Représentations à base de parties pour la vision 3D de haut niveau

Stefan Kinauer

université PARIS-SACLAY

CentraleSupélec

NNT : 2018SACLC059

Thèse de doctorat

Représentations à base de parties pour la vision 3D de haut niveau

Part-Based Representations for High-Level 3D Vision

Thèse de doctorat de l'Université Paris-Saclay
préparée à CentraleSupélec

Ecole doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Traitement du signal et des images

Thèse présentée et soutenue à Gif-sur-Yvette, le 31 août 2018, par

Stefan Kinauer

Composition du Jury :

Nikos Paragios Professeur, CentraleSupélec (CVN)	Président
Stefanos Zafeiriou Reader, Imperial College London	Rapporteur
Dimitris Samaras Associate Professor, Stony Brook University	Rapporteur
Celine Hudelot Professeur des Universités, CentraleSupélec (MICS)	Examinateur
Chaohui Wang Maître de Conférences, Université Paris-Est	Examinateur
Iasonas Kokkinos Senior Lecturer, University College London	Directeur de thèse

Acknowledgments

First of all, I want to thank Iasonas, my advisor, who had the patience and endurance to stick with me throughout this thesis. Many things I learned in these years, are due to his model and advice. His practical and hands-on approach shaped my PhD and when deadlines were closing in, Iasonas was never shy of throwing everything into it. His effort and dedication was motivating and inspiring. My special thanks are due to letting me form and develop his algorithmic "baby".

Second, I would like to thank Dimitris Samaras and Stefanos Zafeiriou who agreed to review this thesis, as well as Celine Hudelot, Nikos Paragios and Chaohui Wang who accepted to act as jury over my PhD defense.

Third, I want to acknowledge the support grant of the European Union for the RECONFIG project to which I contributed in my PhD. Not only was this project successful in terms of project goals, but it also enabled my research and widened my insight into related research fields. I also want to thank INRIA and the Ecole Centrale de Paris (now CentraleSupelec) for providing me a home during the last years and the setup for my work. And not the least I want to thank again Nikos, the (former) head of the laboratory, for giving me the opportunity to do my PhD with CentraleSupelec.

I had the pleasure of working with a number of great colleagues throughout my PhD. I want to mention especially Maxim, Alp and Siddhartha collaborated closely with me on two papers. Furthermore it is important to mention Jie Ying who helped me a great deal with correcting language-based errors in my thesis, and Khue and Maxim who translated the abstract into French.

Next to above mentioned friends, I want to say thanks also to Mihir, Khue, Jiaqian, Enzo, Hari, Puneet, Maria, Stavros, Marie-Carol, Eugene, Evgenios, Erwan, Haithem, Raphael, Matthew and Eva. Their moral support and also technical advice were indispensable in these years. There are even more colleagues to be mentioned, and all together they made these years entertaining, funny and instructive.

I also want to mention my colleagues in the RECONFIG project. Our contact was always very friendly and cooperative, such that the time spent together on the project was fun, enjoyable and productive.

Moreover I want to thank Natalia, the good soul of the lab, who held the lab afloat with her organizational skills. She helped me and others countless times while settling in and with bureaucracy within the school and public institutions.

And finally not forgotten is the help I received from my family at home. Without the education and liberty they let me obtain, their love and moral support I would not be here.

My special thoughts go to Paola who never doubted me, who always found the right words and has become my best friend and so much more.

Abstract

In this work we use Deformable Part Models (DPMs) to learn and detect object parts in three dimensions. Given a single RGB image of an object, the objective is to determine the location of the object's parts. This is a problem of high relevance for many practical applications. The resulting optimization problem is non-convex and challenging due to its large solution space.

Our first contribution consists in extending DPMs into the third dimension through an efficient Branch-and-Bound algorithm that operates over a three-dimensional pose space. We exploit the problem structure and devise a customized, efficient algorithm that is two orders of magnitude faster than a naive approach. Additionally to being fast, we also inherit the global-optimality guarantees of Branch-and-Bound algorithms. An object class is modeled by several viewpoint-specific models. We derive each model's three-dimensional geometry from a class-specific three-dimensional structure, but train viewpoint-specific part appearance terms. We demonstrate our approach on the task of three-dimensional object pose estimation, learning three-dimensional part models based on deep learning features and determining the object pose within a fraction of a second.

Our second contribution allows us to perform efficient inference with part-based models where the part connections form a graph with loops, thereby extending the set of part relationships that can be modelled. For this, we use the Alternating Direction Method of Multipliers (ADMM) in combination with the introduced Branch-and-Bound algorithm. With ADMM we decouple a potentially hard problem and solve iteratively a set of smaller and easier sub-problems to reach an approximately optimal solution. We compute three-dimensional unary and pairwise model parameters in a Convolutional Neural Network for three-dimensional human pose estimation. Then we append the developed inference algorithm as final layer to this neural network to improve precision. This yields state of the art performance on the Human3.6M dataset in the three-dimensional human pose estimation task.

Résumé

Dans cette thèse, nous utilisons des modèles de parties déformables (Deformable Part Models – DPMs) pour apprendre à détecter des parties d'objets. Pour une image d'un objet, l'objectif est de déterminer l'emplacement des parties de cet objet dans l'image. Ceci est un problème de haute importance pour de nombreuses applications pratiques. Le problème d'optimisation qui en résulte est non-convexe et difficile en raison de son grand espace de recherche.

Notre première contribution consiste à étendre les DPMs à la troisième dimension, grâce à un algorithme par séparation et évaluation (Branch-and-Bound) efficace qui fonctionne sur un espace de pose en 3D. Nous exploitons la structure du problème et élaborons un algorithme personnalisé et efficace qui est deux fois plus rapide qu'une approche naïve. En plus d'être rapide, notre algorithme hérite également de la garantie d'optimalité globale des algorithmes par séparation et évaluation. L'appartenance d'un object a une classe est caractérisée par plusieurs modèles de point de vue. Nous dérivons pour chaque modèle tridimensionnel, une structure tridimensionnels spécifiques a la classe de l'objet. Cependant, nous entrainons un algorithm prenant en compte chaque sous point de vue de l'apparence. Nous démontrons notre approche sur la tache de l'estimation tridimensionnel de la posture, via l'apprentissage de plusieurs modèles pour chaque partie basé sur des caractéristiques provenant de l'apprentissage profond et de determiner la posture de l'objet en une fraction de second.

Notre deuxième contribution nous permet d'effectuer une inférence efficace sur des modèles où les connexions des parties forment un graphe avec des boucles, étendant ainsi l'ensemble des relations de parties qui peuvent être modélisées. Pour cela, nous utilisons l'algorithme des directions alternées (Alternating Direction Method of Multipliers – ADMM) en combinaison avec l'algorithme par séparation et évaluation introduit. Avec ADMM, nous découplons un problème potentiellement difficile et résolvons itérativement un ensemble de sous-problèmes plus petits et plus faciles pour atteindre une solution approximativement optimale. Nous calculons les paramètres unaires et paires du modèle via un Réseaux Neuronal Convolutif pour la determination de la posture tridimensionnel. L'inference développée est utilisée comme dernière couche du réseau neural afin d'améliorer la precision. Cela permet d'obtenir une performance à l'état de l'art sur le jeu de données Human3.6M pour la tâche d'estimation de pose humaine en 3D.

Contents

1 Overview **7**

2 Background / Previous Work **11**
2.1 Object Representation 12
2.1.1 Object Detection with Sliding Windows 14
2.1.2 Part-Based Models 18
2.1.3 Convolutional Neural Networks 22
2.2 Optimization Methods for Part-Based Models 27
2.2.1 Generalized Distance Transform with Dynamic Programming 27
2.2.2 Cascades 30
2.2.3 Dual-Tree Branch-and-Bound 31
2.3 Model Learning 35
2.3.1 Support Vector Machines (SVMs) 35
2.3.2 Structured SVM 36
2.3.3 Latent Variable SVM 38
2.4 Outlook 39

3 3D Object Pose Estimation with 2D Unary Potentials **41**
3.1 Prior Work on 3D Pose Estimation 42
3.2 3D DPM - Modeling 46
3.2.1 3D DPM with 2D Unary Potentials 46
3.2.2 Viewpoint-Independent Structure 48
3.2.3 Viewpoint-Specific Models 52
3.2.4 Model Training 54
3.3 3D DPM - Inference 56
3.3.1 Joint Inference with Depth Variables 56
3.3.2 Anchoring in Depth 61
3.4 Evaluation 62
3.4.1 Model Validation 62
3.4.2 Runtime Performance 66

4 Monocular 3D Human Pose Estimation **69**
4.1 Prior Work in Human Pose Estimation 70
4.2 DPM and Inference in Higher Dimensions 73
4.2.1 Deformable Part Models in 3D 73
4.2.2 Inference with Dual-Tree Branch-and-Bound in 3D 74
4.3 Training with Deep Supervision 74
4.4 DPM Inference on Arbitrary Graph Topologies 76
4.4.1 Graph Decomposition with ADMM 78
4.4.2 Inference of ADMM-Augmented Subproblem 82

4.4.3 Inference Visualization 83
4.5 Results on 3D Human Pose Estimation 90
4.5.1 Evaluation Setup 90
4.5.2 Implementation Details 90
4.5.3 Quantitative Comparison 91
4.5.4 Ablation Study with Graph Topologies 91
4.5.5 Improvement of 2D Joint Localizations 99
4.5.6 Qualitative Evaluation on the Leeds Sports Dataset 100
4.5.7 Runtime 101

5 Future Work **103**
5.1 Conclusion 103
5.2 Future Work 104
5.2.1 Applications 104
5.2.2 Optimization Algorithm 105

A DPMs for Viewpoint Invariant Detection in Robotics **107**
A.1 Object Detection for Identity Anchoring 107
A.1.1 Identity Anchoring 108
A.1.2 Exigence of Practical Applications in Robotics 108
A.1.3 Previous Approaches to Object Detection in Robotics 110
A.2 Efficient Implementation 112
A.2.1 DPM Training 112
A.2.2 Data Collection 113
A.2.3 Inference Algorithms for Object Detection 114

Bibliography **117**

List of Figures

1.1 3D human pose estimation example (Leeds Sports Dataset [Johnson 2010]). The body pose in (a) is easily detectable for the human eye. The obtained solution is indicated in (b) and (c). . . . 7
(a) . . . 7
(b) . . . 7
(c) . . . 7
1.2 Summary of results in one figure. In (a)-(c) we illustrate results presented in Chapter 3 where (a) shows a 3-dimensional mesh of a car that adapts to the given image instance in (b) and (c) to recover the objects' pose. [Kinauer 2016] In (d)-(g) we demonstrate human pose estimation in 3D of Chapter 4. The input image (d) is given to our CNN which predicts the 3D human pose, illustrated in (e) in the image and in (f), (g) in new perspectives, respectively. 9

2.1 Instances of the "car" object class. A car detector should detect the cars in all three images. The instances exhibit variation in the car's model, color, scale, the camera's viewpoint and the background. . . 12
2.2 Top row: results of a car detector. Bottom row: results of a car detector with parts. We point out that we have created this and further examples of object detection in Section 2.1 by running the object detector of [Girshick 2012]. Where needed, we have adapted the models, for example by removing object parts in the context of rigid object models. . . . 13
2.3 Object detector example. We apply a "bicycle", a "person" and a "car" detector. The detections are indicated as red bounding boxes. 15
2.4 HOG-feature pyramid for an example image. An object detector is run against all pyramid levels, removing the need of training different object detectors for different scales. We depict 6 pyramid levels of HOG features, computed by successively downscaling the input image and extracting the features. As a result, at different pyramid levels different levels of details stand out. . . . 16
2.5 A "person" model with 6 mixture models. We can identify the rough head shape and varying portions of the human body. The second mixture model is the mirrored mixture model of the first one, similarly are the fourth and sixth ones mirrored mixture models of the third and fifth one. . . . 17
(a) M_1 . . . 17
(b) M_2 . . . 17
(c) M_3 . . . 17
(d) M_4 . . . 17

(e) M_5 . . . 17
(f) M_6 . . . 17
2.6 Illustration of a sliding window object detector. A window of the size of the model, M_6 of Figure 2.5f, is slid over the feature pyramid levels and the corresponding score according to Equation 2.1 is calculated. The resulting scores are visualized as a heat-map, where red corresponds to high scores and blue to low scores. In the last column we show the bounding boxes that correspond to scores over a certain threshold value. A detection in a high-resolution pyramid level corresponds to small bounding box, and vice versa. . . . 18
2.7 Example of non-maximum suppression. In the image on the left there are several almost redundant detections. In the image on the right, strongly overlapping bounding boxes have been removed using non-maximum suppression (NMS). . . . 19
(a) Detection results for "person". Several redundant bounding boxes clutter the detections. . . . 19
(b) Final detections after applying NMS. Strongly overlapping bounding boxes have been removed. . . . 19
2.8 Example DPM and its factor graph. . . . 21
2.9 Visualization of a "person" DPM. The top row shows the root-filter, w_1, that by itself offers a rigid model identical to the model in Figure 2.5. The bottom row shows the part filters, $w_2, ..., w_N$, in their spatial configuration. The green arrows illustrate the flexibility of part positions, $1/C_{1,2}, ..., 1/C_{1,N}$, of the corresponding parts, such that a long arrow corresponds to a small penalty for a deformation in this direction. Each column represents one mixture model. We show 3 out of 6 mixture models, since the other 3 are mirrored equivalents. . . . 22
2.10 Object part detection with DPMs. The blue rectangle stands for the low-resolution center part, the red rectangles indicate the other parts. Two people are detected with the same mixture model, but the flexible pairwise term allows varying poses. . . . 23
2.11 Neural Network with a convolutional and a fully connected layer. In a convolutional layer, neurons of the layer are connected to overlapping subsets of neurons in the layer below. Edge weights are shared in this layer. We visualize this by coloring edges accordingly in red, green and blue. In fully connected layers, all neurons of the layer are connected to all neurons of the layer below. Here, parameters are not shared. For reasons of clarity we single out the edges to a single node in both layers. . . . 24
2.12 Visualization of features of the ZF-net [Zeiler 2014]. The visualizations are created using deconvolutions of individual features in the first, second and third layer. . . . 25

2.13 Comparison of HOG features with $conv_5$ activations over several scales [Girshick 2015b]. The first column shows the input image, the remaining columns correspond to features extracted at different scales of the image pyramid. The first row displays the HOG feature pyramid, similar to Figure 2.4. In rows 2 to 4, the activations of 1 channel (out of 256) of the 5-th convolutional layer are visualized. The network is trained on the ImageNet dataset for scene classification. The chosen channel corresponds to the "head" label. 26

2.14 Generalized Distance Transform (GDT) [Felzenszwalb 2004]. (a) an example input. (b) the result after the first pass over the vertical dimension. (c) the final result. 29

2.15 Left: Input image with object detection result. Right: Binary maps demonstrating hypotheses pruning. The inference with cascades accelerates detection by pruning hypotheses below a certain threshold with simplified models. From top to bottom and left to right, the maps show the increasingly complex models are evaluated only on small portions of the image, indicated in white. [Felzenszwalb 2010a] 31

(a) Input image with detected object. The object is delineated in red, the object parts in blue bounding boxes. 31

(b) Evaluation of part appearance models. The binary images indicate where the part appearance model was evaluated in the course of the inference. The images from left to right and top to bottom refer to object parts $i = 2, ..., 7$. Large black parts indicate that significant portions have been skipped to save computation time. The appearance model of the center node $i = 1$ is evaluated everywhere and is not shown in this figure. 31

2.16 Object detection with Branch-and-Bound [Kokkinos 2011]. The first figure shows the original image with the optimal object bounding box in red. The second image illustrates which intervals the algorithm evaluated and therefore spent time on. The colors refer to the upper bound computed for the respective interval. The third image displays the exact score of the detector, evaluated densely. For both images on the right, blue stands for low scores and red stands for high scores. 33

2.17 Precomputing the unary potential. Left: example unary potential given as array. Right: resulting Kd-Tree. 34

2.18 Linear SVM in a 2 dimensional example case. 2.18a: Input data for binary classification into triangles and circles. 2.18b: Potential linear classifiers, all of which can correctly classify the given data points. 2.18c: The solution found by SVM maximizes the margin, in other words, the space around the separating line. 35

(a) 35

(b) 35

(c) 35

3.1 Example 3D DPM and its factor graph. 47

3.2 Mesh extracted with NRSfM for the "bicycle"-class. Data points, marked by "X", are drawn in a common coordinate system, each color corresponding to a different keypoint. The associated point on the mesh is shown as a filled circle. The computation of one nominal offset, $\mu_{i,j}$, is indicated in dark blue. 48

3.3 A wheelchair under varying viewpoint angles. The same three selected landmarks and their surrounding image patches are enlarged in both images (b) and (c). The corresponding image patches show a high degree of similarity in image (b) and (c). The landmarks' relative position to each other is indicated by the dashed triangles. We overlay the triangle of the landmarks in image (b) onto image (c) to highlight change under rotation. This motivates us to subdivide the viewing sphere into a small number of viewpoint bins and to train a DPM for every bin. 53

3.4 Illustration of the calculation of the pairwise term, $\bar{\mathcal{P}}_{i,j}(\xi_i, \xi_j)$. For better readability the third dimension is not shown. The interval, Γ_i, is given as the center (x_i, y_i, z_i) and the half size (x_i^s, y_i^s, z_i^s) (light red color). The interval, Γ_j, is given as the center (x_j, y_j, z_j) and the half size (x_j^s, y_j^s, z_j^s) (light green color). We subsume the nominal offset, μ, under the interval, $\Gamma_{j\mu}$, with the center being $(x_j + \mu_x, y_j + \mu_y, z_j + \mu_z)^T$. This corresponds to a shift of the interval, Γ_j, by μ. We illustrate the individual terms of Equation 3.20 in blue. 61

3.5 Illustration of the Hausdorff Distance between two shapes X and Y. The Hausdorff Distance is computed as the maximum between the smallest distance from a point on X to any Y such that no other point on X would yield a larger distance, and its symmetric counterpart with X and Y exchanged. Image from [Commons 2007]. 63

3.6 Example images of the PASCAL3D+ dataset with object detections. We have chosen 3 positive examples per category in the first three columns and one failure case in the fourth column. Yellow indicates small distance to the camera, whereas blue signifies a large distance. 66

3.7 Runtime comparison between the GDT-based optimization in blue and the Branch-and-Bound based optimization in yellow (proposed). The x-axis gives the number of discrete depth values available in the solution space. The y-axis shows the resulting runtime in seconds. The computation with GDT for a depth resolution greater than 300 was impossible due to memory limits. 67

3.8 Acceleration of Branch-and-Bound by eliminating equivalent solutions by anchoring the model at a certain depth. The ground truth's depth range is between 80 and 120. The x-axis gives the number of discrete depth values available in the solution space. The y-axis shows the resulting runtime in seconds. The red curve shows runtime for Branch-and-Bound without anchoring the center node's depth. The yellow curve shows performance with anchoring of the center node's depth in the middle of the available depth space. 67

4.1 Overview over our approach for 3D human pose estimation. A CNN computes unary and pairwise potentials in 3D. The subsequent discrete optimization via ADMM and Branch-and-Bound finds the most coherent pose among all possible configurations. 73

4.2 Unary 3D coordinates via quantized regression. Left: Sigmoid function on classified voxels and regressed residual vectors (in black) for two joints. Right: Regressed residual vectors for all joints. To efficiently regress the unary 3D coordinates, we use a combination of classification and regression. We first quantize the 3D space into voxels. We then estimate the score of each joint belonging to each of these voxels using a classifier. Finally we regress a residual vector per voxel, $r_i(v)$, which indicates the offset between the center of the voxel and the continuous 3D position of each joint (Equation 4.2). . 75

4.3 Example human body representations. In (a) 16 body joints as they are used in the Human MPII dataset, connected as a tree-shaped graph following the kinematic chain. In (b) we show the same 16 joints, now connected with additional edges, representing transitive (green) and symmetric (red) relations. This graph contains loops. . 77

4.4 Example graph decomposition. It is obvious that there are many alternative graph decompositions. 79

(a) A loopy graph G has to be decomposed into loop-free subgraphs. 79

(b) Star-shaped subgraph G_1 79

(c) Star-shaped subgraph G_2 79

(d) Star-shaped subgraph G_3 79

4.5 Selected graph configurations. 92

4.6 Monocular 3D pose estimation results on LSP dataset: we observe that our results transfer to unseen datasets, with highly different statistics from the Human3.6M dataset. 101

4.7 Runtime of Branch-and-Bound with varying size of label space, $|\Omega_{3D}|$. The time spent on building unary tree is displayed in red. The time spent in the Branch-and-Bound algorithm is depicted in blue. . . . 102

A.1 Left: Example of multiple toy cars. Right: Toy car bounding boxes obtained by our DPM object detector. We train a single model that covers the visual variability of all three car configurations. The objects are detected within one pass of the detection algorithm. 110
A.2 Determination of 3D coordinates based on 2D image coordinates and the robot's geometry. The robot's viewing ray to the object, determined by the robot's joints and the center of the object's bounding box, is intersected with the floor plane to determine the object's 3D position. 113
A.3 The effect of retraining DPMs in a specific environment. The model is a shampoo-bottle. On the left is the input image. In the middle a true positive and a false positive detection is found. On the right we show the result after retraining the model with additional negative samples (Figure A.4). 115
A.4 Additional negative samples in the laboratory setup we add to the training set in order to diminish false detections of the shampoo-bottle. 115

Chapter 1

Overview

(a) (b) (c)

Figure 1.1: 3D human pose estimation example (Leeds Sports Dataset [Johnson 2010]). The body pose in (a) is easily detectable for the human eye. The obtained solution is indicated in (b) and (c).

Looking at an image and recognizing not only the visible objects but also the 3D shape and pose of these objects is an easy task for humans. If we look at Figure 1.1a then we instantly understand the player's pose. We do this by drawing from our knowledge of objects and materials, lighting and perspective. For computers, this is a challenging task. In this thesis we predict the 3D structure of objects and humans by localizing the major (body) parts (see Figure 1.1b, 1.1c).

In particular, we advance algorithms for efficient structured prediction in 3D. We follow the Deformable Part Model (DPM) paradigm and model objects as collection of object parts, connected by geometric relationships. The detection of these inter-connected object parts requires solving a non-convex optimization problem. We solve this problem efficiently and exactly in a 3D pose space using a Branch-and-Bound approach.

The work is structured as follows:

- In Chapter 2 we provide the necessary background to the remainder of this thesis. We develop object representations for object detection and object pose estimation, going from rigid to part-based models. Specifically we focus on Deformable Part Models (DPMs). We consider DPMs that use not only classical Histogram of Oriented Gradients-based image representations, but also rich features computed by Convolutional Neural Networks (CNNs), and

we discuss the connection between CNNs and DPMs. Then we present optimization methods for DPMs, covering both linear (Generalized Distance Transform [Felzenszwalb 2004]) and sublinear methods, such as Cascades [Felzenszwalb 2010a], and describe the central ideas of the Dual-Tree Branch-and-Bound algorithm for DPMs. Finally, we cover training criteria and methods for the models formulated in the previous sections.

- In Chapter 3 we extend Branch-and-Bound to 3 dimensions. We analyze the given problem structure and devise a customized, efficient inference algorithm. This algorithm guarantees optimality and its runtime is a fraction of a second, despite the large label space in 3D.

  We learn object models based on CNN features, where DPM object parts correspond to partially annotated keypoints. To cover images from varying viewpoints, we train a small number of viewpoint-specific object models. We use soft pairwise constraints that allow small deformations and render prediction robust to minor perspective changes. We demonstrate our approach for object keypoint detection on the PASCAL3D+ dataset.

- In Chapter 4 we expand the expressivity of object models, allowing pairwise relationships to form loops. The complexity of the optimization of these models depends on the graph that is formed by the pairwise connections between object parts. In particular, loops make the problem harder. Our contribution allows us to perform inference with DPMs where the part relationships form a graph with loops. We propose to apply the Alternating Direction Method of Multipliers (ADMM), an approximate optimization scheme, that subdivides the original problem into sub-problems. ADMM steers iteratively these partial solutions towards a commonly agreed solution. The resulting ADMM term is naturally incorporated in our inference algorithm, maintaining the efficiency of our contribution from the previous chapter.

  To evaluate this approach, we train a CNN for human pose estimation in 3D and augment its functionality by integrating our optimization approach into the last layer of the network. This ensures the structural consistency of the solution. We demonstrate the benefits of this approach on the Human3.6M dataset and obtain state of the art results.

We have presented above summarized contributions mainly in two published papers.

- Firstly, the techniques in Chapter 3 are described in the paper "Monocular Surface Reconstruction using 3D Deformable Part Models" [Kinauer 2016], accepted at the workshop "Geometry Meets Deep Learning" of the European Conference on Computer Vision (ECCV) 2016. This workshop aims at recovering 3D geometry from image data in the context of deep neural networks. Our work seems well positioned in this workshop, recovering the 3D pose of objects from 2D images while exploiting rich deep learning features. We

depict results with this technique in Figure 1.2 (a)-(c), where, based on a 3-dimensional mesh (a), we extract a car's 3D shape (b) and (c).

- Secondly, the more advanced optimization techniques in Chapter 4 have been reviewed and accepted in the conference on Energy Minimization Methods in Computer Vision and Pattern Recognition 2017 (EMMCVPR). Under the title "Structured Output Prediction and Learning for Deep Monocular 3D Human Pose Estimation" [Kinauer 2017] we have presented the optimization of 3D graphical models using ADMM and Branch-and-Bound for human pose estimation. Figure 1.2 (d)-(g) illustrates an example input and output, where (d) is the input image and (e)-(g) the output, the estimated pose under different viewpoints.

In the paper "Robot-Robot Gesturing for Anchoring Representations" [Kontaxakis 2018], we employ part-based object detection methods in a more applied setting. The article aims at implicit robot-robot communication for object identity anchoring. To this end we translate the object detector of [Girshick 2012] to a practical application in a robotics environment. We defer the details to the Appendix A as this work does not provide a high degree of novelty, but applies known techniques, providing insights into their practicability for real-world scenarios. This work has been accepted as Regular Paper in the IEEE Transactions on Robotics (T-RO) journal.

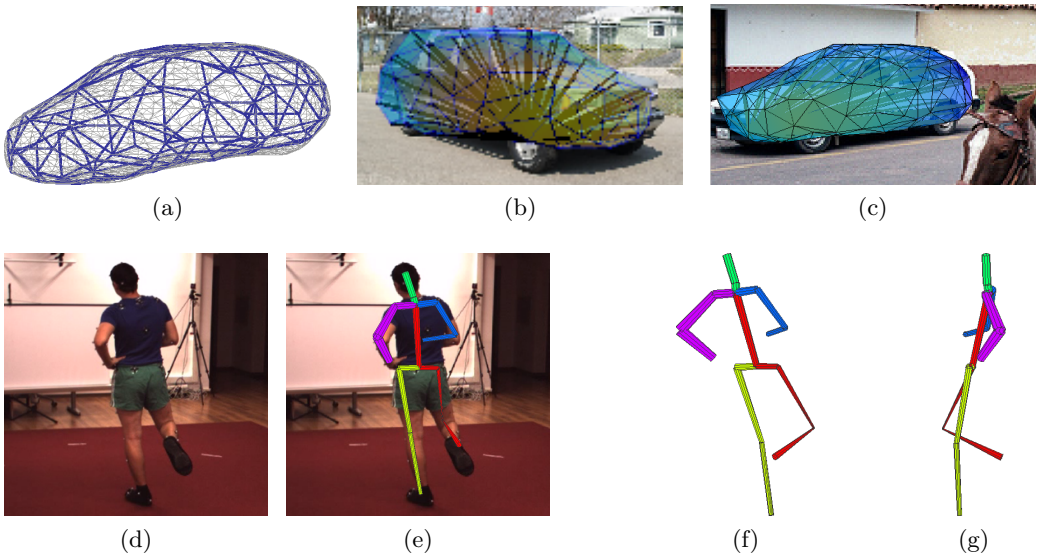


Figure 1.2: Summary of results in one figure. In (a)-(c) we illustrate results presented in Chapter 3 where (a) shows a 3-dimensional mesh of a car that adapts to the given image instance in (b) and (c) to recover the objects' pose. [Kinauer 2016] In (d)-(g) we demonstrate human pose estimation in 3D of Chapter 4. The input image (d) is given to our CNN which predicts the 3D human pose, illustrated in (e) in the image and in (f), (g) in new perspectives, respectively.

Table of Acronyms

DPM	**D**eformable **P**art **M**odel
DTBB	**D**ual-**T**ree **B**ranch-and-**B**ound
ADMM	**A**lternating **D**irection **M**ethod of **M**ultipliers
MRF	**M**arkov **R**andom **F**ield
DP	**D**ynamic **P**rogramming
GDT	**G**eneralized **D**istance **T**ransform
SVM	**S**upport **V**ector **M**achine
NMS	**N**on-**M**aximum **S**uppression
mAP	**m**ean **A**verage **P**recision
CNN	**C**onvolutional **N**eural **N**etwork
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
HOG	**H**istogram of **O**riented **G**radients
NRSfM	**N**on-**R**igid **S**tructure **f**rom **M**otion

Chapter 2

Background / Previous Work

Contents

2.1 Object Representation 12
2.1.1 Object Detection with Sliding Windows 14
2.1.2 Part-Based Models 18
2.1.3 Convolutional Neural Networks 22
2.2 Optimization Methods for Part-Based Models 27
2.2.1 Generalized Distance Transform with Dynamic Programming 27
2.2.2 Cascades 30
2.2.3 Dual-Tree Branch-and-Bound 31
2.3 Model Learning 35
2.3.1 Support Vector Machines (SVMs) 35
2.3.2 Structured SVM 36
2.3.3 Latent Variable SVM 38
2.4 Outlook 39

This chapter provides the basis for the contributions of this thesis, and covers problem aspects related to representation, optimization and learning with part-based models. The chapter is divided into three parts:

In the **first part** we introduce a basic technique for object detection, the *sliding window* detector. We refine the object representation by introducing object parts, following the *Deformable Part Model* paradigm. This allows for object and human pose estimation. We describe how results can be improved by employing *Convolutional Neural Networks* to replace the classical hand-crafted image representations. Additionally we highlight that a Deformable Part Model can be simulated by a Convolutional Neural Network and that both concepts blend together naturally.

In the **second part** we present the optimization problem that arises from Deformable Part Models and we discuss optimization algorithms for this specific problem class. We examine a common *dynamic programming* approach that is efficiently implemented with *generalized distance transforms*. The complexity of this problem is originally quadratic but the approach with dynamic programming and generalized distance transforms is linear in the number of pixels. Complementary to this, we make known the optimization with *Cascades*. Finally we introduce the *Dual-Tree Branch-and-Bound* algorithm which reduces the best-case complexity to

logarithmic in the number of pixels. We build on this optimization algorithm in the subsequent chapters.

In the **third part** we discuss learning algorithms for Deformable Part Models. We begin with standard *support vector machines* for binary classification, from where we advance to *structured support vector machines*. These serve in training the Deformable Part Models. If there are no part-level annotations, *latent variable support vector machines* are an effective tool to train DPMs. Finally, we show how a CNN is trained with stochastic gradient descent.

2.1 Object Representation

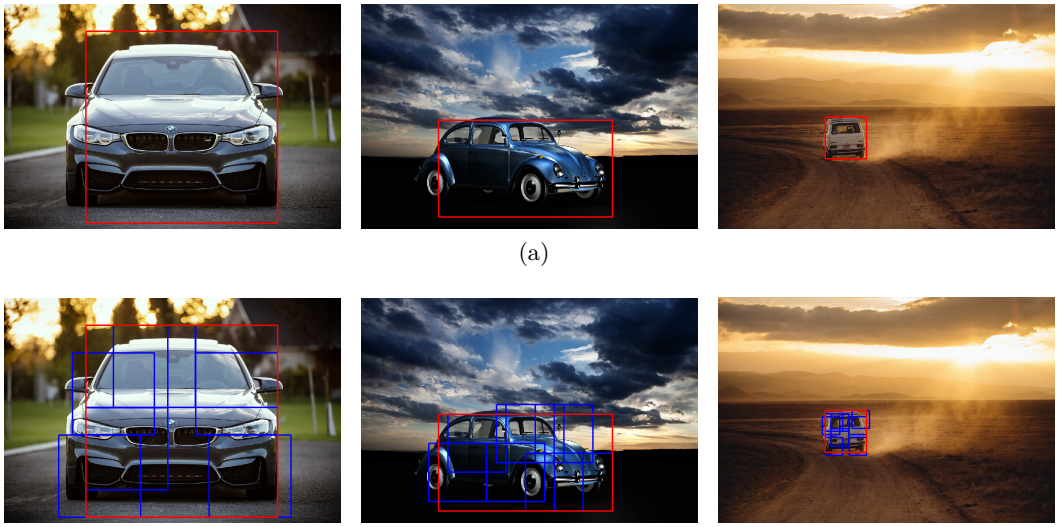
We structure the first part following the progression from simple towards more challenging and complex problems, extending object detection along different axes towards the problem of deformable object pose estimation. This sets the stage for the main goal of this thesis which is to advance object and human pose estimation in 3D.

Object detection is the task of detecting the position of an object in a given image. As illustrated in Figure 2.1, this is a challenging task: a "car"-detector should be able to detect all kinds of cars, from a middle-class limousine, over a vintage car to a minivan (Figure 2.1, (a), (b) and (c), respectively). Furthermore we observe cars of varying colors, viewpoints and scales. Although cars exhibit large visual variability, they all belong to the same object class. One classic approach is to capture an object's appearance with a rigid object model based on image features that are roughly invariant to appearance variability [Dalal 2005]. Then the location of an object matching the model is determined by the "sliding window" algorithm (Figure 2.2a).

