.
over parts and the offending state is the maximum in each loss-augmented field. This objective is piecewise differentiable and we resort to a stochastic sub-gradient method. We apply ADADELTA (Zeiler, 2012), with decay parameter $0.95$ and $\epsilon = 10^{-8}$.

We execute only a finite number of mean field iterations and in our experiments we chose a fixed number of $10$ iterations to calculate the marginals $q_i^p(x_i^p)$ from eq. (3.21). The performance does not depend on any convergence that may occur when the inference is run longer. When optimizing eq. (3.21) we take this into account by computing the gradient of the marginals with respect to the parameters by backpropagating the objective eq. (3.21) through the mean field updates as illustrated in fig. 3.6. This is an application of the back-mean-field idea of Domke (2013), a procedure advocated for learning with approximate inference when predicting with marginal inference.

Our implementation computes all gradients with automatic differentiation. This allows us to implement the gradient for basic computational building blocks only and chain these operations without worrying about the overall gradient. Automatic differentiation differs from symbolic differentiation, which derives the gradient by inspecting the analytic form of the function, and numeric differentiation, which computes gradients by applying the difference quotient. Nocedal and Wright (2006, chapter 8.2) or Griewank and Andrea Walther (2008) contain an overview of this topic. Symbolic differentiation might be prohibitive to apply for complex implementations like ours that rely on branching code that is difficult to formulate in an analytic form. Numerical differentiation requires multiple evaluations of the objective function with slightly shifted inputs. In contrast to the two methods, automatic differentiation is defined on the implementation of the objective function. It tracks the invoked building blocks depending on the input to the function and applies the chain rule to compute the gradient either during the forward pass through the algorithm, an additional backward pass or an intermediate version of the two. We have experimented with different implementations of automatic differentiation that work on either variable values or vectors, or that employ different evaluation strategies. A vector based backward-mode implementation worked best for us and allowed us to quickly change and try objective variants.

3.5 Experiments

We empirically test the proposed method with the standard benchmark dataset of LSP (Johnson and Everingham, 2010). This dataset consists of 1000
training and 1000 test images of people performing sports activities and is challenging due to strong body pose articulations. We have included the result of test-set images for visual inspection in fig. 3.7.

3.5.1 **Comparison to Pictorial Structures**

The idea of reparametrization the body pose problem can in principle be applied to other PS variants. Here we chose the model (Yang and Ramanan, 2011), and thus it serves as the PS “counterpart” we compare against. Note that the described FoP model uses exactly the same unary potentials and exactly the same features for the pairwise potentials. Also we use the same preprocessing steps: clustering and assignment of the types on the training dataset. Both models have almost identical number of parameters, a total of about 130k most of them unary parameters $w_{unary}$. Any performance difference of the two methods thus can be attributed solely to the change in model structure, learning objective and inference.

Table 3.1 reports the direct comparison using $APK$, some example comparing the detections are depicted in fig. 3.8. First we compare FoP to the PS counterpart and observe that we obtain an improvement for every body part, while being on par on “wrist”. The improvement in average $APK$ is 5.2%. For all FoP results we use the top prediction per image only, and have not implemented Non-Maximum-Suppression to retrieve multiple detections. The results of Yang and Ramanan (2013) when reporting only the top scoring part are also included in the table, in this case the performance gain is 6.0%. The results increase over all body parts, most prominently on the feet, for example more than 12% on ankles.

When comparing the extensions (Bilateral, Segmentation, Loopy) against the FoP model we observe a modest but consistent improvement. Again results increase across all parts. Since all models are trained in the same way this effect can only be due to the image conditioning terms and extra connections that we introduced. Figure 3.9 elaborates on how segmentation information in the pairwise connections affects inference qualitatively.

3.5.2 **Comparison with State-of-the-Art**

We also compare using the Percentage of Correct Parts ($PCP$) measure to Yang and Ramanan (2013) and recent results from the literature (cf. table 3.2). The

---

1 We thank Yang and Ramanan (2013) for making the code (version 1.3) publicly available.
Figure 3.9: Comparison of the FoP model with and without additional segmentation information for a test image from the LSP dataset: (a) and (b) show the predictions of the FoP model. We depict the marginal distribution on the left by overlaying the gray-scale version of the image with colored squares that are more opaque the closer the probability is to 1. The right side of the pair shows a stick-man representation of the prediction. The additional segmentation information captured by the pairwise connections of FoP-Segmentation model in (d) and (e) guides the marginals away from the stone statue and considerably sharpens them. The predictions of Yang and Ramanan (2013) are included in (c).
### Table 3.1: Comparison of pose estimation results on the LSP dataset. Shown are the Average Precision of Key-Points (APK) results with observer-centric annotations from Eichner and Ferrari (2012). Better methods achieve a higher score.

<table>
<thead>
<tr>
<th>Model</th>
<th>Head</th>
<th>Shoulder</th>
<th>Elbow</th>
<th>Wrist</th>
<th>Hip</th>
<th>Knee</th>
<th>Ankle</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoP-Unary</td>
<td>44.7</td>
<td>28.7</td>
<td>2.1</td>
<td>3.3</td>
<td>5.6</td>
<td>5.8</td>
<td>25.8</td>
<td>16.6</td>
</tr>
<tr>
<td>FoP</td>
<td>83.1</td>
<td>76.5</td>
<td>55.2</td>
<td>29.0</td>
<td>74.8</td>
<td>70.3</td>
<td>63.7</td>
<td>64.7</td>
</tr>
<tr>
<td>FoP-Bilateral</td>
<td>83.3</td>
<td>77.0</td>
<td>56.2</td>
<td>30.9</td>
<td>76.1</td>
<td>71.2</td>
<td>64.5</td>
<td>65.6</td>
</tr>
<tr>
<td>FoP-Segmentation</td>
<td>84.9</td>
<td>77.7</td>
<td>56.9</td>
<td>29.7</td>
<td>78.1</td>
<td>71.9</td>
<td>65.2</td>
<td>66.4</td>
</tr>
<tr>
<td>FoP-Loopy</td>
<td>83.0</td>
<td>76.2</td>
<td>55.7</td>
<td>29.0</td>
<td>77.7</td>
<td>72.0</td>
<td>64.3</td>
<td>65.4</td>
</tr>
<tr>
<td>Yang and Ramanan (2013)</td>
<td>80.0</td>
<td>75.2</td>
<td>48.2</td>
<td>28.9</td>
<td>70.4</td>
<td>60.5</td>
<td>53.2</td>
<td>59.5</td>
</tr>
<tr>
<td>Yang and Ramanan (2013): single detection</td>
<td>79.5</td>
<td>74.9</td>
<td>47.6</td>
<td>28.4</td>
<td>69.9</td>
<td>59.0</td>
<td>51.6</td>
<td>58.7</td>
</tr>
<tr>
<td>Pishchulin et al. (2013b)</td>
<td>88.0</td>
<td>80.6</td>
<td>60.4</td>
<td>38.2</td>
<td>81.8</td>
<td>74.9</td>
<td>65.4</td>
<td>69.9</td>
</tr>
</tbody>
</table>

### Table 3.2: Pose estimation results using the Percentage of Correct Parts (PCP) criterion on the Leeds Sport Poses dataset. A more precise location of the body parts results in a higher PCP score. We compare our method against the current top performing methods in the literature that train on this dataset with observer-centric annotations from Eichner and Ferrari (2012).

<table>
<thead>
<tr>
<th>Model</th>
<th>Torso</th>
<th>Upper leg</th>
<th>Lower leg</th>
<th>Upper arm</th>
<th>Forearm</th>
<th>Head</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoP</td>
<td>82.2</td>
<td>71.8</td>
<td>66.5</td>
<td>52.0</td>
<td>27.7</td>
<td>76.8</td>
<td>59.5</td>
</tr>
<tr>
<td>FoP-Bilateral</td>
<td>83.4</td>
<td>72.8</td>
<td>67.0</td>
<td>52.2</td>
<td>28.0</td>
<td>77.0</td>
<td>60.0</td>
</tr>
<tr>
<td>FoP-Segmentation</td>
<td>84.4</td>
<td>74.4</td>
<td>67.1</td>
<td>53.3</td>
<td>27.4</td>
<td>78.4</td>
<td>60.7</td>
</tr>
<tr>
<td>FoP-Loopy</td>
<td>81.8</td>
<td>73.7</td>
<td>66.9</td>
<td>52.0</td>
<td>26.8</td>
<td>77.3</td>
<td>59.8</td>
</tr>
<tr>
<td>Yang and Ramanan (2013)</td>
<td>81.0</td>
<td>67.4</td>
<td>63.9</td>
<td>51.0</td>
<td>31.8</td>
<td>77.3</td>
<td>58.6</td>
</tr>
<tr>
<td>Andriluka, Roth, and Schiele (2009)</td>
<td>80.9</td>
<td>67.1</td>
<td>60.7</td>
<td>46.5</td>
<td>26.4</td>
<td>74.9</td>
<td>55.7</td>
</tr>
<tr>
<td>Pishchulin et al. (2013a)</td>
<td>87.5</td>
<td>75.7</td>
<td>68.0</td>
<td>54.2</td>
<td>33.9</td>
<td>78.1</td>
<td>62.9</td>
</tr>
<tr>
<td>Pishchulin et al. (2013b)</td>
<td>88.7</td>
<td>78.8</td>
<td>73.4</td>
<td>61.5</td>
<td>44.9</td>
<td>85.6</td>
<td>69.2</td>
</tr>
<tr>
<td>Eichner and Ferrari (2012)</td>
<td>86.2</td>
<td>74.3</td>
<td>69.3</td>
<td>56.5</td>
<td>37.4</td>
<td>80.1</td>
<td>64.3</td>
</tr>
</tbody>
</table>
FoP model performs better than the PS models (Andriluka, Roth, and Schiele, 2009; Yang and Ramanan, 2013).

When comparing the differences between Yang and Ramanan (2013) and the FoP models we observe that a higher APK number is not directly translating into higher PCP scores. Especially on the arms, the APK criterion with a threshold of $\alpha = 0.1$ that was used during training, appears not to be indicative of PCP performance. The FoP model makes more points correct in terms of APK and we conjecture that switching to a parametrization based on sticks, the model will improve results on the PCP loss.

Methods that make use of richer appearance information, e.g., Poselets in Pishchulin et al. (2013a), Poselets and extra Deformable Part Model (DPM) detectors for every body part in Pishchulin et al. (2013b) or assumptions about the background color distribution from Eichner and Ferrari (2012), achieve higher results in terms of PCP. We are encouraged by the result of Eichner and Ferrari (2012) and believe that adapting their color background model should result in similar performance gains, especially, since they extend Yang and Ramanan (2013) by an additional unary factor. Pishchulin et al. (2013a) and Pishchulin et al. (2013b) make use of mid-level representations for bodies. We already discussed a possibility to adapt and extend their approach to a locally conditioned term in section 3.3.2.

3.6 Future Work

We mention some additional possibilities for future work. Beyond standard RGB, different texture and color information can be encoded in $z$. An interesting example is the mid-level representation used in Pishchulin et al. (2013a). The authors condition the pairwise terms of a PS model globally on responses of a poselet detector (Bourdev et al., 2010) and report impressive performance gains. With the FoP model this type of evidence can be included locally. A connection strength between variables can be modulated given that they are in mutual agreement with a poselet response at a corresponding position.

Another route is to combine the FoP model with another body parametrization as a collection of sticks/card-boards. For example a Field of Sticks can be fused into the model in the same way the body part fields are connected.

We expect a larger increase in performance from applying different features and substitute the underlying HOG features with a deep neural network, similar to Tompson, A. Jain, et al. (2014). Training of all parameters would still be possible and we believe that the model would profit from both the flexibil-
ity for the unary features and the additional encoded structured knowledge in the FoP model on top.

3.7 Conclusion

We have introduced the FoP model, a binary CRF formulation for human pose estimation. Despite being different in structure, it allows to encode a similar spatial dependency structure as done in PS. Further, it permits extensions with more general image conditioned part connections. We have shown two applications of this, by including color and segmentation information as extra features. We have demonstrated how to perform inference and learning in this model through a technical extension of Krähenbühl and Koltun (2011), and a max-margin estimator for parameter learning. Because inference complexity depends linearly on almost all relevant model dimensions we also implemented a variant with denser connections than just along the kinematic chain. Experimentally, we validated that the FoP model outperforms Yang and Ramanan (2013) on equal ground.

The important new dimension of the proposed parametrization is that it opens up connections to image segmentation. We have discussed interesting extensions of this model in section 3.3.2: image conditioned part configurations, combination with cardboard models, changes in graph topology, etc. Extensions to an explicit person and/or body part segmentation can be easily included, especially, because the inference needs not to be changed.

An interesting aspect of the FoP model is that it explains the image locally at every position; it is not affected by image size, number of persons in the image, or their size. This output semantic differs drastically compared to the PS model. Further along this direction, we believe that we can change the sequential process of current pose estimation pipelines into a single process that performs joint detection and pose estimation of multiple people. Pishchulin, Insafutdinov, et al. (2016) state an Integer Linear Program (ILP) to solve the part association problem and to reason about consistent body prediction.
The revolutionary success of high-capacity deep learning architectures, e.g., Krizhevsky, Sutskever, and Hinton (2012), Zhu et al. (2015), and Long, Shelhamer, and Darrell (2015), may flag the beginning of a paradigm shift in computer vision. One fundamental question arising in this context is how the required ground truth labels for training these models can be generated at very large scales (i.e., > 100k images). While for some tasks large annotated datasets are already available today (e.g., image classification in Russakovsky et al. (2015)), other tasks such as semantic segmentation of street scenes lack this information as human annotation is labor-intensive. There is an inherent trade-off between the number of annotated images and time it takes to manually label an image, as depicted in fig. 4.1. We aim to make a step towards the solution of this problem.

One option to circumvent this problem is to exploit auxiliary tasks for which large annotated datasets are available. While generalization to the target domain can be achieved to some extent, discriminative cues which solve the auxiliary problem will dominate the learned representation (Zhou et al., 2015). A second option is the creation of synthetic datasets. Unfortunately, our community still lacks rich generative image formation models which are able to produce realistic and diverse imagery from the true underlying distribution of the 3-dimensional world we live in. In this work, we therefore propose an alternative approach which leverages additional 3-dimensional information to simplify the 2-dimensional annotation task.

Recently, applications such as autonomous cars and humanoid robots have attracted significant attention. For research in these applications, a street view video dataset with dense semantic labels will be very useful. Motivated by those needs, our work focuses on the challenging task of semantic and instance video annotation of street scenes for which pixelwise labeling requires up to 60 minutes per image for a human annotator as acknowledged in Badrinarayanan, Galasso, and Cipolla (2010). Inspired by the easy usage of 3-dimensional modeling tools (Blender, SketchUp) we propose to annotate scenes directly in 3 dimensions and then transfer this knowledge back into the image
Figure 4.1: There is an inherent trade-off between the size of a training data set and the time required to label a single sample. In this work we propose a method that achieves a low annotation time per image and a large annotated sample set.

