Learning Inference Models for Computer Vision

# Learning Inference Models for Computer Vision

### Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)


vorgelegt von

## Varun Jampani
aus Tenali, Indien


Tübingen

2016

To my grandfather *K. Anjaneyulu*

# Abstract

Computer vision can be understood as the ability to perform *inference* on image data. Breakthroughs in computer vision technology are often marked by advances in inference techniques, as even the model design is often dictated by the complexity of inference in them. This thesis proposes learning based inference schemes and demonstrates applications in computer vision. We propose techniques for inference in both generative and discriminative computer vision models.

Despite their intuitive appeal, the use of generative models in vision is hampered by the difficulty of posterior inference, which is often too complex or too slow to be practical. We propose techniques for improving inference in two widely used techniques: Markov Chain Monte Carlo (MCMC) sampling and message-passing inference. Our inference strategy is to learn separate discriminative models that assist Bayesian inference in a generative model. Experiments on a range of generative vision models show that the proposed techniques accelerate the inference process and/or converge to better solutions.

A main complication in the design of discriminative models is the inclusion of prior knowledge in a principled way. For better inference in discriminative models, we propose techniques that modify the original model itself, as inference is simple evaluation of the model. We concentrate on convolutional neural network (CNN) models and propose a generalization of standard spatial convolutions, which are the basic building blocks of CNN architectures, to bilateral convolutions. First, we generalize the existing use of bilateral filters and then propose new neural network architectures with learnable bilateral filters, which we call 'Bilateral Neural Networks'. We show how the bilateral filtering modules can be used for modifying existing CNN architectures for better image segmentation and propose a neural network approach for temporal information propagation in videos. Experiments demonstrate the potential of the proposed bilateral networks on a wide range of vision tasks and datasets.

In summary, we propose learning based techniques for better inference in several computer vision models ranging from inverse graphics to freely parameterized neural networks. In generative vision models, our inference techniques alleviate some of the crucial hurdles in Bayesian posterior inference, paving new ways for the use of model based machine learning in vision. In discriminative CNN models, the proposed filter generalizations aid in the design of new neural network architectures that can handle sparse high-dimensional data as well as provide a way for incorporating prior knowledge into CNNs.

# Zusammenfassung

Maschinelles Sehen kann als die Fähigkeit verstanden werden Bilddaten zu interpretieren. Durchbrüche in diesem Feld gehen oft einher mit Fortschritten in Inferenztechniken, da die Komplexität der Inferenz die Komplexität der verwendeten Modelle bestimmt. Diese Arbeit beschreibt lernbasierte Inferenzmechanismen und zeigt Anwendungen im maschinellen Sehen auf, wobei auf Techniken für Inferenz in sowohl generativen als auch diskriminativen Modellen eingegangen wird.

Obwohl naheliegend und intuitiv verständlich, sind generative Modelle im maschinellen Sehen häufig nur eingeschränkt nutzbar, da die Berechnung der A-Posteriori-Wahrscheinlichkeiten oft zu komplex oder zu langsam ist, um praktikabel zu sein. Wir beschreiben Techniken zur Verbesserung der Inferenz in zwei weit verbreiteten Inferenzverfahren: 'Markov Chain Monte Carlo Sampling' (MCMC) und 'Message-Passing'. Die vorgeschlagene Verbesserung besteht darin, mehrere diskriminative Modelle zu lernen, die die Grundlage für Bayes'sche Inferenz über einem generativen Modell bilden. Wir demonstrieren anhand einer Reihe von generativen Modellen, dass die beschriebenen Techniken den Inferenzprozess beschleunigen und/oder zu besseren Lösungen konvergieren.

Eine der größten Schwierigkeiten bei der Verwendung von diskriminativen Modellen ist die systematische Berücksichtigung von Vorkenntnissen. Zur Verbesserung der Inferenz in diskriminativen Modellen schlagen wir Techniken vor die das ursprngliche Modell selbst verndern, da Inferenz in diesen die schlichte Auswertung des Modells ist. Wir konzentrieren uns auf 'Convolutional Neural Networks' (CNN) und schlagen eine Generalisierung der Faltungsoperation vor, die den Kern jeder CNN-Architektur bildet. Dazu verallgemeinern wir bilaterale Filter und präsentieren eine neue Netzarchitektur mit trainierbaren bilateralen Filtern, die wir 'Bilaterale Neuronale Netze' nennen. Wir zeigen, wie die bilateralen Filtermodule verwendet werden knnen, um existierende Netzwerkarchitekturen fr Bildsegmentierung zu verbessern und entwickeln ein auf Bilateralen Netzen basierendes Modell zur zeitlichen Integration von Information fr Videoanalyse. Experimente mit einer breiten Palette von Anwendungen und Datensätzen zeigen das Potenzial der vorgeschlagenen bilateralen Netzwerke.

Zusammenfassend schlagen wir Lernmethoden für bessere Inferenz in einer Reihe von Modellen des maschinellen Sehens vor, von inversen Renderern bis zu trainierbaren neuronalen Netzwerken. Unsere Inferenz-Techniken helfen bei der Berechnung der A-Posteriori-Wahrscheinlichkeiten in generativen Modellen und ermöglichen so neue Ansätze des modellbasierten machinellen Lernens im Bereich des maschinellen Sehens. In diskriminativen Modellen wie CNNs helfen die vorgeschlagenen verallgemeinerten

Filter beim Entwurf neuer Netzarchitekturen, die sowohl hochdimensionale Daten verarbeiten können als auch Vorkenntnisse in die Inferenz einbeziehen.

# Acknowledgements

First, my heartfelt thanks to my PhD advisor Prof. Peter Gehler without whom this thesis work would not have materialized. His suggestions helped me work on very interesting research problems in computer vision and machine learning whereas stimulating discussions with him helped me propose compelling techniques for those problems. He also taught me how to be critical about various aspects of a research project. I thank Dr. Sebastian Nowozin for being a co-advisor in my first PhD research project and teaching me how to be rigorous in research.

Thanks to Prof. Michael Black for allowing me to be a part of his vibrant research group in Tübingen, where I conducted my PhD research. Special thanks to Prof. Bipin Indurkhya for teaching me how to be mature in our reasoning and thinking while doing research. Thanks to my masters advisor Prof. Jayanthi Sivaswamy for her guidance during the start of my research career. Thanks to Dr. Pushmeet Kohli, Dr. Ali Eslami, Dr. John Winn and Dr. Daniel Tarlow for giving me a great internship opportunity at Microsoft Research that helped me getting a broader perspective over the fields of computer vision and machine learning. Thanks to Prof. Jan Kautz for giving me a wonderful opportunity at Nvidia to further pursue my research interests after my PhD.

I am thankful to Prof. Hendrik Lensch, Prof. Peter Gehler, Prof. Felix Wichmann and Prof. Matthias Bethge for being a part of my thesis committee and thanks to Prof. Hendrik Lensch and Prof. Peter Gehler for providing reviews to this thesis work. Special thanks to Dr. Iasonas Kokkinos for providing an additional review for this thesis. Thanks to Thomas Nestmeyer, Dr. Laura Sevilla, Angjoo Kanazawa, Fatma Güney, Jonas Wulff and Christoph Lassner for their helpful feedback and comments on this thesis manuscript.

I am fortunate to have collaborated with several great researchers during my PhD: Dr. Martin Kiefel, Raghudeep Gadde, Dr. Sebastian Nowozin, Dr. Ali Eslami, Dr. John Winn, Dr. Pushmeet Kohli, Dr. Matthew Loper, Prof. Michael Black, Dr. Laura Sevilla, Dr. Deqing Sun, Daniel Kappler, Dr. Renaud Marlet and Dr. Daniel Tarlow. I am very grateful to them. Special thanks to Raghudeep Gadde and Dr. Laura Sevilla for trusting me and letting me be a part of their research projects. I am highly thankful to Raghudeep Gadde for the countless hours we spent together on brainstorming various research questions and techniques.

I want to express my gratitude to my wonderful colleagues at Max-Planck Institute Thomas Nestmeyer, Raghudeep Gadde, Abhilash Srikantha, Dr. Laura Sevilla, Dr. Silvia Zuffi, Jonas Wulff, Fatma Güney, Dr. Martin Kiefel, Dr. Andreas Lehrmann, Christoph Lassner, Sergey Prokudin, Dr. Osman Ulusoy, Dr. Federica Bogo, Daniel Kappler, Prof. Michael Black, Dr. Andreas Geiger, Prof. Jürgen Gall, Dr. Chaohui Wang, Dr. Javier

# Contents

## Bibliography                                                              161

# Symbols and Notation

Unless otherwise mentioned, we use the following notation and symbols in this thesis. Here, we only list those symbols which are used across multiple chapters. Those symbols that are specific to particular sections or chapters are not listed here.

| Symbol | Description |
|---|---|
| $\mathbf{x}$ | Observation variables (in vectorized form) |
| $\mathbf{y}$ | Target variables (in vectorized form) |
| $\bar{\mathbf{y}}$ | Intermediate/Proposed target variables |
| $K_t, m_t, \cdots$ | Random variables at time step $t$ |
| $\theta$ | Set of all model parameters |
| $\alpha, \beta, \mu, \gamma$ | Model or training parameters |
| $\mathbf{f}$ | Pixel or superpixel features such as $(x, y, r, g, b)$ |
| $P(\cdot\|\cdot)$ | Probability distribution or density |
| $\mathcal{N}(\cdot\|\cdot)$ | Gaussian distribution |
| $\psi_u$ | Unary potential at each pixel/superpixel |
| $\psi_p$ | Pairwise potential between two pixels/superpixels |
| $L(\cdot), E(\cdot)$ | Loss/Objective/Energy function |
| $\mathcal{F}(\cdot)$ | Generic function relating input to output variables |
| $\Lambda$ | Diagonal matrix for scaling image features say $(x, y, r, g, b)$ |

# Acronyms

| Acronym | Full Form |
| --- | --- |
| AR | Acceptance Rate |
| BCL | Bilateral Convolutional Layer |
| BI | Bilateral Inception |
| BNN | Bilateral Neural Network |
| BP | Belief Propagation |
| CMP | Consensus Message Passing |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| CRF | Conditional Random Field |
| DDMCMC | Data Driven Markov Chain Monte Carlo |
| EP | Expectation Propagation |
| FC | Fully Connected |
| GPU | Graphics Processing Unit |
| IoU | Intersection over Union |
| INF-INDMH | Informed Independent Metropolis Hastings |
| INF-MIXMH | Informed Mixture Metropolis Hastings |
| INF-MH | Informed Metropolis Hastings |
| KDE | Kernel Density Estimation |
| MAP | Maximum a Posteriori |
| MCMC | Markov Chain Monte Carlo |
| MF | Mean-Field |
| MH | Metropolis Hastings |
| MHWG | Metropolis Hastings Within Gibbs |
| MP | Message Passing |
| PGM | Probabilistic Graphical Model |
| PSNR | Peak Signal to Noise Ratio |
| PSRF | Potential Scale Reduction Factor |
| PT | Parallel Tempering |
| REG-MH | Regenerative Metropolis Hastings |
| RF | Random Forest |
| RMSE | Root Mean Square Error |
| VMP | Variational Message Passing |
| VPN | Video Propagation Network |

# Chapter 1

# Introduction

Computer vision is the task of inferring properties of the world from the observed visual data. The observed visual data can originate from a variety of sensors such as color or depth cameras, laser scans etc. The properties of the world range from low-level material properties such as reflectance to high-level object properties such as 3D shape and pose. The field of computer vision encompasses a broad range of problems involving a variety of sensor data and world properties. Some example problems include: 'Inferring 3D pose and shape of an object from a depth image'; 'Inferring the actions of the persons from video' etc. Computer vision is hard because of variability in lighting, shape and texture in the scene. Moreover, there is sensor noise and the image signal is non-additive due to occlusion. Vision problems are inherently ambiguous as the sensor data is an incomplete representation of the richer 3D world. As a result, probabilistic frameworks are typically employed to deal with such ambiguity.

Following [204], there are three main components in any vision system: *model*, *learning* algorithm and *inference* algorithm. The *Model* forms the core of any vision system describing the mathematical relationship between the observed data and the desirable properties of the world. The set of parameters and/or structure of this mathematical model is learned using a *learning* algorithm. Once the model is learned, an *inference* algorithm is used to predict the world properties from a given observation.

Since models form the core of any vision system, let us briefly discuss the two broad categories in computer vision models: *Generative* and *Discriminative* models, which can be viewed as complementary and inverse to each other. Let us denote the observed data as a vector of random variables $\mathbf{x} \in \mathbb{R}^k$ and the target world properties as another vector of random variables $\mathbf{y} \in \mathbb{R}^l$. For example, $\mathbf{x}$ can be a vectorized representation of image pixels and $\mathbf{y}$ can be a vector representing the parameterized shape of an object in the image. Generative models characterize the probability of observed data given the world properties $P(\mathbf{x}|\mathbf{y}, \theta)$ (called 'likelihood') as well as a prior on target variables $P(\mathbf{y})$, where $\theta$ denotes the parameters of the model. Some example generative models include graphics systems and probabilistic graphical models. In this thesis, we use the term 'generative' more loosely in the sense that any model which characterizes the likelihood $P(\mathbf{x}|\mathbf{y}, \theta)$ and/or prior over target variables $P(\mathbf{y})$ is considered a 'generative' model. Discriminative models characterize the probability of world properties given the observed

data $P(\mathbf{y}|\mathbf{x}, \theta)$ (called 'posterior distribution'). In other words, generative models model the image formation process as a function of world parameters, whereas discriminative approaches model the target world parameters as a function of the given image. Once the model is defined, a learning algorithm is used to learn the model parameters $\theta$ and then an inference algorithm is used to predict the posterior distribution $P(\mathbf{y}|\mathbf{x}, \theta)$. In Chapter 2, we will discuss more about these models along with the inference and learning in them.

Depending on the type of model, a specialized learning or inference algorithm may not be required. For example, in the case of manually specified generative models (e.g. graphics system or fully specified Bayesian network), there is no need for a specialized learning algorithm since all the model parameters are already hand specified, but specialized inference techniques are required to invert such models. In the case of discriminative models, where the posterior distribution $P(\mathbf{y}|\mathbf{x}, \theta)$ is directly modeled, (e.g. neural networks or random forests), the inference mechanism just reduces to a simple evaluation of the model.

This dissertation focuses on improving the *inference* in prominent computer vision models. Inference plays a crucial role in any vision system as this would produce the desired end result for a given observation. The breakthroughs in computer vision technology are often marked by the advances in the inference techniques as even the model design is often dictated by the complexity of the inference in them. The inference result is what matters at the end and having a model with high fidelity but with no feasible or practical inference scheme (for instance recent photo-realistic graphics systems) is of little use for addressing vision problems. Thus, better inference techniques not only improve the existing computer vision systems but also help to develop better models. This thesis work proposes techniques for better inference in existing and widely used computer vision models.

## 1.1  Thesis Overview

In this section, we will discuss the objective of this thesis followed by the organization and contributions of various chapters.

### 1.1.1  Objective

The main aim of the work presented in this thesis is to improve the performance of inference algorithms used in different computer vision models. Inference is highly interlinked with model design and depending on the type of model, we propose different techniques to improve inference.

Generative models characterize the image formation process and the inference is typically performed via Bayesian inference techniques. Despite their intuitive appeal, the use

of generative models in vision is hampered by the difficulty of posterior inference (estimating $P(\mathbf{y}|\mathbf{x}, \theta)$). Existing inference techniques are either often too complex or too slow to be practical. In this thesis, we aim to alleviate some of these inference challenges in generative models. Specifically, we concentrate on improving two different inference schemes for generative vision models: First one is Markov chain Monte Carlo (MCMC) inference for inverting graphics engines and the second one is message passing inference in layered graphical models. A common strategy that we follow to improce inference in generative models is to learn a new discriminative model that is separate from the given generative model and propose modified inference schemes that make use of this new discriminative model for better inference.

Discriminative models directly model the posterior distribution $P(\mathbf{y}|\mathbf{x}, \theta)$ of the desired world parameters given the observed data. Thus the inference amounts to simple evaluation of the model. One of the main limitations of inference in discriminative models in the lack of principled techniques to incorporate our prior knowledge about the task. This is especially the case with the prominent convolutional neural network (CNN) models. In this thesis, we concentrate on CNN models and propose techniques to improve inference in them. We modify the original CNN models and make them more amenable for the incorporation of prior knowledge.

In summary, the aim of this thesis is to improve inference in general computer vision models. We do this by leveraging machine learning techniques to learn a new model for inference that is either separate from the original model (in case of generative models) or modifying the original model itself (in case of discriminative models). The work in this thesis deals with the construction and learning of such inference models and how such models can be integrated into the original vision models for better inference. We propose techniques for inference in diverse computer vision models ranging from hand-specified graphics systems to freely-parameterized neural network models. We concentrate on three types of models which are prevalent in modern computer vision systems: 1. Graphics systems; 2. Layered graphical models and 3. Convolutional neural networks.

## 1.1.2 Organization and Contributions

Since models form the core part of any vision system and this thesis involves the construction of new models for inference, in Chapter 2, we give an overview of different computer vision models along with the learning and inference mechanisms that are usually employed in them. In addition, we review some existing techniques that aim to improve inference in vision models by combining generative and discriminative approaches.

**Part I: Inference in Generative Vision Models**  In Part I (Chapters 3 and 4) of the thesis, we propose techniques for inference in generative computer vision models.

**Chapter 3 - The Informed Sampler**   In this Chapter, we propose a new sampling technique for inference in complex generative models like graphics systems. Markov chain Monte Carlo (MCMC) sampling is one of the most widely used and most generic inference scheme in such complex generative models. Although generic, in practice, MCMC sampling suffers from slow convergence unless the posterior is a unimodal low-dimensional distribution. By leveraging discriminative learning techniques with ancillary clustering and random forest models, we devise a mixture sampling technique that helps in faster mixing without losing the acceptance rate. We call this 'Informed Sampler' and demonstrate it using challenging generative graphics models and a popular model of human bodies [117]. Our method is similar in spirit to Data Driven Markov Chain Monte Carlo methods [281].

**Chapter 4 - Consensus Message Passing**   In this Chapter, we concentrate on layered and loopy graphical models that are prevalent in computer vision applications. When factors (relationship between the variables) in such graphical models are from a pre-defined family of distributions, inference is generally performed using standard message passing techniques such as 'expectation propagation' [184] and 'variational message passing' [267]. We observe that these inference techniques fail to converge or even diverge when the graphical model is loopy with a large number of variables. The failure of these inference techniques can be attributed to the algorithm's inability to determine the values of a relatively small number of influential variables which we call global variables (e.g. light in a scene). Without accurate estimation of these global variables, it can be very difficult for message passing to make meaningful progress on the other variables in the model. As a remedy, we exploit the layered structure of the model and learn ancillary random forest models that learn to predict these influential variables and use them for better message passing inference. We call this method 'Consensus Message Passing' (CMP) and demonstrate it on a variety of layered vision models. Experiments show that CMP leads to significantly more accurate inference results whilst preserving the computational efficiency of standard message passing.

**Part II: Inference in Discriminative Vision Models**   In Part II (Chapters 5, 6 and 7) of the thesis, we focus on inference in discriminative CNN models.

**Chapter 5 - Learning Sparse High Dimensional Filters**   2D spatial convolutions form the basic unit of CNN models. Spatial convolutions are perhaps the simplest, fastest and most used way of propagating information across pixels. Despite their staggering success in a wide range of vision tasks, spatial convolutions have several drawbacks: There are no well-established ways of incorporating prior knowledge into spatial filters; Spatial convolutions quickly get intractable when filtering data of increasing dimensionality; and the receptive fields of the filters are image-agnostic. Spatial convolutions are usually confined to a local neighborhood of pixels and thus many deep layers of spatial convolutions or post-processing conditional random field (CRF) formulations are required for long-range propagation of information across pixels. Bilateral

filtering [15, 251], on the other hand, provides a simple yet powerful framework for long range information propagation across pixels. But the traditional use of bilateral filtering is confined to a manually chosen parametric from, usually a Gaussian filter. In this chapter, we generalize the bilateral filter parameterization using a sparse high-dimensional linear approximation and derive a gradient descent algorithm, so the filter parameters can be learned from data. We demonstrate the use of learned bilateral filters in several diverse applications where Gaussian bilateral filters are traditionally employed: color up-sampling, depth up-sampling [148] and 3D mesh denoising [82]. The ability to learn generic high-dimensional sparse filters allows us to stack several parallel and sequential filters like in convolutional neural networks (CNN) resulting in a generalization of 2D CNNs which we call 'Bilateral Neural Networks' (BNN). We demonstrate the use of BNNs using an illustrative segmentation problem and sparse character recognition. Gaussian bilateral filters are also employed for mean-field inference in fully connected conditional random fields (DenseCRF) [151]. Existing works on DenseCRFs are confined to using Gaussian pairwise potentials due to the traditional use of Gaussian kernels in bilateral filtering. By learning bilateral filters, we remove the need of confining to Gaussian pairwise potentials which has the added advantage of directly learning the pairwise potentials for a given task. We showcase the use of learning edge potentials in DenseCRF with experiments on semantic segmentation and material segmentation. In summary, we propose a general technique for learning sparse high-dimensional filters that help in improving the model and inference in DenseCRF models and also generalizes 2D CNNs.

**Chapter 6 - Video Propagation Networks**   Videos carry redundant information across frames and the information propagation across video frames is valuable for many computer vision applications such as video segmentation, color propagation etc. In this chapter, we propose a novel neural network architecture for video information propagation. We leverage learnable bilateral filters, developed in the previous chapter, and propose a 'Video Propagation Network' (VPN) that processes video frames in an adaptive manner. The model is applied online: it propagates information forward without the need to access future frames other than the current ones. In particular we combine two components, a temporal bilateral network for dense and video adaptive filtering, followed by a spatial network to refine features and increased flexibility. We present experiments on video object segmentation and semantic video segmentation and show increased performance comparing to the best previous task-specific methods, while having favorable runtime. Additionally we demonstrate our approach on an example regression task of propagating color in a grayscale video.

**Chapter 7 - Bilateral Inception Networks**   In this Chapter, we propose a new CNN module which we call the 'Bilateral Inception' (BI) module that can be inserted into *existing* segmentation CNN models. BI modules help in image adaptive long-range information propagation across intermediate CNN units at multiple scales. We show empirically

that this alleviates some of the need for standard post-processing inference techniques such as DenseCRF. In addition, our module helps in recovering the full resolution of segmentation result, which is generally lost due to max-pooling and striding. Experiments on different base segmentation networks and datasets showed that our BI modules result in reliable performance gains in terms of both speed and accuracy in comparison to traditionally employed DenseCRF/Deconvolution techniques and also recently introduced dense pixel prediction techniques.

## 1.2  List of Publications

The contributions in this thesis mainly comprise of work from the following publications [135, 131, 143, 134, 88, 133]:

- V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler. The informed sampler: A discriminative approach to bayesian inference in generative computer vision models. *CVIU*, 2015. [135]

- V. Jampani, S. M. A. Eslami, D. Tarlow, P. Kohli, and J. Winn. Consensus message passing for layered graphical models. In *AISTATS*, 2015. [131]

- M. Kiefel, V. Jampani, and P. V. Gehler. Permutohedral lattice CNNs. In *ICLR Workshops*, 2015. [143]

- V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks. In *CVPR*, 2016. [134]

- R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. Gehler. Superpixel convolutional networks using bilateral inceptions. In *ECCV*, 2016. [88]

- V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, 2017. [133]

The following publications [132, 132, 89] are a part of my PhD research but are outside the scope of this thesis:

- V. Jampani, R. Gadde, and P. V. Gehler. Efficient facade segmentation using auto-context. In *WACV*, 2015. [132]

- L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *CVPR*, 2016. [221]

- R. Gadde, V. Jampani, R. Marlet, and P. V. Gehler. Efficient 2D and 3D facade segmentation using auto-context. *PAMI*, 2017. [89]

# Chapter 2

# Models and Inference in Computer Vision

Models play a central role in the study of both biological and artificial vision. Helmholtz, in the 19th century, popularized the human vision as a result of psychological inference in learned models [85, 46] as opposed to native processing in lower visual system or eyes. With the advent of computers in the 20th century, researchers are able to formulate, learn and evaluate several computational models of vision. More recently, with powerful parallel computing hardware like graphics processing units (GPU), researchers are able to learn and do inference in highly complex models with even millions of parameters. In this chapter, we present an overview of different computer vision models and discuss the inference and learning techniques therein. Since this thesis work mainly constitutes the development of new inference techniques, we emphasize the difficulty of inference in different models and discuss several remedies proposed in the literature.

## 2.1 Models in Computer Vision

Models describe the mathematical relationship between the observed data and the desired properties of the world. Computer vision models are often probabilistic in nature due to the inherent ambiguity in vision problems. Due to the broad range of problems in computer vision, there is no single model that can work well for various vision tasks. Depending on the nature of problem and the availability of data, different models work well for different scenarios. Visual data is complex with variability arising due world properties such as occlusion, lighting, texture, geometry, depth ordering etc. It is very difficult to model the relationship between all the aspects of the world and the visual data, and do inference therein. Vision models usually are highly specific to one or few aspects of the world.

As briefly mentioned in Chapter 1, computer vision models can be broadly classified into two types: *Generative* and *Discriminative* models, which can be viewed as complementary and inverse to each other. Generative models characterize the probability of observed data given the world properties $P(\mathbf{x}|\mathbf{y}, \theta)$ and discriminative models characterize the probability of world properties given the observed data $P(\mathbf{y}|\mathbf{x}, \theta)$, where $\theta$

denotes the parameters of the model. In other words, generative models model the image formation process as a function of world parameters, whereas discriminative approaches model the desired world parameters as a function of the given image. As mentioned in Chapter 1, we use the term 'generative' more loosely in the sense that any model which characterizes the likelihood $P(\mathbf{x}|\mathbf{y}, \theta)$ and/or prior over target variables $P(\mathbf{y})$ is considered a 'generative' model. Next, we give an overview of these two complementary models discussing the advantages and disadvantages of both.



(a) Graphics Renderings    (b) A face generative model    (c) Example face data

Figure 2.1: **Sample generative models in computer vision.** (a) Sample renderings from modern graphics engines [2, 4]. Modern graphics provide renderings with stunning level of realism and vision problems can be approached as inverting such graphics systems. Images courtesy from the official websites of CryEngine [2] and Lumberyard [4]. (b) A layered graphical model (factor graph) for faces (explained in Sec. 2.2.2). Here the vision problem could be inferring the reflectance map, normal map and light direction from a given face image. (c) Sample face data from Yale B face dataset [94, 161].

## 2.2 Generative Vision Models

A conceptually elegant view on computer vision is to consider a generative model of the physical image formation process. The observed image becomes a function of unobserved variables of interest (for instance the presence and positions of objects) and nuisance variables (for instance light sources and shadows). When building such a generative model, we can think of a scene description $\mathbf{y}$ that produces an image $\mathbf{x} = G(\mathbf{y}, \theta)$ using a deterministic rendering engine $G$ with parameters $\theta$, or more generally, results in a distribution over images, $P(\mathbf{x}|\mathbf{y}, \theta)$. Generative models provide a powerful framework for probabilistic reasoning and are applicable across a wide variety of domains, including computational biology, natural language processing, and computer vision. For example, in computer vision, one can use graphical models to express the process by which a face is lit and rendered into an image, incorporating knowledge of surface normals, lighting and even the approximate symmetry of human faces. Models that make effective use of this information will generalize well, and they will require less labelled training data than their discriminative counterparts (e.g. random forests or neural networks) in order to make accurate predictions.

Given an image observation $\mathbf{x}$ and a prior over scenes $P(\mathbf{y})$ we can then perform *Bayesian inference* to obtain the posterior distribution over the target variables $P(\mathbf{y}|\mathbf{x}, \theta)$ (also called 'posterior inference'):

$$P(\mathbf{y}|\mathbf{x}, \theta) = \frac{P(\mathbf{x}|\mathbf{y}, \theta)P(\mathbf{y})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{y}, \theta)P(\mathbf{y})}{\sum_{\mathbf{y}'} P(\mathbf{x}|\mathbf{y}', \theta)P(\mathbf{y}')}. \tag{2.1}$$

The summation in the denominator runs over all the possible values of $\mathbf{y}$ variable and would become integral in the case of continuous variables. $P(\mathbf{x}|\mathbf{y}, \theta)$ is the *likelihood* of the observed data. Inspecting the above Bayesian formula shows that it is straight forward to compute the numerator as that involves simply evaluating the generative model. The key difficulty with Bayesian inference is computing the denominator in Eq. (2.1). Often, it is not feasible to evaluate the summation over all the possible target variables even for slightly non-trivial models. This makes it difficult to obtain closed-form solutions for posterior inference resulting in the development and use of several approximate inference techniques such as Markov Chain Monte Carlo (MCMC) sampling and variational inference, which we will briefly discuss later.

There are different types of generative models with varying model fidelity and complexity of inference therein. In general, generative models which accurately model the image formation process (e.g. graphics engines) have complex non-linear functions resulting in a challenging inference task. Building a good generative model for a given task involves finding a good trade-off between model fidelity and inference complexity.

Figure 2.1 shows some sample generative models in computer vision. Advances in graphics and physics of light transport resulted in generative models with high fidelity as is evident in modern computer games and animation movies. But it is difficult to

Figure 2.2: **An example 'inverse graphics' problem.** A graphics engine renders a 3D body mesh and a depth image using an artificial camera. By Inverse Graphics we refer to the process of estimating the posterior probability over possible bodies given the depth image.

invert such complex models. Probabilistic graphical models provide the widely adopted framework for generative computer vision models. Although graphical models have less fidelity and only model one or two aspects of the world properties, they are generally preferred over graphic systems as inference in them is faster and more reliable. Next, we will discuss these two prominent generative vision models: *Inverse graphics* and *Probabilistic graphical models*.

## 2.2.1  Inverse Graphics

Modern graphics engines (e.g., game engines like CryEngine [2] and Lumberyard [4]) leverage dedicated hardware setups and provide real-time renderings with stunning level of realism. Some sample renderings from modern game engines [2, 4] are shown in Fig. 2.1(a). Vision problems can be tackled with posterior inference (Eq. (2.1)) in such accurate computer graphics systems. This approach for solving vision problems can be understood as *Inverse Graphics* [22]. The target variables $\mathbf{y}$ correspond to the input to the graphics system and the observation variables are the output $\mathbf{x} = G(\mathbf{y}, \theta)$. The deterministic graphics system $G$ can be converted into a probabilistic generative model by defining a prior over the target variables (input to the graphics system) $P(\mathbf{y})$ and also defining an approximate likelihood function $P(\mathbf{x}|G(\mathbf{y}, \theta))$ characterizing the model imperfections. If the model imperfections are neglected, the likelihood can be given using the *delta* function $P(\mathbf{x}|\mathbf{y}) = \delta(\mathbf{x} - G(\mathbf{y}, \theta))$. An example 'inverse graphics' problem, which we tackle in the next chapter, is depicted in Fig. 2.2, where the graphics engine renders a depth image given 3D body mesh and camera parameters, and a vision problem would be the inverse estimation of body shape given a depth image.

Modern graphic engines are based on physics principles and thus most of the rendering parameters $\theta$ are set to mimic the real world physics. Learning in graphics generative

models mainly involves learning the prior $P(\mathbf{y})$ over the target world properties which are input to the graphics system. Depending on the type of model, several learned priors are proposed in the literature. An example is the SCAPE model [11, 117] for modeling the prior over human body shape and pose for the example shown in Fig. 2.2.

Since modern rendering engines involve complex non-linear functions, it is usually not feasible to obtain a closed-form solution for the posterior distribution (Eq. (2.1)). Even several approximate inference techniques like variational optimization techniques [147] cannot be easily employed for posterior inference in complex graphics systems. 'Monte Carlo' sampling provides a generic inference technique for such complex models and can be used even when the internals of the graphics systems are not known. The aim of sampling methods is to characterize the posterior distribution with *independent and identically distributed* samples. There exists many Monte Carlo sampling strategies such as uniform sampling, rejection sampling, importance sampling [107, 216] etc. Here, we limit our discussion to the widely used Markov Chain Monte Carlo (MCMC) sampling techniques.

**MCMC Sampling**

MCMC sampling [181] is a particular instance of sampling methods that generates a sequence of target variables (samples) by simulating a *reversible* Markov chain.

$$\mathbf{y}_1 \rightarrow \mathbf{y}_2 \rightarrow \mathbf{y}_3 \rightarrow \cdots \rightarrow \mathbf{y}_n \qquad (2.2)$$

This Markov chain of samples approximates the target distribution (posterior distribution in the case of 'inverse graphics'). The Markov property states that at every sequence step $t$, given the present sample $\mathbf{y}_t$ in the sequence, the next sample $\mathbf{y}_{t+1}$ is independent of all the previous samples $P(\mathbf{y}_{t+1}|\mathbf{y}_t, \ldots, \mathbf{y}_1) = P(\mathbf{y}_{t+1}|\mathbf{y}_t)$.

Let us denote the target distribution with $\pi(\cdot)$. In the inverse graphics setting, the target distribution is the posterior $\pi(\mathbf{y}) = P(\mathbf{y}|\mathbf{x}, \theta)$. And, let us denote the transition probability between the two states (samples) in the Markov chain be $T(\mathbf{y}_{t+1}|\mathbf{y}_t)$ or $T(\mathbf{y}_t \rightarrow \mathbf{y}_{t+1}) \in [0, 1]$. One way to ensure that the Markov chain is *reversible* and converging to the target distribution is to check whether the following *detailed balance condition* holds for any two states $\mathbf{y}_t$ and $\bar{\mathbf{y}}$ [147]:

$$\pi(\mathbf{y}_t)T(\mathbf{y}_t \rightarrow \bar{\mathbf{y}}) = \pi(\bar{\mathbf{y}})T(\bar{\mathbf{y}} \rightarrow \mathbf{y}_t) \qquad (2.3)$$

Note that the above detailed balance condition is satisfied when the transition probability distribution is close to the target distribution. Since we do not know the target distribution, designing transition probability distributions that satisfies the detailed balance condition is difficult. The Metropolis-Hastings (MH) algorithm [181], instead of devising special transition probabilities, introduces the *acceptance probability* to each Markov chain transition $A(\mathbf{y}_t \rightarrow \bar{\mathbf{y}}) \in [0, 1]$. The detailed balance condition then becomes:

$$\pi(\mathbf{y}_t)T(\mathbf{y}_t \to \bar{\mathbf{y}})A(\mathbf{y}_t \to \bar{\mathbf{y}}) = \pi(\bar{\mathbf{y}})T(\bar{\mathbf{y}} \to \mathbf{y}_t)A(\bar{\mathbf{y}} \to \mathbf{y}_t) \tag{2.4}$$

It can be verified [147] that the following acceptance probability satisfies the above detailed balance condition:

$$A(\mathbf{y}_t \to \bar{\mathbf{y}}) = \min\left(1, \frac{\pi(\bar{\mathbf{y}})T(\bar{\mathbf{y}} \to \mathbf{y}_t)}{\pi(\mathbf{y}_t)T(\mathbf{y}_t \to \bar{\mathbf{y}})}\right) \tag{2.5}$$

With the use of the above acceptance rule, instead of designing task-specific transition probability distributions, any transition probability distribution $T$ with non-zero probability over the range of all target variables can be used for MH sampling. $T$ is also called 'proposal distribution' since it is used to propose the next sample in the Markov chain, which is then either accepted or rejected based on the acceptance probability. Below, we summarize the Metropolis-Hastings (MH) MCMC algorithm.

**Metropolis-Hastings (MH) MCMC:**   Sampling from $\pi(\cdot)$ consists of repeating the following two steps [174]:

1. Propose a transition using a *proposal distribution $T$* and the current state $\mathbf{y}_t$

$$\bar{\mathbf{y}} \sim T(\cdot|\mathbf{y}_t)$$

2. Accept or reject the transition based on Metropolis Hastings (MH) acceptance rule:

$$\mathbf{y}_{t+1} = \begin{cases} \bar{\mathbf{y}}, & \text{rand}(0,1) < \min\left(1, \frac{\pi(\bar{\mathbf{y}})T(\bar{\mathbf{y}} \to \mathbf{y}_t)}{\pi(\mathbf{y}_t)T(\mathbf{y}_t \to \bar{\mathbf{y}})}\right), \\ \mathbf{y}_t, & \text{otherwise.} \end{cases}$$

Different MCMC techniques mainly differ in the type of the proposal distribution $T$. Note that we do not need to compute the target (posterior) probabilities, but only the *ratio* of posterior probabilities $\frac{\pi(\bar{\mathbf{y}})}{\pi(\mathbf{y}_t)}$. This makes MCMC sampling suitable for the inverse graphics setting where it is not feasible to get a closed-form solution for the normalization constant in the posterior distribution (denominator in Eq. (2.1)).

The key aspect of the MH sampling is the number of steps it requires until it converges to the target distribution. If the proposal distribution is very different from the target distribution, the samples tend to be frequently rejected resulting in a long wait for convergence. In practice, it is difficult to measure the convergence of any sampler since we do not know the target distribution. In Chapter 3, we discuss several diagnostic measures that indicate the convergence of MCMC sampling. In the case of 'inverse graphics', each forward rendering step takes a considerable amount of time and we would like the sampler to accept as many samples as possible. A key for improving the MH sampling efficiency is to design the proposal distributions that match the target posterior distribution. In Chapter 3, we devise such technique by leveraging discriminative learning

Figure 2.3: **An example factor graph.** Variable nodes (circles) and factor nodes (squares) representing the factorized function $P(a,b,c,d,e,f) = \psi_i(a)\psi_j(a,b)\psi_k(b,d)\psi_l(b,c,e,f)\psi_m(f)\psi_n(c,e,f)$. Also shown are sample variable-to-factor messages (green arrows) and, factor-to-variable messages (red arrows).

techniques for learning the proposal distribution. Refer to [174, 147] for more details on MCMC sampling. In Chapter 3, we study the behavior of MCMC sampling and its variants for inverting graphics engines and propose techniques for improving the sampling efficiency.

## 2.2.2 Probabilistic Graphical Models

Probabilistic graphical models (PGM) provide a rigorous mathematical framework, based on probability and graph theory, for modeling the relationship between the world and image properties. PGMs have been popular not only in computer vision but also in related fields such as natural language processing, speech processing, etc. Several model representations, learning and inference schemes haven been developed in the PGM literature and even a concise description of them would be an inundating task and is outside the scope of this thesis. Refer to [147] for a comprehensive overview of PGMs. PGMs generally represent input-output relationships with factorized functions and are typically confined to a restricted domain so that efficient inference techniques can be applied.

PGMs are popular models of choice when the joint distribution of all the target and observation variables can be factorized into independent distributions each involving a subset of variables. This factorization of the joint distribution is represented with the structure of the graph, where each node represents a subset of variables and edges between the nodes represent the joint or conditional distributions between the corresponding node variables.

### Factor Graph Representation

Factor graphs provide a useful visualization and mathematical formalization for representing probabilistic graphical models. Factor graphs are *bipartite* graphs where nodes in the graph are divided into two types: 'variable' nodes represented as circles and 'factor' nodes represented as squares. Variable nodes represent the random variables in the

graphical model and factor nodes represent the statistical relationship between the variable nodes that they are connected to. Every edge in the factor graph connects a variable node to factor node. That is, there are no *direct* edges among the variable nodes or factor nodes. An example factor graph is shown in Fig. 2.3, where we represent variable nodes as $a, b, c, \cdots$ and factor nodes as $i, j, k, \cdots$. The factor function associated with a factor node $a$ is represented as $f_a$ and the states of variables associated with a variable node $i$ is represented as $i$. The joint function in Fig. 2.3, $P(a, b, c, d, e)$ is factorized into five independent factors $\psi_i(a)\psi_j(a,b)\psi_k(b,d)\psi_l(b,c,e,f)\psi_m(f)f_n(c,e,f)$. Each factor in the graph represent a function and the product of all factor functions makes up the joint function. In the context of probabilistic generative models, these functions are probability distributions. The edges in the factor can be either directed or un-directed representing the joint and conditional distributions respectively. In the case of discrete variables, the factor functions are probability tables assigning the probabilities for all the states of the factor variables. And, in the case of continuous variables, the factor functions are probability density functions. Another example of factor graph, representing a generative model of faces, is shown in Fig. 2.1(b). We will discuss more about this face model later in this section.

Graphical representations like factor graphs have several advantages. They provide an intuitive way of representing the statistical dependencies between the variables. Given the factor graph, it is easy to visualize *conditional independence* between variables. For a factor graph with undirected edges, variables $s$ and $t$ are independent given a set of variables $v$ ($s \perp\!\!\!\perp t | v$) if every path between $s$ and $t$ have some node $v \in v$. For instance, in the factor graph shown in Fig. 2.3, variable $d$ is independent of $a, c, e$ and $f$ given $b$ is observed: $d \perp\!\!\!\perp a, c, e, f | b$. Perhaps the most important advantage of graphical representations like factor graphs is that we can perform Bayesian inference by using the graph structure. For example, the widely used *message-passing* inference is performed by passing messages between factor and variable nodes.

**Message Passing**

Message Passing (MP) inference forms a general class of algorithms that are used to estimate the factor graph distributions, i.e. maximizing or minimizing the joint distribution (P(a,b,c,d,e,f) in Fig. 2.3). MP inference proceeds by passing messages (distributions or density functions) between variable and factor nodes. Here, we describe the message passing in the famous 'Sum-Product' belief-propagation (BP) algorithm. Messages are probability distributions that represent the beliefs over the variables to/from which the messages are being sent. MP inference in factor graphs has two types of messages: Messages from variable to factor nodes $\mu_{a \to i}$ (green arrows in Fig. 2.3) and messages from factor to variable nodes $\mu_{i \to a}$ (red arrows in Fig. 2.3). In the case of Sum-Product BP, these messages are defined as follows.

*Variable-to-Factor Message:* A message from variable to factor node is the product of all the messages that the variable node receives from its neighboring factor nodes except

the recipient factor node. In Fig. 2.3, the message $\mu_{b \to m}$ from variable $b$ to factor $m$ is the product of the messages that $b$ receives, i.e. $\mu_{k \to b} \mu_{l \to b}$. In general, a message from a variable node $g$ to factor node $r$ is defined as:

$$\mu_{g \to r}(x_g) = \prod_{\hat{r} \in N(g) \setminus \{r\}} \mu_{\hat{r} \to g}(x_g), \qquad (2.6)$$

where $N(g)$ is the set of neighboring nodes to $g$ and $x_g$ represent the states of $g$.

*Factor-to-Variable Message:* A message from factor to variable node is first computed as the product of factor function with all the incoming messages from variables except the target variable. The resulting product is then marginalized over all the variables except the one associated with the target node variables. For example, in Fig. 2.3, the message from $l$ to $b$ is computed by marginalizing the product $\psi_l(b, c, e, f) \mu_{c, e \to l} \mu_{f \to l}$ over non-target variables $c, e, f$. In general, a message from factor node $r$ to variable node $g$ is defined as:

$$\mu_{r \to g}(x_g) = \sum_{x' : V_r \setminus g} \psi_r(x') \prod_{\hat{g} \in N(r) \setminus \{g\}} \mu_{\hat{g} \to r}(x_{\hat{g}}), \qquad (2.7)$$

where $N(r)$ represents the neighboring nodes of $r$ and $V_r$ represent all the variables attached to $r$. The summation in the above equation becomes an integral when dealing with continuous variables.

In a typical message passing inference run, the above messages are repeatedly computed and sent between factor and variable nodes. Depending on the structure of the factor graph, different message priorities are used. Upon convergence or passing messages for a pre-defined number of iterations, the marginal distribution at each variable node is computed as the product of all its incoming messages from the neighboring factor nodes:

$$P(x_g) = \prod_{\hat{r} \in N(g)} \mu_{\hat{r} \to g}(x_g). \qquad (2.8)$$

Messages represent a probability of each possible state of a variable. In the case of discrete variables with small number of states, it is easy to represent messages as probability distribution tables. However, in the case of continuous variables, the messages must be full functions of the variables. Some approximations are typically required to represent messages with continuous distributions. Different message passing algorithms differ in the way the messages are approximated. Some MP algorithms assume Gaussian form for the messages with either restricting the type of factor functions [266] that always result in Gaussian messages or projecting the computed messages into Gaussian form [184]. Some other MP algorithms use mixture of Gaussians [238] or set of particles/samples [125] for representing messages.

**Variational Inference**

Except for small graphical models, exact Bayesian inference (say, with the above mentioned Sum-Product BP) in PGMs is usually not possible. This is especially true for computer vision models which typically involve several hundreds or thousands of variables. Variational Bayesian inference is one of the widely used technique for performing inference in PGMs. These methods try to find an approximation $Q(\mathbf{y})$ to the true posterior distribution $P(\mathbf{y}|\mathbf{x})$ via optimization techniques. This approximate distribution is usually taken from a known simpler family of distributions (such as exponential family) and the optimization is performed over the space of that family of distributions. For example, for the above joint distribution function $P(a,b,c,d,e,f)$, an approximation could be a factorized distribution $Q = Q_1(a)Q_2(b)Q_3(c)Q_4(d)Q_5(e)Q_6(f)$, each from an exponential family of distributions. The most commonly used optimization function is minimizing the Kullback-Leibler (KL) divergence of $P$ from $Q$ in order to find a close approximation to $P$:

$$D_{KL}(Q||P) = \underbrace{\sum_z Q(\mathbf{y}) \frac{Q(\mathbf{y})}{P(\mathbf{y},\mathbf{x})}}_{-\mathcal{L}(Q)} + \log P(\mathbf{x}) \qquad (2.9)$$

Minimizing the above KL-divergence translates to maximizing the variational lower bound $\mathcal{L}(Q)$ as $P(\mathbf{x})$ is constant. $\mathcal{L}(Q)$ is also called *energy functional* or *variational free energy* as it can be written as the sum of energy $\mathbb{E}_Q[\log P(\mathbf{y},\mathbf{x})]$ and the entropy of $Q$. $Q$ is chosen in such a way that $\mathcal{L}(Q)$ becomes tractable and maximizable.

In general, the energy functional is maximized by passing messages between different variable nodes in the factor graph (e.g. factor graphs in Figs. 2.3, 2.1(b)). Following [147], depending on the type of approximations to $Q$ and the energy functional, methods for maximizing $\mathcal{L}(Q)$ can be categorized into three types. The first category of methods optimizes approximate versions of the energy functions by passing messages in the simplified versions of the given factor graph. This includes the *loopy belief propagation* [86, 265] algorithm. The second category of methods try to maximize the exact energy functional but uses approximate message propagation steps, for example approximating complex messages with distributions from the exponential family. This is equivalent to using relaxed consistency constraints on $Q$. These class of methods are also known as *expectation propagation* (EP) [184] algorithms. The third commonly used category of methods maximize the exact energy functional but restrict $Q$ to simple factorized distributions, which is called *mean-field* approximation. One of the most commonly used message passing technique for optimizing the energy functional is *variational message passing* (VMP) [267]. In Chapter 4, we show how EP and VMP fail to converge for inference in model shown in Fig. 2.1(b) and propose a remedy for that.

Several other inference techniques such as MCMC sampling can be used for inference in PGMs. Refer to [83] for a tutorial on variational Bayesian inference and to [147, 261] for a comprehensive review of various inference techniques in PGMs.

**Two Example Models**

Here, we give a brief overview of two popular types of PGMs in vision which are later used in this thesis: Layered graphical models and fully connected CRFs.

**Layered Graphical Models:** Several vision models are hierarchical in nature and can be naturally expressed with layered graphical models. Figure 2.1(b) shows an example layered *factor graph* model for faces. Here, a vision task could be: Given an observation of pixels $\mathbf{x} = \{x_i\}$, we wish to infer the reflectance value $r_i$ and normal vector $\mathbf{n_i}$ for each pixel $i$ (see Fig. 2.1(c)). The model shown in Fig. 2.1(b) represents the following approximate image formation process: $x_i = (\mathbf{n_i} \cdot \mathbf{l}) \times r_i + \varepsilon$, thereby assuming Lambertian reflection and an infinitely distant directional light source with variable intensity. Each factor in the graph is a conditional probability distribution providing the factorization for the joint distribution:

$$P(\mathbf{x}, \mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{n}, \mathbf{l}) = P(\mathbf{n})P(\mathbf{l})P(\mathbf{r}) \prod_i P(x_i)P(x_i|z_i)P(z_i|r_i, s_i)P(s_i|\mathbf{n}_i, \mathbf{l}), \qquad (2.10)$$

where $s_i$ and $z_i$ represent the intermediate shading and non-noisy image observation variables. We omitted the model parameters $\theta$ in the above equation for the sake of simplicity. A vision task could to estimate the posterior distribution $P(\mathbf{r}, \mathbf{s}, \mathbf{n}, \mathbf{l}|\mathbf{x})$. Note that this generative model is only a crude approximation of the true image formation process (e.g. each pixel is modeled independently and it does not account for shadows or specularities). Such approximations are customary to PGMs as several PGM inference techniques cannot be applied for models with complex non-linear factors. Note that even for a relatively small image of size $96 \times 84$, the face model contains over 48,000 latent variables and 56,000 factors, and as we will show in Chapter 4, standard message passing routinely fails to converge to accurate solutions.

**Fully Connected CRFs:** Fully connected conditional random fields, also known as DenseCRFs, are CRF models where every variable in the image is connected to every other variable via pairwise edge potentials. For illustration purposes, let us consider the task of semantic segmentation which is labelling each pixel in a given image with a semantic class. See Fig. 2.4 for an illustration. For the segmentation problem, DenseCRFs are generally used to encode the prior knowledge about the problem: 'Pixels that are spatially and photometrically similar are more likely to have the same label'.