(a) (b) (c)

Figure 2.1: Instances of the "car" object class. A car detector should detect the cars in all three images. The instances exhibit variation in the car's model, color, scale, the camera's viewpoint and the background.

We describe two different directions to improve upon this: (i) by obtaining more detailed information about the object, namely the object's geometry and pose, and (ii) by using better features to increase accuracy and robustness. The former has been shown to improve detection performance slightly over a simpler one-component



(a)

(b)

Figure 2.2: Top row: results of a car detector. Bottom row: results of a car detector with parts.
We point out that we have created this and further examples of object detection in Section 2.1 by running the object detector of [Girshick 2012]. Where needed, we have adapted the models, for example by removing object parts in the context of rigid object models.

model [Divvala 2012, Felzenszwalb 2008], but we emphasize that retrieving the object's pose goes beyond object detection and instead further describes the shape of the object instance. The latter is achieved through deep learning and has received great attention in many fields of computer vision research in recent years. In this thesis we profit from these advances, but the development of these techniques is not the focus of this thesis. Instead we focus on the optimization problems in context with part-based models.

(i) Object Pose The results shown in Figure 2.2a may suffice for estimating the location of the object, but we may want to extract more detailed information about the object in the image, namely the object's geometry and pose. Therefore the object can not be treated as a big rectangle, as shown in Figure 2.2a. Instead we divide the object into smaller entities, introducing object parts, that together make up a more thorough object description (see Figure 2.2b). This gives more comprehensive information about the object's geometry and pose which is useful in many practical applications. As consequence the optimization problem becomes considerably more complex as combinations of parts have to be taken into account. We consider efficient optimization methods for the resulting optimization problem in Section 2.2.

(ii) Improved Features What we described so far is the state of the art for object detection with part-based models in 2010. In subsequent years, Convolutional Neural Networks improved object detection and pose estimation results [Girshick 2014, Wei 2016, Cao 2016]. Convolutional Neural Networks learn to compute features optimized for a given task and training data. The object models presented so far are largely agnostic to the features that specify the resulting optimization problem. Consequently, Deformable Part Models are improved by learned features [Savalle 2014, Ranjan 2015, Ouyang 2015, Girshick 2015b].

Moreover, Deformable Part Models can be expressed as layers of Convolutional Neural Networks [Girshick 2015b]. This allows for a natural integration of Deformable Part Models and Convolutional Neural Networks. We use the combination of both approaches throughout this thesis.

Having outlined the two main axes related to object representation, we now move in more detail from object detection with sliding windows to object pose estimation.

2.1.1 Object Detection with Sliding Windows

The goal object detection is to find an object of a certain object class in an image and describe it by a tightly enclosing bounding box. As an example we show in Figure 2.3 a street image and apply an object detector for "bicycle", "person" and "car" and show the respective results.

To simplify the task of object detection, we think of an image not of a grid of color values, but operate on an intermediate representation, namely features. Working directly with the colored pixels of an image poses a number of challenges, for example sensitivity to lighting conditions, color, shifts, rotations, noise and distortions. Thus a variety of feature descriptors, hereafter just called features, have been developed in the interest of abstracting the image from pixelwise color values to a more robust representation, for example [Harris 1988, Lowe 2004, Swain 1991, Dalal 2005, Viola 2001, Tola 2010]. Most features are calculated based on gradients which reduces the effect of lighting and coloring in the image. Earlier features [Harris 1988, Lowe 2004] have been used to determine interest points, like corners, of an image. These sparse points are useful for image stitching and instance detection [Hartley 2003, Szeliski 2006]. Features like Histogram of Oriented Gradients (HOG) [Dalal 2005] or Daisy features [Tola 2010] are computed densely over the image. They form an image representation designed to be robust to illumination and color changes, small offsets and slight rotation. The exact choice of features is irrelevant for the methods presented here, so we will generically denote a stacked feature vector as $f(\xi)$. The vector f is formed by stacking the features that are extracted around image coordinate, ξ, where the coordinate ξ lies within an input image I.

Scale variation causes further difficulty because objects can appear at different sizes and resolutions in the image. Consequently the features covering the object diverge from one scale to another. To deal with the effect

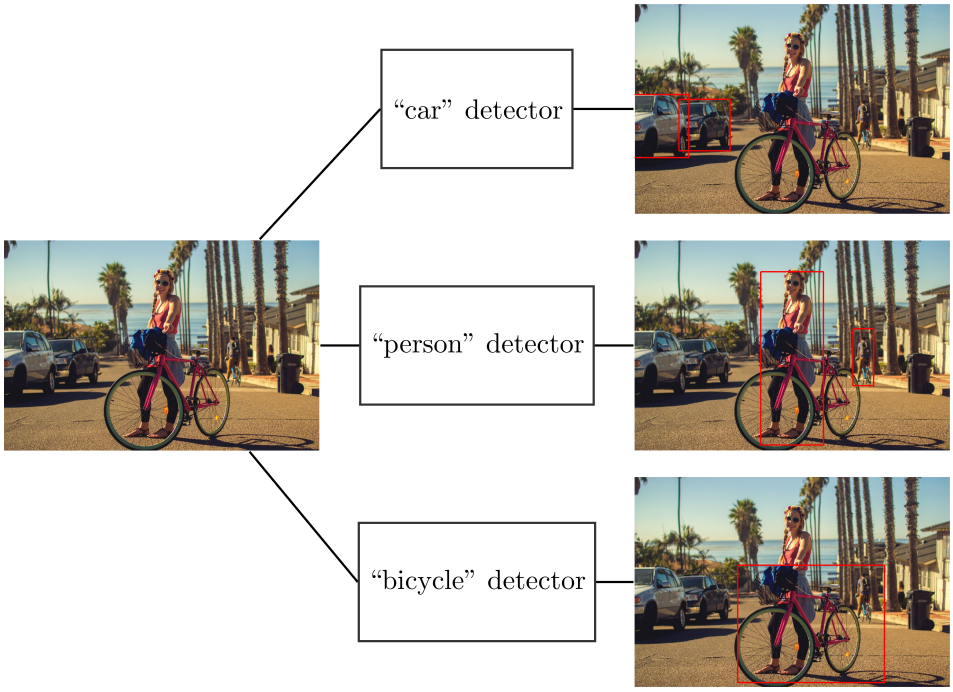


Figure 2.3: Object detector example. We apply a “bicycle”, a “person” and a “car” detector. The detections are indicated as red bounding boxes.

of scale variation in the image, feature pyramids are an established technique ([Shneier 1980, Adelson 1984]). Feature pyramids are computed by iteratively downsampling the image and computing features, constituting a multi-scale pyramid of features [Felzenszwalb 2010b, Dalal 2005, Dollár 2009, He 2014, Ren 2017]. The advantage of feature pyramids is that instead of having different object detectors for different scales, the same object detector is run against every pyramid level, essentially removing scale variation from the problem. We illustrate a HOG feature pyramid for the previous example image in Figure 2.4, as computed by [Girshick 2012], and demonstrate its use with the sliding window object detector (Figure 2.6).

2.1.1.1 Sliding Windows Detection

A discriminatively trained object model provides a scoring function for the presence of an object. The object detector of [Dalal 2005] compares an image patch with an object model by computing the inner product between the patch's features f and the model weight vector w:

$$S = \langle f, w \rangle \tag{2.1}$$

If the score, S, is greater than the model-specific threshold, θ, then an object is detected. Both model-specific parameters constitute the model, $M := (w, \theta)$. By means of the weight vector, w, the model defines the features that characterize an object of the wanted class. We can visualize the weight vector, w, by showing

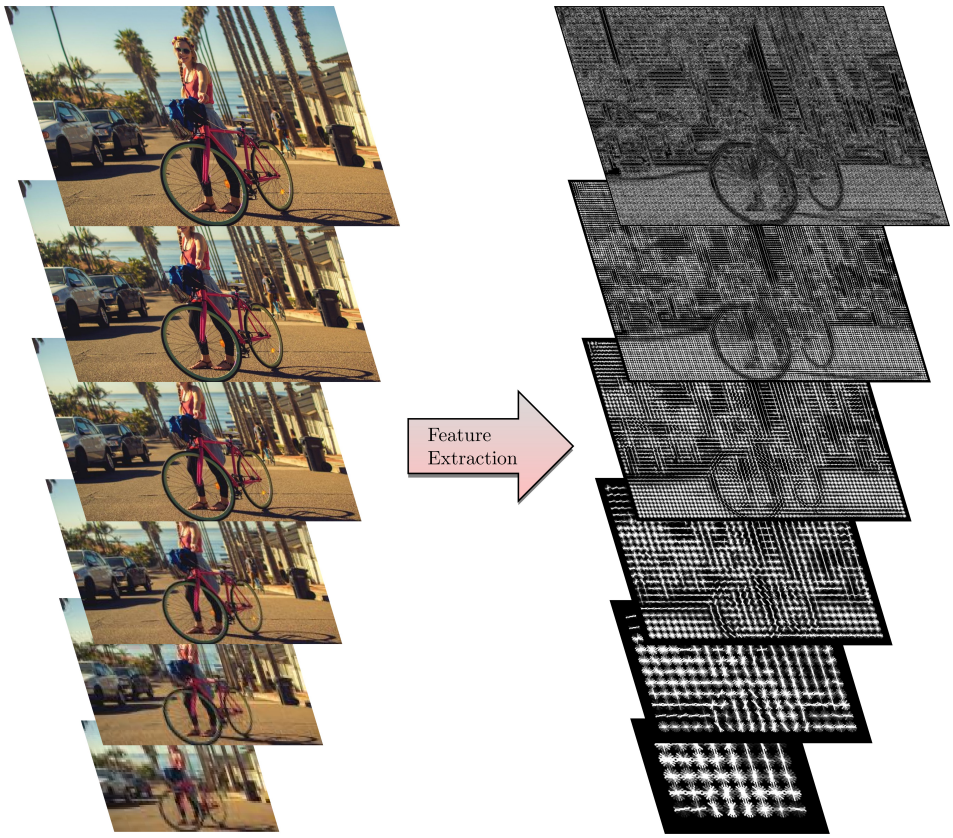
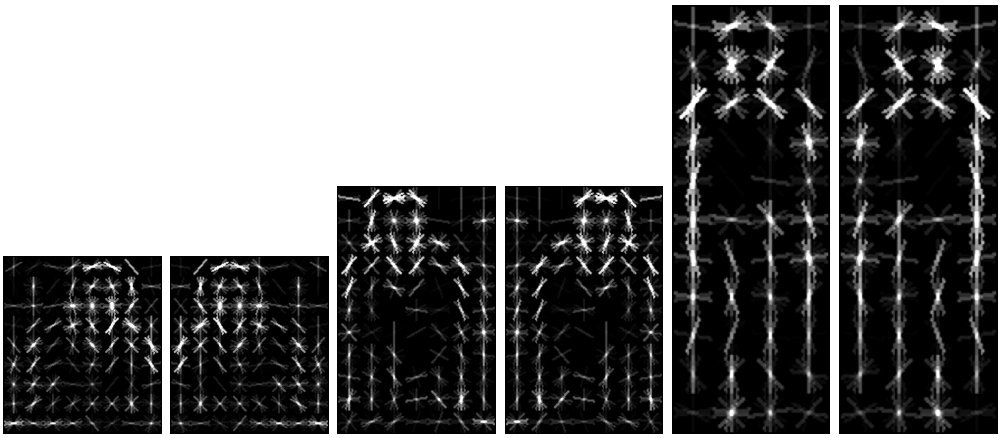


Figure 2.4: HOG-feature pyramid for an example image. An object detector is run against all pyramid levels, removing the need of training different object detectors for different scales. We depict 6 pyramid levels of HOG features, computed by successively downscaling the input image and extracting the features. As a result, at different pyramid levels different levels of details stand out.

the features that reach the highest score, see Figure 2.5 (a) to (f) for examples of a "person" model. Such a model, M, including the threshold parameter, can be learned from training data with support vector machines (SVMs) [Steinwart 2008]. Learning with support vector machines is shortly presented in Section 2.3.1. For clarity in our examples, we simplified the "person" model of [Girshick 2012] by removing object parts, yielding essentially the model of [Dalal 2005]. Models with parts are treated in Section 2.1.2.

Instance and viewpoint variation can be handled with mixture models. For example, an image of the same car from behind or from the side may look very different. Multiple mixture models enrich object models to better capture visual variability of real-world objects [Weber 2000, Felzenszwalb 2010b, Schneiderman 2000, Sheikh 2005, McKenna 1999, Moghaddam 1995]. A multiple mixture model, MM, consists of a number of K mixture models, $MM := \{M_1, M_2, ..., M_K\}$, where $M_1, ..., M_K$ are each models, for instance $M_i = (w_i, \theta_i)$. To detect all objects of one class in an image, the corresponding object detector is run once for each of its

mixtures $M_1, ..., M_K$ and the resulting detections are accumulated.



(a) M_1 (b) M_2 (c) M_3 (d) M_4 (e) M_5 (f) M_6

Figure 2.5: A "person" model with 6 mixture models. We can identify the rough head shape and varying portions of the human body. The second mixture model is the mirrored mixture model of the first one, similarly are the fourth and sixth ones mirrored mixture models of the third and fifth one.

Based on the described object model, we can detect objects at one specific location in an image. To detect objects everywhere in the image, the detector needs to be run for all possible patches in the image. This is called the "sliding window" approach: (i) An image patch is selected, (ii) features are extracted within the selected image patch and (iii) the score for object to be within the patch is calculated according to Equation 2.1. This sequence is repeated with varying image patches until all possible image patches have been covered. It resembles looking at the input image through a window that slides across the image, thus the term "sliding windows". The window leaves free only a patch that matches the model's size at a time.

We demonstrate this algorithm in Figure 2.6 with the person model M_6 of Figure 2.5f. First, the image is represented as a HOG feature pyramid (first column). Second, a 11×4 window of features, matching M_6, is cut out and its score is computed according to Equation 2.1 with weight vector w of model M_6. This yields one pixel of the heat map (second column). Red stands for high scores, blue for low scores. This step is repeated with a shifted window until the heat map is completed. Third, the scores are thresholded with the threshold parameter θ of M_6. All windows that correspond to scores above the threshold are shown in red in the last column.

Non-maximum suppression (NMS) is a greedy algorithm that removes redundant candidate solutions that overlap strongly with other candidates. This is often necessary because the close vicinity of detections also receives a high detection score. This is also true between neighboring feature pyramid levels. In Figure 2.7a we depict all detections obtained by a person detector above a certain

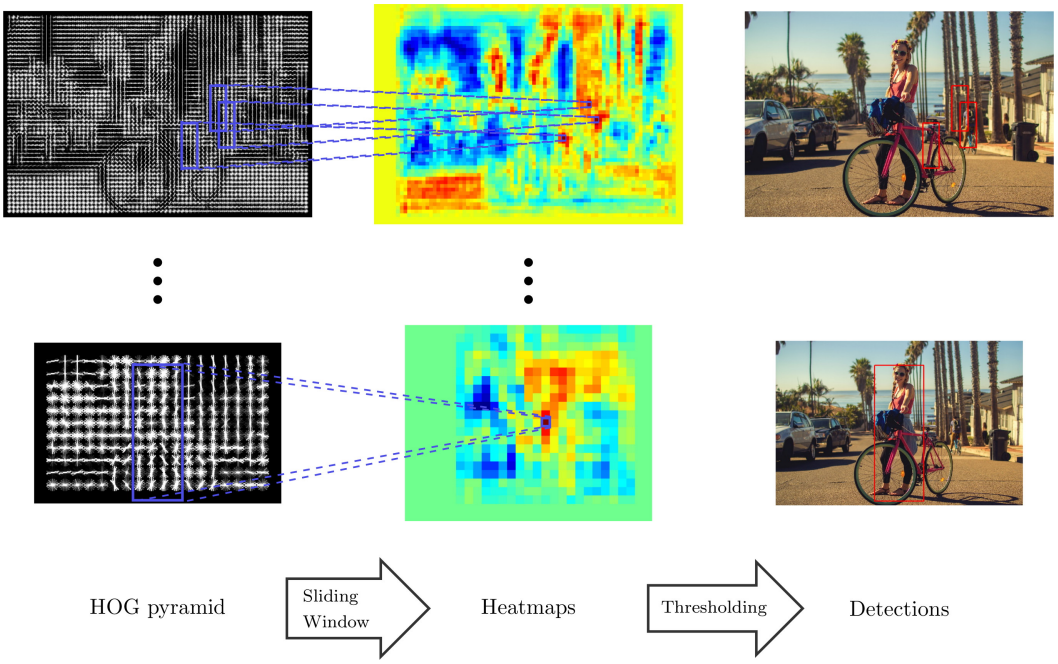


Figure 2.6: Illustration of a sliding window object detector. A window of the size of the model, M_6 of Figure 2.5f, is slid over the feature pyramid levels and the corresponding score according to Equation 2.1 is calculated. The resulting scores are visualized as a heat-map, where red corresponds to high scores and blue to low scores. In the last column we show the bounding boxes that correspond to scores over a certain threshold value. A detection in a high-resolution pyramid level corresponds to small bounding box, and vice versa.

threshold. We count 10 detections in the image, but they correspond onlsentially to 2 true and 2 false-positive detections. To eliminate heavily overlapping bounding boxes, all candidate bounding boxes are sorted by their detection score. Then NMS steps through the list and adds the current element to the solution set if the solution set does not already contain an element with which it overlaps strongly. In [Girshick 2012, Girshick 2014, Zitnick 2014, Bourdev 2010, Liu 2016] the overlap is quantified by the "intersection over union" (IoU) measure, computed as $IoU = (A \cap B)/(A \cup B)$ for candidates A and B. As result, selected detections prevail and neighboring bounding boxes are discarded, as shown in Figure 2.7b.

2.1.2 Part-Based Models

The techniques presented so far allow to detect objects and to mark them by a rectangular bounding box. Now we go a step further and extract the locations of object parts, which we jointly refer to as the object's pose. This serves two purposes: Firstly, the object's pose is valuable information for real-world applications. Secondly, a richer model can improve detection performance; the work of [Felzenszwalb 2008] shows an improvement of average precision from 0.24 to 0.34

(a) Detection results for "person". Several redundant bounding boxes clutter the detections.

(b) Final detections after applying NMS. Strongly overlapping bounding boxes have been removed.

Figure 2.7: Example of non-maximum suppression. In the image on the left there are several almost redundant detections. In the image on the right, strongly overlapping bounding boxes have been removed using non-maximum suppression (NMS).

by moving from rigid models (Section 2.1.1.1) to part-based models.

In this chapter we describe the Deformable Part Model (DPM) following the sequence of works [Felzenszwalb 2005, Felzenszwalb 2008, Felzenszwalb 2010b], while the idea has been developed in a long line of research [Fischler 1973, Ioffe 2001, Fergus 2003, Felzenszwalb 2005, Crandall 2005, Fergus 2005, Burl 1998, Yuille 1992]. More current results with DPMs are discussed in Section 3.1.

2.1.2.1 The Score Function of a DPM

Formally, a DPM is a graphical model [Koller 2009] where object parts are nodes, Ξ, in an undirected graph, $G = (\Xi, E)$, where the edges, E, model pairwise constraints between object parts. Let the object consist of N parts, then its associated graph has N nodes, $\Xi := \{\xi_1, ..., \xi_N\}$. A variable ξ_i specifies the location of the i-th object part and has a sample space equal to the image's coordinate space.

Based on this structure, the DPM score function is defined as

$$S(\Xi) = \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{(i,j)\in E} \mathcal{P}_{(i,j)}(\xi_i, \xi_j), \tag{2.2}$$

consisting of unary terms, $\mathcal{U}_i$, for each part i and pairwise terms, $\mathcal{P}_{i,j}$, for each edge $(i, j) \in E$. The object pose, Ξ^*, is found as the argmax of the score function:

$$\Xi^* = \arg\max_{\Xi} S(\Xi). \tag{2.3}$$

The unary term for the i-th part,

$$\mathcal{U}_i(\xi_i) = \langle f(\xi_i), w_i \rangle, \tag{2.4}$$

is the inner product of the feature vector $f(\xi_i)$, computed around the presumed location for part i, ξ_i, and the model's weight vector for the i-th part, w_i. The features, $f(\xi_i)$, may be extracted from distinct levels of the feature pyramid and therefore have a specific resolution in image space.

The pairwise term,

$$\mathcal{P}_{i,j}(\xi_i, \xi_j) \coloneqq -(\xi_i - \xi_j - \mu_{i,j})^T C_{i,j}(\xi_i - \xi_j - \mu_{i,j}), \quad (i,j) \in E, \tag{2.5}$$

constrains the relative positions of part i and part j. This term incurs a quadratic penalty to the score if the difference, $\xi_i - \xi_j$, is not equal to the nominal offset $\mu_{i,j} \in \mathbb{R}^2$. This penalty is weighted by the diagonal precision matrix $C_{i,j}$. The pairwise term acts as a spring between pairs of parts that permits small deviations from a nominal position, but prevents large deformations.

In [Felzenszwalb 2010b], the set of edges forms a star-shaped graph such that every part is connected to a single central part that is identified with the object's position. Without loss of generality, we will always denote the "center" or "root" node with the index 1. This determines the set of edges to be $E = \{(1,j)|j \in \{2,...,N\}\}$. In distinction to the "center" node (index 1), the nodes with indices $\{2,...,N\}$ are referred to as part nodes. Thus the object's position, ξ_1, determines the maximum score of associated poses as sum over individual part contributions:

$$S(\xi_1) = \sum_{i=1}^{N} \max_{\xi_i} (\mathcal{U}_i(\xi_i) + \mathcal{P}_{i,1}(\xi_i, \xi_1)), \tag{2.6}$$

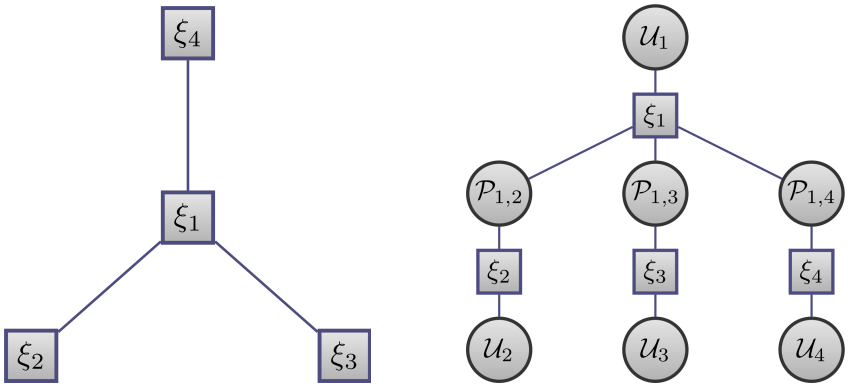
with $\mathcal{P}_{1,1}(\xi_1, \xi_1) = 0$. We make use of this star-shaped graph structure in this thesis, until we loosen the restraints again in Chapter 4 and allow loopy graphs.

We visualize an example DPM with 4 nodes in Figure 2.8. On the left we show a very simple star-shaped graph where part nodes, ξ_2, ξ_3 and ξ_4, are connected to the root node, ξ_1. On the right we visualize this DPM as factor graph, displaying unary and pairwise potentials as factors in connection with the part positions. The variables ξ_1, ξ_2, ξ_3 and ξ_4 are framed in squares and the factors are depicted as circles.

We note that this graphical model can be interpreted probabilistically as Markov Random Field (MRF). In this context the variables determining the pose, $\Xi = \{\xi_1, ..., \xi_N\}$, are random variables. The edges, E, describe conditional dependencies, specifically that variables, ξ_i and ξ_j, are conditionally independent if there does not exist an edge, $(i,j) \notin E$ (Markov property). Then, as stated by the Hammersley-Clifford Theorem [Hammersley 1971], the MRF corresponds a Gibbs distribution:

$$\mathbb{\$}(\Xi) = \frac{1}{Z} \prod_{i=1}^{N} e^{\mathcal{U}_i(\xi_i)} * \prod_{(i,j)\in E} e^{\mathcal{P}_{(i,j)}(\xi_i, \xi_j)}. \tag{2.7}$$

We obtain the score function in Equation 2.2 by taking the logarithm of the probability distribution, $\mathbb{\$}$. We note that the maximum a posteriori probability (MAP)



(a) Graph of a star-shaped DPM with 4 parts.

(b) Factor graph of a star-shaped DPM with 4 parts. Random variables are depicted as squares, factors as circles.

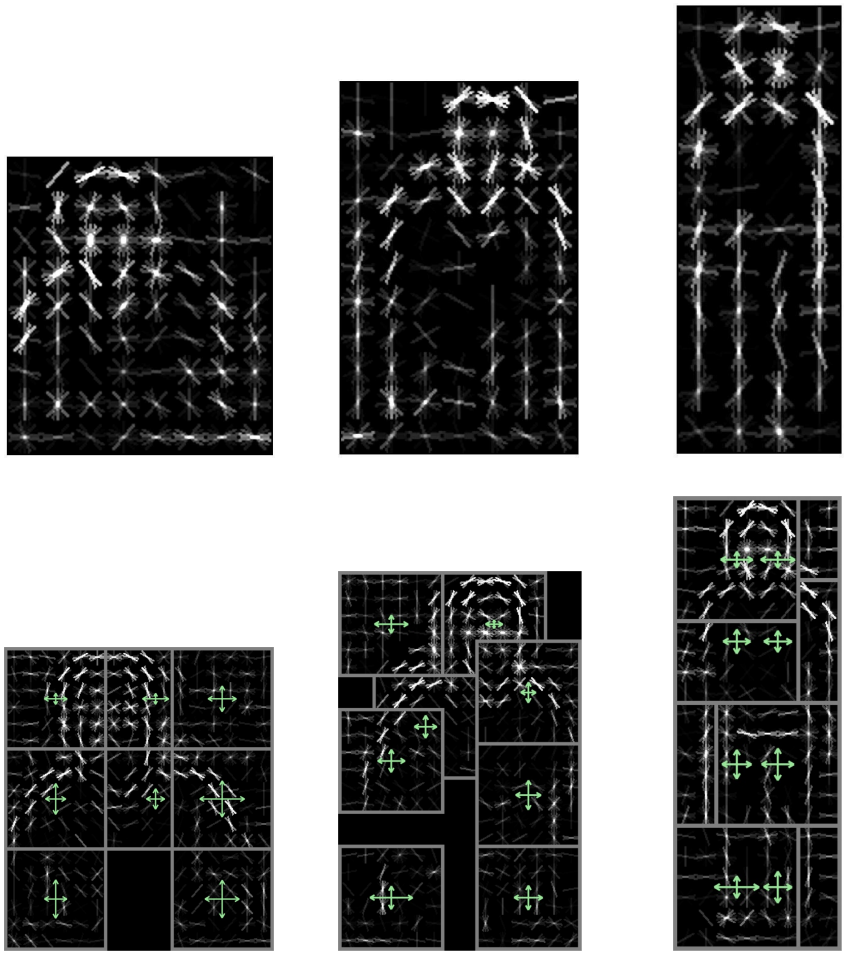
Figure 2.8: Example DPM and its factor graph.

estimate of this distribution is equivalent to the argmax of the score function 2.3. The normalization factor, $Z = \sum_{\Omega^N} \left(\prod_{i=1}^{N} e^{\mathcal{U}_i(\xi_i)} * \prod_{(i,j)\in E} e^{\mathcal{P}_{(i,j)}(\xi_i,\xi_j)} \right)$, with Ω^N being the pose sample space consisting of N part positions, is an additive constant under the logarithm and therefore irrelevant for the optimization.

2.1.2.2 DPM - Visualizations

A visualization of a "person" DPM is shown in Figure 2.9. The top row shows the "center"-part of the "person" model, identical to the previous, rigid "person" model (Figure 2.5). The other parts are visualized in the bottom row. Every part j is placed relative to the center part according to the pairwise offsets $\mu_{1,j}$. We explicitly indicate the flexibility of the corresponding pairwise term by yellow arrows. The size of an arrow cross relates to the inverse of $C_{1,j}$, such that a long arrow conveys a flexible pairwise relation. We point out that this model contains parts that operate with features of different resolutions. [Felzenszwalb 2008] argues that a low resolution center part covers the general appearance of an object, whereas the other parts capture the object's finer details.

The relative part positions are not rigid, as illustrated in Figure 2.10. Both detected persons are assigned to the mixture on the right of Figure 2.9. We observe that the arrangement of parts (red rectangles) adapts to the image, resulting in two different poses between both persons. The blue rectangle corresponds to the center part to which all other parts are connected via pairwise terms. Due to its lower resolution, the center part covers a larger area than the other parts.



(a) (b) (c)

Figure 2.9: Visualization of a "person" DPM. The top row shows the root-filter, w_1, that by itself offers a rigid model identical to the model in Figure 2.5. The bottom row shows the part filters, $w_2, ..., w_N$, in their spatial configuration. The green arrows illustrate the flexibility of part positions, $1/C_{1,2}, ..., 1/C_{1,N}$, of the corresponding parts, such that a long arrow corresponds to a small penalty for a deformation in this direction. Each column represents one mixture model. We show 3 out of 6 mixture models, since the other 3 are mirrored equivalents.

2.1.3 Convolutional Neural Networks

The DPM of [Felzenszwalb 2010b] set the state of the art in object detection on the PASCAL VOC benchmark [Everingham 2010] at the time of its publication. In the following years, Convolutional Neural Networks established themselves to be one of the most successful methods for object detection [Ren 2015, Sermanet 2013], pose estimation [Newell 2016, Toshev 2014, Tompson 2014] and a variety of other computer vision tasks [Simonyan 2014, Long 2015, Chen 2016b, Kokkinos 2016]. This data-driven approach has been facilitated by the growing computing capabilities

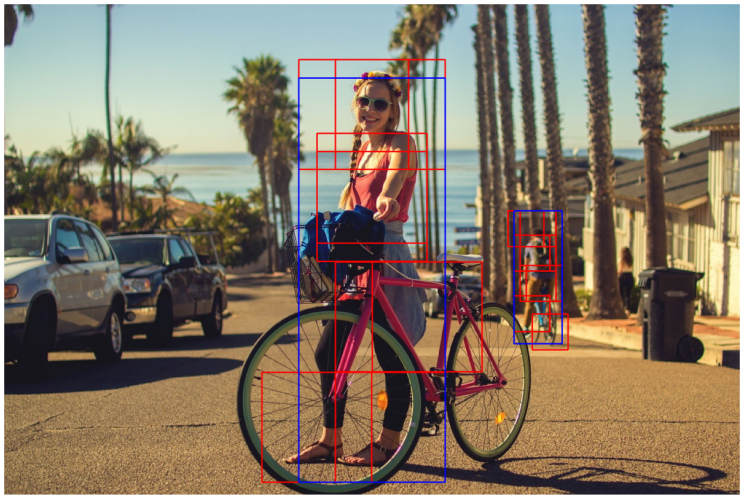


Figure 2.10: Object part detection with DPMs. The blue rectangle stands for the low-resolution center part, the red rectangles indicate the other parts. Two people are detected with the same mixture model, but the flexible pairwise term allows varying poses.

of modern graphics cards and large amounts of available training data. It turns out that hand-crafted features like HOG are the bottleneck for object detection performance and the results are improved by learned features [Girshick 2015b]. In this chapter we describe briefly Convolutional Neural Networks, leading to features learned from data, and then turn to the integration of DPMs with Convolutional Neural Networks.

2.1.3.1 Convolutional Architecture

A Convolutional Neural Network (CNN) is a feedforward neural network that contains convolutional layers [LeCun 1998]. In a convolutional layer, the neurons are connected to a neighborhood of neurons in the layer below. In contrast a multilayer perceptron employs fully-connected layers where every neuron of one layer is connected to all neurons in the previous layer. A peculiarity of convolutional layers is that edge weights are shared among nodes of the same layer, more precisely, the same convolution parameters are used for each neuron. Due to this weight-sharing, the convolutional layer contains fewer parameters than a comparable fully-connected layer. This reduces over-fitting and improves generalization of CNNs over fully connected networks [LeCun 1989]. Additionally, the emphasis on neighboring neurons in the underlying layers promotes locality and shared weights drive the computation of shift-invariant features [LeCun 1995].

We visualize a convolutional and a fully-connected layer in Figure 2.11. “Layer 1” is a convolutional layer, each neuron is connected to 3 neurons in “Layer 0”. The edge weights are shared among this layer: The same set of 3 weights is used for all neurons in “Layer 1”. The edges are colored accordingly in red, green

and blue, indicating shared parameters. The fully-connected layer simply connects every node of "Layer 2" with every node of "Layer 1" underneath.

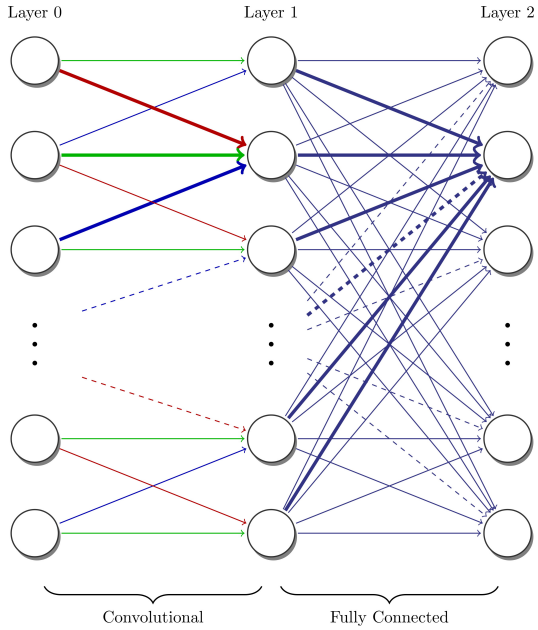


Figure 2.11: Neural Network with a convolutional and a fully connected layer. In a convolutional layer, neurons of the layer are connected to overlapping subsets of neurons in the layer below. Edge weights are shared in this layer. We visualize this by coloring edges accordingly in red, green and blue. In fully connected layers, all neurons of the layer are connected to all neurons of the layer below. Here, parameters are not shared. For reasons of clarity we single out the edges to a single node in both layers.