Domain. The required 3-dimensional information can be obtained from various sources including Structure-from-Motion (SfM), stereo or laser scanners. This approach has advantages over labeling in 2 dimensions: First, objects often project into multiple images of the video sequence, thus lowering annotation efforts considerably. Further, the obtained 2-dimensional instance annotations are temporally coherent as they are associated with a single object in 3 dimensions. And finally, our 3-dimensional annotations might be useful by themselves for reasoning in 3 dimensions (Zhang, Geiger, and Urtasun, 2013; Geiger and C. Wang, 2015) or to enrich 2-dimensional annotations with approximate 3-dimensional geometry.

Unfortunately, obtaining dense and accurate 2-dimensional labels from sparse noisy point clouds and coarse 3-dimensional annotations is a challenging task by itself. Towards solving this problem, we propose a non-local multi-field CRF model which reasons jointly about semantic and instance labels of all 3-dimensional points and all pixels in the image as illustrated in fig. 4.2. This approach offers several advantages over methods which reason purely in 2 dimensions (Badrinarayanan, Budvytis, and Cipolla, 2014; Vijayanarasimhan and Grauman, 2012): Occluders and occludees which exhibit complex boundaries when projected onto the image plane, e.g., a tree in front of a building, are often easier to separate in 3 dimensions. Besides, our approach is not affected by missing labels due to occlusions or drift in optical flow. Further, our
model allows to specify a tractable semantic instance loss for principled and efficient end-to-end parameter learning. And finally, the probabilistic nature of our model allows for estimating label uncertainties which can be used to increase label accuracy when only a subset of the pixels require a label. In summary, we make the following two contributions in this work:

• We present a novel geo-registered dataset of suburban scenes recorded by a moving platform. The dataset comprises over 400k images and over 100k laser scans, and we provide semantic 3-dimensional annotations for all static scene elements.

• We propose a method which is able to transfer these labels from 3 dimensions into 2 dimensions, yielding pixelwise semantic instance annotations. We demonstrate the potential of our approach in ablation studies and with respect to several 2-dimensional and 3-dimensional baselines.

4.1 RELATED WORK

In this section, we first review semi-supervised video annotation methods, followed by an overview over existing semantic and instance segmentation datasets.

4.1.1 Methods

Compared to annotating individual images (Xu, Schwing, and Urtasun, 2014; Guillaumin, Küttel, and Ferrari, 2014; Liu, Yuen, and Torralba, 2011), video sequences offer the advantage of temporal coherence between adjacent frames. Label propagation techniques exploit this fact by transferring labels from a sparse set of annotated key-frames to all unlabeled frames based on color and motion information. While in some works a single foreground object is assumed (S. D. Jain and Grauman, 2014; Tsai et al., 2012), here we focus on methods which can handle multiple object categories. Towards this goal, Badrinarayanan, Galasso, and Cipolla (2010) and Budvytis, Badrinarayanan, and Cipolla (2010) proposed a coupled Bayesian network based on video epitomes and semantic regions to propagate label information between two annotated keyframes. To better account for errors in label propagation, Nagaraja et al. (2012) proposed a hierarchy of local classifiers for this task and Badrinarayanan, Budvytis, and Cipolla (2014) leveraged a mixture-of-tree model for temporal
Figure 4.2: 3- to 2-dimensional label transfer. (a) We annotate all objects in 3 dimensions using bounding primitives. (b) Our model then transfers this information into 2 dimensions by jointly reasoning about 3-dimensional geometric cues, sparse 3-dimensional points, as well as image pixels. (c) This allows us to infer temporally consistent semantic instance annotations for every frame in the video.

association. The problem of selecting the most promising key frames for annotation has been considered in Vijayanarasimhan and Grauman (2012).

In contrast to the aforementioned methods which propagate labels in 2 dimensions, in this work we propose to annotate directly in 3 dimensions and then project these annotations into the 2-dimensional domain. While this approach requires a source of 3-dimensional information (e.g., SfM, stereo, laser), it is able to produce more accurate semantic and temporally consistent instance annotations. Further, our experiments indicate that annotation in 3 dimensions is more time efficient than labeling in 2 dimensions as scene el-
ments can be separated more easily and often project into multiple images of the input video sequence.

There exists little work on 3-dimensional to 2-dimensional label transfer. A notable exception is the approach of L.-C. Chen, Fidler, et al. (2014), where annotations from KITTI (Geiger, Lenz, Stiller, et al., 2013) as well as 3-dimensional car models are leveraged to infer separate figure-ground segmentations for all vehicles in the image. In comparison, our approach reasons jointly about all objects in the scene and also handles categories for which CAD models or 3-dimensional point measurements are unavailable (e.g., “tree”, “sky”). In the context of street view image segmentation, Xiao and Quan (2009) present a hybrid method where annotated 3-dimensional points from SfM are projected onto superpixels in the image and users interactively correct wrong predictions with 2-dimensional scribbles. However, as no occlusion reasoning is performed, their method can only be applied to scenes with little variations in depth (e.g., facades). Other methods (G. Brostow et al., 2008; Munoz, J. Andrew Bagnell, et al., 2009; Munoz, James Andrew Bagnell, and Hebert, 2012; Namin et al., 2015; Martinović et al., 2015) which model the interaction between image pixels and 3-dimensional points focus primarily on improving classification performance or efficiency by exploiting multiple input modalities while our goal is to transfer ambiguous 3-dimensional primitive labels to every pixel in the image.

4.1.2 Datasets

While some datasets such as PASCAL VOC 2012 (Everingham et al., 2010) or MS COCO (Lin et al., 2014) provide semantic labels for a subset of pixels in the image, here we focus on datasets with dense semantic annotations. Most of these datasets provide only a small number (~1k) of accurately annotated indoor (Silberman et al., 2012) or outdoor (Jamie Shotton et al., 2009; Gould, Fulton, and Koller, 2009) images. A notable exception is LabelMe (Russell et al., 2008) with more than 10k images labeled using crowdsourcing techniques. Compared to the smaller datasets, however, not all images are densely annotated, quality varies heavily among annotators, and polygons have been chosen over pixels as more efficient but less accurate representation.

A number of works have also considered the annotation of video sequences (G. J. Brostow, Fauqueur, and Cipolla, 2009; Xiao, Owens, and Torralba, 2013; Song, Lichtenberg, and Xiao, 2015). In Xiao, Owens, and Torralba (2013), eight RGB-D sequences of indoor scenes have been manually annotated us-
ing an interactive tool which propagates 2-dimensional polygons from one frame to another. The recently proposed SUN RGB-D dataset (Song, Lichtenberg, and Xiao, 2015) provides labeled 2-dimensional polygons as well as 3-dimensional cuboids for 10k RGB-D images captured indoors. For street scenes, less annotated data is available (Behley, Steinhage, and Cremers, 2012; Munoz, J. Andrew Bagnell, et al., 2009; Munoz, James Andrew Bagnell, and Hebert, 2012; Riemenschneider et al., 2014; Valentin et al., 2013). While KITTI (Geiger, Lenz, and Urtasun, 2012) provides semantic information only for a few object categories¹, CamVid (G. J. Brostow, Fauqueur, and Cipolla, 2009) offers pixel-accurate labels, but without instances and for a very limited number of frames. Recently, the Cityscapes dataset (Cordts et al., 2016) has been proposed with 5k manually annotated individual 2-dimensional images of street scenes². Our dataset differs from Cityscapes in that we provide temporally coherent semantic instance annotations at a much larger scale as well as omnidirectional imagery, 3-dimensional laser scans and 3-dimensional annotations which might also be directly useful for reasoning in 3 dimensions. While Cordts et al. (2016) focuses on inner-city scenes, our dataset comprises mainly suburban areas, thus both datasets complement each other.

4.2 Method

In this work, we are interested in generating semantic instance annotations for urban scenes at large scale by transferring labels from sparse 3-dimensional point clouds into the images. In particular, we focus on static scene elements which dominate suburban scenes. Dynamic objects could be handled via 3-dimensional models (Menze and Geiger, 2015; L.-C. Chen, Fidler, et al., 2014) but as our dataset comprises little dynamic objects we leave this extension for future work. This section describes our data collection efforts, our 3-dimensional annotation process, as well as the proposed label transfer model.

4.2.1 Data Collection

For our data collection, we equipped a station wagon with one 180° fisheye camera to each side and a 90° perspective stereo camera (baseline 60 cm) to the front. Furthermore, we mounted a Velodyne HDL-64E and a SICK LMS 200 laser scanning unit in pushbroom configuration on top of the roof. This setup

¹ http://www.cvlibs.net/datasets/kitti/
² http://www.cityscapes-dataset.net/
is similar to the one used in KITTI (Geiger, Lenz, and Urtasun, 2012; Geiger, Lenz, Stiller, et al., 2013), except that we gain a full 360° field of view due to the additional fisheye cameras and the pushbroom laser scanner while KITTI only provides perspective images and Velodyne laser scans with a 26.8° vertical field of view. Compared to omnidirectional camera systems (Schönbein and Geiger, 2014; Schönbein, Strauss, and Geiger, 2014) our setup benefits from increased resolution. Approximate localization is provided by an IMU/GPS measurement unit.

Using this setup, we recorded several suburbs of a mid-size city corresponding to over 400k images and 100k laser scans. We estimated all vehicle and camera poses using SfM (Heng, B. Li, and Pollefeys, 2013). More specifically, we minimize 3-dimensional reprojection errors based on all feature matches while regularizing against the GPS solution. This results in accurate geo-registered camera poses. While our label transfer approach does not assume geo-localization, geo-spatial information\(^3\) can facilitate the 3-dimensional annotation task.

4.2.2 Annotation

We augmented our dataset with 3-dimensional annotations in the form of bounding primitives, i.e., we placed cuboids and ellipsoids around objects in 3 dimensions and assigned a semantic label to each of them. More specifically, we asked a group of annotators to tightly enclose the 3-dimensional points belonging to an object by the respective primitive. For this purpose, we developed a 3-dimensional annotation tool based on WebGL (see fig. 4.2a) which visualizes the colored point clouds (obtained by projecting the 3-dimensional points back onto multiple images), two camera views, and provides tools to facilitate navigation and annotation. To enable efficient annotation, our primitives are rough approximations of the true object shapes and thus are allowed to overlap in 3 dimensions (see fig. 4.2b). For stuff categories (e.g., “road”, “sidewalk”, “grass”) we allow users to draw 2-dimensional polygons in bird’s eye view which are then extruded into 3 dimensions to better approximate the shape and to facilitate annotation. Ambiguities are resolved using our label transfer method described in the following section. Annotating a single batch comprising 200 laser scans and 800 images required about 3 hours. While the focus of this work is on annotating static scene elements which cover the majority of pixels in general, our annotation GUI could be extended to a keyframe

\(^3\) [http://www.openstreetmap.org/](http://www.openstreetmap.org/)
based dynamic 3-dimensional video annotation tool which visualizes point clouds and images over time akin to the annotation utility developed for labeling the KITTI dataset (Geiger, Lenz, and Urtasun, 2012; Geiger, Lenz, Stiller, et al., 2013).

4.2.3 Model

Given sparse point clouds and 3-dimensional annotations, we are interested in generating dense semantic instance annotations for all images. Towards this goal, we propose a CRF model which reasons jointly about the labels of the 3-dimensional points and all pixels in the image, leveraging the calibration and registration described in section 4.2.1. Note that our 3-dimensional annotations are sparse and noisy, i.e., 3-dimensional points can carry none, one or multiple labels due to overlapping bounding primitives in 3 dimensions. The algorithm described in this section is designed to resolve these situations and infers marginal estimates for all 3-dimensional points and pixels in the image. In order to make our approach more robust in regions where appearance is not discriminative, we investigate additional geometric cues of the 3-dimensional point cloud such as 3-dimensional surface folds and curbs (see fig. 4.3b). If de-
4.2 Method

tected, these cues can provide accurate boundaries between semantic classes in
the image.

More formally, let $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{F}$ denote the index sets of image pixels, sparse
3-dimensional points from laser/stereo, and detected 3-dimensional fold or
curb segments, respectively. For each pixel $i \in \mathcal{P}$ and each 3-dimensional
point $l \in \mathcal{L}$, we specify random variables $Y_i$ and $Y_l$ taking values from the set
of semantic (or instance) labels $\mathcal{Y}$. For instance inference, we assign a unique
ID to each object which projects into the image. Thus, semantic and instance
inference can be treated equally under our model and we refer to both as “se-
matic labels” in the following.

Let $\mathbf{y} = \{y_i \mid i \in \mathcal{P}\} \cup \{y_l \mid l \in \mathcal{L}\}$ denote the semantic configuration. We specify our
CRF in terms of the following Gibbs energy function (cf. section 2.2.1):

$$
E(\mathbf{y}) = \sum_{i \in \mathcal{P}} \Phi_{i}^{\mathcal{P}}(y_i) + \sum_{l \in \mathcal{L}} \Phi_{l}^{\mathcal{L}}(y_l) + \sum_{m \in \mathcal{F}} \sum_{i \in \mathcal{P}} \Phi_{m,i}^{\mathcal{F}}(y_i)
+ \sum_{i,j \in \mathcal{P}} \Phi_{i,j}^{\mathcal{P},\mathcal{P}}(y_i, y_j) + \sum_{l,k \in \mathcal{L}} \Phi_{l,k}^{\mathcal{L},\mathcal{L}}(y_l, y_k) + \sum_{i \in \mathcal{P}, l \in \mathcal{L}} \Phi_{i,l}^{\mathcal{P},\mathcal{L}}(y_i, y_l)
$$

(4.1)

with unary potentials $\Phi_{i}^{\mathcal{P}}$, $\Phi_{l}^{\mathcal{L}}$ and $\Phi_{m,i}^{\mathcal{F}}$ and pairwise potentials $\Phi_{i,j}^{\mathcal{P},\mathcal{P}}$, $\Phi_{l,k}^{\mathcal{L},\mathcal{L}}$ and $\Phi_{i,l}^{\mathcal{P},\mathcal{L}}$. For notational clarity, we omit all conditional dependencies on
the input images, 3-dimensional points, 3-dimensional annotations and on
the parameter vector $\mathbf{w}$.

4.2.3.1 Pixel Unary Potentials

The pixel unary potentials $\Phi_{i}^{\mathcal{P}}(y_i \mid \mathbf{w})$ encode how likely pixel $i$ is taking label
$y_i$:

$$
\Phi_{i}^{\mathcal{P}}(y_i \mid \mathbf{w}) = w_{1}^{\mathcal{P}}(y_i) \xi_{i}^{\mathcal{P}}(y_i) - w_{2}^{\mathcal{P}}(y_i) \ln p_{i}^{\mathcal{P}}(y_i)
$$

(4.2)

where $w_{1}^{\mathcal{P}}$ and $w_{2}^{\mathcal{P}}$ denote learned feature weights that are part of $\mathbf{w}$. Our first
constraint $\xi_{i}^{\mathcal{P}}(s_i)$ determines the set of admissible labels and is obtained by
projecting the 3-dimensional bounding primitives, which are an upper bound
on the objects’ extent, into the image. We formulate the constraint via a binary
feature $\xi_{i}^{\mathcal{P}}(y_i) \in \{0, 1\}$ which takes 0 for pixel $i$ if its ray passes through a
primitive of class $y_i$, and 1 otherwise.