For an image $\mathbf{x}$ with $n$ pixels, the semantic segmentation task is to produce a labelling $\mathbf{y}$ with discrete values $\{y_1, \ldots, y_n\}$ in the label space $y_i \in \{1, \ldots, \mathcal{L}\}$. The DenseCRF model has unary potentials $\psi_u(y) \in \mathbb{R}$, e.g., these can be the output of CNNs. The pairwise potentials, as introduced in [151], are of the form $\psi_p^{ij}(y_i, y_j) = \mu(y_i, y_j)k(\mathbf{f}_i, \mathbf{f}_j)$ where $\mu$ is a label compatibility matrix, $k$ is a Gaussian kernel $k(\mathbf{f}_i, \mathbf{f}_j) = \exp(-(\mathbf{f}_i - \mathbf{f}_j)^\top \Sigma^{-1}(\mathbf{f}_i -$

| ■ Road | ■ Sidewalk | ■ Building | ■ Wall | ■ Fence | ■ Pole | ■ Traffic Light | ■ Traffic Sign |
| ■ Vegetation | ■ Terrain | ■ Sky | ■ Person | ■ Rider | ■ Car | ■ Truck |
| ■ Bus | ■ Train | ■ Motorcycle | ■ Bicycle |

(a) Sample Image          (b) Ground Truth Semantics

Figure 2.4: **Illustration of semantic segmentation task.** A sample image from Cityscapes street scene dataset [57] and the corresponding ground truth semantic labels.

$\mathbf{f}_j))$ and the vectors $\mathbf{f}_i$ are feature vectors at each point. Commonly used features are position and color values at the pixels (e.g., $\mathbf{f} = (x, y, r, g, b)^\top$). In the DenseCRF model, the energy functional for an image $\mathbf{x}$ thus reads:

$$P(\mathbf{y}|\mathbf{x}) \propto \exp(-\sum_i \psi_u(y_i) - \sum_{i>j} \psi_p^{ij}(y_i, y_j)). \qquad (2.11)$$

Because of the dense connectivity, exact MAP or marginal inference is intractable. The main result of [151] is to derive the mean-field approximation for this model and to relate it to bilateral filtering which enables tractable approximate inference. As described above, mean-field approximation is a type of variational inference where the approximate distribution $Q$ is considered to be fully-factorized across pixels: $Q = \prod_{i \in n} Q_i(x_i)$. Variational inference then solves for $Q$ by minimizing the KL divergence of $P$ from $Q$(see Eq. (2.9)).The work of [151] showed that the inference can be performed with efficient bilateral filtering [15, 234, 251, 7] operations. Specifially, mean-field inference results in a fixed point equation which can be solved iteratively $t = 0, 1, \ldots$ to update the marginal distributions $Q_i$:

$$Q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp(-\psi_u(x_i) - \sum_{l \in \mathcal{L}} \underbrace{\sum_{j \neq i} \psi_p^{ij}(x_i, l) Q_j^t(l)}_{\text{bilateral filtering}}), \qquad (2.12)$$

where $Z_i$ denotes a normalization constant and can be easily computed as $Q_i$ is a single dimensional distribution. Although we used semantic segmentation task for illustration purposes, DenseCRFs are shown to be useful for tackling other tasks such as material segmentation [25], optical flow estimation [239] and intrinsic image decomposition [24].

One of the fundamental limitations of the existing use of DenseCRFs is the confinement of pairwise potentials $\psi_p^{ij}(y_i, y_j)$ to be Gaussian as bilateral filtering is traditionally implemented with a Gaussian kernel. In Chapter 5, we show how we can learn a more

general form of bilateral filters and apply that technique for learning pairwise edge potentials in DenseCRF.

### 2.2.3 Advantages and Limitations

This generative modeling view is appealing as it is relatively easy to incorporate our knowledge of physics and light transport into models and was advocated since the late 1970 [120, 104, 280, 188, 179, 274]. For example, the knowledge of how light reflects on objects with different material properties or the knowledge of how roads and buildings are structured are relatively easy to incorporate into generative models. Due to incorporation of strong prior knowledge into the systems, generative models usually work better when there is little or no data available for a particular problem. Since a single generative model can model different world and image characteristics, it can be used for many different applications. In addition, it is easier to diagnose the flaws in generative model as most of the model is manually designed.

Despite its intuitive appeal and advantages, in practice, generative models are used only for a few vision problems. The few successes of the idea have been in limited settings. In the successful examples, either the generative model was restricted to few high-level latent variables, e.g., [195], or restricted to a set of image transformations in a fixed reference frame, e.g., [29], or it modeled only a limited aspect such as object shape masks [74], or the generative model was merely used to generate training data for a discriminative model [228, 90, 215, 214, 222]. With all its intuitive appeal, its beauty and simplicity, it is fair to say that the track record of generative models in computer vision is poor. As a result, the field of computer vision is now dominated by efficient but data-hungry discriminative models, the use of empirical risk minimization for learning, and energy minimization on heuristic objective functions for inference.

Why did generative models not succeed? There are two key problems that need to be addressed, the design of an accurate generative model, and the inference therein. The first key problem which is the design of accurate generative model is partly addressed by recent advances in graphics. Although modern graphics provide rendering with stunning level of realism, priors of world parameters are difficult to characterize. This results in complex priors together with more complex forward models for accurate generative models which in turn results in difficult inference.

This brings us to the second key problem in the generative world view which is the difficulty of posterior inference at test time. This difficulty stems from a number of reasons: *first*, the target variable $\mathbf{y}$ is typically high-dimensional and so is the posterior. *Second*, given $\mathbf{y}$, the image formation process realizes complex and *dynamic* dependency structures, for example when objects occlude or self-occlude each other. These intrinsic ambiguities result in multi-modal posterior distributions. *Third*, while most renderers are real-time, each simulation of the forward process is expensive and prevents exhaustive enumeration. Overall, the limitations of generative approaches out-weigh their advantages making them not succeed in building practical computer vision systems.

Despite these limitations, we still believe in the usefulness of generative models in computer vision, but argue that we need to leverage existing discriminative or even heuristic computer vision methods for alleviating some of the difficulties in the posterior inference. Inference techniques proposed in this thesis are steps in this direction.

## 2.3  Discriminative Vision Models

With the advances in internet and image capturing technology, there is an explosive growth of visual data during the last few years. Moreover, presence of crowd-sourcing platforms like 'Amazon Mechanical Turk' [1] make it easier to annotate large amounts of data by millions of people. Discriminative models directly model the contingency of world properties on the observed data $P(\mathbf{y}|\mathbf{x})$. Unlike generative models, discriminative models are task-specific and learning takes the central role in defining the model. Discriminative models comprise of functions directly approximating the posterior distribution $P(\mathbf{y}|\mathbf{x}, \theta)$, where $\theta$ denote the parameters of the model. Supervised learning with annotated training data is usually employed to fit the model parameters to the given task. Since discriminative models directly characterize the posterior distribution, inference is reduced to simple evaluation of the model.

Due to the availability of large amounts of training data and the computing power that can handle rich high-capacity models, discriminative models have been very successful in many vision problems. In addition, inference is fast since this involves a simple evaluation of the model. This makes discriminative models particularly attractive for many practical applications. Many mathematical functions that are rich enough to capture the relationship between the observation and target variables can be used as discriminative models. Hence, many types of discriminative models have been used in the computer vision literature.

Discriminative models are traditionally partitioned into two modules: *feature extraction* and *prediction* modules. Before the advent of modern convolutional neural networks (CNNs), these two components are studied separately in the literature. We briefly discuss these two components in the discriminative models.

**Feature Extraction:**   Depending on the type of vision task, features are extracted either at all pixels (points) in the observed data or only at some key points. For example, registering two images taken from different view-points requires finding corresponding points (key points) in each image and then matching. For such tasks, an additional step of key point detection is required before feature computation. For image classification, a single feature vector is extracted for the entire image.

An ideal feature representation should be compact, efficient to compute and invariant to specific transformations. As an example, for semantic segmentation, features should be invariant to intra-class variations such as illumination, scale, rotation, object articulations, etc., while being sensitive to changes across different semantic categories. Several

feature extraction schemes have been proposed in the vision literature, most of them are hand crafted. Some popular choices include SIFT [178], HoG [23], SURF [61], DAISY [250], etc. Models for feature extraction and prediction are plentiful and discussing all of them is outside the scope of this thesis. With the recent advances in CNNs, feature extraction is coupled with prediction which are learned together end-to-end.

**Prediction:**   Once the image features are extracted, the task is to estimate the posterior distribution $P(\mathbf{y}|\mathbf{f}(\mathbf{x}))$, where $\mathbf{f}(\mathbf{x})$ denotes the features. A common strategy is to learn a rich parametric or non-parametric model with supervised learning techniques. This makes the availability of training data crucial for discriminative approaches. Several learning based prediction models have become popular in tackling vision tasks including support vector machines (SVM) [58], boosting [218, 84], random forests [35, 118], deep convolutional neural networks [158], etc. Refer to [87] for a review of different prediction techniques. Next, we briefly review random forests and CNN models as we either make use of or propose extensions to these models in this thesis.

## 2.3.1 Random Forests

Random forests [35, 118] are an ensemble of $K$ randomly trained prediction (classification or regression) trees, where each tree $T(\mathbf{y}|\mathbf{f}(\mathbf{x}), \theta^k)$ represents a non-linear function approximating the posterior $P(\mathbf{y}|\mathbf{f}(\mathbf{x}))$. The trees are typically binary trees and can be viewed as performing a discriminative hierarchical clustering of the feature space. And a simple model fit (e.g., linear model) is used in each cluster.

Trees are grown incrementally from the root node to the leaves and each node represents a partition of the feature space. These partitions can be any linear or non-linear functions, but the simple axis-aligned partitions are the most used ones due to their simplicity and efficiency. For simplicity, let us assume the partition functions are axis-aligned. At each node, a feature $\kappa$ and its split value $\tau$ are chosen to split the feature space, so as to minimize an energy function $E$. Let us consider training the $j^{th}$ node in a $k^{th}$ tree. Let all the data points falling in that node be $\mathcal{S}_j$ (due to splitting of its ancestor nodes) and $\mathcal{T}_j$ denotes the discrete set of *randomly* selected feature axes $\{(\kappa, \tau)_i\}$ (feature indices and their corresponding values) for the node $j$. Training the $j^{th}$ node corresponds to choosing the optimal split $\theta_j^k \in \mathcal{T}_j$ among the randomly chosen splits that minimizes an energy function $E$:

$$\theta_j^k = \operatorname*{argmin}_{\gamma \in \mathcal{T}_j} E(\mathcal{S}_j, \gamma) \qquad (2.13)$$

Depending on the type of task and data, different energy functions $E$ are used. Each split $\gamma$ partitions the training data $\mathcal{S}_j$ in the node $j$ into two parts $\mathcal{S}_j^L$ and $\mathcal{S}_j^R$ which are assigned to left and right child nodes respectively. A common energy function measures how well a regression/classification model fit the data in each of the left and right child

nodes created by a split $\gamma$:

$$E(\mathcal{S}_j, \gamma) = -(M(\mathcal{S}_j^L, \beta) + M(\mathcal{S}_j^R, \beta)). \qquad (2.14)$$

Where $M(\mathcal{S}_j, \beta)$ denotes the model likelihood i.e., how well the model with parameters $\beta$ can explain the data $\mathcal{S}_j$. For example, in the case of regression tasks, $M$ can be a linear regression fit and in the case of classification (like in semantic segmentation), $M$ can be the classification accuracy. Like this, the trees are recursively grown by splitting the leaf nodes into left and right child nodes. The set of all node splits $\theta^k = \{\theta_j^k\}_{j=1,\cdots,J}$ represents the parameters of the $k^{th}$ tree. Once a tree is trained, a simple prediction model is fitted to the data in the leaf nodes. A deep tree might overfit the data and a shallow tree would under-fit the data and miss important structure. Restricting the tree size corresponds to regularizing the model and size should be adaptively chosen based on the training data. Some training stopping criteria include setting the maximum depth of the trees; minimum number of data points in each node; a threshold for energy function $E$, etc.

Random forests are distinguished from other tree-based supervised learning techniques such as boosted decision trees, by the way different trees are trained in a forest. Each tree in a random forest is trained independently and randomness is added either in terms of choosing a random subset of training data for each tree (called *bagging* [34]) and/or randomly choosing the split candidates (feature indices and their values) at each node. Typically, the estimates across the trees $T(\mathbf{y}|\mathbf{f}(\mathbf{x}), \theta^k)$ are averaged to get the final model $P(\mathbf{y}|\mathbf{f}(\mathbf{x}))$:

$$P(\mathbf{y}|\mathbf{f}(\mathbf{x})) = \frac{1}{K} \sum_k T(\mathbf{y}|\mathbf{f}(\mathbf{x}), \theta^k). \qquad (2.15)$$

Due to the randomness, different trees are identically distributed resulting in a low-variance estimate when the final estimate is taken as the average across the trees. Random forests are highly flexible and several different types of models are conceivable with using a combination of different splitting criteria. Due to their simplicity and flexibility, random forests have become a popular choice for supervised learning in vision. Random forests are easy to implement and train. Also, they can be easily adapted to a wide range of classification and regression tasks with relatively simple changes to the model. Moreover, they are non-parametric in nature with the ability to consume large amounts of training data. Random forests are successfully applied for vision tasks such as human pose estimation [228], semantic segmentation [229], etc. In the case of semantic segmentation, a popular model is to extract TextonBoost features [230] at each pixel and then train a random forest classifier to predict the class label at each pixel. One of the crucial advantages of random forests with respect to neural networks is that the loss function $E$ need not be differentiable. In Chapter 3, we use random forest models to improve inference in inverse graphics via our informed sampler approach. In Chapter 4, we use random forests for predicting messages resulting in improved variational inference in

layered graphical models. Refer to [59] for a comprehensive overview of random forests and their applications in computer vision and medical image analysis.

## 2.3.2 Convolutional Neural Networks

Neural networks are a class of models with complex parametric non-linear functions relating the input $\mathbf{x}$ to the target $\mathbf{y}$. The complex non-linear function is usually realized by stacking a series of simple and differentiable linear and non-linear functions:

$$P(\mathbf{y}|\mathbf{x}, \theta) = \mathcal{F}_1(\mathcal{F}_2(\cdots \mathcal{F}_k(\mathbf{x}, \theta_k) \cdots, \theta_2), \theta_1). \quad (2.16)$$

Learning involves finding the parameters $\{\theta_1, \theta_2, \cdots, \theta_k\}$ that best approximates the desired relationship between the input and target variables. The component functions are usually simple linear functions such as convolutions $\mathcal{F}(\mathbf{s}, \theta) = \mathbf{W}(\theta)\mathbf{s} + b$ (where $\mathbf{s} \in \mathbb{R}^q$, $\mathbf{W} \in \mathbb{R}^{p \times q}$) interleaved with simple non-linear functions such as rectified linear units (ReLU) $\mathcal{F}(\mathbf{s}) = max(0, \mathbf{s})$. A linear function together with a non-linearity is usually called a single layer in the network. Intermediate layers in a neural network are also called *hidden* layers and the number of units in intermediate layers determine the *width* of the network. A theoretical result [60, 121] is that any complex continuous function can be approximated by a simple two layered neural network, given sufficient number of intermediate units (width of the network). From a practical point of view, neural networks are attractive because of their fast inference (simple forward pass through the network) and an end-to-end prediction (going from input to output variables without the intermediate handcrafted feature extraction) capabilities.

Convolutional neural networks (CNN) are a special class of neural networks tailored for processing 2D or higher dimensional visual data on a grid. The main characteristic of CNNs is the use of spatial convolutions instead of *fully-connected* matrix-vector multiplications for building linear functions. This greatly reduces the amount of parameters due to parameter sharing across different spatial locations and speeds up the network computation and training. One of the main hurdles for the success of CNNs was the lack of computational resources required to train models with millions of parameters. Recent availability of large datasets together with efficient model and training implementations in GPUs made it possible to successfully apply CNNs to real-world vision tasks. Since CNNs typically have millions of parameters, they are highly prone to overfit the training data. Advances in simple yet powerful regularization techniques (such as DropOut [235]) are another reason for the successful deployment of CNN models. Currently, CNNs are state-of-the-art in many traditional vision problems such as image classification [111, 152], object detection [97, 209, 211], semantic segmentation [176, 53], etc.

CNN architectures are typically composed of the following layers: Convolution, pooling, non-linearity, fully-connected (FC) and loss layers. Convolution layers are simple spatial convolutions, pooling layers do spatial downsampling and FC layers connect each

Figure 2.5: **Sample CNN architecture for character recognition.** LeNet-7 [159] architecture generally used for character recognition (used for Assamese character recognition in Chapter 5). 'C$n$' corresponds to convolution layer with $n \times n$ filters; 'MP$n$' corresponds to max-pooling layer with window size $n$; 'IP' corresponds to fully-connected inner-product layer; 'ReLU' and 'TanH' corresponds to rectified linear units and tanh non-linear layers and 'Softmax' layer produces probabilities for 183 output classes.

output unit to all input units. Non-linear layers are simple yet important functions that model non-linearities in the CNN model. Some popular non-linear functions include ReLU, TanH, sigmoid function, etc. Loss layers are problem specific layers that are used at the end of the network and implement the differentiable empirical loss $\sigma(\hat{\mathbf{y}}, \mathbf{y}^*)$ between the predicted target $\hat{\mathbf{y}}$ and the ground truth target $\mathbf{y}^*$. Figure 2.5 shows a sample CNN architecture generally used for character recognition. The input is a grayscale image $\mathbf{x}$ with the size $96 \times 96$ and the output is a vector of probabilities for each class $P(\mathbf{y}|\mathbf{x})$. Also shown are the sizes of intermediate CNN representations.

Training the parameters $\theta_k$ of each layer involves back-propagating the empirical loss from the loss layers backwards to the early CNN layers. To avoid over-fitting to the training data, the loss is usually augmented with a regularization over the network parameters. The optimization objective for a given dataset with $m$ training instances is given as an average loss $L$:

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sigma(\hat{\mathbf{y}}_i, \mathbf{y}_i^*) + \lambda r(\theta), \tag{2.17}$$

where $r$ denotes the regularization over the parameters $\theta$ with weight $\lambda$. Then the parameters $\theta$ are updated using gradient descent methods such as stochastic gradient

descent (SGD), AdaDelta [275], Adam [144], etc. The parameter update steps to update a single parameter $\theta^i \in \theta$ in SGD are given as:

$$
\begin{aligned}
v_{t+1} &= \mu v_t - \gamma \nabla L(\theta_t^i) \\
\theta_{t+1}^i &= \theta_t + v_{t+1}
\end{aligned}
\tag{2.18}
$$

where $v, \gamma, \mu \in \mathbb{R}$, $v_t$ denotes the parameter update in the previous step and $\nabla L(\theta_t^i)$ denotes the gradient of loss $L$ with respect to the parameter $\theta^i$. Thus the parameter update $v_{t+1}$ is a weighted combination of the previous update and the negative gradient of loss $L$. The weights $\mu$ and $\gamma$ are called *momentum* and *learning rate* respectively which are generally chosen to obtain good performance on a given validation data. In practice, since the size of the dataset $m$ is large, only a small subset (batch) of dataset is used for computing the loss and updating parameters in each step.

Instead of computing the gradients of loss $L$ with respect to all the network parameters in one step, the gradients are back-propagated across the layers. Thus, one of the fundamental requirements for a component function (except the first layer) in CNNs is that it should be differentiable with respect to both its inputs and its parameters. Once the network is trained, inference is a simple forward pass through the network. Like in many discriminative models, inference in CNNs amounts to evaluation of the model.

In Chapter 5, we generalize the standard spatial convolutions found in CNNs to sparse high-dimensional filters and in Chapter 7, we propose an efficient CNN module for long-range spatial propagation of information across intermediate CNN representations. There are several other types of neural networks, such as recurrent neural networks, that are currently being used in the computer vision community. Refer to [26, 219] for more details regarding CNNs or neural networks in general.

### 2.3.3 Advantages and Limitations

Discriminative models are mainly data-driven methods where inference amounts to simple evaluation of the model. In general, discriminative models are fast, robust to model mismatch and are also high performing when trained with enough amounts of data. Discriminative models are attractive because of their practical utility and also their flexibility in terms of being able to use same model architecture and training for different vision tasks.

On the other hand, discriminative models also have several limitations.

- Discriminative models are data hungry and typically fail to work where there is little data available. Availability of large datasets for several vision tasks and online annotation tools like Amazon Mechanical Turk [1] help in mitigating this limitation.

Figure 2.6: **Sample CNN architecture for semantic segmentation.** Semantic segmentation CNN architectures typically consists of convolution (Conv.), pooling (Pool) and $1 \times 1$ convolution layers (FC) interleaved with non-linearities (ReLU). The use of pooling results in lower resolution CNN output which is generally up-sampled with either interpolation, deconvolution and/or CRF techniques. CRF techniques also help in incorporating prior knowledge about semantic segmentation.

- Since discriminative models tend to have a large number of parameters which are directly learned from data, when the performance is not as expected, it is difficult to find the cause of the problem and accordingly modify the models. As a result, there are no guaranteed ways to find the right model architecture for a given task. These problems are generally handled with either regularizations on model complexity or with trial-and-error strategy on various model architectures.

- One of the fundamental limitations of discriminative models is the lack of key approaches to inject prior knowledge into the models. This is especially true in the case of end-to-end trained models like CNNs. In the case of hand crafted features, we can inject some prior context in the form of image or pixel features. It is not easy to inject the knowledge of generative models into discriminative approaches such as CNNs. For example, in the case of semantic segmentation, post-processing steps such as DenseCRFs are generally employed to model the relationship (prior knowledge) between the pixels and output labels. Figure 2.6 shows a prominent CNN architecture for semantic segmentation.

- Discriminative models are task-specific and a single trained model can not be easily transferred to other vision problems. Although several transfer learning techniques [196] exists that can help transfer the knowledge across different discriminative models, they are generally task specific and have limited success. One of the main advantages of CNNs, in comparison to other discriminative models, is that a CNN trained on image classification task is shown to perform reasonably well on other related tasks such as semantic segmentation, object detection etc. with only minor adaptions to the new task.

Recently, hybrid models combining generative and discriminative models are pro-

posed to alleviate some of the limitations in both and make use of their complementary advantages. We will discuss these models in the next section.

## 2.4 Combining Generative and Discriminative Models

As we have argued in the previous sections, generative and discriminative models have complementary advantages and limitations. Generative models have the advantage of incorporating prior knowledge while being slow; whereas discriminative models are fast and robust but it is difficult to incorporate prior knowledge. Typically, generative models suffer from high bias due to model mismatch, whereas discriminative models suffer from higher variance. In general, discriminative models work well and are robust to model mismatch when the available annotated training data is large. If the available data is small in comparison to the required model complexity, we need ways to constrain model parameters with the use of prior knowledge. Generative models provide principled ways to incorporate such prior knowledge and can even make use of unlabelled data which is generally abundant. The work of [192] is one of the first comparative studies on generative and discriminative models resulting in the common knowledge of using discriminative models when the data is abundant, otherwise use generative approach for a given problem.

We hypothesize that combining generative and discriminative models can leverage the advantages in both. At the same time, combining these complementary models can also bring forward the limitations in both. The generative and discriminative models are often studied in isolation, but during the past decade, several synergistic combinations of generative and discriminative models have been proposed.

There are 3 ways in which generative and discriminative models can be combined. 1. Use a generative model to improve the model or the inference in the discriminative model (indicated as 'Generative $\rightarrow$ Discriminative'); 2. Use a discriminative model for improving the model and/or inference in the generative model (indicated as 'Discriminative $\rightarrow$ Generative'); and 3. Hybrid generative and discriminative models (indicated as 'Generative $\leftrightarrow$ Discriminative').

### Generative $\rightarrow$ Discriminative

One way to use generative models for improving discriminative models is by feature extraction using generative models. The work of [127] showed that the gradients of the generative models can be used as features in discriminative models. The gradients of a generative model are called 'Fisher vectors' and are particularly useful for building kernel functions (Fisher kernels) that can be used in kernel based techniques such as SVMs.

Another popular way to incorporate generative prior knowledge is to provide prior constraints while training discriminative models. For instance, CNN models can be

trained with extra loss layers encoding the prior relationship between the output variables. The overall training loss is a combination of the discriminative prediction loss and also a generative prior loss. A related strategy for training CNNs is to first train a discriminative CNN using generative prior loss with large amounts of unlabelled data, and then fine-tune the network using discriminative prediction loss with limited labelled data. Instead of training a single discriminative model with prior constraints, [256] proposed to train a sequence of discriminative predictors, each taking as input not only the input features but also features from previous stage predictions. This way, it is easy to incorporate the prior constraints on output target variables using features extracted on predictions. This technique is called 'Auto-Context' [256] and the sequence of predictors is usually trained with stacked generalization method [268]. Despite being simple, auto-context method is shown to be powerful and useful in many vision problems (for e.g., [256, 137, 132]).

More recently, structured prediction layers [126, 279, 220, 48] are introduced into discriminative CNN frameworks. These layers are mainly adapted from the models and inference techniques in generative models. For example, in the case of semantic segmentation CNNs, mean-field inference in fully-connected CRFs can be formulated as recurrent neural network modules [69, 279] and is used to augment the existing CNN architectures resulting in better performance. The work of [48] proposed a way to incorporate Gaussian CRFs into end-to-end trained semantic segmentation CNNs. In Chapter 5, we generalize the standard spatial convolutions in CNNs to sparse high-dimensional filters and show it can be used to incorporate structured prior knowledge into CNNs resulting in better model and inference. In Chapter 7, we propose a specialized structured prediction module to be used in CNNs for dense pixel prediction tasks such as semantic segmentation.

**Discriminative → Generative**

Although generative models provide an elegant formalism to encode prior knowledge about the problem, their use is mainly hampered by the difficulty in the posterior inference. Since discriminative models directly characterize the posterior distribution, they have the potential to be useful for inference in a corresponding generative model.

Since many of the generative models require approximate Bayesian inference for estimating the posterior distribution, some components of the Bayesian inference can be completely replaced with discriminative models. *Inference machines* [217] are a successful example of such technique. Inference machines pose the message passing inference in a given generative model as a sequence of computations that can be performed efficiently by training discriminative models like random forests. Instead of learning complex potential functions and computing messages between the variables, discriminative predictors that directly learn to pass the messages are proposed. This technique is shown to perform well on real world tasks [217, 207, 224] such as human pose estimation, 3D surface layout estimation and 3D point cloud estimation. Inference machines help

bridging the gap between the message passing and random forest techniques. Similar technique [113] is also shown to be useful to predict messages for expectation propagation [184] inference in generative models. More recently, [171] proposed to use discriminative deep learning models for predicting messages in message passing inference.

By completely replacing the components of Bayesian inference with discriminative predictors, we lose the theoretical guarantees from the original inference techniques. However, discriminative models can still be used to improve the inference process. Data driven Markov chain Monte Carlo (DDMCMC) [257] methods leverage discriminative models to speed up the MCMC inference. DDMCMC methods have been used in image segmentation [257], object recognition [281], and human pose estimation [162]. In Chapters 3 and 4, we propose principal techniques for leveraging discriminative models for Bayesian inference in inverse graphics and layered graphical models respectively.

Another way of using discriminative models to improve generative approaches is to use discriminative prediction loss for training the generative model parameters. This is called 'discriminatively training generative models' [30, 119, 272] and is akin to using a generative prior loss for training discriminative models. Such models are also called hybrid models [157] (discussed more below) if different parameters are used for defining discriminative and generative models.

### Generative ↔ Discriminative

It is possible to define both discriminative and generative models for the same task and train them together. This synergistic training can help in a better model fit in both. With such hybrid models, it is possible to train with both unlabelled and labelled data together [157, 185].

Recent advances in deep learning showed that neural network models can also be used as good approximators for generative models of images (for e.g., [71, 103, 249]). Thus, it is possible to define a hybrid model with different neural networks approximating the corresponding generative and discriminative models for a task, and then train them together. One popular model in this category is 'Auto-encoding variational Bayes' [145, 213]. Here, a generative model with exponential family distributions is approximated with a neural network. At the same time, variational Bayesian posterior inference in that model is approximated with a different neural network. Both generative and discriminative (inference) networks are trained by minimizing the variational lower bound (Eq. (2.9)). The work of [73] uses such hybrid models with recurrent neural networks and an attention mechanism to tackle vision problems involving multiple unknown number of objects in an image. These models are shown to perform well on small scale vision problems such as character recognition and are not scaled for tackling mainstream vision problems. The formulation of such hybrid models is elegant and has potential to be useful for many vision tasks.

Very recently, in a similar spirit to auto-context, [269] proposes to learn a top-down

CNN for capturing the contextual relationships between the target variables. The top-down generative CNN learns to predict the target variables from the surrounding target variables (context). This top-down CNN is then coupled with original discriminative CNN to serve as top-down constraints for intermediate CNN representations. As an advantage over the auto-context framework where different models are learned at different stages, a single discriminative model is learned and shown to be sufficient.

Hybrid generative and discriminative CNN models are a very active area of research with different architectures being proposed frequently. We only discussed a few model architectures here. It is plausible that in the near future, hybrid generative and discriminative models dominate the field of computer vision.

## 2.5  Discussion and Conclusions

In this chapter, we have discussed various generative and discriminative computer vision models. Models form the core of any computer vision system, we choose to discuss some prominent models in relation to this thesis while highlighting the advantages and limitations in popular generative and discriminative models.

Generative models are conceptually elegant to incorporate prior knowledge about the task while their use is mainly hampered by the difficulty of posterior inference. Discriminative models, on the other hand, are robust to model mismatch while being fast but require large amount of labelled data and there is a lack of standard approaches for incorporating prior knowledge into them. Hybrid generative and discriminative models, discussed in the previous section, try to bridge the gap between these two complementary approaches.

The main aim of this thesis is to improve inference with in various computer vision models. In Part I of the thesis, we concentrate on improving inference in generative vision models. We do this by learning separate discriminative models and propose algorithms for better inference in prominent generative models in vision namely inverse graphics models (Chapter 3) and layered graphical models (Chapter 4).

In Part II of the thesis, we concentrate on improving inference in discriminative vision models. Since inference is simple evaluation of the model in discriminative models, we propose techniques for modifying the model itself enabling the introduction of prior knowledge into CNN models. Specifically, we generalize the standard spatial convolutions in prominent CNN models to sparse high-dimensional filtering (Chapter 5) and then propose a neural network approach for propagating information across video frames (Chapter 6). In Chapter 7, we propose a new CNN module that can be added to existing segmentation CNN architectures that helps in image adaptive filtering of intermediate CNN units.

# Part I

# Inference in Generative Vision Models

# Chapter 3

# The Informed Sampler

In the previous chapter, we briefly discussed the generative models (Section 2.2) with their advantages and limitations. With all its intuitive appeal, beauty and simplicity, it is fair to say that the track record of generative models in computer vision is poor, which is mainly due to the difficulty of posterior inference. As a result the computer vision community has favored efficient discriminative approaches. We still believe in the usefulness of generative models in computer vision, but argue that we need to leverage existing discriminative or even heuristic computer vision methods. In this chapter, we implement this idea in a principled way with an *informed sampler*, which is a mixture sampling technique, and in careful experiments demonstrate its use on challenging generative models which contain renderer programs as their components. We concentrate on posterior inference in 'inverse graphics' models which is briefly described in Section 2.2.1. With experiments on diverse 'inverse graphics' models, we show that the informed sampler, using simple discriminative proposals based on existing computer vision technology, achieves significant improvements of inference.

## 3.1 Introduction

As discussed in Section 2.2.1, modern computer graphic systems that leverage dedicated hardware setups produce a stunning level of realism with high frame rates. We believe that these systems will find its way in the design of generative models and will open up exciting modeling opportunities. This observation motivates the research question of this chapter, the design of a general inference technique for efficient posterior inference in accurate computer graphics systems. As such it can be understood as an instance of *Inverse Graphics* [22], which is briefly discussed in Section 2.2.1 and illustrated in Fig. 3.1 with one of our applications.

The key problem in the 'inverse graphics' is the difficulty of posterior inference at test-time. This difficulty stems from a number of reasons as outlined in Section 2.2.3. Our aim in this work is to devise an inference technique that is general and allow reuse in several different models and novel scenarios. On the other hand we want to maintain correctness in terms of the probabilistic estimates that they produce. One way to improve on inference efficiency in generative models is to leverage existing computer vision fea-

Figure 3.1: **An example 'inverse graphics' problem.** A graphics engine renders a 3D body mesh and a depth image using an artificial camera. By 'Inverse Graphics', we refer to the process of estimating the posterior probability over possible bodies given the depth image.

tures and discriminative models. In this chapter, we propose the *informed sampler*, a Markov chain Monte Carlo (MCMC) method with discriminative proposal distributions. It can be understood as an instance of a data-driven MCMC method [281], and our aim is to design a method that is general enough such that it can be applied across different problems and is not tailored to a particular application.

During sampling, the informed sampler leverages computer vision features and discriminative models to make informed proposals for the state of latent variables and these proposals are accepted or rejected based on the generative model. The informed sampler is simple and easy to implement, but it enables inference in generative models that were out of reach for current *uninformed* samplers. We demonstrate this claim on challenging models that incorporate rendering engines, object occlusion, ill-posedness, and multi-modality. We carefully assess convergence statistics for the samplers to investigate their correctness about the probabilistic estimates. Our informed sampler uses existing computer vision technology such as histogram-of-gradients features (HoG) [61], and the OpenCV library, [33], to produce informed proposals. Likewise one of our models is an existing computer vision model, the *BlendSCAPE* model, a parametric model of human bodies [117].

In Section 3.2, we discuss related work and explain our informed sampler approach in Section 3.3. Section 3.4 presents baseline methods and experimental setup. Then we present experimental analysis of informed sampler with three diverse problems of estimating camera extrinsics (Section 3.5.1), occlusion reasoning (Section 3.5.2) and estimating body shape (Section 3.5.3). We conclude with a discussion of future work in Section 3.6.

# 3.2 Related Work

This work stands at the intersection of computer vision, computer graphics, and machine learning; it builds on previous approaches we will discuss below.

There is a vast literature on approaches to solve computer vision problems by means of generative models. We mention some works that also use an accurate graphics process as a generative model. This includes applications such as indoor scene understanding [65], human pose estimation [162], hand pose estimation [63] and many more. Most of these works are however interested in inferring MAP solutions, rather than the full posterior distribution.

Our method is similar in spirit to a *Data Driven Markov Chain Monte Carlo* (DDM-CMC) methods that use a bottom-up approach to help convergence of MCMC sampling. DDMCMC methods have been used in image segmentation [257], object recognition [281], and human pose estimation [162]. The idea of making Markov samplers data dependent is very general, but in the works mentioned above, lead to highly problem specific implementations, mostly using approximate likelihood functions. Since these methods provide specialized solutions for a particular problem, they are not easily transferable to new problems. In contrast, we aim to provide a simple, yet efficient and general inference technique for problems where an accurate generative model exists. Because our method is general we believe that it is easy to adapt to a variety of new models and tasks.

The idea to invert graphics [22] in order to understand scenes also has roots in the computer graphics community under the term 'inverse rendering'. The goal of inverse rendering however is to derive a direct mathematical model for the forward light transport process and then to analytically invert it. The work of [208] falls in this category. The authors formulate the light reflection problem as a convolution, to then cast the inverse light transport problem as a deconvolution. While this is a very elegant way to pose the problem, it requires a specification of the inverse process, a requirement generative modeling approaches try to circumvent.

Our approach can also be viewed as an instance of a probabilistic programming approach. In the recent work of [179], the authors combine graphics modules in a probabilistic programming language to formulate an approximate Bayesian computation. Inference is then implemented using Metropolis-Hastings (MH) sampling. This approach is appealing in its generality and elegance, however we show that for our graphics problems, a plain MH sampling approach is not sufficient to achieve reliable inference and that our proposed informed sampler can achieve robust convergence in these challenging models. Another piece of work from [236] is similar to our proposed inference method in that the knowledge about the forward process is learned as "stochastic inverses", then applied for MCMC sampling in a Bayesian network. In the present work, we devise an MCMC sampler that works in both a multi-modal problem as well as for inverting an existing piece of image rendering code. In summary, our method can be understood in a similar context as the above-mentioned papers, including [179].

## 3.3 The Informed Sampler

In general, inference about the posterior distribution is challenging because for a complex model $p(\mathbf{x}|\mathbf{y})$ no closed-form simplifications can be made. This is especially true in the case that we consider, where $p(\mathbf{x}|\mathbf{y})$ corresponds to a graphics engine rendering images. Despite this apparent complexity, we observe the following: for many computer vision applications there exist well performing discriminative approaches, which, given the image, predict some target variables $\mathbf{y}$ or distributions thereof. These do not correspond to the posterior distribution that we are interested in, but, *intuitively* the availability of discriminative inference methods should make the task of inferring $p(\mathbf{y}|\mathbf{x})$ easier. *Furthermore* a physically accurate generative model can be used in an offline stage prior to inference to generate as many samples as we would like or can afford computationally. Again, *intuitively* this should allow us to prepare and summarize useful information about the distribution in order to accelerate the test-time inference.

Concretely, in our case we will use a discriminative method to provide a global density $T_G(\mathbf{y}|\mathbf{x})$, which we then use in a valid MCMC inference method. The standard Metropolis-Hasting Markov Chain Monte Carlo (MCMC) is already described in Section 2.2.1 of the previous chapter, where in each time-step, a proposal is made with a proposal distribution which is then either accepted or rejected based on the acceptance probability:

1. Propose a transition using a *proposal distribution T* and the current state $\mathbf{y}_t$

$$\bar{\mathbf{y}} \sim T(\cdot|\mathbf{y}_t)$$

2. Accept or reject the transition based on Metropolis Hastings (MH) acceptance rule:

$$\mathbf{y}_{t+1} = \begin{cases} \bar{\mathbf{y}}, & \text{rand}(0,1) < \min\left(1, \frac{\pi(\bar{\mathbf{y}})T(\bar{\mathbf{y}} \rightarrow \mathbf{y}_t)}{\pi(\mathbf{y}_t)T(\mathbf{y}_t \rightarrow \bar{\mathbf{y}})}\right), \\ \mathbf{y}_t, & \text{otherwise.} \end{cases}$$

Refer to Section 2.2.1 for more details about MCMC sampling. Different MCMC techniques mainly differ in the type of the proposal distribution $T$. Next, we describe our informed proposal distribution which we use in standard Metropolis-Hastings sampling resulting in our proposed 'Informed Sampler' technique.

### 3.3.1 Informed Proposal Distribution

We use a common mixture kernel for Metropolis-Hastings (MH) sampling. Given the present target sample $\mathbf{y}_t$, the informed proposal distribution for MH sampling is given as:

$$T_\alpha(\cdot|\mathbf{x},\mathbf{y}_t) = \alpha\, T_L(\cdot|\mathbf{y}_t) + (1-\alpha)\, T_G(\cdot|\mathbf{x}). \tag{3.1}$$

Here $T_L$ is an ordinary *local* proposal distribution, for example a multivariate Normal distribution centered around the current sample $\mathbf{y}_t$, and $T_G$ is a *global* proposal distribution independent of the current state. We inject knowledge by conditioning the global proposal distribution $T_G$ on the image observation $\mathbf{x}$. We learn the informed proposal $T_G(\cdot|\mathbf{x})$ discriminatively in an offline training stage using a non-parametric density estimator described below.

The mixture parameter $\alpha \in [0, 1]$ controls the contribution of each proposal, for $\alpha = 1$ we recover MH. For $\alpha = 0$ the proposal $T_\alpha$ would be identical to $T_G(\cdot|\mathbf{x})$ and the resulting Metropolis sampler would be a valid Metropolized independence sampler [174]. With $\alpha = 0$, we call this baseline method 'Informed Independent MH' (INF-INDMH). For intermediate values, $\alpha \in (0, 1)$, we combine local with global moves in a valid Markov chain. We call this method 'Informed Metropolis Hastings' (INF-MH).

## 3.3.2 Discriminatively Learning $T_G$

The key step in the construction of $T_G$ is to include some discriminative information about the sample $\mathbf{x}$. Ideally we would hope to have $T_G$ propose global moves which improve mixing and even allow mixing between multiple modes, whereas the local proposal $T_L$ is responsible for exploring the density locally. To see that this is possible in principle, consider the case of a perfect global proposal where $T_G$ matches the true posterior distribution, that is, $T_G(\mathbf{y}|\mathbf{x}) = P(\mathbf{y}|\mathbf{x})$. In this case, we would get independent samples with $\alpha = 0$ because every proposal is accepted. In practice $T_G$ is only an approximation to the true posterior $P(\mathbf{y}|\mathbf{x})$. If the approximation is good enough then the mixture of local and global proposals will have a high acceptance rate and explore the density rapidly.

In principle, we can use any conditional density estimation technique for learning a proposal $T_G$ from samples. Typically high-dimensional density estimation is difficult and even more so in the conditional case; however, in our case we do have the true generating process available to provide example pairs $(\mathbf{y}, \mathbf{x})$. Therefore we use a simple but scalable non-parametric density estimation method based on clustering a feature representation of the observed image, $v(\mathbf{x}) \in \mathbb{R}^d$. For each cluster we then estimate an unconditional density over $\mathbf{y}$ using kernel density estimation (KDE). We chose this simple setup since it can easily be reused in many different scenarios, in the experiments we solve diverse problems using the same method. This method yields a valid transition kernel for which detailed balance holds. In addition to the KDE estimate for the global transition kernel we also experimented with a random forest approach that maps the observations to transition kernels $T_G$. More details will be given in Section 3.5.3.

For the feature representation, we leverage successful discriminative features and heuristics developed in the computer vision community. Different task specific feature representations can be used in order to provide invariance to small changes in $\mathbf{y}$ and to nuisance parameters. The main inference method remains the same across all problems.

We construct the KDE for each cluster and we use a relatively small kernel bandwidth in order to accurately represent the high probability regions in the posterior. This is

---

**Algorithm 1** Learning a global proposal $T_G(\mathbf{y}|\mathbf{x})$

---

1. Simulate $\{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\}_{i=1,\dots,n}$ from $p(\mathbf{x}|\mathbf{y})\, p(\mathbf{y})$
2. Compute a feature representation $v(\mathbf{x}^{(i)})$
3. Perform k-means clustering of $\{v(\mathbf{x}^{(i)})\}_i$
4. For each cluster $C_j \subset \{1,\dots,n\}$, fit a kernel density estimate $KDE(C_j)$ to the vectors $\mathbf{y}^{\{C_j\}}$

---

**Algorithm 2** INF-MH (Informed Metropolis-Hastings)

---

**Input:** observed image $\mathbf{x}$
$T_L \leftarrow$ Local proposal distribution (Gaussian)
$C \leftarrow$ cluster for $v(\mathbf{x})$
$T_G \leftarrow KDE(C)$ (as obtained by Alg. 1)
$T = \alpha T_L + (1-\alpha)T_G$
$\pi(\mathbf{y}|\mathbf{x}) \leftarrow$ Posterior distribution $P(\mathbf{y}|\mathbf{x})$
Initialize $\mathbf{y}_1$
**for** $t = 1$ **to** $N-1$ **do**
  1. Sample $\bar{\mathbf{y}} \sim T(\cdot)$
  2. $\gamma = \min\left(1, \frac{\pi(\bar{\mathbf{y}}|\mathbf{x})T(\bar{\mathbf{y}}\to\mathbf{y}_t)}{\pi(\mathbf{y}_t|\mathbf{x})T(\mathbf{y}_t\to\bar{\mathbf{y}})}\right)$
  **if** rand$(0,1) < \gamma$ **then**
    $\mathbf{y}_{t+1} = \bar{\mathbf{y}}$
  **else**
    $\mathbf{y}_{t+1} = \mathbf{y}_t$
  **end if**
**end for**

---

similar in spirit to using only high probability regions as "darts" in the *Darting Monte Carlo* sampling technique of [233]. We summarize the offline training in Algorithm 1.

At test time, this method has the advantage that given an image $\mathbf{x}$ we only need to identify the corresponding cluster once using $v(\mathbf{x})$ in order to sample efficiently from the kernel density $T_G$. We show the full procedure in Algorithm 2.

This method yields a transition kernel that is a mixture kernel of a reversible symmetric Metropolis-Hastings kernel and a metropolized independence sampler. The combined transition kernel $T$ is hence also reversible. Because the measure of each kernel dominates the support of the posterior, the kernel is ergodic and has the correct stationary distribution [38]. This ensures correctness of the inference and in the experiments we investigate the efficiency of the different methods in terms of convergence statistics.

# 3.4 Setup and Baseline Methods

In the remainder of the chapter we demonstrate the proposed method in three different experimental setups. For all experiments, we use four parallel chains initialized at different random locations sampled from the prior. The reported numbers are median statistics over multiple test images except when noted otherwise.

## 3.4.1 Baseline Methods

**Metropolis Hastings (MH)**    Described in Section 2.2.1, corresponds to $\alpha = 1$, we use a symmetric diagonal Gaussian distribution, centered at $\mathbf{y}_t$.

**Metropolis Hastings within Gibbs (MHWG)**    We use a Metropolis Hastings scheme in a Gibbs sampler, that is, we draw from one-dimensional conditional distributions for proposing moves and the Markov chain is updated along one dimension at a time. We further use a blocked variant of this MHWG sampler, where we update blocks of dimensions at a time, and denote it by BMHWG.

**Parallel Tempering (PT)**    We use Parallel Tempering to address the problem of sampling from multi-modal distributions [241, 95]. This technique is also known as "replica exchange MCMC sampling" [124]. We run different parallel chains at different temperatures $T$, sampling $\pi(\cdot)^{\frac{1}{T}}$ and at each sampling step propose to exchange two randomly chosen chains. In our experiments we run three chains at temperature levels $T \in \{1, 3, 27\}$ that were found to be best working out of all combinations in $\{1, 3, 9, 27\}$ for all experiments individually. The highest temperature levels corresponds to an almost flat distribution.

**Regeneration Sampler (REG-MH)**    We implemented a regenerative MCMC method [191] that performs adaption [96] of the proposal distribution during sampling. Adapting the proposal distribution with existing MCMC samples is not straight-forward as this would potentially violate the Markov property of the chain [13]. One approach is to identify *times of regeneration* at which the chain can be restarted and the proposal distribution can be adapted using samples drawn previously. Several approaches to identify good regeneration times in a general Markov chain have been proposed [14, 194]. We build on [191] that proposed two *splitting* methods for finding the regeneration times. Here, we briefly describe the method that we implemented in this study.

Let the present state of the sampler be $\mathbf{y}_t$ and let the independent global proposal distribution be $T_G$. When $\bar{\mathbf{y}} \sim T_G$ is accepted according to the MH acceptance rule, the probability of a regeneration is given by:

$$r(\mathbf{y}_t, \bar{\mathbf{y}}) = \begin{cases} \max\{\frac{c}{w(\mathbf{y}_t)}, \frac{c}{w(\bar{\mathbf{y}})}\}, & \text{if } w(\mathbf{y}_t) > c \text{ and } w(\bar{\mathbf{y}}) > c, \\ \max\{\frac{w(\mathbf{y}_t)}{c}, \frac{w(\bar{\mathbf{y}})}{c}\}, & \text{if } w(\mathbf{y}_t) < c \text{ and } w(\bar{\mathbf{y}}) < c, \\ 1, & \text{otherwise,} \end{cases} \qquad (3.2)$$

where $c > 0$ is an arbitrary constant and $w(\mathbf{y}_t) = \frac{\pi(\mathbf{y}_t)}{T_G(\mathbf{y}_t)}$. The value of $c$ can be set to maximize the regeneration probability. At every sampling step, if a sample from the independent proposal distribution is accepted, we compute regeneration probability using equation (3.2). If a regeneration occurs, the present sample is discarded and replaced with one from the independent proposal distribution $T_G$. We use the same mixture proposal distribution as in our informed sampling approach where we initialize the global proposal $T_G$ with a prior distribution and at times of regeneration fit a KDE to the existing samples. This becomes the new adapted distribution $T_G$. Refer to [191] for more details of this regeneration technique. In the work of [10], this regeneration technique is used with success in a Darting Monte Carlo sampler.

We use the mixture kernel (Eq. (3.1)) as proposal distribution and adapt only the global part $T_G(\cdot|\mathbf{x})$. This is initialized as the prior over target variables $P(\mathbf{y})$ and at times of regeneration we fit a KDE to the already drawn samples. For comparison we used the same mixture coefficient $\alpha$ as for INF-MH.

### 3.4.2 MCMC Diagnostics

We use established methods for monitoring the convergence of our MCMC method [139, 81]. In particular, we report different diagnostics. We compare the different samplers with respect to the number of iterations instead of time. The forward graphics process significantly dominates the runtime and therefore the iterations in our experiments correspond linearly to the runtime.

**Acceptance Rate (AR)** The ratio of accepted samples to the total Markov chain length. The higher the acceptance rate, the fewer samples we need to approximate the posterior. Acceptance rate indicates how well the proposal distribution approximates the true distribution *locally*.

**Potential Scale Reduction Factor (PSRF)** The PSRF diagnostics [93, 37] is derived by comparing within-chain variances with between-chain variances of sample statistics. For this, it requires independent runs of multiple chains (4 in our case) in parallel. Because our sample $\mathbf{y}$ is multi-dimensional, we estimate the PSRF for each parameter dimension separately and take the maximum PSRF value as final PSRF value. A value close to one indicates that all chains characterize the same distribution. This does not imply convergence, as the chains may all collectively miss a mode. However, a PSRF

value much larger than one is a certain sign of the lack of convergence. PSRF also indicates how well the sampler visits different modes of a multi-modal distribution.