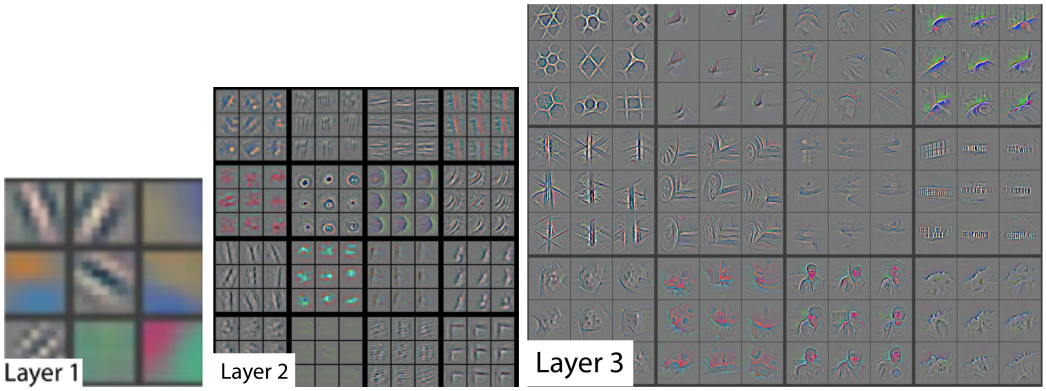
The work by Krizhevsky et al. [Krizhevsky 2012] made CNNs popular by obtaining state of the art results on image classification by a wide margin (15.4% top 5 test error rate over 26.2%). Two years later, Girshick et al. [Girshick 2014] transferred this success to the task of object detection, followed by improved versions [Girshick 2015a, Ren 2015]. The idea behind this work is to feed the network a large number of region proposals and select the best-scoring proposals via NMS.

2.1.3.2 Dense Feature Maps

A CNN does not only classify images, but computes, as intermediate results, image features in the inner layers of the network. The complexity of features computed in a CNN depends on the number of layers up to the feature map and size of the receptive field. The receptive field is the neighborhood to which every neuron is connected in layers below. As an example, assume that every node in the layers of a CNN is connected to a 3×3 neighborhood, as in [Simonyan 2014]. Thus the receptive field of a neuron in the first layer is 3×3. Then the receptive field of a neuron in the second layer is 5×5 in relation to the zero-th layer, 7×7 in the next

layer and so forth. This allows the CNN to learn increasingly complex features on an increasingly large support in the input image. These features are highly discriminative and show a large visual variability.

This is illustrated in the work on ZF-nets [Zeiler 2014] who provide insight into what the CNN is learning by illustrating the features of various layers. Figure 2.12 shows the features computed by the first three layers of the ZF-network trained on the ImageNet dataset for image classification. While the first layer contains some basic shapes like lines or pairs of lines in certain directions, the second layer contains more complex shapes like circles and corners. The third layer is activated on patterns and shapes that look like wheels of a car, for example.



(a) (b) (c)

Figure 2.12: Visualization of features of the ZF-net [Zeiler 2014]. The visualizations are created using deconvolutions of individual features in the first, second and third layer.

We turn now to how CNNs can be used in conjunction with DPMs. The unary term, as previously introduced for the DPM formalism, is computed as the inner product between feature vector, $f(\xi)$, and model specific weight vector, w, (Equation 2.4). Such a feature vector, $f(\xi)$, can be obtained as the activation of the neuron at the indicated position, ξ, in a certain layer. These feature vectors form dense feature maps that can be used to efficiently compute dense unary potentials.

In [Girshick 2015b, Savalle 2014], convolutional features are used to compute a pyramid of dense feature maps. Some feature maps are shown in Figure 2.13. The first column shows the input image, the other columns contain feature pyramid levels. Figure 2.13 compares HOG features (first row) with convolutional features, more precisely the activation of one channel of the 5-th layer of their CNN (other rows). The chosen channel corresponds to a "head", an important feature for labels that involve humans in the original classification task [Russakovsky 2015]. A high activation is displayed in white, a low activation is shown as black. The activations are scale sensitive, peaking at level 6 for the face image.

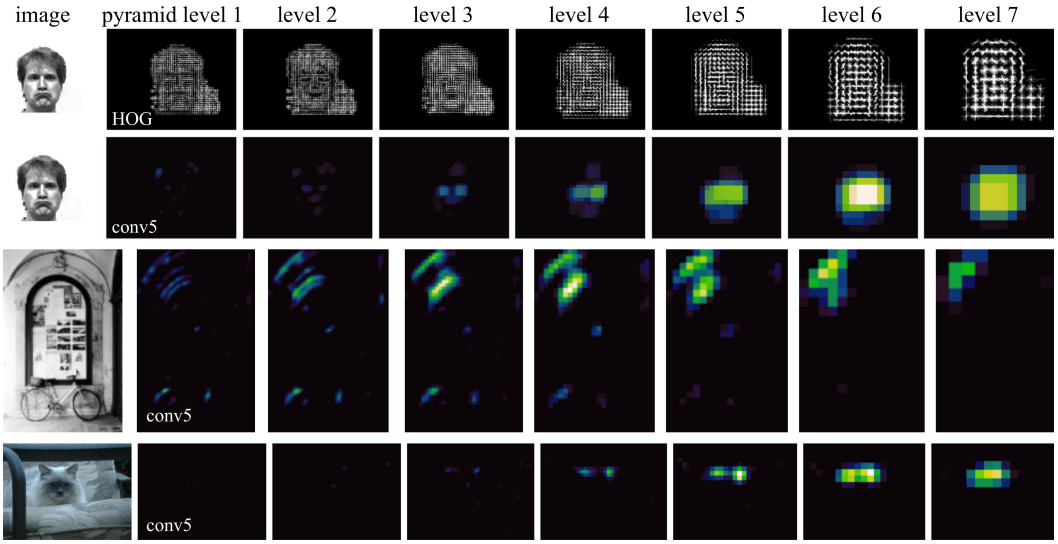


Figure 2.13: Comparison of HOG features with $conv_5$ activations over several scales [Girshick 2015b]. The first column shows the input image, the remaining columns correspond to features extracted at different scales of the image pyramid. The first row displays the HOG feature pyramid, similar to Figure 2.4. In rows 2 to 4, the activations of 1 channel (out of 256) of the 5-th convolutional layer are visualized. The network is trained on the ImageNet dataset for scene classification. The chosen channel corresponds to the "head" label.

2.1.3.3 Part-Based Models and Convolutional Neural Networks

So far we have introduced the DPM as a graphical model approach for object detection and pose estimation. Furthermore we have briefly characterized CNNs that can be used for object detection and who calculate features learned from training data. Girshick et al. [Girshick 2015b] argue that both ideas blend naturally and that in fact a DPM can be expressed as CNN.

In order to prove that DPMs are CNNs, [Girshick 2015b] propose a CNN that simulates the optimization of a DPM. They propose a CNN that consists of two phases: (i) computation of dense feature maps over an image pyramid (see Figure 2.13) and (ii) DPM-style optimization by unrolling the steps of the optimization into layers of a convolutional network architecture. This results in a unified CNN that detects object with DPMs, based on CNN features.

Beyond the insight that DPMs are CNNs, this work confirms that the combination of DPMs and CNN features improves performance for object detection. [Girshick 2015b] reports a detection performance of 33.7% mAP with HOG features and 44.4% mAP with CNN features on the PASCAL VOC 2007 benchmark [Everingham a]. Similarly, the work of [Savalle 2014] compares DPMs with HOG features and DPMs with features of a very similar CNN and reports improvements for detection (48.2% mAP on the same benchmark). In this thesis, we apply the same idea in Chapter 3 to obtain unary potentials based on convolutional features.

2.2 Optimization Methods for Part-Based Models

In the second part of this introductory chapter, we show how to optimize efficiently star-shaped DPMs like the ones introduced in Section 2.1.2 in 2D. We will address this limitation in Chapter 4 and expand to loopy graph structures. Our objective here is to maximize the following score function, S, as introduced in Equation 2.2:

$$S(\Xi) = \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{(i,j)\in E} \mathcal{P}_{(i,j)}(\xi_i, \xi_j), \tag{2.8}$$

The problem is computationally challenging to solve due to the large size of the solution space. Denote the set of pixel coordinates within an image as Ω_{2D}, such that each variable can take on $|\Omega_{2D}|$ distinct values, in particular the number of pixels in the image. If the input image contains $|\Omega_{2D}|$ pixels, there are $|\Omega_{2D}|^N$ different object poses. However, in a star-shaped DPM, all parts are connected only to one common center part, rendering their contribution to the score function independent from each other, following the Markov property of the MRF. Thus the complexity of a sliding windows implementation is $\mathcal{O}(N|\Omega_{2D}|^2)$, quadratic in the size of the input image.

A variety of approaches have been proposed to efficiently solve a pairwise MRF of this type. We first focus on a dynamic programming approach with generalized distance transforms (Section 2.2.1), reducing the algorithm's complexity to linear in image size. We also consider a cascaded optimization (Section 2.2.2), which is a complementary acceleration technique. In the final part of this section, we introduce a Branch-and-Bound implementation that accelerates optimization to a near logarithmic runtime. This approach is central for the rest of this thesis.

2.2.1 Generalized Distance Transform with Dynamic Programming

[Felzenszwalb 2005] combines dynamic programming with the Generalized Distance Transform of [Felzenszwalb 2005] to an efficient inference procedure. This makes inference possible within about 10 seconds per image [Felzenszwalb 2010a] on a single CPU core and 2 seconds on a multi-core system [Felzenszwalb 2010b].

2.2.1.1 Dynamic Programming

Dynamic programming (DP) [Bellman 1952, Bellman 2015] is a general programming technique designed to reduce computational complexity by re-using already computed magnitudes. DP reduces the complexity of inference on loop-free graphical models from $\mathcal{O}(|\Omega_{2D}|^N)$ with a naive implementation to $\mathcal{O}(N|\Omega_{2D}|^2)$ with DP, where N is the number of nodes in the MRF.

DP was originally proposed to break down sequential problems, like computation of shortest paths, recursively into simpler partial problems. The application of dynamic programming to non-serial problems is referred to as the max-product

algorithm [Weiss 2001, Pearl 2014]. Since the term dynamic programming is used also in the context of tree-structured DPMs, we will omit the more precise term "max-product" in favor of "dynamic programming" to avoid confusion.

Assume a small-sized DPM that involves three object parts. For this example we view a chain-structured graph instead of a star-structured one. So we consider the score function S to be

$$S(\xi_1,\xi_2,\xi_3) = \mathcal{U}_1(\xi_1) + \mathcal{U}_2(\xi_2) + \mathcal{U}_3(\xi_3) + \mathcal{P}_{1,2}(\xi_1,\xi_2) + \mathcal{P}_{2,3}(\xi_2,\xi_3). \tag{2.9}$$

The task is to compute the set of variable values $(\xi_1,\xi_2,\xi_3)^T$ that maximizes $S(\xi_1,\xi_2,\xi_3)$. Using commutations, we obtain

$$\begin{aligned}
&\max_{\xi_1,\xi_2,\xi_3} S(\xi_1,\xi_2,\xi_3) = \\
&\quad = \max_{\xi_1}\max_{\xi_2}\max_{\xi_3}\Big(\mathcal{U}_1(\xi_1) + \mathcal{U}_2(\xi_2) + \mathcal{U}_3(\xi_3) + \mathcal{P}_{1,2}(\xi_1,\xi_2) + \mathcal{P}_{2,3}(\xi_2,\xi_3)\Big) && (2.10)\\
&\quad = \max_{\xi_1}\max_{\xi_2}\max_{\xi_3}\Big(\mathcal{U}_1(\xi_1) + \mathcal{P}_{1,2}(\xi_1,\xi_2) + \mathcal{U}_2(\xi_2) + \mathcal{P}_{2,3}(\xi_2,\xi_3) + \mathcal{U}_3(\xi_3)\Big) && (2.11)\\
&\quad = \max_{\xi_1}\max_{\xi_2}\Big(\mathcal{U}_1(\xi_1) + \mathcal{P}_{1,2}(\xi_1,\xi_2) + \mathcal{U}_2(\xi_2) + \underbrace{\max_{\xi_3}\left(\mathcal{P}_{2,3}(\xi_2,\xi_3) + \mathcal{U}_3(\xi_3)\right)}_{m_{3\to 2}(\xi_2)}\Big) && (2.12)\\
&\quad = \max_{\xi_1}\Big(\mathcal{U}_1(\xi_1) + \underbrace{\max_{\xi_2}\Big(\mathcal{U}_2(\xi_2) + \mathcal{P}_{1,2}(\xi_1,\xi_2) + m_{3\to 2}(\xi_2)\Big)}_{m_{2\to 1}(\xi_1)}\Big) && (2.13)\\
&\quad = \max_{\xi_1}\Big(\mathcal{U}_1(\xi_1) + m_{2\to 1}(\xi_1)\Big) && (2.14)
\end{aligned}$$

The commutability of sums and maximization allow us to move the max-operators further inside, creating smaller, nested optimization problems. The problem in Equation 2.10 can now be solved by recursively solving subproblems. First, the smallest problem, $m_{3\to 2}$, is solved, returning a function of ξ_2. Since the domain of ξ_2 is the set of pixels and therefore discrete, this function can be stored as array. Second, the second smallest problem, $m_{2\to 1}$, is solved, using the already computed sub-solution $m_{3\to 2}$. Again the resulting function of ξ_1 is stored as matrix. Finally the original problem can be solved as in Equation 2.14. By utilizing the already computed $m_{2\to 1}$, the problem has become considerably easier to solve. The gain in computational efficiency is explained by avoiding to re-compute partial results.

2.2.1.2 Generalized Distance Transform

The Generalized Distance Transform (GDT) [Felzenszwalb 2004] accelerates the computation of subproblems of the form $m_{j\to i}(\xi_i) = \max_{\xi_j} \mathcal{U}_j(\xi_j) + \mathcal{P}_{i,j}(\xi_i,\xi_j)$. DP

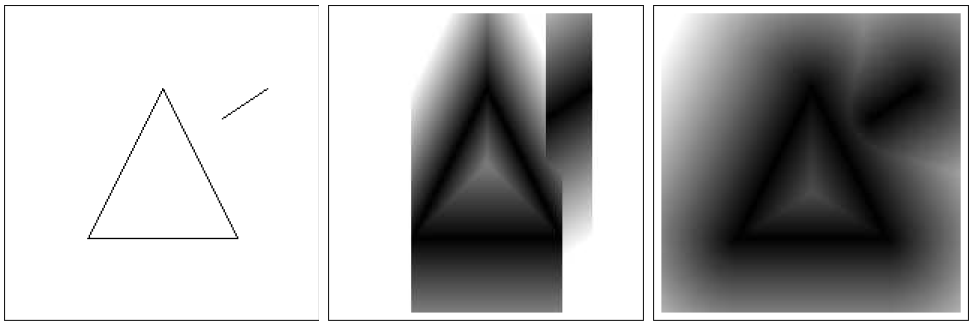
allows one to perform inference of a loop-free DPM by solving N subproblems of this type. A naive implementation of these subproblems runs in the order of $\mathcal{O}(|\Omega_{2D}|^2)$ each, with GDT they can be solved in $\mathcal{O}(|\Omega_{2D}|)$.

GDT accommodate to a more general class of problems, only requiring that the term $\mathcal{P}$ is a distance and that the unary term, $\mathcal{U}$, is embedded in a grid on which the distance is computed. In case the distance is separable over dimensions, like the pairwise term of the DPM (Equation 2.5), the distance transform can be computed as successive 1-dimensional operations:

$$m_{j\to i}(\xi_i) = \max_{\xi_j} \mathcal{U}_j(\xi_j) + C_{i,j_x}(\xi_{i_x} - \xi_{j_x} - \mu_{i,j_x})^2 + C_{i,j_y}(\xi_{i_y} - \xi_{j_y} - \mu_{i,j_y})^2 \quad (2.15)$$

$$= \max_{\xi_{j_y}} \big(\max_{\xi_{j_x}} (\mathcal{U}_{\xi_j}(\xi_j) + C_{i,j_x}(\xi_{i_x} - \xi_{j_x} - \mu_{i,j_x})^2) + C_{i,j_y}(\xi_{i_y} - \xi_{j_y} - \mu_{i,j_y})^2 \big). \quad (2.16)$$

The nominal offset $\mu_{i,j}$ and quadratic weight $C_{i,j}$, as in the definition of the pairwise term (Equation 2.5), is omitted here, but can be incorporated in the calculation of the parabolas. Equation 2.16 describes a 2-dimensional distance transform as concatenation of two 1-dimensional operations. Figure 2.14 visualizes this approach. From the input image in Figure 2.14a, showing an example unary term $\mathcal{U}$, first the GDT over columns is computed (Figure 2.14b) and in a second pass over rows the final result is obtained (Figure 2.14c).



(a) (b) (c)

Figure 2.14: Generalized Distance Transform (GDT) [Felzenszwalb 2004]. (a) an example input. (b) the result after the first pass over the vertical dimension. (c) the final result.

The GDT in 1D consists of two steps: (i) computing the lower envelope of parabolas anchored at the grid points and (ii) sampling of the envelope at the grid locations.

Step (i) A grid point $g \in \Omega_{2D}$ generates a parabola rooted at $(g, \mathcal{U}(g))$. The lower envelope is computed first for a single grid point and is updated incrementally with parabolas for all grid points. The lower envelope is maintained in a combinatorical data structure, keeping track in which interval a parabola is below all others.

Step (ii) The distance transform is computed by sampling the lower envelope at the grid points.

Both steps are computable in $\mathcal{O}(\Omega_{2D})$, resulting in a linear overall complexity.

2.2.2 Cascades

Cascades for object detection with DPMs [Felzenszwalb 2010a] accelerate inference by applying a series of simplified object classifiers, successively narrowing down the search space for subsequent, increasingly complex classifiers. [Felzenszwalb 2010a] report an average runtime of 10.1 seconds with GDT and DP and 0.313 seconds with Cascades for the *motorbike*-object class.

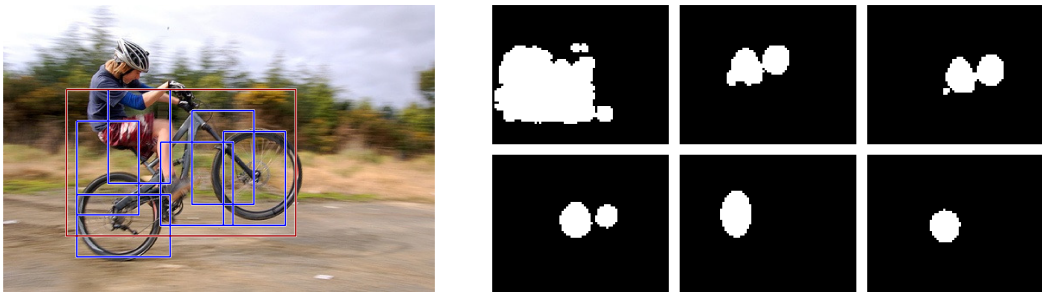
Evaluation of a complex classifier like DPM with many parts is considerably slower than a simple classifier with few or no moving parts. [Felzenszwalb 2010a] propose to apply a hierarchy of simpler classifiers in a sliding window fashion. Assume a DPM defined for a star-shaped graph $G = (V, E)$ with N nodes, including the center node. The hierarchy of classifiers consists of N models based on the subgraphs $G_1, ..., G_{N-1}, G_N$, with G_i containing the first i parts of the original model. Graph G_N corresponds to the original DPM.

A hypothesis is defined by the location of the center node; the part locations refer only to the center node. The location of each part, ξ_i, is determined independently, given the center node position, ξ_1, in order to maximize the part's contribution to the score (Equation 2.6. Each partial model, from G_1 to G_N, is applied successively to the image, generating a set of hypotheses as base for the next larger partial model. These hypotheses are pruned based on thresholds, t_τ, for each stage, τ. If the score of a hypothesis, associated with ξ_1, at stage τ is lower than threshold, t_τ, the hypothesis is discarded and is not evaluated in later stages. Furthermore the score computation of the additional part is narrowed to image portions where the previous score of the hypothesis plus the pairwise term of this part, $\mathcal{P}_{1,i}$, is above a certain threshold, t'_τ. As consequence, the evaluation of the partial models is avoided in large portions of the image for all but the first partial model.

Figure 2.15 illustrates the process of stepwise pruning hypotheses. On the left a test image is overlaid by the final bounding boxes (red for the object bounding box, blue for the part bounding boxes). The images on the right show in white where part appearance models for parts, $i = \{2, ..., 6\}$, have been evaluated. White indicates the locations that have to be evaluated with the corresponding part and corresponding partial model.

To further accelerate detection, [Felzenszwalb 2010a] propose to create $2N$ models in the hierarchy by duplicating the hierarchy of DPMs and simplifying the appearance models of the first N models. The appearance model is simplified by reducing the features' dimensions by projecting the features into a subspace obtained by Principle Component Analysis (PCA). Simplified appearance models are evaluated about 6 times faster than the full ones. For these simplified models different thresholds are determined.

Cascaded detection speeds up inference by a factor of 7.8 without the simplified appearance models and by a factor of 22 with PCA approximated appearance models.



(a) (b)

Figure 2.15: Left: Input image with object detection result. Right: Binary maps demonstrating hypotheses pruning. The inference with cascades accelerates detection by pruning hypotheses below a certain threshold with simplified models. From top to bottom and left to right, the maps show the increasingly complex models are evaluated only on small portions of the image, indicated in white. [Felzenszwalb 2010a]

2.2.3 Dual-Tree Branch-and-Bound

The DP algorithm in Section 2.2.1.1 improves over a sliding window approach by avoiding to repeat computations, speeding up inference from quadratic to linear in the number of pixels. Yet all part scores are computed at least once. This can be avoided with Dual-Tree Branch-and-Bound (DTBB) [Kokkinos 2011], reducing runtime to logarithmic in the best-case scenario. The algorithm is efficient because it attributes computational resources only to interesting parts of the image.

In the remainder of this chapter, we present the main ideas for DTBB, but we will defer some technical details to Chapter 3. This establishes the base for the 3D Branch-and-Bound algorithm in Chapter 3 and 4.

2.2.3.1 General Branch-and-Bound

Branch-and-Bound searches over intervals, based on an upper bound of the score function in the interval. The general Branch-and-Bound algorithm works as follows: Initialize a priority queue with an interval that contains the full label space. Then iteratively execute the following two steps:

1. **Branch**: Pick the top interval from the priority queue and split the interval into two child intervals.

2. **Bound**: Determine the upper bound of the score for both child intervals and push them onto the priority queue with the upper bound as sorting criterion.

The stopping criterion is reached when an interval is picked that contains only one element. This is the globally optimal solution.

2.2.3.2 Branch-and-Bound for DPM

For optimization of star-shaped DPMs, the objective is to maximize the score function

$$S(\Xi) = \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{i=2}^{N} \mathcal{P}_{(1,i)}(\xi_1, \xi_i) \tag{2.17}$$

$$\tag{2.18}$$

as described in Equation 2.2). This assigns a label to every node in the graph $G = (V, E)$. The label space, Ω_{2D}, of each node in V consists of the set of coordinates of the input image, that is, $|\Omega_{2D}| = w*h$, with w representing the width and h representing the height of the image.

Due to the star-shaped graph structure, the elements of the sum in the right hand term are independent of each other and relate solely to the position of the center node. As in Equation 2.6 we can write the score of a pose, $\Xi = \{\xi_1, ..., \xi_N\}$, as a function of the center node, ξ_1:

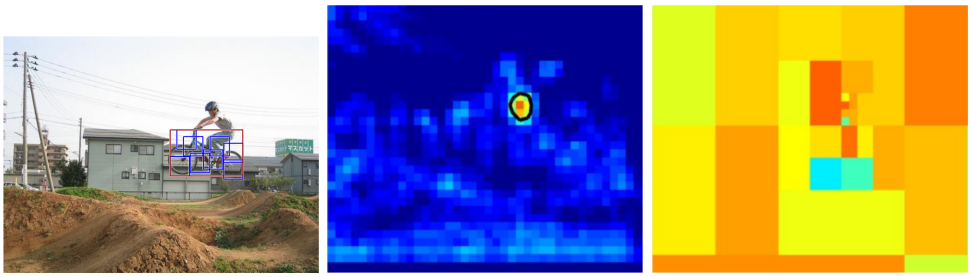
$$S(\xi_1) = \sum_{i=1}^{N} \max_{\xi_i} (\mathcal{U}_i(\xi_i) + \mathcal{P}_{1,i}(\xi_1, \xi_i)), \tag{2.19}$$

We consider an example of detecting a bicycle in Figure 2.16a. In Figure 2.16b the exact score is displayed, evaluating Equation 2.19 for every pixel. In this example the score's maximum is closely focused around one point, allowing the Branch-and-Bound algorithm to disregard significant parts of the image. A similarly peaked score distribution is common in natural images. The DTBB profits from this, as illustrated in Figure 2.16c where we show which intervals of the image are evaluated during inference. Large portions of the image are covered by few large rectangles, saving computation time. The subdivision is finer around the actual detection location, indicating that the algorithm spends its resources on interesting areas of the image.

Branch-and-Bound consists of splitting the image into intervals and determining a fast computable upper bound for each of those solution intervals. The next iteration picks the interval with the currently best upper bound. Thus intervals with a high upper bound are prioritized to be refined over intervals with a low upper bound. Intervals with a low upper bound will be sorted to the end of the queue and will therefore effectively be skipped, saving computational resources. In practice, large parts of the image contain no or little image evidence for a part, allowing the algorithm to focus on the relevant image parts (Figure 2.16c).

2.2.3.3 Dual-Tree Branch-and-Bound

Now we apply Branch-and-Bound to maximize the score, $\max_{\xi_1} S(\xi_1)$, Equation 2.19. In particular we divide the label space of ξ_1 into two parts (two rectangular intervals) and compute an upper bound for each of those. In order to compute



(a) (b) (c)

Figure 2.16: Object detection with Branch-and-Bound [Kokkinos 2011]. The first figure shows the original image with the optimal object bounding box in red. The second image illustrates which intervals the algorithm evaluated and therefore spent time on. The colors refer to the upper bound computed for the respective interval. The third image displays the exact score of the detector, evaluated densely. For both images on the right, blue stands for low scores and red stands for high scores.

an upper bound over an interval of labels for ξ_1, we need to compute an upper bound of the sum of N part contributions, each another optimization problem: $S_{\text{partial}}(\xi_i|\xi_1) := \max_{\xi_i}(\mathcal{U}_i(\xi_i) + \mathcal{P}_{1,i}(\xi_1, \xi_i))$. This problem is again upper bounded over intervals. This constitutes the Dual-Tree Branch-and-Bound, consisting of two nested Branch-and-Bound levels, one for the center node and one for each part node.

Part Upper Bound As required by the Bound-step, an upper bound of the i-th part contribution, $S_{\text{partial}}(\xi_i)$, needs to be computed. For the sake of clarity we simplify this problem by replacing the pairwise term, $\mathcal{P}_{1,i}(\xi_1, \xi_i)$, by a quadratic function, $||\xi_i||_2^2$. In Chapter 3 we discuss in detail the bounding of the full pairwise term.

Then the upper bound for $\max_{\xi_i}(\mathcal{U}_i(\xi_i) + ||\xi_i||_2^2)$ can be efficiently computed over an interval ν as

$$\max_{\xi_i \in \nu} (\mathcal{U}_i(\xi_i) + ||\xi_i||_2^2) \leq \max_{\xi_i \in \nu} \mathcal{U}(\xi_i) + \max_{\xi_i \in \nu} ||\xi_i||_2^2. \tag{2.20}$$

The inequality in Equation 2.20 decouples the unary term $\mathcal{U}(\xi_i)$ from the quadratic term $||\xi_i||_2^2$, allowing it to efficiently compute an upper bound:

- $\max_{\xi_i \in \nu} ||\xi_i||_2^2$ can be computed analytically for any rectangular interval $[x_a^\nu, x_b^\nu] \times [y_a^\nu, y_b^\nu]$, corresponding to a node ν, as $x_b^2 + y_b^2$.
- $\max_{\xi_i \in \nu} \mathcal{U}(\xi_i)$ can be precomputed for every interval.

Then the maximum score of part i is determined as the maximum over intervals $\max_{k=1}^{K} \max_{\xi_i \in \nu_k} S_{\text{partial}}(\xi_i)$, with $\bigcap \nu_1, ..., \nu_K = \Omega_{2D}$ that cover the image space.

We need to precompute the upper bound only over certain intervals. We note that splitting the label space, Ω_{2D}, repeatedly into 2 equally sized parts in the

branching step corresponds to traversing a Kd-Tree over the image domain, where the Kd-Tree's root represents the full image domain and the tree's leaves correspond to image pixels. The Kd-Tree's inner nodes correspond to rectangular image intervals. In such a tree the leaves contain the unary potentials, $\mathcal{U}$. The Kd-Tree's inner nodes are filled with the maximum over their child nodes. It takes linear runtime in the number of pixels to create this tree. This is illustrated in Figure 2.17. On the left we show an example unary potential as array. On the right we show a resulting Kd-Tree, built by incrementally splitting the array into two intervals and assigning every tree node the maximum of its children.

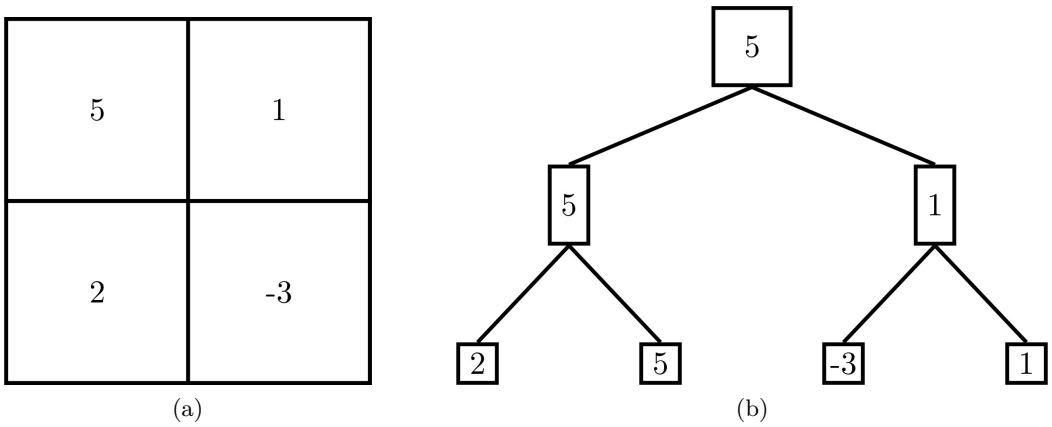


Figure 2.17: Precomputing the unary potential. Left: example unary potential given as array. Right: resulting Kd-Tree.

To avoid confusion, we point the reader towards the double use of the word *node*: there are nodes within a Kd-Tree and there are nodes as part of the graph, $G = (V, E)$, describing the DPM. We refer to a node within a Kd-Tree explicitly as "Kd-Tree node" or as "interval", indicating its connection to a rectangular interval in the image. Otherwise a "node" belongs to the DPM graph, G.

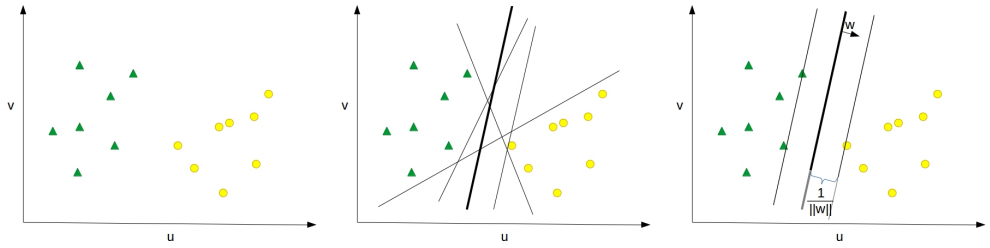
Center Upper Bound In order to compute an upper bound of the score, $S(\xi_1)$, with ξ_1 with an interval ν, the Dual-Tree Branch-and-Bound traverses two sets of trees in parallel. In every branch-step the DTBB descends in two trees: in the Kd-Tree for the center node and in the set of N Kd-Trees for parts. In the bound-step, the upper bound for the center node is computed as sum over the upper bounds of part contributions. Smaller intervals ν for the part nodes render the estimation of the parts' contributions to the final score more precise, as the tightness of the inequality in Equation 2.20 depends on the size of the interval, ν. Larger intervals cover greater portions of the image space, thus reducing the computational cost for upper-bounding. To trade-off between computational cost and tightness of upper bounds, DTBB traverses the Kd-Trees of the center node at the same rate as the Kd-Trees of the other part nodes.

2.3 Model Learning

Up to now we have described part-based models to represent objects and discussed efficient methods to optimize these graphical models. Now we turn towards learning the parameters of these models from given training data. We start from binary classification models for detection before moving to structured models that are appropriate for training DPMs. Then we discuss the special case when the training data lacks part annotations.