In addition, we leverage appearance information by projecting all non-
occluded sparse 3-dimensional points into all adjacent frames of the image se-
quence and training a pixel-wise classifier (Jamie Shotton et al., 2009) based
on these projections. This results in a per-pixel probability distribution over semantic labels $p^p_i(y_i)$. The intuition behind this feature is that regions of the same semantic class are similar in adjacent frames and thus yield highly discriminative cues for the current frame.

### 4.2.3.2 3-Dimensional Point Unary Potentials

The 3-dimensional point unary potentials $\phi^C_i(y_l)$ encode the unary energy of a 3-dimensional point $l$ taking label $y_l$:

$$\phi^C_i(y_l | w) = -w^C_i(y_l) \xi^C_i(y_l)$$

where $\xi^C_i(y_l)$ denotes a feature which takes 0 if the 3-dimensional point $l$ lies within a 3-dimensional primitive of class $y_l$, and 1 otherwise. As the “sky” class can’t be modeled with primitives we set $\xi^C_i(y_l)$ to 0 if $y_l$ takes the label “sky”. Additionally, we create “virtual sky points” at infinity for all pixels whose ray doesn’t intersect any 3-dimensional primitive. Note that these pixels must correspond to sky regions as we assume that each object is completely contained in one or several bounding 3-dimensional primitives.

### 4.2.3.3 Geometric Unary Potentials

We encourage label changes at curbs or folds which we detect in 3 dimensions using plane fitting as described in section 4.2.3.7. Given the projections into 2 dimensions, we introduce the following constraint:

$$\phi_{m,i}^F(y_i | w) = w^F_i \frac{1 [z_i^p \in \mathcal{R}_m \land v_m(z_i^p) \neq y_i]}{\exp(\text{dist}(z_i^p, y_m))}$$

Here, $1$ is the indicator function from eq. (2.4), $z_i^p$ denotes the 2-dimensional location of pixel $i$ and $\mathcal{R}_m$ represents a 2-dimensional disc around curb or fold segment $m$ projected into 2 dimensions (yielding a line segment $y_m$) as illustrated in fig. 4.4. $v_k$ is a function which takes as input a pixel location and returns the semantic label predicted by fold $m$. More specifically, we project the 3-dimensional fold into 2 dimensions and compute the majority label at its two sides from the sparse projected 3-dimensional points. The denominator in eq. (4.4) ensures a penalty decay towards the disc boundaries.

### 4.2.3.4 Pixel Pairwise Potentials

Our dense pairwise term encourages semantic label coherence and connects
4.2 Method

Figure 4.4: Geometric unary potentials. (a) We encourage label changes at 3-dimensional curbs or folds after projection into the image domain. (b) This constraint ($\phi_{m}^{F}$) is implemented by pixel unary potentials inside each minimum bounding disc $\mathcal{R}_{m}$ around each 2-dimensional curb or fold segment.

all pixels in the image via Gaussian edge potentials

$$
\phi_{i,j}^{p,p}(y_{i}, y_{j}) = \mu_{1}^{p,p}(y_{i}, y_{j}) \exp\left(-\frac{\|z_{i}^{p} - z_{j}^{p}\|_{2}^{2}}{2 \Sigma_{1}^{p,p}}\right) + \mu_{2}^{p,p}(y_{i}, y_{j}) \exp\left(-\frac{\|z_{i}^{p} - z_{j}^{p}\|_{2}^{2}}{2 \Sigma_{2}^{p,p}} - \frac{\|z_{i}^{c} - z_{j}^{c}\|_{2}^{2}}{2 \Sigma_{3}^{p,p}}\right) \quad (4.5)
$$

where $z_{i}^{p}$ is the 2-dimensional location of pixel $i$ and $z_{i}^{c}$ denotes its color value. Further, $\mu_{1}^{p,p}$ and $\mu_{2}^{p,p}$ are learned pairwise feature weights and $\Sigma^{p,p}$ parametrizes the kernel width.

4.2.3.5 3-Dimensional Pairwise Potentials

Similarly, we apply a Gaussian edge kernel to encourage label consistency between 3-dimensional points based on their 3-dimensional location and surface normals

$$
\phi_{i,k}^{\mathcal{L},\mathcal{L}}(y_{i}, y_{k}) = \mu_{1}^{\mathcal{L},\mathcal{L}}(y_{i}, y_{k}) \exp\left(-\frac{\|z_{i}^{3d} - z_{k}^{3d}\|_{2}^{2}}{2 \Sigma_{1}^{\mathcal{L},\mathcal{L}}} - \frac{(z_{i}^{n} - z_{k}^{n})^{2}}{2 \Sigma_{2}^{\mathcal{L},\mathcal{L}}}\right) \quad (4.6)
$$

where $z_{i}^{3d}$ is the 3-dimensional location of point $i$ and $z_{i}^{n}$ denotes the vertical (up) component of its normal. We use the normal’s third component as it is the most discriminative cue for indicating label changes between horizontal (e.g., “road”, “sidewalk”) and vertical (e.g., side of “car”, “wall”) surfaces. We estimate the respective normals using principle component analysis in a local neighborhood around each 3-dimensional point.
4.2.3.6 2-Dimensional/3-Dimensional Pairwise Potentials

Finally, we encourage coherence between all 3-dimensional points and the image pixels

$$\Phi_{i, l}^{p, l}(y_i, y_l) = \mu^{p, l}(y_i, y_l) \exp\left(-\frac{\|z_i^p - z_l^l\|^2}{2 \sum^{y_p, l}}\right)$$  \hspace{1cm} (4.7)

where $z_l^l$ denotes the projection of the 3-dimensional laser or stereo point $l$ onto the image plane. Importantly, we project only points into the image which are likely to be visible. We determine these points by meshing the 3-dimensional point cloud using the ball-pivoting method of Bernardini et al. (1999), considering only 3-dimensional points in front of the mesh. We also tried state-of-the-art multi-view reconstruction approaches (Jancosek and Pajdla, 2011) for mesh generation, but obtained better results with the described meshing approach.

4.2.3.7 3-Dimensional Fold/Curb Detection

We detect folds and curbs in the 3-dimensional point cloud to disambiguate the semantic class at object boundaries. We first extract all relevant object class boundaries by finding all gradients over semantic classes in the annotated 3-dimensional point cloud that pass a fixed threshold, i.e., we sweep a 3-dimensional gradient operator over the semantic 3-dimensional point cloud. For each boundary point, we fit two perpendicular 3-dimensional planes and extract their intersection in terms of a 3-dimensional fold (see fig. 4.4). The sole exception are boundaries between “road” and “sidewalk” for which we detect the bottom part of the curb by training an SVM on shape context features (Belongie, Malik, and Puzicha, 2002) (see fig. 4.3b). Due to the small elevation of the curb and the noise in the 3-dimensional data we found this to perform better than 3-dimensional plane fitting in terms of separating the objects in 3 dimensions.

As the fold detections are noisy, we model the true fold location as a random variable and penalize the deviation of the estimate $\hat{y}_m$ from the detection $\bar{y}_m$ while encouraging continuity and smoothness. We associate a random variable $Y_m$ with each 3-dimensional fold or curb $m \in \mathcal{F}$ which specifies the location and orientation of the fold segment in 3 dimensions. We discretize the set of possible fold segments for each detection by sampling from a local neighborhood around the parameters of the detection, i.e., the discrete outcomes of $Y_m$ depend on the detection $\bar{y}_m$. Each sample is associated with the corresponding fold segment parameters. We formulate a CRF model for optimizing
the placement of fold and curb segments with an energy function which encourages smoothness of adjacent segments in 2 dimensions:

$$E(y^F) = \sum_{m \in F} \phi_m^F(y_m) + \sum_{m,n \in F} \phi_{m,n}^{F,F}(y_m, y_n). \quad (4.8)$$

3-DIMENSIONAL FOLD/CURB UNARY POTENTIALS. The unary potential for the 3-dimensional fold segments and curbs is specified by a quadratic loss on the deviation of the estimated fold $y_m$ from its 3-dimensional detection $\bar{y}_m$:

$$\phi_m^F(y_m) = w_F \sum_{c \in \mathcal{C}} \| \rho_m(y_m, c) - \rho_m(\bar{y}_m, c) \|^2. \quad (4.9)$$

Here, $\mathcal{C} \subseteq [0, 1]$ is a finite set of 1-dimensional control points along the fold segment and $\rho_m(y_m, c)$ returns the corresponding 3-dimensional point. The potentials are illustrated in fig. 4.5.

3-DIMENSIONAL FOLD/CURB PAIRWISE POTENTIALS. For smoothing the boundaries, we introduce a pairwise term which encourages continuity between neighboring fold segments and curbs

$$\phi_{m,n}^{F,F}(y_m, y_n) = \begin{cases} 
\phi_m^F(y_m, y_n), & \text{if } (m, n) \in N^F \\
0, & \text{otherwise.}
\end{cases} \quad (4.10)$$

Figure 4.5: We model the uncertainty in the folds by introducing an auxiliary random variable $y_m$, $m \in F$, for each fold. The unary potential (a) encourages the folds direction to stay close to the plane fitted estimate $\bar{y}_m$. We also connect each fold to adjacent folds in (b) to smooth the folds’ orientations.
where smoothness of neighboring folds is defined via

$$
\Phi_{m,n}^{F,F}(y_m, y_n) = \frac{w_1^{F,F}}{2} \left( 1 - \frac{|\pi_m(y_m)^T \pi_n(y_n)|}{\|\pi_m(y_m)\|_2 \|\pi_n(y_n)\|_2} \right)
$$

(4.11)

+ $w_2^{F,F} \text{dist}(\rho_m(y_m, 1), \pi_n(y_n))$

+ $w_2^{F,F} \text{dist}(\rho_n(y_n, 0), \pi_m(y_m))$

and $\mathcal{N}^{F}$ denotes the set of neighboring folds in 3 dimensions, i.e., folds for which the endpoint of one fold segment is within a small distance from the start-point of the next segment. The function $\rho$ is defined as above. Function $\pi$ projects a point or fold segment from 3 dimension onto the image plane. This potential is illustrated in fig. 4.5b.

**Fold Inference.** Equation (4.8) corresponds to a non-loopy pairwise CRF as folds are connected in chains, e.g., along the sidewalk-road boundary. We obtain a global minimizer of the corresponding Gibbs energy via belief propagation as presented in section 2.3.2. The parameters of the model $w^{F}, w_1^{F,F}$ and $w_2^{F,F}$ have been set empirically to yield smooth results.

4.2.4 Learning and Inference

This section describes inference and parameter estimation in our label transfer model.

4.2.4.1 Inference

At test time, we are interested in estimating the marginal distribution of each semantic or instance label in $y$ under our model, specified by the Gibbs distribution defined in eq. (4.1). The most likely configuration can then be estimated by variable-wise maximization of these marginals. As our graphical model is loopy, exact inference in polynomial time is intractable. We resort to variational inference and approximate the probability distribution on $y$ by replacing it with a factorized mean field distribution $q(y) = \prod_{i \in \mathcal{P} \cup \mathcal{L}} q_i(y_i)$. This mean field approximation can be computed efficiently using bilateral filtering (Krähenbühl and Koltun, 2011) as described in section 2.3.4. As our model comprises three sets of dense connections (namely $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{P} \leftrightarrow \mathcal{L}$), we exploit the algorithm described in chapter 3 which generalizes Krähenbühl and Koltun (2011) to multiple fields.
4.2.4.2 Learning

We employ empirical risk minimization in order to learn the parameters in our model, considering the univariate logistic loss from eq. (2.48) on the marginal PMF $q(y)$. Let us subsume all model parameters into $w = \{w_1^P, w_2^P, w_3^F, w_1^P, w_2^P, w_3^P, w_4^L, w_5^L\}$. We define our minimization problem as a regularized univariate logistic instance:

$$\min_w \sum_{i=1}^{n} \sum_{j \in p} - \ln(q_{i,j}(y_{i,j}^* | w)) + \lambda \|w\|^2_2. \quad (4.12)$$

Here, $n$ is the number of training images, $y_{i,j}^*$ denotes the ground truth semantic label and $q_{i,j}$ the marginal at pixel $j$ in image $i$, calculated via mean field approximation. $\|w\|^2_2$ is a quadratic regularizer on the parameter vector $w$. We whiten all features and use a single value $\lambda$ which we select via cross-validation on the training set.

For learning the instance segmentation parameters we exploit the same optimization problem as for semantic segmentation. For instance segmentation, we assign unique labels to each individual object, e.g., different cars are assigned different labels even if they occlude each other. In order to associate 2-dimensional ground truth instances with 3-dimensional instances we project all visible 3-dimensional points into the image and find a consensus via the majority vote which gave good results in practice. As the number of instances per semantic class varies between images, we learn intra- and inter-class pairwise potentials using parameter tying.

We solve the optimization problem in eq. (4.12) using stochastic gradient descent and obtain $\frac{\partial q}{\partial w}$ using automatic differentiation as in section 3.4.2.3. We make use of the ADADELTA algorithm (Zeiler, 2012) with decay parameter $0.95$ and $\epsilon = 10^{-8}$, and randomly sample a batch of 16 training images at each iteration for which all gradients can be computed in parallel.

4.3 Experimental Evaluation

In this section, we first evaluate our method in ablation studies and with respect to several label transfer baselines. Besides, we exploit the uncertainty in our predictions to increase accuracy for semi-dense predictions. Finally, we show some qualitative results of our method. As input to our method, we accumulate all laser measurements in a common world coordinate system and augment them with 3-dimensional points from stereo matching.
To reduce outliers, we consider only points up to 15 m distance, and apply left-right as well as forward-backward consistency checks over 5 frames. We fuse all 3-dimensional points into one global point cloud and remove all points which are closer than 5 cm to their nearest neighbor. For evaluation, we manually annotated 160 images from 8 different suburbs with dense pixel-wise ground truth. From the 160 frames, 120 frames have been labeled in equidistant steps of 5 frames for comparison with 2-dimensional label transfer methods. We learn the parameters in our model and the baseline models using 2-fold cross validation at the sequence level to avoid any bias caused by the correlation of adjacent frames within a sequence. The kernel width parameters in our model have been chosen empirically to $\Sigma_1^{p,p} = 3$, $\Sigma_2^{p,p} = 43$, $\Sigma_3^{p,p} = 9$, $\Sigma_1^{L,L} = 0.05$ and $\Sigma_2^{L,L} = 1.0$ through coordinate descent on the validation set. Overall we found that our model is not very sensitive to the exact setting of these parameters.

### 4.3.1 Quantitative Evaluation

This section presents our quantitative evaluation. We compare our method with respect to several baselines on the semantic and instance segmentation tasks.

#### 4.3.1.1 Semantic Segmentation

For evaluating semantic segmentation performance, we map the 27 semantic labels in our 3-dimensional annotations to the most frequently occurring 14 categories. We measure overall performance by the average Intersection over Union (IoU) weighted by the class frequency and the average pixel accuracy.

The upper half of each row section in table 4.1 shows results of several 2-dimensional to 2-dimensional label transfer methods on all 120 equidistantly labeled frames. Here, the task is to predict the center frame from two annotated images (±5 frames corresponding to 0.5 seconds of driving or ~5 meters travel distance). Our first baseline (“Label Propagation”) is the label transfer approach presented in Vijayanarasimhan and Grauman (2012). To ensure that all baselines have access to the same information, we do not select frames in an active fashion but use equidistantly spaced labels for all methods. Note the driving speed during recording was nearly constant. We construct a second baseline (“Sparse Tracking + GC”) using the feature tracking approach of Sundaram, Brox, and Keutzer (2010) to propagate semantic labels from the two closest labeled frames to the target frame. To densify the label map, we ap-
4.3 Experimental Evaluation

Figure 4.6: Performance with respect to estimated pixels. This figure shows the average IoU score (a) and the average accuracy (b) when estimating only a fraction of the pixels which is selected according to the uncertainty in our predictions.