**Root Mean Square Error (RMSE)**   During our experiments we have access to the input parameters $\mathbf{y}^*$ that generated the image. To assess whether the posterior distribution recovers the *correct* value, we report the RMSE between the posterior expectation $\mathbb{E}_{P(\mathbf{y}|\mathbf{x})}[G(\mathbf{y})]$ and the value $G(\mathbf{y}^*)$ of the generating input. Since there is noise being added to the observation, we do not have access to the ground truth posterior expectation and therefore this measure is only an indicator. Under convergence all samplers would agree on the same correct value.

### 3.4.3  Parameter Selection

For each sampler we individually selected hyper-parameters that gave the best PSRF value after $10k$ iterations. In case the PSRF does not differ for multiple values, we chose the one with the highest acceptance rate. We include a detailed analysis of the baseline samplers and parameter selection in Appendix A.1.

## 3.5  Experiments

We studied the use of informed sampler with three different problem scenarios: Estimating camera extrinsics (Section 3.5.1), occlusion reasoning (Section 3.5.2) and estimating body shape (Section 3.5.3).

### 3.5.1  Estimating Camera Extrinsics

We implement the following simple graphics scenario to create a challenging multi-modal problem. We render a cubical room of edge length 2 with a point light source in the center of the room, $(0,0,0)$, from a camera somewhere inside the room. The camera parameters are described by its $(x,y,z)$-position and the orientation, specified by yaw, pitch, and roll angles. The inference process consists of estimating the posterior over these 6D camera parameters $\mathbf{y}$. See Fig. 3.2 for two example renderings. Posterior inference is a highly multi-modal problem because the room is a cubical and thus symmetric. There are 24 different camera parameters that will result in the same image. This is also shown in Fig. 3.2 where we plot the position and orientation (but not camera roll) of all camera parameters that create the same image. A rendering of a $200 \times 200$ image with a resolution of $32bit$ using a single core on an Intel Xeon 2.66GHz machine takes about 11ms on average.

A small amount of isotropic Gaussian noise is added to the rendered image $G(\mathbf{y})$, using a standard deviation of $\sigma = 0.02$. The posterior distribution we try to infer is then:

Figure 3.2: **Sample images for the experiment on 'Estimating Camera Extrinsics'.** Two rendered room images with possible camera positions and headings that produce the same image. Not shown are the orientations; in the left example all six headings can be rolled by 90,180, and 270 degrees for the same image.

$p(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = \mathcal{N}(\mathbf{x}|G(\mathbf{y}),\sigma^2)\,\text{Uniform}(\mathbf{y})$. The uniform prior over location parameters ranges between $-1.0$ and $1.0$ and the prior over angle parameters is modeled with wrapped uniform distribution over $[-\pi,\pi]$.

To learn the informed part of the proposal distribution from data, we computed a histogram of oriented gradients (HOG) descriptor [61] from the image, using 9 gradient orientations and cells of size $20 \times 20$ yielding a feature vector $v(\mathbf{x}) \in \mathbb{R}^{900}$. We generated $300k$ training images using a uniform prior over the camera extrinsic parameters, and performed k-means clustering using $5k$ cluster centers based on the HOG feature vectors. For each cluster cell, we then computed and stored a KDE for the 6 dimensional camera parameters, following the steps in Algorithm 1. As test data, we create 30 images using extrinsic parameters sampled uniform at random over their range.

**Results**

We show results in Fig. 3.3. We observe that both MH and PT yield low acceptance rates (AR) compared to other methods. However parallel tempering appears to overcome the multi-modality better and improves over MH in terms of convergence. The same holds for the regeneration technique, we observe many regenerations, good convergence and AR. Both INF-INDMH and INF-MH converge quickly.

In this experimental setup, we have access to the 24 different exact modes. We analyze how quickly the samplers visit the modes and whether or not they capture all of them. For every different instance, the pairwise distances between the modes changes, therefore we chose to define 'visiting a mode' in the following way. We compute a Voronoi tessellation with the modes as centers. A mode is visited if a sample falls into its corresponding Voronoi cell, that is, it is closer than to any other mode. Sampling uniform at random

Figure 3.3: **Results of the 'Estimating Camera Extrinsics' experiment.** Acceptance Rates (left), PSRFs (middle), and average number of modes visited (right) for different sampling methods. We plot the median/average statistics over 30 test examples.

would quickly find the modes (depending on the cell sizes) but is not a valid sampler that characterizes the desired posterior distribution. We also experimented with balls of different radii around the modes and found a similar behavior to the one we report here. Figure 3.3 (right) shows results for various samplers. We find that INF-MH discovers different modes quicker when compared to other baseline samplers. Just sampling from the global proposal distribution INF-INDMH is initially visiting more modes (it is not being held back by local steps) but is dominated by INF-MH over some range. This indicates that the mixture kernel takes advantage of both local and global moves, either one of them is exploring slower. Also in most examples, all samplers miss some modes under our definition, the average number of discovered modes is 21 for INF-MH and even lower for MH.

Figure 3.4 shows the effect of mixture coefficient ($\alpha$) on the informed sampling INF-MH. Since there is no significant difference in PSRF values for $0 \leq \alpha \leq 0.7$, we chose 0.7 due to its high acceptance rate. Likewise, the parameters of the baseline samplers are chosen based on the PSRF and acceptance rate metrics. See Appendix A.1.1 for the analysis of the baseline samplers and the parameter selection.

We also tested the MHWG sampler and found that it did not converge even after $100k$ iterations, with a PSRF value around 3. This is to be expected since single variable updates will not traverse the multi-modal posterior distributions fast enough due to the high correlation of the camera parameters. In Fig. 3.5, we plot the median auto-correlation of samples obtained by different sampling techniques, separately for each of the six extrinsic camera parameters. The informed sampling approach (INF-MH and INF-INDMH) appears to produce samples which are more independent compared to other baseline samplers.

As expected, some knowledge of the multi-modal structure of the posterior needs to be available for the sampler to perform well. The methods INF-INDMH and INF-MH have this information and perform better than baseline methods and REG-MH.

Figure 3.4: **Role of mixture coefficient.** PRSFs and Acceptance rates corresponding to various mixture coefficients ($\alpha$) of INF-MH sampling in 'Estimating Camera Extrinsics' experiment.



Figure 3.5: **Independence of obtained samples.** Auto-Correlation of samples obtained by different sampling techniques in camera extrinsics experiment, for each of the six extrinsic camera parameters.

## 3.5.2 Occluding Tiles

In a second experiment, we render images depicting a fixed number of six quadratic tiles placed at a random location $(x, y)$ in the image at a random depth $z$ and orientation $\theta$. We blur the image and add a bit of Gaussian random noise ($\sigma = 0.02$). An example is

depicted in Fig. 3.6(a), note that all the tiles are of the same size, but farther away tiles look smaller. A rendering of one $200 \times 200$ image takes about 25ms on average. Here, as prior, we again use the uniform distribution over the 3D cube for tile location parameters, and wrapped uniform distribution over $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ for tile orientation angle. To avoid label switching issues, each tile is given a fixed color and is not changed during the inference.

We chose this experiment such that it resembles the 'dead leaves model' of [160], because it has properties that are commonplace in computer vision. It is a scene composed of several objects that are independent, except for occlusion, which complicates the problem. If occlusion did not exist, the task is readily solved using a standard OpenCV [33] rectangle finding algorithm (minAreaRect). The output of such an algorithm can be seen in Fig. 3.6(c), and we use this algorithm as a discriminative source of information. This problem is higher dimensional than the previous one (24, due to 6 tiles of 4 parameters). Inference becomes more challenging in higher dimensions and our approach without modification does not scale well with increasing dimensionality. One way to approach this problem, is to factorize the joint distribution into blocks and learn informed proposals separately. In the present experiment, we observed that both baseline samplers and the plain informed sampling fail when proposing all parameters jointly. Since the tiles are independent except for the occlusion, we can approximate the full joint distribution as product of block distributions where each block corresponds to the parameters of a single tile. To estimate the full posterior distribution, we learn global proposal distributions for each block separately and use a block-Gibbs like scheme in our sampler where we propose changes to one tile at a time, alternating between tiles.



(a) Given Image  (b) Ground Truth  (c) OpenCV Heuristic  (d) MHWG  (e) INF-BMHWG  (f) INF-BMHWG (Mean)

Figure 3.6: **A visual result in 'Occluding Tiles' experiment.** (a) A sample rendered image, (b) Ground truth squares, (c) Rectangle fitting with OpenCV MinAreaRect algorithm and most probable estimates from 5000 samples obtained by (d) MHWG sampler (best baseline) and (e) the INF-BMHWG sampler. (f) Posterior expectation of the square boundaries obtained by INF-BMHWG sampling (The first 2000 samples are discarded as burn-in).

The experimental protocol is the same as before, we render $500k$ images, apply the OpenCV MinAreaRect algorithm to fit rectangles and take their found four parameters as features for clustering ($10k$ clusters). Again KDE distributions are fit to each cluster and at test time, we assign the observed image to its corresponding cluster. The KDE in that chosen cluster determines the global sampler $T_G$ for that tile. We then use $T_G$ to propose

Figure 3.7: **Results of the 'Occluding Tiles' experiment.** Acceptance Rates (left), PSRFs (middle), and RMSEs (right) for different sampling methods. Median results for 10 test examples.

an update to all 4 parameters of the tile. We refer to this procedure as INF-BMHWG. Empirically we find $\alpha = 0.8$ to be optimal for INF-BMHWG sampling. Analysis of various samplers is presented in Appendix A.1.2.

## Results

An example visual result is shown in Fig. 3.6. We found that the the MH and INF-MH samplers fail entirely on this problem. Both use a proposal distribution for the entire state and due to the high dimensionality, there is almost no acceptance ($< 1\%$) and thus they do not reach convergence. The MHWG sampler, updating one dimension at a time, is found to be the best among the baseline samplers with acceptance rate of around 42%, followed by a block sampler that samples each tile separately. The OpenCV algorithm produces a reasonable initial guess but fails in occlusion cases.

The median diagnostic statistical curves for 10 test examples are shown in Fig. 3.7, INF-BMHWG by far produces lower reconstruction errors. The block wise informed sampler INF-BMHWG converges quicker, with higher acceptance rates ($\approx 53\%$), and lower reconstruction error. Also in Fig 3.6(f) the posterior distribution is visualized, fully visible tiles are more localized, position and orientation of occluded tiles more uncertain. Figure A.5, in the Appendix A.1, shows some more visual results. Although the model is relatively simple, all the baseline samplers perform poorly and discriminative information is crucial to enable accurate inference. Here the discriminative information is provided by a readily available heuristic in the OpenCV library.

This experiment illustrates a variation of the informed sampling strategy that can be applied to sampling from high-dimensional distributions. Inference methods for general high-dimensional distributions is an active area of research and intrinsically difficult. The occluding tiles experiment is simple but illustrates this point, namely that all non-

block baseline samplers fail. Block sampling is a common strategy in such scenarios and many computer vision problems have such block-structure. Again the informed sampler improves in convergence speed over the baseline method. Other techniques that produce better fits to the conditional (block-) marginals should give faster convergence.

### 3.5.3 Estimating Body Shape

The last experiment is motivated by a real world problem: estimating the 3D body shape of a person from a single static depth image. With the recent availability of cheap active depth sensors, the use of RGBD data has become ubiquitous in computer vision [223, 138].

To represent a human body, we use the *BlendSCAPE* model [117], which updates the originally proposed SCAPE model [11] with better training and blend weights. This model produces a 3D mesh of a human body as shown in Fig. 3.8 as a function of shape and pose parameters. The shape parameters allow us to represent bodies of many builds and sizes, and includes a statistical characterization (being roughly Gaussian). These parameters control directions in deformation space, which were learned from a corpus of roughly 2000 3D mesh models registered to scans of human bodies via PCA. The pose parameters are joint angles which indirectly control local orientations of predefined parts of the model.

Our model uses 57 pose parameters and any number of shape parameters to produce a 3D mesh with 10,777 vertices. We use the first 7 SCAPE components to represent the shape of a person and keep the pose fixed. The camera viewpoint, orientation, and pose of the person is held fixed. Thus a rendering process takes $\mathbf{y} \in \mathbb{R}^7$, generates a 3D mesh representation of it and projects it through a virtual depth camera to create a depth image of the person $\mathbf{x}$. This can be done in various resolutions, we chose $430 \times 260$ with depth values represented as 32bit numbers in the interval $[0, 4]$. On average, a full render path takes about 28ms. We add Gaussian noise with standard deviation of 0.02 to the created depth image. See Fig.3.8(left) for an example.

We used very simple low level features for feature representation. In order to learn the global proposal distribution we compute depth histogram features on a $15 \times 10$ grid on the image. For each cell we record the mean and variance of the depth values. Additionally we add the height and the width of the body silhouette as features resulting in a feature vector $v(\mathbf{x}) \in \mathbb{R}^{302}$. As normalization, each feature dimension is divided by the maximum value in the training set. We used $400k$ training images sampled from the standard normal prior distribution and $10k$ clusters to learn the KDE proposal distributions in each cluster cell.

For this experiment, we also experimented with a different conditional density estimation approach using a forest of random regression trees [36, 35]. See Section 2.3.1 for a brief overview of random forests. In the previous experiments, utilizing the KDE estimates, the discriminative information entered through the feature representation. Then, suppose if there was no relation between some observed features and the variables that

| Given Image | Ground Truth Mesh | Mean Mesh from Prior | Mean Mesh from Samples | Angular Error | Std. Deviation in Angular Error |

Figure 3.8: **Inference of body shape from a depth image.** A sample test result showing the result of 3D mesh reconstruction with the first 1000 samples obtained using our INF-MH sampling method. We visualize the angular error (in degrees) between the estimated and ground truth edge and project onto the mesh.

we are trying to infer, we would require a large number of samples to reliably estimate the densities in the different clusters. The regression forest can adaptively partition the parameter space based on observed features and is able to ignore uninformative features, thus may lead to better fits of the conditional densities. It can thus be understood as the adaptive version of the k-means clustering technique that solely relies on the used metric (Euclidean in our case).

In particular, we use the same features as for k-means clustering but grow the regression trees using a mean square error criterion for scoring the split functions. A forest of 10 binary trees with a depth of 15 is grown, with the constraint of having a minimum of 40 training points per leaf node. Then for each of the leaf nodes, a KDE is trained as before. At test time the regression forest yields a mixture of KDEs as the global proposal distribution. We denote this method as INF-RFMH in the experiments.

Instead of using one KDE model for each cluster, we could also explore a regression approach, for example using a discriminative linear regression model to map observations into proposal distributions. By using informative covariates in the regression model, one should be able to overcome the curse of dimensionality. Such a semi-parametric approach would allow to capture explicit parametric dependencies of the variables (for example linear dependencies) and combine them with non-parametric estimates of the residuals. We are exploring this technique as future work.

Again, we chose parameters for all samplers individually, based on empirical mixing rates. For informed samplers, we chose $\alpha = 0.8$, and a local proposal standard deviation of 0.05. The full analysis for all samplers is included in Appendix A.1.3.

48

**Results**

We tested the different approaches on 10 test images that are generated by the shape parameters drawn from the standard normal prior distribution. Figure 3.9 summarizes the results of the sampling methods. We make the following observations. The baselines methods MH, MHWG, and PT show inferior convergence results and, MH and PT also suffer from lower acceptance rates. Just sampling from the distribution of the discriminative step (INF-INDMH) is not enough, because the low acceptance rate indicates that the global proposals do not represent the correct posterior distribution. However, combined with a local proposal in a mixture kernel, we achieve a higher acceptance rate, faster convergence and a decrease in RMSE. The regression forest approach has slower convergence than INF-MH. In this example, the regeneration sampler REG-MH does not improve over simpler baseline methods. We attribute this to rare regenerations which may be improved with more specialized methods.

We believe that our simple choice of depth image representation can also be significantly improved. For example, features can be computed from identified body parts, something that the simple histogram features have not taken into account. In the computer vision literature, some discriminative approaches for pose estimation do exist, most prominent being the influential work on pose recovery in parts for the Kinect Xbox system [228]. In future work, we plan to use similar methods to deal with pose variation and complicated dependencies between parameters and observations.



Figure 3.9: **Results of the 'Body Shape' experiment.** Acceptance Rates (left), PSRFs (middle), and RMSEs (right) for different sampling methods in the body shape experiment. Median results over 10 test examples.

**3D Mesh Reconstruction**

In Fig. 3.8, we show a sample 3D body mesh reconstruction result using the INF-MH sampler after only 1000 iterations. We visualized the difference of the mean posterior

Figure 3.10: **Body measurements with quantified uncertainty.** Box plots of three body measurements for three test subjects, computed from the first 10*k* samples obtained by the INF-MH sampler. Dotted lines indicate measurements corresponding to ground truth SCAPE parameters.

and the ground truth 3D mesh in terms of mesh edge directions. One can observe that most differences are in the belly region and the feet of the person. The retrieved posterior distribution allows us to assess the model uncertainty. To visualize the posterior variance we record standard deviation over the edge directions for all mesh edges. This is back-projected to achieve the visualization in Fig. 3.8(right). We see that posterior variance is higher in regions of higher error, that is, our model predicts its own uncertainty correctly [62]. In a real-world body scanning scenario, this information will be beneficial; for example, when scanning from multiple viewpoints or in an experimental design scenario, it helps in selecting the next best pose and viewpoint to record. Fig. A.6, in appendix, shows more 3D mesh reconstruction results using our sampling approach.

**Body Measurements**

Predicting body measurements has many applications including clothing, sizing and ergonomic design. Given pixel observations, one may wish to infer a distribution over measurements (such as height and chest circumference). Fortunately, our original shape training corpus includes a host of 47 different per-subject measurements, obtained by professional anthropometrists; this allows us to relate shape parameters to measurements. Among many possible forms of regression, regularized linear regression [282] was found to best predict measurements from shape parameters. This linear relationship allows us to transform any posterior distribution over SCAPE parameters into a posterior over measurements, as shown in Fig. 3.10. We report for three randomly chosen subjects' (S1, S2,

and S3) results on three out of the 47 measurements. The dashed lines corresponds to ground truth values. Our estimate not only faithfully recovers the true value but also yields a characterization of the full conditional posterior.

**Incomplete Evidence**

Another advantage of using a generative model is the ability to reason with missing observations. We perform a simple experiment by occluding a portion of the observed depth image. We use the same inference and learning codes, with the same parametrization and features as in the non-occlusion case, but retrain the model to account for the changes in the forward graphics process. The result of INF-MH, computed on the first 10$k$ samples is shown in Fig. 3.11. The 3D reconstruction is reasonable even under large occlusion; the error and the edge direction variance did increase as expected.

## 3.6 Discussion and Conclusions

This work proposes a method to incorporate discriminative methods into Bayesian inference in a principled way. We augment a sampling technique with discriminative information to enable inference with global accurate generative models. Empirical results on three challenging and diverse computer vision experiments are discussed. We carefully analyze the convergence behavior of several different baselines and find that the informed sampler performs well across all different scenarios. This sampler is applicable to general scenarios and in this work we leverage the accurate forward process for offline training, a setting frequently found in computer vision applications. The main focus is the generality of the approach, this inference technique should be applicable to many different problems and not be tailored to a particular problem.

We show that even for very simple scenarios, most baseline samplers perform poorly or fail completely. By including a global image-conditioned proposal distribution that is informed through discriminative inference we can improve sampling performance. We deliberately use a simple learning technique (KDEs on k-means cluster cells and a forest of regression trees) to enable easy reuse in other applications. Using stronger and more tailored discriminative models should lead to better performance. We see this as a way where top-down inference is combined with bottom-up proposals in a probabilistic setting.

There are some avenues for future work; we understand this method as an initial step into the direction of general inference techniques for accurate generative computer vision models. Identifying conditional dependence structure should improve results, e.g. recently [236] used structure in Bayesian networks to identify such dependencies. One assumption in our work is that we use an accurate generative model. Relaxing this assumption to allow for more general scenarios where the generative model is known only approximately is important future work. In particular for high-level computer vision

Figure 3.11: **Inference with incomplete evidence.** Mean 3D mesh and corresponding errors and uncertainties (std. deviations) in mesh edge directions, for the same test case as in Fig. 3.8, computed from first 10*k* samples of our INF-MH sampling method with (bottom row) occlusion mask in image evidence. (blue indicates small values and red indicates high values)

problems such as scene or object understanding there are no accurate generative models available yet but there is a clear trend towards physically more accurate 3D representations of the world. This more general setting is different to the one we consider in this chapter, but we believe that some ideas can be carried over. For example, we could create the informed proposal distributions from manually annotated data that is readily available in many computer vision data sets. Another problem domain are trans-dimensional models, that require different sampling techniques like reversible jump MCMC methods [102, 38]. We are investigating general techniques to *inform* this sampler in similar ways as described in this chapter.

We believe that generative models are useful in many computer vision scenarios and that the interplay between computer graphics and computer vision is a prime candidate for studying probabilistic inference and probabilistic programming [179]. However, current inference techniques need to be improved on many fronts: efficiency, ease of usability, and generality. Our method is a step towards this direction: the informed sampler leverages the power of existing discriminative and heuristic techniques to enable a principled Bayesian treatment in rich graphics generative models. Our emphasis is on generality; we aimed to create a method that can be easily reused in other scenarios with existing code bases. The presented results are a successful example of the inversion of an involved rendering pass. In the future, we plan to investigate ways to combine existing computer vision techniques with principled generative models, with the aim of being general rather than problem specific.

# Chapter 4

# Consensus Message Passing

In the last chapter, we have proposed a technique to speed up the sampling process for inverse graphics. Despite having faster convergence with techniques like informed sampler, sampling based inference is often too slow for practical applications. An alternative inference approach in vision, which is often faster, is message-passing in factor graph models (see Section 2.2.2). Generative models in computer vision tend to be large, loopy and layered as discussed in Section 2.2.2. We find that widely-used, general-purpose message passing inference algorithms such as Expectation Propagation (EP) and Variational Message Passing (VMP) fail on the simplest of vision models. With these models in mind, we introduce a modification to message passing that learns to exploit their layered structure by passing *consensus* messages that guide inference towards good solutions. Experiments on a variety of problems show that the proposed technique leads to significantly more accurate inference results, not only when compared to standard EP and VMP, but also when compared to competitive bottom-up discriminative models. Refer to Section 2.2.2 for an overview of factor graphs and message-passing inference.

## 4.1 Introduction

As discussed in Section 2.2.3, perhaps, the most significant challenge of the generative modeling framework is that inference can be very hard. Sampling-based methods run the risk of slow convergence, while message passing-based methods (which are the focus of this chapter) can converge slowly, converge to bad solutions, or fail to converge at all. Whilst significant efforts have been made to improve the accuracy of message passing algorithms (*e.g.* by using structured variational approximations), many challenges remain, including difficulty of implementation, the problem of computational cost and the question of how the structured approximation should be chosen. The work in this chapter aims to alleviate these problems for general-purpose message-passing algorithms.

Our initial observation is that general purpose message passing inference algorithms (*e.g.* EP and VMP; [184, 267]) fail on even the simplest of computer vision models. We claim that in these models the failure can be attributed to the algorithms' inability to determine the values of a relatively small number of influential variables which we call 'global' variables. Without accurate estimation of these global variables, it can be very

difficult for message passing to make meaningful progress on the other variables in the model.

Latent variables in vision models are often organized in a layered structure (also discussed in Section 2.2.2), where the observed image pixels are at the bottom and high-level scene parameters are at the top. Additionally, knowledge about the values of the variables at level $l$ is sufficient to reason about any global variable at layer $l + 1$. With these properties in mind, we develop a method called *Consensus Message Passing* (CMP) that learns to exploit such layered structures and estimate global variables during the early stages of inference.

Experimental results on a variety of problems show that CMP leads to significantly more accurate inference results while preserving the computational efficiency of standard message passing. The implication of this work is twofold. First, it adds a useful tool to the toolbox of techniques for improving general-purpose inference, and second, in doing so it overcomes a bottleneck that has restricted the use of model-based machine learning in computer vision.

This chapter is organized as follows. In Section 4.2, we discuss related work and explain our CMP approach in Section 4.3. Then we present experimental analysis of CMP with three diverse generative vision models in Section 4.4. We conclude with a discussion in Section 4.5.

## 4.2  Related Work

Inspiration for CMP stems from the kinds of distinctions that have been made for decades between so-called 'intuitive', bottom-up, fast inference techniques, and iterative 'rational' inference techniques [115]. CMP can be seen as an implementation of such ideas in the context of message passing, where the consensus messages form the 'intuitive' part of inference and the following standard message passing forms the 'rational' part. Analogues to intuitive and rational inference also exist for sampling, where bottom-up techniques are used to compute proposals for MCMC, leading to significant speedup in inference [257, 236] (Chapter 3). The works of [213] and [145] proposed techniques for learning the parameters of both the generative model and the corresponding recognition model.

The idea of 'learning to infer' also has a long history. Early examples include [116], where a dedicated set of 'recognition' parameters are learned to drive inference. In more modern instances of such ideas [190, 217, 68, 224, 189], message passing is performed by a sequence of predictions defined by a graphical model, and the predictors are jointly trained to ensure that the system produces coherent labelings. However, in these techniques, the resulting inference procedure no longer corresponds to the original (or perhaps to any) graphical model. An important distinction of CMP is that the predictors fit completely within the framework of message passing and final inference results correspond to valid fixed points in the original model of interest.

Finally, we note recent works of [113] and [75] that make use of regressors (neural networks and random forests, respectively) to learn to pass EP messages. These works are concerned with reducing the computational cost of computing individual messages and do not make any attempt to change the accuracy or rate of convergence in message passing inference as a whole. In contrast, CMP learns to pass messages specifically with the aim of reducing the total number of iterations required for accurate inference in a given generative model.

## 4.3 Consensus Message Passing

Consensus message passing exploits the layered characteristic of vision models in order to overcome the aforementioned inference challenges. For illustration, two layers of latent variables of such a model are shown in Fig. 4.1a using factor graph notation (black). Here the latent variables below ($\mathbf{h}^b = \{h_k^b\}$) are a function of the latent variables above ($\mathbf{h}^a = \{h_k^a\}$) and the global variables $p$ and $q$, where $k$ ranges over pixels (in this case $|k| = 3$). As we will see in the experiments that follow, this is a recurring pattern that appears in many models of interest in vision. For example, in the case of face modeling, the $\mathbf{h}^a$ variables correspond to the normals $\mathbf{n}_i$, the global variable $p$ to the light vector $\mathbf{l}$, and $\mathbf{h}^b$ to the shading intensities $s_i$ (see Fig. 4.6b).

Our reasoning follows a recursive structure. Assume for a moment that in Fig. 4.1a, the messages from the layer below to the inter-layer factors (blue) are both informative and accurate (*e.g.* due to being close to the observed pixels). We will refer to these messages collectively as *contextual messages*. It would be desirable, for purposes of both speed and accuracy, that we could ensure that the messages sent to the layer above ($\mathbf{h}^a$) are also accurate and informative. If we had access to an oracle that could give us the correct belief for the global variables ($p$ and $q$) for the image, we could send accurate initial messages from $p$ and $q$ and then compute informative and accurate messages from the inter-layer factors to the layer above.

In practice, however, we do not have access to such an oracle. In this work we train regressors to *predict* the values of the global variables given all the messages from the layer below. Should this prediction be good enough, the messages to the layer above will be informative and accurate, and the inductive argument will hold for further layers (if any) above in the factor graph. We describe how these regressors are trained in Section 4.3.1. The approach consists of the following two components:

1. Before inference, for each global variable in different layers of the model, we train a regressor to predict some oracle's value for the target variable given the values of all the messages from the layer below (*i.e.* the *contextual messages*, Fig. 4.1a, blue),

2. During inference, each regressor sends this belief in the form of a *consensus message* (Fig. 4.1a, red) to its target variable.

(a) Type A          (b) Type B

Figure 4.1: **Consensus message passing.** Vision models tend to be large, layered and loopy. (a) Two adjacent layers of the latent variables of a model of this kind (black). In CMP, consensus messages (red) are computed from contextual messages (blue) and sent to global variables ($p$ and $q$), guiding inference in the layer. (b) Consensus message passing of a different kind for situations where loops in the graphical model are due to global variables in other layers.

In some models, it will be useful to employ a second type of CMP, illustrated graphically in Fig. 4.1b, where global layer variables are absent and loops in the graphical model are due to global variables in other layers. In this case, a consensus message is sent to each variable in the latent layer above, given all the contextual messages.

Any message passing schedule can be used subject to the constraint that the consensus messages are given maximum priority within a layer and that they are sent bottom up. Naturally, a consensus message can only be sent after its contextual messages have been computed. It is desirable to be able to ensure that the fixed point (result at convergence) reached under this scheme is also a fixed point of standard message passing in the model. One approach for this is to reduce the certainty of the consensus messages over the course of inference, or to only pass them in the first few iterations. In our experiments we found that even passing consensus messages only in the first iteration led to accurate inference, and therefore we follow this strategy for the remainder of the chapter. It is worth emphasizing that message-passing equations remain unchanged and we used the same scheduling scheme in all our experiments (*i.e.* no need for manual tuning).

It is important to highlight a crucial difference between consensus message passing and heuristic *initialization*. In the latter, predictions are made from the *observations* no matter how high up in the hierarchy the target variable is, whereas in CMP predictions are made using *messages* that are sent from variables immediately below the target variables of interest. The CMP prediction task is much simpler, since the relationship between the target variables and the variables in the layer immediately below is much less complex

than the relationship between the target variables and the observations. Furthermore, we know from the layered structure of the model that all relevant information from the observations is contained in the variables in the layer below. This is because target variables at layer $l+1$ are conditionally independent of all layers $l-1$ and below, given the values of layer $l$.

One final note on the capacity of the regressors. Of course it is true that an infinite capacity regressor can make perfect predictions given enough data (whether using CMP or heuristic initialization). However, we are interested in practical ways of obtaining accurate results for models of increasing complexity, where lack of capable regressors and unlimited data is inevitable. One important feature of CMP is that it makes use of predictors in a scalable way, since regressions are only made between adjacent latent layers.

### 4.3.1 Predicting Messages for CMP

To recap, the goal is to perform inference in a layered model of observed variables $\mathbf{x}$ with latent variables $\mathbf{h}$. Each predictor $\Delta^t$ (with target $t$) is a function of a collection of its contextual messages $\mathbf{c} = \{c_k\}$ (incoming from the latent layer below $\mathbf{h}^b$), that produces the consensus message $m$, *i.e.* $m = \Delta^t(\mathbf{c})$.

We adopt an approach in which we *learn* a function for this task that is parameterized by $\boldsymbol{\theta}$, *i.e.* $\overline{m} \equiv \mathcal{F}(\mathbf{c}|\boldsymbol{\theta})$. This can be seen as an instance of the canonical regression task. For a given family of regressors $\mathcal{F}$, the goal of training is to find parameters $\boldsymbol{\theta}$ that capture the relationship between context and consensus message pairs $\{(\mathbf{c}_d, m_d)\}_{d=1...D}$ in some set of training examples.

#### Choice of Predictor Training Data

First we discuss how this training data is obtained. There can be at least three different sources:

**1. Beliefs at Convergence.** Standard message passing inference is run in the model for a large number of iterations until convergence and for a collection of different observations $\{\mathbf{x}_d\}$. Message passing is scheduled in precisely the same way as it would be if CMP were present, however no consensus messages are sent. For each observation $\mathbf{x}_d$, the collection of the marginals of the latent variables in the layer below the predictor ($\mathbf{h}_d^b = \{h_{dk}^b\}$, (see *e.g.* Fig. 4.1a) at the *first* iteration of message passing is considered to be the context $\mathbf{c}_d$, and the marginal of the target variable $t$ at the *last* iteration of message passing is considered to be the oracle message $m_d$. The intuition is that during inference on new problems, a predictor trained in this way would send messages that *accelerate* convergence to the fixed-point that message passing would have reached by itself anyway. This technique is only useful if standard message passing works but is slow.

**2. Samples from the Model.** First a collection of samples from the model is generated, giving us both the observation $\mathbf{x}_d$ and its corresponding latent variables $\mathbf{h}_d$ for each

sample. Standard message passing inference is then run on the observations $\{\mathbf{x}_d\}$ only for a single iteration. Message passing is scheduled as before. For each observation $\mathbf{x}_d$, the marginals of the latent variables in the layer below $\mathbf{h}_d^b$ at the *first* iteration of message passing is the context $\mathbf{c}_d$, and the oracle message $m_d$ is considered to be a point-mass centered at the sampled value of the target variable $t$. The intuition is that during inference on new problems, a predictor trained in this way would send messages that guide inference to a fixed-point in which the marginal of the target variable $t$ is close to its sampled value. This technique is useful if standard message passing fails to reach good fixed points no matter how long it is run for.

**3. Labelled Data.** As above, except the latent variables of interest $\mathbf{h}_d$ are set from real data instead of being sampled from the model. The oracle message $m_d$ is therefore a point-mass centered at the label provided for the target variable $t$ for observation $\mathbf{X}_d$. The aim is that during inference on new problems, a predictor trained in this way would send messages that guide inference to a fixed-point in which the marginal of the target variable $t$ is close to its labelled value, even in the presence of a degree of model mismatch. We demonstrate each of the strategies in the experiments in Section 4.4.

**Random Regression Forests for CMP**

Our goal is to learn a mapping $\mathcal{F}$ from contextual messages $\mathbf{c}$ to the consensus message $m$ from training data $\{(\mathbf{c}_d, m_d)\}_{d=1...D}$. This is challenging since the inputs and outputs of the regression problem are both messages (*i.e.* distributions), and special care needs to be taken to account for this fact. We follow closely the methodology of [75], which use random forests to predict outgoing messages from a factor given the incoming messages to it. Please refer to Section 2.3.1 for a brief review of random forests.

In approximate message passing (*e.g.* EP [184] and VMP [267]), messages can be represented using only a few numbers, *e.g.* a Gaussian message can be represented by its natural parameters. We represent the contextual messages $\mathbf{c}$ collectively, in two different ways: first as a concatenation of the parameters of its constituent messages which we refer to as 'regression parameterization' and denote by $\mathbf{r}_c$; and second as a vector of features computed on the set which we refer to as 'tree parameterization' and denote by $\mathbf{t}_c$. This parametrization typically contains features of the set as a whole (*e.g.* moments of their means). We represent the outgoing message $m$ by a vector of real valued numbers $\mathbf{r}_m$.

**Prediction Model.** Each leaf node is associated with a subset of the labelled training data. During testing, a previously unseen set of contextual messages represented by $\mathbf{t}_c$ traverses the tree until it reaches a leaf which by construction is likely to contain similar training examples. Therefore, we use the statistics of the data gathered in that leaf to predict the consensus message with a multivariate regression model of the form: $\mathbf{r}_m = \mathbf{W} \cdot \mathbf{r}_c + \varepsilon$ where $\varepsilon$ is a vector of normal error terms. We use the learned matrix of coefficients $\mathbf{W}$ at test time to make predictions $\bar{\mathbf{r}}_m$ for a given $\mathbf{r}_c$. To recap, $\mathbf{t}_c$ is used to traverse the contextual messages down to leaves, and $\mathbf{r}_c$ is used by a linear regressor to

predict the parameters $\mathbf{r}_m$ of the consensus message.

**Training Objective Function.** Recall the training procedure of random forests from Section 2.3.1. Each node is in a tree represents a partition of feature space and split function in each node is chosen in a greedy manner minimizing a splitting criterion $E$. A common split criterion, which we also use here, is the sum of data likelihood in the node's left and right child clusters (see Eq. (2.14)). We use the 'fit residual' as defined in [75] as the likelihood (model fit) function for optimizing splits at each node. In other words, this objective function splits the training data at each node in a way that the relationship between the incoming and outgoing messages is well captured by the regression model in each child.

**Ensemble Model.** During testing, a set of contextual messages simultaneously traverses every tree in the forest from their roots until it reaches their leaves. Combining the predictions into a single forest prediction might be done by averaging the parameters $\mathbf{r}_m^t$ of the predicted messages $\overline{m}^t$ by each tree $t$, however this would be sensitive to the chosen parameterization for the messages. Instead we compute the moment average $\overline{m}$ of the distributions $\{\overline{m}^t\}$ by averaging the first few moments of the predictions across trees, and solving for the distribution parameters which match the averaged moments (see *e.g.* [105]).

# 4.4 Experiments

We first illustrate the application of CMP to two diagnostic models: one of circles and a second of squares. We then use the approach to improve inference in a more challenging vision model: intrinsic images of faces. In the first experiment, the predictors are trained on beliefs at convergence, in the second on samples from the model, and in the third on annotated labels, showcasing various use-cases of CMP. We show that in all cases, the proposed technique leads to significantly more accurate inference results whilst preserving the computational efficiency of message passing. The experiments were performed using Infer.NET [186] with default settings, unless stated otherwise. For random forest predictors, we set the number of trees in each forest to 8.

## 4.4.1 A Generative Model of Circles

We begin by studying the behavior of standard message passing on a simplified Gauss and Ceres problem [246]. Given a noisy sample of points $\mathbf{x} = \{\mathbf{x}_i\}_{i=1...N}$ on a circle in the 2D plane (Fig. 4.2a, black, $\mathcal{N}(0, 0.01)$ noise on each axis), the aim is to infer the coordinates of the circle's center $\mathbf{c}$ (Fig. 4.2a, red) and its radius $r$. We can express the data generation process using a graphical model (Fig. 4.2b). The Cartesian point $(0, r)$ is rotated $a_i$ radians to generate $\mathbf{p}_i$, then translated by $\mathbf{c}$ to generate the latent $\mathbf{z}_i$, which finally produces the noisy observation $\mathbf{x}_i$. This model can be expressed in a few lines of code in Infer.NET. The circle model is interesting for our purposes ,since it is both

Figure 4.2: **The circle problem.** (a) Given a sample of points on a circle (black), we wish to infer the circle's center (red) and its radius. Two sets of samples are shown. (b) The graphical model for this problem.

layered (the $z_i$s, $p_i$s and $a_i$s each form a layer) and loopy (due to the presence of two variables outside the plate).

We use this example to highlight the fact that although inference may require many iterations of message passing, message initialization can have a significant effect on the speed of convergence, and to demonstrate how this can be done automatically using CMP.

Vanilla message passing inference in this model can take a surprisingly large number of iterations to converge. We draw 10 points $\{x_i\}$ from circles with random centers and radii, run VMP and record the accuracy of the marginals of the latent variables at each iteration. We repeat the experiment 50 times and plot results in Fig. 4.3 (dashed black). As can be seen from the figure, the marginals contain significant errors even after 50 iterations of message passing.

We then experiment with consensus message passing. A predictor $\Delta^c$ is trained to send a consensus message to **c** in the initial stages of inference, given the messages coming up from all of the $z_i$ (indicated graphically in Fig. 4.2b, red). The predictor is trained on final beliefs at 100 iterations of standard message passing on $D = 500$ sample problems.

As can be seen in Fig. 4.3 (red), this single consensus message has the effect of significantly increasing the rate of convergence (as indicated by slope) and also inference robustness (as indicated by error bars). For comparison, we also plot how well a regressor of the same capacity as the one used by CMP can directly estimate the latent variables without using the graphical model in Fig. 4.3 (blue). Consensus message passing gives

(a) Center  (b) Radius

Figure 4.3: **Accelerated inference using CMP for the circle problem.** (a) Distance of the mean of the marginal posterior of center $c$ from its true value as a function of number of inference iterations (Forest: direct prediction, MP: standard VMP, CMP: VMP with consensus). Consensus message passing significantly accelerates convergence. (b) Similar plot for radius $r$.

us the best of both worlds in this example: speed that is more comparable to one-shot bottom-up prediction and the accuracy of message passing inference in a good model for the problem.

## 4.4.2 A Generative Model of Squares

Next, we turn our attention to a more challenging problem for which even the best message passing scheme that we could devise frequently finds completely inaccurate solutions. The task is to infer the center $\mathbf{c}$ and side length $r$ of a square in an image (Fig. 4.4a). Unlike the previous problem where we knew that all points belonged to the circle, here we must first determine which pixels belong to the square and which do not. To do so we might also wish to reason about the color of the foreground $\mathbf{fg}$ and background $\mathbf{bg}$, making the task of inference significantly harder. The graphical model for this problem is shown in Fig. 4.4b. Let $\mathbf{c}$ and $l$ denote square center and side length respectively. At each pixel position $p_i$, $s_i$ is a boolean variable indicating the square's presence. Depending on the value of $s_i$, the gate copies the appropriate color ($\mathbf{fg}$ or $\mathbf{bg}$) to $\mathbf{z}_i$.

We experiment with 50 test images (themselves samples from the model), perform inference using EP and with a sequential schedule, recording the accuracy of the marginals of the latent variables at each iteration. We additionally place damping with step size 0.95 on messages from the square factor to the center $\mathbf{c}$. We found these choices led to the best performing standard message passing algorithm. Despite this, we observed inference accuracy to be disappointingly poor (see Fig. 4.5). In Fig. 4.5a we see that, for

Figure 4.4: **The square problem.** (a) We wish to infer the square's center and its side length. (b) A graphical model for this problem. $s_i$ is a boolean variable indicating the square's presence at position $p_i$. Depending on the value of $s_i$, the gate copies the appropriate color (**fg** or **bg**) to $\mathbf{z}_i$.

many images, message passing converges to highly inaccurate marginals for the center. The low quality of inference can also be seen in quantitative results of Figs. 4.5(b-d).

We implement CMP predictors at two different layers of the model (see Fig. 4.4b, red). In the first layer, $\Delta^{\mathbf{fg}}$ and $\Delta^{\mathbf{bg}}$ send consensus messages to **fg** and **bg** respectively, given the messages coming up from all of the $\mathbf{z}_i$ which take the form of independent Gaussians centered at the appearances of the observed pixels (we use a Gaussian noise model). Therefore $\Delta^{\mathbf{fg}}$ and $\Delta^{\mathbf{bg}}$ effectively make initial guesses of the values of the foreground and background colors in the image given the observed image. Split features in the internal nodes of the regression forest are designed to test for equality of two randomly chosen pixel positions, and sparse regressors are used at the leaves to prevent overfitting.

In the second layer, $\Delta^l$ sends a consensus message to $l$ given the messages coming up from all of the $s_i$. The messages from $s_i$ take the form of independent Bernoullis indicating the algorithm's current beliefs about the presence of the square at each pixel. Therefore, the predictor's job is to predict the square's side length from this probabilistic segmentation map. Note that it is much easier to implement a regressor to perform this task (effectively one only needs to count) than it is to do so using the original observed

(a) Center

(b) Center

(c) Side length

(d) BG color

Figure 4.5: **Robustified inference using CMP for the square problem.** (a) Position of inferred centers relative to ground-truth. Image boundaries shown in blue for scale. (b,c,d) Distance of the mean of the posterior of **c**, $l$ and **bg** from their true values. CMP consistently increases inference accuracy. Results have been averaged over 50 different problems. 1 stage CMP only makes use of the lower predictors $\Delta^{\mathbf{fg}}$ and $\Delta^{\mathbf{bg}}$.

image pixels $x_i$. We find these predictors to be sufficient for stable inference and so we do not implement a fourth predictor for **c**. We experiment with single stage CMP, where only the lower predictors $\Delta^{\mathbf{fg}}$ and $\Delta^{\mathbf{bg}}$ are active, and with two stage CMP, where all three predictors are active. The predictors are trained on $D = 500$ samples from the model.

The results of these experiments are shown in Fig. 4.5. We observe that CMP significantly improves the accuracy of inference for the center **c** (Figs. 4.5a, 4.5b) but also for the other latent variables (Figs. 4.5c, 4.5d). Note that single stage CMP appears to be insufficient for guiding message passing to good solutions. Whereas in circle example CMP accelerated convergence, this example demonstrates how it can make inference possible in models that were outside the capabilities of standard message passing.

Normal $\{\mathbf{n}_i\}$

Shading $\{s_i\}$

Reflectance $\{r_i\}$

Observed image
$\{x_i\}$

(a)

Inner
Product

$\mathbf{n_i}$

$\mathbf{l}$

$\Delta^{\mathbf{l}}$

$s_i$

$r_i$

$\times$

$\Delta_i^{\mathbf{r}}$

$z_i$

Gaussian

$x_i$

(b)

Figure 4.6: **The face problem.** (a) We observe an image and wish to infer the corresponding reflectance map and normal map (visualized here as 3D shape). (b) A graphical model for this problem. Symmetry priors not shown.

### 4.4.3  A Generative Model of Faces

In this sectin, we also investigate a more realistic application: face modeling. The estimation of reflectance and shape from a single image of a human face is a well-studied problem in computer vision (see *e.g.* [94, 161, 264, 140, 244]). A primary motivation for this task is that reflectance and shape are invariant to confounding light effects, and are therefore useful for downstream tasks such as recognition. The problem is ill-posed and modern approaches make use of prior knowledge in order to obtain good solutions, *e.g.* in the form of average reflectance and normal statistics [27, 28] or morphable 3D models [278, 264].

**Model.** Given an observation of image pixels $\mathbf{x} = \{x_i\}$, the aim is to infer the reflectance value $r_i$ and normal vector $\mathbf{n_i}$ for each pixel $i$ (see Fig. 4.6a). In Fig. 4.6b, a model is shown for these variables that represents the following image formation process: $x_i = (\mathbf{n_i} \cdot \mathbf{l}) \times r_i + \varepsilon$, thereby assuming Lambertian reflection and an infinitely distant directional light source with variable intensity. We place Gaussian priors over reflectances $\{r_i\}$, normals $\{\mathbf{n_i}\}$, and the light $\mathbf{l}$; and set the parameters of the priors using training

Figure 4.7: **A visual comparison of inference results for the face problem.** For 4 randomly chosen test images, we show inference results obtained by competing methods. (a) Observed images. (b) Inferred reflectance maps. *GT* is the stereo estimate which we use as a proxy for ground-truth, *BU* is the bottom-up reflectance estimate of Biswas *et al.* (2009), *MP* refers to standard variational message passing, *Forest* is the consensus prediction and *CMP* is the proposed consensus message passing technique. (c) The variance of the inferred reflectance estimate produced by CMP (normalized across rows). High variance regions correlate strongly with cast shadows. (d) Visualization of inferred light. (e) Inferred normal maps.

data. We additionally place a soft symmetry prior on the $\{r_i\}$ (the reflectance value on one side of the face should be close to its value on the other side) and on the $\{\mathbf{n_i}\}$ (normal vectors on each side should be approximately symmetric), reflecting our prior knowledge about faces. These symmetry priors can be added to the model in just a few lines of code, illustrating the way in which model-based methods lend themselves to rapid prototyping and experimentation.

Although this model is only a crude approximation to the true image formation process (*e.g.* it does not account for shadows or specularities), similar approximations have been found to be useful in prior work [27, 28, 140]. Additionally, if we can successfully develop algorithms that perform accurate and reliable inference in this class of models, we would then be able to increase its usefulness by updating it to reflect the true image formation process more accurately. Note that even for a relatively small image of size $96 \times 84$, the model contains over 48,000 latent variables and 56,000 factors, and as we will show below, standard message passing in the model routinely fails to converge to accurate solutions.

**Consensus Message Passing.** We use predictors at two levels in the model (see Fig. 4.6b) to tackle this problem. The first sends consensus messages to *each* reflectance pixel $r_i$, making it an instance of type B of CMP as described in Fig. 4.1b. Here, each consensus message is predicted using information from all the contextual messages from the $z_i$. We denote each of these predictors by $\Delta_i^{\mathbf{r}}$. The second predictor sends a consensus

Observed    'GT'    BU    MP    Forest    **CMP**    Variance
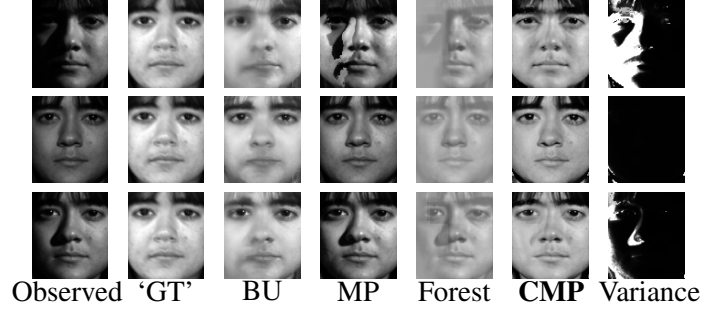
Figure 4.8: **Robustness to varying illumination.** Left to right: observed image, photometric stereo estimate (proxy for ground-truth), [27] estimate, VMP result, consensus forest estimate, CMP mean, and CMP variance.

message to **l** using information from all the messages from the $s_i$ and is denoted by $\Delta^{\mathbf{l}}$. The first level of predictors effectively make a guess of the reflectance image from the denoised observation, and the second layer predictor produces an estimate of the light from the shading image (which is likely to be easier to do than directly from the observation). The reflectance predictors $\{\Delta_i^{\mathbf{r}}\}$ are all powered by a single random forest, however the pixel position $i$ is used as a feature that it can exploit to create location specific behaviour. The tree parameterization of the contextual messages **c** for use in the reflectance predictor $\Delta_i^{\mathbf{r}}$ also includes 16 features such as mean, median, max, min and gradients of a $21 \times 21$ patch around the pixel. The tree parameterization of the contextual messages for use in the lighting predictor $\Delta^{\mathbf{l}}$ consists of means of the mean of the shading messages in $12 \times 12$ blocks. We deliberately use simple features to maintain generality but one could imagine the use of more specialized regressors for maximal performance.

**Datasets.** We experiment with the 'Yale B' and 'Extended Yale B' datasets [94, 161]. Together, they contain images of 38 subjects each with 64 illumination directions. We remove images taken with extreme light angles (azimuth or elevation $\geq 85$ degrees) that are almost entirely in shadow, leaving around 45 images for each subject. Images are down-sampled to $96 \times 84$. There are no ground-truth normals or reflectances for this dataset, however it is common practice to create proxy ground-truths using photometric stereo, which we obtain using the code of [205]. We use images from 22 subjects for training and test on the remaining 16 subjects.