2.3.1 Support Vector Machines (SVMs)

Support Vector Machines (SVM) can be used to train linear classifiers, dividing datapoints of two classes by a hyperplane in feature space. The task is to find a linear classifier w, given a set of annotated data points $(x_1, y_1), ..., (x_n, y_n)$, with $f(x_i)$ the feature vector for x_i and $y_i \in \{-1; +1\}$ the corresponding labels. Please refer to Figure 2.18 as an example. The triangles and circles correspond to a label y_i of -1 and $+1$, respectively. Here, the feature space is 2-dimensional, so we plot



(a) (b) (c)

Figure 2.18: Linear SVM in a 2 dimensional example case. 2.18a: Input data for binary classification into triangles and circles. 2.18b: Potential linear classifiers, all of which can correctly classify the given data points. 2.18c: The solution found by SVM maximizes the margin, in other words, the space around the separating line.

the features as u-v-coordinates and the hyperplane reduces to a straight line in 2D. The black lines in Figure 2.18b show several options of valid classifiers w each of which separating perfectly datapoints of the two classes. A data point with features $f(x)$ is labeled with as

$$y = \begin{cases} +1 & \text{if } \langle f(x), w \rangle + b \geq 1 \\ -1 & \text{if } \langle f(x), w \rangle + b \leq -1 \end{cases} \tag{2.21}$$

In general, infinitely many hyperplanes will label the training set correctly if data is separable on the training set. SVM chooses the hyperplane that maximizes the minimum margin, in other words, it maximizes the minimum distance between the separating hyperplane and the data points (Figure 2.18c). This ensures maximum stability to small perturbations of the input data and generalization to test data.

SVM training is efficient and yields good performance if given sufficiently separable features.

To ensure the data is correctly labeled, we need $y_i\langle f(x_i), w\rangle + b \geq 1$. The distance between hyperplane and its closest (supporting) data points with $y_i\langle f(x_i), w\rangle + b = 1$ is computed as $1/||w||$. To maximize the margin, we solve the following optimization problem:

$$\max \frac{1}{||w||} = \min ||w|| \tag{2.22}$$
$$\text{s.t. } y_i(\langle f(x_i), w\rangle + b) \geq 1 \quad \forall i$$

This convex optimization problem can be solved with a primal-dual formulation [Mangasarian 1999, Frie 1998] or subgradient descent [Zhang 2004, Shalev-Shwartz 2011].

The "hard" SVM has two downsides: Firstly it is very susceptive to outliers and secondly training data are usually not perfectly separable. To address this, the soft margin SVM [Cortes 1995] allows some points to lie within the margin or to be wrongly classified. For every point that does not satisfy the hard margin constraint, a penalty has to be payed. To allow points to be misclassified, [Cortes 1995, Vapnik 1998] introduced slack variables ζ_i for each constraint in Equation 2.22, and the optimization problem is written as

$$\min_{w,\zeta\geq 0} \frac{1}{2}\langle w, w\rangle + \frac{c}{n}\sum_{i}^{n}\zeta_i \tag{2.23}$$
$$\text{s.t. } y_i(\langle f(x_i), w\rangle + b) \geq 1 - \zeta_i \quad \forall i,$$

with parameter c determining the strictness of the SVM. Setting parameter $c = \infty$, the problem reduces to the previous case (Equation 2.22). If $\zeta_i = 0\ \forall i$, the soft- and hard-margin solutions are identical.

2.3.2 Structured SVM

The structured support vector machine (SSVM) is the extension of SVM towards structured output prediction. To this end, [Tsochantaridis 2004] propose a feature function that includes the labels. Thus a structured label, $\bar{y} = (\bar{y}_1, ..., \bar{y}_d)^T \in \{0, 1\}^d$, is determined as $\bar{y} = \arg\max_y \langle w, f(x, y)\rangle$. Analogous to SVM, a slack variable ζ_i is introduced for every data pair (x_i, y_i), $i = \{1, ..., n\}$, that subsumes the loss occurring for this pair. [Tsochantaridis 2004] distinguishes between margin and slack rescaling to upper bound the loss, resulting in slightly different algorithms [Blaschko 2016]. We will follow the path of margin rescaling, summing up the loss of labeling and the classification score

$$\Delta_M(y, \bar{y}) = \max_{\bar{y}\in Y}(\Delta(y, \bar{y}) - w^T f(x, y) + w^T(f(x, \bar{y}))), \tag{2.24}$$

with

$$\Delta(y, \bar{y}) = \sum_{l=1}^{d} \mathbb{1}(y_l \neq \bar{y}_l) \tag{2.25}$$

being the number of differing labels between y and $\bar{y}$ over all d dimensions, indexed by $(.)_l$. Plugging this into Equation 2.23, we obtain

$$\min_{w,\zeta \geq 0} \frac{1}{2}\langle w, w \rangle + \frac{c}{n} \sum_{i}^{n} \zeta_i \tag{2.26}$$

$$\text{s.t. } \forall i, \quad \forall y_i \in Y : w^T[f(x_i, y_i) - f(x_i, \bar{y}_i)] \geq \Delta(y, \bar{y}_i) - \zeta_i.$$

If the slack variable, ζ_i, equals 0, then the constraint simplifies to

$$\langle w, f(x_i, y_i) \rangle - \langle w, f(x_i, \bar{y}_i) \rangle \geq \Delta(y, \bar{y}_i), \tag{2.27}$$

such that the score difference between the ground truth configuration and any other configuration is larger than the loss incurred by that labeling.

Although the output space Y may be very large (in the application of DPM, this is usually the number of pixels in the image), [Joachims 2009] state that this quadratic problem can be solved efficiently. [Joachims 2009] proposes a cutting-plane algorithm that solves this problem and is linear in the number of data samples. The important insight is that only very few constraints are active at any point and involved in reaching an approximately optimal solution. It is sufficient to take only a subset of constraints into consideration, called the working set [Tsochantaridis 2004]. To this end, they propose the "1-slack" formulation, subsuming all slack variables in one single slack ζ. The number of constraints increases as the slack has to account for all combinations of solutions y_i for all data points (x_i, y_i) at once. The following optimization problem is derived:

$$\min_{w,\zeta \geq 0} \frac{1}{2}\langle w^T, w \rangle + c\zeta \tag{2.28}$$

$$\text{s.t. } \frac{1}{n} w^T \sum_{i=1}^{n} (f(x_i, y_i) - f(x_i, \hat{y}_i)) \geq \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, \hat{y}_i) - \zeta \quad \forall (\hat{y}_1, ..., \hat{y}_n) \in Y^n.$$

This results in an iterative algorithm. Initially, the working set W of constraints is the empty set. Then, each iteration of the "1-slack" algorithm consists of two steps: First, the quadratic problem is solved for the current working set of constraints, W. Second, this working set is iteratively extended by the most violated constraint per data sample. To this end, compute

$$W = W \cup \arg\max_{\hat{y}} w^T f(x_i, \hat{y}) + \Delta(y_i, \hat{y}) \tag{2.29}$$

adding the most violated constraint to the set, W. These two steps are repeated until the constraint violation including the new constraints falls below a chosen

threshold ε.

2.3.3 Latent Variable SVM

The goal is to train a DPM such that the model predicts the training data correctly. To this end it is useful to reformulate the DPM such that it depends linearly on the model parameters. In a second step we discuss the latent SVM approach presented in [Felzenszwalb 2008].

We write the score function of the DPM, as given in Equation 2.2, for a star-shaped graph:

$$S(\Xi) = \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{(1,i)\in E} \mathcal{P}_{(1,i)}(\xi_1, \xi_i), \tag{2.30}$$

where the unary term, $\mathcal{U}_i(\xi_i) = \langle w_i, f(\xi_i)\rangle$, is a inner product between the model weight vector, w_i, and the feature vector f of coordinate ξ_i.

The pairwise term, $\mathcal{P}_{1,i}(\xi_1, \xi_i)$, as in Equation 2.5, can be expanded in order to yield the linear coefficients, $a_2, ..., a_N, b_2, ...b_N$, up to a constant with respect to the pose, $\xi_1, ..., \xi_N$:

$$\mathcal{P}_{1,i}(\xi_1, \xi_i) = -(\xi_1 - \xi_i - \mu_{1,i})^T C_{1,i}(\xi_1 - \xi_i - \mu_{1,i}) \tag{2.31}$$

$$= \underbrace{\langle \mathsf{diag}(C_{1,i}), (\xi_1 - \xi_i)^2\rangle}_{a_i} + \underbrace{\langle -2\mathsf{diag}(C_{1,i})\mu_{1,i}, (\xi_1 - \xi_i)\rangle}_{b_i} + \underbrace{\langle \mathsf{diag}(C_{1,i}), \mu_{1,i}^2\rangle}_{constant}, \tag{2.32}$$

where $\mathsf{diag}(.)$ extracts the main diagonal as vector from a quadratic matrix. We identify pairwise features as $\tilde{\xi}_i = \xi_1 - \xi_i$ and $\tilde{\xi}_i^2 = (\xi_1 - \xi_i)^2$.

We now stack the part unary weight vectors, $w_1, ..., w_N$, and pairwise parameters, $a_2, ..., a_N, b_2, ...b_N$, to a model parameter vector, $\beta := (w_1, ..., w_N, a_2, ..., a_N, b_2, ...b_N)^T$. Similarly we stack the unary features, $f(\xi_1), ..., f(\xi_N)$, with the pairwise features, $\tilde{\xi}_2, ..., \tilde{\xi}_N, \tilde{\xi}_2^2, ..., \tilde{\xi}_N^2)^T$. This gives the optimization problem as inner product between model parameters and feature vectors:

$$\max_{\Xi} S(\Xi) = \max_{\xi_1,...,\xi_N} \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{(i,j)\in E} \mathcal{P}_{(1,i)}(\xi_1, \xi_i) \tag{2.33}$$

$$= \max_{\xi_1,...,\xi_N} \sum_{i=1}^{N} \langle w_i, f(\xi_i), \rangle + \sum_{(1,i)\in E} \langle \begin{pmatrix} a_i \\ b_i \end{pmatrix}, \begin{pmatrix} \tilde{\xi}_i \\ \tilde{\xi}_i^2 \end{pmatrix} \rangle \tag{2.34}$$

$$= \max_{\xi_1,...,\xi_N} \langle \beta, f(\Xi)\rangle. \tag{2.35}$$

We discuss now the training of this DPM with latent SVM. Latent variable SVM (LSVM) was adapted by [Felzenszwalb 2008] to train DPMs within a few hours from the bounding boxes on the PASCAL VOC dataset. The PASCAL VOC dataset [Everingham 2010] does not annotate object parts, thus requiring to train

the DPM model with weak supervision.

To train the model, one can minimize a hinge loss of the following form:

$$\beta^*(D) = \arg\min_{\beta} \lambda||\beta||^2 + \sum_{i=1}^{m} \max(0, 1 - y_i S_\beta(x_i)), \tag{2.36}$$

with D being the set of labeled examples, $D = \{(x_1, y_1), ..., (x_m, y_m)\}$, where y_i is 1 for positive samples and -1 for negative samples and x_i is the i-th training image, and $S_\beta(x_i)$ is the DPM score function. To account for the missing part annotations, the latent variables, $z_1, ..., z_m$, are introduced. The score function, $S_\beta(x_i) = \max_{z_i} \langle \beta, f_{x_i}(z_i) \rangle$, computes the maximum score with respect to the latent variables z_i.

[Felzenszwalb 2008] notes that the score function $S_\beta(x_i)$ is the maximum of functions, each linear in β, and thus convex. Consequently, the hinge loss in Equation 2.36 is convex for $y_i = -1$. If the latent variables, z_i, are fixed for every sample, x_i, defining a series of fixed latent variables, z_i, then the loss function is even convex for positive samples with $y_i = 1$. After initialization of the latent variables heuristically such that they roughly cover the given bounding box for each instance, Felzenszwalb et al. suggest the following iterative training algorithm:

- Fix all latent variables, z_i, and optimize the model parameters, β, as standard SVM.
- Fix the model parameters, β, and optimize the latent variables, z_i, for all positive samples as $\arg\max_{z_i} \langle \beta, f_{x_i}(z_i) \rangle$.

This training algorithm solves iteratively a convex and a concave optimization problem. This guarantees converge to a local optimum [Yuille 2003].

2.4 Outlook

Having outlined the background for the thesis, we now turn to an overview of our main contributions. In Chapter 3 we learn 3D DPMs for 3D object detection with parts. Due to the additional dimension, the label space, Ω_{3D}, is now cubic in the image's resolution. This corresponds typically to a multiplying factor of more than 100, rendering approaches with GDT impractical. We thus extend the DTBB algorithm and customize it to exploit the discrepancy between 2D unary and 3D pairwise potentials. Our detection algorithm runs in a fraction of a second while reaching the global optimum. In Chapter 4 we turn towards inference for 3D human pose estimation based on 3D unaries. To this end, we integrate the inference algorithm into a CNN layer, augmenting the CNNs capabilities with efficient, global inference for star-shaped DPMs. Motivated by the structure of the human skeleton, we search to employ richer graph structures, especially loopy models. To this end, we overcome the limitation of star-shaped graph structures by applying ADMM, a graph decomposition algorithm. We show that efficient inference is possible for

loopy graphs by combining ADMM with DTBB. Experiments confirm that these models benefit the pose estimation accuracy.

Chapter 3

3D Object Pose Estimation with 2D Unary Potentials

Contents

3.1 Prior Work on 3D Pose Estimation 42
3.2 3D DPM - Modeling 46
3.2.1 3D DPM with 2D Unary Potentials 46
3.2.2 Viewpoint-Independent Structure 48
3.2.3 Viewpoint-Specific Models 52
3.2.4 Model Training 54
3.3 3D DPM - Inference 56
3.3.1 Joint Inference with Depth Variables 56
3.3.2 Anchoring in Depth 61
3.4 Evaluation 62
3.4.1 Model Validation 62
3.4.2 Runtime Performance 66

This chapter proposes an extension of the Deformable Part Model (DPM) to 3D. Our focus lies on pose estimation in 3D, namely the estimation of the location of object parts in 3D space. The output of our system includes a more detailed object description in terms of object parts and their relative 3D position. Thus it resembles a coarse object reconstruction when given the rough object location.

There are different approaches to obtain 3D reconstructions, but they require either specialized hardware or multiple images: One avenue is the use of depth sensors, like the kinect camera and impressive reconstructions have been achieved [Newcombe 2011, Whelan 2012, Sturm 2012, Hornung 2013]. While these cameras have become available for consumers, they usually fail for larger ranges and specular surfaces. Without this specialized hardware, another avenue for object reconstruction are stereo methods. Based on at least two images of the same object from different perspectives corresponding points are found and the 3D location is triangulated (refer to [Hartley 2003, Seitz 2006] for a comprehensive treatment of the topic). However, often there is only a single image available, leading to single, monocular image reconstructions.

We aim at solving 3D reconstruction from single, monocular images with minimal requirements. We approach the challenging task of 3D pose estimation by

applying a DPM in 3D, representing objects within an object class. We train this DPM from 2D annotations without requiring the existence of 3D CAD models or 3D keypoint locations. For loop-free DPMs efficient and exact optimization is possible, linear in the number of pixels with GDT and DP (Section 2.2.1). In 3D, the number of pixels is cubic in the granularity of the discretization along each dimension. Therefore the number of operations needed for inference is cubic in the discretization of the solution space, resulting in augmented optimization runtime (tens of seconds instead of a fraction of a second).

Our main contribution lies in an algorithm for solving the resulting optimization problem. We adapt the Branch-and-Bound algorithm (Section 2.2.3) to 3-dimensional problems. A straightforward implementation of a 3D Branch-and-Bound would extend each unary tree by one dimension, copying the unary for each depth in the discretization. This idea has one main weakness: The size of the Kd-Trees would again be linear in the number of pixels and therefore cubic in the discretization. Creating these trees is costly in computation time and memory. Instead we propose to exploit the structure of the unary term and traverse a 2D unary tree and a tree corresponding to depth in parallel. This approach avoids instanciating a full 3D volume, but does not discard potential solutions. It allows us to reduce complexity from linear in the number of pixels to logarithmic in the number of pixels and runtime is typically a fraction of a second. Note that we achieve the global optimum in every case. The exact runtime depends on the image instance, however our experiments show very little variance.

In this chapter, we first analyze approaches that existed in the literature prior to our approach. We then describe how an object class is modeled as a DPM in 3D. A single compact 3D model is created from minimal annotations, namely the silhouette and a few manually annotated object keypoints. Then we train viewpoint-specific model parameters using structured SVM (SSVM), based on CNN features. We customize a Branch-and-Bound approach to efficiently solve the resulting optimization problem. Finally we validate our approach by comparing its performance to the state of the art method on the PASCAL3D+ dataset.

3.1 Prior Work on 3D Pose Estimation

There is a long line of research for object detection in general, and part-based object detection and pose estimation in specific, but a much smaller portion is dedicated to 3-dimensional problems. We focus on works that are closely related to our work, namely 3D part-based pose estimation from a single RGB image. In this context, we discuss three different aspects, (i) training data annotation, (ii) modeling approach and (iii) inference problem in 3D part-based models.

Training Data Annotation

Part-based object models contain more parameters than similar simple, rigid models. Thus larger amounts of training data are needed in order to train such

a model. There are large datasets for 2D object detection [Everingham 2010, Russakovsky 2015, Lin 2014], where an object is annotated as a bounding box. However, creating detailed, part-based annotations is tedious and therefore expensive. To obtain 3D models, annotations in 3D are favourable, but creating 3D annotations by hand is hard, although tools have been created to aid annotation [Xiang 2014].

To circumvent this obstacle, synthetically created models, computer-aided design (CAD) models, are available for a range of different objects, for example in [Chang 2015] and the Google 3D Warehouse collection. Synthetically generated models and images possess different characteristics to natural objects and scenes [Aubry 2014, Massa 2016]. This leads to inferior performance if a model is trained exclusively on synthetic data and tested on real-world images [Yu 2010, Michels 2005, Pepik 2015a].

Realistic CAD models are used in [Aubry 2014, Lim 2014] in combination with part-based models. [Aubry 2014] extracts a large number of distinctive "parts" from rendered CAD models. In a test image these parts are recognized and combined for a final result, enforcing consistency with a DPM-like formulation. More sophisticated, [Lim 2014] adapt to natural occurrences by weighting the importance of parts according to natural occlusion statistics and discriminativity. Therefore a small amount of annotated real-world images is needed, where the annotated parts need to coincide with the parts contained in the models trained on CAD models.

[Liebelt 2010] proposes to take advantage of synthetical CAD models to learn the objects 3D geometry, but uses real-world photos to learn appearance terms. The parts are defined on the grid points of a regular grid covering the object. The same grid is laid over the rendering of a 3D CAD model and allows to connect the appearance from natural images with the geometry of CAD models. This relieves them from part-based annotations on real-world photos and diminishes the effects of using synthetically generated models.

A different approach has been taken by [Kar 2015] by utilizing a minimum amount of annotations on real-world data. Based on the non-rigid structure from motion (NRSfM) algorithm of [Torresani 2008, Bregler 2000], they use only silhouette data and a few 2D keypoints. From this they generate a 3D visual hull for each object category and estimate a shape deformation basis.

Modeling

3D reconstruction from a single 2D image is an ill-posed problem, because there are infinitely many scenes giving rise to the same image. This can be remedied by imposing priors on the expected reconstruction, giving preference to a certain solution. Three main flavors have been explored for 3D object reconstruction: Basis models, exemplars and part-based models.

[Kar 2015] proposes a shape bases approach that is purely data-driven. A single shape is rigid, but the linear combination of deformation bases allows for class-specific deformations in order to adapt to an image at test time. These coefficients are determined in a gradient descent optimization. To recover high fre-

quency shape details, the shape is refined in a final step by exploiting shading cues [Barron 2012, Barron 2015]. The first step depends on the quality of the initial object segmentation and is furthermore susceptible to local optima. A similar idea has been developed in [Blanz 1999], where faces are modelled as a linear combination of face basis shapes. This idea reaches state of the art results on face shape reconstruction in [Booth 2017] who derive an iterative optimization scheme that incorporates a learnable texture model to adapt to "in the wild" instances. The main contribution of [Prasad 2010] is to reconstruct objects based on edge curvature features instead of requiring full correspondences on the training set.

The work [Trigeorgis 2016] proposes to learn to deform a face mean shape such that the shape aligns with the given image instance. The deformation in each iteration is calculated by a neural network and is thus not limited to the basis shapes estimated in above mentioned works.

A model-free approach relies on comparing new images to annotated training data, so called exemplars. This simplicity comes at the price of requiring large amounts of data from which to draw exemplars, as well as a high computational cost in order to compute the comparison of exemplars with the input image.

An exemplar approach is demonstrated in [Aubry 2014], where parts of rendered CAD models are used as exemplars and combined to obtain a full object reconstruction in 3D, as mentioned above. In contrast to [Malisiewicz 2011] who align CAD models as a whole with the image, [Aubry 2014] searches viewpoint-specific parts in the image. The exemplars are extracted from rendered CAD models, resulting in presumably distinctive 2D patches. To account for the visual variation of chairs and the viewpoint change, more than 800K exemplars are extracted from 1393 CAD models.

An advancement is described in [Massa 2016] where exemplars are compared to the input image in terms of CNN features. The image features are transformed to match more closely the features created from rendered CAD models. The computation of similarity is parallelized in the parallel structure of neural networks, reducing runtime from minutes to seconds.

Lastly, there has been a range of part-based 3D models, structured as a graph connected by pairwise constraints and closely related to DPMs. We will characterize some of them in the following section.

Optimization of 3D Part-Based Models

[Lim 2014] formulates a star-shaped DPM with a large number of mixture models, each corresponding to a 3D pose of the object. Each mixture models are obtained from rendered CAD models, therefore reducing inference to solving many 2-dimensional problems, solvable with DP (Section 2.2.1).

[Hejrati 2014] reconstructs 3D objects by finding a 3D shape that matches the image evidence. The shape is formed as a linear combination of basis shapes, each consisting of a number of keypoints associated with a local template, encoding the visual appearance, similar to DPM part filters [Felzenszwalb 2008]. To this end, the 3D optimization is avoided by "rendering" part templates into a global 2D object

template for a large number of viewpoint parameters.

[Fidler 2012] create a 3D part-based model by gluing 2D parts onto 3D cuboids, resulting in an MRF in the form of a tree of depth 3: The cuboid center, the cuboid face position in relation to the cuboid center and the 2D part position in relation to the face center. Via DP (Section 2.2.1.1) the score function is maximized for a discrete set of viewpoint angles. The maximum score corresponds to the final viewpoint. The location of parts on faces and the deformation of faces on the cuboid correspond to a star-shaped MRF. Both problems are solved in 2D with GDT (Section 2.2.1.2).

[Pepik 2012a] describes a full 3D DPM with 3D part locations as the 3D extension of [Felzenszwalb 2010b]. The unary terms are defined as HOG features for discrete viewpoint bins, whereas the 3D structure is defined once for the model. For inference, the 3D problem is reduced to multiple 2D problems by discretizing the desired viewpoint range and solving individual 2D problems with GDT

The training of such a model with latent 3D parts is detailed in [Pepik 2012b]. During training, the latent variables, namely the part positions in 3D, have to be inferred. This is the same problem as the inference during test time, where the part filters are fixed. This non-convex optimization problem is solved with standard coordinate descent and thus can get stuck in local optima.

Similarly, [Pepik 2015c] defines a 3D DPM with 3D pairwise terms and viewpoint-specific part filters. At test time the problem is projected into the image plane and the part filters are interpolated from neighboring viewpoints. Still, they do not optimize in 3D.

The work of [Pepik 2015b] is based on the success of CNNs for object detection and results in an aligned 3D CAD model with the image. This approach proposes a pipeline that successively builds up a 3D model, but also avoids part inference in 3D. First, a rough object bounding box is determined using RCNN [Girshick 2014]. Second, from this CNN a viewpoint is regressed as well. Third, within the region of the bounding box, a 2D DPM is applied to determine the location of 2D keypoints. Fourth, due to correspondence of located keypoints to points on the CAD model, the detection can be lifted into 3D.

Similarities to our Work

We have discussed three model approaches in the previous section, namely the linear combination of basis shapes and exemplar methods and part-based models, like DPMs. We argue that a DPM is the more flexible model than the former, because its independent parts allow to express poses beyond the linear combination of basis shapes. Additionally the Branch-and-Bound optimization for DPMs guarantees finding the global optimum, whereas the parameters of a basis model are estimated with an iterative scheme that can end up in a local optimum [Blanz 1999, Prasad 2010]. Furthermore the DPM allows for a more compact model representation with fewer parameters, benefiting its generalization in absence of abundant training data.

Regarding the use of data annotation, we follow the ideas of [Kar 2015], relying

only on silhouette and a few manually annotated keypoints. This results in less precise and articulated object models that those trained with 3D CAD models. Despite these restrictions, we show appealing performance in reconstructing 3D object models.

Similar to the work of [Pepik 2015c], we deduce the 3D pairwise terms from one global 3D object structure. Unary terms, specific to a discretized viewpoint, follow the ideas in [Felzenszwalb 2008, Pepik 2012a, Zhu 2015, Fan 2005, Kushal 2007, Yan 2007, Stockman 1987]

The work by Pepik et al. [Pepik 2015c] comes closest to a realization of true 3D DPMs. However, in the end no inference method is proposed for part-based 3D object pose estimation or 3D object reconstruction. In this chapter we present an efficient and exact inference method for 3D DPMs with 3D parts. For comparison, we also implement inference with 3D GDT, as hinted at in [Pepik 2015c]. We compare the runtime of both approaches in Section 3.4.

3.2 3D DPM - Modeling

In this chapter we apply a DPM as a flexible and trainable model to represent a 3D object structure. In order to advance the DPM from 2D to 3D, first we introduce depth variables for every object part. Our goal is to build a compact 3D structure for which we derive viewpoint-specific parameters. To this end, we describe systematically how to build and train such a model. We use a non-rigid structure from motion algorithm to form a 3D mesh for each object category, on which we base the graph of the DPM. From this, we derive viewpoint-specific mixture models to reflect viewpoint-specific appearance variability. Finally, the remaining viewpoint-specific model parameters are trained using SSVM.

3.2.1 3D DPM with 2D Unary Potentials

We consider a DPM, as presented in Section 2.1.2.1, with graph, $G = (V, E)$, where $E = \{(1, j)|j \in \{2, ..., N\}\}$, and variables, $\Xi = (\xi_1, ..., \xi_N)$. The N-tuple, Ξ, of 2-dimensional image coordinates is the object's configuration. We raise the label space into the third dimension, from coordinates in a 2-dimensional image plane, Ω_{2D}, with $|\Omega_{2D}| = w * h$, to a volumetric label space, $\Omega_{3D} := \{1, ..., w\} \times \{1, ..., h\} \times \{1, ..., d\}$. Thus the desired output consists of an N-tuple of 3-dimensional vectors, encompassing 3D coordinates of all object parts $i \in \{1, ..., N\}$. We refer to the third coordinate dimension also as depth or "z"-coordinate.

We opt for a joint optimization over these three dimensions. Consequently we stack the two dimensional image coordinates, $\xi_i \in \mathbb{R}^2$, with an additional "z" variable to yield $\xi_i := (\xi_{i_x}, \xi_{i_y}, \xi_{i_z})^T \in \mathbb{R}^3$, a 3-dimensional coordinate. Again, we write $(.)_X$ to index a specific element in a vector.

The DPM score function, S, is

$$S(\xi_1, ..., \xi_N) = \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{(i,j)\in E} \mathcal{P}_{(i,j)}(\xi_i, \xi_j), \tag{3.1}$$

where the unary term

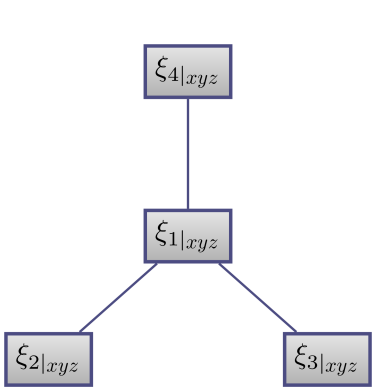
$$\mathcal{U}_i(\xi_i) = \mathcal{U}_i((\xi_{i_x}, \xi_{i_y})) = \langle w, f(\xi_{i_x}, \xi_{i_y}) \rangle \tag{3.2}$$

is independent of the third coordinate. The pairwise term

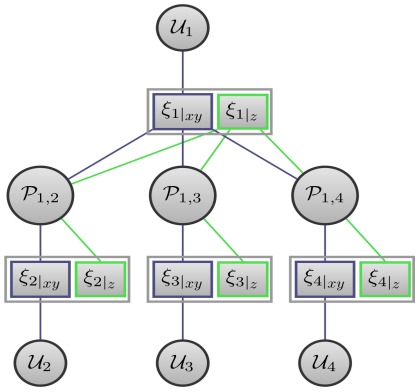
$$\mathcal{P}_{i,j}(\xi_i, \xi_j) = (\xi_i - \xi_j - \mu_{i,j})^T C_{i,j} (\xi_i - \xi_j - \mu_{i,j}) \tag{3.3}$$

is the extension of Equation 2.5 to the third dimension, with $\mu_{i,j} \in \mathcal{R}^3$ and diagonal matrix $C_{i,j} \in \mathbb{R}^{3\times3}$ being parameters in 3D. This pairwise term implicitly assumes an orthographic projection, that is, deviations of $(\xi_i - \xi_j)$ from the nominal offset $\mu_{i,j}$ are punished in short and in long distance from the camera equally. This is a good approximation for large distances, but may introduce inaccuracies at close distances.

The new MRF is illustrated in Figure 3.1 as a factor graph. Squares depict variables in the optimization problem and circles depict the factors, namely the unary and the pairwise terms. The depth, accentuated by variables ξ_{i_z} in green, is not observed directly in the image, hence is not connected to the unary terms $\mathcal{U}_i$. Although the third dimension has no effect on the unary potential, it may influence other variables via pairwise potentials.



(a) Graph of a star-shaped DPM with 4 parts. We emphasize the 3 dimensions, x, y, z, of the variables, $\xi_1, ..., \xi_4$.



(b) Factor graph representing the DPM score function in 3D with added depth variables. Variables are displayed as squares and factors are shown as circles. The newly introduced variables ξ_{i_z}, $i = \{1, ..., 4\}$, are drawn in green. We indicate the stacked variables ξ_i with light grey rectangles.

Figure 3.1: Example 3D DPM and its factor graph.

3.2.2 Viewpoint-Independent Structure

We start by describing how we construct the 3D structure that is common to all viewpoint-specific DPMs of the same object class. To this end, we follow [Kar 2015] to obtain a 3D mesh from 2D keypoints and 2D silhouettes. We then refine this mesh such that nodes of the mesh correspond to object-specific keypoints.

[Bregler 2000] propose a non-rigid structure from motion (NRSfM) technique, applied to a video of people to obtain the 3D structure of a face. This approach estimates jointly the 3D positions of class-specific keypoints and the camera viewpoint for each training instance. In [Kar 2015] model object shapes as linear combination of a "mean shape" and deformation bases. The basis shapes are estimated in an energy formulation using silhouette information, namely ground truth object segmentation masks. As result, they obtain an estimate of the viewpoint for each training instance and a 3D object shape corresponding to the object's visual hull.

We associate the annotated keypoints (lifted to 3D by NRSfM) to points on the "mean shape" mesh by determining the closest mesh node to the mean of the 3D keypoints. Figure 3.2 visualizes the keypoints of all training instances in a common coordinate frame. Then we refine the model by adding mesh points to the model using Geodesic Surface Remeshing [Peyré 2006]. This ensures that the shape is sampled with roughly equidistant points. Our final DPM contains 100 nodes.