Graph Cut (GC) with contrast sensitive edge potentials (Boykov and Kolmogorov, 2004).

In order to evaluate the value of 3-dimensional information, we implemented a third baseline (“3D Propagation + GC”) which works similar to the previous one, but replaces the sparse tracking part with correspondences obtained by transferring pixels of the two closest labeled frames to the target image via the visible vertexes of our 3-dimensional mesh followed by GC propagation. Finally, we train the segmentation model of Krähenbühl and Koltun (2011) (“Fully Connected CRF”) on all annotated adjacent frames of the test sequence.

From the 2-dimensional label transfer baselines, the mesh transfer method which uses projected 3-dimensional information performs best. Furthermore, and maybe surprisingly, the image-specific fully connected CRF model performs on par or even better than special purpose label transfer methods. According to our experiments, this is caused by the fact that optical flow, which is used in Vijayanarasimhan and Grauman (2012) and Sundaram, Brox, and Keutzer (2010), often fails for street scenes like ours due to large displacements, perspective distortions, textureless regions and challenging lighting conditions. On the other hand, the fully connected model performs weaker for less frequent or textureless classes such as “trailer” or “box”.
<table>
<thead>
<tr>
<th>Method</th>
<th>Road</th>
<th>Park</th>
<th>Sdwlk</th>
<th>Terr</th>
<th>Bldg</th>
<th>Vegt</th>
<th>Car</th>
<th>Trler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Propagation</td>
<td>93.4</td>
<td>51.8</td>
<td>73.5</td>
<td>58.3</td>
<td>80.2</td>
<td>69.9</td>
<td>61.5</td>
<td>22.4</td>
</tr>
<tr>
<td>Sparse Tracking + GC</td>
<td>89.6</td>
<td>37.1</td>
<td>69.0</td>
<td>54.2</td>
<td>84.6</td>
<td>79.5</td>
<td>78.2</td>
<td>2.5</td>
</tr>
<tr>
<td>3D Propagation + GC</td>
<td>91.3</td>
<td>44.5</td>
<td>74.0</td>
<td>62.4</td>
<td>86.2</td>
<td>81.8</td>
<td>81.6</td>
<td>5.2</td>
</tr>
<tr>
<td>Fully Connected CRF</td>
<td>88.5</td>
<td>37.8</td>
<td>68.4</td>
<td>55.8</td>
<td>85.5</td>
<td>79.8</td>
<td>76.8</td>
<td>2.5</td>
</tr>
<tr>
<td>3D Primitives + GC</td>
<td>78.7</td>
<td>46.4</td>
<td>43.9</td>
<td>46.5</td>
<td>54.9</td>
<td>55.4</td>
<td>55.1</td>
<td>72.3</td>
</tr>
<tr>
<td>3D Mesh + GC</td>
<td>92.1</td>
<td>66.4</td>
<td>72.4</td>
<td>66.1</td>
<td>69.1</td>
<td>74.9</td>
<td>87.7</td>
<td>88.9</td>
</tr>
<tr>
<td>3D Points + GC</td>
<td>93.1</td>
<td>72.4</td>
<td>78.9</td>
<td>72.4</td>
<td>81.9</td>
<td>77.2</td>
<td>88.1</td>
<td>92.4</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>95.3</td>
<td>80.6</td>
<td>86.4</td>
<td>81.0</td>
<td>90.9</td>
<td>86.9</td>
<td>91.5</td>
<td>94.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Carvn</th>
<th>Gate</th>
<th>Wall</th>
<th>Fence</th>
<th>Box</th>
<th>Sky</th>
<th>IoU</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Propagation</td>
<td>42.3</td>
<td>30.6</td>
<td>45.3</td>
<td>45.7</td>
<td>32.5</td>
<td>89.6</td>
<td>74.4</td>
</tr>
<tr>
<td>Sparse Tracking + GC</td>
<td>35.3</td>
<td>3.2</td>
<td>38.9</td>
<td>32.9</td>
<td>7.0</td>
<td>91.0</td>
<td>77.8</td>
</tr>
<tr>
<td>3D Propagation + GC</td>
<td>38.6</td>
<td>12.7</td>
<td>47.4</td>
<td>42.0</td>
<td>15.0</td>
<td>88.8</td>
<td>80.2</td>
</tr>
<tr>
<td>Fully Connected CRF</td>
<td>30.6</td>
<td>2.9</td>
<td>38.3</td>
<td>32.4</td>
<td>0.0</td>
<td>92.8</td>
<td>77.9</td>
</tr>
<tr>
<td>3D Primitives + GC</td>
<td>54.6</td>
<td>51.0</td>
<td>40.2</td>
<td>52.3</td>
<td>40.2</td>
<td>55.4</td>
<td>56.4</td>
</tr>
<tr>
<td>3D Mesh + GC</td>
<td>88.5</td>
<td>61.9</td>
<td>51.4</td>
<td>60.7</td>
<td>30.4</td>
<td>46.4</td>
<td>72.5</td>
</tr>
<tr>
<td>3D Points + GC</td>
<td>91.1</td>
<td>70.2</td>
<td>66.8</td>
<td>68.7</td>
<td>62.4</td>
<td>69.9</td>
<td>80.5</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>91.8</td>
<td>73.6</td>
<td>78.9</td>
<td>79.4</td>
<td>73.0</td>
<td>91.0</td>
<td>89.2</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison to label transfer baselines on semantic segmentation task. We compare our method to 2-dimensional label transfer baselines (top) “Label Propagation” from Vijayanarasimhan and Grauman (2012), “Sparse Tracking + GC” from Sundaram, Brox, and Keutzer (2010) and “Fully Connected CRF” from Krähenbühl and Koltun (2011) to 3-dimensional to 2-dimensional label transfer baselines (bottom) on 120 consecutive images.

The bottom half of each row section in table 4.1 compares the proposed method with respect to several 3-dimensional to 2-dimensional label transfer baselines which in contrast to the 2-dimensional to 2-dimensional label transfer methods exploit our 3-dimensional annotations and does not require equidistantly labeled 2-dimensional annotations. As evidenced by our results, simply projecting 3-dimensional primitives or meshes into the image and smoothing via GC does not perform well due to the crude approximation of
<table>
<thead>
<tr>
<th>Method</th>
<th>Road</th>
<th>Park</th>
<th>Sdwlk</th>
<th>Terr</th>
<th>Bldg</th>
<th>Vegt</th>
<th>Car</th>
<th>Trler</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA</td>
<td>92.2</td>
<td>64.6</td>
<td>77.9</td>
<td>67.5</td>
<td>85.2</td>
<td>81.9</td>
<td>81.7</td>
<td>85.7</td>
</tr>
<tr>
<td>LA+3D</td>
<td>95.0</td>
<td>76.9</td>
<td>85.5</td>
<td>73.3</td>
<td>87.9</td>
<td>84.3</td>
<td>89.4</td>
<td>88.2</td>
</tr>
<tr>
<td>LA+PW</td>
<td>92.5</td>
<td>68.6</td>
<td>79.5</td>
<td>73.1</td>
<td>87.3</td>
<td>84.2</td>
<td>84.1</td>
<td>89.9</td>
</tr>
<tr>
<td>LA+PW+CO</td>
<td>93.0</td>
<td>72.7</td>
<td>81.2</td>
<td>73.8</td>
<td>87.7</td>
<td>84.5</td>
<td>85.7</td>
<td>90.9</td>
</tr>
<tr>
<td>LA+PW+CO+3D</td>
<td>93.2</td>
<td>78.6</td>
<td>85.0</td>
<td>76.3</td>
<td>90.6</td>
<td>86.7</td>
<td>89.1</td>
<td>90.9</td>
</tr>
<tr>
<td>+ 3D PW</td>
<td>94.9</td>
<td>80.1</td>
<td>85.9</td>
<td>80.0</td>
<td>90.6</td>
<td>87.0</td>
<td>91.2</td>
<td>91.3</td>
</tr>
<tr>
<td>Full Model</td>
<td>95.4</td>
<td>80.1</td>
<td>87.1</td>
<td>80.0</td>
<td>90.6</td>
<td>87.0</td>
<td>91.2</td>
<td>91.3</td>
</tr>
<tr>
<td>Full Model (90%)</td>
<td>98.1</td>
<td>92.3</td>
<td>94.7</td>
<td>92.4</td>
<td>95.3</td>
<td>93.5</td>
<td>96.5</td>
<td>95.8</td>
</tr>
<tr>
<td>Full Model (80%)</td>
<td>98.8</td>
<td>95.3</td>
<td>96.7</td>
<td>94.9</td>
<td>96.8</td>
<td>95.5</td>
<td>97.5</td>
<td>96.4</td>
</tr>
<tr>
<td>Full Model (70%)</td>
<td>99.2</td>
<td>96.8</td>
<td>97.9</td>
<td>96.4</td>
<td>97.5</td>
<td>96.8</td>
<td>97.9</td>
<td>97.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Carvn</th>
<th>Gate</th>
<th>Wall</th>
<th>Fence</th>
<th>Box</th>
<th>Sky</th>
<th>IoU</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA</td>
<td>81.5</td>
<td>46.8</td>
<td>62.1</td>
<td>60.3</td>
<td>49.4</td>
<td>83.1</td>
<td>82.1</td>
</tr>
<tr>
<td>LA+3D</td>
<td>90.2</td>
<td>68.8</td>
<td>74.6</td>
<td>74.0</td>
<td>63.7</td>
<td>83.4</td>
<td>86.2</td>
</tr>
<tr>
<td>LA+PW</td>
<td>85.9</td>
<td>48.7</td>
<td>66.2</td>
<td>64.9</td>
<td>54.5</td>
<td>86.6</td>
<td>84.4</td>
</tr>
<tr>
<td>LA+PW+CO</td>
<td>88.4</td>
<td>57.7</td>
<td>70.4</td>
<td>69.6</td>
<td>57.6</td>
<td>86.9</td>
<td>85.2</td>
</tr>
<tr>
<td>LA+PW+CO+3D</td>
<td>92.7</td>
<td>68.5</td>
<td>77.8</td>
<td>78.9</td>
<td>67.8</td>
<td>90.7</td>
<td>88.2</td>
</tr>
<tr>
<td>+ 3D PW</td>
<td>93.8</td>
<td>72.6</td>
<td>78.1</td>
<td>78.5</td>
<td>69.3</td>
<td>90.8</td>
<td>88.8</td>
</tr>
<tr>
<td>Full Model</td>
<td>93.9</td>
<td>72.6</td>
<td>78.4</td>
<td>78.6</td>
<td>69.4</td>
<td>90.8</td>
<td>89.0</td>
</tr>
<tr>
<td>Full Model (90%)</td>
<td>97.6</td>
<td>83.7</td>
<td>90.7</td>
<td>90.7</td>
<td>84.0</td>
<td>94.6</td>
<td>94.9</td>
</tr>
<tr>
<td>Full Model (80%)</td>
<td>98.5</td>
<td>86.4</td>
<td>93.7</td>
<td>93.4</td>
<td>87.9</td>
<td>96.4</td>
<td>96.6</td>
</tr>
<tr>
<td>Full Model (70%)</td>
<td>99.0</td>
<td>88.1</td>
<td>95.0</td>
<td>94.6</td>
<td>90.1</td>
<td>97.2</td>
<td>97.5</td>
</tr>
</tbody>
</table>

Table 4.2: Ablation study on semantic segmentation task. This table shows the importance of the different components in our model on all 160 images. The components are abbreviated as follows: LA = local appearance (\(p^P\)), PW = 2-dimensional pairwise constraints (\(\phi^{p,p}\)), CO = 3-dimensional primitive constraints (\(\xi^{p}\)), 3D = 3-dimensional points (\(\phi^{E,E}\)), 3D PW = 3-dimensional pairwise constraints (\(\phi^{E,E}\)), Full Model = all potentials including folds.
the geometry (“3D Primitives + GC”; “3D Mesh + GC”). Better results are obtained when projecting the visible 3-dimensional points followed by spatial propagation (“3D Points + GC”).

Finally, we observe that all baselines are outperformed by the proposed method, which populates the last row, in almost all categories. Importantly, note that the 2-dimensional methods require every 10th frame to be labeled, while our method—along with the other 3-dimensional baselines—requires 3-dimensional annotations in the form of primitives. Assuming 60 minutes annotation time per image, this amounts to 20 hours of annotation time per batch of 200 frames when labeling one 2-dimensional image every 10th frame, while the respective 3-dimensional annotations for this scene can be obtained in less than 3 hours. Note that labeling each frame of the sequence manually would require 200 hours. This gain multiplies with the frame rate and the number of cameras; our setup comprises four.
4.3.1.2 Ablation Study

We evaluate the importance of the individual components of our model in table 4.2 (top). Starting with the appearance classifier trained on the projected sparse 3-dimensional points \((p^3)\), we incrementally add the terms related to the 3-dimensional points \((\phi^3, \phi^p, \xi^3)\), the semantic pairwise term between pixels \((\phi^p, \phi^p)\), the 3-dimensional primitive constraints \((\xi^3)\), the 3-dimensional pairwise constraints \((\phi^{p-p})\) and finally the remaining terms \((\phi^F)\) as specified in eq. (4.1). We note that each component is able to increase performance. As expected, we obtain the largest improvement by reasoning about the relationship between points in 3 dimensions and pixels in the image. Integrating 3-dimensional fold and curb detections improves road boundaries slightly.

4.3.1.3 Semi-Dense Inference

Often, it is not necessary to label all pixels in every image for training a semantic segmentation model. In this section, we therefore leverage our model’s awareness of label uncertainty to estimate semi-dense label maps with high accuracy. To quantify uncertainty, we measure the entropy of the label marginal distribution at every pixel. Sorting all pixels according to their entropy allows us to predict the most certain regions in the image. Table 4.2 (bottom) and fig. 4.6 show our results when predicting only those parts of the image. Note how this helps to boost our performance to 94.9% IoU and 97.4% accuracy when predicting at 90% pixel density. In contrast, uncertainty is not directly accessible in most of the baseline models as they are deterministic or rely on MAP estimates. The only exception is the “Fully Connected CRF” baseline which we found to perform worse than the proposed method.

4.3.1.4 Instance Segmentation

As time consistent 2-dimensional instance ground truth is hard to obtain, most existing 2-dimensional label transfer methods focus on the semantic segmentation problem. Therefore, we chose to evaluate instance segmentation performance in an ablation study. We annotated the classes “building”, “car”, “trailer”, “caravan” and “box” with instances in our 2-dimensional ground truth. While the remaining classes (e.g., “road”, “sky”) do not admit unambiguous instance labels, we also report their performance as our model reasons about all instance and semantic classes jointly. Table 4.3 shows our results. Note how the instance segmentation results are on par with the semantic segmentations, demonstrating our model’s intra-class separation ability.
Table 4.3: Ablation study on instance segmentation task using the same abbreviations as in table 4.2.