**Results.** We begin by qualitatively assessing the different inference schemes. In Fig. 4.7 we show inference results for reflectance maps, normal maps and lights that are obtained after 100 iterations of message passing (VMP). For reflectance (Fig. 4.7b), we would like inference to produce estimates that match closely the ground-truth produced by photometric stereo (GT). We also display the reflectance estimates produced by the strong baseline (BU) of [27] for reference. We note that the baseline achieves excellent accuracy in regions with strong lighting, however it produces blurry estimates in regions under shadow.

(a) Without shadows

(b) With shadows

Figure 4.9: **Reflectance inference accuracy demonstrated through recognition accuracy.** CMP allows us to make use of the full potential of the generative model, thereby outperforming the competitive bottom-up method of [27].



(a) Without shadows

(b) With shadows

Figure 4.10: **Light inference accuracy.** The presence of cast shadows makes the direct prediction task easier, however CMP is accurate even in their absence.

As can be seen in Fig. 4.7b (MP), standard variational message passing finds solutions that are highly inaccurate with continued presence of illumination and artifacts in areas of cast show. In contrast, inference using CMP produces artefact-free results that much more closely resemble the stereo ground-truths. Arguably CMP also improves over the baseline [27], since its estimates are not blurry in regions with cast shadows. This can be attributed to the presence of symmetry priors in the model. Additionally, we note that the variance of the CMP inference for reflectance (Fig. 4.7c) correlates strongly with cast shadows in the observed images (*i.e.* the model is uncertain where it should be) suggesting that in future work it would be fruitful to have the notion of cast shadows explicitly built into the model. Figs. 4.7d and 4.7e show analogous results for lighting and normal maps, and Fig. 4.8 demonstrates CMP's ability to robustly infer reflectance maps for images of a single subject taken under varying lighting conditions. More visual

results are shown in the supplementary at the end of this thesis (Figs. A.7, 4.8).

We use the task of subject recognition (using estimated reflectance) as a quantitative measure of inference accuracy, as it can be difficult to measure in more direct ways (*e.g.* RMSE strongly favors blurry predictions). The reflectance estimate produced by each algorithm is compared to all training subjects' ground-truth reflectances and is assigned the label of its closest match. We have found this evaluation to reflect the quality of inference estimates. Fig. 4.9 shows the result of this experiment, both for real images and also synthetic images that were produced by taking the stereo ground-truths and adding artificial lighting (but with no cast shadows). We show analogous results for light in Fig. 4.10, where error is defined to be the cosine angle distance between the estimated light and the photometric stereo reference. First, we note that standard variational message passing (MP) performs poorly, producing reflectance estimates that are much less useful for recognition than those from [27]. Second, we note that CMP in the same model (both 1 stage and 2 stage versions) produces inferences that are significantly more useful downstream. The horizontal line labelled 'Forest' represents the accuracy of the consensus messages without any message passing, showing that the model-based fine-tuning provides a significant benefit. Finally, we highlight the fact that initializing light directly from the image and running message passing (Fig. 4.9, Init+MP) leads to worse estimates than CMP demonstrating the use of layered predictions as opposed to direct predictions from the observations. These results demonstrate that CMP helps message passing find better fixed points even in the presence of model mis-match (shadows) and make use of the full potential of the generative model.

## 4.5  Discussion and Conclusions

We have presented Consensus Message Passing and shown that it is a computationally efficient technique that can be used to improve the accuracy of message passing inference in a variety of vision models. The crux of the approach is to recognize the importance of global variables, and to take advantage of layered model structures commonly seen in vision to make rough estimates of their values.

The success of CMP depends on the accuracy of the random forest predictors. The design of forest features is not yet completely automated, but we took care in this work to use generic features that can be applied to a broad class of problems. Our forests are implemented in an extensible manner, and we envisage building a library of them that one can choose from, simply by inspecting the data types of the contextual and target variables.

In future work, we would like to exploit the benefits of the CMP framework by applying it to more challenging problems from computer vision. Each of the examples in Section 4.4 can be extended in various ways, *e.g.* by making considerations for multiple objects, incorporating occlusion in the squares example and cast shadows in the faces example, or by developing more realistic priors. We are also seeking to understand in

what other domains the application of our ideas may be fruitful.

More broadly, a major challenge in machine learning is that of enriching models in a scalable way. We continually seek to ask our models to provide interpretations of increasingly complicated, heterogeneous data sources. Graphical models provide an appealing framework to manage this complexity, but the difficulty of inference has long been a barrier to achieving these goals. The CMP framework takes us one step in the direction of overcoming this barrier.

# Part II

# Inference in Discriminative Vision Models

# Chapter 5

# Learning Sparse High Dimensional Filters

In the Part-II of this thesis, we focus on inference in discriminative CNN models. Since *inference* amounts to simple evaluation of the *model* in discriminative models, we propose modifications to the original model itself for better inference. This is unlike the inference strategies proposed in Part-I of this thesis, where we proposed to learn a separate inference model that helps in the Bayesian inference of a given generative model. In this chapter, we propose a learning technique for general sparse high-dimensional filters and show how this can be used for generalizing standard spatial convolutions in CNNs to learnable bilateral convolutions with long-range image-adaptive connections.

Bilateral filters have wide spread use due to their edge-preserving properties. The common use case is to manually choose a parametric filter type, usually a Gaussian filter. In this chapter, we will generalize the parametrization and in particular derive a gradient descent algorithm so the filter parameters can be learned from data. This derivation allows to learn high dimensional linear filters that operate in sparsely populated feature spaces. We build on the permutohedral lattice construction for efficient filtering. The ability to learn more general forms of high-dimensional filters can be used in several diverse applications. First, we demonstrate the use in applications where single filter applications are desired for runtime reasons. Further, we show how this algorithm can be used to learn the pairwise potentials in densely connected conditional random fields and apply these to different image segmentation tasks. Finally, we introduce layers of bilateral filters in CNNs and propose *bilateral neural networks* for the use of high-dimensional sparse data. This view provides new ways to encode model structure into network architectures. A diverse set of experiments empirically validates the usage of general forms of filters.

## 5.1 Introduction

Image convolutions are basic operations for many image processing and computer vision applications. In this chapter, we will study the class of bilateral filter convolutions and propose a general image adaptive convolution that can be learned from data.

The bilateral filter [15, 234, 251] 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, in the following, we will directly state the general case. Given an image $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n), \mathbf{x}_i \in \mathbb{R}^c$ with $n$ pixels and $c$ channels, and for every pixel $i$, a $d$ dimensional feature vector $\mathbf{f}_i \in \mathbb{R}^d$ (*e.g.* the $(x, y)$ position in the image $\mathbf{f}_i = (x_i, y_i)^\top$). The bilateral filter then computes

$$\mathbf{x}'_i = \sum_{j=1}^n \mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j} \mathbf{x}_j. \tag{5.1}$$

for all $i$. Almost the entire literature refers to the bilateral filter as a synonym of the Gaussian parametric form $\mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j} = \exp\left(-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^\top \Sigma^{-1} (\mathbf{f}_i - \mathbf{f}_j)\right)$. The features $\mathbf{f}_i$ 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 five-dimensional feature $\mathbf{f} = (x, y, r, g, b)^\top$. Two pixels $i, j$ have a strong influence $\mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j}$ 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 behavior 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 two-dimensional image plane that changes, the filter values $\mathbf{w}_{\mathbf{f}, \mathbf{f}'}$ do not change. The filter itself can be of $c$ dimensions $\mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j} \in \mathbb{R}^c$, 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 [183].

The filter operation of Eq. (5.1) is a sparse high-dimensional convolution, a view advocated in [19, 199]. An image $\mathbf{x}$ 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 five-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 5D space it is. The edge-preserving behavior is possible due to the higher dimensional operation. Other data can naturally be understood as sparse signals, *e.g.* 3D surface points.

The main 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 re-uses 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 [199], Gaussian KD-trees [8], and the permutohedral lattice [7]. 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 [7] and show how to compute gradients of the convolution (Section 5.3) in lattice space. This enables learning the filter from data.

This insight has several useful consequences. We discuss applications where the bilateral filter has been used before: image filtering (Section 5.4) and CRF inference (Section 5.5). Further we will demonstrate how the free parametrization of the filters enables us to use them in deep convolutional neural networks (CNN) and allow convolutions that go beyond the regular spatially connected receptive fields (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.

**Image Adaptive Filtering:** The literature in this area is rich and we can only provide a brief overview. Important classes of image adaptive filters include the bilateral filters [15, 251, 234], non-local means [43, 16], locally adaptive regressive kernels [243], guided image filters [109] and propagation filters [49]. The kernel least-squares regression problem can serve as a unified view of many of them [183]. In contrast to the present work that learns the filter kernel using supervised learning, all these filtering schemes use a predefined kernel. Because of the importance of bilateral filtering to many applications in image processing, much effort has been devoted to derive fast algorithms; most notably [199, 7, 8, 92]. Surprisingly, the only attempt to learn the bilateral filter we found is [122] that casts the learning problem in the spatial domain by rearranging pixels. However, the learned filter does not necessarily obey the full region of influence of a pixel as in the case of a bilateral filter. The bilateral filter also has been proposed to regularize a large set of applications in [21, 20] and the respective optimization problems are parametrized in a bilateral space. In these works the filters are part of a learning system but unlike this work restricted to be Gaussian.

**Dense CRF:** The key observation of [151] 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 [7]. This quickly found wide-spread use, *e.g.* the combination of CNNs with a dense CRF is among the best performing segmentation models [52, 279, 25]. These works combine structured prediction frameworks on top of CNNs, to model the relationship between the desired output variables thereby significantly improving upon the CNN result. Bilateral neural networks, 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. Several works [150, 69, 142, 279, 220] demonstrate how to learn free parameters of the dense CRF model. However, the parametric form of the pairwise term always remains a Gaussian. Campbell *et al.* [47] embed complex pixel dependencies into an 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 will discuss how to learn the pairwise potentials, while retaining the efficient inference strategy of [151].

**Neural Networks:**   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.* [42] 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.* [128] propose a spatial transformation of signals within the neural network to learn invariances for a given task. The work of [126] propose matrix back-propagation techniques which can be used to build specialized structural layers such as normalized-cuts. Graham *et al.* [101] propose extensions from 2D CNNs to 3D sparse signals. Our work enables sparse 3D 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 [7] 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 $\mathbf{w}_{\mathbf{f}_i,\mathbf{f}_j}$ and perform the summation independently. The view of $\mathbf{w}$ as a linear filter in a higher dimensional space, as proposed by [199], opened the way for new algorithms. Here, we will build on the permutohedral lattice convolution developed in Adams *et al.* [7] for approximate Gaussian filtering. The most common application of bilateral filters use photometric features (XYRGB). We chose the permutohedral lattice as it is particularly designed for this dimensionality, see Fig. 7 in [7] for a speed comparison.
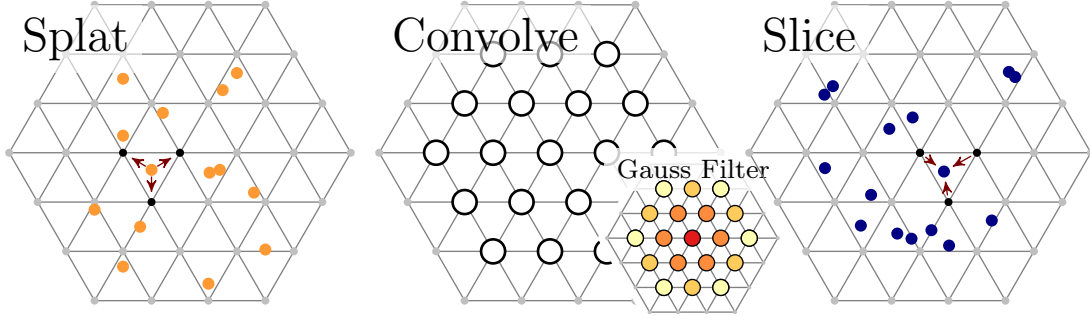
Figure 5.1: **Schematic of the permutohedral convolution.** Left: splatting the input points (orange) onto the lattice corners (black); Middle: The extent of a filter on the lattice with a $s = 2$ neighborhood (white circles), for reference we show a Gaussian filter, with its values color coded. The general case has a free scalar/vector parameter per circle. Right: The result of the convolution at the lattice corners (black) is projected back to the output points (blue). Note that in general the output and input points may be different.

### 5.3.1 Permutohedral Lattice Convolutions

We first review the permutohedral lattice convolution for Gaussian bilateral filters from Adams *et al.* [7] and describe its most general case.

As before, we assume that every image pixel $i$ is associated with a $d$-dimensional feature vector $\mathbf{f}_i$. Gaussian bilateral filtering using a permutohedral lattice approximation involves 3 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 2D 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. We refer to [7] for details of the lattice construction and its properties. 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 original $d$-dimensional feature locations, called *slicing*. For example, if the features used are a combination of position and color $\mathbf{f}_i = (x_i, y_i, r_i, g_i, b_i)^\top$, the input signal is mapped into the 5D cross product space of position and color and then convolved with a 5D tensor. Afterwards, the filtered result is mapped back to the original space. In practice we use a feature scaling $\Lambda\mathbf{f}$ with a diagonal matrix $\Lambda$ and use separate scales for position and color features. The scale determines the distance of points and thus the size of the lattice cells. More formally, the computation is written by $\mathbf{x}' = S_{\text{slice}}BS_{\text{splat}}\mathbf{x}$ and all involved matrices are defined below. For notational

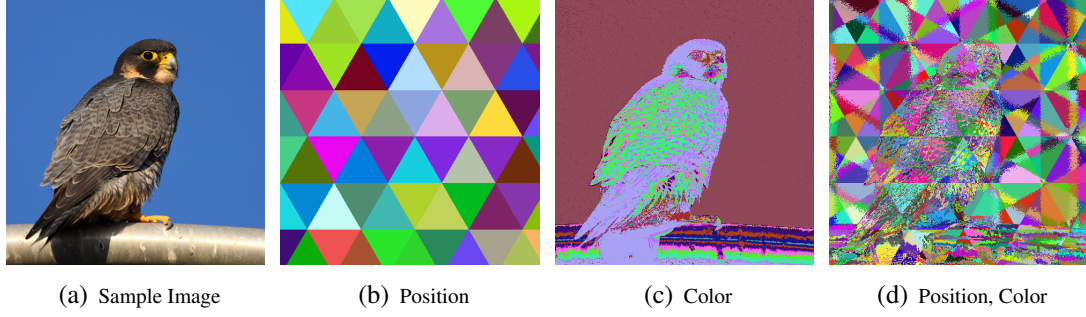(a) Sample Image    (b) Position    (c) Color    (d) Position, Color

Figure 5.2: **Visualization of the permutohedral lattice.** (a) Input image; Lattice visualizations for different feature spaces: (b) 2D position features: $0.01(x, y)$, (c) color features: $0.01(r, g, b)$ and (d) position and color features: $0.01(x, y, r, g, b)$. All pixels falling in the same simplex cell are shown with the same color.

convenience we will assume scalar input signals $x_i$, the vector valued case is analogous, the lattice convolution changes from scalar multiplications to inner products.

**Splat:** The splat operation (cf. left-most image in Fig. 5.1) finds the enclosing simplex in $\mathcal{O}(d^2)$ on the lattice of a given pixel feature $\mathbf{f}_i$ 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 $t_{i,j} \in \mathbb{R}$ inside the simplex. Thus, the value $\ell_j \in \mathbb{R}$ at a lattice point $j$ is computed by summing over all enclosed input points; more precisely, we define an index set $J_i$ for a pixel $i$, which contains all the lattice points $j$ of the enclosing simplex

$$\ell = S_{\text{splat}}\mathbf{x}; (S_{\text{splat}})_{j,i} = t_{i,j}, \text{ if } j \in J_i, \text{ otherwise } 0. \tag{5.2}$$

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

$$\ell' = B\ell; (B)_{j',j} = w_{j,j'}, \text{ if } j' \in N_s(j), \text{ otherwise } 0. \tag{5.3}$$

An illustration of a two-dimensional permutohedral filter is shown in Fig. 5.1 (middle). Note that we already presented the convolution in the general form that we will make use of. The work of [7] chooses the filter weights such that the resulting operation approximates a Gaussian blur, which is illustrated in Fig. 5.1. Further, the algorithm of [7] 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 $B$.

**Slice:** The slice operation (cf. right-most image in Fig. 5.1) computes an output value $x'_{i'}$ for an output pixel $i'$ again based on its barycentric coordinates $t_{i,j}$ and sums over the corner points $j$ of its lattice simplex

$$\mathbf{x}' = S_{\text{slice}}\ell'; (S_{\text{slice}})_{i,j} = t_{i,j}, \text{ if } j \in J_i, \text{ otherwise } 0 \tag{5.4}$$

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 $\mathbf{w}$ from [7] in Eq. (5.3) is designed to approximate a Gaussian filter. However, the convolution kernel $\mathbf{w}$ can naturally be understood as a general filtering operation in the permutohedral lattice space with free parameters. In the exposition above we already presented this general case. As we will show in more detail later, this modification has non-trivial consequences for bilateral filters, CNNs and probabilistic graphical models.

The size of the neighborhood $N_s(k)$ for the blur in Eq. (5.3) compares to the filter size of a spatial convolution. 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. A comparable filter on the permutohedral lattice with an $s$ neighborhood is specified by $(s+1)^{d+1} - s^{d+1} \in \mathcal{O}(s^d)$ elements (cf. Appendix A.3.1). Thus, both share the same asymptotic size.

By computing the gradients of the filter elements, we enable the use of gradient based optimizers, *e.g.* back-propagation for CNN in the same way that spatial filters in a CNN are learned. The gradients with respect to $\mathbf{x}$ and the filter weights in $B$ of a scalar loss $L$ are:

$$\frac{\partial L}{\partial \mathbf{x}} = S'_{\text{splat}} B' S'_{\text{slice}} \frac{\partial L}{\partial \mathbf{x}'}, \tag{5.5}$$

$$\frac{\partial L}{\partial (B)_{i,j}} = \left( S'_{\text{slice}} \frac{\partial L}{\partial \mathbf{x}} \right)_i (S_{\text{splat}} \mathbf{x})_j. \tag{5.6}$$

Both gradients are needed during back-propagation and in experiments, we use stochastic back-propagation for learning the filter kernel. The permutohedral lattice convolution is parallelizable, and scales linearly with the filter size. Specialized implementations run at interactive speeds in image processing applications [7]. Our implementation in the Caffe deep learning framework [136] allows arbitrary filter parameters and the computation of the gradients on both CPU and GPU. The code is available at http://bilateralnn.is.tuebingen.mpg.de.

|  (a) Input  |  (b) Guidance  |  (c) Ground Truth  |  (d) Bicubic  |  (e) Gauss-BF  |  (f) Learned-BF  |

Figure 5.3: **Guided up-sampling.** Color (top) and depth (bottom) $8\times$ up-sampling results using different methods: Bicubic - Bicubic interpolation; Gauss-BF - Gaussian bilateral upsampling; Learned-BF - Learned bialteral up-sampling (best viewed on screen).

## 5.4  Single Bilateral Filter Applications

In this section, we will consider the problems of joint bilateral up-sampling [148] and 3D body mesh denoising as prominent instances of single bilateral filter applications. See [200] for a recent overview of other bilateral filter applications. Further experiments on image denoising are included in Appendix A.3, together with details about exact experimental protocols and more visualizations.

### 5.4.1  Joint Bilateral Upsampling

A typical technique to speed up computer vision algorithms is to compute results on a lower scale and up-sample the result to the full resolution. This up-sampling step may use the original resolution image as a guidance image. A joint bilateral up-sampling approach for this problem setting was developed in [148]. We describe the procedure for the example of up-sampling a color image. Given a high resolution gray scale image (the guidance image) and the same image on a lower resolution but with colors, the task is to up-sample the color image to the same resolution as the guidance image. Using the permutohedral lattice, joint bilateral up-sampling proceeds by splatting the color image into the lattice, using 2D position and 1D intensity as features and the 3D RGB values as the signal. A convolution is applied in the lattice and the result is read out at the features of the high resolution image, that is using the 2D position and intensity of the guidance image. The possibility of reading out (slicing) points that are not necessarily the input points is an appealing feature of the permutohedral lattice convolution.

**Color Up-sampling**

For the task of color up-sampling, we compare the Gaussian bilateral filter [148] against a learned generalized filter. We experimented with two different datasets: Pascal VOC2012

| Upsampling factor | Bicubic | Gaussian | Learned |
|---|---|---|---|
| Color Upsampling (PSNR) | | | |
| 2x | 24.19 / 30.59 | 33.46 / 37.93 | **34.05 / 38.74** |
| 4x | 20.34 / 25.28 | 31.87 / 35.66 | **32.28 / 36.38** |
| 8x | 17.99 / 22.12 | 30.51 / 33.92 | **30.81 / 34.41** |
| 16x | 16.10 / 19.80 | 29.19 / 32.24 | **29.52 / 32.75** |
| Depth Upsampling (RMSE) | | | |
| 8x | 0.753 | 0.753 | **0.748** |

Table 5.1: **Joint bilateral up-sampling.** (top) PSNR values corresponding to various up-sampling factors and up-sampling strategies on the test images of the Pascal VOC12 segmentation / high-resolution 2MP dataset; (bottom) RMSE error values corresponding to up-sampling depth images estimated using [72] computed on the test images from the NYU depth dataset [231].

| | | Test Factor | | | |
|---|---|---|---|---|---|
| | | **2×** | **4×** | **8×** | **16×** |
| | **2×** | **38.45** | 36.12 | 34.06 | 32.43 |
| Train Factor | **4×** | 38.40 | **36.16** | **34.08** | 32.47 |
| | **8×** | 38.40 | 36.15 | **34.08** | 32.47 |
| | **16×** | 38.26 | 36.13 | 34.06 | **32.49** |

Table 5.2: **Color upsampling with different train and test up-sampling factors.** PSNR values corresponding to different up-sampling factors used at train and test times on the 2 megapixel image dataset, using our learned bilateral filters.

segmentation [77] using train, validation and test splits, and 200 higher resolution (2MP) images from Google image search [3] with 100 train, 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 5.1 we report result in terms of Peak-Signal-to-Noise ratio (PSNR) for the up-sampling factors $2\times, 4\times, 8\times$ and $16\times$. We compare a standard bicubic interpolation, that does not use a guidance image, the Gaussian bilateral filter case (with feature scales optimized on the validation set), and the learned filter. All filters have the same support. For all up-sampling factors, joint bilateral Gaussian up-sampling outperforms bicubic interpolation and is in turn improved using a learned filter. A result of the up-sampling is shown in Fig. 5.3 and more results are included in Section A.3.3. The learned filter recovers finer details in the images.

We also performed the cross-factor analysis of training and testing at different up-sampling factors. Table 5.2 shows the PSNR results for this analysis. Although, in terms of PSNR, it is optimal to train and test at the same up-sampling factor, the differences are small when training and testing up-sampling factors are different.

**Depth Up-sampling**

We experimented with depth up-sampling as another joint up-sampling task. We use the dataset of [231] that comes with pre-defined train, validation and test splits. The approach of [72] 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 down-sample the $640 \times 480$ images to $320 \times 240$ as a pre-processing step before CNN convolutions. The final depth result is bicubic interpolated to the original resolution. It is this interpolation that we replace with a Gaussian and learned joint bilateral up-sampling. The features are five-dimensional position and color information from the high resolution input image. The filter is learned using the same protocol as for color up-sampling minimizing MSE prediction error. The quantitative results are shown in Table 5.1, the Gaussian filter performs equal to the bicubic interpolation, the learned filter is better. Qualitative results are shown in Fig 5.3, both joint bilateral up-sampling respect image edges in the result. For this [80] and other tasks specialized interpolation algorithms exist, *e.g.* deconvolution networks [276]. Part of future work is to equip these approaches with bilateral filters. More qualitative results are presented in Appendix A.3.3.



Figure 5.4: **Sample data for 3D mesh denoising.** (top) Some 3D body meshes sampled from [177] and (bottom) the corresponding noisy meshes used in denoising experiments.

Figure 5.5: **4D isomap features for 3D human bodies.** Visualization of 4D isomap features for a sample 3D mesh. Isomap feature values are overlaid onto mesh vertices.

## 5.4.2 3D 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 3D meshes.

We sample 3D human body meshes using a generative 3D body model from [177]. To the clean meshes, we add Gaussian random noise displacements along the surface normal at each vertex location. Figure 5.4 shows some sample 3D meshes sampled from [177] and corresponding noisy meshes. The task is to take the noisy meshes as inputs and recover the original 3D body meshes. We create 1000 training, 200 validation and another 500 testing examples for the experiments. Although we use synthetically generated meshes and noise for our experiments for the sake of training, our technique could be potentially used for denoising the noisy meshes arising from 3D scanning devices.

**Mesh Representation:** The 3D human body meshes from [177] are represented with 3D vertex locations and the edge connections between the vertices. We found that this signal representation using global 3D coordinates is not suitable for denoising with bilateral filtering. Therefore, we first smooth the noisy mesh using mean smoothing applied to the face normals [271] and represent the noisy mesh vertices as 3D vector displacements with respect to the corresponding smoothed mesh. Thus, the task becomes denoising the 3D vector displacements with respect to the smoothed mesh.

**Isomap Features:** To apply permutohedral convolution, we need to define features at each input vertex point. We use a 4 dimensional isomap embedding [248] of the given 3D mesh graph as features. The given 3D mesh is converted into a weighted edge graph with edge weights set to the Euclidean distance between the connected vertices and to infinity between the non-connected vertices. Then the 4 dimensional isomap embedding is

|  | Noisy Mesh | Normal Smoothing | Gauss Bilateral | Learned Bilateral |
|---|---|---|---|---|
| **Vertex Distance (RMSE)** | 5.774 | 3.183 | 2.872 | **2.825** |
| **Normal Angle Error** | 19.680 | 19.707 | 19.357 | **19.207** |

Table 5.3: **Body denoising.**  Vertex distance RMSE values and normal angle error (in degrees) corresponding to different denoising strategies averaged over 500 test meshes.



Ground Truth          Given Noisy Mesh          Denoised Mesh

Figure 5.6: **Sample denoising result.**  Ground truth mesh (left), corresponding given noisy mesh (middle) and the denoised result (right) using the learned bilateral filter.

computed for this weighted edge graph using a publicly available implementation [247]. Figure 5.5 shows the visualization of isomap features on a sample 3D mesh.

**Experimental Results:**  Mesh denoising with a bilateral filter proceeds by splatting the input 3D mesh vectors (displacements with respect to the smoothed mesh) into the 4D isomap feature space, filtering the signal in this 4D space and then slicing back into original 3D input space. The Table 5.3 shows quantitative results as RMSE for different denoising strategies. The normal smoothing [271] already reduces the RMSE. The Gauss bilateral filter results in significant improvement over normal smoothing and learning the filter weights again improves the result. A visual result is shown in Fig. 5.6.

# 5.5 Learning Pairwise Potentials in Dense CRFs

The bilateral convolution from Section 5.3 generalizes the class of DenseCRF models for which the mean-field inference from [151] applies. The DenseCRF models have found wide-spread use in various computer vision applications [239, 24, 279, 259, 258, 25]. Recall from Section 2.2.2, for a DenseCRF model with unary potentials $\psi_u$ and pairwise potentials $\psi_p^{ij}$, the mean-field inference results in a fixed point equation which can be solved iteratively to update the marginal distributions $Q_i$. In iteration $t$, we have:

$$Q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\{-\psi_u(x_i) - \underbrace{\sum_{l \in \mathcal{L}} \sum_{j \neq i} \psi_p^{ij}(x_i, l) Q_j^t(l)}_{\text{bilateral filtering}}\}. \tag{5.7}$$

Thus bilateral filtering is used for fast mean-field inference in DenseCRF models. One of the fundamental limitations with the existing use of DenseCRFs is the confinement of pairwise potentials $\psi_p^{ij}(y_i, y_j)$ to be Gaussian as bilateral filtering is traditionally applied with Gaussian kernel.

## 5.5.1 Learning Pairwise Potentials

The proposed bilateral convolution generalizes the class of potential functions $\psi_p^{ij}$, since they allow a richer class of kernels $k(\mathbf{f}_i, \mathbf{f}_j)$ that furthermore can be learned from data. So far, all dense CRF models have used Gaussian potential functions $k$, we replace it with the general bilateral convolution and learn the parameters of kernel $k$, 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 $k$ we make use of the gradients for filter parameters in $k$ and use back-propagation through the mean-field iterations [69, 169] to learn them.

The work of [150] derived gradients to learn the feature scaling $\Lambda$ but not the form of the kernel $k$, which still was Gaussian. In [47], the features $\mathbf{f}_i$ 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 pre-processing step, not integrated in a end-to-end learning framework. Both aforementioned works are generalizations that are orthogonal to our development and can be used in conjunction.

## 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 VOC data [77] and material classification [25]. We use pre-trained models from the literature and compare the relative change when learning the pairwise potentials, as in the last section. For both the experiments, we use

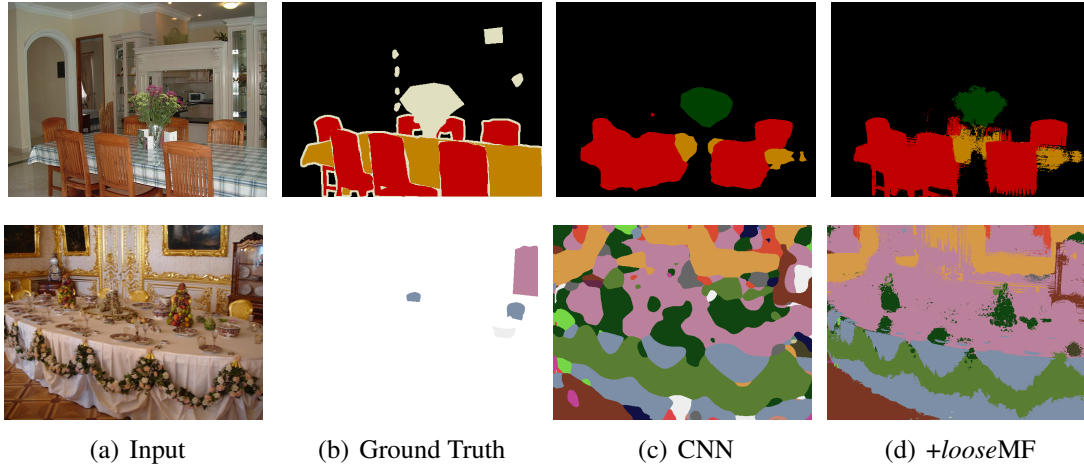| (a) Input | (b) Ground Truth | (c) CNN | (d) +*loose*MF |

Figure 5.7: **Segmentation results.** An example result for semantic (top) and material (bottom) segmentation. (c) depicts the unary results before application of MF, (d) after two steps of *loose*-MF with a learned CRF. More examples with comparisons to Gaussian pairwise potentials can be found in the supplementary material.

multinomial logistic classification loss and learn the filters via back-propagation [69]. This has also been understood as recurrent neural network variants [279], the following experiments demonstrate the learnability of bilateral filters.

**Semantic Segmentation**

Semantic segmentation is the task of assigning a semantic label to every pixel. We choose the DeepLab network [52], a variant of the VGGnet [232] for obtaining unaries. The DeepLab architecture runs a CNN model on the input image to obtain a result that is down-sampled by a factor of 8. The result is then bilinear interpolated to the desired resolution and serves as unaries $\psi_u(x_i)$ in a dense CRF. We use the same Pott's label compatibility function $\mu$, and also use two kernels $k^1(\mathbf{f}_i, \mathbf{f}_j) + k^2(p_i, p_j)$ with the same features $\mathbf{f}_i = (x_i, y_i, r_i, g_i, b_i)^\top$ and $\mathbf{p}_i = (x_i, y_i)^\top$ as in [52]. Thus, the two filters operate in parallel on color & position, and spatial domain respectively. We also initialize the mean-field update equations with the CNN unaries. The only change in the model is the type of the pairwise potential function from Gauss to a generalized form.

We evaluate the result after 1 step and 2 steps of mean-field inference and compare the Gaussian filter versus the learned version (cf. Tab. 5.4). First, as in [52] we observe that one step of mean field improves the performance by 2.48% in Intersection over Union (IoU) score. However, a learned potential increases the score by 2.93%. The same behavior is observed for 2 steps: the learned result again adds on top of the raised 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 [169]. This 'loose' mean-field

|  | + **MF-1step** | + **MF-2 step** | + *loose* **MF-2 step** |
|---|---|---|---|
| Semantic segmentation (IoU) - CNN [52]: 72.08 / 66.95 | | | |
| Gauss CRF | +2.48 | +3.38 | +3.38 / +3.00 |
| Learned CRF | +2.93 | +3.71 | **+3.85 / +3.37** |
| Material segmentation (Pixel Accuracy) - CNN [25]: 67.21 / 69.23 | | | |
| Gauss CRF | +7.91 / +6.28 | +9.68 / +7.35 | +9.68 / +7.35 |
| Learned CRF | +9.48 / +6.23 | +11.89 / +6.93 | **+11.91 / +6.93** |

Table 5.4: **Improved mean-field inference with learned potentials.** (top) Average IoU score on Pascal VOC12 validation/test data [77] for semantic segmentation; (bottom) Accuracy for all pixels / averaged over classes on the MINC test data [25] for material segmentation.

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.

**Material Segmentation**

We adopt the method and dataset from [25] for the material segmentation task. Their approach proposes the same architecture as in the previous section; a CNN to predict the 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 and Lab color/position features as in [25]. Results for pixel accuracy and class-averaged pixel accuracy are shown in Table 5.4. Following the CRF validation in [25], we ignored the label 'other' for both the training and evaluation. For this dataset, the availability of training data is small, 928 images with only sparse segment annotations. While this is enough to cross-validate few hyper-parameters, we would expect the general bilateral convolution to benefit from more training data. Visual results are shown in Fig. 5.7 and more are included in Appendix A.3.3.

# 5.6 Bilateral Neural Networks

Probably the most promising opportunity for the generalized bilateral filter is its use in Convolutional Neural Networks. Since we are not restricted to the Gaussian case, we can stack several filters in both parallel and sequential manner in the same way as filters are ordered in layers in typical spatial CNN architectures. Having the gradients available allows for end-to-end training with back-propagation, without the need for any change in CNN training protocols. We refer to the layers of bilateral filters as 'bilateral convolution layers' (BCL). 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 will refer to CNNs that include at least one bilateral convolutional layer as a bilateral neural network (BNN).

| Dim.-Features | d-dim caffe | BCL |
|---|---|---|
| 2D-$(x,y)$ | **3.3 ± 0.3 / 0.5 ± 0.1** | 4.8 ± 0.5 / 2.8 ± 0.4 |
| 3D-$(r,g,b)$ | 364.5 ± 43.2 / 12.1 ± 0.4 | **5.1 ± 0.7 / 3.2 ± 0.4** |
| 4D-$(x,r,g,b)$ | 30741.8 ± 9170.9 / 1446.2 ± 304.7 | **6.2 ± 0.7 / 3.8 ± 0.5** |
| 5D-$(x,y,r,g,b)$ | out of memory | **7.6 ± 0.4 / 4.5 ± 0.4** |

Table 5.5: **Runtime comparison: BCL vs. spatial convolution.** Average CPU/GPU runtime (in ms) of 50 1-neighborhood filters averaged over 1000 images from Pascal VOC. All scaled features $(x,y,r,g,b) \in [0,50]$. BCL includes splatting and splicing operations which in layered networks can be re-used.

What are the possibilities of a BCL compared to a standard spatial layer? First, we can define a feature space $\mathbf{f}_i \in \mathbb{R}^d$ to define proximity between elements to perform the convolution. This can include color or intensity as in the previous example. We performed a runtime comparison (Tab. 5.5) between our current implementation of a BCL and the caffe [136] implementation of a $d$-dimensional convolution. For 2D positional features (first row), the standard layer is faster since the permutohedral algorithm comes with an overhead. For higher dimensions $d > 2$, the runtime depends on the sparsity; but ignoring the sparsity is quickly leading to intractable runtimes for the regular $d$-dimensional convolution. The permutohedral lattice convolution is in effect a sparse matrix-vector product and thus performs favorably in this case. In the original work [7], it was presented as an approximation to the Gaussian case, here we take the viewpoint of it being the definition of the convolution itself.

Next we illustrate two use cases of BNNs and compare against spatial CNNs. Appendix A.3 contains further explanatory experiments with examples on MNIST digit recognition.

## 5.6.1 An Illustrative Example: Segmenting Tiles

In order to highlight the model possibilities of using higher dimensional sparse feature spaces for convolutions through BCLs, we constructed the following illustrative problem. A randomly colored foreground tile with size $20 \times 20$ is placed on a random colored background of size $64 \times 64$. Gaussian noise with standard deviation of 0.02 is added and color values are normalized to $[0,1]$, example images are shown in Fig. 5.8a. The task is to segment out the smaller tile. A pixel classifier can not distinguish foreground from background since the color is random. We train CNNs with three convolution/ReLU layers and varying filters of size $n \times n, n \in \{9, 13, 17, 21\}$. The schematic of the architecture is shown in Fig 5.8b (32, 16, 2 filters). We create 10k training, 1k validation and 1k test images and, use the validation set to choose learning rates. In Fig. 5.8c, we plot the validation IoU against training epochs.

Now, we replace all spatial convolutions with bilateral convolutions for a full BNN. The features are $\mathbf{f}_i = (x_i, y_i, r_i, g_i, b_i)^\top$ and the filter has a neighborhood of 1. The total
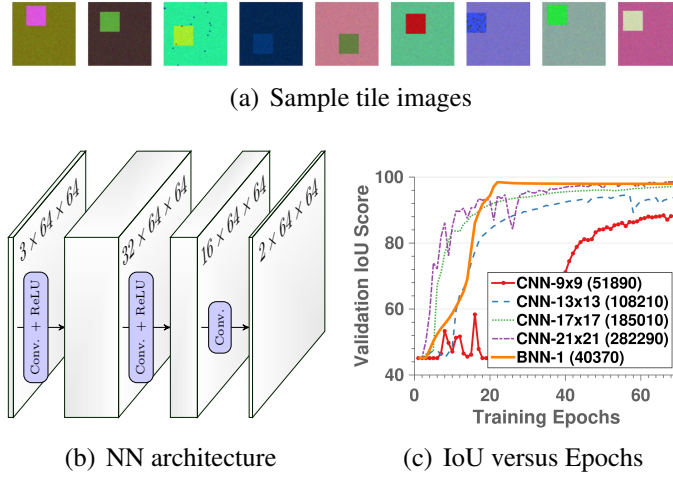
(a) Sample tile images



(b) NN architecture

(c) IoU versus Epochs

Figure 5.8: **Segmenting Tiles.** (a) Example tile input images; (b) the 3-layer NN architecture used in experiments. 'Conv' stands for spatial convolutions, resp. bilateral convolutions; (c) Training progress in terms of validation IoU versus training epochs.

number of parameters in this network is around $40k$ compared to $52k$ for $9 \times 9$ up to $282k$ for a $21 \times 21$ CNN. With the same training protocol and optimizer, the convergence rate of BNN is much faster. In this example as in semantic segmentation discussed in the last section, color is a discriminative information for the label. The bilateral convolutions *see* the color difference, the points are already pre-grouped in the permutohedral lattice and the task remains to assign a label to the two groups.

## 5.6.2 Character Recognition

The results for tile, semantic, and material segmentation when using general bilateral filters mainly improved because the feature space was used to encode useful prior information about the problem (similar RGB of close-by pixels have the same label). Such prior knowledge is often available when structured predictions are to be made, but the input signal may also be in a sparse format to begin with. Let us consider handwritten character recognition, one of the prime cases for CNN use.

The Assamese character dataset [17] contains 183 different Indo-Aryan symbols with 45 writing samples per class. Some sample character images are shown in Fig. 5.9a. 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 about 3% of the pixels contain a pen stroke, which we will denote by $I_i = 1$.

A CNN is a natural choice to approach this classification task. We experiment with two CNN architectures that have been used for this task, LeNet-7 from [158] and DeepCNet [56, 100]. The LeNet is a shallower network with bigger filter sizes whereas DeepCNet is deeper with smaller convolutions. Both networks are fully specified in Appendix A.3.3. In order to simplify the task for the networks we cropped the characters

(a) Sample Assamese character images (9 classes, 2 samples each)



(b) LeNet training  (c) DeepCNet training

Figure 5.9: **Character recognition.** (a) Sample Assamese character images [17]; and training progression of various models with (b) LeNet and (c) DeepCNet base networks.

by placing a tight bounding box around them and providing the bounding boxes as input to the networks. We will call these networks Crop-LeNet and Crop-DeepCNet. For training, we randomly divided the data into 30 writers for training, 6 for validation and the remaining 9 for test. Fig.5.9b and Fig. 5.9c show the training progress for various LeNet and DeepCNet models respectively. DeepCNet is a better choice for this problem and for both cases, pre-processing the data by cropping improves convergence.

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 bilateral convolutions using the features $\mathbf{f}_i = (x_i, y_i)^\top$ and we *only* consider the foreground points $I_i = 1$. 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. In effect, the lattice is very sparse which reduces runtime because the convolutions are only performed on 3% of the points that are actually observed. A 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 splatted at all points $(x_i, y_i)$ and passed on to the remaining spatial layers. The convergence behavior is shown in Fig.5.9 and again we find faster convergence and also better validation accuracy. The empirical results of this experiment for all tested architectures are summarized in Table 5.6, with BNN variants clearly outperforming their spatial counterparts.

The absolute results can be vastly improved by making use of virtual examples, *e.g.* by affine transformations [100]. The purpose of these experiments is to compare the networks on equal grounds while we believe that additional data will be beneficial for both networks. We have no reason to believe that a particular network benefits more.

| | LeNet | Crop-LeNet | BNN-LeNet | DeepCNet | Crop-DeepCNet | BNN-DeepCNet |
|---|---|---|---|---|---|---|
| Validation | 59.29 | 68.67 | **75.05** | 82.24 | 81.88 | **84.15** |
| Test | 55.74 | 69.10 | **74.98** | 79.78 | 80.02 | **84.21** |

Table 5.6: **Results on Assamese character images.** Total recognition accuracy for the different models.

## 5.7 Discussion and Conclusions

We proposed to learn bilateral filters from data. In hindsight, 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 relaxed the non-separability in the algorithm from [7] to allow for more general filter functions. There is a wide range of possible applications for learned bilateral filters [200] and we discussed some generalizations of previous work. These include joint bilateral up-sampling and inference in dense CRFs. We further demonstrated a use case of bilateral convolutions in neural networks.

The bilateral convolutional layer allows for filters whose receptive field change given the 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, segmentation, surfaces. The proposed filtering operation is then a natural candidate to define a filter convolutions on these objects, it takes advantage of sparsity and scales to higher dimensions. Therefore, we believe that this view will be useful for several 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.

In summary, the proposed technique for learning sparse high-dimensional filters results in a generalization of bilateral filters that helps in learning task-specific bilateral filters. In the context of inference, learnable bilateral filters can be used for better modeling and thus inference in neural network as well as DenseCRF models.

# Chapter 6

# Video Propagation Networks

In this chapter, we leverage the learnable bilateral filters developed in the previous chapter, and develop a novel neural network architecture for inference in video data. We focus on the task of propagating information across video frames. Standard CNNs are poor candidates for filtering video data. Standard spatial CNNs have fixed receptive fields whereas the video content changes differently in different videos depending on the type of scene and camera motion. So, filters with video adaptive receptive fields are better candidates for video filtering.

Based on this observation, we adapt the bilateral convolution layers (BCL) proposed in the previous chapter for filtering video data. By stacking several BCL and standard spatial convolutional layers, we develop a neural network architecture for video information propagation which we call 'Video Propagation Network' (VPN). We evaluate VPN on different tasks of video object segmentation and semantic video segmentation and show increased performance comparing to the best previous task-specific methods, while having favorable runtime. Additionally we demonstrate our approach on an example regression task of propagating color in a grayscale video.

## 6.1 Introduction

We focus on the problem of propagating structured information across video frames in this chapter. This problem appears in many forms (e.g., semantic segmentation or depth estimation) and is a pre-requisite for many applications. An example instance is shown in Fig. 6.1. Given an accurate object mask for the first frame, the problem is to propagate this mask forward through the entire video sequence. Propagation of semantic information through time and video colorization are other problem instances.

Videos pose both technical and representational challenges. The presence of scene and camera motion lead to the difficult association problem of optical flow. Video data is computationally more demanding than static images. A naive per-frame approach would scale at least linear with frames. These challenges complicate the use of standard convolutional neural networks (CNNs) for video processing. As a result, many previous works for video propagation use slow optimization based techniques.

Figure 6.1: **Video Propagation with VPNs.** The end-to-end trained VPN network is composed of a bilateral network followed by a standard spatial network and can be used for propagating information across frames. Shown here is an example result of foreground mask from the $1^{st}$ frame to other video frames.

We propose a generic neural network architecture that propagates information across video frames. The main innovation is the use of image adaptive convolutional operations that automatically adapt to the video stream content. This allows the network to adapt to the changing content of the video stream. It can be applied to several types of information, e.g. labels, colors, etc. and runs online, that is, only requiring current and previous frames.

Our architecture is composed of two components (see Fig. 6.1). A temporal *bilateral network* that performs image-adaptive spatio-temporal dense filtering. This part allows to connect densely all pixels from current and previous frames and to propagate associated pixel information to the current frame. The bilateral network allows the specification of a metric between video pixels and allows a straight-forward integration of temporal information. This is followed by a standard *spatial CNN* on the filter output to refine and predict for the present video frame. We call this combination a *Video Propagation Network (VPN)*. In effect we are combining a filtering technique with rather small spatial

CNNs which leads to a favorable runtime compared to many previous approaches.

VPNs have the following suitable properties for video processing:

**General applicability:** VPNs can be used for propagating any type of information content i.e., both discrete (e.g., semantic labels) and continuous (e.g. color) information across video frames.

**Online propagation:** The method needs no future frames and so can be used for online video analysis.

**Long-range and image adaptive:** VPNs can efficiently handle a large number of input frames and are adaptive to the video.

**End-to-end trainable:** VPNs can be trained end-to-end, so they can be used in other deep network architectures.

**Favorable runtime:** VPNs have favorable runtime in comparison to several current best methods, also making them amenable for learning with large datasets.

Empirically we show that VPNs, despite being generic, perform better or on-par with current best approaches on video object segmentation and semantic label propagation while being faster. VPNs can easily be integrated into sequential per-frame approaches and require only a small fine-tuning step that can be performed separately.

## 6.2 Related Work

The literature on propagating information across video frames contains a vast and varied number of approaches. Here, we only discuss those works that are related to our technique and applications.

**General propagation techniques** Techniques for propagating content across image or video pixels are predominantly optimization based or filtering techniques. Optimization based techniques typically formulate the propagation as an energy minimization problem on a graph constructed across video pixels or frames. A classic example is the color propagation technique from [164] which uses graph structure that encodes prior knowledge about pixel colors in a local neighborhood. Although efficient closed-form solutions [165] exists for certain scenarios, optimization tends to be slow due to either large graph structures for videos and/or the use of complex connectivity resulting in the use of iterative optimization schemes. Fully-connected conditional random fields (CRFs) [151] open a way for incorporating dense and long-range pixel connections while retaining fast inference.

Filtering techniques [148, 49, 109] aim to propagate information with the use of image or video filters resulting in fast runtimes compared to optimization techniques. Bilateral filtering [15, 251] is one of the popular filters for long-range information propagation.

We have already discussed bilateral filtering and its generalization in the previous chapter. A popular application, that is also discussed in previous chapter, is joint bilateral upsampling [148] that up-samples a low-resolution signal with the use of a high-resolution guidance image. Chapter 5 and the works of [169, 69, 279, 220, 21] showed that one can back-propagate through the bilateral filtering operation for learning filter parameters (Chapter 5) or doing optimization in the bilateral space [21, 20]. Recently, several works proposed to do upsampling in images by learning CNNs that mimics edge-aware filtering [270] or that directly learns to up-sample [167, 123]. Most of these works are confined to images and are either not extendible or computationally too expensive for videos. We leverage some of these previous works and propose a scalable yet robust neural network based approach for video content propagation.

**Video object segmentation**   Prior work on video object segmentation can be broadly categorized into two types: Semi-supervised methods that require manual annotation to define what is foreground object and unsupervised methods that does segmentation completely automatically. Unsupervised techniques such as [78, 166, 163, 197, 263, 277, 245, 70] use some prior information about the foreground objects such as distinctive motion, saliency etc. And, they typically fail if these assumptions do not hold in a video.

In this work, we focus on the semi-supervised task of propagating the foreground mask from the first frame to the entire video. Existing works predominantly use graph-based optimization frameworks that perform graph-cuts [31, 32, 227] on video data. Several of these works [212, 168, 203, 262, 146, 130] aim to reduce the complexity of graph structure with clustering techniques such as spatio-temporal superpixels and optical flow [255]. Another direction was to estimate correspondence between different frame pixels [9, 18, 156] by using nearest neighbor fields [79] or optical flow [55] and then refine the propagated masks with the use of local classifiers. Closest to our technique are the works of [202] and [180]. [202] proposed to use fully-connected CRF over the refined object proposals across the video frames. [180] proposed a graph-cut in the bilateral space. Our approach is similar in the regard that we also use a bilateral space embedding. Instead of graph-cuts, we learn propagation filters in the high-dimensional bilateral space with CNNs. This results in a more generic architecture and allows integration into other deep learning frameworks.