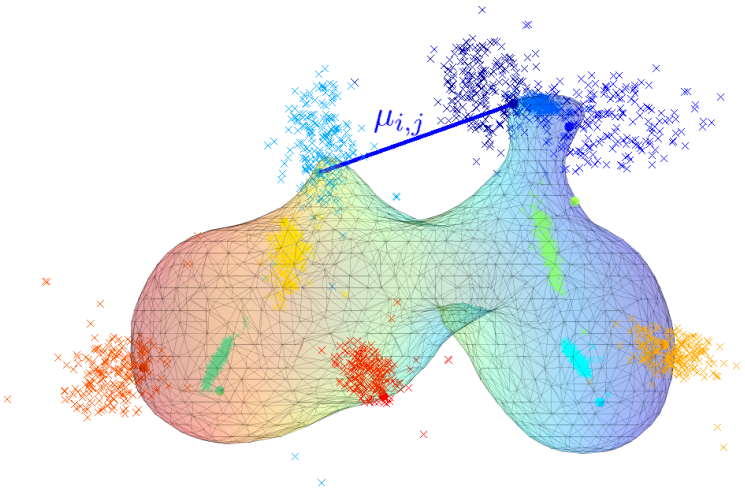
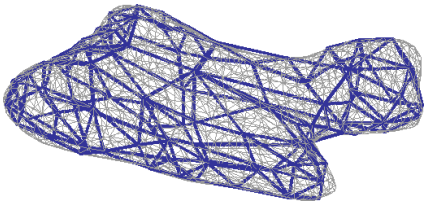
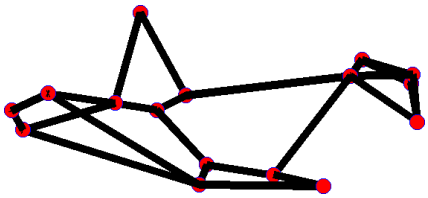
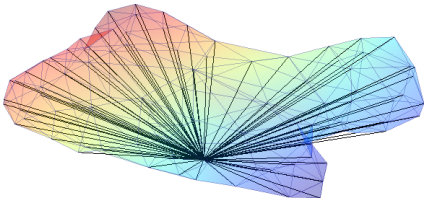
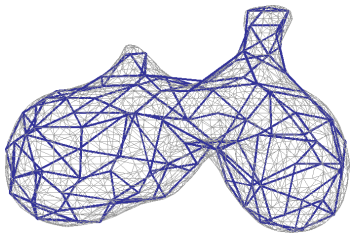
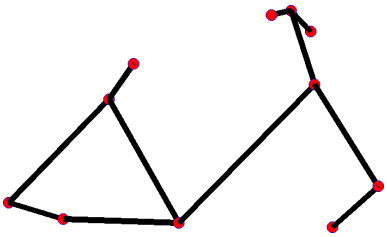
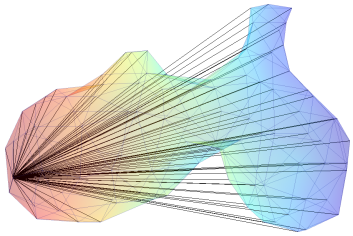


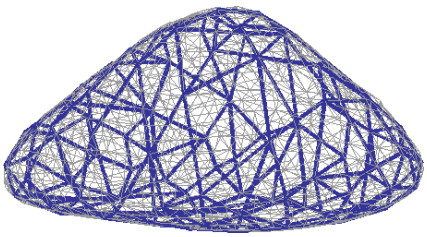
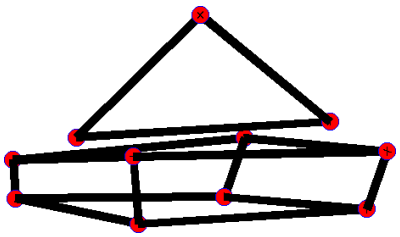
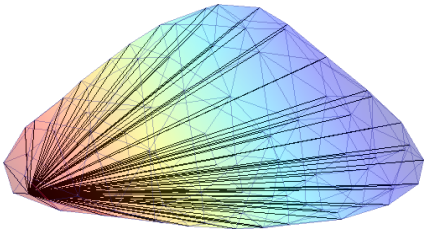
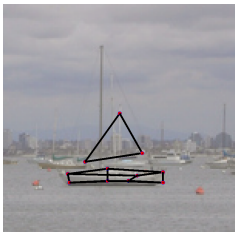
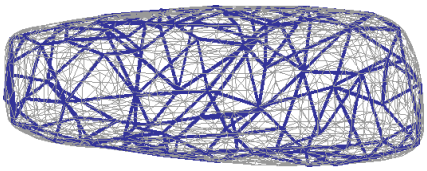
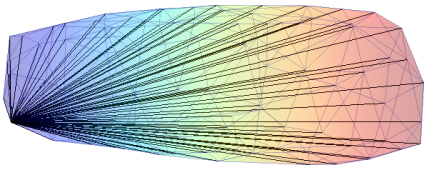
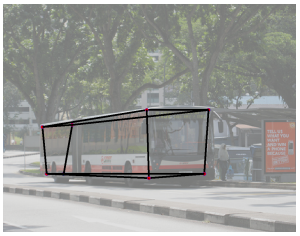
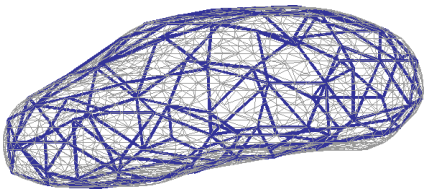
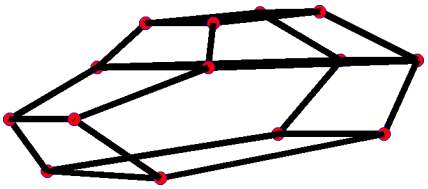
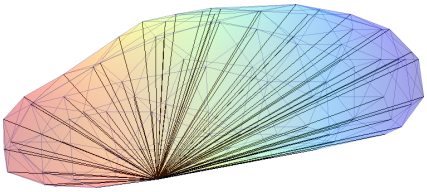
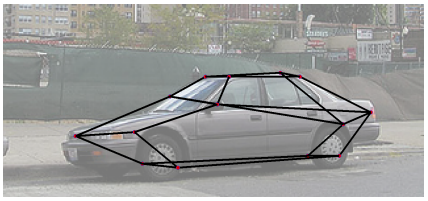
Figure 3.2: Mesh extracted with NRSfM for the "bicycle"-class. Data points, marked by "X", are drawn in a common coordinate system, each color corresponding to a different keypoint. The associated point on the mesh is shown as a filled circle. The computation of one nominal offset, $\mu_{i,j}$, is indicated in dark blue.

We compute the 3D nominal offsets, $\mu_{i,j}$, as the difference between nodes i and j of the refined graph (see Figure 3.2). The result is an articulated 3D object model for each category. We show some object models in Table 3.1. For each object class we provide 4 different views: in the top-left we show the object mesh with 100 nodes with a triangulation shown in blue for better visibility. In the top-right

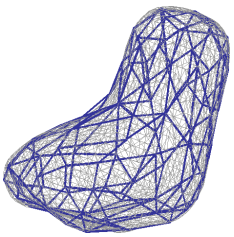
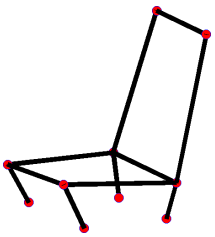
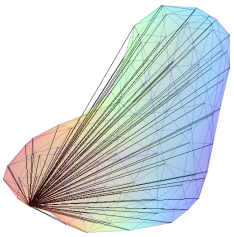
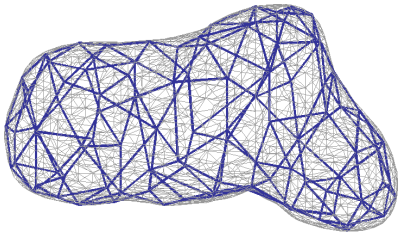
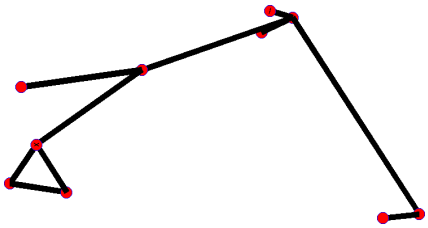
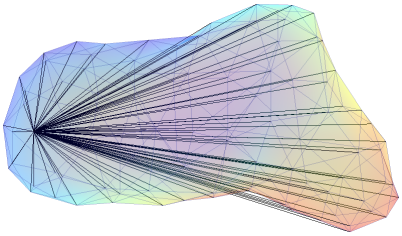
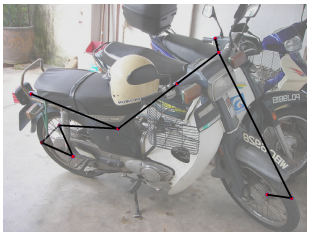
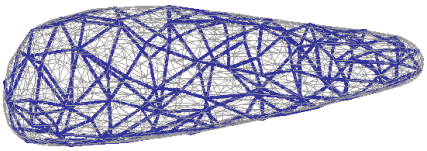
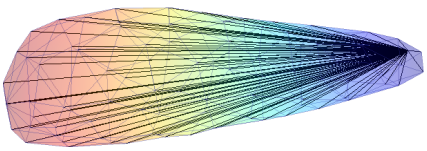
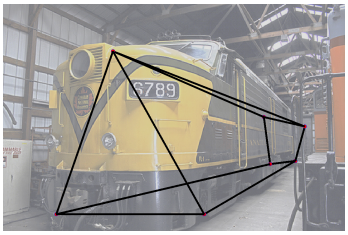
we provide a simplified structure that is based only on the annotated (between 7 and 16) keypoints of the training dataset. In the bottom-left we show one possible 100-node star structure. For better orientation, we color code the object's depth. Finally, in the bottom-right we show a detection result by our approach, illustrated by the simplified graph structure that is based on the annotated keypoints.

Object Mesh	Simplified Representation
Star Structure	Example Detection
Aeroplane	
	
	
Bicycle	
	
	

continued on the next page

Object Mesh Star Structure	Simplified Representation Example Detection
Boat	
	
	
Bus	
	
	
Car	
	
	

continued on the next page

Object Mesh Star Structure	Simplified Representation Example Detection
Chair	
	
	
Motorbike	
	
	
Train	
	
	

continued on the next page

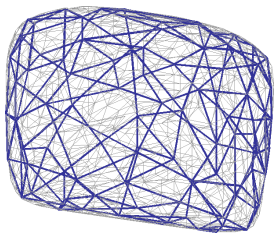
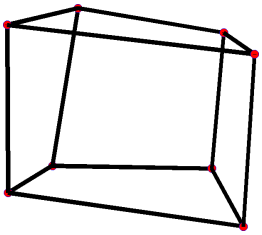
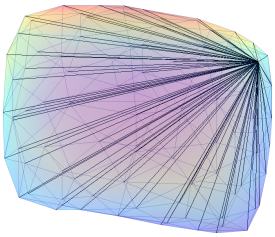
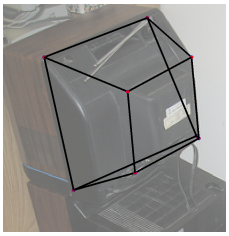
Object Mesh Star Structure	Simplified Representation Example Detection
TV-Monitor	
	
	

Table 3.1: We illustrate our object class models in 4 different ways. Top left quadrant: the object structure extracted by NRSfM. A sub-sampled mesh with 100 nodes is visualized in blue. Top right quadrant: A simplified representation based on the keypoints that are annotated in the PASCAL3D+ dataset. The black lines have been added for clarity. Bottom left quadrant: a star structured graph based on the sub-sampled mesh in the left column. The color encodes depth in a canonical coordinate space (red is close, purple is far). Bottom right quadrant: an example detection pictured in the simplified representation. The model successfully adapts to the given image instance.

3.2.3 Viewpoint-Specific Models

We model an object by a set of DPMs, specific to a number of discretized viewpoint bins. We thereby render object detection robust to viewpoint changes. In this section we define viewpoint-specific models for 3D object detection.

Most objects look very different from various viewpoints, although the object itself does not change. A single model is often insufficient or needs to be complex to capture the vast differences in appearance. So we create a small number of models for each object class such that each model is responsible for a range of viewpoint angles. Therefore one model only has to detect objects under slight viewpoint changes from its canonical viewpoint.

The motivation of our approach is illustrated by Figure 3.3. Due to a large viewpoint rotation, the visual appearance of the wheelchair in Figure 3.3a and 3.3b is very distinct. Thus corresponding keypoints are unlikely to be found in both images. We treat this case with two separate models. Under moderate viewpoint changes, the local appearance of points on the object is expected to stay approximately constant, illustrated by the enlarged image patches around corresponding

landmarks in both Figure 3.3b and 3.3c. Yet the landmarks seem to move in relation to each other. The arrangement of these landmarks is highlighted by the red and green dashed triangles. The superposition of the configuration from Figure 3.3b onto Figure 3.3c shows that the global arrangement of object parts is sensitive to the viewpoint angle. The DPM accommodates such variations of relative part positions in its flexible pairwise terms.

(a) (b) (c)

Figure 3.3: A wheelchair under varying viewpoint angles. The same three selected landmarks and their surrounding image patches are enlarged in both images (b) and (c). The corresponding image patches show a high degree of similarity in image (b) and (c). The landmarks' relative position to each other is indicated by the dashed triangles. We overlay the triangle of the landmarks in image (b) onto image (c) to highlight change under rotation. This motivates us to subdivide the viewing sphere into a small number of viewpoint bins and to train a DPM for every bin.

We divide the sphere of viewing angles in $\mathbb{V}$ evenly spaced viewpoint bins (in our experiments we use $\mathbb{V} = 8$ viewpoint bins). For object pose estimation, we discretize the viewpoint only along the azimuth, such that the range of azimuth for bin v is $[(v-1)(2\pi/\mathbb{V}); v(2\pi/\mathbb{V})[$, for $v = \{1, ..., \mathbb{V}\}$. The azimuth exhibits the highest variance over instances, especially the in-plane rotation is rare in natural images and, for example, cars are rarely photographed from below or above a certain viewpoint. For every viewpoint bin we compute the central angle $\alpha_v = (v - 0.5)(2\pi/\mathbb{V})$. Then we define the viewpoint-specific score function S_v for the v-th viewpoint bin as

$$S_v(\xi_1, ..., \xi_N) = \sum_{i=1}^{N} \underbrace{\langle w_{i,v}, f(\xi_i) \rangle}_{\mathcal{U}_{i,v}(\xi_i)} + \sum_{(i,j) \in E} \underbrace{-(\xi_i - \xi_j - R_v \mu_{i,j})^T C_{i,j,v} (\xi_i - \xi_j - R_v \mu_{i,j})}_{=\mathcal{P}_{i,j,v}(\xi_i, \xi_j)}, \tag{3.4}$$

where $w_{i,v}$ is the viewpoint-specific weight vector for part i, $f(\xi_i)$ is the feature vector extracted for position ξ_i and R_v is the rotation matrix for the rotation in azimuth by α_v.

A large number of viewpoints provides a more faithful adaption to the particular viewpoint present in an image. The main disadvantage is that the more models need to be trained, the less data samples remain to train each one of them. We argue that a small number of viewpoint-specific models is sufficient, because each model pertains a certain tolerance to small viewpoint changes.

Our choice to model an object by a combination of DPMs, each capable of detecting the object within a certain range of viewpoint angles, is further supported by the robustness of deep learning features. It has been noted [Goodfellow 2009, Le 2013] that deep features are fairly robust to small out-of-plane rotations.

3.2.4 Model Training

After having outlined the 3D score function and the derivation of a viewpoint-specific mixture model, we now turn to model training. Discriminative training allows us to design a cost function such that the model's prediction is approximating the ground truth labels. We train the parameters for the unary, w_v, and pairwise potentials, $C_{i,j,v}$, jointly and end-to-end, using a structured SVM (Section 3.2.4), solved by a cutting-plane optimizer.

We have defined the viewpoint-independent structure of our model, namely the pairwise offsets, $\mu_{i,j}$, in a canonical camera viewpoint, in Section 3.2.2 and derived the viewpoint-specific offsets as $R_v\mu_{i,j}$ by rotating them into viewpoint bin v (Section 3.2.3). Now we are left to estimate the viewpoint-specific model parameters, namely the weight vector in the unary potentials and precision matrix, $C_{i,j,v}$, in the pairwise term. For this we need as training data the viewpoint and the 3D keypoint coordinates from each training sample. The ground truth viewpoint annotation is used during training to assign training instances to the corresponding viewpoint-specific model. The 3D keypoints are needed to train the model parameters. We follow [Kar 2015] and apply NRSfM to obtain these keypoints. We train unary parameters, $w_{i,v}$, only for nodes that correspond to annotated keypoints and are therefore precisely localized in the training images. For the remaining nodes we set the unary term to 0.

As discussed in Section 3.2.3, an object is modeled by $\mathbb{V}$ separate DPMs, each corresponding to one viewpoint bin. In the following paragraphs we explain the training of a single mixture model for object detection in 3D, corresponding to a specific viewpoint, $v \in \{1, ..., \mathbb{V}\}$. We assume that training data has been assigned to the viewpoint bin, so that we train the model for a viewpoint bin with the samples assigned to this bin only.

We recognize similarities in the learning to the idea of [Felzenszwalb 2008] in that they train a set of object models to reflect a variability of the object that is not captured by a single DPM. Their approach resembles a viewpoint classification, but the lack of viewpoint annotations may lead to misclassified samples and consequently a weaker classifier. In contrast to the latent variable approach in [Girshick 2012, Felzenszwalb 2008] and Appendix A, we use ground truth viewpoint annotations to assign training instances to the correct mixture. This reduces the

potential error introduced by false bin assignments and allows to train on a more semantically consistent subset.

The score function, $S_v(\xi_1, ..., \xi_N)$, as in Equation 3.1, can be written as an inner product

$$S_v(\xi_1, ..., \xi_N) = \sum_{i=1}^{N} \mathcal{U}_{i,v}(\xi_i) + \sum_{(i,j)\in E} \mathcal{P}_{i,j,v}(\xi_i, \xi_j) \tag{3.5}$$

$$= \sum_{i=1}^{N} \langle f_{\mathcal{U}}(\xi_i), w_{i,v} \rangle + \sum_{(i,j)\in E} \langle f_{P_v}, w_{\mathcal{P}_v} \rangle \tag{3.6}$$

$$= \langle \begin{pmatrix} f_{\mathcal{U}} \\ f_{P_v} \end{pmatrix}, \begin{pmatrix} w_{\mathcal{U}_v} \\ w_{\mathcal{P}_v} \end{pmatrix} \rangle, \tag{3.7}$$

with $f_{\mathcal{U}}$ and f_{P_v} being stacked unary and pairwise features, respectively, and $w_{\mathcal{U}_v}$ and $w_{\mathcal{P}_v}$ being the stacked viewpoint-specific weight vectors for unary and pairwise potentials, respectively. We will now explain these in detail.

The unary potential, as in Equation 3.2, is computed as $\mathcal{U}_{i,v}(\xi_i) = \langle f_U(\xi_i), w_{i,v} \rangle$. We stack the part weight vectors, $w_{1,v}, ..., w_{N,v}$, to obtain the weight vector $w_{\mathcal{U}_v}$. Similarly we obtain the feature vectors, $f_U := (f_U(\xi_1), ..., f_U(\xi_N))^T$, by stacking part feature vectors. The feature vector for the i-th part, $f_U(\xi_i)$, is computed from a deep convolutional network, extracted around coordinate, ξ_i. We use the DeepLab Network [Chen 2016b], a CNN trained for semantic segmentation, and we extract the activations in the 3-rd and 5-th layer of the network. This rich unary feature vector, $f_U(\xi_i)$, has $D = 769$ dimensions for each part position ξ_i.

The pairwise features are computed as $f_{\mathcal{P},v} = (\xi_i - \xi_j - R_v \mu_{i,j})^2$, $(i,j) \in E$, with R_v being the rotation matrix for the v-th viewpoint bin. These geometric features describe the squared difference of the location of part j and its nominal position in relation to part i. The inner product, $\langle f_{\mathcal{P},v}, w_{\mathcal{P}_v} \rangle$, computes the sum of pairwise terms in the DPM score function in Equation 3.1, $\sum_{(i,j)\in E} \mathcal{P}_{i,j,v}(\xi_i, \xi_j)$, with $w_{\mathcal{P}_v}$ being the stacked precision matrices, $C_{i,j,v}$, This results in a $3(N-1)$-dimensional weight vector for the pairwise term, $w_{\mathcal{P}_v}$.

We train this linear classifier with SSVM as described in Section 2.3.2. Let $\Xi = (\xi_1, ..., \xi_N)$ and $\bar{\Xi} = (\bar{\xi_1}, ..., \bar{\xi_N})$ be the ground truth object configuration and an arbitrary object configuration, respectively. We replace the loss function proposed in Equation 2.25 by the mean euclidean distance, $\Delta(\Xi, \bar{\Xi}) = \frac{1}{N} \sum_{i=1}^{N} ||\xi_i - \bar{\xi_i}||_2$. It penalizes how much the predicted 3D coordinates deviate from the ground truth position of the associated keypoints. Additionally, this loss decomposes over object parts, leveraging the tractability of the loss augmented inference during training (Equation 3.9).

The most violated constraint is found by solving the optimization problem

$$\bar{\Xi}^k = \arg\max_{\bar{\Xi}} S_v^k(\bar{\Xi}) + \Delta(\Xi, \bar{\Xi}), \tag{3.8}$$

where S_v^k is the score function based on the weight vector $\mathbf{w}_v^k$ of iteration, k. This

corresponds to the update step for the working set in Equation 2.29 of the SSVM algorithm. Intuitively, the label configuration Ξ^k receives a high score with the model of iteration k, but causes a high loss and is added to the working set of constraints for the $(k+1)$-th iteration.

The problem is equivalent to maximizing the score of the 3D DPM (Equation 3.1) with modified unaries.

$$\tilde{S}_v(\bar{\Xi}) = \sum_{i=1}^{N} \left(\mathcal{U}_{i,v}(\bar{\xi}_i) + \delta(\xi_i, \bar{\xi}_i) \right) + \sum_{(i,j)\in E} \mathcal{P}_{i,j,v}(\bar{\xi}_i, \bar{\xi}_j), \tag{3.9}$$

where the partial loss $\delta(\xi_i, \bar{\xi}_i) = (1/N)||\xi_i - \bar{\xi}_i||_2$, incurred by the respective labeling, is added to the unary potential.

Thus the complexity of calculating the most violated constraint is identical to the complexity of our inference algorithm. We note that this is possible because the loss function decomposes over object parts, allowing for efficient optimization of Equation 3.9. With this, the joint training of unary and pairwise terms is carried out within the tractable time of a few hours per model and viewpoint.

3.3 3D DPM - Inference

In the last section we have introduced and trained a 3D viewpoint-specific mixture model. The optimization problem is very similar to the well-known DPM optimization, presented in Section 2.2.1, but the higher dimensionality renders these approaches inefficient. We present an efficient inference method for the 3D optimization problem by customizing Branch-and-Bound to the given problem. We then show that there is a set of equivalent solutions to this problem, and we can further accelerate Branch-and-Bound by committing all computational resources to exactly one of the equivalent solutions.

3.3.1 Joint Inference with Depth Variables

In this section we delineate our algorithmic approach to solving the optimization problem, $\arg\max_{\xi_1,\ldots,\xi_N} S(\xi_1, \ldots, \xi_N)$, defined in Equation 3.1. We first show a straight forward adaption of Dual-Tree Branch-and-Bound (DTBB, Section 2.2.3) to the 3D problem. Then we customize the DTBB, considering the problem's special structure to save computational resources. This approach renders memory requirements for data structures independent of depth resolution.

We solve a 3D optimization problem where each part can take a value in $\Omega_{3D} = (1, \ldots, w) \times (1, \ldots, h) \times (1, \ldots, d)$, with w and h being the width and the height of the input image, respectively. The maximum depth, d, is set to encompass the depth of an object within the image.

A straight forward approach to solve inference of a 3D DPM is to extend the 2-dimensional unaries to 3D by stacking d replicas of the 2D unary array to yield a 3-dimensional unary. Then one treats the depth variables ξ_z as another dimension of

the variables ξ and constructs a 3 dimensional Kd-Tree upon the now 3-dimensional unaries. The inference follows Algorithm 1.

```
1: function TraverseTree
2:    PQ := priority queue
3:    PQ ← (Ω_3D, 0.0)
4:    while NOT isEmpty(PQ) do
5:       ν ← PQ.pop()
6:       if |ν| == 1 then
7:          return ν
8:       else
9:
10:         ν_left ← ν.leftChild
11:         ν_right ← ν.rightChild
12:         PQ ← (ν_left, S̄(ν_left))
13:         PQ ← (ν_right, S̄(ν_right))
14:
15:
16:
17:
18:
19:
20:      end if
21:   end while
22:   return { }
23: end function
```

Algorithm 1: Tree Traversal in Dual-Tree Branch-and-Bound. The interval, ν, is a 3-dimensional cube in the label space, Ω_{3D}. This requires the creation of a Kd-Tree on a 3-dimensional volume.

```
1: function TraverseWithDepthTree
2:    PQ := priority queue
3:    PQ ← ((Ω_2D, Ω_z), 0.0)
4:    while NOT isEmpty(PQ) do
5:       (ν, κ) ← PQ.pop()
6:       if (ν, κ) are leaf nodes then
7:          return (ν, κ)
8:       else
9:          if |ν| ≥ |κ| then
10:            ν_left ← ν.leftChild
11:            ν_right ← ν.rightChild
12:            PQ ← ((ν_left, κ), S̄(ν_left, κ))
13:            PQ ← ((ν_right, κ), S̄(ν_right, κ))
14:         else
15:            κ_left ← κ.leftChild
16:            κ_right ← κ.rightChild
17:            PQ ← ((ν, κ_left), S̄(ν, κ_left))
18:            PQ ← ((ν, κ_right), S̄(ν, κ_right))
19:         end if
20:      end if
21:   end while
22:   return { }
23: end function
```

Algorithm 2: Tree Traversal with a separated Kd-Tree for depth variables. The interval, ν, corresponds to a 2-dimensional rectangle in the x and y dimension. The interval, κ, represents the z-dimension in a 1-dimensional interval. By treating these dimensions separately, it is enough to create a 2D and a 1D Kd-Tree and combine them to 3D intervals on the fly without explicitly instantiating a 3D volume.

We explain Algorithm 1 line by line. In line 3, we initialize the priority queue, PQ, with the solution space as a whole. This priority queue, PQ, sorts the contained intervals by their upper bound in descending order, thus retrieving first the most promising candidates. Line 4 enters a loop that continues until the priority queue, PQ, is empty or the algorithm terminates with a solution. In line 5, the top scoring

interval is popped from the priority queue, PQ, and stored as interval, ν. Line 6 asks if the interval, ν contains only a single element. If true then this is the global solution and is returned in line 7. Otherwise each iteration executes now the two Branch-and-Bound steps: Firstly, in line 10 and 11 the interval, ν, is split into two sub-intervals, ν_{left} and ν_{right}, in the (**branch**-step). This corresponds to traversing from the node ν to its left and right children in the Kd-Tree. Secondly, in line 12 and 13 an upper bound of the score of both intervals, ν_{left} and ν_{right}, is estimated (**bound**-step), and the sub-spaces are inserted into the priority queue, PQ, with their upper bound. The computation of the upper bound of intervals is detailed in Section 3.3.1.1.

We analyze this approach: The memory required for the Kd-Tree is in $\mathcal{O}(w{*}h{*}d)$, since the 3D volume is explicitly represented. Analogously, the runtime required to construct the Kd-Tree is $\mathcal{O}(w{*}h{*}d)$. In both aspects our proposed algorithm is a factor of d faster and requires d times less memory.

Our approach is more efficient in both memory and computational requirements. The key idea is to separate the depth dimension from the 2 image dimensions, avoiding to instantiate a 3 dimensional data structure. To this end we create a Kd-Tree for the 2D unary potentials and an additional Kd-Tree over the one-dimensional depth domain, $\Omega_z = \{1, ..., d\}$. In the branch-step of the Branch-and-Bound, we traverse either the 2-dimensional Kd-Tree for the unaries or the 1-dimensional Kd-Tree for the depth.

We explain Algorithm 2 in detail. We denote intervals in the unary Kd-Tree with ν and intervals in the separate Kd-Tree for depth with κ. The tuple of both correspond to a 3D interval. Similar to Algorithm 1, line 3 initializes the priority queue, PQ, with the whole solution space, described by the tuple of root nodes of the 2D and 1D Kd-Tree. Line 4 enters a loop that continues until the priority queue, PQ, is empty or the algorithm terminates with a solution. In line 5, the top scoring 3D interval is popped from the priority queue, PQ, and stored in the tuple, (ν, κ). Again, as in Algorithm 1, this interval is the solution if it contains only one element (line 6) and is returned (line 7). Otherwise the iteration executes now the two Branch-and-Bound steps:

Branch: To split the interval (ν, κ) into two sub-intervals, similarly to Algorithm 1, we test which dimension is the largest one in line 9. Depending on this, the interval ν or the interval κ is split in line 10 and 11 or line 15 and 16, respectively. This corresponds to descending in the unary Kd-Tree or the depth Kd-Tree to its left and right children, respectively.

Bound: In line 12 and 13 the upper bound of the score of both sub-intervals, $(\nu_{\text{left}}, \kappa)$ and $(\nu_{\text{right}}, \kappa)$, is estimated and the intervals are inserted into the priority queue, PQ, with their respective upper bound. The computation of the upper bound of intervals is identical to Algorithm 1 and is detailed in Section 3.3.1.1. Line 17 and 18 function analog for the depth Kd-Tree.

We analyze the presented Algorithm 2. The memory required for the Kd-Tree is in $\mathcal{O}(w{*}h + d)$, because the 3D volume is represented in a 2D and a 1D Kd-Tree. Analogously, the runtime required to construct the Kd-Trees is $\mathcal{O}(w{*}h + d)$. The

summand d can be canceled out, assuming w and h to be in the same order of magnitude as d. Thus our approach is a factor of d more efficient and requires d-times less memory.

3.3.1.1 Upper Bound Computation

In this section the computation of the upper bound of the score over intervals of part positions is detailed. This follows the original DTBB [Kokkinos 2011]. To this end a geometric bound is derived for pairwise potentials.

In lines 12 and 13 of Algorithm 1 we search to compute an upper bound, $\bar{S}(\nu)$, of the exact maximum score, $\max_{\xi_1,...,\xi_N} S(\xi_1,...,\xi_N)$, with ξ_1 being in an interval, ν. The same problem is solved in Algorithm 2, lines 12, 13, 17 and 18. An upper bound can be estimated efficiently by applying

$$\max_{\xi_1,...,\xi_N} S(\xi_1,...,\xi_N) = \max_{\xi_1 \in \nu} \left(\mathcal{U}_1(\xi_1) + \sum_{j=2}^{N} \max_{\xi_j} S_{j\xi_1}(\xi_j)\right) \tag{3.10}$$

$$\leq \max_{\xi_1 \in \nu} \left(\mathcal{U}_1(\xi_1)\right) + \sum_{j=2}^{N} \max_{\xi_1 \in \nu} \max_{\xi_j} S_{j\xi_1}(\xi_j) \tag{3.11}$$

$$= \max_{\xi_1 \in \nu} \left(\mathcal{U}_1(\xi_1)\right) + \sum_{j=2}^{N} \max_{\xi_j} \bar{S}_{j\nu}(\xi_j) \tag{3.12}$$

where $\bar{S}_{j\nu}(\xi_j) = \max_{\xi_1 \in \nu} S_{j\xi_1}(\xi_j)$ is the best score the part position ξ_j can achieve when the center node lies within interval ν. The bound over the unary potential, $\max_{\xi_1 \in \nu} \mathcal{U}_1(\xi)$, is precomputed in a Kd-Tree. The upper bound (Equation 3.11) decouples the choice of ξ_1 in its unary term, $\mathcal{U}_1$, from the computation of the maximum part contributions $\bar{S}_{j\nu}$, conditioned on the interval, ν, instead of the exact choice of ξ_1. Thus an upper bound over the score in Equation 3.10 can be computed separately by parts, allowing for efficient computation. The Inequality 3.11 is exact if the interval, ν, consists of only one single element.
The part contribution can be upper-bounded itself as

$$\max_{\xi_j} \bar{S}_{j\nu}(\xi_j) = \max_{\xi_j} \left(\mathcal{U}_j(\xi_j) + \max_{\xi_1 \in \nu} \mathcal{P}_{1,j}(\xi_1, \xi_j)\right) \tag{3.13}$$

$$\leq \max_{\upsilon \in Y} \left(\max_{\xi_j \in \upsilon} \mathcal{U}_j(\xi_j) + \max_{\xi_j \in \upsilon} \max_{\xi_1 \in \nu} \mathcal{P}_{1,j}(\xi_1, \xi_j)\right) \tag{3.14}$$

where Y is a partitioning of the solution space Ω. In particular, every level of a Kd-Tree gives a partition of the solution space, Ω. Again the upper bound decouples the unary from the pairwise term, but limits the variable, ξ_j, to lie within the same partition, υ. The inequality in Equation 3.13 becomes equality when the partitioning contains only individual elements, $|\upsilon| = 1$, $\forall \upsilon \in Y$ and $|Y| = |\Omega|$. The size of the partitions of Y determine the tightness of the upper bound, $\bar{S}_{j\nu}$. The smaller are the partitions of Y, the tighter is the estimation, but the more costly is its computation. In order to balance both factors, the size of partitions, υ, is

aligned with the size of the interval ν. In other words, the DTBB descends in the Kd-Tree of parts $2, ..., N$ in parallel to the descent in the primary Kd-Tree (line 10 and 11, 15 and 16, Algorithm 2) once per Branch-and-Bound iteration.