4.3.2 Qualitative Evaluation

Figure 4.7 illustrates our dense inference results qualitatively for 6 different scenes in terms of semantic instance segmentation. The last row shows the error maps where colors indicate the true label. While the proposed method is able to delineate most object boundaries satisfyingly, some challenges remain. Errors occur in low-contrast image regions with overlapping 3-dimensional annotations (scene 1: car/road boundary) and in regions where 3-dimensional points are absent due to sensor occlusion (scene 4: building roof). Another source of errors are inherent label ambiguities which occur for porous objects such as fences or trees (scene 6: tree boundary) where even 2-dimensional ground truth annotation is a hard and ambiguous task. Finally, also manual 2-dimensional annotations contain errors, in particular at complex boundaries which are hard to delineate (scene 4: trees, scene 5: hedge). However, note that our semi-dense inference is able to successfully identify those regions as shown in fig. 4.6.
We presented a method for semantic instance labeling of large datasets from annotated 3-dimensional primitives. In the presence of 3-dimensional data, our method yields better results compared to several state-of-the-art 2-dimensional label transfer baselines while lowering annotation time. Furthermore, our method yields temporally consistent instance labels and explicitly exposes label uncertainty. We also proposed a novel dataset comprising 400k images, laser point clouds and annotations for all objects which we make publicly available. In future work, we plan to extend our method to dynamic scenes by joint inference over multiple frames.
PERMUTOHEDRAL CONVOLUTIONAL NEURAL NETWORKS

5.1 INTRODUCTION

The use of CNNs together with large annotated sets of training data has recently lead to significant performance increases in various computer vision domains. For problems including image classification (Krizhevsky, Sutskever, and Hinton, 2012), object detection (Girshick et al., 2014), and semantic segmentation (Long, Shelhamer, and Darrell, 2015; L.-C. Chen, Papandreou, et al., 2015; Farabet et al., 2013; Zheng et al., 2015) the use of learned feature representations over hand-crafted models lead to significantly improved performance and CNNs are currently the dominating approach.

When applied to images, a convolutional layer of a CNN is almost exclusively associated with a spatial convolution in the image domain apart from the notable exception of Bruna et al. (2013). In this thesis we study the class of bilateral filter convolutions and propose a general image adaptive convolution that can be learned from data. The bilateral filter (Aurich and Weule, 1995; Smith and Brady, 1997; Tomasi and Roberto, 1998) was originally introduced for the task of image denoising as an edge preserving filter. Since the bilateral filter contains the spatial convolution as a special case, we directly state the general case. Given an image $I = (I_1, \ldots, I_d), I_i \in \mathbb{R}^c$ with $d$ pixels with $c$ channels, and for every pixel $i$, a $k$-dimensional feature vector $z_i \in \mathbb{R}^k$, e.g., the $(x, y)$ position in the image $z_i = (x_i, y_i)^T$, a bilateral filter computes

$$I'_i = \sum_{j=1}^{n} k(z_i, z_j) I_j \quad \forall i.$$  \hfill (5.1)

Almost the entire literature refers to the bilateral filter as a synonym of the Gaussian parametric form $k(z_i, z_j) = \exp\left(-\frac{1}{2}(z_i - z_j)^T \Sigma^{-1} (z_i - z_j)\right)$. The features $z_i$ are most commonly chosen to be position $(x_i, y_i)$ and color $(r_i, g_i, b_i)$ or pixel intensity. To appreciate the edge-preserving effect of the bilateral filter, consider the 5-dimensional feature

$$z = (x, y, r, g, b)^T.$$  \hfill (5.2)
Two pixels $i$ and $j$ have a strong influence $k(z_i, z_j)$ on each other only if they are close in position and color. At edges the color changes, therefore pixels lying on opposite sides have low influence and thus this filter does not blur across edges. This behaviour is sometimes referred to as “image adaptive”, since the filter has a different shape when evaluated at different locations in the image. More precisely, it is the projection of the filter to the 2-dimensional image plane that changes; the filter values $k(z, z')$ do not change. The filter itself can be of $c$ dimensions, i.e., $k(z_i, z_j) \in \mathbb{R}^c$, in which case the multiplication in eq. (5.1) becomes an inner product. For the Gaussian case the filter can be applied independently per channel. For an excellent review of image filtering we refer to Milanfar (2013).

The filter operation of eq. (5.1) is a sparse high-dimensional convolution, a view advocated in Barash (2002) and Paris and Durand (2009). An image $I$ is not sparse in the spatial domain, we observe pixels values for all locations $(x, y)$. However, when pixels are understood in a higher dimensional feature space, e.g., $(x, y, r, g, b)$, the image becomes a sparse signal, since the $r, g, b$ values lie scattered in this 5-dimensional space. This view on filtering is the key difference of the bilateral filter compared to the common spatial convolution. An image edge is not “visible” for a filter in the spatial domain alone, whereas in the 5-dimensional space it is. The edge-preserving behaviour is possible due to the higher dimensional operation. Other data can naturally be understood as sparse signals, e.g., 3-dimensional surface points.

The contribution of this work is to propose a general and learnable sparse high dimensional convolution. Our technique builds on efficient algorithms that have been developed to approximate the Gaussian bilateral filter and reuses them for more general high-dimensional filter operations. Due to its practical importance (see related work in section 5.2) several efficient algorithms for computing eq. (5.1) have been developed, including the bilateral grid (Paris and Durand, 2009), Gaussian KD-trees (Adams, Gelfand, et al., 2009), and the permutohedral lattice (Adams, Baek, and Davis, 2010). The design goal for these algorithms was to provide a) fast runtimes and b) small approximation errors for the Gaussian filter case. The key insight of this work is to use the permutohedral lattice and use it not as an approximation of a predefined kernel but to freely parametrize its values. We relax the separable Gaussian filter case from Adams, Baek, and Davis (2010) and show how to compute gradients of the convolution (section 5.3) in lattice space. This enables learning the filter from data.
The main result is to demonstrate that the properties of the image adaptive bilateral filter can be carried over to convolutional architectures of CNNs and probabilistic graphical models. As we show, the class of densely connected graphical models for which mean field inference is tractable can be largely increased from Gaussian edge potentials to highly parameterized pairwise potential functions. The insight generalizes the result in Krähenbühl and Koltun (2011).

In this work we report on several contributions that are a consequence of the proposed general permutohedral lattice convolution:

- A learnable bilateral filtering operation for low-level vision tasks (section 5.4).
- A direct generalization of the result from Krähenbühl and Koltun (2011) from Gaussian edge potentials to more highly parameterized models (section 5.5).
- A generalization of the spatial convolutional layer in CNNs. The proposed convolution is able to process sparse features and has an image adaptive receptive field (section 5.6).

For all domains, we present various empirical evaluations with a wide range of applications.

5.2 RELATED WORK

We categorize the related work according to the three different generalizations of this work.

5.2.1 Image Adaptive Filtering

The literature on image adaptive filtering is rich and we can only provide a brief overview. Important classes of image adaptive filters include the bilateral filters (Aurich and Weule, 1995; Tomasi and Roberto, 1998; Smith and Brady, 1997), non-local means (Buades, Coll, and Morel, 2005; Awate and Whitaker, 2005), locally adaptive regressive kernels (Takeda, Farsiu, and Milanfar, 2007), guided image filters (He, J. Sun, and Tang, 2013) and propagation filters (Chang and Y.-C. F. Wang, 2015). The kernel least-squares regression problem can serve as a unified view of them (Milanfar, 2013). In contrast to the presented work that learns the filter kernel using supervised learn-
ing, all these filtering schemes use a predefined kernel. Because of the importance of the bilateral filtering to image processing applications, much effort has been devoted to derive fast algorithms; most notably Paris and Durand (2009), Adams, Baek, and Davis (2010), Adams, Gelfand, et al. (2009), and Gastal and Manuel M. Oliveira (2012). The only attempt to learn the bilateral filter we found is Hu and Haan (2007) that casts the learning problem in the spatial domain by rearranging pixels. However, the learned filter does not necessarily consider the full region of influence of a pixel as in the case of a bilateral filter. Barron and Poole (2015) and Barron, Adams, et al. (2015) propose to use the bilateral filter to regularize a large set of applications and the respective optimization problems. In these works the filters are part of a learning system but unlike this work restricted to be Gaussian.

5.2.2 Dense CRF

The key observation of Krähenbühl and Koltun (2011), that is described in section 2.3.4 and is applied in chapters 3 and 4, is that mean field inference update steps in densely connected CRFs with Gaussian edge potentials require Gaussian bilateral filtering operations. This enables tractable inference through the application of a fast filter implementation from Adams, Baek, and Davis (2010). This quickly found wide-spread use, e.g., the combination of CNNs with a dense CRF is among the best performing segmentation models (L.-C. Chen, Papandreou, et al., 2015; Zheng et al., 2015; Bell, Upchurch, et al., 2015). These works combine structured prediction frameworks on top of CNNs, to model the relationship between the desired output variables thereby improving upon the CNN result. Bilateral Neural Networks (BNNs), that are presented in this work, provide a principled framework for encoding the output relationship, using the feature transformation inside the network itself thereby alleviating some of the need for later processing. Chapter 3 and several works (Krähenbühl and Koltun, 2013; Domke, 2013; Zheng et al., 2015; Schwing and Urtasun, 2015) demonstrate how to learn free parameters of the dense CRF model. However, the parametric form of the pairwise term always remains a Gaussian. Campbell, Subr, and Kautz (2013) embed complex pixel dependencies into a Euclidean space and use a Gaussian filter for pairwise connections. This embedding is a pre-processing step and can not directly be learned. In section 5.5 we discuss how to learn the pairwise potentials, while retaining the efficient inference strategy of Krähenbühl and Koltun (2011).
5.2.3 CNNs

In recent years, the use of CNNs enabled tremendous progress in a wide range of computer vision applications. Most CNN architectures use spatial convolution layers, which have fixed local receptive fields. This work suggests to replace these layers with bilateral filters, which have a varying spatial receptive field depending on the image content. The equivalent representation of the filter in a higher dimensional space leads to sparse samples that are handled by a permutohedral lattice data structure. Similarly, Bruna et al. (2013) propose convolutions on irregularly sampled data. Their graph construction is closely related to the high-dimensional convolution that we propose and defines weights on local neighborhoods of nodes. However, the structure of the graph is bound to be fixed and it is not straightforward to add new samples. Furthermore, re-using the same filter among neighborhoods is only possible with their costly spectral construction. Both cases are handled naturally by our sparse convolution. Jaderberg et al. (2015) propose a spatial transformation of signals within the neural network to learn invariances for a given task. The work of Ionescu, Vantzos, and Sminchisescu (2015) propose matrix backpropagation techniques which can be used to build specialized structural layers such as normalized-cuts. Ben Graham (2015) propose extensions from 2-dimensional CNNs to 3-dimensional sparse signals. Our work enables sparse 3-dimensional filtering as a special case, since we use an algorithm that allows for even higher dimensional data.

5.3 LEARNING SPARSE HIGH DIMENSIONAL FILTERS

In this section, we describe the main technical contribution of this work, we generalize the permutohedral convolution (Adams, Baek, and Davis, 2010) and show how the filter can be learned from data.

Recall the form of the bilateral convolution from eq. (5.1). A naive implementation would compute for every pixel $i$ all associated filter values $k(z_i, z_j)$ for all $j$ and perform the summation independently. The view of $k$ as a linear filter in a higher dimensional space, as proposed by Paris and Durand (2009), opened the way for new algorithms. Here, we build on the permutohedral lattice convolution developed in Adams, Baek, and Davis (2010) for approximate Gaussian filtering. The most common application of bilateral filters use photometric features $(x, y, r, g, b)$. We chose the permutohedral lattice as it
Figure 5.1: The computational steps of a permutohedral convolution consist of splat, convolution and slice. Splat in (a) distributes the input points (red) onto the lattice corners (black). Each grid point (white circles) in (b) is the center of a convolution, which considers a $s = 2$ neighborhood around it. For reference we show a Gaussian filter with its values color coded in the inset. The general case has a free parameter per circle. The result of the convolution at the lattice corners (black) is projected back to the output points (blue) in (c) during the slice operation. Note that in general the output and input points may be different.
is particularly designed for this dimensionality, see fig. 7 in Adams, Baek, and Davis (2010) for a speed comparison.

5.3.1 Permutohedral Lattice Convolutions

We first review the permutohedral lattice convolution for Gaussian bilateral filters from Adams, Baek, and Davis (2010) and describe its most general form. As before, we assume that every image pixel $i$ is associated with a $d$-dimensional feature vector $\mathbf{z}_i$. We distinguish the signal that we want to blur from the actual image with $\mathbf{v}$ and $\mathbf{I}$ respectively. In the classical setup of bilateral convolution the two coincide. Gaussian bilateral filtering using a permutohedral lattice approximation involves three steps. We begin with an overview of the algorithm, then discuss each step in more detail in the next paragraphs. Figure 5.1 schematically shows the three operations for 2-dimensional features.

First, interpolate the image signal on the $d$-dimensional grid plane of the permutohedral lattice, which is called *splatting*. A permutohedral lattice is the tessellation of space into permutohedral simplices. In fig. 5.2, we visualize the permutohedral lattice in the image plane, where every simplex cell receives a different color. All pixels of the same lattice cell have the same color. Second, *convolve* the signal on the lattice. And third, retrieve the result by interpolating the signal at the $d$-dimensional output feature locations, called *slicing*. For example, if the used features are a combination of position and color $\mathbf{z}_i = (x_i, y_i, r_i, g_i, b_i)^T$, the input signal is mapped into the 5-dimensional cross product space of position and color and then convolved with a 5-dimensional tensor. The result is then mapped back to the original space. In practice we use a feature scaling $\mathbf{S} \mathbf{z}$ with a diagonal matrix $\mathbf{S}$ and use separate scales for pixel position and RGB color features. The scale determines the distance of points and thus the size of the lattice cells. More formally, we assume that the computation from eq. (5.1) has the following structure

$$\mathbf{v}' = \mathbf{S}_{\text{slic}} \mathbf{B} \mathbf{S}_{\text{splat}} \mathbf{v}, \quad \text{with}$$

$$k(\mathbf{z}_i, \mathbf{z}_j) = [\mathbf{S}_{\text{slic}} \mathbf{B} \mathbf{S}_{\text{splat}}]_{i,j}$$

(5.3)

and all involved matrices are defined below. For notational convenience we discuss scalar input signals $\mathbf{v} \in \mathbb{R}^d$ only. The vector-valued case per pixel can be handled analogously: the lattice convolution changes from scalar multiplications to inner products.

The permutohedral lattice (Adams, Baek, and Davis, 2010) is the result of the projection of the set $\mathbb{Z}^{\kappa+1}$—the set of all $\kappa + 1$-dimensional integer
Figure 5.2: Visualization of the permutohedral lattice. Given an input image in (a) we mark all pixels with the same color that belong to the same lattice cell for different feature spaces: 2-dimensional position features \((x, y)\) in (b), 3-dimensional color features \((r, g, b)\) in (c) and position and color features \((x, y, r, g, b)\) in (d). This visualization gives an impression of closeness and structure of the underlying lattice.

vectors—onto a plane defined by the normal vector \(1 \in \mathbb{R}^{\kappa+1}\). This \(\kappa\)-dimensional plane is embedded into \(\mathbb{R}^{\kappa+1}\). Figure 5.1 is an example of a lattice of two dimensions. The lattice points tessellate the subspace with regular cells. Given a point from the embedded space, it is efficient to find the enclosing simplex. We represent a sparse set of points from \(\mathbb{R}^\kappa\) with a sparse set of simplex corners in the lattice. Importantly, the number of corners does not grow exponentially but linearly with the dimension \(\kappa\) as it would for an axis-align simplex representation.