Two contemporary works [45, 141] proposed CNN based approaches for video object segmentation. Both works rely on fine-tuning a deep network using the first frame annotation of a given test sequence. This could potentially result in overfitting to the background. In contrast, the proposed approach relies only on offline training and thus can be easily adapted to different problem scenarios.

**Semantic video segmentation**   Earlier methods such as [40, 237] use structure from motion on video frames to compute geometrical and/or motion features. More recent works [76, 50, 64, 182, 253, 154] construct large graphical models on videos and enforce temporal consistency across frames. [50] used dynamic temporal links in their
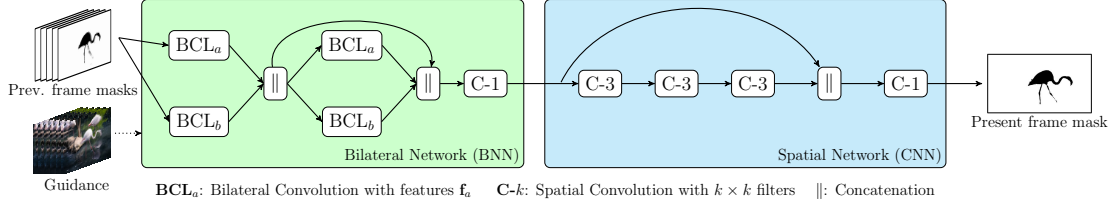
Figure 6.2: **Computation Flow of Video Propagation Network.** Bilateral networks (BNN) consist of a series of bilateral filterings interleaved with ReLU non-linearities. The filtered information from BNN is then passed into a spatial network (CNN) which refines the features with convolution layers interleaved with ReLU non-linearities, resulting in the prediction for the current frame.

CRF energy formulation. [64] proposes to use Perturb-and-MAP random field model with spatio-temporal energy terms based on Potts model and [182] propagate predictions across time by learning a similarity function between pixels of consecutive frames.

In the recent years, there is a big leap in the performance of semantic image segmentation [176, 52] with the use of CNNs but mostly applied to images. Recently, [225] proposed to retain the intermediate CNN representations while sliding the image based CNN across the frames. Another approach, which inspired our work, is to take unary predictions from CNN and then propagate semantic information across the frames. A recent prominent approach in this direction is of [154] which proposes a technique for optimizing feature spaces for fully-connected CRF.

## 6.3 Video Propagation Networks

We aim to adapt the bilateral filtering operation to predict information forward in time, across video frames. Formally, we work on a sequence of $n$ (color or grayscale) images $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ and denote with $\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$ a sequence of outputs, one per frame. Consider as an example, a sequence $\mathbf{y}_1, \ldots, \mathbf{y}_n$ of foreground masks for a moving object in the scene. Our goal is to develop an online propagation method, that is, a function that has no access to the future frames. Formally we predict $\mathbf{y}_t$, having observed the video up to frame $t$ and possibly previous $\mathbf{y}_{1, \cdots, t-1}$

$$\mathcal{F}(\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \cdots; \mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \cdots) = \mathbf{y}_t. \tag{6.1}$$

If training examples $(\mathbf{x}, \mathbf{y})$ with full or partial knowledge of $\mathbf{y}$ are available, it is possible to learn $\mathcal{F}$ and for a complex and unknown relationship between input and output, a deep CNN is a natural design choice. However, any learning based method has to face
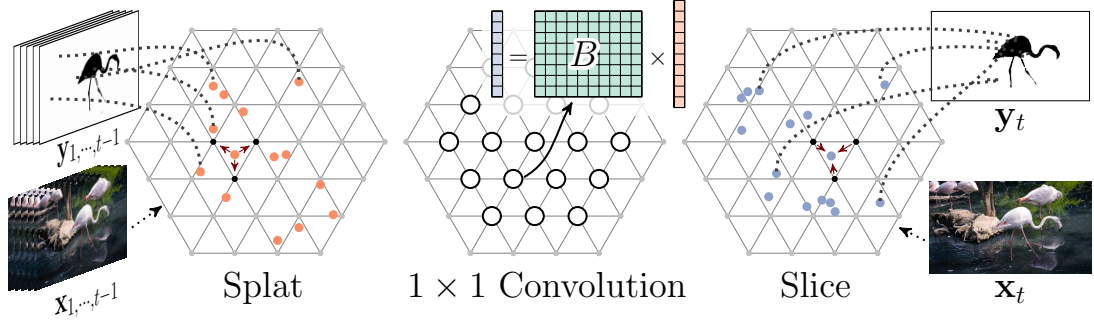
Figure 6.3: **Schematic of Fast Bilateral Filtering for Video Processing.** Mask probabilities from previous frames $V_{1,\cdots,t-1}$ are splatted on to the lattice positions defined by the image features $f_{I_1}, f_{I_2}, \cdots, f_{I_{t-1}}$. The splatted result is convolved with a $1 \times 1$ filter $B$, and the filtered result is sliced back to the original image space to get $V_t$ for the present frame. Input and output need not be $V_t$, but can also be an intermediate neural network representation. $B$ is learned via back-propagation through these operations.

the main challenge: the scene and camera motion and its effect on **y**. Since no motion in two different videos is the same, fixed-sized static receptive fields of CNN units are insufficient. We propose to resolve this with video-adaptive convolutional component, an adaption of the bilateral filtering to videos. Our Bilateral Network (Section 6.3.1) has a connectivity that adapts to video sequences, its output is then fed into a common Spatial Network (Section 6.3.2) that further refines the desired output. The combined network layout of this Video Propagation Network is depicted in Fig. 6.2. It is a sequence of learnable bilateral and spatial filters that is efficient, trainable end-to-end and adaptive to the video input.

## 6.3.1 Bilateral Network (BNN)

In this section, we describe the extension of the learnable bilateral filtering, proposed in Chapter 5 to video data. Several properties of bilateral filtering make it a perfect candidate for information propagation in videos. In particular, our method is inspired by two main ideas that we extend in this work: joint bilateral up-sampling [148] and learnable bilateral filters (Chapter 5). Although, bilateral filtering has been used for filtering video data before [198], its use has been limited to fixed filter weights (say, Gaussian).

**Fast Bilateral Up-sampling across Frames** The idea of joint bilateral up-sampling [148] is to view up-sampling as a filtering operation. A high resolution guidance image is used to up-sample a low-resolution result. In short, a smaller number of input points $\mathbf{y}_i$ and

the corresponding features $\mathbf{f}_i$ are given $\mathbf{y}_i, \mathbf{f}_i; i = 1, \ldots, N_{in}$, for example a segmentation result $\mathbf{y}_i$ at a lower resolution. This is then scaled to a larger number of output points $\mathbf{f}_j; j = 1, \ldots, N_{out}$ using the bilateral filtering operation, that is to compute the following bilateral filtering equation:

$$\mathbf{y}_i' = \sum_{j=1}^{n} \mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j} \mathbf{y}_j \tag{6.2}$$

where the sum runs over all $N_{in}$ points and the output is computed for all $N_{out}$ positions. We will use this idea to propagate content from previous frames to the current frame (all of which have the same dimensions), using the current frame as a guidance image. This is illustrated in Fig. 6.3. We take all previous frame results $\mathbf{y}_{1,\cdots,t-1}$ and splat them into a lattice using the features computed on video frames $\mathbf{x}_{1,\cdots,t-1}$. A filtering (described below) is applied to every lattice point and the result is then sliced back using the current frame $\mathbf{x}_t$. This result need not be the final $\mathbf{y}_t$, in fact we compute a filter bank of responses and continue with further processing as will be discussed.

For videos, we need to extend bilateral filtering to temporal data, and there are two natural choices. First, one can simply attach a frame index $t$ as an additional time dimension to the input data, yielding a six dimensional feature vector $\mathbf{f} = (x, y, r, g, b, t)^{\top}$ for every pixel in every frame. The summation in Eq. (6.2) now runs over *all* previous frames and pixels. Imagine a video where an object moves to reveal some background. Pixels of the object and background will be close spatially $(x, y)$ and temporally $(t)$ but likely be of different color $(r, g, b)$. Therefore they will have no strong influence on each other (being splatted to distant positions in the six-dimensional bilateral space). In summary, one can understand the filter to be adaptive to color changes across frames, only pixels that are static and have similar color have a strong influence on each other (end up nearby in the lattice space). The second possibility is to use optical flow. If the perfect flow is available, the video frames could be warped into a common frame of reference. This would resolve the corresponding problem and make information propagation much easier. We can make use of an optical flow estimate by warping pixel positions $(x, y)$ by their displacement vector $(u_x, u_y)$ to $(x + u_x, y + u_y)$.

Another property of permutohedral filtering that we exploit is that the *inputs points need not lie on a regular grid* since the filtering is done in the high-dimensional lattice. Instead of splatting millions of pixels on to the lattice, we randomly sample or use super-pixels and perform filtering using these sampled points as input to the filter. In practice, we observe that this results in big computational gains with minor drop in performance (more in Sec. 6.4.1).

**Learnable Bilateral Filters** The property of propagating information forward using a guidance image through filtering solves the problem of pixel association. But a Gaussian filter may be insufficient and further, we would like to increase the capacity by using a filter bank instead of a single fixed filter. We propose to use the technique proposed in previous chapter to learn the filter values in the permutohedral lattice using back-

propagation.

The process works as follows. A input video is used to determine the positions in the bilateral space to splat the input points $\mathbf{y}(i) \in \mathbb{R}^D$ i.e. the features $\mathbf{f}$ (e.g. $(x, y, r, g, b, t)$) define the splatting matrix $S_{splat}$. This leads to a number of vectors $\mathbf{y}_{splatted} = S_{splat}\mathbf{y}$, that lie on the permutohedral lattice, with dimensionality $\mathbf{y}_{splatted} \in \mathbb{R}^D$. In effect, the splatting operation groups points that are close together, that is, they have similar $\mathbf{f}_i, \mathbf{f}_j$. All lattice points are now filtered using a filter bank $B \in \mathbb{R}^{F \times D}$ which results in $F$ dimensional vectors on the lattice points. These are sliced back to the $N_{out}$ points of interest (present video frame). The values of $B$ are learned by back-propagation. General parameterization of $B$ from previous chapter allows to have any neighborhood size for the filters. Since constructing the neighborhood structure in high-dimensions is time consuming, we choose to use $1 \times 1$ filters for speed reasons. This makes up one *Bilateral Convolution Layer (BCL)* which we will stack and concatenate to form a Bilateral Network. See Fig. 6.3 for an illustration of a BCL.

**BNN Architecture** The Bilateral Network (BNN) is illustrated in the green box of Fig. 6.2. The input is a video sequence $\mathbf{x}$ and the corresponding predictions $\mathbf{y}$ up to frame $t$. Those are filtered using two BCLs with 32 filters each. For both BCLs, we use the same features $\mathbf{f}$ but scale them with different diagonal matrices $\mathbf{f}_a = \Lambda_a\mathbf{f}, \mathbf{f}_b = \Lambda_b\mathbf{f}$. The feature scales are found by cross-validation. The two 32 dimensional outputs are concatenated, passed through a ReLU non-linearity and passed to a second layer of two separate BCL filters that uses same feature spaces $\mathbf{f}_a, \mathbf{f}_b$. The output of the second filter bank is then reduced using a $1 \times 1$ spatial filter (C-1) to map to the original dimension of $\mathbf{y}$. We investigated scaling frame inputs with an exponential time decay and found that, when processing frame $t$, a re-weighting with $(\alpha\mathbf{y}_{t-1}, \alpha^2\mathbf{y}_{t-2}, \alpha^3\mathbf{y}_{t-3}\cdots)$ with $0 \leq \alpha \leq 1$ improved the performance a little bit.

In the experiments, we also included a simple BNN variant, where no filters are applied inside the permutohedral space, just splatting and slicing with the two layers $BCL_a$ and $BCL_b$ and adding the results. We will refer to this model as *BNN-Identity*, it corresponds to an image adaptive smoothing of the inputs $\mathbf{y}$. We found this filtering to have a positive effect and include it as a baseline in our experiments.

## 6.3.2 Spatial Network

The BNN was designed to propagate the information from the previous frames, respecting the scene and object motion. We then add a small spatial CNN with 3 layers, each with 32 filters of size $3 \times 3$, interleaved with ReLU non-linearities. The final result is then mapped to the desired output of $\mathbf{y}_t$ using a $1 \times 1$ convolution. The main role of this spatial CNN is to refine the information in frame $t$. Depending on the problem and the size of the available training data, other network designs are conceivable. We use the same network architecture shown in Fig. 6.2 for all the experiments to demonstrate the generality of VPNs.

# 6.4 Experiments

We evaluated VPN on three different propagation tasks: foreground masks, semantic labels and color information in videos. Our implementation runs in Caffe [136] using standard settings. We used Adam [144] stochastic optimization for training VPNs, multinomial-logistic loss for label propagation networks and Euclidean loss for training color propagation networks. Runtime computations were performed using a Nvidia TitanX GPU and a 6 core Intel i7-5820K CPU clocked at 3.30GHz machine. We will make available all the code and experimental results.

## 6.4.1 Video Object Segmentation

The task of class-agnostic video object segmentation aims to segment foreground objects in videos. Since the semantics of the foreground object is not pre-defined, this problem is usually addressed in a semi-supervised manner. The goal is to propagate a given foreground mask of the first frame to the entire video frames. Object segmentation in videos is useful for several high level tasks such as video editing, summarization, rotoscoping etc.

**Dataset** We use the recently published DAVIS dataset [201] for experiments on this task. The DAVIS dataset consists of 50 high-quality (1080p resolution) unconstrained videos with number of frames in each video ranging from 25 to 104. All the frames come with high-quality per-pixel annotation of the foreground object. The videos for this dataset are carefully chosen to contain motion blur, occlusions, view-point changes and other occurrences of object segmentation challenges. For robust evaluation and to get results on all the dataset videos, we evaluate our technique using 5-fold cross-validation. We randomly divided the data into 5 folds, where in each fold, we used 35 images for training, 5 for validation and the remaining 10 for the testing. For the evaluation, we used the 3 metrics that are proposed in [201]: Intersection over Union (IoU) score, Contour accuracy ($\mathcal{F}$) score and temporal instability ($\mathcal{T}$) score. The widely used IoU score is defined as $TP/(TP + FN + FP)$, where TP: True positives; FN: False negatives and FP: False positives. Please refer to [201] for the definition of the contour accuracy and temporal instability scores. We are aware of some other datasets for this task such as JumpCut [79] and SegTrack [254], but we note that the number of videos in these datasets is too small for a learning based approach.

**VPN and Results** In this task, we only have access to foreground mask for the first frame $V_1$. For the ease of training VPN, we obtain initial set of predictions with *BNN-Identity*. We sequentially apply *BNN-Identity* at each frame and obtain an initial set of foreground masks for the entire video. These BNN-Identity propagated masks are then used as inputs to train a VPN to predict the refined masks at each frame. We refer to this VPN model as *VPN-Stage1*. Once VPN-Stage1 is trained, its refined training mask predictions are in-turn used as inputs to train another VPN model which we refer to as

|             | Fold-1 | Fold-2 | Fold-3 | Fold-4 | Fold-5 | All  |
|-------------|--------|--------|--------|--------|--------|------|
| BNN-Identity | 56.4   | 74.0   | 66.1   | 72.2   | 66.5   | 67.0 |
| VPN-Stage1  | 58.2   | 77.7   | 70.4   | 76.0   | 68.1   | 70.1 |
| VPN-Stage2  | **60.9** | **78.7** | **71.4** | **76.8** | **69.0** | **71.3** |

Table 6.1: **5-Fold Validation on DAVIS Video Segmentation Dataset.** Average IoU scores for different models on the 5 folds.
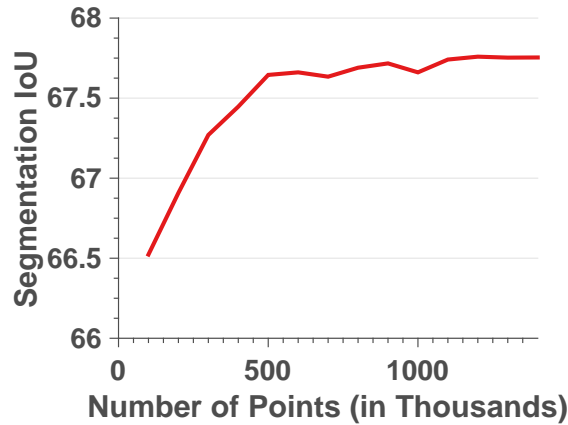


Figure 6.4: **Random Sampling of Input Points vs. IoU.** The effect of randomly sampling points from input video frames on object segmentation IoU of BNN-Identity on DAVIS dataset. The points sampled are out of $\approx$2 million points from the previous 5 frames.

*VPN-Stage2.* This resulted in further refinement of foreground masks. Training further stages did not result in any improvements.

Following the recent work of [180] on video object segmentation, we used scaled features $\mathbf{f} = (x, y, Y, Cb, Cr, t)$ with YCbCr color features for bilateral filtering. To be comparable with the one of the fastest state-of-the-art technique [180], we do not use any optical flow information. First, we analyze the performance of BNN-Identity by changing the number of randomly sampled input points. Figure 6.4 shows how the segmentation IoU changes with the increase in the number of sampled points (out of 2 million points) from the previous frames. The IoU levels out after sampling 25% of points. For further computational efficiency, we used superpixel sampling instead of random sampling. Usage of superpixels reduced the IoU slightly (0.5%), while reducing the number of input points by a factor of 10 in comparison to a large number of randomly sampled points. We used 12000 SLIC [6] superpixels from each frame computed using the fast GPU implementation from [210]. For predictions at each frame, we input mask probabilities of previous 9 frames into VPN as we observe no significant improvements with more frames. We set

|  | *IoU*↑ | $\mathcal{F}$↑ | $\mathcal{T}$↓ | *Runtime*(s) |
|---|---|---|---|---|
| BNN-Identity | 67.0 | 67.1 | 36.3 | 0.21 |
| VPN-Stage1 | 70.1 | 68.4 | 30.1 | 0.48 |
| VPN-Stage2 | 71.3 | 68.9 | 30.2 | 0.75 |
| *With pre-trained models* | | | | |
| DeepLab | 57.0 | 49.9 | 47.8 | 0.15 |
| VPN-DeepLab | **75.0** | **72.4** | 29.5 | 0.63 |
| OFL [255] | 71.1 | 67.9 | 22.1 | >60 |
| BVS [180] | 66.5 | 65.6 | 31.6 | 0.37 |
| NLC [78] | 64.1 | 59.3 | 35.6 | 20 |
| FCP [202] | 63.1 | 54.6 | 28.5 | 12 |
| JMP [79] | 60.7 | 58.6 | **13.2** | 12 |
| HVS [106] | 59.6 | 57.6 | 29.7 | 5 |
| SEA [206] | 55.6 | 53.3 | 13.7 | 6 |

Table 6.2: **Results of Video Object Segmentation on DAVIS dataset.** Average IoU score, contour accuracy ($\mathcal{F}$), temporal instability ($\mathcal{T}$) scores, and average runtimes (in seconds) per frame for different VPN models along with recent published techniques for this task. VPN runtimes also include superpixel computation (10ms). Runtimes of other methods are taken from [180, 202, 255] and only indicative and are not directly comparable to our runtimes. Runtime of VPN-Stage1 includes the runtime of BNN-Identity which is in-turn included in the runtime of VPN-Stage2. Runtime of VPN-DeepLab model includes the runtime of DeepLab.

$\alpha$ to 0.5 and the feature scales for bilateral filtering are presented in Tab. A.4.

Table 6.1 shows the IoU scores for each of the 5 folds and Tab. 6.2 shows the overall scores and runtimes of different VPN models along with the best performing segmentation techniques. The performance improved consistently across all 5 folds with the addition of new VPN stages. BNN-Identity already performed reasonably well. And with 1-stage and 2-stage VPNs, we outperformed the present fastest BVS method [180] by a significant margin on all the performance measures of IoU, contour accuracy and temporal instability scores, while being comparable in runtime. We perform marginally better than OFL method [255] while being at least $80\times$ faster and OFL relies on optical flow whereas we obtain similar performance without using any optical flow. Further, VPN has the advantage of doing online processing as it looks only at previous frames whereas BVS processes entire video at once. One can obtain better VPN performance with using better superpixels and also incorporating optical flow, but this increases runtime as well. Figure 6.5 shows some qualitative results and more are present in Figs. A.16. A couple of failure cases are shown in Fig. A.17. Visual results indicate that learned VPN is able to retain foreground masks even with large variations in viewpoint and object size.

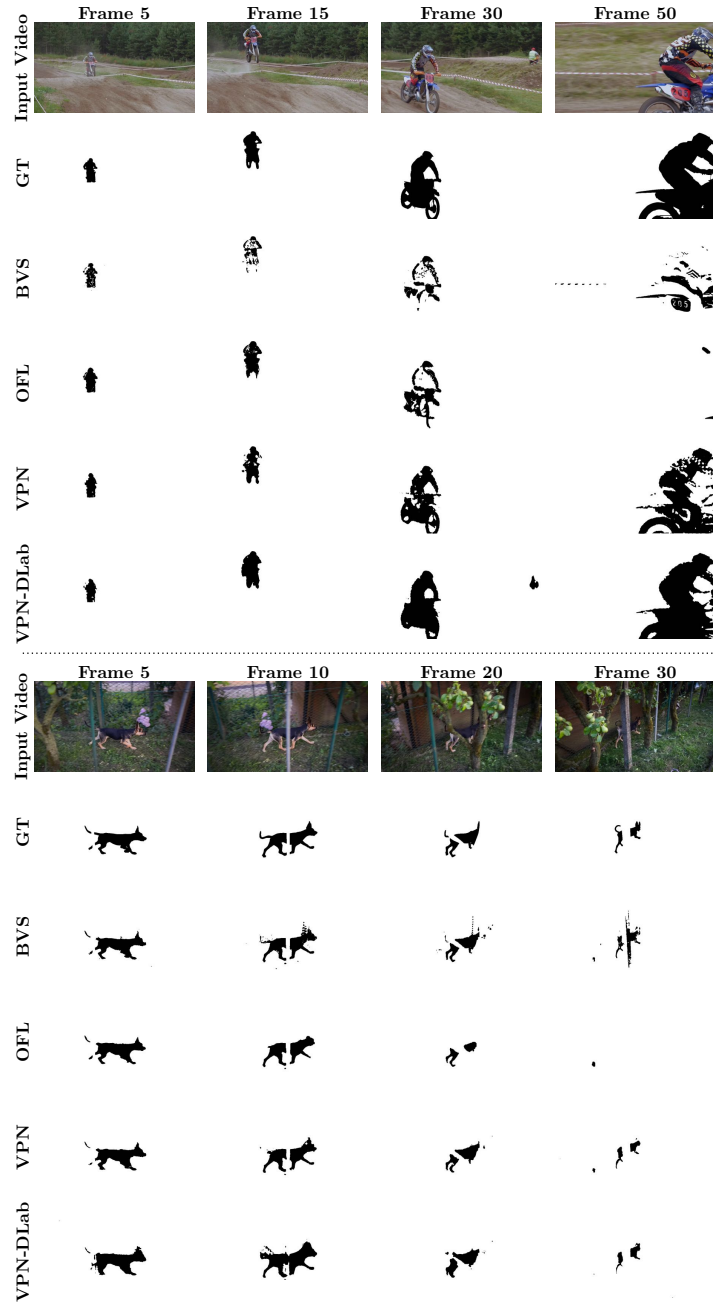Figure 6.5: **Video Object Segmentation.**  Shown are the different frames in example videos with the corresponding ground truth (GT) masks, predictions from BVS [180], OFL [255], VPN (VPN-Stage2) and VPN-DLab (VPN-DeepLab) models.

**Augmenation of Pre-trained Models:**   One of the main advantages of the proposed VPN architecture is that it is end-to-end trainable and can be easily integrated into other

deep neural network architectures. To demonstrate this, we augmented VPN architecture with standard DeepLab segmentation architecture from [52]. We replaced the last classification layer of DeepLab-LargeFOV model from [52] to output 2 classes (foreground and background) in our case and bi-linearly up-sampled the resulting low-resolution probability map to the original image dimension. 5-fold fine-tuning of the DeepLab model on DAVIS dataset resulted in the IoU of 57.0 and other scores are shown in Tab. 6.2. Then, we combine the VPN and DeepLab models in the following way: The output from the DeepLab network and the bilateral network are concatenated and then passed on to the spatial network. In other words, the bilateral network propagates label information from previous frames to the present frame, whereas the DeepLab network does the prediction for the present frame. The results of both are then combined and refined by the spatial network in the VPN architecture. We call this 'VPN-DeepLab' model. We trained this model end-to-end and observed big improvements in performance. As shown in Tab. 6.2, the VPN-DeepLab model has the IoU score of 75.0 and contour accuracy score of 72.4 resulting in significant improvements over the published results. Since DeepLab has also fast runtime, the total runtime of VPN-DeepLab is only 0.63s which makes this also one of the fastest video segmentation systems. A couple of visual results of VPN-DeepLab model are shown in Fig. 6.5 and more are present in Figs. A.16 and A.17.

## 6.4.2 Semantic Video Segmentation

A semantic video segmentation assigns a semantic label to every video pixel. Since the semantics between adjacent frames does not change radically, intuitively, propagating semantic information across frames should improve the segmentation quality of each frame. Unlike mask propagation in the previous section where the ground-truth mask for the first frame is given, we approach semantic video segmentation in a fully automatic fashion. Specifically, we start with the unary predictions of standard CNNs and use VPN for propagating semantics across the frames.

**Dataset**   We use the CamVid dataset [39] that contains 4 high quality videos captured at 30Hz while the semantically labelled 11-class ground truth is provided at 1Hz. While the original dataset comes at a resolution of 960×720, similar to previous works [273, 154], we operate on a resolution of 640×480. We use the same splits proposed in [237] resulting in 367, 100 and 233 frames with ground-truth for training, validation and testing. Following common practice, we report the IoU scores for evaluation.

**VPN and Results**   Since we already have CNN predictions for every frame, we train a VPN that takes the CNN predictions of previous *and* present frames as input and predicts the refined predictions for the present frame. We compare with the state-of-the-art CRF approach for this problem [154] which we refer to as 'FSO-CRF'. Following [154], we also experimented with optical flow in our framework and refer that model as *VPN-Flow*. We used the fast optical flow method that uses dense inverse search [153] to compute

105

|  | *IoU* | *Runtime*(s) |
|---|---|---|
| CNN from  [273] | 65.3 | 0.38 |
| + FSO-CRF [154] | 66.1 | >10 |
| + BNN-Identity | 65.3 | 0.31 |
| + BNN-Identity-Flow | 65.5 | 0.33 |
| + VPN (Ours) | 66.5 | 0.35 |
| + VPN-Flow (Ours) | **66.7** | 0.37 |
| CNN from  [214] | 68.9 | 0.30 |
| + VPN-Flow (Ours) | **69.5** | 0.38 |

Table 6.3: **Results of Semantic Segmentation on the CamVid Dataset.** Average IoU and runtimes (in seconds) per frame of different models on *test* split. Runtimes exclude CNN computations which are shown separately. VPN and BNN-Identity runtimes include superpixel computation which takes up large portion of computation time (0.23s).



Input  GT  CNN  +VPN(Ours)

Figure 6.6: **Semantic Video Segmentation.** Input video frames and the corresponding ground truth (GT) segmentation together with the predictions of CNN [273] and with VPN-Flow.

flows and modify the positional features of previous frames. We used the superpixels method of Dollar et al. [66] for this dataset as gSLICr [210] has introduced artifacts.

We experimented with predictions from two different CNNs: One is with dilated convolutions [273] (CNN-1) and another one [214] (CNN-2) is trained with the additional

|              | *PSNR* | *Runtime*(s) |
| ------------ | ------ | ------------ |
| BNN-Identity | 27.89  | 0.29         |
| VPN-Stage1   | **28.15** | 0.90      |
| Levin et al. [164] | 27.11 | 19       |

Table 6.4: **Results of Video Color Propagation.** Average PSNR results and runtimes of different methods for video color propagation on images from DAVIS dataset.

data obtained from a video game, which is the present state-of-the-art on this dataset. For CNN-1 and CNN-2, using 2 and 3 previous frames respectively as input to VPN is found to be optimal. Other parameters of the bilateral network are presented in Tab. A.4. Table 6.3 shows quantitative results on this dataset. Using BNN-Identity only slightly improved the CNN performance whereas training the entire VPN significantly improved the CNN performance by over 1.2% IoU, with both VPN and VPN-Flow networks. Moreover, VPN is at least 25× faster, and simpler to use compared to the optimization based FSO-CRF which relies on LDOF optical flow [41], long-term tacks [240] and edges [67]. We further improved the performance of the state-of-the-art CNN [214] with the use of VPN-Flow model. Using better optical flow estimation might give even better results. Figure A.14 shows some qualitative results and more are presented in Fig. A.18.

## 6.4.3 Video Color Propagation

We also evaluate VPNs on a different kind of information and experimented with propagating color information in a grayscale video. Given the color image for the first video frame, the task is to propagate the color to the entire video. Note that this task is fundamentally different from automatic colorization of images for which recent CNN based based methods have become popular. For experiments on this task, we again used the DAVIS dataset [201] with the first 25 frames from each video. We randomly divided the dataset into 30 train, 5 validation and 15 test videos.

We work with YCbCr representation of images and propagate CbCr values from previous frames with pixel intensity, position and time features as guidance for VPN. The same strategy as in object segmentation is used, where an initial set of color propagated results was obtained with BNN-Identity and then used to trained a VPN-Stage1 model. Training further VPN stages did not improve the performance. Table 6.4 shows the PSNR results. We use 300K radomly sampled points from previous 3 frames as input to the VPN network. We also show a baseline result of [164] that does graph based optimization and uses optical flow. We used fast DIS optical flow [153] in the baseline method [164] and we did not observe significant differences with using LDOF optical flow [41]. Figure 6.7 shows a visual result with more in Fig. A.19. From the results, VPN works reliably better than [164] while being 20× faster. The method of [164] relies

Figure 6.7: **Video Color Propagation.** Input grayscale video frames and corresponding ground-truth (GT) color images together with color predictions of Levin et al. [164] and VPN-Stage1 models.

heavily on optical flow and so the color drifts away with incorrect flow. We observe that our method also bleeds color in some regions especially when there are large viewpoint changes. We could not compare against recent video color propagation techniques such as [114, 226] as their codes are not available online. This application shows general applicability of VPNs in propagating different kinds of information.

## 6.5  Discussion and Conclusions

We proposed a fast, scalable and generic neural network based learning approach for propagating information across video frames. The video propagation network uses bilateral network for long-range video-adaptive propagation of information from previous frames to the present frame which is then refined by a standard spatial network. Experiments on diverse tasks show that VPNs, despite being generic, outperformed the current state-of-the-art task-specific methods. At the core of our technique is the exploitation and modification of learnable bilateral filtering for the use in video processing. We used a simple and fixed network architecture for all the tasks for showcasing the generality of the approach. Depending on the type of problems and the availability of data, using more filters and deeper layers would result in better performance. In this work, we manually tuned the feature scales which could be amendable to learning. Finding optimal yet fast-to-compute bilateral features for videos together with the learning of their scales is an important future research direction.

# Chapter 7

# Bilateral Inception Networks

Following up on previous chapters, where we introduced learnable bilateral filters and their application to a wide range of problems, in this chapter, we construct a CNN module which we call 'Bilateral Inception' that can be inserted into *existing* CNN architectures for better inference in pixel prediction tasks. The bilateral inception module performs bilateral filtering, at multiple feature-scales, between superpixels in an image. The feature spaces for bilateral filtering and other parameters of the module are learned end-to-end using standard back-propagation techniques. Instead of using learnable bilateral filtering proposed in Chapter 5, here, we explicitly construct the Gaussian filter kernel between input and output superpixels. We show how this explicit Gaussian filtering results in fast runtimes and also enables the learning of bilateral features.

We focus on the problem of semantic segmentation. The bilateral inception module addresses two issues that arise with general CNN segmentation architectures. First, this module propagates information between (super)pixels while respecting image edges, thus using the structured information of the problem for improved results. Second, the layer recovers a full resolution segmentation result from the lower resolution solution of a CNN.

In the experiments, we modify several existing CNN architectures by inserting our inception module between the last CNN ($1 \times 1$ convolution) layers. Empirical results on three different datasets show reliable improvements not only in comparison to the baseline networks, but also in comparison to several dense-pixel prediction techniques such as CRFs, while being competitive in time.

## 7.1 Introduction

In this work, we propose a CNN architecture for semantic image segmentation. Given an image $\mathbf{x} = (x_1, \ldots, x_N)$ with $N$ pixels $x_i$, the task of semantic segmentation is to infer a labeling $\mathbf{y} = (y_1, \ldots, y_N)$ with a label $y_i \in \mathcal{Y}$ for every pixel. This problem can be naturally formulated as a structured prediction problem $g : \mathbf{x} \to \mathbf{y}$. Empirical performance is measured by comparing $\mathbf{y}$ to a human labeled $\mathbf{y}^*$ via a loss function $\Delta(\mathbf{y}, \mathbf{y}^*)$, *e.g.* with the Intersection over Union (IoU) or pixel-wise Hamming Loss.
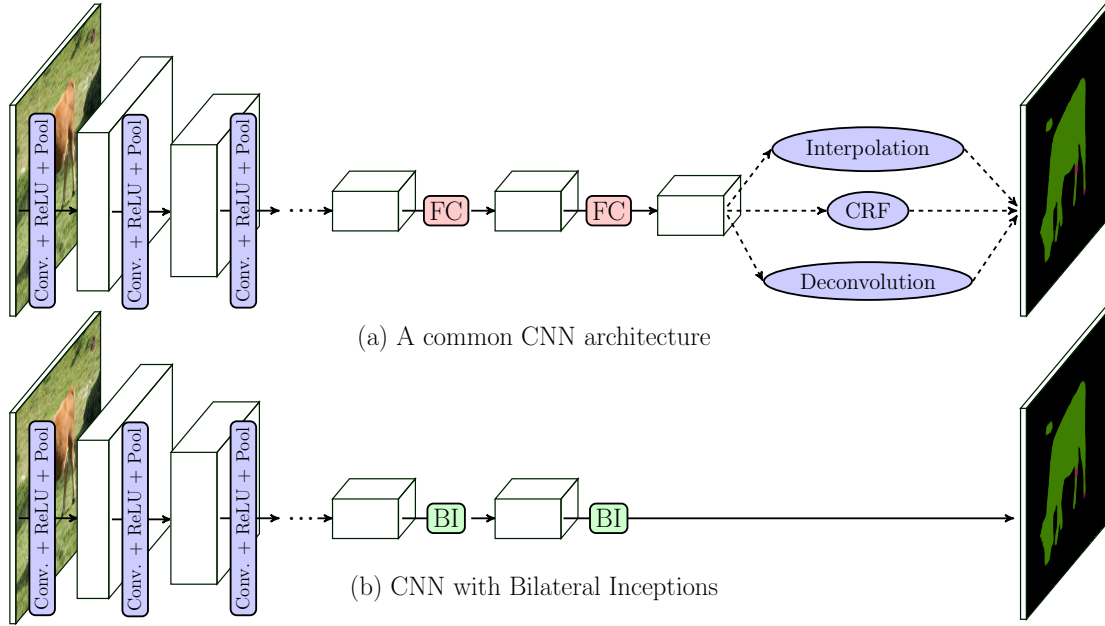
(a) A common CNN architecture

(b) CNN with Bilateral Inceptions

Figure 7.1: **Illustration of CNN layout.** We insert the *Bilateral Inception (BI)* modules between the *FC* ($1 \times 1$ convolution) layers found in most networks thus removing the necessity of further up-scaling algorithms. Bilateral Inception modules also propagate information between distant pixels based on their spatial and color similarity and work better than other label propagation approaches.

A direct way to approach this problem would be to ignore the structure of the output variable **y** and train a classifier that predicts the class membership of the center pixel of a given image patch. This procedure reduces the problem to a standard multi-class classification problem and allows the use of standard learning algorithms. The resulting classifier is then evaluated at every possible patch in a sliding window fashion (or using coarse-to-fine strategies) to yield a full segmentation of the image. With high capacity models and large amounts of training data, this approach would be sufficient, given that the loss decomposes over the pixels. Such a per-pixel approach ignores the relationship between the variables $(y_1, \ldots, y_N)$, which are not i.i.d. since there is an underlying common image. Therefore, besides learning discriminative per-pixel classifiers, most segmentation approaches further encode the output relationship of **y**. A dominating approach is to use Conditional Random Fields (CRF) [155], which allows an elegant and principled way to combine single pixel predictions and shared structure through unary, pairwise and higher order factors.

What relates the outputs $(y_1, \ldots, y_N)$? The common hypothesis that we use in this chapter could be summarized as: *Pixels that are spatially and photometrically similar*

*are more likely to have the same label.* Particularly if two pixels $x_i, x_j$ are close in the image and have similar *RGB* values, then their corresponding labels $y_i, y_j$ will most likely be the same. The most prominent example of spatial similarity encoded in a CRF is the Potts model (Ising model for the binary case). The work of [151] described a densely connected pairwise CRF (DenseCRF) that includes pairwise factors encoding both spatial *and* photometric similarity. The DenseCRF has been used in many recent works on image segmentation which find also empirically improved results over pure pixel-wise CNN classifiers [52, 25, 279, 51].

In this chapter, we implement the above-mentioned hypothesis of nearby pixels which are photometrically similar sharing a common label, by designing a new 'Bilateral Inception' (BI) module that can be inserted before/after the last $1 \times 1$ convolution layers (which we refer to as 'FC' layers - 'Fully-Connected' in the original image classification network) of the standard segmentation CNN architectures. The bilateral inception module does edge-aware information propagation across different spatial CNN units of the previous FC layer. Instead of using the spatial grid-layout that is common in CNNs, we incorporate the superpixel-layout for information propagation. The information propagation is performed using standard bilateral filters with Gaussian kernels, at different feature scales. This construction is inspired by [242, 172]. Feature spaces and other parameters of the modules can be learned end-to-end using standard back-propagation techniques. The application of superpixels reduces the number of necessary computations and implements a long-range edge-aware inference between different superpixels. Moreover, since superpixels provides an output at the full image resolution, it removes the need for any additional post-processing step.

We introduce BI modules in the CNN segmentation models of [52, 279, 25]. See Fig. 7.1 for an illustration. This achieves better segmentation results than the proposed interpolation/inference techniques of DenseCRF [25, 52], on all three datasets that we experimented with, while being faster. Moreover, the results compare favorably against some recently proposed dense pixel prediction techniques. As illustrated in Fig. 7.1, the BI modules provide an alternative approach to commonly used up-sampling and CRF techniques.

## 7.2 Related Work

The literature on semantic segmentation is large and therefore we will limit our discussion to those works that perform segmentation with CNNs and discuss the different ways to encode the output structure.

A natural combination of CNNs and CRFs is to use the CNN as unary potential and combine it with a CRF that also includes pairwise or higher order factors. For instance [52, 25] observed large improvements in pixel accuracy when combining a DenseCRF [151] with a CNN. The mean-field steps of the DenseCRF can be learned and back-propagated as noted by [69] and implemented by [279, 134, 169, 220] for se-

mantic segmentation and [142] for human pose estimation. The works of [54, 170, 175] use CNNs also in pairwise and higher order factors for more expressiveness. The recent work of [51] replaced the costly DenseCRF with a faster domain transform performing smoothing filtering while predicting the image edge maps at the same time. Our work was inspired by DenseCRF approaches but with the aim to replace the expensive mean-field inference. Instead of propagating information across unaries obtained by a CNN, we aim to do the edge-aware information propagation across *intermediate* representations of the CNN. Experiments on different datasets indicate that the proposed approach generally gives better results in comparison to DenseCRF while being faster.

A second group of works aims to inject the structural knowledge in intermediate CNN representations by using structural layers among CNN internal layers. The deconvolution layers model from [276] are being widely used for local propagation of information. They are computationally efficient and are used in segmentation networks, *e.g.* [176]. They are however limited to small receptive fields. Another architecture proposed in [110] uses spatial pyramid pooling layers to max-pool over different spatial scales. The work of [126] proposed specialized structural layers such as normalized-cut layers with matrix back-propagation techniques. All these works have either fixed local receptive fields and/or have their complexity increasing exponentially with longer range pixel connections. Our technique allows for modeling long range (super)pixel dependencies without compromising the computational efficiency. A very recent work [273] proposed the use of dilated convolutions for propagating multi-scale contextual information among CNN units.

A contribution of this work is to define convolutions over superpixels by defining connectivity among them. In [112], a method to use superpixels inside CNNs has been proposed by re-arranging superpixels based on their features. The technique proposed here is more generic and alleviates the need for rearranging superpixels. A method to filter irregularly sampled data has been developed in [42] which may be applicable to superpixel convolutions. The difference being that their method requires a pre-defined graph structure for every example/image separately while our approach directly works on superpixels. We experimented with Isomap embeddings [248] of superpixels but for speed reasons opted for the more efficient kernels presented in this chapter. The work of [187] extracted multi-scale features at each superpixel and performs semantic segmentation by classifying each superpixel independently. In contrast, we propagate information across superpixels by using bilateral filters with learned feature spaces.

Another core contribution of this work is the end-to-end trained bilateral filtering module. Several recent works on bilateral filtering [21, 20] (including ours in Chapter 5) back-propagate through permutohedral lattice approximation [7], to either learn the filter parameters (Chapter 5) or do optimization in the bilateral space [21, 20]. Most of the existing works on bilateral filtering use pre-defined feature spaces. In [47], the feature spaces for bilateral filtering are obtained via a non-parametric embedding into an Euclidean space. In contrast, by explicitly computing the bilateral filter kernel, we are able to back-propagate through features, thereby learning the task-specific feature spaces for
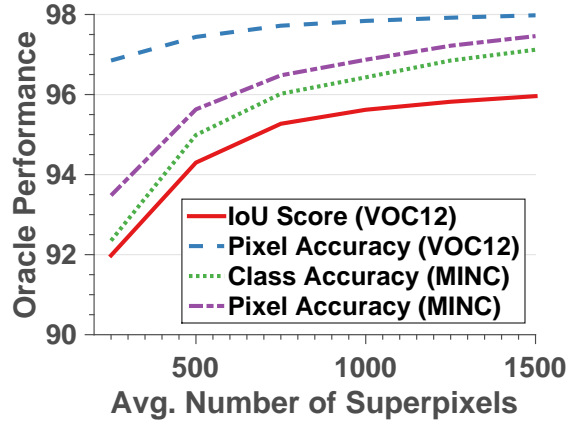
Figure 7.2: **Superpixel Quantization Error.** Best achievable segmentation performance with a varying number of SLIC superpixels [6] on Pascal VOC12 segmentation [77] and MINC material segmentation [25] datasets.

bilateral filters through integration into end-to-end trainable CNNs.

## 7.3 Bilateral Inception Networks

We first formally introduce superpixels in Sec. 7.3.1 before we describe the bilateral inception modules in Sec. 7.3.2.

### 7.3.1 Superpixels

The term *superpixel* refers to a set of $n_i$ pixels $s_i = \{t_1, \ldots, t_{n_i}\}$ with $t_k \in \{1, \ldots, N\}$ pixels. We use a set of $M$ superpixels $S = \{s_1, \ldots, s_M\}$ that are disjoint $s_i \cap s_j = \emptyset, \forall i, j$ and decompose the image, $\cup_i s_i = \mathcal{I}$.

Superpixels have long been used for image segmentation in many previous works, *e.g.* [99, 98, 193, 187], as they provide a reduction of the problem size. Instead of predicting a label $y_i$ for every pixel $x_i$, the classifier predicts a label $y_i$ per superpixel $S_i$ and extends this label to all pixels within. A superpixel algorithm can pre-group pixels based on spatial and photometric similarity, reducing the number of elements and also thereby regularizing the problem in a meaningful way. The downside is that superpixels introduce a quantization error whenever pixels within one segment have different ground truth label assignments.

Figure 7.2 shows the superpixel quantization effect with the best achievable performance as a function in the number of superpixels, on two different segmentation datasets: PascalVOC [77] and Materials in Context [25]. We find that the quantization effect is small compared to the current best segmentation performance. Practically, we use the
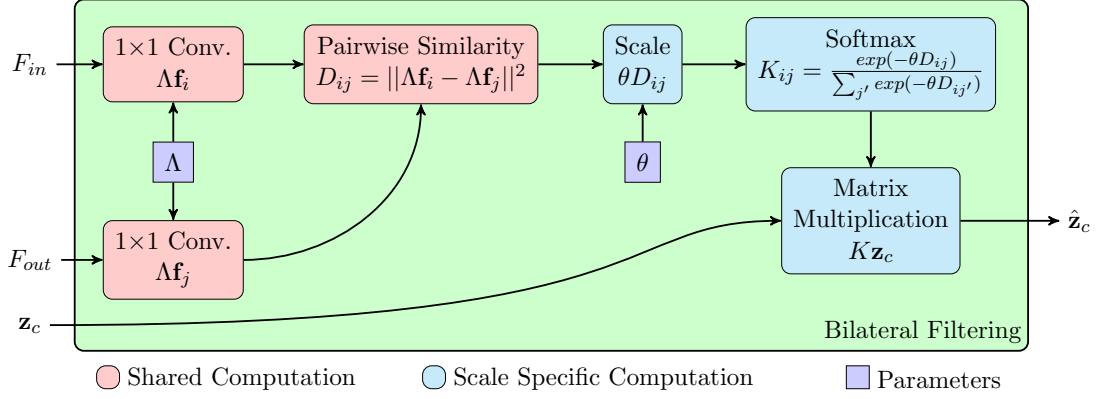
Figure 7.3: **Computation flow of the Gaussian bilateral filtering.** We implemented the bilateral convolution with five separate computation blocks. $\Lambda$ and $\theta$ are the free parameters.

SLIC superpixels [6] for their runtime and [66] for their lower quantization error to decompose the image into superpixels. For details of the algorithms, please refer to the respective papers. We use the publicly-available real-time GPU implementation of SLIC, called gSLICr [210], which runs at over 250 frames per second. And the publicly available Dollar superpixels code [66] computes a super-pixelization for a $400 \times 500$ image in about 300ms using an Intel Xeon 3.33GHz CPU.

## 7.3.2 Bilateral Inceptions

Next, we describe the *Bilateral Inception* (BI) module that performs Gaussian bilateral filtering on multiple scales of the representations within a CNN. The BI module can be inserted in between layers of existing CNN architectures.

**Bilateral Filtering:** We first describe the Gaussian bilateral filtering, the building block of the BI module. A visualization of the necessary computations is shown in Fig. 7.3. Let the previous layer CNN activations be $\mathbf{z} \in \mathbb{R}^{P \times C}$, that is $P$ points and $C$ filter responses. With $\mathbf{z}_c \in \mathbb{R}^P$ we denote the vector of activations of filter $c$. Additionally we have for every point $j$ a feature vector $\mathbf{f}_j \in \mathbb{R}^D$. This denotes its spatial position ($D = 2$, not necessarily a grid), position and RGB color ($D = 5$), or others. Separate from the input points with features $F_{in} = \{\mathbf{f}_1, \ldots, \mathbf{f}_P\}$, we have $Q$ output points with features $F_{out}$. These can be the same set of points, but also fewer ($Q < P$), equal ($Q = P$), or more ($Q > P$) points. For example we can filter a $10 \times 10$ grid ($P = 100$) and produce the result on a $50 \times 50$ grid ($Q = 2500$) or vice versa.