We detail the computation of the upper bound (Equation 3.14) of the pairwise term $\bar{\mathcal{P}}_{i,j}(\xi_i, \xi_j)$, with $\xi_i \in ([x_i - x_i^s; x_i + x_i^s], [y_i - y_i^s; y_i + y_i^s], [z_i - z_i^s; z_i + z_i^s])^T$ lying in an interval, Γ_i, of the i-th Kd-Tree and $\xi_j \in ([x_j - x_j^s; x_j + x_j^s], [y_j - y_j^s; y_j + y_j^s], [z_j - z_j^s; z_j + z_j^s])^T$ lying in an interval, Γ_j, of the j-th Kd-Tree. Please refer to Figure 3.4 for an illustration. We drop part indices from the offset $\mu_{i,j}$ and precision matrix $C_{i,j}$ for better readability, as there is no possible confusion.

$$\bar{\mathcal{P}}_{i,j}(\xi_i, \xi_j) = \max_{\xi_i \in \Gamma_i} \max_{\xi_j \in \Gamma_j} -(\xi_i - \xi_j - \mu)^T C (\xi_i - \xi_j - \mu) \tag{3.15}$$

$$= \max_{\xi_i \in \Gamma_i} \max_{\xi_j \in \Gamma_j} -(\xi_i - \xi_j - \mu)^T C (\xi_i - \xi_j - \mu) \tag{3.16}$$

$$= - \min_{\substack{\xi_{i_x} \in \Gamma_{i_x} \\ \xi_{j_x} \in \Gamma_{j_x}}} (\xi_{i_x} - \xi_{j_x} - \mu_x)^2 C_x - \min_{\substack{\xi_{i_y} \in \Gamma_{i_y} \\ \xi_{j_y} \in \Gamma_{j_y}}} (\xi_{i_y} - \xi_{j_y} - \mu_y)^2 C_y - \min_{\substack{\xi_{i_z} \in \Gamma_{i_z} \\ \xi_{j_z} \in \Gamma_{j_z}}} (\xi_{i_z} - \xi_{j_z} - \mu_z)^2 C_z. \tag{3.17}$$

We simplify this formulation by replacing $(\xi_j + \mu)$ by $\xi_{j\mu}$ and accordingly shift the interval Γ_j with center (x_j, y_j, z_j) by μ, yielding interval $\Gamma_{j\mu}$ with center $(x_{j\mu}, y_{j\mu}, z_{j\mu}) \coloneqq (x_j, y_j, z_j) + (\mu_x, \mu_y, \mu_z)$ (compare Figure 3.4):

$$= - \min_{\substack{\xi_{i_x} \in \Gamma_{i_x} \\ \xi_{j\mu_x} \in \Gamma_{j\mu_x}}} (\xi_{i_x} - \xi_{j\mu_x})^2 C_x - \min_{\substack{\xi_{i_y} \in \Gamma_{i_y} \\ \xi_{j\mu_y} \in \Gamma_{j\mu_y}}} (\xi_{i_y} - \xi_{j\mu_y})^2 C_y - \min_{\substack{\xi_{i_z} \in \Gamma_{i_z} \\ \xi_{j\mu_z} \in \Gamma_{j\mu_z}}} (\xi_{i_z} - \xi_{j\mu_z})^2 C_z \tag{3.18}$$

$$= -(\min_{\substack{\xi_{i_x} \in \Gamma_{i_x} \\ \xi_{j\mu_x} \in \Gamma_{j\mu_x}}} |\xi_{i_x} - \xi_{j\mu_x}|)^2 C_x - (\min_{\substack{\xi_{i_y} \in \Gamma_{i_y} \\ \xi_{j\mu_y} \in \Gamma_{j\mu_y}}} |\xi_{i_y} - \xi_{j\mu_y}|)^2 C_y - (\min_{\substack{\xi_{i_z} \in \Gamma_{i_z} \\ \xi_{j\mu_z} \in \Gamma_{j\mu_y}}} |\xi_{i_z} - \xi_{j\mu_z}|)^2 C_z \tag{3.19}$$

$$\begin{aligned} = &- \max(0, |x_i - x_{j\mu}| - (x_i^s + x_j^s))^2 C_x \\ &- \max(0, |y_i - y_{j\mu}| - (y_i^s + y_j^s))^2 C_y \\ &- \max(0, |z_i - z_{j\mu}| - (z_i^s + z_j^s))^2 C_z. \end{aligned} \tag{3.20}$$

The last reformulation can be visualized using Figure 3.4. The square term is minimized when both coordinates, ξ_i and $\xi_{j\mu}$, take on values as close as possible to each other within their respective intervals. This is calculated as the absolute difference of interval centers, (x_i, y_i, z_i) and $(x_{j\mu}, y_{j\mu}, z_{j\mu})$, minus the sum of the intervals' half sizes, x_i^s, y_i^s, z_i^s and x_j^s, y_j^s, z_j^s. If this term is negative in a dimension, then this dimension's contribution is 0, as in the y-dimension in the example of Figure 3.4.

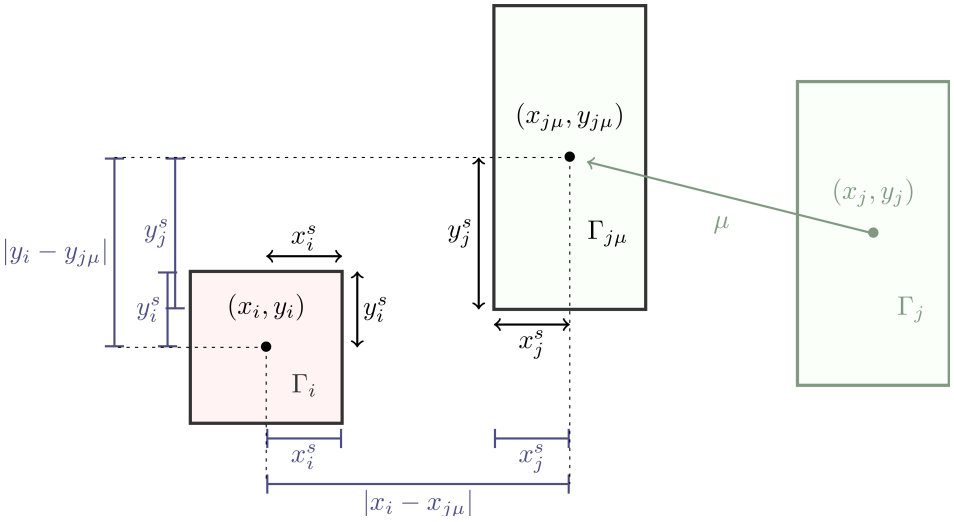


Figure 3.4: Illustration of the calculation of the pairwise term, $\bar{\mathcal{P}}_{i,j}(\xi_i, \xi_j)$. For better readability the third dimension is not shown. The interval, Γ_i, is given as the center (x_i, y_i, z_i) and the half size (x_i^s, y_i^s, z_i^s) (light red color). The interval, Γ_j, is given as the center (x_j, y_j, z_j) and the half size (x_j^s, y_j^s, z_j^s) (light green color). We subsume the nominal offset, μ, under the interval, $\Gamma_{j\mu}$, with the center being $(x_j + \mu_x, y_j + \mu_y, z_j + \mu_z)^T$. This corresponds to a shift of the interval, Γ_j, by μ. We illustrate the individual terms of Equation 3.20 in blue.

3.3.2 Anchoring in Depth

We can accelerate inference by recognizing the special structure of the presented optimization problem. The problem's score is invariant to global shifts in the depth dimension. Therefore anchoring the model's center node at a certain depth avoids exploring equivalent solutions and accelerates convergence.

The ambiguity between scale and depth is well known in 3D reconstruction problems. In the monocular setting, distinguishing if an object is small in the image because it is far away from the camera or because the object is small by itself, is a challenging task. This ambiguity has another consequence for our approach. Due to the assumed orthographic camera projection, objects will appear the same and have the same size regardless of depth. This is reflected in the DPM score for 3D with depth variables. The detection score is determined according to Equation 3.1. We show that we can add a constant offset, o, in depth and still yield the same score. Since the unary term is invariant to depth, it is sufficient to consider at the pairwise term

$$\mathcal{P}_{i,j}(\begin{pmatrix} \xi_{i_x} \\ \xi_{i_y} \\ \xi_{i_z}+o \end{pmatrix}, \begin{pmatrix} \xi_{j_x} \\ \xi_{j_y} \\ \xi_{j_z}+o \end{pmatrix}) = \begin{pmatrix} (\xi_{i_x}-\xi_{j_x}-\mu_{i,j_x})^2 \\ (\xi_{i_y}-\xi_{j_y}-\mu_{i,j_y})^2 \\ ((\xi_{i_z}+o)-(\xi_{j_z}+o)-\mu_{i,j_z})^2 \end{pmatrix}^T \begin{pmatrix} C_{i,j_x} \\ C_{i,j_y} \\ C_{i,j_z} \end{pmatrix} \tag{3.21}$$

$$= \mathcal{P}_{i,j}(\begin{pmatrix} \xi_{i_x} \\ \xi_{i_y} \\ \xi_{i_z} \end{pmatrix}, \begin{pmatrix} \xi_{j_x} \\ \xi_{j_y} \\ \xi_{j_z} \end{pmatrix}). \tag{3.22}$$

Therefore the pairwise term decomposes over the variables' dimensions, because the coefficient matrix, $C_{i,j}$, is assumed to be diagonal. This yields Equation 3.21. We can see that the offset o cancels out. This implies that relative depth is independent of absolute depth.

The optimization problem in 3.1 may have multiple equivalent solutions, indistinguishable by score, given more available depth layers than the model occupies. Our Branch-and-Bound approach would explore all intervals containing one of the solutions. There are two main ideas to consider. Firstly, reduce the available depth range. This is ideal only if the object is represented exactly within this depth range. Secondly, we propose to provide ample depth layers to the algorithm, but fix the center node's depth to a depth ϕ, for example, to $\phi = \Omega_z/2$. This eliminates all equivalent solutions except one, where $\xi_{1_z} = \phi$. The solutions where the center node is not at the anchor depth will yield low upper bounds in the first iterations, narrowing down the further considered label space in the depth dimension to an adequate size.

3.4 Evaluation

We demonstrate the validity of our approach and the efficiency of our implementation for 3D object detection on the challenging PASCAL3D+ dataset [Xiang 2014]. From 2D images of objects like *cars*, *bicycles*, etc., we infer the object's 3D keypoints. The result resembles a sparse 3D object reconstruction. We analyze the quantitative and qualitative performance, as well as the runtime behaviour with respect to an increasing number of depth labels.

3.4.1 Model Validation

The PASCAL3D+ dataset [Xiang 2014] is an extension to the challenging PASCAL VOC 2012 dataset [Everingham 2010] and contains 3D object annotation for 12 object classes. We follow the practice of [Kar 2015], using a test set of 1408 images out of 10 categories: *plane*, *bike*, *boat*, *bus*, *car*, *chair*, *sofa*, *train*, *tv-monitor* and *motor-bike*. However, the initial NRSfM-step failed for the "sofa" category due to the scarcity of training images, so we removed it from all evaluation. For all classes, the best matching 3D CAD model and the best viewpoint alignment are given. We use the viewpoint-aligned 3D CAD models for evaluation only. Furthermore every object instance is annotated with an occlusion label and comes with between 7 and 16 manually annotated 2D keypoints.

As metric to evaluate our approach, we compute the Hausdorff Distance d_H [Huttenlocher 1993], following [Kar 2015]:

$$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\}, \tag{3.23}$$

where X and Y are sets of points on the surface of the ground truth solution and the inferred shape, respectively, and $d(x,y)$ the euclidean distance. The Hausdorff

metric punishes the largest minimum distance between any points on the surface of the two shapes. This is illustrated in Figure 3.5. The greatest distance from any point on one shape to the closest point on the other shape is measured for both shapes X and Y.

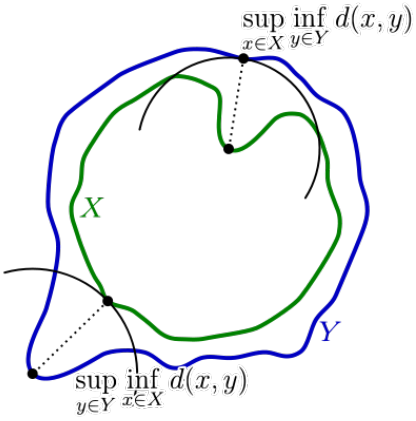


Figure 3.5: Illustration of the Hausdorff Distance between two shapes X and Y. The Hausdorff Distance is computed as the maximum between the smallest distance from a point on X to any Y such that no other point on X would yield a larger distance, and its symmetric counterpart with X and Y exchanged. Image from [Commons 2007].

We use a viewpoint prediction system at test time to choose a viewpoint bin [Bregler 2000]. An alternative is to test all viewpoint models on the input image and chose the best performing one. This would increase the runtime proportionally to the number of viewpoint bins.

We evaluate our approach with the Hausdorff metric and verify that it yields competitive results. To this end, we compute the Hausdorff distance (Equation 3.23) between the predicted mesh and the ground truth mesh, defined by the annotated CAD model. We show these results in Table 3.2. Additionally we test our approach in a multi-scale fashion: We create an image pyramid with 3 levels by re-scaling the input image. Then we run our algorithm against this pyramid. We show in Table 3.2 that this improves reconstruction quality, indicating a certain scale sensitivity of our model. In most categories we are competitive to [Kar 2015]. Our keypoint model is not very articulated, resulting in a lower score than the one obtained in the competing approach. In average over all categories we achieve improved results (3.81) over the competition (4.03) in the Hausdorff metric.

We present qualitative results of our object detection algorithm for a selection of instances of the PASCAL3D+ dataset in Figure 3.6. Some images are displayed in a low resolution, as they are scaled up and cropped to adequately show the small objects within. The estimated depth is encoded in color with yellow representing close surfaces and blue signifying large distance to the camera. We aid the viewer's depth perception by showing the surface's triangle mesh in thin blue lines. The transparent black lines within the detected structure highlight the model's star-

	plane	bicycle	boat	bus	car
Kar *et al.* [Kar 2015]	**2.2**	4.4	6.0	3.9	3.2
ours	2.4	4.1	6.1	4.1	**3.1**
ours multiscale	2.4	**4.0**	**5.7**	**3.8**	**3.1**
	chair	**train**	**tv-monitor**	**motorbike**	**mean**
Kar *et al.* [Kar 2015]	**2.6**	6.6	4.5	**2.9**	4.03
ours	3.2	6.3	3.4	3.0	3.97
ours multiscale	3.0	**6.0**	**3.3**	3.0	**3.81**

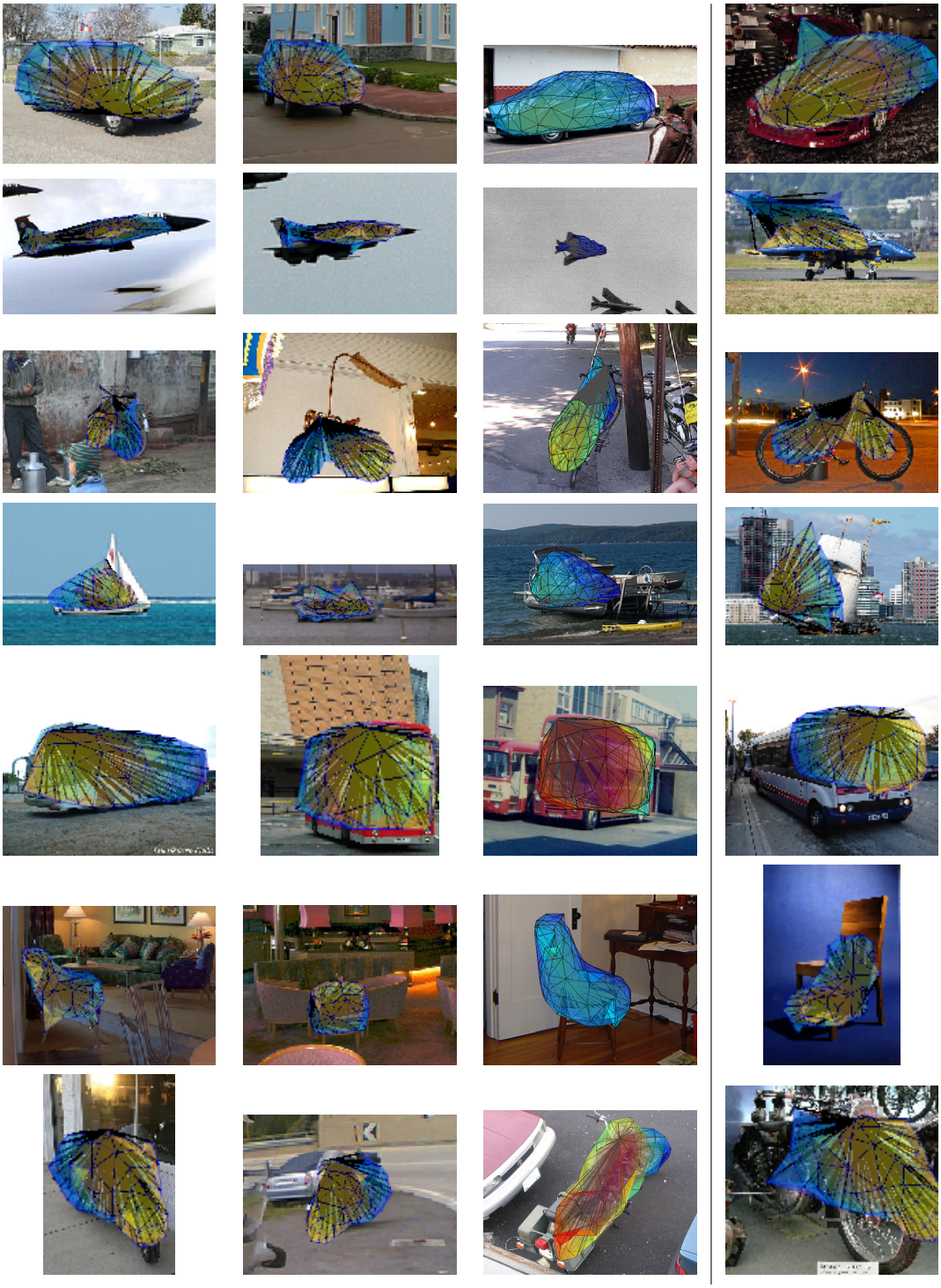
Table 3.2: The Hausdorff error with respect to the centered ground-truth CAD models provided in PASCAL3D+ (lower is better). We show our 9 categories and the mean over those categories.

shaped graph structure. In every row, the first three columns display positive examples, whereas in the fourth column we show a failure case for each category. Failure cases arise due to mainly three reasons:

- **Low resolution** Keypoints cannot reliably be detected below a certain spatial resolution. We find examples of this in the *tv-monitor* and the *plane* category, as they often appear far in the background or at large distances. The failure case of *tv-monitor* in Figure 3.6 serves as example.

- **Extreme perspective** Images of the PASCAL3D+ dataset are taken from all possible perspectives. While we explicitly deal with viewpoint rotation in azimuth, rare camera angles and extreme close-up views can not be adequately handled. We attribute difficulties with objects close to the camera largely to projective distortion. As example we consider the failure cases for *cars*, *trains* and *motorbike* in Figure 3.6.

- **Large intra-class variation** The PASCAL3D+ dataset is challenging because of its high variability even within one object class. The *boat* category, for example, contains sailing boats as much as steamboats and rowing boats, and the *chair* category extends from armchairs to folding chairs. These instances are dissimilar not just visually, but also by structure. The failure cases in the *boat* and *chair* categories in Figure 3.6 are examples of this. To improve performance a more diverse set of object models would be necessary: for instance the boat category has diverse shapes, including steam boats and rowing boats.



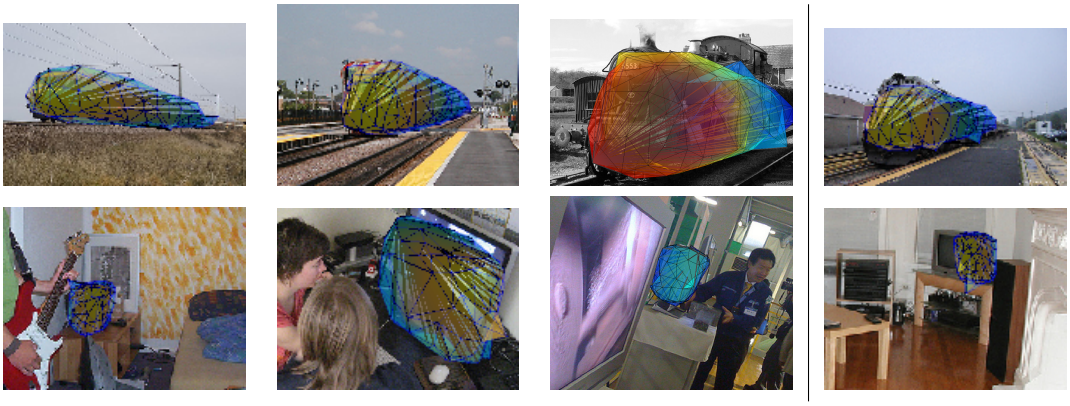


Figure 3.6: Example images of the PASCAL3D+ dataset with object detections. We have chosen 3 positive examples per category in the first three columns and one failure case in the fourth column. Yellow indicates small distance to the camera, whereas blue signifies a large distance.

3.4.2 Runtime Performance

Firstly, we compare the runtime of two alternative implementations: A GDT-based implementation in the spirit of Felzenszwalb et al. [Felzenszwalb 2010b] and our Branch-and-Bound approach. Figure 3.7 illustrates the scalability of our approach (yellow line) in comparison to the long time "standard"-solution using GDT and dynamic programming (blue line).

Using a single depth layer, both approaches perform almost identically. With an increasing number of depth layers, the runtime required for the GDT-based approach increases linearly, whereas the Branch-and-Bound approach remains sub-linear. We notice that beyond 150 depth layers the increase of runtime in our approach is marginal. Keypoints beyond a certain depth are punished by the pairwise term for being too far from the depth that has been determined as anchor depth. Consequently further depth layers will be largely ignored by Branch-and-Bound. Already with a moderate amount of 100 discrete depth values available to the algorithm, our approach (yellow line) is about two orders of magnitude faster than the GDT-based solution (blue line). We also observe that the memory consumption for GDT is directly proportional to the number of depth layers, but our implementation of Branch-and-Bound with depth separate from the unary potential retains a constant memory footprint.

Secondly, we show the benefits of anchoring the solution at a certain depth, as described in Section 3.3.2. We measure the influence on performance for a variety of available depth layers Ω_z between 120 and 400, and compare to the runtime without anchoring the central node's depth (Figure 3.8). We observe that runtime without anchoring increases almost linearly with a number of depth layers larger than $\sim$100, the ground truth depth range. This confirms that indeed all equal solutions are explored. In contrast, the runtime of Branch-and-Bound with anchoring increases only marginally with the number of depth layers.

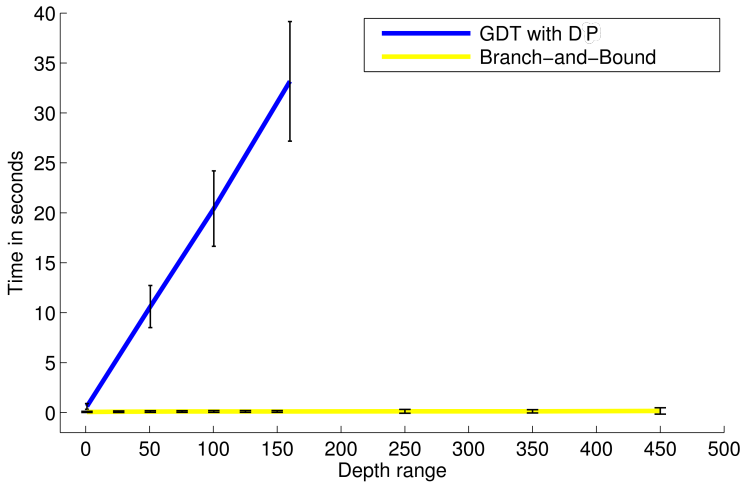


Figure 3.7: Runtime comparison between the GDT-based optimization in blue and the Branch-and-Bound based optimization in yellow (proposed). The x-axis gives the number of discrete depth values available in the solution space. The y-axis shows the resulting runtime in seconds. The computation with GDT for a depth resolution greater than 300 was impossible due to memory limits.

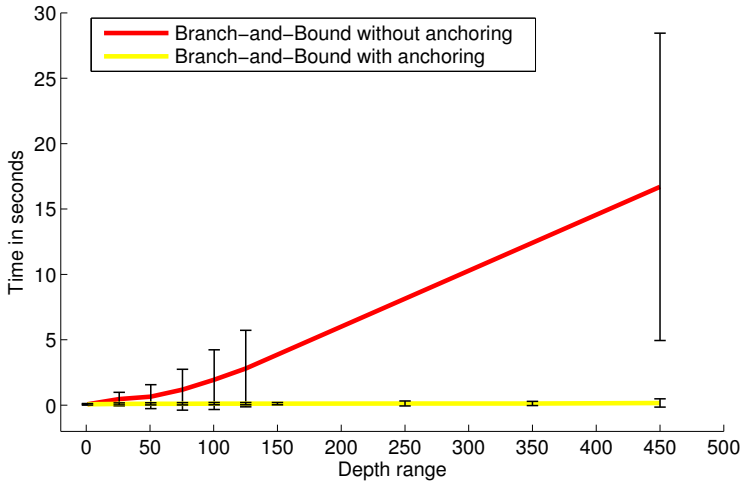


Figure 3.8: Acceleration of Branch-and-Bound by eliminating equivalent solutions by anchoring the model at a certain depth. The ground truth's depth range is between 80 and 120. The x-axis gives the number of discrete depth values available in the solution space. The y-axis shows the resulting runtime in seconds. The red curve shows runtime for Branch-and-Bound without anchoring the center node's depth. The yellow curve shows performance with anchoring of the center node's depth in the middle of the available depth space.

Chapter 4

Monocular 3D Human Pose Estimation

Contents

4.1 Prior Work in Human Pose Estimation 70
4.2 DPM and Inference in Higher Dimensions 73
4.2.1 Deformable Part Models in 3D 73
4.2.2 Inference with Dual-Tree Branch-and-Bound in 3D 74
4.3 Training with Deep Supervision 74
4.4 DPM Inference on Arbitrary Graph Topologies 76
4.4.1 Graph Decomposition with ADMM 78
4.4.2 Inference of ADMM-Augmented Subproblem 82
4.4.3 Inference Visualization 83
4.5 Results on 3D Human Pose Estimation 90
4.5.1 Evaluation Setup 90
4.5.2 Implementation Details 90
4.5.3 Quantitative Comparison 91
4.5.4 Ablation Study with Graph Topologies 91
4.5.5 Improvement of 2D Joint Localizations 99
4.5.6 Qualitative Evaluation on the Leeds Sports Dataset 100
4.5.7 Runtime 101

3D human pose estimation is an important building block for human-computer interfaces and other real-world applications. The adequate representation of the human body is challenging due to the increased complexity that comes with more realistic models, for example in the form of interconnected body joint locations in 3D. The computational burdens can be reduced by breaking down the problem into a 2D and then 3D problem, but leads to inferior detection precision. This chapter explores further the DPM in 3 dimensions, its efficient inference with Branch-and-Bound and its extension to arbitrary, especially loopy graph topologies. In [Kinauer 2017] we apply these techniques to 3D human pose estimation and show state of the art results on the Human3.6M dataset. To this end, we harvest the power of deep neural networks to generate the parameters of an optimization problem. The result is a combination between state of the art CNN methods and a discrete optimization stage to refine results and resolve eventual ambiguities.

4.1 Prior Work in Human Pose Estimation

Human pose estimation from monocular images is a well researched topic in computer vision, especially in contrary to pose estimation of inanimate objects. The great number of works on human pose estimation is consequent because humans are more articulated and more frequently deformed than most objects in daily life. We focus on the development from 2D to 3D human pose estimation and the two major challenges that come with it: firstly, the question of how to represent the human body and encode 3D priors into the models, and secondly, the question of how to handle the computational demands of algorithms in 3D. We start with 2D human pose estimation, then discuss, as in-between step, 3D human pose estimation as lifting of 2D detections into 3D, and finally talk about full 3D approaches.

2D Human Pose Estimation Human Pose Estimation is most commonly phrased as the location of a number of keypoints, usually corresponding to body joints, like the elbows, the wrists, the neck, the hips, the knees and the feet. The research community has developed models with a varying number of model nodes, mostly between 14 and 27 [Andriluka 2014, Ladicky 2013, Pishchulin 2012, Ramanan 2007, Johnson 2010, Toshev 2014, Pishchulin 2016, Insafutdinov 2016, Newell 2016, Wei 2016, Chen 2014, Yang 2011, Yang 2013] in order to represent the human body well without over-parameterizing it. We follow this trend and use a set of 16 body joints, aligning with [Andriluka 2014]. The works of [Bogo 2016, Lassner 2017] go a step further to reconstruct a full body model that includes the body shape.

The 2D human pose estimation problem has been solved efficiently and with convincing results even under challenging circumstances like "in-the-wild" images with multiple persons and in a wide range of poses, like in [Insafutdinov 2016] and [Cao 2016] as state of the art. 2D human pose estimation has improved considerably through the use of neural networks, similarly to the case of object detection (Section 2.1.3). State of the art works without neural networks [Andriluka 2014, Yang 2013] correctly detect about 44% of body joints on the MPII-Human pose dataset [Andriluka 2014]. One year later the work of [Tompson 2015] almost doubled performance, correctly detecting 82% of body joints by harnessing the power of CNNs. State of the art works achieve 91.6% in [Güler 2016] and 92.1% in [Ke 2018] of correctly detected body joints on the MPII benchmark.

2D Human Pose Estimation with Lifting While 2D human pose estimation has attained a high level of precision and reliability, the arguably more difficult 3D task has not yet reached this level of success. In this section we describe the transition from purely 2D approaches towards 3D approaches.

To transfer the success of 2D to 3D pose estimation, the 3D problem can be treated as a 2D problem that is lifted into the third dimension in a subsequent step. This has two advantages: Firstly, it avoids the computationally heavy burden of dealing with 3 dimensions at once. Secondly, it al-

lows to easily exploit large datasets with 2D landmark annotations, for example [Andriluka 2014, Johnson 2010, Ferrari 2008, Dantone 2013]. 3D annotated datasets are difficult to obtain with current outdoor motion capture systems [Vlasic 2007, Ionescu 2011, Pons-Moll 2011] and are hence scarce (H3D [Bourdev 2009]) or of limited variation due to laboratory environments (HumanEva [Sigal 2010] and Human3.6M datasets [Ionescu 2014]).

The lifting step requires to add prior knowledge in the form of constraints to the originally ill-posed problem in order to obtain a plausible solution. We present different ideas that incorporate a 3D prior into the lifting step. These models come mainly in two flavors: exemplar-based and as basis shapes.

The **exemplar approach** is simple: the best matching 3D exemplar is found and aligned to 2D joint detections. [Jiang 2010, Yasin 2016, Chen 2016a] use this approach to lift 2D detections to 3D. The method to generate 2D detections from the image can vary, [Yasin 2016] use a random forest in combination with a DPM like energy, [Chen 2016a] apply a CNN to obtain 2D joint positions. [Jiang 2010] assume 2D joint locations to be given. All three methods project a large number of 3D exemplars into 2D and use a slightly modified k-nn algorithm to determine the best match. A final optimization or warping procedure refines the result in order to better match the image evidence. In [Jiang 2010] the number of exemplars is extended by combining upper and lower body parts from different poses. The result is a likely human pose lying in the set of poses in the training data. [Chen 2016a] reach a performance only slightly inferior to [Zhou 2016], discussed in the next paragraph.

Basis-based methods, for example [Ramakrishna 2012, Zhou 2015, Akhter 2015, Zhou 2016], form a more compact model representation by summarizing a large number of poses in a small number of bases. New, unseen poses are generated as linear combinations of these basis shapes. An optimization scheme is in place to align the predicted shape to the 2D image by estimating camera rotation and position parameters [Ramakrishna 2012] and enforce plausibility constraints [Akhter 2015]. The best-performing approach of [Zhou 2016] uses a CNN to predict dense 2D joint location probabilities to which the 3D model is aligned. Although the matching does not influence the probabilities computed for each joint, the 2D detection precision is improved during the lifting step, indicating the potential of full 3D approaches.

3D Human Pose Estimation The main disadvantage of lifting-based approaches is that errors made in 2D detection cannot be recovered and propagate to 3D.

A bridge between lifting approaches and full 3D approaches is the work of [Tome 2017]. This work attempts to amend the problem of error propagation by iterating 2D and 3D joint position estimation. Indeed they report improved performance over the course of iterations.

In recent works [Mehta 2017, Pavlakos 2016, Li 2014], joint 3D approaches are

explored and the isolated 2D estimation is avoided. When the depth and 2D image positions are determined jointly, errors in the 2D prediction can be recovered in 3D. This is shown to boost the precision of predictions in the process. [Pavlakos 2016] report a per joint 3D error of 78.10mm in a 2D-3D decoupled variant, and a per joint 3D error of 69.77mm in their joint 3D prediction approach.

Challenges lie in the large number of parameters that have to be learned and in memory and efficiency considerations. The work of [Pavlakos 2016] proposes a neural network that predicts 3D joint positions from a single network. To this end, they discretize the 3D pose space into a regular grid and predict a dense likelihood volume for every joint to be present at a given grid node. To reduce computational overhead a coarse-to-fine prediction scheme is proposed: The z-dimension is at first discretized in a low resolution and the first part of the network is trained to predict this low-resolution target. Subsequent network parts build on top of these first predicted heat maps, as well as on the features predicted for this image. The first part of this coarse-to-fine approach, with a single depth layer, can be trained with 2D human pose annotated datasets.

In [Li 2014] an approach is proposed that avoids a 3D volumetric representation by regressing the 3D coordinate for each joint directly from the input image. This is very memory efficient, but is outperformed by the volumetric representation of [Pavlakos 2016]. This indicates that a 3D volumetric representation, can be more effective for training a neural network [Pavlakos 2016].

Similarities to our Work In our work we aim at full 3D human pose estimation by joint estimation of image and depth coordinates. Similarly to [Pavlakos 2016] we demonstrate in Section 4.5.5 that the joint optimization of 2D and 3D poses improves 2D localization considerably.

In this chapter we implement a final optimization stage into our CNN that combines local appearance with semi-global pairwise offsets. A similar idea has been proposed for 2D human pose estimation in [Yang 2016, Tompson 2014]. Both works approximate the MRF inference by a loopy belief propagation approach. [Yang 2016] rolls out three iterations of loopy belief propagation in the last three layers of the CNN, while [Tompson 2014] modifies the probability function for better integration with back-propagation. In a feed-forward architecture this limits the loopy belief propagation to a fixed number of iterations. Instead we propose a single inference layer that efficiently optimizes loopy MRF problems for DPMs. Considering memory consumption in 3D and computational performance, we choose a Branch-and-Bound approach and combine it with ADMM.

The prediction of high-resolution volumetric probability maps is computationally challenging. We follow the approach of [Girshick 2015a, Güler 2016] and combine both approaches [Pavlakos 2016] and [Li 2014]: we discretize the 3D volume to predict likelihoods densely for every joint and we regress joint refinements as offset from the coarse grid locations.