5.3.1.1 Splat

The splat operation (cf. fig. 5.1a) finds the enclosing simplex in \(O(\kappa^2)\) in the lattice of a given pixel feature \(z_i\) and distributes its value \(v_i\) onto the corners of the simplex. How strong a pixel contributes to a corner \(j\) is defined by its barycentric coordinate \(b_{i,j} \in \mathbb{R}\) inside the simplex. Thus, the value \(l_j \in \mathbb{R}\) at a lattice point \(j \in J_i\) is computed by summing over all enclosed input points. We define an index set \(J_i\) for a pixel \(i\), which contains all the lattice points \(j\) of the enclosing simplex

\[
l = S_{\text{splat}} v,
\]

\[
[S_{\text{splat}}]_{j,i} = \begin{cases} b_{i,j}, & \text{if } j \in J_i, \\ 0, & \text{otherwise}. \end{cases}
\]
Learning Sparse High Dimensional Filters

Figure 5.3: Construction and illustration of a 1-dimensional permutohedral lattice construction. The projection of the 2-dimensional $\mathbb{Z}^2$ space (blue and green dots) along $(1, 1)^T$ results in a 1-dimensional permutohedral lattice (red dots); $d = 1$. The number of elements of a filter with a neighborhood size of $s = 4$ can be identified with the number of grid points from $\mathbb{Z}^2$ with at least one of the coordinates equal to 0 (blue dots). Their number is equal to the number of elements in the hyper-cube ($\subset \mathbb{Z}^2$) with edge length $s + 1$ minus the number of elements in the hyper-cube ($\subset \mathbb{Z}^2$) with edge length $s$: $(s + 1)^d - s^d = 9$.

5.3.1.2 Convolve

The permutohedral convolution is defined on the lattice neighborhood $N_s(j)$ of lattice point $j$, e.g., only $s$ grid hops away,

$$l' = B l,$$

$$[B]_{j',j} = \begin{cases} w_{j',j}, & \text{if } j' \in N_s(j), \\ 0, & \text{otherwise}. \end{cases} \quad (5.5)$$

An illustration of a two-dimensional permutohedral filter is shown in fig. 5.1b. Note that we already presented the convolution in the general form. The work
of Adams, Baek, and Davis (2010) chooses the filter weights $w$ such that the resulting operation approximates a Gaussian blur,

$$k(z_i, z_j) = \left[ S_{\text{slice}} B(w) S_{\text{splat}} \right]_{i,j} \approx \exp\left( -\frac{\|z_i - z_j\|^2}{2} \right).$$

(5.6)

Further, the algorithm of Adams, Baek, and Davis (2010) takes advantage of the separability of the Gaussian kernel. Since we are interested in the most general case, we extended the convolution to include non-separable filters in $B$. However, it is possible to gain similar speed-ups as Adams, Baek, and Davis (2010) by learning a separable filter only.

5.3.1.3 Slice

The slice operation (cf. fig. 5.1c) computes an output value $v'_0$ for an output pixel $i'$ based on its barycentric coordinates $b_{i,j}$ and sums over the corner points $j$ of its lattice simplex

$$v' = S_{\text{slice}} v',

[S_{\text{slice}}]_{i,j} = \begin{cases} b_{i,j}, & \text{if } j \in J_i, \\ 0, & \text{otherwise}. \end{cases}$$

(5.7)

The splat and slice operations take a role of an interpolation between the different signal representations: the irregular and sparse distribution of pixels with their associated feature vectors and the regular structure of the permutohedral lattice points. Since high-dimensional spaces are usually sparse, performing the convolution densely on all lattice points is inefficient. So, for speed reasons, we keep track of the populated lattice points using a hash table and only convolve at those locations.

5.3.2 Learning Permutohedral Filters

The fixed set of filter weights $w$ from Adams, Baek, and Davis (2010) in eq. (5.5) are designed to approximate a Gaussian filter. However, the convolution weights $w$ can naturally be understood as a general filter operation in the permutohedral lattice space with free parameters. In the exposition above we already presented this general case. As we show in more detail later, this modification has non-trivial consequences for bilateral filters, probabilistic graphical models and CNNs.
The size of the neighborhood $N_s(k)$ for the blur in eq. (5.5) compares to the filter size of a spatial convolution and plays a crucial role in the time-complexity of the operation. The filtering kernel of a common spatial convolution that considers $s$ points to either side in all dimensions has $(2s + 1)^d \in \mathcal{O}(s^d)$ parameters. For example, considering $s = 1$ neighbors to either side of grid point in a 2-dimensional space results in a $3 \times 3$ grid, thus a filter size of 9. A comparable filter on the permutohedral lattice with an $s$ neighborhood is specified by

$$(s + 1)^d + 1 - s^d + 1 \in \mathcal{O}(s^d)$$

(5.8)

elements. Hence, both share the same asymptotic size. We give an illustrative proof for this filter size in fig. 5.3.

By computing the gradients of the filter elements $w$ we enable the use of gradient based optimizers, e.g., backpropagation for CNNs in the same way that spatial filters are learned. The gradients with respect to $v$ and the filter weights in $B$ of a scalar loss $\ell$ are:

$$\frac{\partial \ell}{\partial v} = S_{\text{splat}}^T B(w) S_{\text{slice}}^T \frac{\partial \ell}{\partial v'}$$

$$\frac{\partial \ell}{\partial [B(w)]_{i,j}} = \left[ S_{\text{slice}}^T \frac{\partial \ell}{\partial v} \right]_{i} [S_{\text{splat}} v]_{j}.$$  

(5.9)

Both gradients are needed during backpropagation for learning the filter weights $w$. The permutohedral lattice convolution is parallelizable, and scales linearly with the filter size. Specialized implementations run at interactive speeds in image processing applications (Adams, Baek, and Davis, 2010). Our implementation in the caffe deep learning framework (Jia et al., 2014) allows arbitrary filter parameters and the computation of the gradients on both CPU and GPU. The code is available online\(^1\).

### 5.4 Bilateral Filter Applications

In this section we consider three application that involve a single bilateral filter. Image denoising and body mesh denoising represent a traditional use-case for this filtering application. We show that performance over the fixed Gaussian case can be improved when the filter weights are learned from data. The problem of joint bilateral upsampling (Kopf et al., 2007) concludes this section with another prominent instance of an application with a single bilateral filter.

\(^1\) http://bilateralnn.is.tuebingen.mpg.de
102  Permutohedral Convolutional Neural Networks

<table>
<thead>
<tr>
<th>Method</th>
<th>Peak Signal-to-Noise Ratio (PSNR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noisy Input</td>
<td>20.17</td>
</tr>
<tr>
<td>Spatial</td>
<td>26.27</td>
</tr>
<tr>
<td>Gaussian Bilateral</td>
<td>26.51</td>
</tr>
<tr>
<td>Learned Bilateral</td>
<td>26.58</td>
</tr>
<tr>
<td>Spatial + Bilateral (Learned)</td>
<td>26.65</td>
</tr>
</tbody>
</table>

Table 5.1: PSNR results of a denoising task using the BSDS500 dataset (Arbeláez et al., 2011). A higher value indicates a better reconstruction.

5.4.1 Image Denoising

The main application that inspired the development of the bilateral filtering operation is image denoising (Aurich and Weule, 1995) using a single Gaussian kernel. Our development allows to learn this filter function from data and we explore how to improve over this baseline with a general bilateral filter.

We use the Berkeley segmentation dataset (BSDS500) (Arbeláez et al., 2011) as a test bed. The color images in the dataset are converted to gray-scale, and corrupted with Gaussian noise with a standard deviation of $\frac{25}{255}$. We compare the performance of four different filter models on the denoising task. The first baseline model (“Spatial” in table 5.1, 25 weights) uses a single spatial filter with a kernel size of 5 and predicts the scalar gray-scale value at the center pixel. The next model (“Gaussian Bilateral”) applies a bilateral Gaussian filter to the noisy input, using position and intensity features $z = (x, y, v)^T$. The third setup (“Learned Bilateral”, 65 weights) takes a Gaussian kernel as initialization and fits all filter weights on the “train” image set to minimize the mean squared error with respect to the clean images. We run a combination of spatial and permutohedral convolutions on spatial and bilateral features (“Spatial + Bilateral (Learned)”) to check for a complementary performance of the two convolutions.

The PSNR scores evaluated on full images of the “test” image set are shown in table 5.1. We find that an untrained bilateral filter already performs better than a trained spatial convolution (26.27 compared to 26.51). A learned convolution further improves the performance slightly. We choose this simple one-kernel setup to validate an advantage of the generalized bilateral filter. A competitive denoising system would employ RGB color information and also
5.4 Bilateral Filter Applications

5.4.1 Visualization of 4-dimensional isomap features for a sample 3-dimensional mesh. Isomap feature values are overlaid onto mesh vertexes and each figure represents one of the coordinates.

needs to be properly adjusted in network size. Multi-layer perceptrons have obtained state-of-the-art denoising results (Burger, C. J. Schuler, and Harmeling, 2012) and the permutohedral lattice layer can readily be used in such an architecture, which is intended future work.

5.4.2 3-Dimensional Mesh Denoising

Permutohedral convolutions can naturally be extended to higher (> 2) dimensional data. To highlight this, we use the proposed convolution for the task of denoising 3-dimensional meshes.

We sample 3-dimensional human body meshes using a generative body model from Loper et al. (2015) and add Gaussian random noise displacements to the clean meshes along the surface normal at each vertex location. Figure 5.5 shows a sample mesh and corresponding noisy mesh. The task is to take the noisy meshes as inputs and recover the original 3-dimensional body meshes. We create 1000 training, 200 validation and another 500 testing examples for the experiments.

5.4.2.1 Mesh Representation

The 3-dimensional human body meshes from Loper et al. (2015) are represented with 3-dimensional vertex locations and the edge connections between
### Table 5.2: Vertex distance RMSE Values and Normal Angle Error (in degrees) Corresponding to Different Body-Mesh Denoising Strategies Averaged Over 500 Test Meshes.

<table>
<thead>
<tr>
<th>Mesh Type</th>
<th>Vertex Distance (RMSE)</th>
<th>Normal Angle Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noisy Mesh</td>
<td>5.774</td>
<td>19.680</td>
</tr>
<tr>
<td>Normal Smoothing</td>
<td>3.183</td>
<td>19.707</td>
</tr>
<tr>
<td>Gaussian Bilateral</td>
<td>2.872</td>
<td>19.357</td>
</tr>
<tr>
<td>Learned Bilateral</td>
<td>2.825</td>
<td>19.207</td>
</tr>
</tbody>
</table>

The vertexes. We found that this signal representation using global 3-dimensional coordinates is not suitable for denoising with a bilateral filter. Therefore, we first smooth the noisy mesh using mean smoothing applied to the face normals (Yagou, Ohtake, and Belyaev, 2002) and represent the noisy mesh vertexes as 3-dimensional vector displacements with respect to the corresponding smoothed mesh. Thus, the task becomes denoising the 3-dimensional vector displacements with respect to the smoothed mesh.

#### 5.4.2.2 Isomap Features

To apply a permutohedral convolution, we need to define features at each input vertex point. For this we use a 4-dimensional isomap embedding (Tenenbaum, de Silva, and Langford, 2000) of the given 3-dimensional mesh as features. The mesh is converted into a weighted edge graph by setting the edge weights to the Euclidean distance between the connected vertexes and to infinity otherwise. A 4-dimensional isomap embedding is computed for this graph using the publicly available implementation from Tenenbaum, de Silva, and Langford (2000) and fig. 5.4 shows the visualization of the isomap features on a sample mesh.

#### 5.4.2.3 Experimental Results

Mesh denoising with a bilateral filter proceeds by splatting the 3-dimensional input displacement vectors into the 4-dimensional isomap feature space, filtering the signal in this 4-dimensional space and then slicing it back into the original 3-dimensional input space. Table 5.2 shows quantitative results as RMSE for different denoising strategies. The normal smoothing (Yagou, Ohtake, and Belyaev, 2002) already reduces the RMSE. The Gaussian bilateral filter re-
results in another improvement over normal smoothing and learning the filter weights improves on top of that. A visual result is shown in fig. 5.5.

5.4.3 Joint Bilateral Upsampling

A typical technique to speed up computer vision algorithms is to compute results on a lower scale and upsample the result to the full resolution. This upsampling step may use the original resolution image as a guidance image. A joint bilateral upsampling approach for this problem setting was developed in Kopf et al. (2007). We describe the procedure for the example of upsampling a color image: given a high resolution gray scale image—the guidance image—and a low resolution color version, the task is to upsample the color image to the same resolution as the guidance image. Using the permutohedral lattice,
### Joint Bilateral Upsampling

<table>
<thead>
<tr>
<th>Upsampling factor</th>
<th>Bicubic</th>
<th>Gaussian</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color Upsampling (PSNR) on PASCAL VOC 2012</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2×</td>
<td>24.19</td>
<td>33.46</td>
<td>34.05</td>
</tr>
<tr>
<td>4×</td>
<td>20.34</td>
<td>31.87</td>
<td>32.28</td>
</tr>
<tr>
<td>8×</td>
<td>17.99</td>
<td>30.51</td>
<td>30.81</td>
</tr>
<tr>
<td>16×</td>
<td>16.10</td>
<td>29.19</td>
<td>29.52</td>
</tr>
<tr>
<td><strong>Color Upsampling (PSNR) on high-res. 2MP</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2×</td>
<td>30.59</td>
<td>37.93</td>
<td>38.74</td>
</tr>
<tr>
<td>4×</td>
<td>25.28</td>
<td>35.66</td>
<td>36.38</td>
</tr>
<tr>
<td>8×</td>
<td>22.12</td>
<td>33.92</td>
<td>34.41</td>
</tr>
<tr>
<td>16×</td>
<td>19.80</td>
<td>32.24</td>
<td>32.75</td>
</tr>
<tr>
<td><strong>Depth Upsampling (RMSE)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8×</td>
<td>0.753</td>
<td>0.753</td>
<td>0.748</td>
</tr>
</tbody>
</table>

Table 5.3: Joint bilateral upsampling. PSNR values corresponding to various upsampling factors and upsampling strategies on the test images of the PASCAL VOC 2012 dataset (top) / high-resolution 2MP dataset (middle). A prediction that is closer to the ground-truth image results in a higher PSNR value. RMSE values corresponding to upsampling depth images that are estimated using Eigen, Puhrsch, and Fergus (2014) computed on the test images from the NYU depth dataset from Silberman et al. (2012) (bottom).

Joint bilateral upsampling proceeds by splatting the color image into the lattice, using the 2-dimensional position and 1-dimensional intensity as features for the 3-dimensional RGB signal. We apply a convolution in the lattice and read out the result at the pixels of the high resolution image, i.e., using the 2-dimensional position and intensity of the guidance image. The possibility of reading out points that are not necessarily the input points is an appealing feature of the permutohedral lattice convolution.