The bilateral filtered result will be denoted as $\hat{\mathbf{z}} \in \mathbb{R}^{Q \times C}$. We apply the same Gaussian bilateral filter to every channel $c$ separately. A filter has two free parameters: the
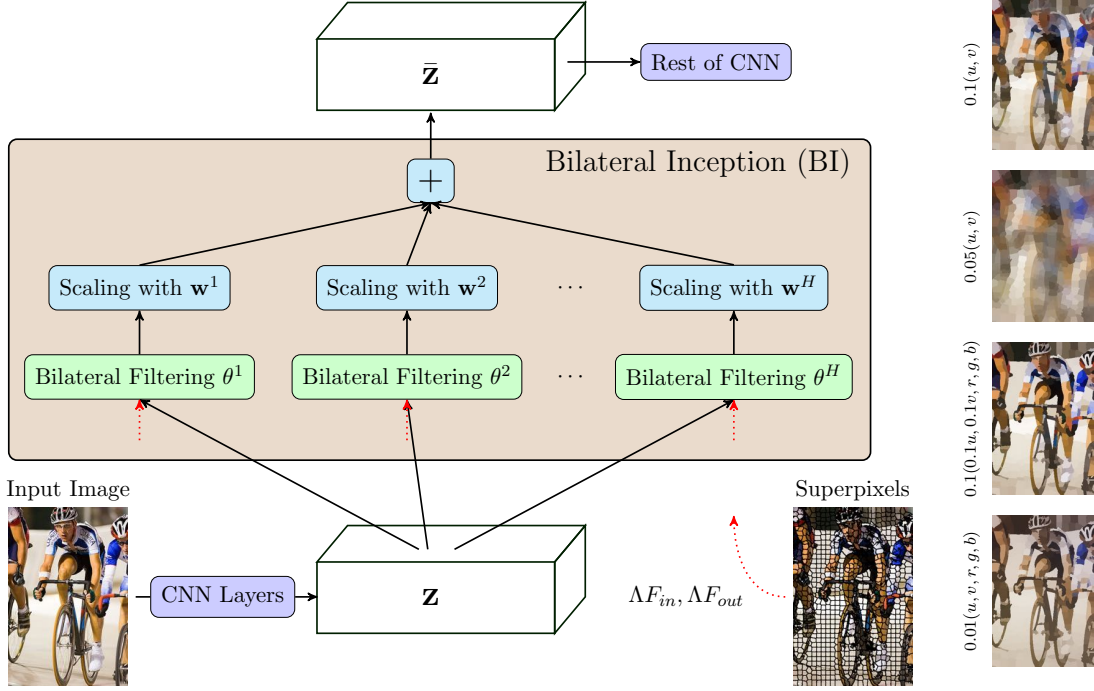
Figure 7.4: **Visualization of a Bilateral Inception (BI) Module.** The unit activations **z** are passed through several bilateral filters defined over different feature spaces. The result is linearly combined to z̄ and passed on to the next network layer. Also shown are sample filtered superpixel images using bilateral filters defined over different example feature spaces. $(u, v)$ correspond to position and $(r, g, b)$ correspond to color features.

filter specific scale $\theta \in \mathbb{R}_+$ and the global feature transformation parameters $\Lambda \in \mathbb{R}^{D \times D}$. For $\Lambda$, a more general scaling could be applied using more features or a separate CNN. Technically the bilateral filtering amounts to a matrix-vector multiplication for all $c$:

$$\hat{\mathbf{z}}_c = K(\theta, \Lambda, F_{in}, F_{out}) \mathbf{z}_c, \tag{7.1}$$

where $K \in \mathbb{R}^{Q \times P}$ and values for $f_i \in F_{out}, f_j \in F_{in}$:

$$K_{i,j} = \frac{\exp(-\theta \| \Lambda \mathbf{f}_i - \Lambda \mathbf{f}_j \|^2)}{\sum_{j'} \exp(-\theta \| \Lambda \mathbf{f}_i - \Lambda \mathbf{f}_{j'} \|^2)}. \tag{7.2}$$

From a kernel learning terminology, $K$ is nothing but a Gaussian Gram matrix and it is symmetric if $F_{in} = F_{out}$. We implemented this filtering in Caffe [136] using different layers as depicted in Fig. 7.3. While approximate computations of $K\mathbf{z}_c$ exist and have improved runtime [7, 199, 91, 8], we chose an explicit and exact computation of $K$ due to

its small size. Our implementation makes use of the GPU and the intermediate pairwise similarity computations are re-used across different modules. The entire runtime is only a fraction of the CNN runtime, but of course applications to larger values of $P$ and $Q$ would require aforementioned algorithmic speed-ups.

**Bilateral Inception Module:** The *bilateral inception module* (BI) is a weighted combination of different bilateral filters. We combine the output of $H$ different filter kernels $K$, with different scales $\theta^1, \ldots, \theta^H$. All kernels use the same feature transformation $\Lambda$ which allows for easier pre-computation of pairwise difference and avoids an over-parametrization of the filters. The outputs of different filters $\hat{\mathbf{z}}^h$ are combined linearly to produce $\bar{\mathbf{z}}$:

$$\bar{\mathbf{z}}_c = \sum_{h=1}^{H} \mathbf{w}_c^h \hat{\mathbf{z}}_c^h, \tag{7.3}$$

using individual weights $\mathbf{w}_c^h$ per scale $\theta^h$ and channel $c$. The weights $\mathbf{w} \in \mathbb{R}^{H \times C}$ are learned using error back-propagation. The result of the inception module has $C$ channels for every of its $Q$ points, thus $\bar{\mathbf{z}} \in \mathbb{R}^{Q \times C}$. The inception module is schematically illustrated in Fig. 7.4. In short, information from CNN layers below is filtered using bilateral filters defined in a transformed feature space ($\Lambda \mathbf{f}$). Most operations in the inception module are parallelizable, resulting in fast runtimes on a GPU. In this work, inspired from the DenseCRF architecture from [151], we used pairs of BI modules: one with position features $(u, v)$ and another with both position and color features $(u, v, r, g, b)$, each with multiple scales $\{\theta^h\}$.

**Motivation and Comparison to DenseCRF:** A BI module filters the activations of a CNN layer. Contrast this with the use of a DenseCRF on the CNN output. At that point the fine-grained information that intermediate CNN layers represent has been condensed already to a low-dimensional vector representing beliefs over labels. Using a mean-field update is propagating information between these beliefs. Similar behavior is obtained using the BI modules but on different scales (using multiple different filters $K(\theta^h)$) and on the intermediate CNN activations $\mathbf{z}$. Since in the end, the to-be-predicted pixels are not i.i.d., this blurring leads to better performance both when using a bilateral filter as an approximate message passing step of a DenseCRF as well as in the system outlined here. Both attempts are encoding prior knowledge about the problem, namely that pixels close in position and color are likely to have the same label. Therefore such pixels can also have the same intermediate representation. Consider one would average CNN representations for all pixels that have the same ground truth label. This would result in an intermediate CNN representation that would be very easy to classify for the later layers.

## 7.3.3 Superpixel Convolutions

The bilateral inception module allows to change how information is stored in the higher level of a CNN. This is where the superpixels are used. Instead of storing information

on a fixed grid, we compute for every image, superpixels *S* and use the mean color and position of their included pixels as features. We can insert bilateral inception modules to change from grid representations to superpixel representations and vice versa. Inception modules in between superpixel layers convolve the unit activations between all superpixels depending on their distance in the feature space. This retains all properties of the bilateral filter, superpixels that are spatially close and have a similar mean color will have a stronger influence on each other.

Superpixels are not the only choice, in principle one can also sample random points from the image and use them as intermediate representations. We are using superpixels for computational reasons, since they can be used to propagate label information to the full image resolution. Other interpolation techniques are possible, including the well known bilinear interpolation, up-convolution networks [276], and DenseCRFs [151]. The quantization error mentioned in Sec. 7.3.1 only enters because the superpixels are used for interpolation. Also note that a fixed grid, that is independent of the image is a hard choice of where information should be stored. One could in principle evaluate the CNN densely, at all possible spatial locations, but we found that this resulted in poor performance compared to interpolation methods.

**Back-propagation and Training.**

All free parameters of the inception module $\mathbf{w}$, $\{\theta^h\}$ and $\Lambda$ are learned via back-propagation. We also back-propagate the error with respect to the module inputs thereby enabling the integration of our inception modules inside CNN frameworks without breaking the end-to-end learning paradigm. As shown in Fig. 7.3, the bilateral filtering can be decomposed into 5 different sub-layers. Derivatives with respect to the open parameters are obtained by the corresponding layer and standard back-propagation through the directed acyclic graph. For example, $\Lambda$ is optimized by back-propagating gradients through $1 \times 1$ convolution. Derivatives for non-standard layers (pairwise similarity, matrix multiplication) are straight forward to obtain using matrix calculus. To let different filters learn the information propagation at different scales, we initialized $\{\theta^h\}$ with well separated scalar values (*e.g.* $\{1, 0.7, 0.3, ...\}$). The learning is performed using the stochastic optimization method of Adam [144]. The implementation is done in the Caffe neural network framework [136], and the code is available at http://segmentation.is.tuebingen.mpg.de.

## 7.4 Experiments

We study the effect of inserting and learning bilateral inception modules in various existing CNN architectures. As a testbed, we perform experiments on semantic segmentation using the Pascal VOC12 segmentation benchmark dataset [77], Cityscapes street scene dataset [57] and on material segmentation using the Materials in Context (MINC) dataset from [25]. We take different CNN architectures from the works of [52, 279, 25] and in-

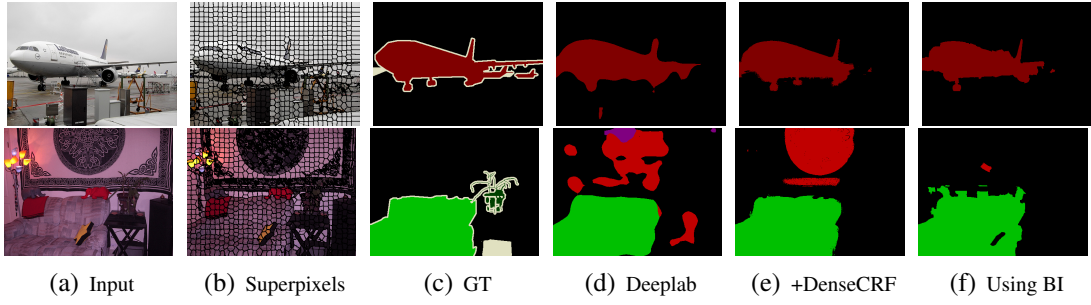| (a) Input | (b) Superpixels | (c) GT | (d) Deeplab | (e) +DenseCRF | (f) Using BI |

Figure 7.5: **Semantic segmentation.** Example results of semantic segmentation on Pascal VOC12 dataset. (d) depicts the DeepLab CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($BI_6(2)+BI_7(6)$) between FC layers.

sert the inception modules before and/or after the spatial FC layers. In Appendix A.5, we presented some quantitative results with approximate bilateral filtering using the permutohedral lattice [7].

### 7.4.1 Semantic Segmentation

We first use the Pascal VOC12 segmentation dataset [77] with 21 object classes. For all experiments on VOC12, we train using the extended training set of 10581 images collected by [108]. Following [279], we use a reduced validation set of 346 images for validation. We experiment on two different network architectures, (a) The DeepLab model from [52] which uses a CNN followed by DenseCRF and (b) The CRFasRNN model from [279] which uses a CNN with deconvolution layers followed by DenseCRF trained end-to-end.

#### DeepLab

We use the publicly available state-of-the-art pre-trained CNN models from [52]. We use the DeepLab-LargeFOV variant as a base architecture and refer to it as 'DeepLab'. The DeepLab CNN model produces a lower resolution prediction ($\frac{1}{8}\times$) which is then bilinearly interpolated to the input image resolution. The original models have been fine-tuned using both the MSCOCO [173] and the extended VOC [108] datasets. Next, we describe modifications to these models and show performance improvements in terms of both IoU and runtimes.

We add inception modules after different FC layers in the original model and remove the DenseCRF post processing. For this dataset, we use 1000 SLIC superpixels [6, 210]. The inception modules after $FC_6$, $FC_7$ and $FC_8$ layers are referred to as $BI_6(H)$, $BI_7(H)$ and $BI_8(H)$ respectively, where $H$ is the number of kernels. All results

| Model | Training | *IoU* | *Runtime*(ms) |
|---|---|---|---|
| DeepLab [52] | | 68.9 | 145 |
| With BI modules | | | |
| $BI_6(2)$ | only BI | 70.8 | +20 |
| $BI_6(2)$ | BI+FC | 71.5 | +20 |
| $BI_6(6)$ | BI+FC | 72.9 | +45 |
| $BI_7(6)$ | BI+FC | 73.1 | +50 |
| $BI_8(10)$ | BI+FC | 72.0 | +30 |
| $BI_6(2)$-$BI_7(6)$ | BI+FC | 73.6 | +35 |
| $BI_7(6)$-$BI_8(10)$ | BI+FC | 73.4 | +55 |
| $BI_6(2)$-$BI_7(6)$ | FULL | **74.1** | +35 |
| $BI_6(2)$-$BI_7(6)$-CRF | FULL | **75.1** | +865 |
| DeepLab-CRF [52] | | 72.7 | +830 |
| DeepLab-MSc-CRF [52] | | **73.6** | +880 |
| DeepLab-EdgeNet [51] | | 71.7 | +30 |
| DeepLab-EdgeNet-CRF [51] | | **73.6** | +860 |

Table 7.1: **Semantic segmentation using DeepLab model.** IoU scores on Pascal VOC12 segmentation test dataset and average runtimes (ms) corresponding to different models. Also shown are the results corresponding to competitive dense pixel prediction techniques that used the same base DeepLab CNN. Runtimes also include superpixel computation (6ms). In the second column, 'BI', 'FC' and 'FULL' correspond to training 'BI', 'FC' and full model layers respectively.

using the DeepLab model on Pascal VOC12 dataset are summarized in Tab. 7.1. We report the 'test' numbers without validation numbers, because the released DeepLab model that we adapted was trained using both train and validation sets. The DeepLab network achieves an IoU of 68.9 after bilinear interpolation. Experiments with the $BI_6(2)$ module indicate that even only learning the inception module while keeping the remaining network fixed results in a reliable IoU improvement ($+1.9$). Additional joint training with FC layers significantly improved the performance. The results also show that more kernels improve performance. Next, we add multiple modules to the base DeepLab network at various stages and train them jointly. This results in further improvement of the performance. The $BI_6(2)$-$BI_7(6)$ model with two inception modules shows significant improvement in IoU by 4.7 and 0.9 in comparison to the baseline model and Dense-CRF application respectively. Finally, fine-tuning the entire network (FULL in Tab. 7.1) boosts the performance by 5.2 and 1.4 compared to the baseline and DenseCRF application.

Some visual results are shown in Fig. A.14 and more are included in Appendix A.5.2. Several other variants of using BI are conceivable. During our experiments, we have observed that more kernels and more modules improve the performance, so we expect that

| Model | IoU | Runtime(ms) |
|---|---|---|
| DeconvNet(CNN+Deconv.) | 72.0 | 190 |
| With BI modules | | |
| $BI_3(2)$-$BI_4(2)$-$BI_6(2)$-$BI_7(2)$ | **74.9** | 245 |
| CRFasRNN (DeconvNet-CRF) | 74.7 | 2700 |

Table 7.2: **Semantic segmentation using CRFasRNN model.** IoU scores and runtimes corresponding to different models on Pascal VOC12 test dataset. Note that runtime also includes superpixel computation.

even better results can be achieved. In Tab. 7.1, the runtime in milliseconds is included for several models. These numbers have been obtained using a Nvidia Tesla K80 GPU and standard Caffe time benchmarking [136]. DenseCRF timings are taken from [51]. The runtimes indicate that the overhead with BI modules is quite minimal in comparison to using Dense CRF.

In addition, we include the results of some other dense pixel prediction methods that are build on top of the same DeepLab base model. DeepLab-MSc-CRF is a multi-scale version [52] of DeepLab with DenseCRF on top. DeepLab-EdgeNet [51] is a recently proposed fast and discriminatively trained domain transform technique for propagating information across pixels. Comparison with these techniques in terms of performance and runtime indicates that our approach performs on par with latest dense pixel prediction techniques with significantly less time overhead. Several state-of-the-art CNN based systems [170, 175] have achieved higher results than DeepLab on Pascal VOC12. These models are not yet publicly available and so we could not test the use of BI models in them. A close variant [21] of our work, which propose to do optimization in the bilateral space also has fast runtimes, but reported lower performance in comparison to the application of DenseCRF.

### CRFasRNN

As a second architecture, we modified the CNN architecture trained by [279] that produces a result at an even lower resolution ($\frac{1}{16}\times$). Multiple deconvolution steps are employed to obtain the segmentation at input image resolution. This result is then passed onto the DenseCRF recurrent neural network to obtain the final segmentation result. We insert BI modules after score-pool3, score-pool4, $FC_6$ and $FC_7$ layers, please see [176, 279] for the network architecture details. Instead of combining outputs from the above layers with deconvolution steps, we introduce BI modules after them and linearly combined the outputs to obtain a final segmentation result. Note that we entirely removed both the deconvolution and the DenseCRF parts of the original model [279]. See Tab. 7.2 for results on the DeconvNet model. Without the DenseCRF part and only

Figure 7.6: **Hierarchical clustering analysis.** From left to right: Validation performance when using different super-pixel layouts, visualization of an image with ground truth segmentation, and the $BI_6(2)$-$BI_7(6)$ result with 200, 600, and 1000 superpixels.

evaluating the deconvolutional part of this model, one obtains an IoU score of 72.0. Ten steps of mean field inference increase the IoU to 74.7 [279]. Our model, with few additional parameters compared to the base CNN, achieves a IoU performance of 74.9, showing an improvement of 0.2 over the CRFasRNN model. The BI layers lead to better performance than deconvolution and DenseCRF combined while being much faster.

### Hierarchical Clustering Analysis

We learned the network parameters using 1000 gSLIC superpixels per image, however the inception module allows to change the resolution (a non-square $K$). To illustrate this, we perform agglomerative clustering of the superpixels, sequentially merging the nearest (in spatial and photometric features) two superpixels into a single one. We then evaluated the DeepLab-$BI_6(2)$-$BI_7(6)$ network using different levels of the resulting hierarchy reusing all the trained network parameters. Results in Fig. 7.6 show that the IoU score on the validation set decreases slowly with decreasing number of points and then drops for less than 200 superpixels. This validates that the network generalizes to different superpixel layouts and it is sufficient to represent larger regions of similar color by fewer points. In future, we plan to explore different strategies to allocate the representation to those regions that require more resolution and to remove the superpixelization altogether. Fig. 7.6 shows example image with 200, 600, and 1000 superpixels and their obtained segmentation with BI modules.

## 7.4.2 Material Segmentation

We also experiment on a different pixel prediction task of material segmentation by adapting a CNN architecture fine-tuned for Materials in Context (MINC) [25] dataset. MINC consists of 23 material classes and is available in three different resolutions with the same aspect ratio: low ($550^2$), mid ($1100^2$) and an original higher resolution. The authors of [25] train CNNs on the mid resolution images and then combine with a Dense-CRF to predict and evaluate on low resolution images. We build our work based on the

| Model | Class / Total accuracy | Runtime(ms) |
|---|:---:|:---:|
| AlexNet CNN | 55.3 / 58.9 | 300 |
| $BI_7(2)$-$BI_8(6)$ | 67.7 / 71.3 | 410 |
| $BI_7(6)$-$BI_8(6)$ | **69.4 / 72.8** | 470 |
| AlexNet-CRF | 65.5 / 71.0 | 3400 |

Table 7.3: **Material segmentation using AlexNet.** Pixel accuracies and runtimes (in ms) of different models on the MINC material segmentation dataset [25]. Runtimes also include the time for superpixel extraction (15ms).



(a) Input  (b) Superpixels  (c) GT  (d) AlexNet  (e) +DenseCRF  (f) Using BI
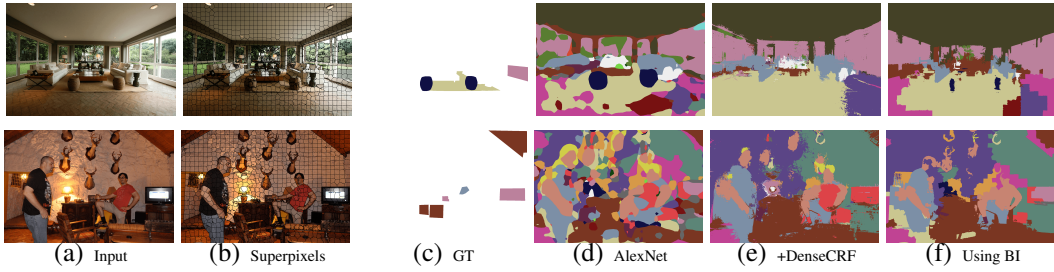
Figure 7.7: **Material segmentation.** Example results of material segmentation. (d) depicts the AlexNet CNN result, (e) CNN + 10 steps of mean-field inference, (f) results obtained with bilateral inception (BI) modules ($BI_7(2)$+$BI_8(6)$) between FC layers.

AlexNet model [152] released by the authors of [25]. To obtain a per pixel labeling of a given image, there are several processing steps that [25] use for good performance. First, a CNN is applied at several scales with different strides followed by an interpolation of the predictions to reach the input image resolution and is then followed by a DenseCRF. For simplicity, we choose to run the CNN network with single scale. The authors used just one kernel with $(u, v, L, a, b)$ features in the DenseCRF part. We used the same features in our inception modules. We modified the base AlexNet model by inserting BI modules after $FC_7$ and $FC_8$ layers. Again, 1000 SLIC superpixels are used for all experiments. Results on the test set are shown in Table 7.3. When inserting BI modules, the performance improves both in total pixel accuracy as well as in class-averaged accuracy. We observe an improvement of 12% compared to CNN predictions and $2 - 4\%$ compared to CNN+DenseCRF results. Qualitative examples are shown in Fig. 7.7 and more are included in Appendix A.5.2. The weights to combine outputs in the BI layers are found by validation on the validation set. For this model we do not provide any learned setup due to very limited segment training data.

| Model | *IoU (Half-res.)* | *IoU (Full-res.)* | *Runtime*(s) |
|---|---|---|---|
| DeepLab CNN | 62.2 | 65.7 | 0.3 |
| $BI_6(2)$ | 62.7 | 66.5 | 5.7 |
| $BI_6(2)$-$BI_7(6)$ | **63.1** | **66.9** | 6.1 |
| DeepLab-CRF | 63.0 | 66.6 | 6.9 |

Table 7.4: **Street scene Segmentation using DeepLab model.** IoU scores and runtimes (in sec) of different models on Cityscapes segmentation dataset [57], for both half-resolution and full-resolution images. Runtime computations also include superpixel computation time (5.2s).



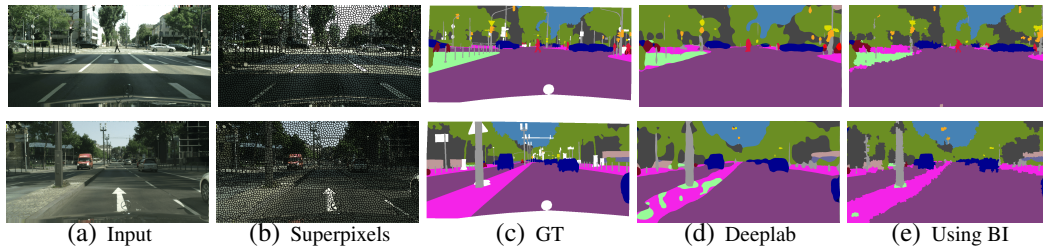| (a) Input | (b) Superpixels | (c) GT | (d) Deeplab | (e) Using BI |

Figure 7.8: **Street scene segmentation.** Example results of street scene segmentation. (d) depicts the DeepLab results, (e) result obtained by adding bilateral inception (BI) modules ($BI_6(2)$+$BI_7(6)$) between FC layers.

## 7.4.3 Street Scene Segmentation

We further evaluate the use of BI modules on the Cityscapes dataset [57]. Cityscapes contains 20K high-resolution ($1024 \times 2048$) images of street scenes with coarse pixel annotations and another 5K images with fine annotations, all annotations are from 19 semantic classes. The 5K images are divided into 2975 train, 500 validation and remaining test images. Since there are no publicly available pre-trained models for this dataset yet, we trained a DeepLab model. We trained the base DeepLab model with half resolution images ($512 \times 1024$) so that the model fits into GPU memory. The result is then interpolated to full-resolution using bilinear interpolation.

We experimented with two layouts: only a single $BI_6(2)$ and one with two inception $BI_6(2)$-$BI_7(6)$ modules. We notice that the SLIC superpixels [6] give higher quantization error than on VOC and thus used 6000 superpixels using [66] for our experiments. Quantitative results on the validation set are shown in Tab. 7.4. In contrast to the findings on the previous datasets, we only observe modest improvements with both DenseCRF and our inception modules in comparison to the base model. Similar to the previous experiments, the inception modules achieve better performance than DenseCRF while being faster. The majority of the computation time in our approach is due to the extraction of

superpixels (5.2$s$) using a CPU implementation. Some visual results with $BI_6(2)$-$BI_7(6)$ model are shown in Fig. 7.8 with more in Appendix A.5.2.

## 7.5 Discussion and Conclusions

The DenseCRF [151] with mean field inference has been used in many CNN segmentation approaches. Its main ingredient and reason for the improved performance is the use of a bilateral filter applied to the beliefs over labels. We have introduced a CNN approach that uses this key component in a novel way: filtering intermediate representations of higher levels in CNNs while jointly learning the task-specific feature spaces. This propagates information between earlier and more detailed intermediate representations of the classes instead of beliefs over labels. Further we show that image adaptive layouts in the higher levels of CNNs can be used to an advantage in the same spirit as CRF graphs have been constructed using superpixels in previous works on semantic segmentation. The computations in the $1 \times 1$ convolution layers scales in the number of superpixels which may be an advantage. Further we have shown that the same representation can be used to interpolate the coarser representations to the full image.

The use of image-adaptive convolutions in between the FC layers retains the appealing effect of producing segmentation masks with sharp edges. This is not a property of the superpixels, using them to represent information in FC layers and their use to interpolate to the full resolution are orthogonal. Different interpolation steps can be used to propagate the label information to the entire image, including bilinear interpolation, bilateral upsampling, up-convolutions and DenseCRFs. We plan to investigate the effect of different sampling strategies to represent information in the higher layers of CNNs and apply similar image-adaptive ideas to videos.

We believe that the Bilateral Inception models are an interesting step that aims to directly include the model structure of CRF factors into the forward architecture of CNNs. The BI modules are easy to implement and are applicable to CNNs that perform structured output prediction.

# Chapter 8

# Conclusions and Outlook

Generative models provide a strong set of tools for modeling the vision world around us. But their use in computer vision is hampered by the complexity of inference in them causing the vision community to favor data-hungry discriminative models. Both generative and discriminative models have complementary advantages and disadvantages as discussed in Chapter 2. Generative models provide an easy handle for incorporating prior knowledge about the task but inference is often too complex in them. Discriminative models, on the other hand, have a straightforward inference scheme as forward evaluation of models, but lack principled ways of incorporating prior knowledge into them.

This thesis work proposed techniques for alleviating some of the key issues with prominent computer vision models by improving inference in them. A common strategy that is followed across several techniques proposed in this thesis is leveraging the complementary models for better inference in a given model. That is, we leverage discriminative models for better inference in generative computer vision models. And we used generative knowledge (in the form of bilateral filters) and enriched the existing discriminative CNN models. This way, this thesis made important steps in bridging the gap between generative and discriminative vision models. The proposed inference techniques are flexible enough to deal with different task scenarios (e.g., availability of large or small amounts of data).

**Inference in Generative Vision Models**    In the case of generative models, we leverage discriminative clustering or random forests techniques to accelerate and/or to improve the Bayesian inference. In Chapter 3, we proposed a new sampling technique called 'Informed Sampler', where discriminative models help in better exploration of target domain, via *informed* proposals, while doing MCMC sampling. In Chapter 4, we proposed a new message passing technique called 'Consensus Message Passing' where random forest predictors are used for predicting *consensus* messages during standard message passing inference resulting in convergence to better solutions.

In both 'Informed Sampler' and 'Consensus Message Passing' (CMP), we made sure that the theoretical guarantees that come with the well established inference techniques are not violated with our modified inference schemes. In the informed sampler, we

achieve this by injecting discriminative knowledge in MCMC sampling via proposal distributions and adhering to detailed balance condition while sampling. And in consensus message passing, we used consensus messages from discriminative predictors only during the first few iterations ensuring that the fixed point reached by our modified inference is also a fixed point of standard message passing in the model. We evaluated both the informed sampler and CMP techniques on three different generative models each, reflecting a wide range of problem scenarios, where we consistently observed improved inference with the proposed techniques in comparison to standard sampling and message passing inference techniques.

**Inference in Discriminative Vision Models**    In this thesis, we focus on the inference in prominent CNN models. Spatial convolutions form the basic building block of most CNN architectures. A key observation in this thesis work is that the bilateral filters [15, 251] are a generalization of spatial convolutions and do not have many of the limitations that spatial convolutions have. The key issue with the existing use of bilateral filters is they are confined to fixed hand-tuned parameterization.

In Chapter 5, we proposed a generalized bilateral filter and devised a gradient based technique for learning the filter parameters. Experiments on wide range of problems showed the superior performance of learnable bilateral filters with respect to using Gaussian filter kernel. Learnable bilateral filters enabled us to stack several filters together and learn all of them via back-propagation. Using this, we proposed novel neural network architectures which we call 'Bilateral Neural Networks' (BNN).

In Chapter 6, we showed how BNNs can be easily adapted to filter video data for propagating temporal information across video frames. In Chapter 7, we proposed new and fast neural network modules, based on explicit Gaussian bilateral filtering called 'Bilateral Inceptions' and showcased how we can modify existing segmentation CNN architectures for big improvements in accuracy while adding little time overhead.

Bilateral filters form the core of mean-field inference in DenseCRF models [151] and provide a way to incorporate prior knowledge about the scene in the form of dense feature-based connectivity across the pixels. By integrating learnable bilateral filters into standard CNN architectures, we brought the worlds of CRF and CNN closer, providing a way to incorporate prior knowledge into CNNs.

## 8.1 Summary of Contributions

The following list summarizes the specific contributions of this thesis work:

- We devised a novel MCMC sampling approach called 'Informed Sampler' (Chapter 3) for doing Bayesian inference in complex generative vision models. The Informed sampler leverages discriminative approaches for improving the efficiency of MCMC sampling. Experiments on a wide range of generative vision models

showed significantly faster convergence of our sampler while maintaining higher acceptance rates. This opens up possibilities for using complex generative models like graphics engines for addressing vision problems.

- We devised a novel message passing technique called 'Consensus Message Passing (CMP), in Chapter 4, for doing Bayesian inference in layered graphical models used in vision. Experiments on diverse graphical models in vision showed that CMP resulted in significantly better performance compared to standard message passing techniques such as expectation propagation or variational message passing. Moreover, CMP is the first instance where the Infer.NET [186] probabilistic programming language is shown to be useful for addressing vision problems.

- We parameterized bilateral filters as general sparse high dimensional filters (Chapter 5) and devised an approach for learning the filter kernels via standard back-propagation learning techniques. This resulted in a general technique for learning sparse high-dimensional filters, which in turn resulted in a generalization of standard bilateral filters. Experiments on wide range of applications showed improved performance with respect to standard Gaussian bilateral filters.

- We also show how learning bilateral filters can generalize the fully-connected conditional random field models (DenseCRF) to arbitrarily learned pairwise potentials (Section 5.5) instead of standard Gaussian pairwise edge potentials. DenseCRF is one of the widely used CRF techniques in vision and this generalization carries forward to most of its existing applications and helps in better integration into end-to-end trained models like convolutional neural networks (CNN).

- Our technique for learning general sparse high dimensional filters also generalizes standard spatial convolutions in CNN frameworks. This opens up possibilities for applying CNNs to sparse high-dimensional data, which is not feasible with many standard CNN techniques. Moreover, our technique can be used for learning image-adaptive filters inside CNNs instead of standard image-agnostic 2D filters.

- We adapted the learnable sparse high dimensional filters for video filtering, in Chapter 6, and proposed a novel neural network approach for propagating content across video frames. We call our networks 'Video Propagation Networks' (VPN). Experiments on video object segmentation and semantic video segmentation showed that VPN outperformed existing task-specific methods while being faster.

- In Chapter 7, we devised a new CNN module called 'Bilateral Inception' that can be readily inserted into standard segmentation CNN models resulting in better performance while producing the result at original image resolution and also alleviating some of the need for post-processing. Experiments on state-of-the-art CNN models resulted in significantly better performance with our inception module while being competitive in time.

## 8.2  Outlook

With the diverse range of experiments on each proposed technique, I hope to have convinced the readers that this thesis work resulted in several important advances for inference in computer vision models. Inference in computer vision models is still a very active area of research and I hope the work presented in this thesis aids in further advances in this area of research. The following are some of the research topics that could benefit from the concepts and techniques presented in this thesis.

**Leveraging Photo-Realistic Graphics for Vision:**  As discussed in Section 2.2.1, modern graphics engines leverage dedicated hardware setups and provide real-time renderings with stunning level of realism. This is made possible with accurate modeling of image formation. Such realistic graphics models are seldom utilized in building vision systems due to the difficulty in posterior inference. The informed sampler (Chapter 3) and consensus message passing (Chapter 4) techniques presented in this thesis made important steps in making the inference faster. But the proposed techniques are still not fast enough for practical purposes. An important research direction is to further improve the efficiency of inference by making use of the internals of state-of-the-art rendering mechanisms such as Metropolis Light Transport [260]. One way to achieve this is by improving MCMC sampling efficiency via pre-rejection using the rejection technique of [149], without the need for doing full rendering for the rejected samples in MCMC. This would result in a better coupling of graphics and Bayesian vision systems.

**CNNs for Sparse 3D Data:**  In recent years, there is an increasing need for techniques that efficiently process 3D data, with the onset of cheaper 3D scanners and the consumer devices that requires processing 3D data (e.g. virtual reality devices such as Oculus Rift [5]). One of the distinguishing characters of high-dimensional data such as 3D point clouds or videos is that they are sparse. 3D point clouds are inherently sparse and the sparsity of video data comes from the redundant representation across frames. Learnable bilateral filters proposed in this thesis (Chapter 5), provide a principled way to process sparse high-dimensional data by enabling long-range data dependent connections. This thesis work also demonstrated that one could easily integrate these sparse high-dimensional filters into other CNN architectures and are also shown to be fruitful for video processing (Chapter 6). I hope this thesis work inspires future research work for efficiently processing 3D data such as point clouds or meshes. Since the convolutions are performed in bilateral space with sparsely populated cells, there is no need to voxelize a given point cloud or meshes for doing 3D convolutions. One of the limitations of bilateral neural networks is that the bilateral feature scales are hand-tuned. There are other recent works such as [150, 154] trying to optimize feature spaces for bilateral filtering but are not integrated into CNN frameworks. An interesting future research direction is to bring these works together by learning feature spaces for bilateral neural networks in an end-to-end fashion.

**Structured CNNs:**  Although images are represented with independent pixel values, real world is highly structured and prior knowledge therein can be captured by rich modeling tools like graphical models. For example, in the case of urban scene understanding, one could leverage the prior knowledge that objects like cars and persons stand on the ground plane; building facades are vertical etc. Most of the existing CNN frameworks are agnostic to such explicit prior information. One may argue that the CNNs are capable of implicitly learning such prior knowledge directly from the training data. But, in practice, the amount of labelled training data is always limited and thus CNNs need external prior constraints to be able to perform well on several real world problems especially when the training data is very limited. In this thesis, we developed techniques for incorporating prior knowledge into CNNs via learnable bilateral filters. Such prior knowledge is low-level and, ideally I would like to have techniques for integrating high-level prior knowledge (e.g., the prior knowledge that is encoded in graphical models). Other existing works in this direction (e.g. [129, 279, 252, 48]) are also either limited in the type of prior knowledge they model and/or graphical model constraints are generally enforced after the main CNN structure. I believe that a fruitful direction to tackle this problem is developing structured filters that can be easily integrated into CNN frameworks. The work presented in this thesis make important steps in this direction.

**On Combining Generative and Discriminative Models:**  Hybrid generative and discriminative models are an active area of research and I believe that the future of computer vision would be dominated by such hybrid models which are scalable and generic to be applicable to a wide range of problem scenarios. In Section 2.4, we discussed several recent works that aim to develop such hybrid models. Several inference techniques presented in this thesis are based on the synergistic combinations of generative and discriminative models. I hope that this thesis work inspires or aids in the further development of hybrid generative and discriminative vision models.

# Appendix A

# Supplementary Material

In this appendix, we present supplementary material for the techniques and experiments presented in the main text.

## A.1  Baseline Results and Analysis for Informed Sampler

Here, we give an in-depth performance analysis of the various samplers and the effect of their hyperparameters. We choose hyperparameters with the lowest PSRF value after $10k$ iterations, for each sampler individually. If the differences between PSRF are not significantly different among multiple values, we choose the one that has the highest acceptance rate.

### A.1.1  Experiment: Estimating Camera Extrinsics

**Parameter Selection**

**Metropolis Hastings (MH)**  Figure A.1a shows the median acceptance rates and PSRF values corresponding to various proposal standard deviations of plain MH sampling. Mixing gets better and the acceptance rate gets worse as the standard deviation increases. The value 0.3 is selected standard deviation for this sampler.

**Metropolis Hastings Within Gibbs (MHWG)**  As mentioned in Section 3.5.1, the MHWG sampler with one-dimensional updates did not converge for any value of proposal standard deviation. This problem has high correlation of the camera parameters and is of multi-modal nature, which this sampler has problems with.

**Parallel Tempering (PT)**  For PT sampling, we took the best performing MH sampler and used different temperature chains to improve the mixing of the sampler. Figure A.1b shows the results corresponding to different combination of temperature levels. The sampler with temperature levels of $[1, 3, 27]$ performed best in terms of both mixing and acceptance rate.

**Effect of Mixture Coefficient in Informed Sampling (INF-MH)**   Figure A.1c shows the effect of mixture coefficient ($\alpha$) on the informed sampling INF-MH. Since there is no significant different in PSRF values for $0 \leq \alpha \leq 0.7$, we chose 0.7 due to its high acceptance rate.
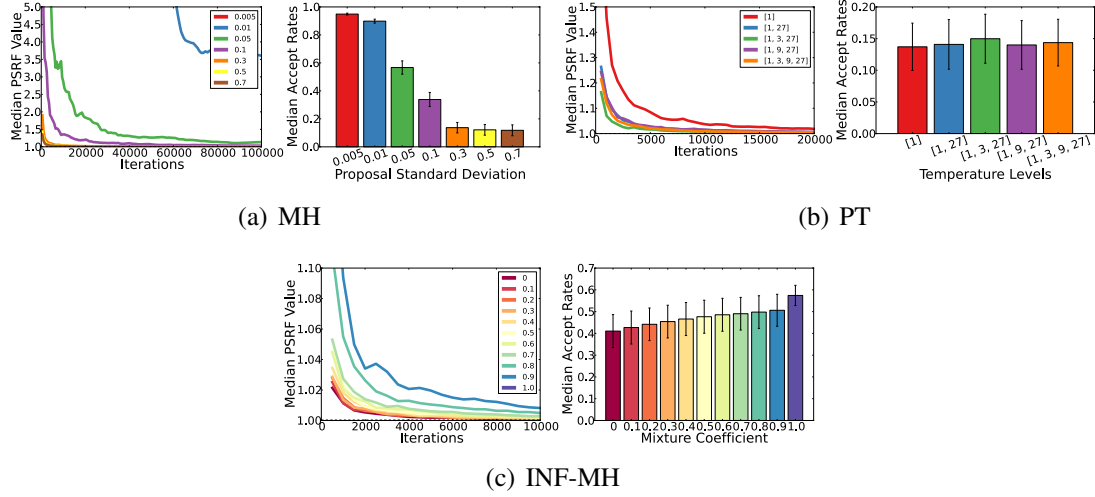


(a) MH

(b) PT



(c) INF-MH

Figure A.1: **Results of the 'Estimating Camera Extrinsics' experiment.** PRSFs and Acceptance rates corresponding to (a) various standard deviations of MH, (b) various temperature level combinations of PT sampling and (c) various mixture coefficients of INF-MH sampling.

## A.1.2 Experiment: Occluding Tiles

**Parameter Selection**

**Metropolis Hastings (MH)**   Figure A.2a shows the results of MH sampling. Results show the poor convergence for all proposal standard deviations and rapid decrease of AR with increasing standard deviation. This is due to the high-dimensional nature of the problem. We selected a standard deviation of 1.1.

**Blocked Metropolis Hastings Within Gibbs (BMHWG)**   The results of BMHWG are shown in Figure A.2b. In this sampler we update only one block of tile variables (of dimension four) in each sampling step. Results show much better performance compared to plain MH. The optimal proposal standard deviation for this sampler is 0.7.

**Metropolis Hastings Within Gibbs (MHWG)**   Figure A.2c shows the result of MHWG sampling. This sampler is better than BMHWG and converges much more quickly. Here a standard deviation of 0.9 is found to be best.

(a) MH

(b) BMHWG

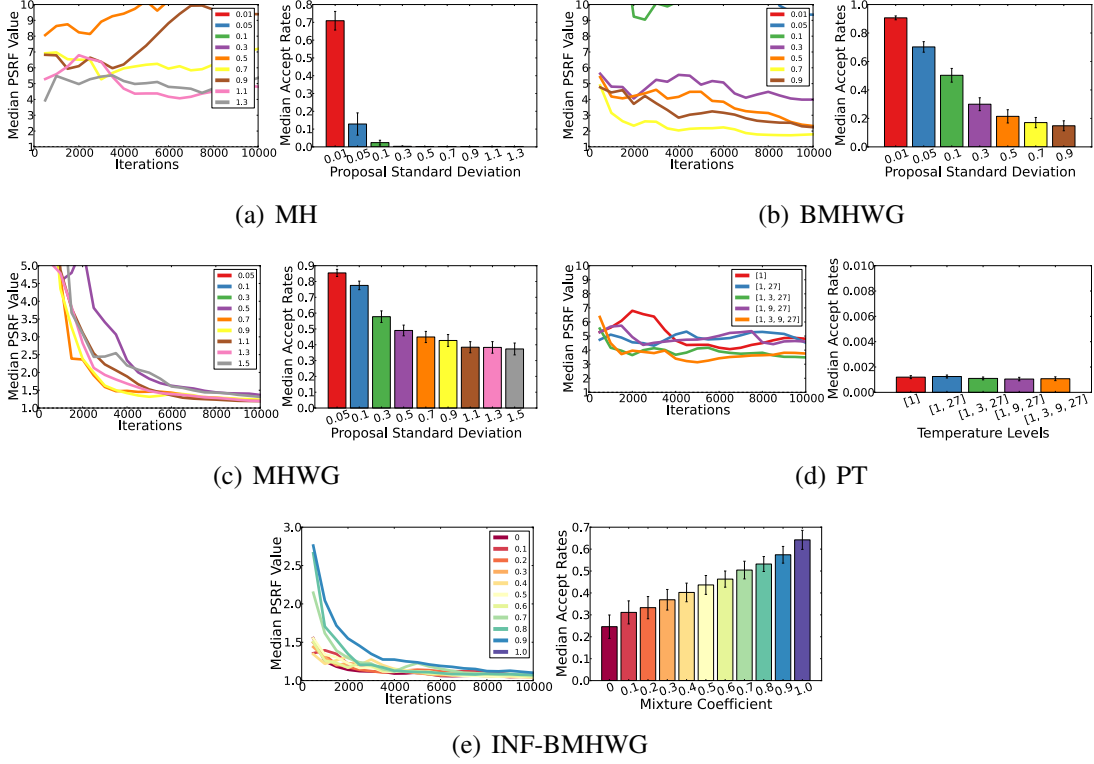(c) MHWG

(d) PT

(e) INF-BMHWG

Figure A.2: **Results of the 'Occluding Tiles' experiment.** PRSF and Acceptance rates corresponding to various standard deviations of (a) MH, (b) BMHWG, (c) MHWG, (d) various temperature level combinations of PT sampling and; (e) various mixture coefficients of our informed INF-BMHWG sampling.

**Parallel Tempering (PT)**   Figure A.2d shows the results of PT sampling with various temperature combinations. Results show no improvement in AR from plain MH sampling and again $[1, 3, 27]$ temperature levels are found to be optimal.

**Effect of Mixture Coefficient in Informed Sampling (INF-BMHWG)**   Figure A.2e shows the effect of mixture coefficient ($\alpha$) on the blocked informed sampling INF-BMHWG. Since there is no significant different in PSRF values for $0 \leq \alpha \leq 0.8$, we chose 0.8 due to its high acceptance rate.

## A.1.3 Experiment: Estimating Body Shape

**Parameter Selection**

**Metropolis Hastings (MH)**   Figure A.3a shows the result of MH sampling with various proposal standard deviations. The value of 0.1 is found to be best.
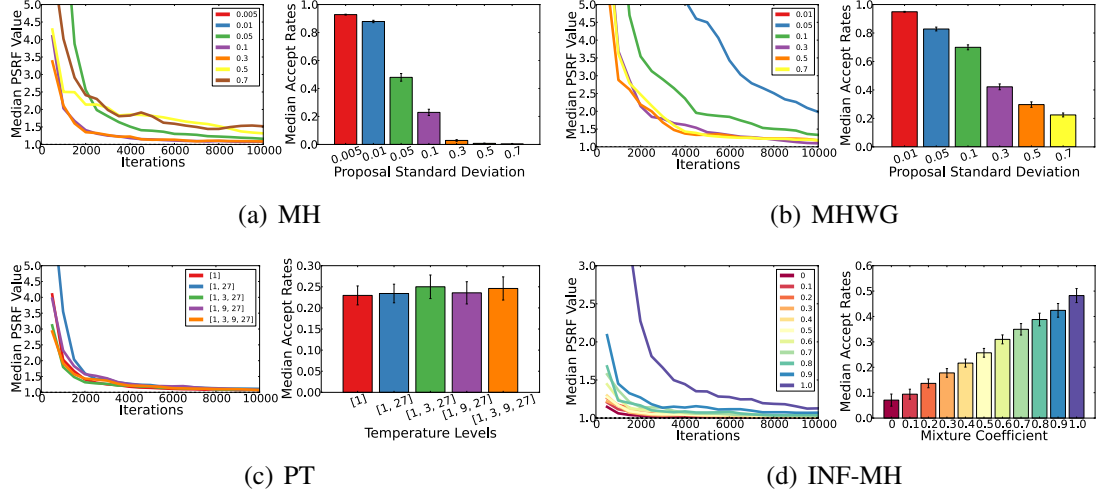
(a) MH

(b) MHWG

(c) PT

(d) INF-MH

Figure A.3: **Results of the 'Body Shape Estimation' experiment.** PRSFs and Acceptance rates corresponding to various standard deviations of (a) MH, (b) MHWG; (c) various temperature level combinations of PT sampling and; (d) various mixture coefficients of the informed INF-MH sampling.

**Metropolis Hastings Within Gibbs (MHWG)**   For MHWG sampling we select 0.3 proposal standard deviation. Results are shown in Fig. A.3b.

**Parallel Tempering (PT)**   As before, results in Fig. A.3c, the temperature levels were selected to be $[1, 3, 27]$ due its slightly higher AR.

**Effect of Mixture Coefficient in Informed Sampling (INF-MH)**   Figure A.3d shows the effect of $\alpha$ on PSRF and AR. Since there is no significant differences in PSRF values for $0 \le \alpha \le 0.8$, we choose 0.8.
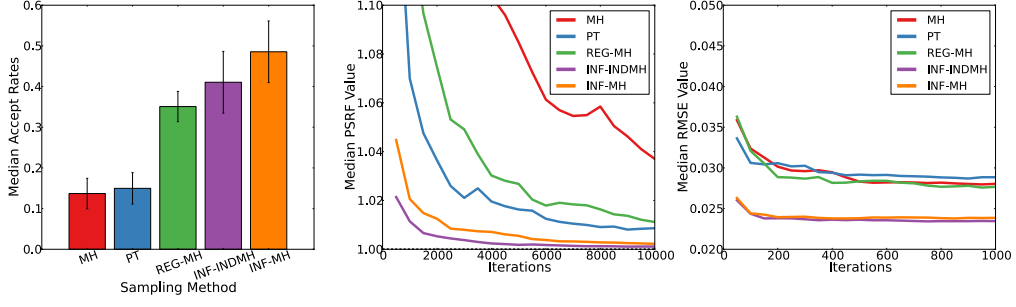
## A.1.4  Results Overview

Figure A.4 shows the summary results of the all the three experimental studies related to informed sampler.
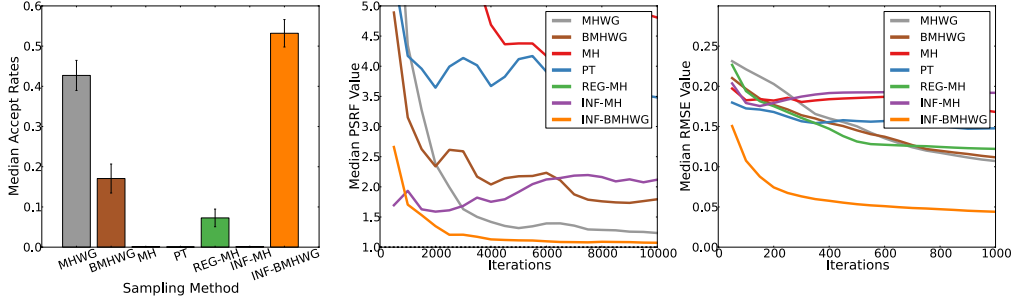
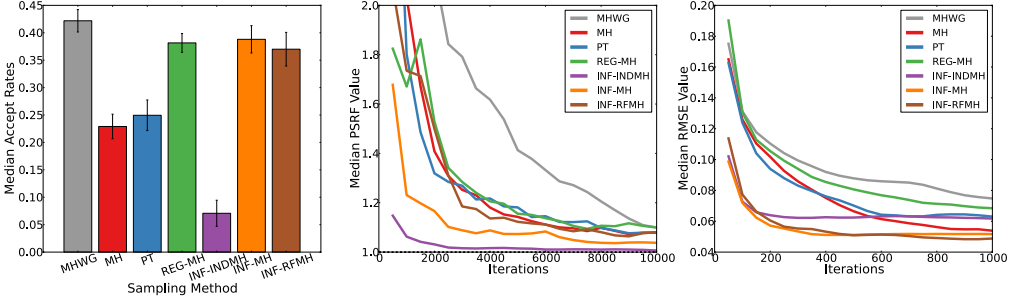## A.1.5  Additional Qualitative Results

### Occluding Tiles

In Figure A.5 more qualitative results of the occluding tiles experiment are shown. The informed sampling approach (INF-BMHWG) is better than the best baseline (MHWG). This still is a very challenging problem since the parameters for occluded tiles are flat

(a) Results for: Estimating Camera Extrinsics



(b) Results for: Occluding Tiles



(c) Results for: Estimating Body Shape

Figure A.4: **Summary of the statistics for the three experiments.** Shown are for several baseline methods and the informed samplers the acceptance rates (left), PSRFs (middle), and RMSE values (right). All results are median results over multiple test examples.

over a large region. Some of the posterior variance of the occluded tiles is already captured by the informed sampler.

**Body Shape**

Figure A.6 shows some more results of 3D mesh reconstruction using posterior samples obtained by our informed sampling INF-MH.

Figure A.5: **Additional qualitative results of the occluding tiles experiment.** From left to right: (a) Given image, (b) Ground truth tiles, (c) OpenCV heuristic and most probable estimates from 5000 samples obtained by (d) MHWG sampler (best baseline) and (e) our INF-BMHWG sampler. (f) Posterior expectation of the tiles boundaries obtained by INF-BMHWG sampling (First 2000 samples are discarded as burn-in).