4.2 DPM and Inference in Higher Dimensions

In this section we discuss the extension of DTBB from 2 to 3 dimensions, in other words, we move from 2-dimensional inference problems in the image plane to inference of objects in a 3-dimensional space. With this, the articulated 3D structure of objects can be inferred in one step. This requires careful algorithmic design since the memory and computational complexity increases with every dimension of the solution space. In our approach all dimensions are inferred jointly, such that the depth estimate can correct and robustify the x and y coordinates of the object within the image and vice versa.

In opposition to the previous chapter we now consider a 3-dimensional unary term and image-dependent pairwise potentials. This affects how the depth, z, is treated. Firstly, in Chapter 3 the unary term is 2-dimensional, linked to the x and y dimension of the input image. Now we obtain 3-dimensional unary potentials directly from a neural network trained to predict 3-dimensional unaries, in x, y and z dimension. And secondly, in Chapter 3 the pairwise terms are independent of the image. Now we train a neural network to predict the parameters of the 3-dimensional pairwise term based on the input image.

We recall that the inferred depth is determined by the unary term and the pairwise term. The unary and pairwise potentials in turn are the output of the neural network and the neural network computes them using the original image as input (compare to Figure 4.1).

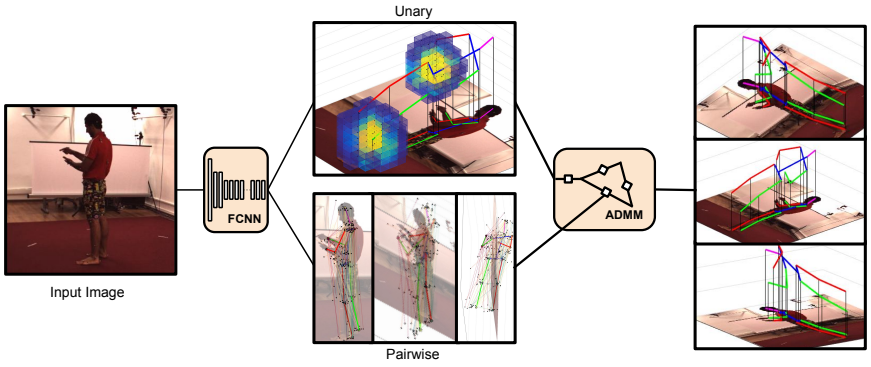


Figure 4.1: Overview over our approach for 3D human pose estimation. A CNN computes unary and pairwise potentials in 3D. The subsequent discrete optimization via ADMM and Branch-and-Bound finds the most coherent pose among all possible configurations.

4.2.1 Deformable Part Models in 3D

We now describe the DPM for 3-dimensional problems. Identically to the previous chapter (Chapter 3), an object is represented by N object parts. This results in an N-tuple of 3-dimensional coordinates $\Xi = (\xi_1, ..., \xi_N)$, $\xi_{1,...,N} \in \mathbb{R}^3$, to denote an

object. These object parts can move independently of each other in the volume Ω_{3D} spanned by the image dimensions and the depth. We denote this dimensionality with $|\Omega_{3D}| = w \times h \times d$, the width and height of the image and the number of discrete depth layers, respectively. The number of depth layers d needs to be set to adequately represent the object's dimensionality. This magnitude is mainly limited by the capacity of the neural network and the memory required during optimization.

The score function $S(\Xi)$ is defined alike Equation 2.2 for star-shaped graphs as

$$S(\Xi) = \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{j=2}^{N} \mathcal{P}_{1,j}(\xi_1, \xi_j). \tag{4.1}$$

with $\mathcal{U}_i$ being the unary potential for node i and $\mathcal{P}_{1,j}$ being the pairwise potential between the center node and part nodes $j \in \{2, ..., N\}$. The unary term $\mathcal{U}_i(\xi_i)$ is defined explicitly for every coordinate $\xi_i \in \Omega_{3D}$, the 3D volume.

4.2.2 Inference with Dual-Tree Branch-and-Bound in 3D

The inference with Branch-and-Bound of a 3-dimensional DPM is largely guided by the inference with DTBB proposed for a 2-dimensional DPM. The unary term $\mathcal{U}_i(\xi_i)$ is again precomputed using the Kd-Tree data structure. Instead of a tree over the 2-dimensional image plane, we now construct a Kd-Tree over the 3D unary volume for each part i. Accordingly, the tree traversal operates on 3-dimensional (instead of 2-dimensional) intervals until reaching a single voxel (instead of a pixel).

This extension to 3D allows inference of 3D DPMs at a very beneficial compromise between speed and resolution. Notably, we do not relax the score function or otherwise approximate the optimum, but we compute the global optimum. The global optimum of the score function 4.1 is determined by optimizing jointly the 2D position and depth of object parts.

Moving from a 2D plane to a 3D volume brings the question of scalability regarding memory consumption and computational complexity of the algorithm applied. The creation of the necessary data structures causes an overhead in memory consumption and runtime, linear in the number of voxels $\mathcal{O}(w*h*d)$, where w, h and d are the width, height of the image and the number of discretized depth layers, respectively. The runtime of the Branch-and-Bound inference is characterized analogously to the DTBB in 2D as $\mathcal{O}(N*log(w*h*d))$ best-case runtime. The worst-case runtime is quadratic in the number of voxels. We confirm experimentally that the runtime is indeed near-logarithmic in practice. The construction of the unary tree is linear in the number of voxels and quickly dominates runtime (Section 4.5.7). We note that the tree construction can be trivially parallelized to alleviate its overhead.

4.3 Training with Deep Supervision

We apply a deep network to predict the unary potentials as well as the pairwise potentials in 3D for the DPM in Equation 4.1. Computing the unary potentials in a

3D volume is challenging in terms of memory consumption and computation speed. For a good trade-off between accuracy and speed/memory consumption, we opt for a combination of a coarse volumetric classification and a continuous regression for high precision refinement. The pairwise term is determined by regression in 3D.

Architecture We use a ResNet [He 2016], a fully Convolutional Neural Network (CNN) of 151 layers, as the trunk of our system. We implement 2D and 3D branches as single convolutional layers to compute unary and pairwise terms.

Unary Term We use a combination of classification and regression is used to attack the image-based regression problem. In the classification stage we associate a confidence value with a set of non-overlapping depth intervals, corresponding to a coarse quantization of the depth value. If we have d classes and a depth range of, say D_r units, the k-th class is associated with a quantized depth of $q_k = k\frac{D_r}{d}$. This however may be at a very coarse depth resolution. We refine this coarse estimate by combining it with the results af a regression layer that aims at recovering the residual between the ground-truth depth values and their quantized depth estimates. As shown in Figure 4.2 this strategy allows us to "retarget" the voxels to 3D positions that lie closer to the actual part positions, without requiring the exhaustive sampling of the 3D space. In particular a voxel v lying at the k-th depth interval will become associated with a novel 3D position of part i,

$$p_i^v = v\frac{D_r}{d} + r_i(v), \tag{4.2}$$

where $r_j(v)$ is the residual regressed by our network for the i-th part type at voxel v.

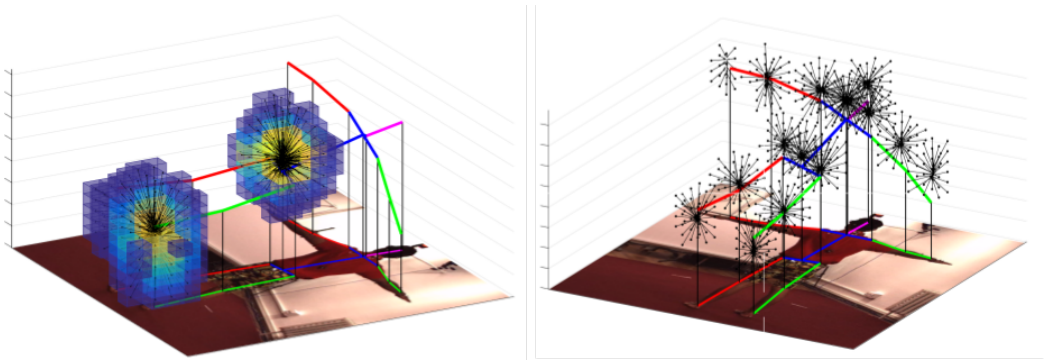


Figure 4.2: Unary 3D coordinates via quantized regression. Left: Sigmoid function on classified voxels and regressed residual vectors (in black) for two joints. Right: Regressed residual vectors for all joints. To efficiently regress the unary 3D coordinates, we use a combination of classification and regression. We first quantize the 3D space into voxels. We then estimate the score of each joint belonging to each of these voxels using a classifier. Finally we regress a residual vector per voxel, $r_i(v)$, which indicates the offset between the center of the voxel and the continuous 3D position of each joint (Equation 4.2).

- **Classification** We train a classifier for every joint j, using sigmoid cross entropy loss that assigns a classification score to each voxel if this voxel is the location of joint j. The ground truth for this classifier arises from the L2 distance of the voxel to the ground truth joint location. This classification score is the unary score $\mathcal{U}_j(p_j^v)$ in the subsequent discrete optimization stage.
- **Regression** To refine each voxels' position v, we regress 3D offsets $r_j(v)$ using the smooth L1 loss. This loss is active only for voxels who are close to the 3D ground truth annotation. With this, we compute the final position of the unary sample as per Equation 4.2.

Pairwise Term

- **Offset Regression** For the pairwise terms, we regress vectors that point from each 3D joint to others, $\mu_{i,j}$. Similar to the unary coordinates, we regress these quantities in a fully-convolutional manner. The smooth L1 loss for the pairwise offsets between a specific joint and the rest of the joints is only active on pixels within certain proximity to the specific joint.
- **2D Detection** In the second step we select one specific offset per pair of parts out of the set of offsets regressed in the previous paragraph. We train 2D joint classifiers and select for each joint j the offset for which the j-th classifier outputs the highest classification score. The loss for this classifier is computed as the L2 distance between the pixel and the ground truth joint location in 2D.

4.4 DPM Inference on Arbitrary Graph Topologies

In this chapter we will extend the Branch-and-Bound approach, presented in Section 4.2, to loopy graph topologies. As a result we observe higher precision in 3D human pose estimation. To tackle the more challenging optimization problem we use the Alternating Direction Method of Multipliers (ADMM) to approximately solve the inference problem.

The DPM treats an object as set of object parts, each with an individual visual appearance, and a relation to other parts. In [Felzenszwalb 2010b] the model has a star structure, connecting all object parts to one center part. This yields good performance for object detection, improving over single part models [Dalal 2005, Zhu 2006, Bosch 2007]. The underlying assumption is that the position of the object center limits the possible locations of the parts around it, in distance and direction. In turn, the object parts' positions help to determine the center's position, rendering the approach more robust. Given the a human's torso, constraints limit the range of positions of the elbow, like the distance to the torso and a certain angle. The position of the hand is constrained to the position of the elbow which is in turn constrained by the torso. This transitive relation over one intermediate actuator is respectively weaker than a direct relation and an estimate

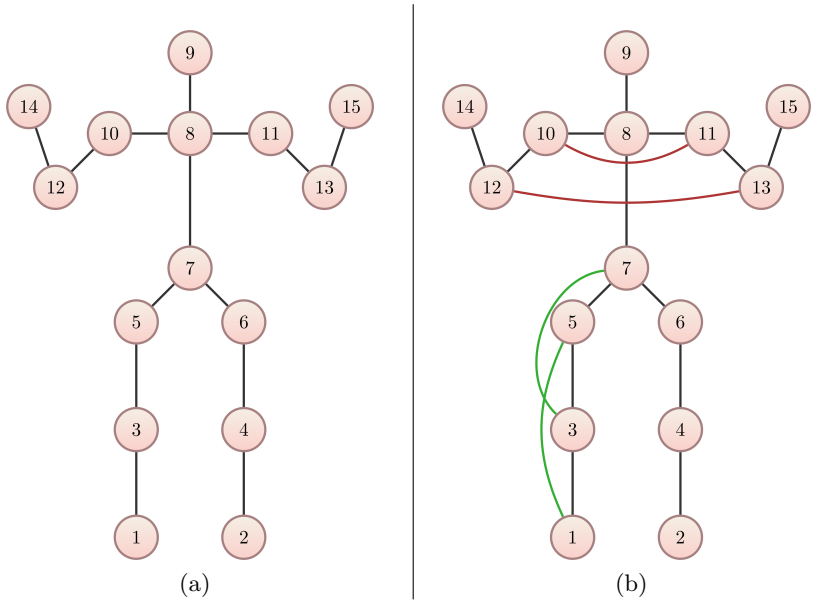


Figure 4.3: Example human body representations. In (a) 16 body joints as they are used in the Human MPII dataset, connected as a tree-shaped graph following the kinematic chain. In (b) we show the same 16 joints, now connected with additional edges, representing transitive (green) and symmetric (red) relations. This graph contains loops.

will be less precise. Because of this, many works use a tree-shaped model to represent the human body [Ferrari 2008, Felzenszwalb 2005, Eichner 2012, Yang 2011, Wang 2008, Ramanan 2007, Ronfard 2002]. We visualize an example tree structure in Figure 4.3a, also often called stick-figure. We note that the resulting optimization problem can be solved exactly using GDTs [Felzenszwalb 2004] and max-product [Pearl 2014], because the graph is loop-free.

Yet still transitive relations can contribute to stabilize human joint detection in the presence of occlusion, confusion with other humans in the image and other errors. Additionally, connections between symmetric body parts, for example right to left leg or right to left arm, are found to help amending the double-counting problem [Yang 2016, Sigal 2006, Wang 2008, Tian 2010]. Naturally a human's legs look alike and this is reflected within the model. Because of that, many human pose estimation systems suffer from either confusing the two legs or assigning both legs to the same location. Erroneously placing the left and the right leg on top of each other is referred to as double-counting. Therefore we follow [Yang 2016, Sigal 2006, Tian 2010] and advocate the use of additional edges in the graph. We exemplify transitive and symmetric edges, in green and red, respectively, by augmenting the stick figure model in Figure 4.3b. This richer structure seems to capture more faithfully the structure of humans and improves detection performance, as we show in Section 4.5.4. However, any edge added to a tree will create loops, rendering the problem hard. We propose an approximate optimization algorithm based on

the Branch-and-Bound algorithm presented in the Chapter 3. Our method covers tree-structured problems as well as loopy problems.

4.4.1 Graph Decomposition with ADMM

We briefly introduce the Alternating Direction Method of Multipliers (ADMM), a method we use to perform approximate optimization of a non-convex objective. ADMM breaks complex problems into subproblems and coordinates them via dual variables [Boyd 2011]. We apply this technique to decompose loopy graph topologies for DPMs into star-shaped subgraphs. The energy corresponding to these subgraphs can be globally optimized using Branch-and-Bound. With this we obtain an efficient optimization algorithm for DPMs not only on star-shaped graphs, but on arbitrary graph structures, including tree-shaped and loopy graphs. As objects are in general not star-shaped, different graph topologies often provide more faithful approximations of real-world objects. This has the potential of improving detection precision, as we show in Section 4.5.4.

4.4.1.1 Graph Splitting

The optimization problem, as described in Equation 4.1, is given as

$$\arg\max_{\Xi} S(\Xi) = \arg\max_{\xi_1,\ldots,\xi_N} \sum_{i=1}^{N} \mathcal{U}_i(\xi_i) + \sum_{(i,j)\in E} \mathcal{P}_{(i,j)}(\xi_i, \xi_j), \tag{4.3}$$

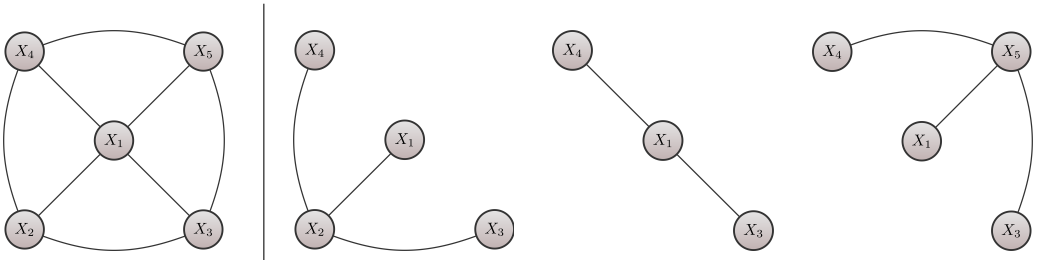
with graph $G = (V, E)$ and $E = \{(i, 1) |\, i{\in}\{2, ..., N\}\}$. We extend the set of edges E to an arbitrary set of edges $E_\circ = \{(i, j) |\, i, j{\in}\{1, ..., N\}, i \neq j\}$ and $G \coloneqq (V, E_\circ)$. This new problem is more challenging because inference on general loopy MRFs with multiple labels is NP-hard [Batra 2010, Shimony 1994]. Approximations can be determined using loopy belief propagation [Murphy 1999], but convergence is not guaranteed. We resort to a different approximate optimization method, based on the "divide and conquer" principle: we split graph G into star-shaped subgraphs using ADMM. This is similar in the idea of [Boussaid 2014] where a loopy graph is split into loop-free chains in the context of landmark detection.

These star-shaped subgraphs can be optimized efficiently with Branch-and-Bound (Section 4.2). Solutions obtained for the subgraphs are brought to an agreement with each other by introducing consensus variables. These consensus variables are updated iteratively and communicated through dual variables.

In general, there are various ways to split the graph, but we enforce partitions following three requirements:

- The union of vertices in subgraphs constitute the full set of vertices, $\bigcup_{s=1}^{K} V_s = V$.
- The union of edges in subgraphs constitute the full set of edges, $\bigcup_{s=1}^{K} E_s = E_\circ$.
- Every subgraph has at least 2 vertices, $|V_s| > 1$, $\forall s$.

For convenience we chose a partition where edges are disjunct between subgraphs, $E_{s_1} \cap E_{s_2} = \emptyset$, $\forall s_1 \neq s_2$. We visualize a decomposition of an example graph in Figure 4.4. From an original graph of 5 nodes with loops, 3 partitions are created, obeying the above conditions. All nodes are covered at least by one subgraph and all edges occur exactly once. We note that a subgraph consisting of a single node is never necessary to partition the graph, but would increase the number of subgraphs, therefore slowing down convergence. Since inference on star-shaped graphs can be done exactly and efficiently, we seek to maximize the size of subgraphs and reduce the number of part problems.



(a) A loopy graph G has to be decomposed into loop-free subgraphs.

(b) Star-shaped subgraph G_1

(c) Star-shaped subgraph G_2

(d) Star-shaped subgraph G_3

Figure 4.4: Example graph decomposition. It is obvious that there are many alternative graph decompositions.

4.4.1.2 The Alternating Direction Method of Multipliers Algorithm

ADMM decomposes loopy graphs for MRF optimization problems into tree-structured subproblems who are coordinated by a master problem. In ADMM, an optimization problem with constraints is relaxed to its Lagrangian form and then augmented with a quadratic term for faster convergence.

In this section we provide a general formulation of ADMM and the next chapters adapt the general formulation to our specific MRF problem.

We closely follow [Boyd 2011] and assume the following optimization problem:

$$\min_{x,z} f(x) + g(z), \tag{4.4}$$

for $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ variables of the functions f and g respectively. Additionally the constraints

$$Ax + Bz = c \tag{4.5}$$

are given, with $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$ and $c \in \mathbb{R}^p$. This optimization problem can be relaxed by its Lagrangian L with the dual variable $y \in \mathbb{R}^p$

$$L(x, z, y) f(x) + g(z) + y^T (Ax + Bz - c), \tag{4.6}$$

and dropping the constraints of 4.5.

In ADMM this term is augmented by a quadratic term, yielding the augmented Lagrangian

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}||Ax + Bz - c||_2^2. \tag{4.7}$$

This formulation does not change the optimum of the Lagrangian in Equation 4.6 because the parabola's peak coincides with the constraints of Equation 4.5, but delivers faster convergence rates [Boyd 2011].

In summary, the ADMM algorithm iteratively solves the following three steps:

$$x^{k+1} := \arg\min_x L_\rho(x, z^k, y^k) \tag{4.8}$$

$$z^{k+1} := \arg\min_z L_\rho(x^{k+1}, z, y^k) \tag{4.9}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c), \tag{4.10}$$

with iteration counter k and step width $\rho > 0$. The hyper-parameter ρ influences the step width of the updates on the dual variable y, as well as the steepness of the parabola augmenting the Lagrangian. This affects convergence speed and the optimum reached. For small ρ typically more iterations are needed to reach convergence than for large ρ. Setting ρ to a large value increases the probability that the algorithm gets stuck in a local optimum or oscillates around a solution.

4.4.1.3 Human Pose Estimation with ADMM

We now apply ADMM to inference on arbitrary graph topologies in the context of DPMs. To this end, we decouple the graph into star-shaped subgraphs. This allows us to optimize the subproblems with Branch-and-Bound.

We reformulate problem 4.3 as follows:

$$\arg\max_\Xi S(\Xi) = \arg\max_{\Xi_1,\dots,\Xi_K} \sum_{s=1}^{K} S_s(\Xi_s), \tag{4.11}$$

$$\text{s.t. } \Xi_s(r) = u(r), \qquad \forall r \in R, \tag{4.12}$$

splitting graph G such that subgraphs G_s are star-shaped with nodes V_s and edge set E_s, as described in Section 4.4.1.1. The variables Ξ_s are copies of $\Xi = (\xi_1, ..., \xi_N)$, with $\Xi_s(i)$ the i-th part position. Nodes $r \in R$ are the nodes shared among graphs G_s.

To avoid double-counting of shared nodes, we define

$$S_s(\Xi_s) = \sum_{i \in V_s} \frac{1}{\gamma_i} \mathcal{U}_i(\xi_i) + \sum_{(i,j) \in E_s} \mathcal{P}_{(i,j)}(\xi_i, \xi_j), \tag{4.13}$$

with $\gamma_i = \sum_{s=1}^{K} \mathcal{I}(i \in V_s)$ being the multiplicity of occurrences in different subgraphs G_s, and $\mathcal{I}(.)$ the indicator function.

The constraints to Equation 4.12 coordinate the solutions on the subgraphs G_s: All corresponding variables $\Xi_s(r)$, with $s = \{\sigma | r \in V_\sigma\}$ the subgraphs that contain the shared node r, are forced to agree on one common value $u(r)$, ensuring consistency. We call $u(r)$ consensus variables. These constraints are now relaxed into the Lagrangian form

$$\arg\max_{\Xi} \sum_{s=1}^{K} S_s(\Xi_s) - \sum_{s=1}^{K} \sum_{r \in R \cap V_s} y_s(r)(\Xi_s(r) - u(r)), \tag{4.14}$$

introducing the Lagrangian multipliers $y_s(r)$ for each constraint in Equation 4.3. The augmented Lagrangian introduced in Equation 4.7 takes the following expression for our case:

$$\begin{aligned} L_\rho(\Xi, u, y) = \underbrace{\sum_{s=1}^{K} S_s(\Xi_s)}_{f(x)} + \underbrace{0}_{g(z)} - \underbrace{\sum_{s=1}^{K} \sum_{r \in R \cap V_s} y_s(r)(\Xi_s(r) - u(r))}_{y^T(Ax+Bz-c)} \\ - \frac{\rho}{2} \underbrace{\sum_{s=1}^{K} \sum_{r \in R \cap V_s} ||\Xi_s(r) - u(r))||_2^2}_{||Ax+Bz-c||_2^2}, \end{aligned} \tag{4.15}$$

adding a quadratic penalty for deviations from the agreement $u(r)$, weighted by a parameter $\rho > 0$. We annotate the corresponding terms in ADMM (Equation 4.7) in curly brackets below Equation 4.15. The consensus variables $u(r)$ correspond to the vector z. Function $g(z) = 0$ is a constant function.

Following ADMM, an iterative optimization scheme is devised from this formulation, involving **(i)** solving the subproblems, **(ii)** computing the consensus solution among them and **(iii)** updating the dual variables in the subproblems. Hence, the problem 4.3 is solved approximately by iteratively performing the following three steps:

$$\textbf{(i)} \quad \Xi^{t+1} = \arg\max_{\Xi} L_\rho(\Xi, u^t, y^t) \tag{4.16}$$

$$\textbf{(ii)} \quad u^{t+1} = \arg\max_{u} L_\rho(\Xi^{t+1}, u, y^t) \tag{4.17}$$

$$\textbf{(iii)} \quad y^{t+1}(r) = y^t + \rho(\Xi^{t+1}(r) - u^{t+1}(r)) \tag{4.18}$$

The first **step (i)** requires the inference of the subproblems in star-shaped subgraphs, created by the reformulation in Equation 4.12. To align individual solutions the function L_ρ comprises the DPM energy S_s and the sum of a linear and a quadratic term, drawing individual solutions to a common solution. We describe the efficient computation of this problem below.

The second **step (ii)** computes the consensus solutions $u(r)$. The solution can

be found in closed form by setting the partial derivative for $u(r)$ to 0. We yield

$$\forall r \in R: \quad \frac{\partial L_\rho(\Xi_s^{t+1}, u, y^t)}{\partial u(r)} = \underbrace{-\sum_{s=1}^{K} y_s^t(r)}_{=0} - \rho \sum_{s=1}^{K} (\Xi_s^{t+1}(r) - u(r)) \stackrel{!}{=} 0 \tag{4.19}$$

$$\Leftrightarrow \quad u(r) = \frac{1}{K} \sum_{s=1}^{K} \Xi_s^{t+1}(r), \tag{4.20}$$

using in Equation 4.19 the fact that the dual of L is invariant to global shifts in y. The consensus value $u(r)$ turns out to be indeed the average over split nodes $\Xi_s(r)$.

Step (iii) updates the dual variables y_s using step width ρ, driving the partial solutions closer together in the next iteration. For our experiments we set $\rho = 1$ constant. Variations with variable ρ are discussed, for example, in [Rockafellar 1976, He 2000]. We observe occasionally small oscillations. Convergence is reached when all shared nodes agree on one value, $\Xi_s(r) = u(r), \forall s, \forall r \in R \cap V_s$.

4.4.2 Inference of ADMM-Augmented Subproblem

In the previous chapter we have outlined the iterative ADMM algorithm to solve DPMs with arbitrary edge sets approximately. We will now detail the efficient calculation of step (i) in one iteration of the ADMM algorithm (4.16). We show that it decouples over the chosen graph partitions $G_s = (V_s, E_s)$ and the resulting subproblems can be efficiently optimized by adapting our Branch-and-Bound approach.

The first of these three steps consists of maximizing $L_\rho(\Xi, u^t, y^t)$. For better readability we will drop iteration counter t. This term decouples over partitions as

$$L_\rho(\Xi, u, y) \tag{4.21}$$

$$= \sum_{s=1}^{K} S_s(\Xi_s) - \sum_{s=1}^{K} \sum_{r \in R \cap V_s} y_s(r)(\Xi_s(r) - u(r)) - \frac{\rho}{2} \sum_{s=1}^{K} \sum_{r \in R \cap V_s} ||\Xi_s(r) - u(r))||_2^2 \tag{4.22}$$

$$= \sum_{s=1}^{K} \Big(S_s(\Xi_s) - \sum_{r \in R \cap V_s} y_s(r)(\Xi_s(r) - u(r)) - \sum_{r \in R \cap V_s} \frac{\rho}{2} ||\Xi_s(r) - u(r))||_2^2 \Big), \tag{4.23}$$

a sum of DPM-like energies. In contrary to the traditional DPM energy, two summands are added to the score function, a linear and a quadratic term in the variables Ξ_s. We write explicitly the score function for a specific subproblem s as

$$S'_s(\Xi_s) = \sum_{i \in V_s} \mathcal{U}_i(\xi_i) + \sum_{(i,j) \in E_s} \mathcal{P}_{(i,j)}(\xi_i, \xi_j) + \sum_{r \in R \cap V_s} \mathcal{A}_r(\Xi_s(r)|y, u), \tag{4.24}$$

with the ADMM-induced term

$$\mathcal{A}_r(\Xi_s(r)|y,u) = -y_s(r)(\Xi_s(r) - u(r)) - (\rho/2)||\Xi_s(r) - u(r)||_2^2. \tag{4.25}$$

We note that the additional term $\mathcal{A}_r(\Xi_s(r)|y,u)$ behaves much like the unary potential, depending on only one node at a time. So we treat the linear and the quadratic term as an additional, parameterized unary term. In Branch-and-Bound we use the fact that $\max_{\xi_j} \mathcal{U}(\xi_j) + \max_{\xi_j} \mathcal{P}_{i,j}(\xi_i,\xi_j) \geq \max_{\xi_j} \left(\mathcal{U}(\xi_j) + \mathcal{P}_{i,j}(\xi_i,\xi_j)\right)$ to compute an upper bound that independently calculates upper bounds for unary and pairwise potentials. We extend this inequality to compute an upper bound including a third term originating from the ADMM formulation. Without loss of generality we assume node 1 to be the center node and we rename nodes V_s to be $\{1,...,N_s\}$ and compute the upper bound $\bar{S}'_s$ as

$$\bar{S}'_s(\nu_1,...,\nu_{N_s}) = \sum_{i\in V_s} \max_{\xi_i\in\nu_i} \mathcal{U}_i(\xi_i) + \sum_{j\in\{2,...,N_s\}} \max_{\substack{\xi_1\in\nu_1\\ \xi_j\in\nu_j}} \mathcal{P}_{1,j}(\xi_1,\xi_j) + \sum_{r\in R\cap V_s} \max_{\xi_r\in\nu_r} \mathcal{A}_r(\xi_r|y,u) \tag{4.26}$$

$$\geq \max_{\substack{\xi_1\in\nu_1\\ ...\\ \xi_{N_s}\in\nu_{N_s}}} \Big(\sum_{i\in V_s} \mathcal{U}_i(\xi_i) + \sum_{j\in\{2,...,N_s\}} \mathcal{P}_{1,j}(\xi_1,\xi_j) + \sum_{r\in R\cap V_s} \mathcal{A}_r(\xi_r|y,u)\Big) \tag{4.27}$$

$$= \max_{\substack{\xi_1\in\nu_1\\ ...\\ \xi_{N_s}\in\nu_{N_s}}} S'_s(\xi_{i\in V_s}) \tag{4.28}$$

with ν_i corresponding to an interval in the label space Ω_{3D} for variable ξ_i.

To compute $\max_{\xi_r\in\nu_r} \mathcal{A}_r(\xi_r|y,u) = -y_s(r)(\Xi_s(r) - u(r)) - (\rho/2)||\Xi_s(r) - u(r)||_2^2$ for one shared node r, we note that the term decouples over dimensions. Therefore the upper bound is determined as the sum over contributions, computed independently for a 1-dimensional interval in x, y and z dimension.

$$\max_{\xi_r\in\nu_r} \mathcal{A}_r(\xi_r|y,u) = -\sum_{d\in\{x,y,z\}} y_s^d(r)(\Xi_s^d(r) - u^d(r)) + (\rho/2)(\Xi_s^d(r) - u^d(r))^2. \tag{4.29}$$

The quadratic's peak is computed as $p^d = y_s^d(r)/\rho + u^d(r)$. Then

$$\max_{\xi_r\in\nu_r} \mathcal{A}_r^d(\xi_r|y,u) = \begin{cases} \mathcal{A}_r^d(p^d|y,u) & \text{if } p^d \in \nu_r^d \\ \max(\mathcal{A}_r^d(\xi^d_{\nu_{low}}|y,u), \mathcal{A}_r^d(\xi^d_{\nu_{high}}|y,u)) & \text{else} \end{cases} \tag{4.30}$$

where interval $\nu_r^d \widehat{=} [\xi^d_{\nu_{low}}; \xi^d_{\nu_{high}}]$.

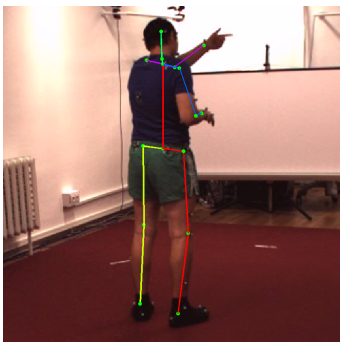
4.4.3 Inference Visualization

We illustrate the progression of the ADMM-guided inference to give the reader an intuitive understanding of how differing partial solutions work together. The task

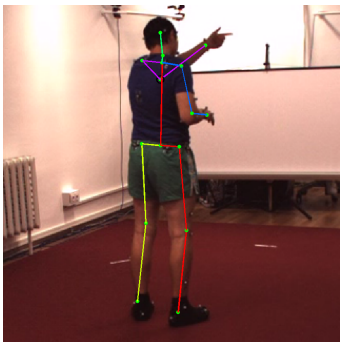
is to determine the pose of the human in a test image Figure 4.5a. The result of our algorithm is shown in Figure 4.5c. For comparison, we provide the ground truth skeleton in Figure 4.5b.

The chosen instance is challenging due to the occlusion of the left elbow. This results in a unary term for the left elbow that is spread out over a wide range of positions (Figure 4.5d). If we were to take only the unary potential into account, we would obtain the pose displayed in Figure 4.5e, where the left elbow (black circle) deviates clearly from its correct position. The solution determined by our approach corrects this mistake to a certain degree (Figure 4.5f). The detailed process is elaborated next.

(a) Test image



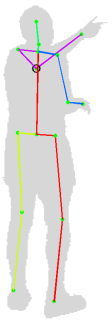
(b) Ground truth solution



(c) Solution with ADMM

(d) Unary potential for left (occluded) elbow

(e) Solution computed as maximum over unary potentials. The position of the left elbow (black circle) is far off its correct position.