### 5.4.3.1 Color Upsampling

For the task of color upsampling, we compare the Gaussian bilateral filter (Kopf et al., 2007) against a learned generalized filter. We experiment with
Figure 5.6: Guided color upsampling results with a scale factor of 8. A bicubic interpolation, a Gaussian bilateral filter and a learned bilateral filter use a down-scaled version (a) and (g) as input to produce a high resolution output. The latter two methods additionally use a gray-scaled version of the original image (b) and (h) as guidance. We show the expected original images in (c) and (i). Both, the Gaussian bilateral filter and the learned bilateral filter profit from the side information and produce images that show more details. It is especially the low level image details that are better preserved with a learned bilateral filter compared to the Gaussian case in (f) and (l).
two different datasets: PASCAL VOC 2012 (Everingham et al., 2010) using training, validation and test splits, and 200 higher resolution (2MP) images from Google image search\(^2\) with 100 training, 50 validation and 50 test images. For training we use the Mean Squared Error (MSE) criterion and perform stochastic gradient descent with a momentum term of 0.9, and weight decay of 0.0005, found using the validation set. In table 5.3 we report results in terms of PSNR for the upsampling factors 2×, 4×, 8× and 16×. We compare a standard bicubic interpolation, that does not use a guidance image, the Gaussian bilateral filter and the learned filter. All filters have the same support and we set the feature scales on the validation set. For all factors, joint bilateral Gaussian upsampling outperforms bicubic interpolation. The learned filter further improves on top of this and visually recovers finer details in the images as can be seen in fig. 5.6. Additionally, we perform a cross-factor analysis of training and testing at different upsampling factors. Table 5.4 shows the PSNR results for this analysis. Although, in terms of PSNR, these results suggest to train and test at the same upsampling factor, the differences are small when training and testing upsampling factors are different.

### 5.4.3.2 Depth Upsampling

The work of Eigen, Puhrsch, and Fergus (2014) predicts depth estimates from single RGB images and we use their results as another joint upsampling task. Our testbed is the dataset of Silberman et al. (2012) that comes with predefined

---

2 [http://images.google.com](http://images.google.com)
Figure 5.7: Guided depth upsampling by a scale factor of 8. A bicubic interpolation, a Gaussian bilateral filter and a learned bilateral filter process a low-resolution depth-estimate given in (a) and (g) as input to produce a high resolution prediction. The the original image in (b) and (h) helps the bilateral filter methods as guidance for the upsampling process. We show the measured depth in (c) and (i). Compared to the bicubic interpolation that can be seen in (d) and (j), the Gaussian bilateral filter (e) and (k), and the learned bilateral filter in reproduce greater details and snap to edges in (f) and (l).
Table 5.5: Mean field inference with learned potential function for semantic segmentation on PASCAL VOC 2012 (Everingham et al., 2010). We show the improvement over the unary performance of the CNN from L.-C. Chen, Papandreou, et al. (2015). The base model achieves an IoU score of 72.08 on the validation set and 66.95 on the test set.

<table>
<thead>
<tr>
<th></th>
<th>MF-1 step</th>
<th>MF-2 steps</th>
<th>loose MF-2 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASCAL VOC 2012 Validation Set (base model: 72.08)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian CRF</td>
<td>+2.48</td>
<td>+3.38</td>
<td>+3.38</td>
</tr>
<tr>
<td>Learned CRF</td>
<td>+2.93</td>
<td>+3.71</td>
<td>+3.85</td>
</tr>
<tr>
<td>PASCAL VOC 2012 Test Set (base model: 66.95)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian CRF</td>
<td></td>
<td></td>
<td>+3.00</td>
</tr>
<tr>
<td>Learned CRF</td>
<td></td>
<td></td>
<td>+3.37</td>
</tr>
</tbody>
</table>

training, validation and test splits. The approach of Eigen, Puhrsch, and Fergus (2014) is a CNN model that produces a result at 1/4th of the input resolution due to down-sampling operations in max-pooling layers. Furthermore, the authors reduce the image size of 640 × 480 to 320 × 240 as a preprocessing step before applying CNN convolutions. The final depth result is interpolated to the original resolution using a bicubic approximation. We aim to replace this interpolation with a Gaussian and learned bilateral filter that take 5-dimensional position and color features as inputs. The filter is trained using the same protocol as for the color upsampling task—minimizing the MSE prediction error. The quantitative results are shown in table 5.3, which conclude that the Gaussian filter performs equally to the bicubic interpolation (p-value 0.311). However, training the filter leads to an improvement (p-value 0.015). Visually both joint bilateral upsampling techniques respect images edges (cf. fig. 5.7). We expect larger gains from a combination of specialized methods (Ferstl, Ruether, and Bischof, 2015; Zeiler et al., 2010) with a generalized permutohedral convolution as part of future work.

5.5 Learning pairwise potentials in dense CRFs

The Gaussian bilateral filter is an essential building block for inference in dense CRF models as introduced in section 2.3.4 and employed in chapters 3 and 4 that have found wide-spread use in computer vision applications (D. Sun et al.,
5.5 Learning Pairwise Potentials in Dense CRFs

The following paragraphs demonstrate how the general permutohedral convolution extends the modeling flexibility of dense CRF inference.

5.5.1 Learning Pairwise Potentials

The presented generalized bilateral convolution augments the class of potential function that allow for efficient inference. The proposed bilateral convolution generalizes the class of potential functions $\phi$, since they allow a richer class of kernels $k(z_i, z_j)$ that furthermore can be learned from data. So far, all dense CRF models have used Gaussian potential functions $\phi$. We replace it with the general bilateral convolution and learn the blur parameters $\mathbf{w}$; thus in effect learn the pairwise potentials of the dense CRF. This retains the desirable properties of this model class—efficient inference through mean field and the feature dependency of the pairwise potential. In order to learn the form of the pairwise potentials $\phi$ we use the gradients for the filter parameters as has been presented in section 5.3.2 and apply back-propagation through the mean field iterations as in Domke (2013) and Y. Li and Zemel (2014) and section 3.3.1.

The work of Krähenbühl and Koltun (2013) derived gradients to learn the feature scaling $\mathbf{z}$ but not the form of the kernel $k$, which still was Gaussian. In Campbell, Subr, and Kautz (2013), the features $z_i$ were derived using a non-parametric embedding into a Euclidean space and again a Gaussian kernel was used. The computation of the embedding was a preprocessing step, not integrated in an end-to-end learning framework. Both aforementioned works are generalizations that are orthogonal to our development and can be used in conjunction with the general permutohedral convolution.

5.5.2 Experimental Evaluation

We evaluate the effect of learning more general forms of potential functions on two pixel labeling tasks: semantic segmentation of PASCAL VOC 2012 (Everingham et al., 2010) and material classification (Bell, Upchurch, et al., 2015). We use pretrained models from the literature and compare the relative change when learning the pairwise potential functions, as in the last section 5.4. For both the experiments, we use the univariate logistic loss (cf. eq. (2.48)) and learn the filters via back-propagation (Domke, 2013).
Figure 5.8: Inference results for the semantic segmentation task on the PASCAL VOC 2012 dataset (Everingham et al., 2010). The images in (a) show the input image from the test-set with their ground truth annotation in (b). An additional “do not care” region at the boundaries of the objects does not influence the measured accuracy of a prediction. Column (c) and (d) depict the class predictions before and after the application of the learned loose mean field inference in the CRF that runs on top of the CNN.

5.5.2.1 Semantic Segmentation

We choose the DeepLab network (L.-C. Chen, Papandreou, et al., 2015), which is a variant of the VGGNet (Simonyan and Zisserman, 2014), to obtain unary predictions for semantic segmentation. The output of the DeepLab architecture is a down-sampled version of the input image by a factor of 8 due to the intermediate pooling layers inside the network. The result is scaled to the desired resolution using a bilinear interpolation, which serves as unary potentials $\Phi_u(x_i)$ in a dense CRF. We use the same Pott’s label compatibility function $\mu$, and also use two kernels $k_1(z_i, z_j) + k_2(p_i, p_j)$ with the same features $z_1 = (x, y, r, g, b)^T$ and $z_2 = (x, y)^T$ as in L.-C. Chen, Papandreou, et al. (2015). Thus, there are two filters that operate in parallel in the color and position domain and in the spatial domain. This allows us to verify that
5.5 Learning Pairwise Potentials in Dense CRFs

<table>
<thead>
<tr>
<th></th>
<th>MF-1 step</th>
<th>MF-2 steps</th>
<th>loose MF-2 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average Accuracy over Classes (base model: 67.21)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian CRF</td>
<td>+6.28</td>
<td>+7.35</td>
<td>+7.35</td>
</tr>
<tr>
<td>Learned CRF</td>
<td>+6.23</td>
<td>+6.93</td>
<td>+6.93</td>
</tr>
<tr>
<td><strong>Average Accuracy over all Pixel (base model: 69.23)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian CRF</td>
<td>+7.91</td>
<td>+9.68</td>
<td>+9.68</td>
</tr>
<tr>
<td>Learned CRF</td>
<td>+9.48</td>
<td>+11.89</td>
<td>+11.91</td>
</tr>
</tbody>
</table>

Table 5.6: Mean field inference with learned potential functions for material segmentation on MINC (Bell, Upchurch, et al., 2015). The table includes the improvement over the baseline of the unary CNN classifier with its accuracy of 67.21 and 69.23 averaged over all pixels and classes respectively.

the change in performance is only caused by the change in potential functions from a Gaussian filter to a trained filter.

We evaluate the result after 1 step and 2 steps of mean field inference and compare the Gaussian filter to the learned version (cf. table 5.6). First, as in L.-C. Chen, Papandreou, et al. (2015) we observe that one step of mean field improves the performance by 2.48% in IoU score. However, a learned potential increases the score by 2.93%. The same behaviour is observed for 2 steps: the learned result again adds on top of the Gaussian mean field performance. Further, we tested a variant of the mean field model that learns a separate kernel for the first and second step (Y. Li and Zemel, 2014). This “loose” mean field model leads to further improvement of the performance. It is not obvious how to take advantage of a loose model in the case of Gaussian potentials. We present visuals in fig. 5.8 and observe that results for the learned CRF potentials lead to slightly dilated classification regions in comparison to a dense Gaussian CRF. We conjecture that the learned potential functions fill-in pixels that have been marked as false negatives with respect to a semantic class.

5.5.2.2 Material Segmentation

We adopt the method and dataset from Bell, Upchurch, et al. (2015) for material segmentation. Their approach proposes the same architecture as in the previous section: a CNN to predict material labels (e.g., “wool”, “glass”, “sky”, etc.) followed by a densely connected CRF using Gaussian potentials and mean field inference. We re-use the pre-trained CNN and choose the CRF parameters
Figure 5.9: Material segmentation results on the test-set of the material segmentation dataset from Bell, Upchurch, et al. (2015). Column (a) shows the given input images of the task with their expected annotation next to it in (b). The inference results of the CNN architecture are depicted in (c) and the final predictions of the trained loose mean field inference of the CRF populate column (d).
and Lab color/position features as in Bell, Upchurch, et al. (2015). Results for pixel accuracy and class-averaged pixel accuracy are shown in table 5.6. Following the CRF validation in Bell, Upchurch, et al. (2015), we ignore the label “other” for both the training and evaluation. For this dataset, the availability of training data is small and there are 928 images with sparse segment annotations. While this is enough to cross-validate hyper-parameters, we would expect the general bilateral convolution to benefit from more training data. Figure 5.9 shows a comparison between the unary prediction and the result of the mean field inference along with the input image and ground truth annotation. Similar to the results for semantic segmentation in the previous section 5.5.2.1 we observe that the dominant class receives a higher attention by the CRF. This leads to significant improvements in overall pixel accuracy. However, this also results in a slight decrease of the accuracy from less frequent class pixels, which appear in only a handful images—thereby reducing the average class accuracy with learning.

5.6 BNNS

Probably the most promising opportunity for the generalized bilateral filter is its use in CNNs. The lifted restriction on Gaussian filters allows us to run stacked filters in the same way as filters in layers in typical spatial CNN architectures. We refer to the layers with bilateral filters as Bilateral Convolution Layers (BCLs). As discussed in the introduction, these can be understood as either linear filters in a high dimensional space or a filter with an image adaptive receptive field. In the remainder we refer to CNNs that include at least one bilateral convolutional layer as a Bilateral Neural Network (BNN).

What are the novel possibilities of a BCL compared to a standard spatial layer? First, we gain additional modeling flexibility from the freedom to choose the feature space $z_i \in \mathbb{R}^k$ that controls the proximity of elements with respect to the convolution. In particular, color or intensity features are an interesting way to encode additional domain knowledge directly into the CNN architecture. Second, a runtime comparison between our implementation of a BCL and the caffe (Jia et al., 2014) implementation of a $\kappa$-dimensional convolution that is included in table 5.7 reveals the benefits of the permutohedral data structure. For 2-dimensional positional features, the standard layer is still faster since the permutohedral algorithm comes with an overhead, but already for dimensions higher than 2, ignoring the sparsity is quickly leading to intractable runtimes. However, the speed-up does depend on the underly-
Table 5.7: Average CPU/GPU runtime (in ms) comparison between a Bilateral Convolution Layer (BCL) and a spatial convolution in caffe (Jia et al., 2014). We run 50 filters with neighborhood size 1 averaged over 1000 images from PASCAL VOC 2012 (Everingham et al., 2010) with different features sets. A BCL includes splatting and splicing operations which in layered networks can be re-used if other BCL layers agree on the same feature configuration.

<table>
<thead>
<tr>
<th>Feature Dimension</th>
<th>Dense Convolution Caffe</th>
<th>BCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-dimensional-(x, y)</td>
<td>3.3 ± 0.3</td>
<td>4.8 ± 0.5</td>
</tr>
<tr>
<td>3-dimensional-(r, g, b)</td>
<td>364.5 ± 43.2</td>
<td>5.1 ± 0.7</td>
</tr>
<tr>
<td>4-dimensional-(x, r, g, b)</td>
<td>30741.8 ± 9170.9</td>
<td>6.2 ± 0.7</td>
</tr>
<tr>
<td>5-dimensional-(x, y, r, g, b)</td>
<td>out of memory</td>
<td>7.6 ± 0.4</td>
</tr>
<tr>
<td>GPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-dimensional-(x, y)</td>
<td>0.5 ± 0.1</td>
<td>2.8 ± 0.4</td>
</tr>
<tr>
<td>3-dimensional-(r, g, b)</td>
<td>12.1 ± 0.4</td>
<td>3.2 ± 0.4</td>
</tr>
<tr>
<td>4-dimensional-(x, r, g, b)</td>
<td>1446.2 ± 304.7</td>
<td>3.8 ± 0.5</td>
</tr>
<tr>
<td>5-dimensional-(x, y, r, g, b)</td>
<td>out of memory</td>
<td>4.5 ± 0.4</td>
</tr>
</tbody>
</table>

Next we illustrate two use cases of BNNs and compare against their spatial counterparts.