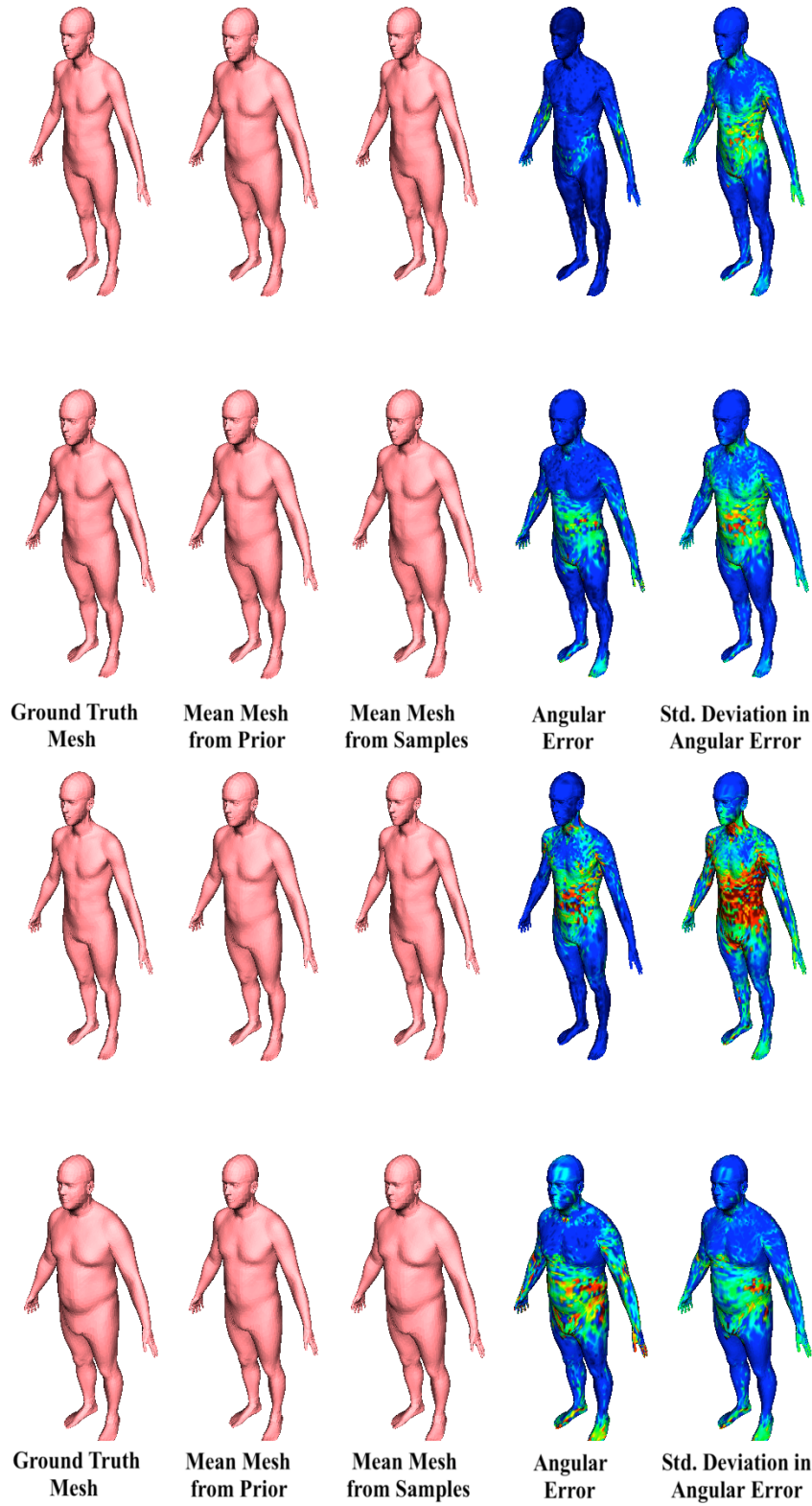| Ground Truth Mesh | Mean Mesh from Prior | Mean Mesh from Samples | Angular Error | Std. Deviation in Angular Error |

Figure A.6: **Qualitative results for the body shape experiment.** Shown is the 3D mesh reconstruction results with first 1000 samples obtained using the INF-MH informed sampling method. (blue indicates small values and red indicates high values)

# A.2 Additional Results on the Face Problem with CMP

Figure A.7 shows inference results for reflectance maps, normal maps and lights for randomly chosen test images, and Fig. A.8 shows reflectance estimation results on multiple images of the same subject produced under different illumination conditions. CMP is able to produce estimates that are closer to the groundtruth across different subjects and illumination conditions.



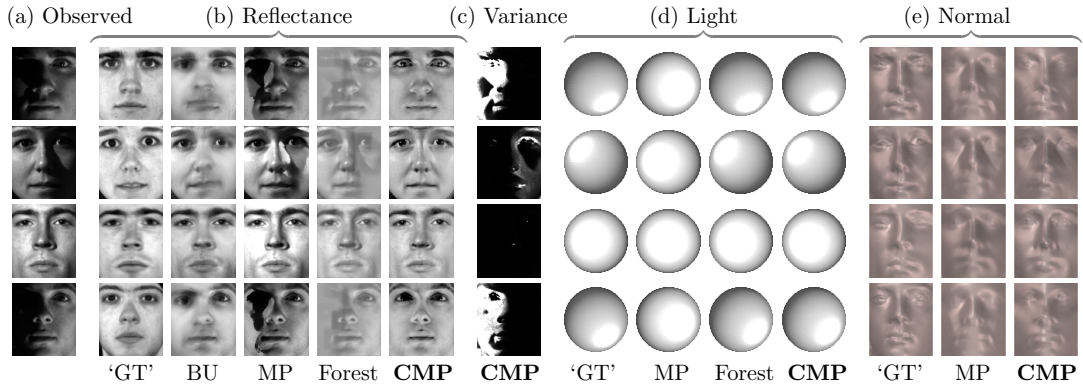| (a) Observed | (b) Reflectance | | | | (c) Variance | (d) Light | | | | (e) Normal | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | 'GT' | BU | MP | Forest | **CMP** | **CMP** | 'GT' | MP | Forest | **CMP** | 'GT' | MP | **CMP** |

Figure A.7: **A visual comparison of inference results.** (a) Observed images. (b) Inferred reflectance maps. *GT* is the photometric stereo groundtruth, *BU* is the Biswas *et al.* (2009) reflectance estimate and *Forest* is the consensus prediction. (c) The variance of the inferred reflectance estimate produced by CMP (normalized across rows).(d) Visualization of inferred light directions. (e) Inferred normal maps.



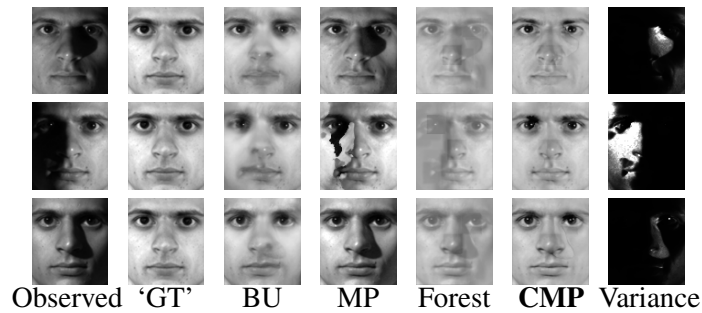Observed  'GT'  BU  MP  Forest  **CMP**  Variance

Figure A.8: **Robustness to varying illumination.** Reflectance estimation on a subject images with varying illumination. Left to right: observed image, photometric stereo estimate (GT) which is used as a proxy for groundtruth, bottom-up estimate of [27], VMP result, consensus forest estimate, CMP mean, and CMP variance.

# A.3 Additional Material for Learning Sparse High Dimensional Filters

This part of supplementary material contains a more detailed overview of the permuto-hedral lattice convolution in Section A.3.1, more experiments in Section A.3.2 and additional results with protocols for the experiments presented in Chapter 5 in Section A.3.3.

## A.3.1 General Permutohedral Convolutions

A core technical contribution of this work is the generalization of the Gaussian permuto-hedral lattice convolution proposed in [7] to the full non-separable case with the ability to perform back-propagation. Although, conceptually, there are minor differences between Gaussian and general parameterized filters, there are non-trivial practical differences in terms of the algorithmic implementation. The Gauss filters belong to the separable class and can thus be decomposed into multiple sequential one dimensional convolutions. We are interested in the general filter convolutions, which can not be decomposed. Thus, performing a general permutohedral convolution at a lattice point requires the computation of the inner product with the neighboring elements in all the directions in the high-dimensional space.

Here, we give more details of the implementation differences of separable and non-separable filters. In the following, we will explain the scalar case first. Recall, that the forward pass of general permutohedral convolution involves 3 steps: *splatting*, *convolving* and *slicing*. We follow the same splatting and slicing strategies as in [7] since these operations do not depend on the filter kernel. The main difference between our work and the existing implementation of [7] is the way that the convolution operation is executed. This proceeds by constructing a *blur neighbor* matrix $K$ that stores for every lattice point all values of the lattice neighbors that are needed to compute the filter output.

The blur neighbor matrix is constructed by traversing through all the populated lattice points and their neighboring elements. This is done recursively to share computations. For any lattice point, the neighbors that are $n$ hops away are the direct neighbors of the points that are $n-1$ hops away. The size of a $d$ dimensional spatial filter with width $s+1$ is $(s+1)^d$ (*e.g.*, a $3 \times 3$ filter, $s = 2$ in $d = 2$ has $3^2 = 9$ elements) and this size grows exponentially in the number of dimensions $d$. The permutohedral lattice is constructed by projecting a regular grid onto the plane spanned by the $d$ dimensional normal vector $(1, \ldots, 1)^\top$. See Fig. A.9 for an illustration of the 1D lattice construction. Many corners of a grid filter are projected onto the same point, in total $t = (s+1)^d - s^d$ elements remain in the permutohedral filter with $s$ neighborhood in $d - 1$ dimensions. If the lattice has $m$ populated elements, the matrix $K$ has size $t \times m$. Note that, since the input signal is typically sparse, only a few lattice corners are being populated in the *slicing* step. We use a hash-table to keep track of these points and traverse only through the populated lattice points for this neighborhood matrix construction.
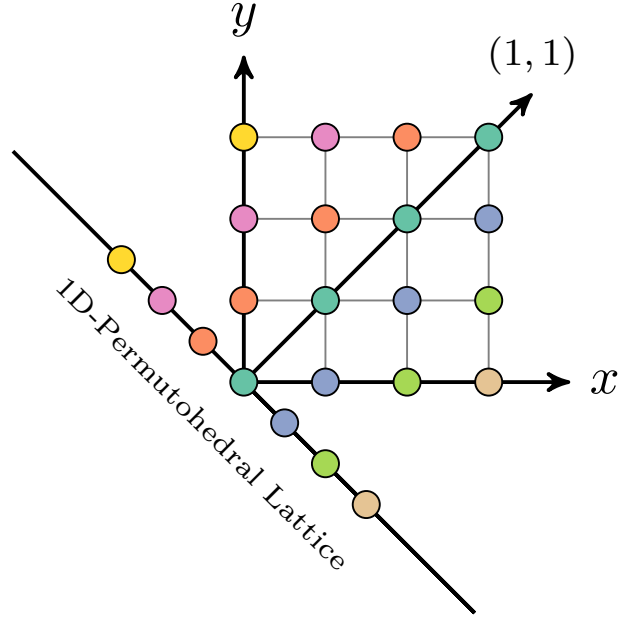
Figure A.9: **Illustration of 1D permutohedral lattice construction.** A $4 \times 4$ $(x,y)$ grid lattice is projected onto the plane defined by the normal vector $(1,1)^\top$. This grid has $s + 1 = 4$ and $d = 2$ $(s+1)^d = 4^2 = 16$ elements. In the projection, all points of the same color are projected onto the same points in the plane. The number of elements of the projected lattice is $t = (s+1)^d - s^d = 4^2 - 3^2 = 7$, that is the $(4 \times 4)$ grid minus the size of lattice that is 1 smaller at each size, in this case a $(3 \times 3)$ lattice (the upper right $(3 \times 3)$ elements).

Once the blur neighbor matrix $K$ is constructed, we can perform the convolution by the matrix vector multiplication

$$\ell' = BK, \tag{A.1}$$

where $B$ is the $1 \times t$ filter kernel (whose values we will learn) and $\ell' \in \mathbb{R}^{1 \times m}$ is the result of the filtering at the $m$ lattice points. In practice, we found that the matrix $K$ is sometimes too large to fit into GPU memory and we divided the matrix $K$ into smaller pieces to compute Eq. (A.1) sequentially.

In the general multi-dimensional case, the signal $\ell$ is of $c$ dimensions. Then the kernel $B$ is of size $c \times t$ and $K$ stores the $c$ dimensional vectors accordingly. When the input and output points are different, we slice only the input points and splat only at the output points.

## A.3.2  Additional Experiments

In this section, we discuss more use-cases for the learned bilateral filters, one use-case of BNNs and two single filter applications for image and 3D mesh denoising.

**Recognition of subsampled MNIST**

One of the strengths of the proposed filter 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 [159]. We sample a sparse set of $N$ points $(x, y) \in [0, 1] \times [0, 1]$ uniformly at random in the input image, use their interpolated 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 the grid positions.

We take a reference implementation of LeNet [158] that is part of the Caffe project [136] and compare it against the same architecture but replacing the first convolutional layer with a bilateral convolution layer (BCL). The filter size and numbers are adjusted to get a comparable number of parameters ($5 \times 5$ for LeNet, 2-neighborhood for BCL).

The results are shown in Table A.1. We see that training on the original MNIST data (column Original, LeNet vs. 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. We introduce an additional bilinear interpolation layer for the LeNet architecture to train on the same data. 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 will be faster due to hashing the sparse input points, as well as less memory demanding in comparison to naive application of a spatial convolution with interpolated values.

**Image Denoising**

The main application that inspired the development of the bilateral filtering operation is image denoising [15], there using a single Gaussian kernel. Our development allows to learn this kernel function from data and we explore how to improve using a *single* but more general bilateral filter.

We use the Berkeley segmentation dataset (BSDS500) [12] 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 a denoising task. The first baseline model ('Spatial' in Table A.2, 25 weights) uses a single spatial filter with

|        |          | Test Subsampling | | |
| Method | Original | 100% | 60% | 20% |
| --- | --- | --- | --- | --- |
| LeNet | **0.9919** | 0.9660 | 0.9348 | **0.6434** |
| BNN | 0.9903 | **0.9844** | **0.9534** | 0.5767 |
| LeNet 100% | 0.9856 | 0.9809 | 0.9678 | **0.7386** |
| BNN 100% | **0.9900** | **0.9863** | **0.9699** | 0.6910 |
| LeNet 60% | 0.9848 | 0.9821 | 0.9740 | 0.8151 |
| BNN 60% | **0.9885** | **0.9864** | **0.9771** | **0.8214** |
| LeNet 20% | 0.9763 | **0.9754** | 0.9695 | 0.8928 |
| BNN 20% | 0.9728 | 0.9735 | **0.9701** | **0.9042** |

Table A.1: Classification accuracy on MNIST. We compare the LeNet [158] implementation that is part of Caffe [136] to the network with the first layer replaced by a bilateral convolution layer (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.

a kernel size of 5 and predicts the scalar gray-scale value at the center pixel. The next model ('Gauss Bilateral') applies a bilateral *Gaussian* filter to the noisy input, using position and intensity features $\mathbf{f} = (x, y, v)^{\top}$. The third setup ('Learned Bilateral', 65 weights) takes a Gauss kernel as initialization and fits all filter weights on the train 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.

| Method | PSNR |
| --- | --- |
| Noisy Input | 20.17 |
| Spatial | 26.27 |
| Gauss Bilateral | 26.51 |
| Learned Bilateral | 26.58 |
| Spatial + Bilateral (Learned) | 26.65 |

Table A.2: PSNR results of a denoising task using the BSDS500 dataset [12]

The PSNR scores evaluated on full images of the test set are shown in Table A.2. We find that an untrained bilateral filter already performs better than a trained spatial convolution (26.27 to 26.51). A learned convolution further improve the performance slightly. We chose 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 needs to be properly adjusted in network size. Multi-layer perceptrons have obtained state-of-the-art denoising results [44] and the permutohedral lattice layer can readily be used in such an architecture, which is intended future work.

## A.3.3 Additional results

This section contains more qualitative results for the experiments presented in Chapter 5.



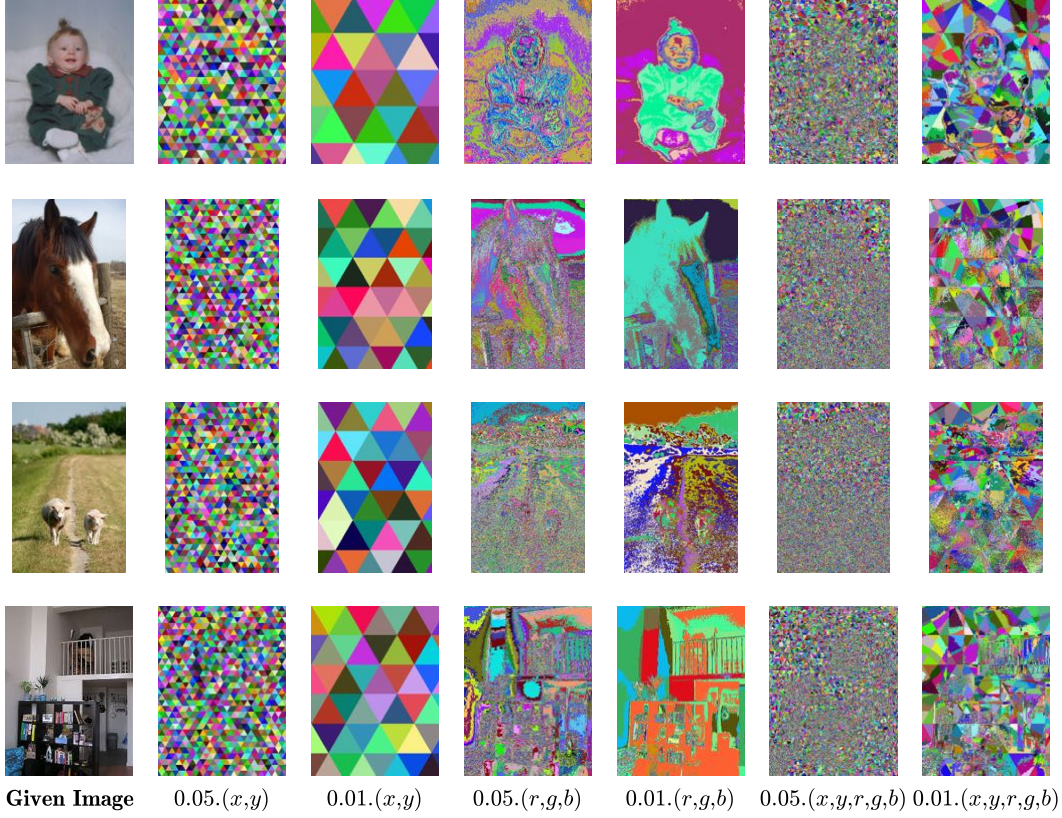| Given Image | 0.05.$(x,y)$ | 0.01.$(x,y)$ | 0.05.$(r,g,b)$ | 0.01.$(r,g,b)$ | 0.05.$(x,y,r,g,b)$ | 0.01.$(x,y,r,g,b)$ |

Figure A.10: **Visualization of the Permutohedral Lattice.** Sample lattice visualizations for different feature spaces. All pixels falling in the same simplex cell are shown with the same color. $(x,y)$ features correspond to image pixel positions, and $(r,g,b) \in [0,255]$ correspond to the red, green and blue color values.

**Lattice Visualization**

Figure A.10 shows sample lattice visualizations for different feature spaces.

**Color Upsampling**

Some images of the upsampling for the Pascal VOC12 dataset are shown in Fig. A.11. It is especially the low level image details that are better preserved with a learned bilateral filter compared to the Gaussian case.

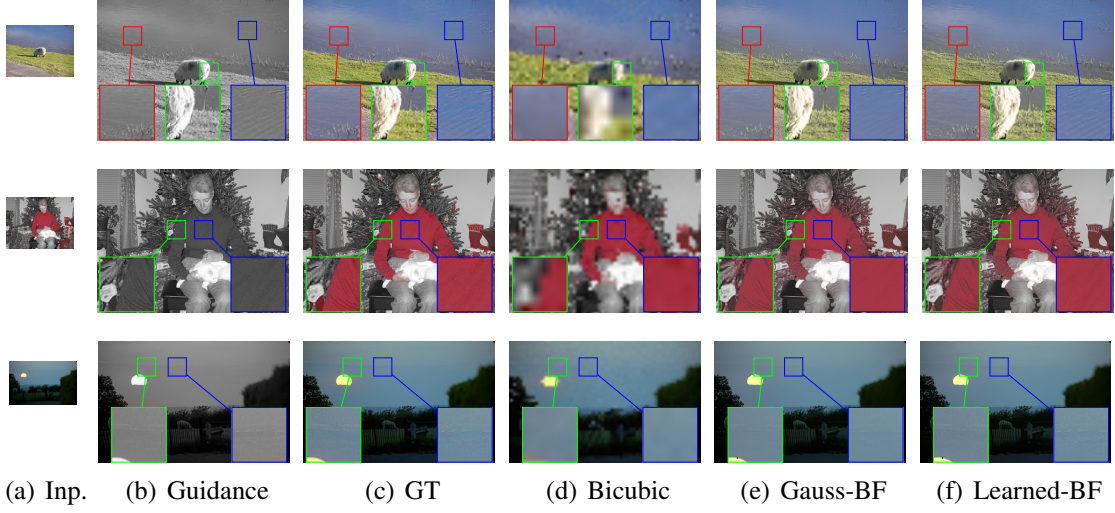| (a) Inp. | (b) Guidance | (c) GT | (d) Bicubic | (e) Gauss-BF | (f) Learned-BF |

Figure A.11: **Color Upsampling.** Color $8\times$ upsampling results using different methods, from left to right, (a) Low-resolution input color image (Inp.), (b) Gray scale guidance image, (c) Ground-truth color image; Upsampled color images with (d) Bicubic interpolation, (e) Gauss bilateral upsampling and, (f) Learned bilateral updampgling (best viewed on screen).

## Depth Upsampling

Figure A.12 presents some more qualitative results comparing bicubic interpolation, Gauss bilateral and learned bilateral upsampling on NYU depth dataset image [231].

## Character Recognition

Figure A.13 shows the schematic of different layers of the network architecture for LeNet-7 [159] and DeepCNet(5, 50) [56, 100]. For the BNN variants, the first layer filters are replaced with learned bilateral filters and are learned end-to-end.

## Semantic Segmentation

Some more visual results for semantic segmentation are shown in Figure A.14. These include the underlying DeepLab CNN[52] result (DeepLab), the 2 step mean-field result with Gaussian edge potentials (+2stepMF-GaussCRF) and also corresponding results with learned edge potentials (+2stepMF-LearnedCRF). In general, we observe that mean-field in learned CRF leads to slightly dilated classification regions in comparison to using Gaussian CRF thereby filling-in the false negative pixels and also correcting some mis-classified regions.
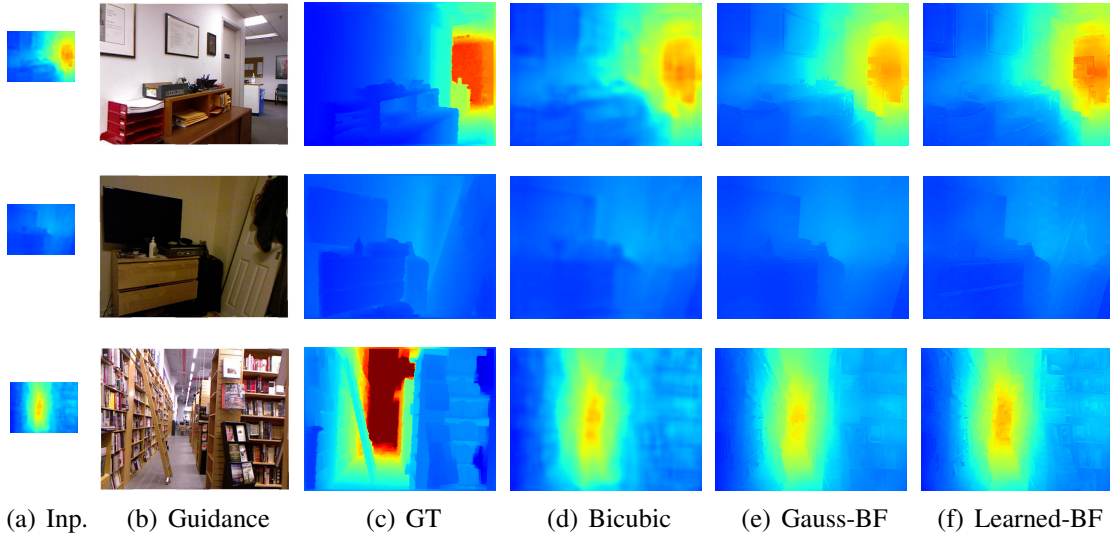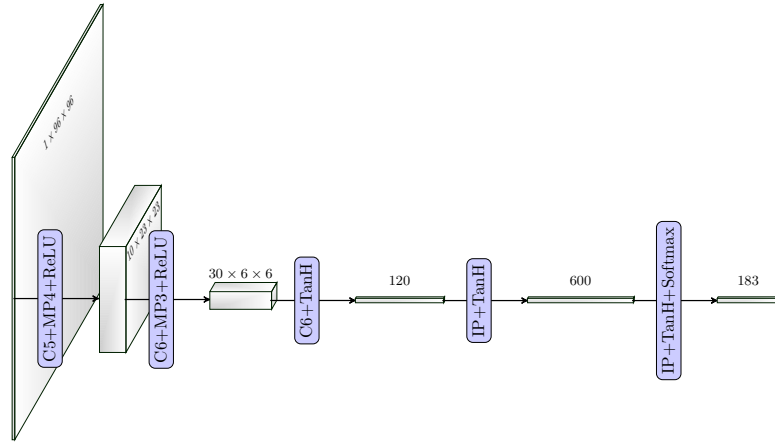
144

|  (a) Inp. | (b) Guidance | (c) GT | (d) Bicubic | (e) Gauss-BF | (f) Learned-BF |

Figure A.12: **Depth Upsampling.** Depth $8\times$ upsampling results using different upsampling strategies, from left to right, (a) Low-resolution input depth image (Inp.), (b) High-resolution guidance image, (c) Ground-truth depth; Upsampled depth images with (d) Bicubic interpolation, (e) Gauss bilateral upsampling and, (f) Learned bilateral updampgling (best viewed on screen).
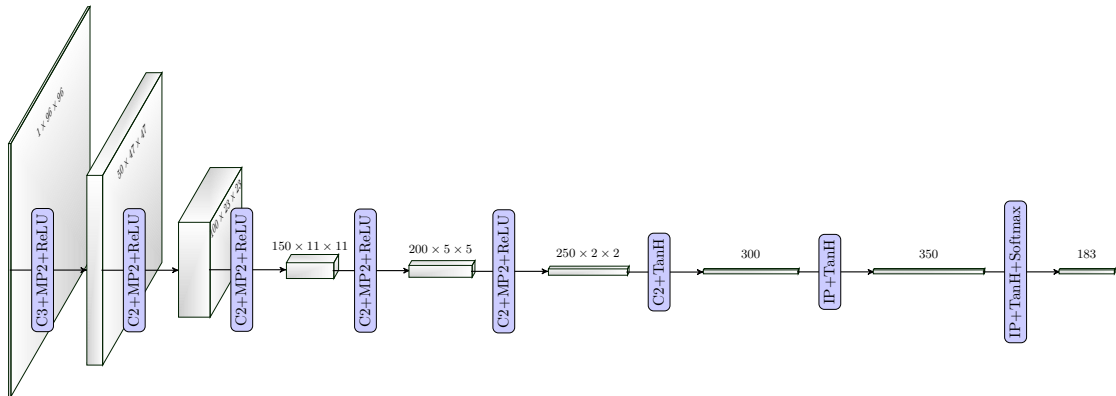
**Material Segmentation**

In Fig. A.15, we present visual results comparing 2 step mean-field inference with Gaussian and learned pairwise CRF potentials. In general, we observe that the pixels belonging to dominant classes in the training data are being more accurately classified with learned CRF. This leads to a significant improvements in overall pixel accuracy. This also results in a slight decrease of the accuracy from less frequent class pixels thereby slightly reducing the average class accuracy with learning. We attribute this to the type of annotation that is available for this dataset, which is not for the entire image but for some segments in the image. We have very few images of the infrequent classes to combat this behaviour during training.

**Experiment Protocols**

Table A.3 shows experiment protocols of different experiments.

(a) LeNet-7



(b) DeepCNet

Figure A.13: **CNNs for Character Recognition.** Schematic of (top) LeNet-7 [159] and (bottom) DeepCNet(5,50) [56, 100] architectures used in Assamese character recognition experiments.

Figure A.14: **Semantic Segmentation.** Example results of semantic segmentation. (c) depicts the unary results before application of MF, (d) after two steps of MF with Gaussian edge CRF potentials, (e) after two steps of MF with learned edge CRF potentials.
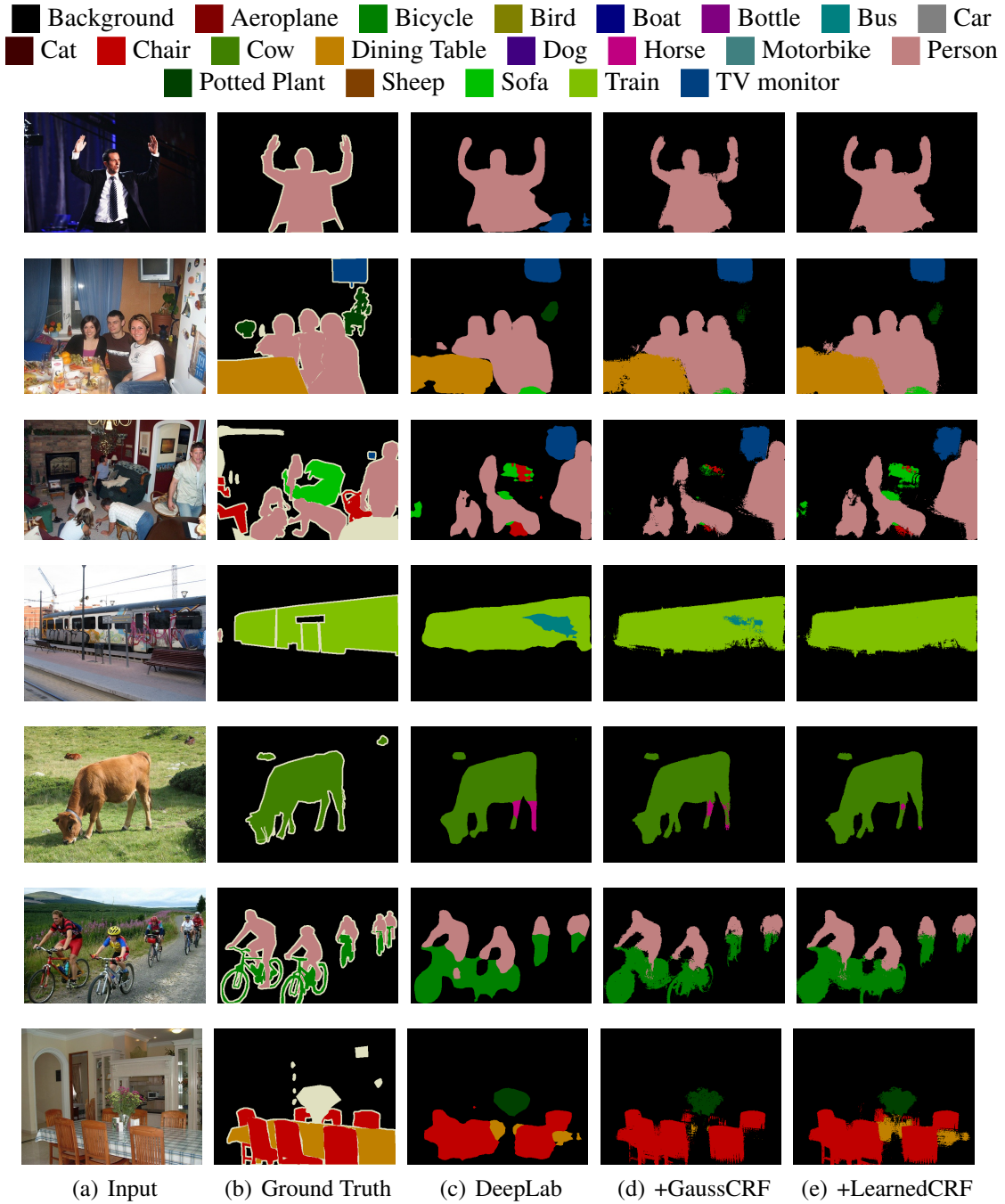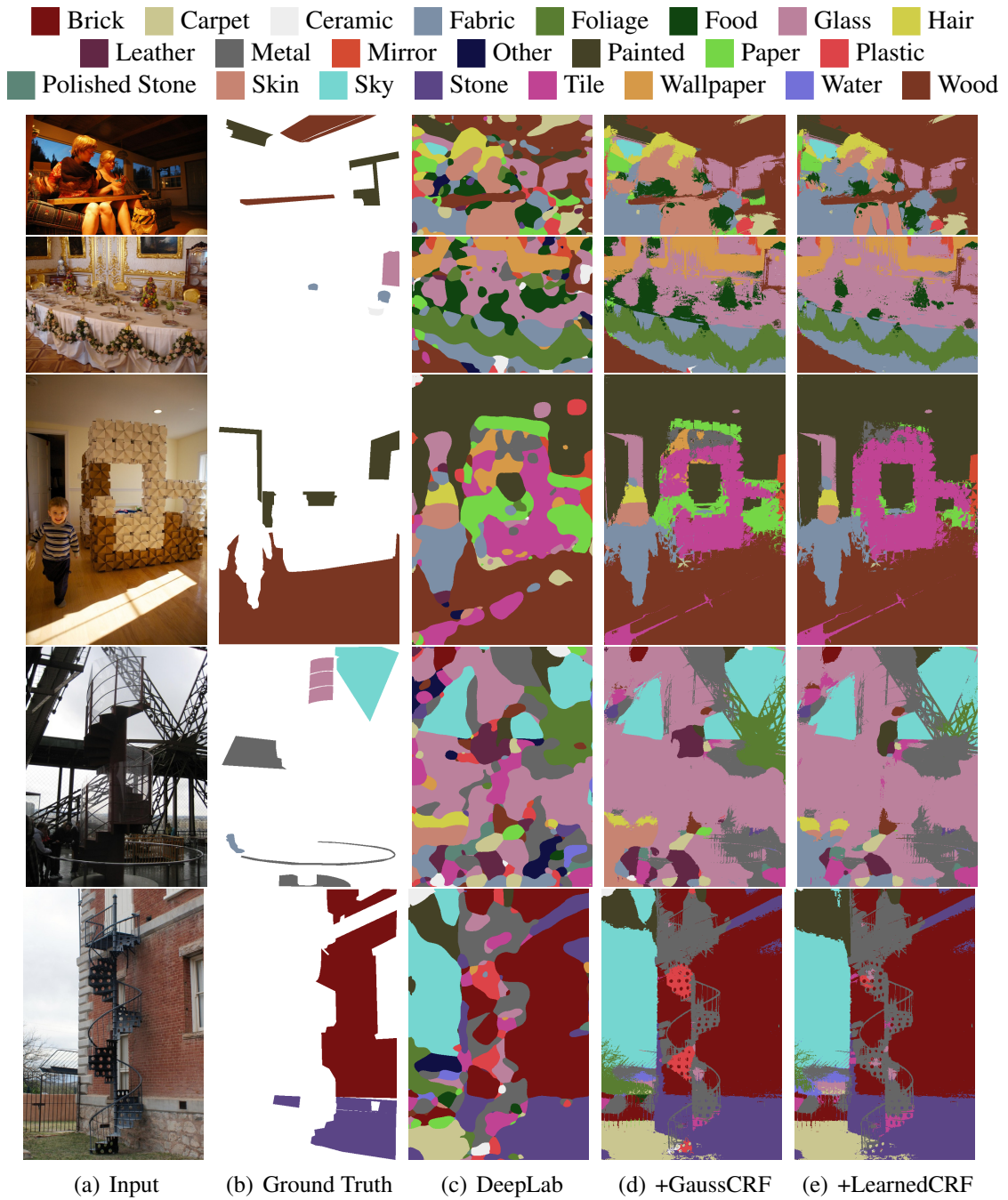
Figure A.15: **Material Segmentation.** Example results of material segmentation. (c) depicts the unary results before application of MF, (d) after two steps of MF with Gaussian edge CRF potentials, (e) after two steps of MF with learned edge CRF potentials.

| Experiment | Feature Types | Feature Scales | Filter Size | Filter Nbr. | Data Statistics | | | Training Protocol | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Train | Val. | Test | Loss Type | LR | Batch | Epochs |
| **Single Bilateral Filter Applications** | | | | | | | | | | | |
| **2× Color Upsampling** | $Position_1$, Intensity (3D) | 0.13, 0.17 | 65 | 2 | 10581 | 1449 | 1456 | MSE | 1e-06 | 200 | 94.5 |
| **4× Color Upsampling** | $Position_1$, Intensity (3D) | 0.06, 0.17 | 65 | 2 | 10581 | 1449 | 1456 | MSE | 1e-06 | 200 | 94.5 |
| **8× Color Upsampling** | $Position_1$, Intensity (3D) | 0.03, 0.17 | 65 | 2 | 10581 | 1449 | 1456 | MSE | 1e-06 | 200 | 94.5 |
| **16× Color Upsampling** | $Position_1$, Intensity (3D) | 0.02, 0.17 | 65 | 2 | 10581 | 1449 | 1456 | MSE | 1e-06 | 200 | 94.5 |
| **Depth Upsampling** | $Position_1$, Color (5D) | 0.05, 0.02 | 665 | 2 | 795 | 100 | 654 | MSE | 1e-07 | 50 | 251.6 |
| **Mesh Denoising** | Isomap (4D) | 46.00 | 63 | 2 | 1000 | 200 | 500 | MSE | 100 | 10 | 100.0 |
| **DenseCRF Applications** | | | | | | | | | | | |
| **Semantic Segmentation** | | | | | | | | | | | |
| | $Position_1$, Color (5D); | | | | | | | | | | |
| **- 1step MF** | $Position_1$ (2D) | 0.01, 0.34; 0.34 | 665; 19 | 2; 2 | 10581 | 1449 | 1456 | Logistic | 0.1 | 5 | 1.4 |
| | $Position_1$, Color (5D); | | | | | | | | | | |
| **- 2step MF** | $Position_1$ (2D) | 0.01, 0.34; 0.34 | 665; 19 | 2; 2 | 10581 | 1449 | 1456 | Logistic | 0.1 | 5 | 1.4 |
| | $Position_1$, Color (5D); | | | | | | | | | | |
| **- *loose* 2step MF** | $Position_1$ (2D) | 0.01, 0.34; 0.34 | 665; 19 | 2; 2 | 10581 | 1449 | 1456 | Logistic | 0.1 | 5 | +1.9 |
| **Material Segmentation** | | | | | | | | | | | |
| **- 1step MF** | $Position_2$, Lab-Color (5D) | 5.00, 0.05, 0.30 | 665 | 2 | 928 | 150 | 1798 | Weighted Logistic | 1e-04 | 24 | 2.6 |
| **- 2step MF** | $Position_2$, Lab-Color (5D) | 5.00, 0.05, 0.30 | 665 | 2 | 928 | 150 | 1798 | Weighted Logistic | 1e-04 | 12 | +0.7 |
| **- *loose* 2step MF** | $Position_2$, Lab-Color (5D) | 5.00, 0.05, 0.30 | 665 | 2 | 928 | 150 | 1798 | Weighted Logistic | 1e-04 | 12 | +0.2 |
| **Neural Network Applications** | | | | | | | | | | | |
| **Tiles: CNN-9×9** | - | - | 81 | 4 | 10000 | 1000 | 1000 | Logistic | 0.01 | 100 | 500.0 |
| **Tiles: CNN-13×13** | - | - | 169 | 6 | 10000 | 1000 | 1000 | Logistic | 0.01 | 100 | 500.0 |
| **Tiles: CNN-17×17** | - | - | 289 | 8 | 10000 | 1000 | 1000 | Logistic | 0.01 | 100 | 500.0 |
| **Tiles: CNN-21×21** | - | - | 441 | 10 | 10000 | 1000 | 1000 | Logistic | 0.01 | 100 | 500.0 |
| **Tiles: BNN** | $Position_1$, Color (5D) | 0.05, 0.04 | 63 | 1 | 10000 | 1000 | 1000 | Logistic | 0.01 | 100 | 30.0 |
| **LeNet** | - | - | 25 | 2 | 5490 | 1098 | 1647 | Logistic | 0.1 | 100 | 182.2 |
| **Crop-LeNet** | - | - | 25 | 2 | 5490 | 1098 | 1647 | Logistic | 0.1 | 100 | 182.2 |
| **BNN-LeNet** | $Position_2$ (2D) | 20.00 | 7 | 1 | 5490 | 1098 | 1647 | Logistic | 0.1 | 100 | 182.2 |
| **DeepCNet** | - | - | 9 | 1 | 5490 | 1098 | 1647 | Logistic | 0.1 | 100 | 182.2 |
| **Crop-DeepCNet** | - | - | 9 | 1 | 5490 | 1098 | 1647 | Logistic | 0.1 | 100 | 182.2 |
| **BNN-DeepCNet** | $Position_2$ (2D) | 40.00 | 7 | 1 | 5490 | 1098 | 1647 | Logistic | 0.1 | 100 | 182.2 |

Table A.3: **Experiment Protocols.** Experiment protocols for the different experiments presented in this work. **Feature Types**: Feature spaces used for the bilateral convolutions. $Position_1$ corresponds to un-normalized pixel positions whereas $Position_2$ corresponds to pixel positions normalized to $[0, 1]$ with respect to the given image. **Feature Scales**: Cross-validated scales for the features used. **Filter Size**: Number of elements in the filter that is being learned. **Filter Nbr.**: Half-width of the filter. **Train**, **Val.** and **Test** corresponds to the number of train, validation and test images used in the experiment. **Loss Type**: Type of loss used for back-propagation. "MSE" corresponds to Euclidean mean squared error loss and "Logistic" corresponds to multinomial logistic loss. "Weighted Logistic" is the class-weighted multinomial logistic loss. We weighted the loss with inverse class probability for material segmentation task due to the small availability of training data with class imbalance. **LR**: Fixed learning rate used in stochastic gradient descent. **Batch**: Number of images used in one parameter update step. **Epochs**: Number of training epochs. In all the experiments, we used fixed momentum of 0.9 and weight decay of 0.0005 for stochastic gradient descent. "'Color Upsampling" experiments in this Table corresponds to those performed on Pascal VOC12 dataset images. For all experiments using Pascal VOC12 images, we use extended training segmentation dataset available from [108], and used standard validation and test splits from the main dataset [77].