5.6.1 Recognition of subsampled MNIST

One of the strengths of the proposed convolution is that it does not require the input to lie on a regular grid. The only requirement is to define a distance between features of the input signal. We highlight this feature with the following experiment using the classical MNIST ten class classification problem from LeCun et al. (1998). We sample a sparse set of N points (x, y) ∈ [0, 1] × [0, 1] uniformly at random in the input image, use their interpolated...
Table 5.8: Classification accuracy on MNIST. We compare the LeNet (LeCun et al., 1998) implementation that is part of caffe (Jia et al., 2014) to the network with the first layer replaced by a BCL. Both are trained on the original image resolution (first two rows). Three more BNN and CNN models are trained with randomly subsampled images (100%, 60% and 20% of the pixels). An additional bilinear interpolation layer samples the input signal on a spatial grid for the CNN model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Original</th>
<th>100%</th>
<th>60%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>0.9919</td>
<td>0.9660</td>
<td>0.9348</td>
<td>0.6434</td>
</tr>
<tr>
<td>BNN</td>
<td>0.9903</td>
<td>0.9844</td>
<td>0.9534</td>
<td>0.5767</td>
</tr>
<tr>
<td>LeNet 100%</td>
<td>0.9856</td>
<td>0.9809</td>
<td>0.9678</td>
<td>0.7386</td>
</tr>
<tr>
<td>BNN 100%</td>
<td>0.9900</td>
<td>0.9863</td>
<td>0.9699</td>
<td>0.6910</td>
</tr>
<tr>
<td>LeNet 60%</td>
<td>0.9848</td>
<td>0.9821</td>
<td>0.9740</td>
<td>0.8151</td>
</tr>
<tr>
<td>BNN 60%</td>
<td>0.9885</td>
<td>0.9864</td>
<td>0.9771</td>
<td>0.8214</td>
</tr>
<tr>
<td>LeNet 20%</td>
<td>0.9763</td>
<td>0.9754</td>
<td>0.9695</td>
<td>0.8928</td>
</tr>
<tr>
<td>BNN 20%</td>
<td>0.9728</td>
<td>0.9735</td>
<td>0.9701</td>
<td>0.9042</td>
</tr>
</tbody>
</table>

values as signal and the continuous (x, y) positions as features. This mimics sub-sampling of a high-dimensional signal. To compare against a spatial convolution, we interpolate the sparse set of values at grid positions with a bilinear approximation.

We take a reference implementation of LeNet (LeCun et al., 1998) that is part of the caffe project (Jia et al., 2014) and compare it against the same architecture but replacing the first convolutional layer with a BCL. The filter size and numbers are adjusted to get a comparable number of parameters (5 × 5 for LeNet, 2-neighborhood for BCL).

The results are shown in table 5.8. We see that training on the original MNIST data (column “Original”, LeNet compared to BNN) leads to a slight decrease in performance of the BNN (99.03%) compared to LeNet (99.19%). The BNN can be trained and evaluated on sparse signals, and we resample the image as described above for N = 100%, 60% and 20% of the total number of pixels. The methods are also evaluated on test images that are subsampled in the same way. Note that we can train and test with different subsampling
rates. In essence, both models perform a spatial interpolation and thus we expect them to yield a similar classification accuracy. Once the data is of higher dimensions the permutohedral convolution is faster due to hashing the sparse input points, as well as less memory demanding in comparison to a naive application of a spatial convolution with interpolated values.

5.6.2 Character Recognition

The results for semantic and material segmentation in section 5.5 mainly profit from the feature space that was used to encode useful prior information about the problem, i.e., close-by pixels with similar color values are likely to have the same label. However—as seen in the MNIST example in section 5.6.1—input signals may also be sparse to begin with. We turn to handwritten character recognition to illustrate this idea.

The Assamese character dataset (Lichman, 2013) contains 183 different Indo-Aryan symbols with 45 writing samples per class. Figure 5.10 shows some examples. This dataset has been collected on a tablet PC using a pen input device and has been pre-processed to binary images of size $96 \times 96$. Only around 3% of the pixels are foreground and contain a pen stroke, which makes this dataset inherently sparse.

A CNN is a natural choice to approach this character recognition. We experiment with two CNN architectures for this experiment that have been used for this task: LeNet-7 from LeCun et al. (1998) and DeepCNet (Ciresan, Meier, and Schmidhuber, 2012; Benjamin Graham, 2014). LeNet is a shallower network with bigger filter sizes whereas DeepCNet is deeper with smaller convolutions. In order to simplify the task for the networks we crop the characters by placing a tight bounding box around them and providing the bounding box information as input to the networks. We call these networks Crop-LeNet, respectively Crop-DeepCNet. For training, we randomly divide the data into

Figure 5.10: Sample Assamese character images from Lichman (2013) show 9 classes (from left to right) with 2 samples each (top and bottom).
<table>
<thead>
<tr>
<th></th>
<th>LeNet</th>
<th>Crop-LeNet</th>
<th>BNN-LeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>59.29</td>
<td>68.67</td>
<td>75.05</td>
</tr>
<tr>
<td>Test</td>
<td>55.74</td>
<td>69.10</td>
<td>74.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>DeepCNet</th>
<th>Crop-DeepCNet</th>
<th>BNN-DeepCNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>82.24</td>
<td>81.88</td>
<td>84.15</td>
</tr>
<tr>
<td>Test</td>
<td>79.78</td>
<td>80.02</td>
<td>84.21</td>
</tr>
</tbody>
</table>

Table 5.9: Total recognition accuracy for the different models for the task of Assamese character recognition from Lichman (2013).

30 writers for training, 6 writers for validation and the remaining 9 writers for test.

The input is spatially sparse and the BCL provides a natural way to take advantage of this. For both networks we create a BNN variant—BNN-LeNet and BNN-DeepCNet—by replacing the first layer with a bilateral convolutions using the features $z_i = (x_i, y_i)^T$ and only consider foreground pixels with an intensity of 1. The values $(x_i, y_i)$ denote the position of the pixel with respect to the top-left corner of the bounding box around the character. The sparseness of the input signal reduces the runtime as the convolution is only performed on 3% of the points that are actually marked by a pen. Our bilateral filter has 7 parameters compared to a receptive field of $3 \times 3$ for the first DeepCNet layer and $5 \times 5$ for the first LeNet layer. Thus, a BCL with the same number of filters has fewer parameters. The result of the BCL convolution is then sliced at all points $(x_i, y_i)$ and passed to the remaining spatial layers. The empirical results of this experiment for all tested architectures are summarized in table 5.9, with the BNN variants outperforming their spatial counterparts.

The absolute results can be vastly improved by making use of virtual examples, e.g., by affine transformations (Benjamin Graham, 2014). The purpose of these experiments is to compare the networks on equal grounds while we believe that additional data is beneficial for both networks. It is unlikely that a particular network benefits more from it.
5.7 CONCLUSION

We propose to learn bilateral filters from data. It may appear obvious that this leads to performance improvements compared to a fixed parametric form, e.g., the Gaussian. To understand algorithms that facilitate fast approximate computation of eq. (5.1) as a parameterized implementation of a bilateral filter with free parameters is the key insight and enables gradient descent based learning. We relax the functional form in the algorithm from Adams, Baek, and Davis (2010) to allow for more general filters. There is a wide range of possible applications for learned bilateral filters (Paris, Kornprobst, et al., 2009) and we discuss some applications to previous work. These include joint bilateral upsampling and mean field inference in dense CRFs. We further demonstrated two use cases of bilateral convolutions in CNNs.

The Bilateral Convolution Layer (BCL) allows for filters with a receptive field that changes given the features of an input image. The feature space view provides a canonical way to encode similarity between any kind of objects; not only pixels, but, e.g., bounding boxes, segmentations, surfaces or super-pixels. The proposed filtering operation is then a natural candidate to define a convolution on these objects. It naturally takes advantage of sparsity and scales to higher dimensions. Therefore, we believe that this view will be useful for future problems where CNNs can be applied. An open research problem is whether the sparse higher dimensional structure also allows for efficient or compact representations for intermediate layers inside CNN architectures.
CONCLUSION

This thesis develops structured prediction methods based on the permutohedral lattice. At the core of all the presented inference mechanisms is a convolution that operates on a discrete permutohedral grid that handles sparse high dimensional data. Over the course of this work we present novel applications and generalizations over the works from Krähenbühl and Koltun (2011) and Adams, Baek, and Davis (2010).

In chapter 3, we introduce the FoP model. It allows to encode structural domain knowledge from human pose estimation into a dense graphical model. We represent each body part with a field of binary random variables and add dense pairwise connection between the fields to mimic the dependencies between the body parts. It is one of the strength of this model to be able to express similarity in a user-defined feature space, e.g., spatial locations in the image for the parts. The permutohedral convolution allows for fast approximate inference that we train end-to-end with a problem specific loss and propagate the error gradient through the inference steps. Additionally, we present three extensions to the FoP model: the FoP-Bilateral model introduces color-dependent pairwise terms that propagate information about the background, the FoP-Loopy model inserts more pairwise connection into the body tree to recover more statistics of the underlying probability distribution and the FoP-Segmentation model adopts segmentation information to drive predictions. All model variations improve the \( \text{APK} \) evaluation score. Compared to the established PS model from Yang and Ramanan (2013) the FoP model achieves a boost in performance on equal grounds.

Chapter 4 picks up the idea of densely-connected fields of random variables in graphical models; we use the same inference and learning methods from chapter 3 to transfer semantic annotations from 3-dimensional point clouds to 2-dimensional RGB video frames. It shows the advantages of the permutohedral convolution to enable inference in models that encode the dense relation between the 3-dimensional world and 2-dimensional video frames. In our baseline experiments we support our claim that a naive projection of the 3-dimensional point cloud data to the image plane leads to a weaker performance and that the probabilistic reasoning results in strong marginal predic-
tions. We create a large set of ground truth image annotations with 400k images and 100k laser scans from sub-urban street scenes with semantic and instance labels on pixel level. Additionally, we leverage latent fold and curb modeling to improve the accuracy for road boundaries in particular. This model ensures instance consistency and clear object boundaries, along with faster labeling times. All labels come with a marginal probability measure that is useful in subsequent reasoning. Furthermore, applications can profit from the tight integration of 3-dimensional information that comes with the novel dataset.

In chapter 5 we broaden the class of convolutional kernels used in bilateral filters, which have been successfully applied in the previous chapters. This removes the restriction to Gaussian convolutional kernels that are the predominant operation in bilateral filters. Instead of using the permutohedral data structure to approximate the Gaussian convolution as well as possible, we understand the operation in the lattice itself as the definition of the parameterized form of the convolution. Not only does this allow to learn pairwise potential functions in dense graphical models, e.g., the FoP model from chapter 3, but it also has implications for bilateral filter applications and CNNs. By backpropagating an empirical loss through the operation we are able to learn task specific bilateral filters that outperform their predefined Gaussian counterparts on mesh denoising, image and material segmentation tasks. Inside a deep CNN architecture the bilateral filter becomes a convolutional building block that can substitute a spatial convolution and opens up CNNs to reason about sparse data, e.g., strokes of a pen of a character recognition task.

Next, we highlight possible directions for future work.

All mentioned applications in this thesis come with predefined features that describe the similarity of output variables through the closeness in a fixed feature space. Potentially, these features can be fitted from data, too. Krähenbühl and Koltun (2013) show how to train a restricted linear transformation before splatting the signal into the permutohedral lattice. Similarly, Jaderberg et al. (2015) modify the input signal before applying a convolution in a CNN. It is an open question how learning a signal transformation relates to learning the analytic form of the convolution as it is done in this work.

We present use-cases of the general permutohedral convolution inside a CNN architecture. However, all settings employ the operation as part of their input or output and never define a permutohedral convolution on an intermediate layer. We believe that this might lead to more compact representations within CNNs. As discussed in chapter 5, the general permutohedral convolu-
tion with bilateral features can be understood as an image adaptive filter when projected back into the spatial domain. Intermediate layers would be able to exploit this fact by concentrating filter responses depending on image dependent information.

In general, we believe that it is important to find a tighter connection to real world applications that take the reasoning of a computer vision system as input. Closing the loop to agents that navigate their environment comes with many issues. Among those, processing latency, structured loss function and limited access to training data will be core issues. This thesis is a small step towards this goal: we tune efficient methods to exactly the loss that matters for the prediction and allow to encode the relation of output variables to reduce the labeling burden. As mentioned in the background chapter 2, structured inference is particularly important for tasks with structured losses. It is those tasks that will profit most from inference algorithms that allow to model these dependencies and probably dominate approaches of pure data augmentation.


Sun, Min, Murali Telaprolu, Honglak Lee, and Silvio Savarese (2012). “An Efficient Branch-and-Bound Algorithm for Optimal Human Pose Estima-


NOTATION

\( n \) \hspace{1em} Number of samples.
\( c \) \hspace{1em} Number of image channels.
\( \kappa \) \hspace{1em} Number of feature dimensions.
\( d \) \hspace{1em} Number of structural elements, e.g., number of pixel.
\( i, j \) \hspace{1em} Index variables.
\( s \) \hspace{1em} Convolutional filter size.
\( P, Q \) \hspace{1em} Probability distribution.
\( p, q \) \hspace{1em} Probability Mass Function (PMF).
\( X, Y \) \hspace{1em} Random variable.
\( X, Y \) \hspace{1em} Multivariate random variable or matrix.
\( x, y \) \hspace{1em} Scalar.
\( x, y \) \hspace{1em} Vector.
\( \hat{x}, \hat{y} \) \hspace{1em} Estimates.
\( \Delta \) \hspace{1em} Loss function.
\( \phi \) \hspace{1em} Potential function.
\( k \) \hspace{1em} Kernel function.
\( f \) \hspace{1em} Factor.
\( \Phi \) \hspace{1em} Feature function.
\( H \) \hspace{1em} Entropy functional.
\( D \) \hspace{1em} Kullback-Leibler divergence.
\([x]_j\) \hspace{1em} Element j in vector x, usually \( x_j := [x]_j \).
\( E \) \hspace{1em} Expectation.
\( \mathbb{R} \) \hspace{1em} Set of real numbers.
\( \mathbb{Z} \) \hspace{1em} Set of integers.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>APK</td>
<td>Average Precision of Key-Points</td>
</tr>
<tr>
<td>BCL</td>
<td>Bilateral Convolution Layer</td>
</tr>
<tr>
<td>BNN</td>
<td>Bilateral Neural Network</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CRF</td>
<td>Conditional Markov Random Field</td>
</tr>
<tr>
<td>DPM</td>
<td>Deformable Part Model</td>
</tr>
<tr>
<td>FOP</td>
<td>Fields of Parts</td>
</tr>
<tr>
<td>GC</td>
<td>Graph Cut</td>
</tr>
<tr>
<td>HOG</td>
<td>Histogram of Oriented Gradients</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Program</td>
</tr>
<tr>
<td>IOU</td>
<td>Intersection over Union</td>
</tr>
<tr>
<td>LSP</td>
<td>Leeds Sport Poses</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum a-Posteriori</td>
</tr>
<tr>
<td>MRF</td>
<td>Markov Random Field</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>PCP</td>
<td>Percentage of Correct Parts</td>
</tr>
<tr>
<td>PMF</td>
<td>Probability Mass Function</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>PS</td>
<td>Pictorial Structures</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>SFM</td>
<td>Structure-from-Motion</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
</tbody>
</table>