# A.4  Parameters and Additional Results for Video Propagation Networks

In this Section, we present experiment protocols and additional qualitative results for experiments on video object segmentation, semantic video segmentation and video color propagation. Table A.4 shows the feature scales and other parameters used in different experiments. Figures A.16 show some qualitative results on video object segmentation with some failure cases in Fig. A.17. Figure A.18 shows some qualitative results on semantic video segmentation and Fig. A.19 shows results on video color propagation.

| Experiment | Feature Type | Feature Scale-1, $\Lambda_a$ | Feature Scale-2, $\Lambda_b$ | $\alpha$ | Input Frames | Loss Type |
|---|---|---|---|---|---|---|
| **Video Object Segmentation** | $(x,y,Y,Cb,Cr,t)$ | (0.02,0.02,0.07,0.4,0.4,0.01) | (0.03,0.03,0.09,0.5,0.5,0.2) | 0.5 | 9 | Logistic |
| **Semantic Video Segmentation** | | | | | | |
| **with CNN1 [273]-NoFlow** | $(x,y,R,G,B,t)$ | (0.08,0.08,0.2,0.2,0.2,0.04) | (0.11,0.11,0.2,0.2,0.2,0.04) | 0.5 | 3 | Logistic |
| **with CNN1 [273]-Flow** | $(x+u_x,y+u_y,R,G,B,t)$ | (0.11,0.11,0.14,0.14,0.14,0.03) | (0.08,0.08,0.12,0.12,0.12,0.01) | 0.65 | 3 | Logistic |
| **with CNN2 [214]-Flow** | $(x+u_x,y+u_y,R,G,B,t)$ | (0.08,0.08,0.2,0.2,0.2,0.04) | (0.09,0.09,0.25,0.25,0.25,0.03) | 0.5 | 4 | Logistic |
| **Video Color Propagation** | $(x,y,I,t)$ | (0.04,0.04,0.2,0.04) | No second kernel | 1 | 4 | MSE |

Table A.4: **Experiment Protocols.** Experiment protocols for the different experiments presented in this work. **Feature Types**: Feature spaces used for the bilateral convolutions, with position $(x,y)$ and color $(R,G,B$ or $Y,Cb,Cr)$ features $\in [0,255]$. $u_x$, $u_y$ denotes optical flow with respect to the present frame and $I$ denotes grayscale intensity. **Feature Scales** $(\Lambda_a, \Lambda_b)$: Cross-validated scales for the features used. $\alpha$: Exponential time decay for the input frames. **Input Frames**: Number of input frames for VPN. **Loss Type**: Type of loss used for back-propagation. "MSE" corresponds to Euclidean mean squared error loss and "Logistic" corresponds to multinomial logistic loss.
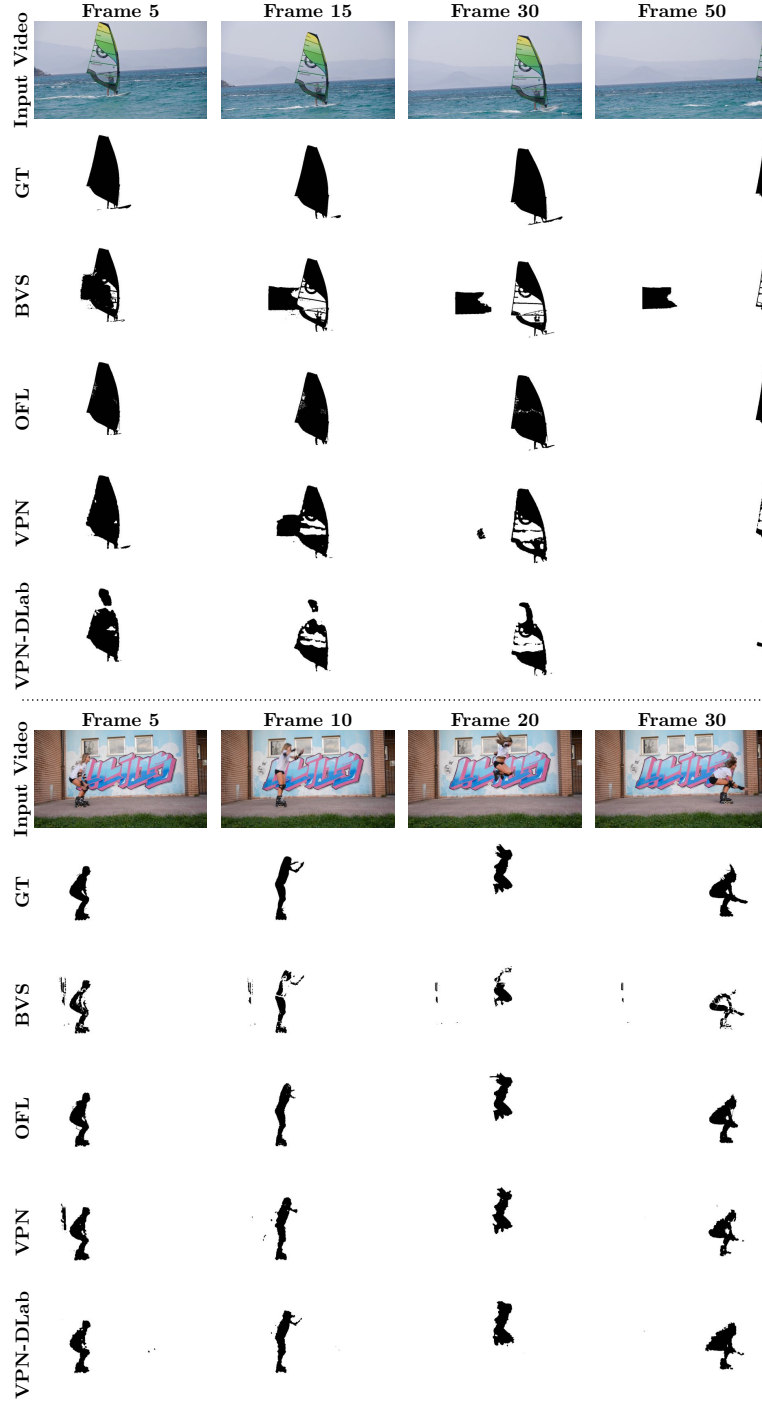
Figure A.16: **Video Object Segmentation.** Shown are the different frames in example videos with the corresponding ground truth (GT) masks, predictions from BVS [180], OFL [255], VPN (VPN-Stage2) and VPN-DLab (VPN-DeepLab) models.
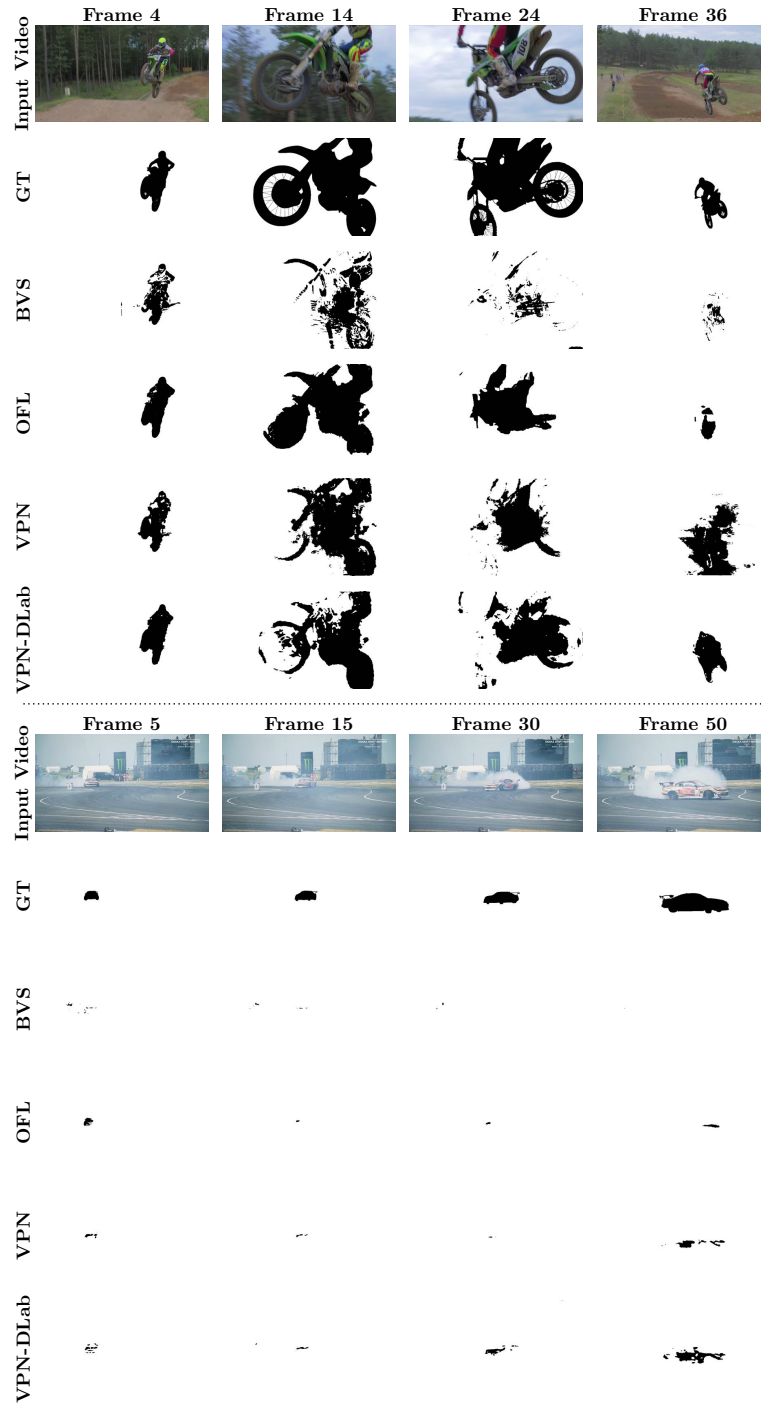
Figure A.17: **Failure Cases for Video Object Segmentation.**  Shown are the different frames in example videos with the corresponding ground truth (GT) masks, predictions from BVS [180], OFL [255], VPN (VPN-Stage2) and VPN-DLab (VPN-DeepLab) models.
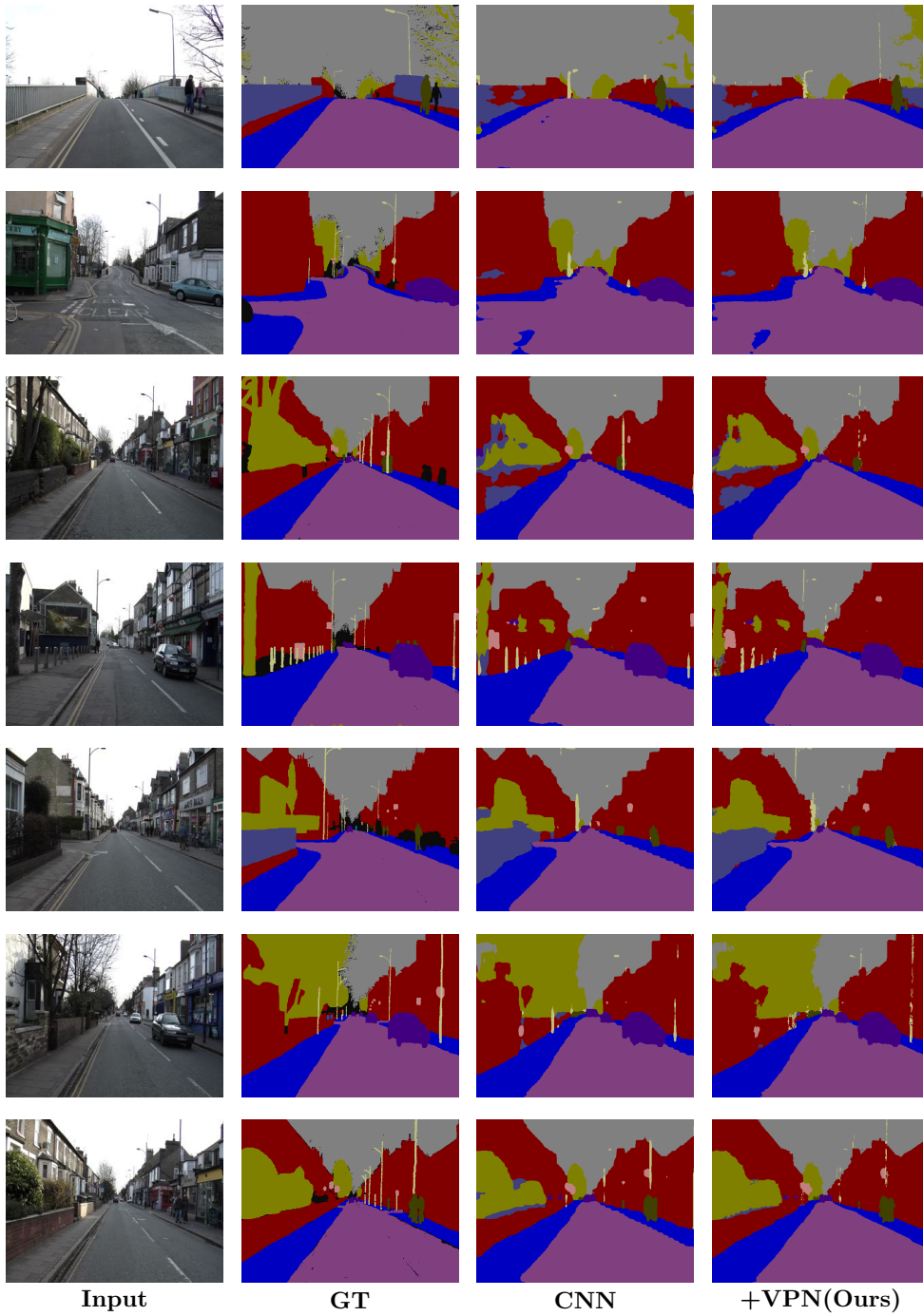
| Input | GT | CNN | +VPN(Ours) |

Figure A.18: **Semantic Video Segmentation.** Input video frames and the corresponding ground truth (GT) segmentation together with the predictions of CNN [273] and with VPN-Flow.
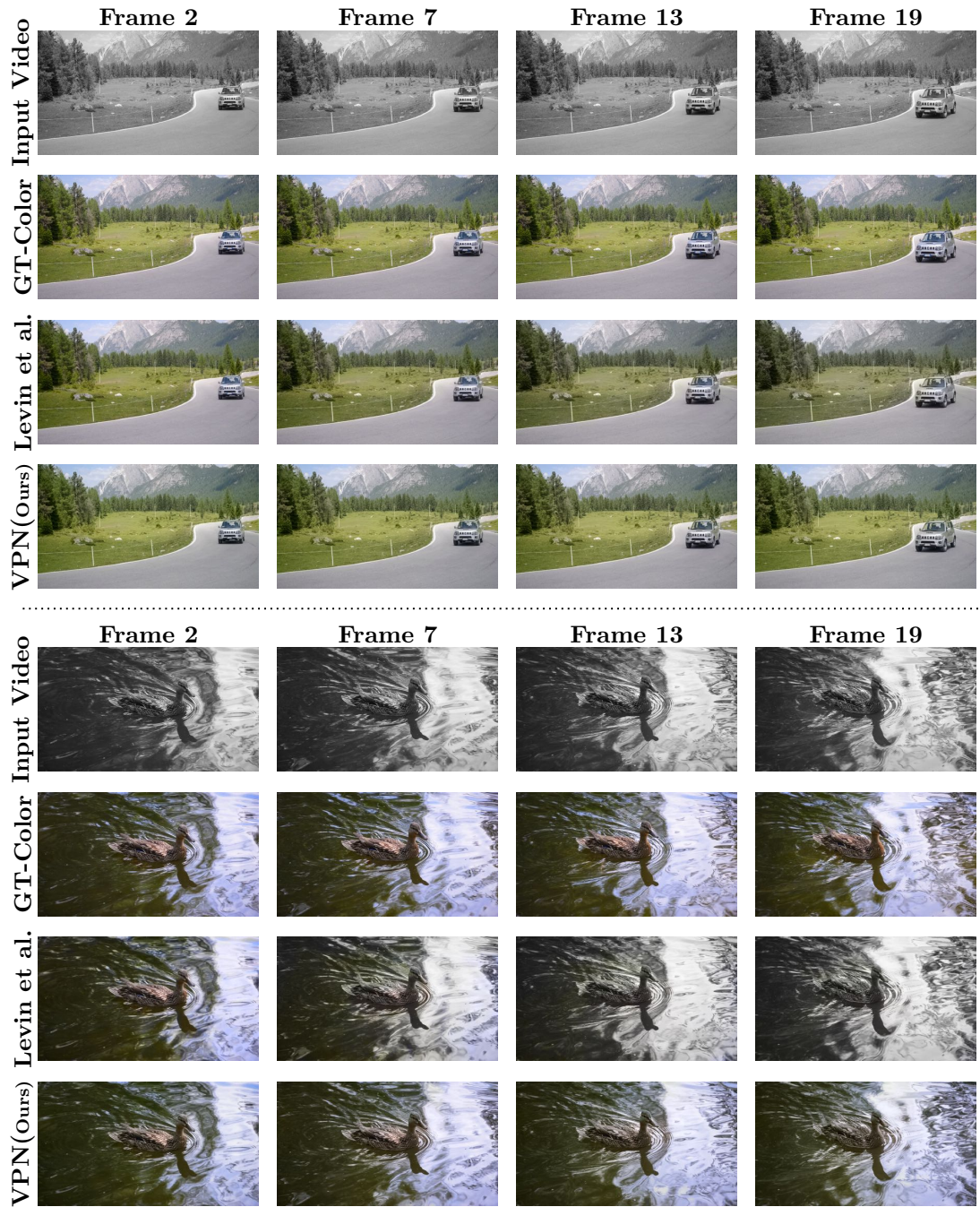
Figure A.19: **Video Color Propagation.** Input grayscale video frames and corresponding ground-truth (GT) color images together with color predictions of Levin et al. [164] and VPN-Stage1 models.

# A.5 Additional Material for Bilateral Inception Networks

In this section of the Appendix, we first discuss the use of approximate bilateral filtering in BI modules (Sec. A.5.1). Later, we present some qualitative results using different models for the approach presented in Chapter 7 (Sec. A.5.2).

## A.5.1 Approximate Bilateral Filtering

The bilateral inception module presented in Chapter 7 computes a matrix-vector product between a Gaussian filter $K$ and a vector of activations $\mathbf{z}_c$. Bilateral filtering is an important operation and many algorithmic techniques have been proposed to speed-up this operation [199, 7, 91]. In the main paper we opted to implement what can be considered the brute-force variant of explicitly constructing $K$ and then using BLAS to compute the matrix-vector product. This resulted in a few millisecond operation. The explicit way to compute is possible due to the reduction to super-pixels, e.g., it would not work for DenseCRF variants that operate on the full image resolution.

Here, we present experiments where we use the fast approximate bilateral filtering algorithm of [7], which is also used in Chapter 5 for learning sparse high dimensional filters. This choice allows for larger dimensions of matrix-vector multiplication. The reason for choosing the explicit multiplication in Chapter 7 was that it was computationally faster. For the small sizes of the involved matrices and vectors, the explicit computation is sufficient and we had no GPU implementation of an approximate technique that matched this runtime. Also it is conceptually easier and the gradient to the feature transformations ($\Lambda\mathbf{f}$) is obtained using standard matrix calculus.

### Experiments

We modified the existing segmentation architectures analogous to those in Chapter 7. The main difference is that, here, the inception modules use the lattice approximation [7] to compute the bilateral filtering. Using the lattice approximation did not allow us to back-propagate through feature transformations ($\Lambda$) and thus we used hand-specified feature scales as will be explained later. Specifically, we take CNN architectures from the works of [52, 279, 25] and insert the BI modules between the spatial FC layers. We use superpixels from [66] for all the experiments with the lattice approximation. Experiments are performed using Caffe neural network framework [136].

**Semantic Segmentation**   The experiments in this section use the Pascal VOC12 segmentation dataset [77] with 21 object classes and the images have a maximum resolution of 0.25 megapixels. For all experiments on VOC12, we train using the extended training set of 10581 images collected by [108]. We modified the DeepLab network architecture

| Model | *IoU* | *Runtime*(ms) |
|---|---|---|
| DeepLab | 68.9 | 145ms |
| $BI_7(2)$-$BI_8(10)$ | **73.8** | +600 |
| DeepLab-CRF [52] | 72.7 | +830 |
| DeepLab-MSc-CRF [52] | **73.6** | +880 |
| DeepLab-EdgeNet [51] | 71.7 | +30 |
| DeepLab-EdgeNet-CRF [51] | **73.6** | +860 |

Table A.5: **Semantic Segmentation using the DeepLab model.** IoU scores on the Pascal VOC12 segmentation test dataset with different models and our modified inception model. Also shown are the corresponding runtimes in milliseconds. Runtimes also include superpixel computations (300 ms with Dollar superpixels [66])

of [52] and the CRFasRNN architecture from [279] which uses a CNN with deconvolution layers followed by DenseCRF trained end-to-end.

**DeepLab Model**    We experimented with the $BI_7(2)$-$BI_8(10)$ inception model. Results using the DeepLab model are summarized in Tab. A.5. Although we get similar improvements with inception modules as with the explicit kernel computation, using lattice approximation is slower.

**CRFasRNN Model**    We add BI modules after score-pool3, score-pool4, $FC_7$ and $FC_8$ $1 \times 1$ convolution layers resulting in the $BI_3(6)$-$BI_4(6)$-$BI_7(2)$-$BI_8(6)$ model and also experimented with another variant where $BI_8$ is followed by another inception module, $G(6)$, with 6 Gaussian kernels. Note that here also we discarded both deconvolution and DenseCRF parts of the original model [279] and inserted the BI modules in the base CNN and found similar improvements compared to the inception modules with explicit kernel computaion. See Tab. A.6 for results on the CRFasRNN model.

**Material Segmentation**    Table A.7 shows the results on the MINC dataset [25] obtained by modifying the AlexNet architecture with our inception modules. We observe similar improvements as with explicit kernel construction. For this model, we do not provide any learned setup due to very limited segment training data. The weights to combine outputs in the bilateral inception layer are found by validation on the validation set.

**Scales of Bilateral Inception Modules**    Unlike the explicit kernel technique presented in the main text (Chapter 7), we didn't back-propagate through feature transformation ($\Lambda$)

| Model | IoU (Val) | IoU (Test) |
|---|---|---|
| CNN | 67.5 | - |
| DeconvNet(CNN+Deconvolutions) | 69.8 | 72.0 |
| $BI_3(6)$-$BI_4(6)$-$BI_7(2)$-$BI_8(6)$ | 71.9 | - |
| $BI_3(6)$-$BI_4(6)$-$BI_7(2)$-$BI_8(6)$-$G(6)$ | 73.6 | **75.2** |
| DeconvNet-CRF(CRF-RNN) [279] | 73.0 | 74.7 |
| Context-CRF-RNN [273] | - | **75.3** |

Table A.6: **Semantic Segmentation using the CRFasRNN model.** IoU score corresponding to different models on Pascal VOC12 reduced validation / test segmentation dataset. The reduced validation set consists of 346 images as used in [279] where we adapted the model from.

| Model | Class / Total accuracy |
|---|---|
| AlexNet CNN | 55.3 / 58.9 |
| $BI_7(2)$-$BI_8(6)$ | 68.5 / 71.8 |
| $BI_7(2)$-$BI_8(6)$-$G(6)$ | 67.6 / 73.1 |
| AlexNet-CRF | 65.5 / 71.0 |

Table A.7: **Material Segmentation using AlexNet.** Pixel accuracy of different models on the MINC material segmentation test dataset [25].

using the approximate bilateral filter technique. So, the feature scales are hand-specified and validated, which are as follows. The optimal scale values for the $BI_7(2)$-$BI_8(2)$ model are found by validation for the best performance which are $\sigma_{xy} = (0.1, 0.1)$ for the spatial (XY) kernel and $\sigma_{rgbxy} = (0.1, 0.1, 0.1, 0.01, 0.01)$ for color and position (RGBXY) kernel. Next, as more kernels are added to $BI_8(2)$, we set scales to be $\alpha*(\sigma_{xy}, \sigma_{rgbxy})$. The value of $\alpha$ is chosen as 1, 0.5, 0.1, 0.05, 0.1, at uniform interval, for the $BI_8(10)$ bilateral inception module.

## A.5.2  Qualitative Results

In this section, we present more qualitative results obtained using the BI module with explicit kernel computation technique presented in Chapter 7. Results on the Pascal VOC12 dataset [77] using the DeepLab-LargeFOV model are shown in Fig. A.20, followed by the results on MINC dataset [25] in Fig. A.21 and on Cityscapes dataset [57] in Fig. A.22.

(a) Input     (b) Superpixels     (c) GT     (d) Deeplab     (e) +DenseCRF     (f) Using BI
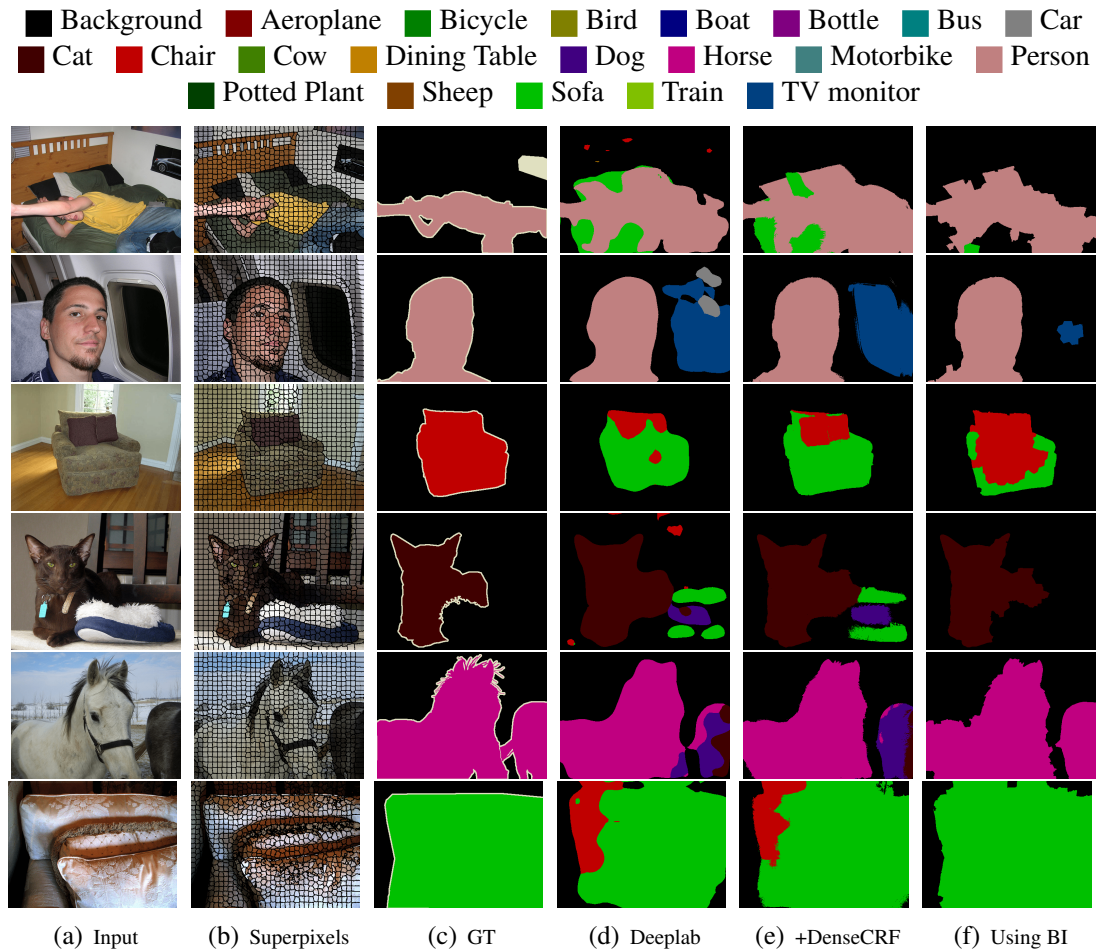
Figure A.20: **Semantic Segmentation.** Example results of semantic segmentation on the Pascal VOC12 dataset. (d) depicts the DeepLab CNN result, (e) CNN + 10 steps of mean-field inference, (f result obtained with bilateral inception (BI) modules ($BI_6(2)+BI_7(6)$) between FC layers.
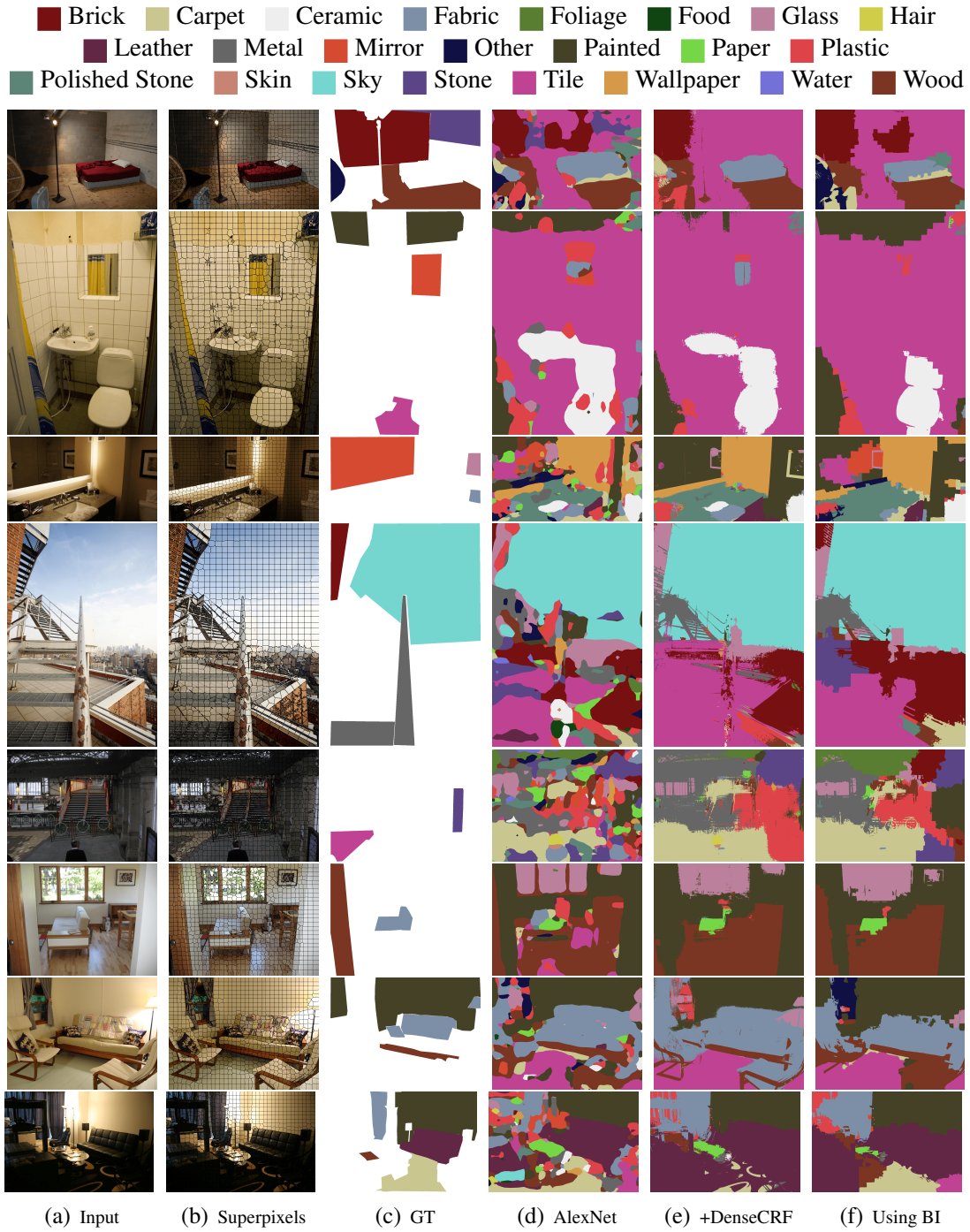
Figure A.21: **Material Segmentation.** Example results of material segmentation. (d) depicts the AlexNet CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($BI_7(2)+BI_8(6)$) between FC layers.

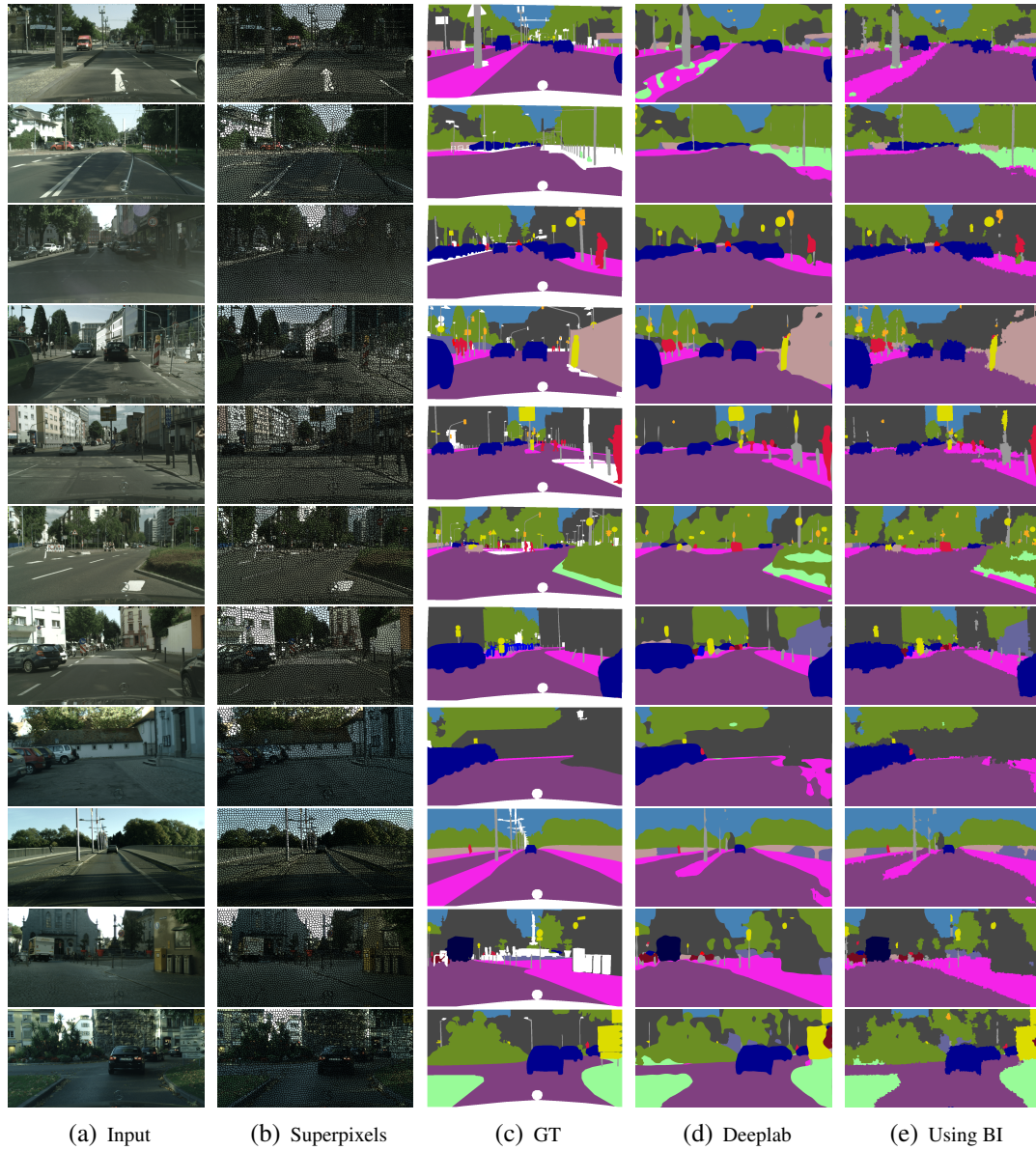(a) Input     (b) Superpixels     (c) GT     (d) Deeplab     (e) Using BI

Figure A.22: **Street Scene Segmentation.** Example results of street scene segmentation. (d) depicts the DeepLab results, (e) result obtained by adding bilateral inception (BI) modules ($BI_6(2)+BI_7(6)$) between FC layers.

# Bibliography

[1] Amazon Mechanical Turk. https://www.mturk.com/mturk/welcome. [Online; accessed 21-August-2016].

[2] CryEngine. https://www.cryengine.com/. [Online; accessed 21-August-2016].

[3] Google Images. https://images.google.com/. [Online; accessed 1-March-2015].

[4] Lumberyard. https://aws.amazon.com/lumberyard/. [Online; accessed 21-August-2016].

[5] Oculus Rift. https://www3.oculus.com/en-us/rift/. [Online; accessed 1-December-2016].

[6] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012.

[7] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, 2010.

[8] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. In *ToG*, 2009.

[9] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ToG*, 2004.

[10] S. Ahn, Y. Chen, and M. Welling. Distributed and adaptive darting Monte Carlo through regenerations. In *AISTATS*, 2013.

[11] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. In *ToG*, 2005.

[12] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 2011.

[13] Y. F. Atchadé and J. S. Rosenthal. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli*, 2005.

[14] K. B. Athreya and P. Ney. A new approach to the limit theory of recurrent Markov chains. *Transactions of the American Mathematical Society*, 1978.

[15] V. Aurich and J. Weule. Non-linear Gaussian filters performing edge preserving diffusion. In *DAGM*. 1995.

[16] S. P. Awate and R. T. Whitaker. Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering. In *CVPR*, 2005.

[17] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[18] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ToG*, 2009.

[19] D. Barash. Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *PAMI*, 2002.

[20] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In *CVPR*, 2015.

[21] J. T. Barron and B. Poole. The fast bilateral solver. In *ECCV*, 2016.

[22] B. G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, Stanford university, 1974.

[23] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[24] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ToG*, 2014.

[25] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. *CVPR*, 2015.

[26] I. G. Y. Bengio and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

[27] S. Biswas, G. Aggarwal, and R. Chellappa. Robust estimation of albedo for illumination-invariant matching and shape recovery. *PAMI*, 2009.

[28] S. Biswas and R. Chellappa. Pose-robust albedo estimation from a single image. In *CVPR*, pages 2683–2690, 2010.

[29] M. J. Black, D. J. Fleet, and Y. Yacoob. Robustly estimating changes in image appearance. *CVIU*, 2000.

[30] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *COMPSTAT*, pages 721–728, 2004.

[31] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

[32] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001.

[33] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. OReilly, 2008.

[34] L. Breiman. Bagging predictors. *Machine learning*, 1996.

[35] L. Breiman. Random forests. *Machine learning*, 2001.

[36] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

[37] S. Brooks and A. Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 1998.

[38] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011.

[39] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2009.

[40] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008.

[41] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *CVPR*, 2009.

[42] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[43] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

[44] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *CVPR*, 2012.

[45] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. *arXiv preprint arXiv:1611.05198*, 2016.

[46] D. Cahan. *Hermann von Helmholtz and the foundations of nineteenth-century science*. Univ. of California Press, 1993.

[47] N. Campbell, K. Subr, and J. Kautz. Fully-connected crfs with non-parametric pairwise potential. In *CVPR*, 2013.

[48] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *ECCV*, 2016.

[49] J.-H. R. Chang and Y.-C. F. Wang. Propagated image filtering. In *CVPR*, 2015.

[50] A. Y. Chen and J. J. Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *WACV*, 2011.

[51] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform. In *CVPR*, 2016.

[52] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[53] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[54] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun. Learning deep structured models. In *ICML*, 2015.

[55] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. *ToG*, 2002.

[56] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, 2012.

[57] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. In *CVPR Workshops*, 2015.

[58] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995.

[59] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

[60] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 2001.

[61] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[62] P. A. Dawid. The well-calibrated Bayesian. *Journal of the American Statistical Association*, 1982.

[63] M. de La Gorce, N. Paragios, and D. J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *CVPR*, 2008.

[64] R. de Nijs, S. Ramos, G. Roig, X. Boix, L. Van Gool, and K. Kühnlenz. On-line semantic perception using uncertainty. In *International Conference on Intelligent Robots and Systems*, 2012.

[65] L. Del Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *CVPR*, 2012.

[66] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[67] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 2015.

[68] J. Domke. Parameter learning with truncated message-passing. *CVPR*, 2011.

[69] J. Domke. Learning graphical model parameters with approximate marginal inference. *PAMI*, 2013.

[70] R. Dondera, V. Morariu, Y. Wang, and L. Davis. Interactive video segmentation using occlusion boundaries and temporally coherent superpixels. In *WACV*, 2014.

[71] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[72] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[73] S. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *NIPS*, 2016.

[74] S. M. A. Eslami, N. Heess, and J. M. Winn. The shape Boltzmann machine: A strong model of object shape. In *CVPR*, 2012.

[75] S. M. A. Eslami, D. Tarlow, P. Kohli, and J. Winn. Just-In-Time Learning for Fast and Flexible Inference. In *NIPS*. 2014.

[76] A. Ess, T. Mueller, H. Grabner, and L. J. Van Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009.

[77] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL VOC2012 challenge results. 2012.

[78] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, 2014.

[79] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. Jumpcut: non-successive mask transfer and interpolation for video cutout. *ToG*, 2015.

[80] D. Ferstl, M. Ruether, and H. Bischof. Variational depth superresolution using example-based edge representations. In *ICCV*, 2015.

[81] J. M. Flegal, M. Haran, and G. L. Jones. Markov chain Monte Carlo: Can we trust the third significant figure? *Statistical Science*, 2008.

[82] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. In *ToG*, 2003.

[83] C. W. Fox and S. J. Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 2012.

[84] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, 1995.

[85] B. J. Frey and N. Jojic. Advances in algorithms for inference and learning in complex probability models. *PAMI*, 2003.

[86] B. J. Frey and D. J. MacKay. A revolution: Belief propagation in graphs with cycles. In *NIPS*, 1998.

[87] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer, 2001.

[88] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. Gehler. Superpixel convolutional networks using bilateral inceptions. In *ECCV*, 2016.

[89] R. Gadde, V. Jampani, R. Marlet, and P. V. Gehler. Efficient 2D and 3D facade segmentation using auto-context. *PAMI*, 2017.

[90] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[91] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *ToG*, 2011.

[92] E. S. Gastal and M. M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ToG*, 2012.

[93] A. Gelman and D. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 1992.

[94] A. S. Georghiades, P. N. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *PAMI*, 2001.

[95] C. J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Proceedings of the 23rd Symposium on the Interface*, 1991.

[96] W. R. Gilks, G. O. Roberts, and S. K. Sahu. Adaptive Markov chain Monte Carlo through regeneration. *Journal of the American statistical association*, 1998.

[97] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[98] J. M. Gonfaus, X. Boix, J. Van de Weijer, A. D. Bagdanov, J. Serrat, and J. Gonzalez. Harmony potentials for joint classification and segmentation. In *CVPR*, 2010.

[99] S. Gould, J. Zhao, X. He, and Y. Zhang. Superpixel graph label transfer with learned distance metric. In *ECCV*, 2014.

[100] B. Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.

[101] B. Graham. Sparse 3D convolutional neural networks. In *BMVC*, 2015.

[102] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 1995.

[103] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[104] U. Grenander. *Pattern Synthesis - Lectures in Pattern Theory*. Springer, New York, 1976.

[105] R. B. Grosse, C. J. Maddison, and R. Salakhutdinov. Annealing between distributions by averaging moments. In *NIPS*. 2013.

[106] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010.

[107] J. M. Hammersley and K. W. Morton. Poor man's Monte Carlo. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1954.

[108] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[109] K. He, J. Sun, and X. Tang. Guided image filtering. *PAMI*, 2013.

[110] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[111] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[112] S. He, R. W. Lau, W. Liu, Z. Huang, and Q. Yang. Supercnn: A superpixelwise convolutional neural network for salient object detection. *ICCV*, 2015.

[113] N. Heess, D. Tarlow, and J. Winn. Learning to Pass Expectation Propagation Messages. In *NIPS*. 2013.

[114] J.-H. Heu, D.-Y. Hyun, C.-S. Kim, and S.-U. Lee. Image and video colorization based on prioritized source propagation. In *ICIP*, 2009.

[115] G. E. Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 1990.

[116] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 1995.

[117] D. A. Hirshberg, M. Loper, E. Rachlin, and M. J. Black. Coregistration: simultaneous alignment and modeling of articulated 3d shape. In *ECCV*, 2012.

[118] T. K. Ho. Random decision forests. In *ICDAR*, 1995.

[119] A. Holub and P. Perona. A discriminative framework for modelling object classes. In *CVPR*, 2005.

[120] B. K. P. Horn. Understanding image intensities. *Artificial Intelligence*, 1977.

[121] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 1991.

[122] H. Hu and G. de Haan. Trained bilateral filters and applications to coding artifacts reduction. In *ICIP*, 2007.

[123] T.-W. Hui, C. C. Loy, and X. Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, 2016.

[124] K. Hukushima and K. Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 1996.

[125] A. T. Ihler and D. A. McAllester. Particle belief propagation. In *AISTATS*, 2009.

[126] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2965–2973, 2015.

[127] T. S. Jaakkola, D. Haussler, et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

[128] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[129] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. *arXiv preprint arXiv:1511.05298*, 2015.

[130] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *ECCV*, 2014.

[131] V. Jampani, S. M. A. Eslami, D. Tarlow, P. Kohli, and J. Winn. Consensus message passing for layered graphical models. In *AISTATS*, 2015.

[132] V. Jampani, R. Gadde, and P. V. Gehler. Efficient facade segmentation using auto-context. In *WACV*, 2015.

[133] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, 2017.

[134] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks. In *CVPR*, 2016.

[135] V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler. The informed sampler: A discriminative approach to bayesian inference in generative computer vision models. *CVIU*, 2015.

[136] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014.

[137] J. Jiang and Z. Tu. Efficient scale space auto-context for image segmentation and labeling. In *CVPR*, 2009.

[138] H. Jungong, S. Ling, X. Dong, and J. Shotton. Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Transactions on Cybernetics*, 2013.

[139] R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal. Markov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 1998.

[140] I. Kemelmacher-Shlizerman and R. Basri. 3d face reconstruction from a single image using a single reference face shape. *PAMI*, 2011.

[141] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. *arXiv preprint arXiv:1612.02646*, 2016.

[142] M. Kiefel and P. V. Gehler. Human pose estimation with fields of parts. In *ECCV*. 2014.

[143] M. Kiefel, V. Jampani, and P. V. Gehler. Permutohedral lattice CNNs. In *ICLR Workshops*, 2015.

[144] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[145] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[146] P. Kohli and P. H. Torr. Dynamic graph cuts for efficient inference in markov random fields. *PAMI*, 2007.

[147] D. Koller and N. Friedman. Probabilistic graphical models: Principles and techniques (adaptive computation and machine learning series). 2009.

[148] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ToG*, 2007.

[149] A. Korattikara, Y. Chen, and M. Welling. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. *arXiv preprint arXiv:1304.5299*, 2013.

[150] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *ICML*.

[151] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. *NIPS*, 2011.

[152] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[153] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool. Fast optical flow using dense inverse search. In *ECCV*, 2016.

[154] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016.

[155] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[156] M. Lang, O. Wang, T. O. Aydin, A. Smolic, and M. H. Gross. Practical temporal consistency for image-based graphics applications. *ToG*, 2012.

[157] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *CVPR*, 2006.

[158] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[159] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[160] A. B. Lee, D. Mumford, and J. Huang. Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model. *IJCV*, 2001.

[161] K.-C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *PAMI*, 2005.

[162] M. W. Lee and I. Cohen. Proposal maps driven MCMC for estimating human body pose in static images. In *CVPR*, 2004.

[163] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011.

[164] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ToG*, 2004.

[165] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *PAMI*, 2008.

[166] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.

[167] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep joint image filtering. In *ECCV*, 2016.

[168] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. *ToG*, 2005.

[169] Y. Li and R. Zemel. Mean-field networks. In *ICML Workshops*, 2014.

[170] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016.

[171] G. Lin, C. Shen, I. Reid, and A. van den Hengel. Deeply learning the messages in message passing inference. In *NIPS*, 2015.

[172] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014.

[173] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[174] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. 2001.

[175] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[176] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[177] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ToG*, 2015.

[178] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[179] V. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum. Approximate Bayesian image interpretation using generative probabilistic graphics programs. In *NIPS*, 2013.

[180] N. Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *CVPR*, 2016.

[181] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 1953.

[182] O. Miksik, D. Munoz, J. A. Bagnell, and M. Hebert. Efficient temporal consistency for streaming video scene analysis. In *ICRA*, 2013.

[183] P. Milanfar. A tour of modern image filtering. *IEEE Signal Processing Magazine*, 2013.

[184] T. Minka. *Expectation Propagation for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[185] T. Minka. Discriminative models, not discriminative training. Technical report, Technical Report MSR-TR-2005-144, Microsoft Research, 2005.

[186] T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer.NET 2.5, 2012. Microsoft Research Cambridge.

[187] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

[188] D. Mumford and A. Desolneux. *Pattern Theory: The Stochastic Analysis of Real-World Signals*. 2010.

[189] D. Munoz. *Inference Machines: Parsing Scenes via Iterated Predictions*. PhD thesis, Carnegie Mellon University, 2013.

[190] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *ECCV*, 2010.

[191] P. Mykland, L. Tierney, and B. Yu. Regeneration in Markov chain samplers. *Journal of the American Statistical Association*, 1995.

[192] A. Ng and A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS*, 2002.

[193] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. In *ECCV*, 2010.

[194] E. Nummelin. A splitting technique for Harris recurrent Markov chains. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 1978.

[195] N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions. *PAMI*, 2000.

[196] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2010.

[197] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013.

[198] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV*, 2008.

[199] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*. 2006.

[200] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand. *Bilateral filtering: Theory and applications*. Now Publishers Inc, 2009.

[201] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[202] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *ICCV*, 2015.

[203] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 2009.

[204] S. J. Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[205] Y. Quéau, F. Lauze, and J.-D. Durou. *Solving the uncalibrated photometric stereo problem using total variation*. Springer, 2013.

[206] S. A. Ramakanth and R. V. Babu. Seamseg: Video object segmentation using patch seams. In *CVPR*, 2014.

[207] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh. Pose machines: Articulated pose estimation via inference machines. In *ECCV*, 2014.

[208] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *Computer graphics and interactive techniques*, 2001.

[209] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[210] C. Y. Ren, V. A. Prisacariu, and I. D. Reid. gslicr: SLIC superpixels at over 250hz. *ArXiv e-prints*, (1509.04232), 2015.

[211] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[212] M. Reso, B. Scheuermann, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Interactive segmentation of high-resolution video content using temporally coherent superpixels and graph cut. In *International Symposium on Visual Computing*, pages 281–292. Springer, 2014.

[213] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[214] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[215] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[216] M. N. Rosenbluth and A. W. Rosenbluth. Monte Carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics*, 1955.

[217] S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell. Learning Message-Passing Inference Machines for Structured Prediction. In *CVPR*, 2011.

[218] R. E. Schapire. The strength of weak learnability. *Machine learning*, 1990.

[219] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

[220] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[221] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *CVPR*, 2016.

[222] A. Shafaei, J. J. Little, and M. Schmidt. Play and learn: Using video games to train computer vision models. *arXiv preprint arXiv:1608.01745*, 2016.

[223] L. Shao, J. Han, D. Xu, and J. Shotton. Computer vision for RGB-D sensors: Kinect and its applications. *IEEE Transactions on Cybernetics*, 2013.

[224] R. Shapovalov, D. Vetrov, and P. Kohli. Spatial inference machines. In *CVPR*, 2013.

[225] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. *arXiv preprint arXiv:1608.03609*, 2016.

[226] B. Sheng, H. Sun, M. Magnor, and P. Li. Video colorization using parallel optimization in feature space. *IEEE Transactions on Circuits and Systems for Video Technology*, 2014.

[227] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 2000.

[228] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[229] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[230] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[231] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*. 2012.

[232] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[233] C. Sminchisescu and M. Welling. Generalized darting Monte Carlo. *Pattern Recognition*, 2011.

[234] S. M. Smith and J. M. Brady. SUSAN – a new approach to low level image processing. *IJCV*, 1997.

[235] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[236] A. Stuhlmüller, J. Taylor, and N. Goodman. Learning stochastic inverses. In *NIPS*, 2013.

[237] P. Sturgess, K. Alahari, L. Ladicky, and P. H. Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC*, 2009.

[238] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. *Communications of the ACM*, 2010.

[239] D. Sun, J. Wulff, E. B. Sudderth, H. Pfister, and M. J. Black. A fully-connected layered model of foreground and background flow. In *CVPR*, 2013.

[240] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010.

[241] R. H. Swendsen and J.-S. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 1986.

[242] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[243] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *TIP*, 2007.

[244] Y. Tang, R. Salakhutdinov, and G. Hinton. Deep Lambertian networks. *arXiv preprint arXiv:1206.6445*, 2012.

[245] B. Taylor, V. Karasev, and S. Soattoc. Causal video object segmentation from persistence of occlusions. In *CVPR*, 2015.

[246] D. Teets and K. Whitehead. The discovery of Ceres: How Gauss became famous. *Mathematics Magazine*, 1999.

[247] J. B. Tenenbaum. Isomap - code. http://isomap.stanford.edu/, 2000. [Online; accessed 12-October-2015].

[248] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

[249] L. Theis and M. Bethge. Generative image modeling using spatial lstms. In *NIPS*, 2015.

[250] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008.

[251] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[252] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014.

[253] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen. Semantic video segmentation: Exploring inference efficiency. In *ISOCC*, 2015.

[254] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label MRF optimization. *IJCV*, 2012.

[255] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *CVPR*, 2016.

[256] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *PAMI*, 2010.

[257] Z. Tu and S.-C. Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *PAMI*, 2002.

[258] V. Vineet, G. Sheasby, J. Warrell, and P. H. Torr. Posefield: An efficient mean-field based method for joint estimation of human pose, segmentation and depth. In *EMMCVPR*, 2013.

[259] V. Vineet, J. Warrell, and P. H. Torr. Filter-based mean-field inference for random fields with higher order terms and product label-spaces. In *ECCV*, 2012.

[260] J. Vorba, O. Karlík, M. Šik, T. Ritschel, and J. Křivánek. On-line learning of parametric mixture models for light transport simulation. *ToG*, 2014.

[261] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 2008.

[262] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Transactions on Graphics (ToG)*, 24(3):585–594, 2005.

[263] W. Wang, J. Shen, and F. Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, 2015.

[264] Y. Wang, L. Zhang, Z. Liu, G. Hua, Z. Wen, Z. Zhang, and D. Samaras. Face relighting from a single image under arbitrary unknown lighting conditions. *PAMI*, 2009.

[265] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 2000.

[266] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graph-ical models of arbitrary topology. *Neural computation*, 2001.

[267] J. Winn and C. M. Bishop. Variational Message Passing. *Journal of Machine Learning Research*, 2005.

[268] D. H. Wolpert. Stacked generalization. *Neural networks*, 1992.

[269] S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. In *ECCV*, 2016.

[270] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, 2015.

[271] H. Yagou, Y. Ohtake, and A. Belyaev. Mesh smoothing via mean and median filtering applied to face normals. In *Geometric Modeling and Processing*, 2002.

[272] O. Yakhnenko, A. Silvescu, and V. Honavar. Discriminatively trained Markov model for sequence classification. In *ICDM*, 2005.

[273] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[274] A. Yuille and D. Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in cognitive sciences*, 2006.

[275] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[276] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional net-works. In *CVPR*, 2010.

[277] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013.

[278] L. Zhang and D. Samaras. Face recognition from a single training image under arbitrary unknown lighting using spherical harmonics. *PAMI*, 2006.

[279] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

[280] S. C. Zhu and D. Mumford. Learning generic prior models for visual computation. In *CVPR*, 1997.

[281] S. C. Zhu, R. Zhang, and Z. Tu. Integrating bottom-up/top-down for object recog-nition by data driven Markov chain Monte Carlo. In *CVPR*, 2000.

[282] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2005.