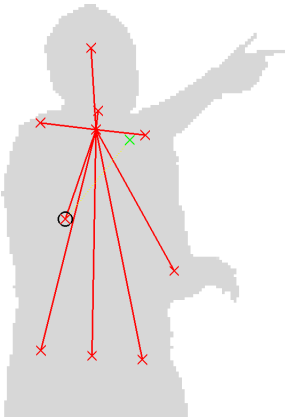
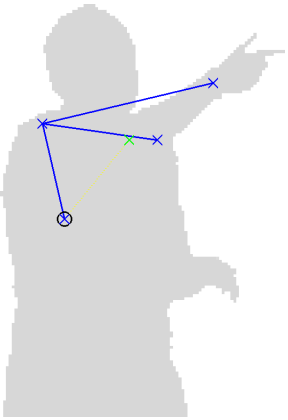
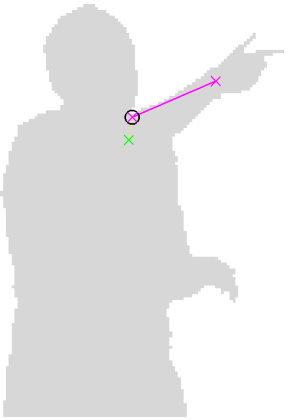
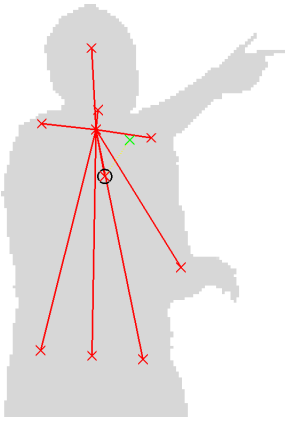
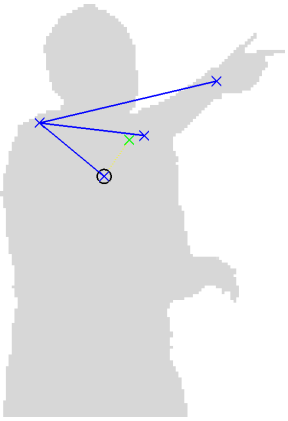
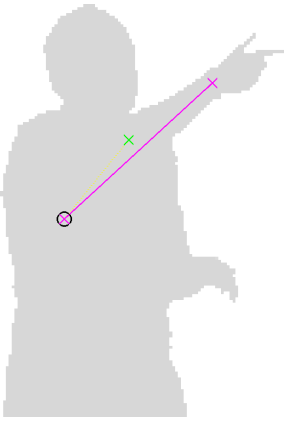
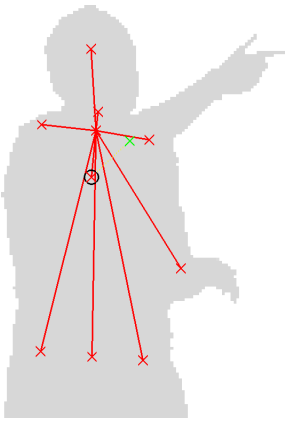
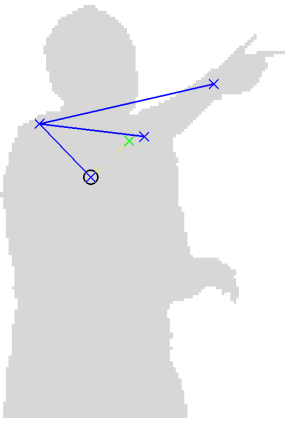
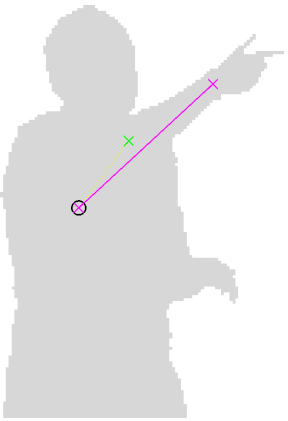
(f) Solution with ADMM, corresponding to (c). The position of the left elbow (black circle) has improved over the "unary-only" solution of (e).

We show the change of partial solutions over the first 6 iterations. To reduce complexity, we focus on one single node of the example image, the "left elbow". This node is part of three subgraphs, visualized as graphs in red, blue and magenta, shown in individual columns in Table 4.1 and 4.2). Each row corresponds to one iteration. Convergence is reached in iteration 6. A partial solution consists of the coordinates of all contained nodes, annotated with small crosses. Additionally we show the graph structure corresponding to the respective subproblem. The left elbow node is again marked with a black circle, as in all illustrations in this chapter.

Its ground truth position is marked as a green cross.

Table 4.2 allows deeper insight into the score function that drives each subproblem. For each iteration we depict the unary term plus ADMM-term, $\mathcal{U}_i + \mathcal{A}_i$, as derived in Equation 4.24, specific to each subproblem. We illustrate every grid point in its refined location by a half-transparent 3D sphere. Blue stands for low, red for intermediate and yellow for high score values. The color coding does not reflect absolute magnitudes as the quadratic term, $\mathcal{A}_i$, changes the value range considerably. Instead we indicate relative magnitudes, showing the 85% lowest scores in a blue color and the highest 1.5% scores in a yellow color. The other values in between are depicted in red.

The maximum of the unary for the left elbow is found roughly at the position of the left kidney (yellow color in Table 4.2, first row). It is obvious that the CNN misses the position of the left wrist as clue to where the left elbow has to be. The pairwise term that connects wrist and elbow is contained in the third subproblem. Thus the partial solution of the third subproblem successfully recovers the position of the left elbow. The other two subproblems do not contain the edge from wrist to elbow. Thus the partial solutions in the first iteration diverge greatly between the first two and the third slave (Table 4.1, first row). In the next iteration, the ADMM term draws the partial solutions of the first two slaves towards the solution of the third slave and vice versa. The subproblems converge on a common solution in iteration 6.

Slave 1	Slave 2	Slave 3	Iteration
			1
			2
			3

continued on the next page

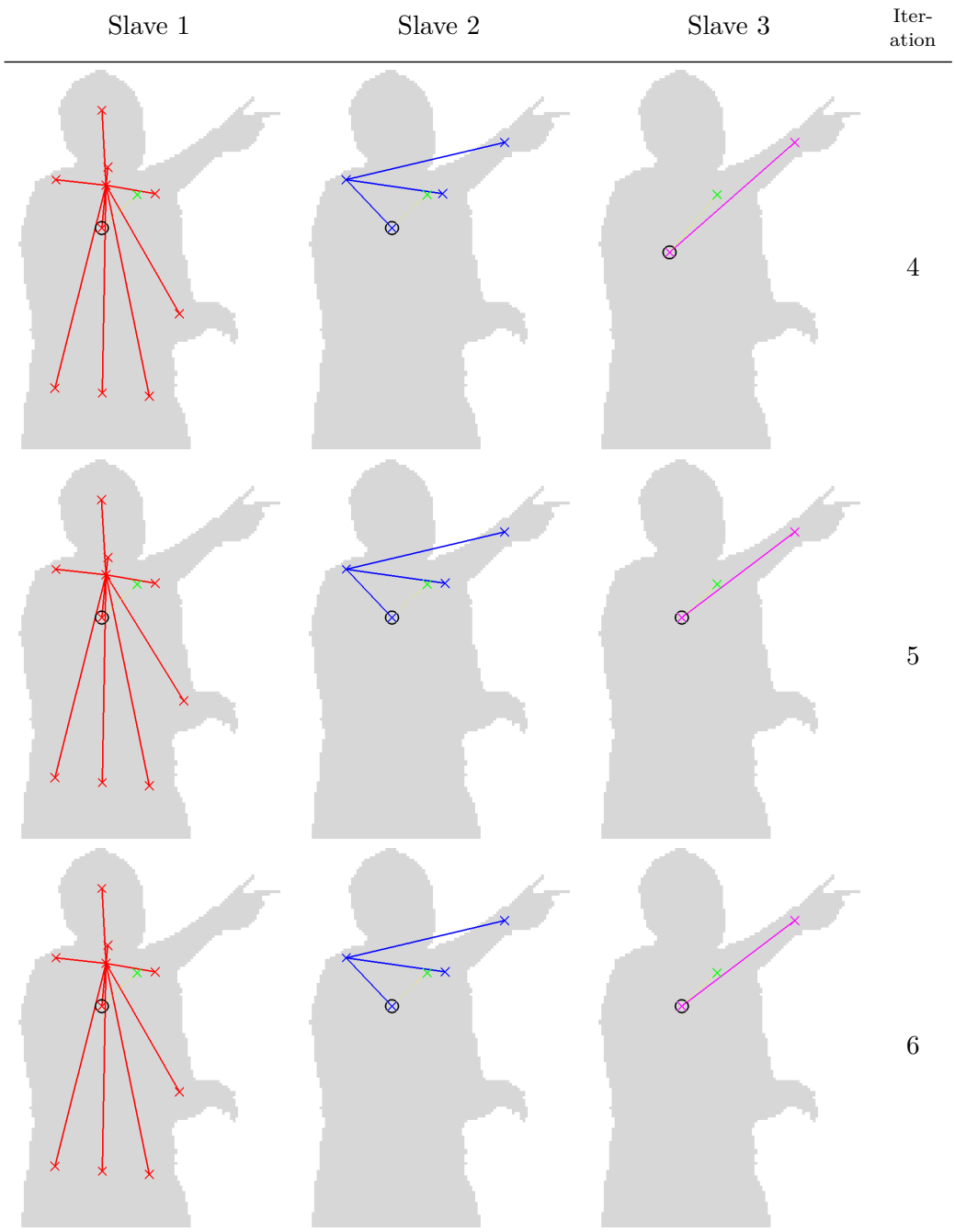
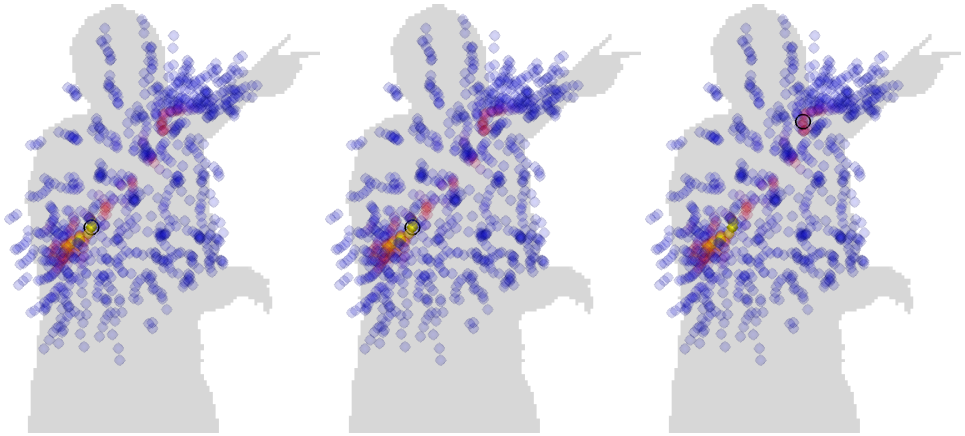
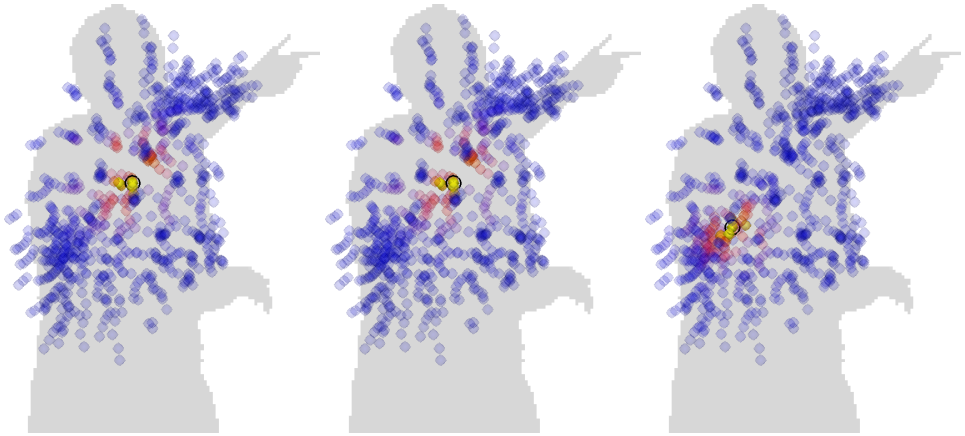
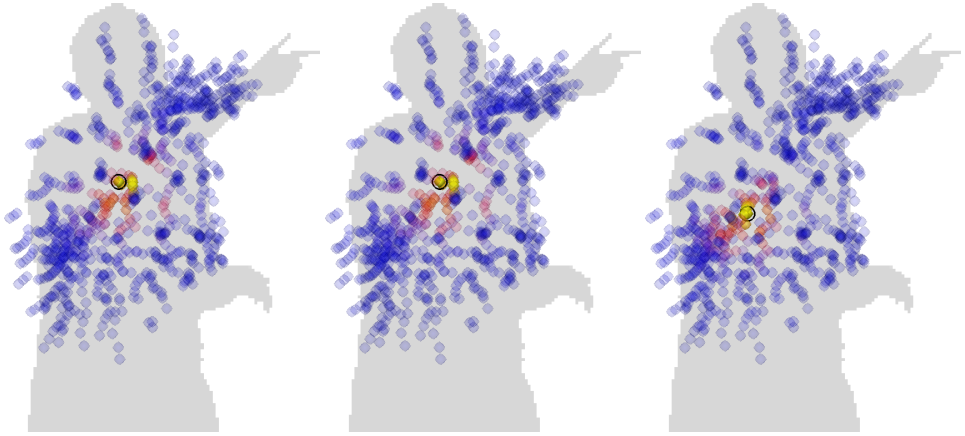


Table 4.1: Visualization of the per-slave solution over 6 ADMM iterations. We display only slaves that contain the left elbow. Ground truth joint locations are marked with green crosses.

Slave 1	Slave 2	Slave 3	Iteration
			1
			2
			3

continued on the next page

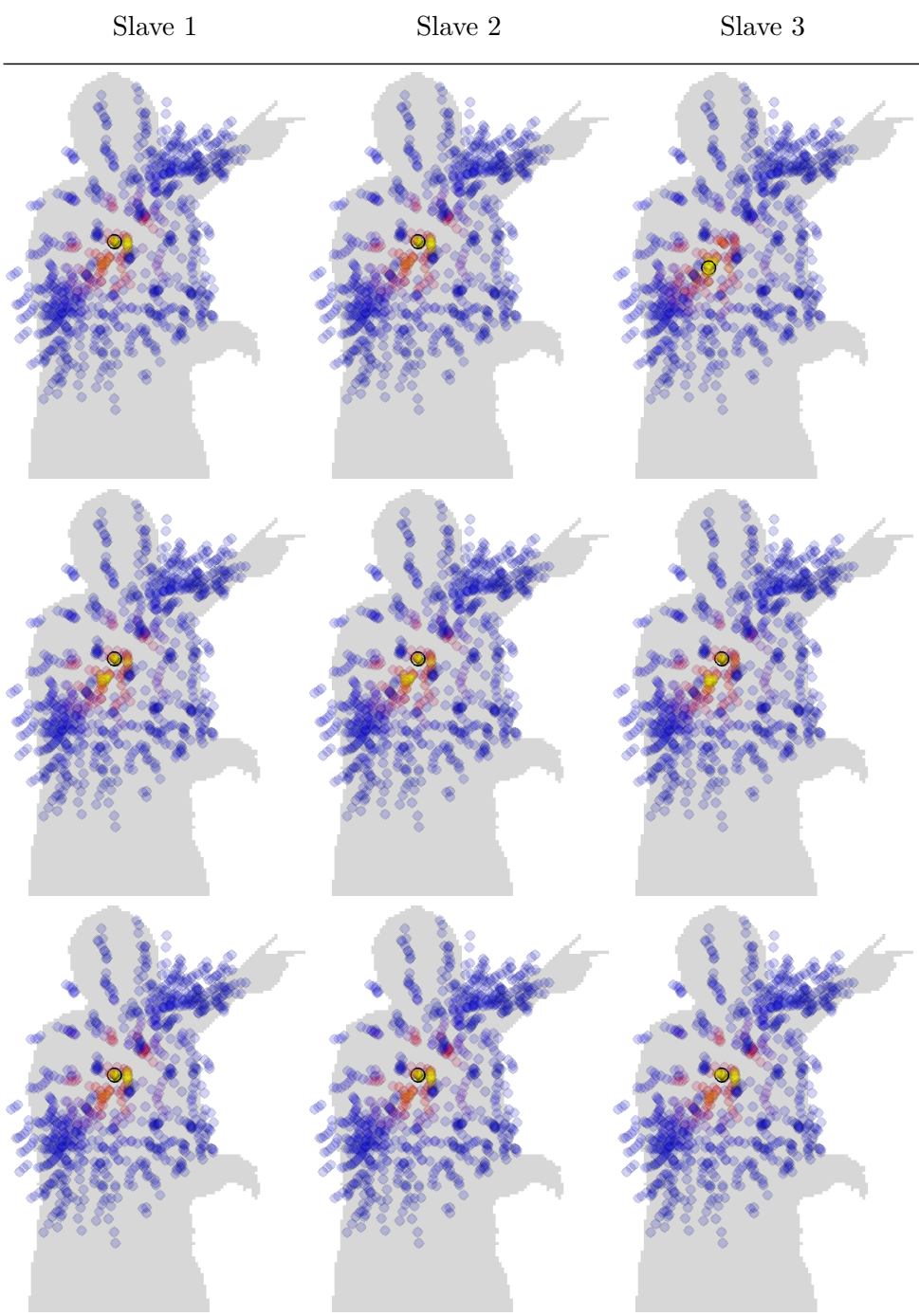


Table 4.2: Visualization of the extended unary potential, $\mathcal{U}_i + \mathcal{A}_i$, over 6 ADMM iterations, where node i corresponds to the left elbow. The partial solution found in the respective iteration is marked with a black circle. In the 0-th iteration the ADMM-term is zero, so the visualization is identical for all three slaves. The colors indicate relative magnitude, with the lower 85% in blue, intermediate values in red and the highest 1.5% in yellow. Best viewed in color.

4.5 Results on 3D Human Pose Estimation

We demonstrate the performance of the described techniques for task of 3D human pose estimation, showing that loopy graph structures deliver state of the art results on the Human3.6M dataset. The performance of 4 different graph structures is analyzed, showing improved performance of loopy graph structures.

4.5.1 Evaluation Setup

In 2014 the largest dataset for 3D human pose estimation, the Human3.6M dataset [Ionescu 2014] has been published. The dataset comprises of 3.6 million video frames, recorded by four cameras in a controlled environment. For each pose, 17 3D body joints are annotated using a motion capture system. In these video sequences, 11 professional actors perform various daily actions like "sitting on a chair", "eating" and "walking". The video sequences of actors S1 to S8 and S10 are used for training, sequences S9 and S11 remain for testing. This follows the practice in recent literature [Pavlakos 2016, Li 2015, Zhou 2016]. We have sub-sampled the video sequences from 50fps to 10fps to remove very similar frames and reduce data size. We also notice that several annotations suffer from drift in the ground truth. These samples have been removed from the testing set. The remaining testing set consists of about 100k frames.

To evaluate the performance of our system, we use the mean euclidean distance over joints of the prediction to the ground truth annotation. As we can not deduce the absolute scale and depth of the body, we first apply the Procrustes transformation. This transformation is estimated as to best align two shapes by transforming one by scaling, rotating and translating. The resulting "reconstruction error" is measured in millimeters.

4.5.2 Implementation Details

Our network is initialized with the ResNet weights trained on the Human MPII dataset [Andriluka 2014] for 2D human pose estimation. We train our network on the Human3.6M dataset [Ionescu 2014] (detailed description of the dataset in Section 4.5.1). We scale the images to the size 320×320 pixels. The network's downscale factor is 16, yielding a 20×20×1024 3D feature map as input to the described 2D and 3D branches. We perform data augmentation by cropping and rotating input images. This increases the network's ability to generalize and robustness towards occlusion and rotations.

We determine a fixed precision term $C_{i,j} = 0.001$ for all joints by cross-validation.

We observe that inserting MPII training data samples while training with Human3.6M improves performance. The MPII is based on a slightly different set of joints and is annotated in 2D. Following [Sun 2017], we adapt the human body labelling of the Human3.6M dataset to the MPII dataset. To this end we add a joint

"thorax" between the shoulders and remove the joints "neck" and "chin". This results in 16 joint annotations identical to the MPII annotation. When training with a Human MPII sample, we deactivate all losses for the unary term generation and leave active only the losses for 2D classification and pairwise offset regression. Additionally we modify the loss for the pairwise offset regression such that the predicted depth component is ignored.

4.5.3 Quantitative Comparison

We compare the performance of our approach with state of the art works [Yasin 2016, Rogez 2016, Tome 2017, Pavlakos 2016]. The results are summarized in Table 4.3. We first evaluate the performance of our neural network *before* the subsequent optimization with ADMM. To this end, we use the highest scoring configuration predicted by the unary term alone, denoted as "Unary". This method (53.48mm) performs close to the state of the art ([Pavlakos 2016], 53.2mm). This serves as a baseline in the following ablation study to analyze various graph topologies. Then, under "ADMM", we calculate the reconstruction error of the estimated pose *after* the optimization with Branch-and-Bound and ADMM. Our approach that unites a deep network with a final discrete optimization scheme, clearly outperforms prior works (50.87mm).

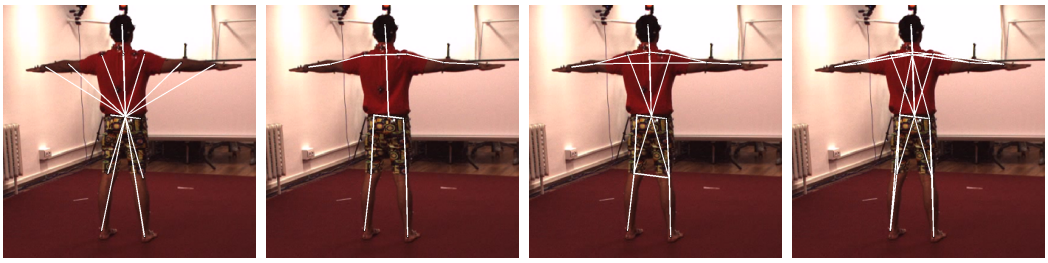
	Average error
Yasin et al. [Yasin 2016]	108.3
Rogez et al. [Rogez 2016]	88.1
Tome et al. [Tome 2017]	70.7
Pavlakos et al. [Pavlakos 2016] [1]	53.2
(Ours)Unary	53.48
(Ours)ADMM	50.87

Table 4.3: A comparison of our approach with recent literature based on the reconstruction error.

4.5.4 Ablation Study with Graph Topologies

To analyze the effects of varying graph layouts, we performed an ablation study for 3D human pose estimation on the well known Human3.6M dataset [Ionescu 2014]. We experiment with a variety of graph structures and monitor its impact on the performance. We have picked a number of configurations by hand, as testing exhaustively all possible graph configurations is prohibitive (for 16 nodes there are already $\sim 7.2 * 10^{16}$ spanning trees). Furthermore the interpretability is higher for certain models than for a random selection.

The following graph configurations have been evaluated:



(a) center star (b) stick figure (c) extended stick figure (d) 2-hop

Figure 4.5: Selected graph configurations.

- **center star** (Figure 4.5a) describes the graph topology where all joints are connected to one central root node at the human's torso.

- **stick figure** (Figure 4.5b) is a graph that directly corresponds to the human skeleton, for instance the wrist is connected to the elbow, the elbow is connected to the shoulder, and so on.

- **extended stick figure** (Figure 4.5c) is an extension to "stick figure", containing all its edges plus additional connections between the elbows of left and right arm, left and right knee, head to shoulders and torso to knees.

- **2-hop** (Figure 4.5d) follows the human skeleton like "stick figure" and adds connections from every joint to its indirect (2-)neighbours in the skeleton. This connects, for example, hand with shoulder and ankle to hips and left to right knee.

In Table 4.4 we report the average reconstruction error for each of the 15 human action categories present in the Human3.6M benchmark, as well as the average reconstruction error over all categories in the last column. In the first line we state the results obtained using the unary potentials only as baseline. In the following lines we analyze the influence of different graph structures on the performance, measured as average reconstruction error. The "center star" configuration performs better than "unary only", indicating that the body center can infer some meaningful information about the other body parts. Better performs the "stick figure", the most prominent representative in human pose estimation. Clearly the shoulder knows better where the elbow has to be than the center node, thus this structure performs better than "center star". Its extension, the "extended stick figure", slightly improves over the normal stick figure. The "2-hop" configuration performs best in this line. Its additional connections roughly follow the body's structure and seem to help to resolve occlusions and improves accuracy.

	Directions	Discussion	Eating	Greeting	Phoning	Photo
UNARY alone	49.69	49.45	47.77	50.69	54.80	57.35
center star	49.41	49.26	47.35	49.93	50.97	56.12
stick figure	49.13	49.19	47.15	49.70	50.50	55.57
extended stick figure	49.16	49.07	47.35	49.82	50.67	55.45
2-hop	**48.89**	**48.75**	**47.07**	**49.40**	**49.82**	**55.31**
	Posing	Purchases	Sitting	Sit. Down	Smoking	Waiting
UNARY alone	43.76	44.11	65.39	95.76	53.53	46.27
center star	43.62	**43.43**	61.50	**78.09**	52.51	45.88
stick figure	43.53	43.59	60.14	79.46	51.52	45.74
extended stick figure	43.60	43.57	**59.94**	78.51	**51.42**	46.01
2-hop	**43.30**	43.47	60.48	78.20	51.69	**45.63**
	Walk Dog	Walking	Walk Tog.	Average		
UNARY alone	51.53	41.59	49.52	53.48		
center star	50.63	41.08	49.41	51.42		
stick figure	50.59	40.73	49.33	51.12		
extended stick figure	50.39	40.89	49.32	51.08		
2-hop	**50.16**	**40.74**	**49.17**	**50.87**		

Table 4.4: Comparison of average reconstruction errors for different graph topologies.

We complete this ablation study with some visual examples from the Human3.6M dataset (Table 4.5). We have selected image instances where the variation is well visible and show the ground truth body pose and two of the graph configurations presented above in Figure 4.5 or the "unary alone" solution. Table 4.5 is subdivided into five blocks:

I This block illustrates two examples where the unary term alone fails to localize a foot. The pairwise terms correct this mistake and stabilize locally noisy detections.

II The "center star" configuration does not connect the feet to the knees or the head to the neck, resulting in partially bizarre human poses. The "stick figure" configuration connects neighboring body joints and improves these cases.

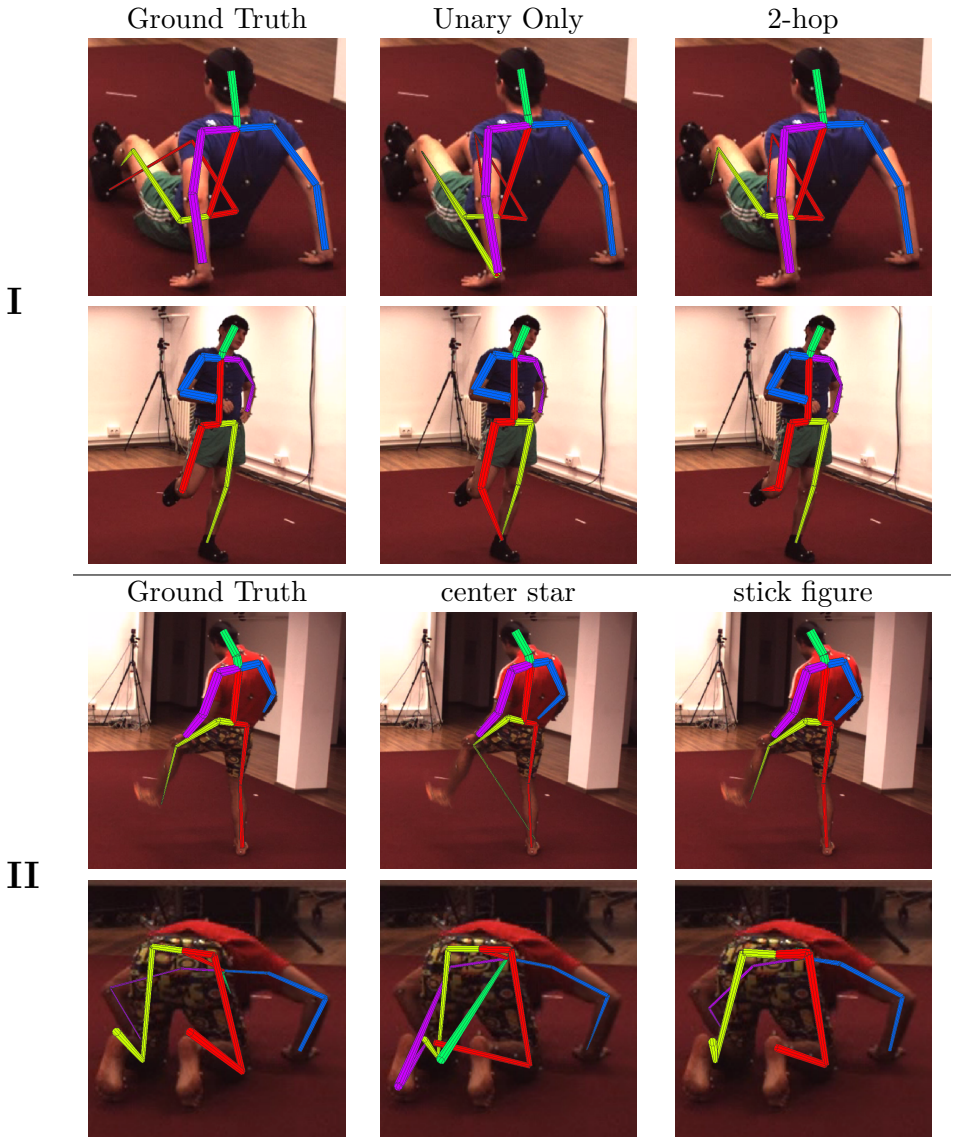
III In both examples the pose detected with the "stick figure" configuration is wrong but seems locally plausible. This corresponds to the "stick figure" configuration. However the additional edges in the "extended stickfigure" configuration between knees and elbows can resolve the errors.

IV When joints are occluded, the pairwise terms help finding a probable pose. In both examples shown here, the partially redundant edges of "2-hop" stabilize the pose estimation against local outliers.

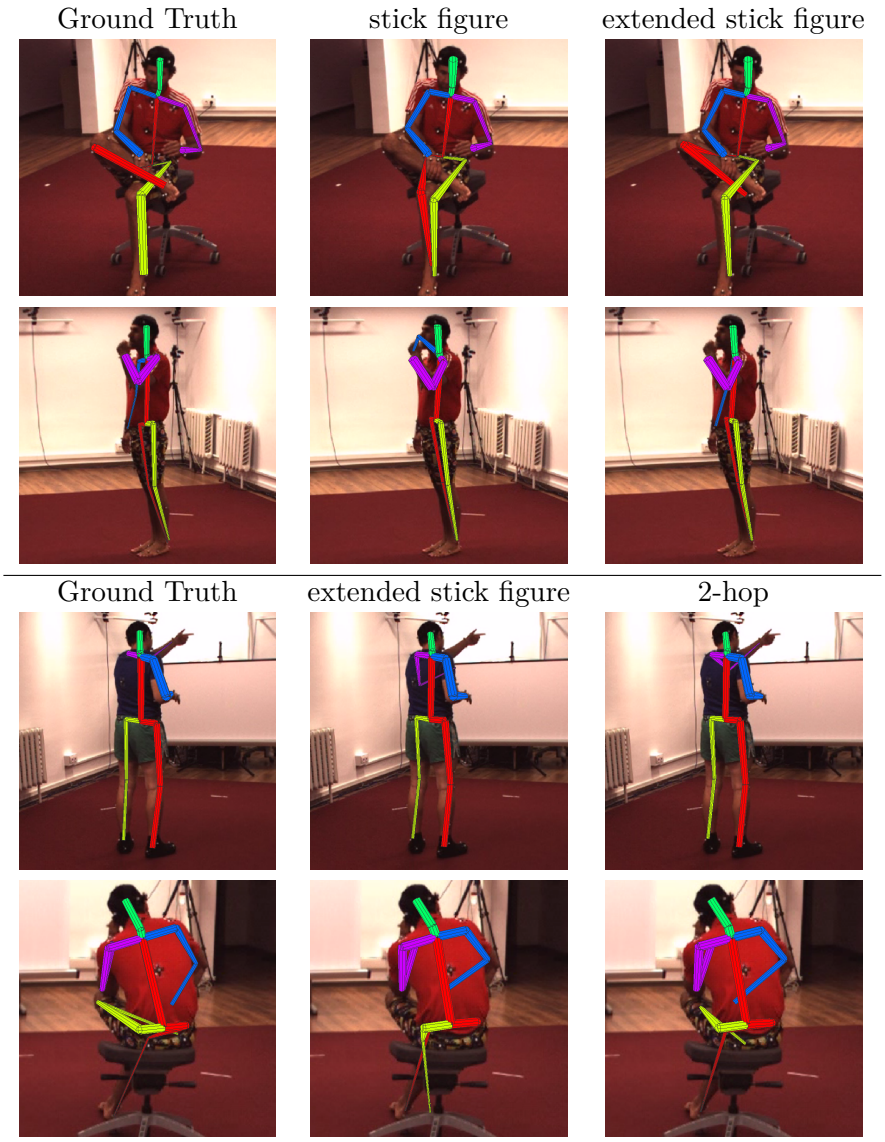
V We show three instances where the human pose estimation fails to predict the human pose accurately. In the first case the pairwise predictions are erroneous, falsifying even correct unary localizations. In the other two cases, the unary fails to predict certain joints completely, not allowing for a recovery with our technique.

In conclusion, more edges result in a more stable output, as outliers are balanced

out by complementary edges. We observe that some edges are more informative and provide more reliable information than others.



continued on the next page



continued on the next page

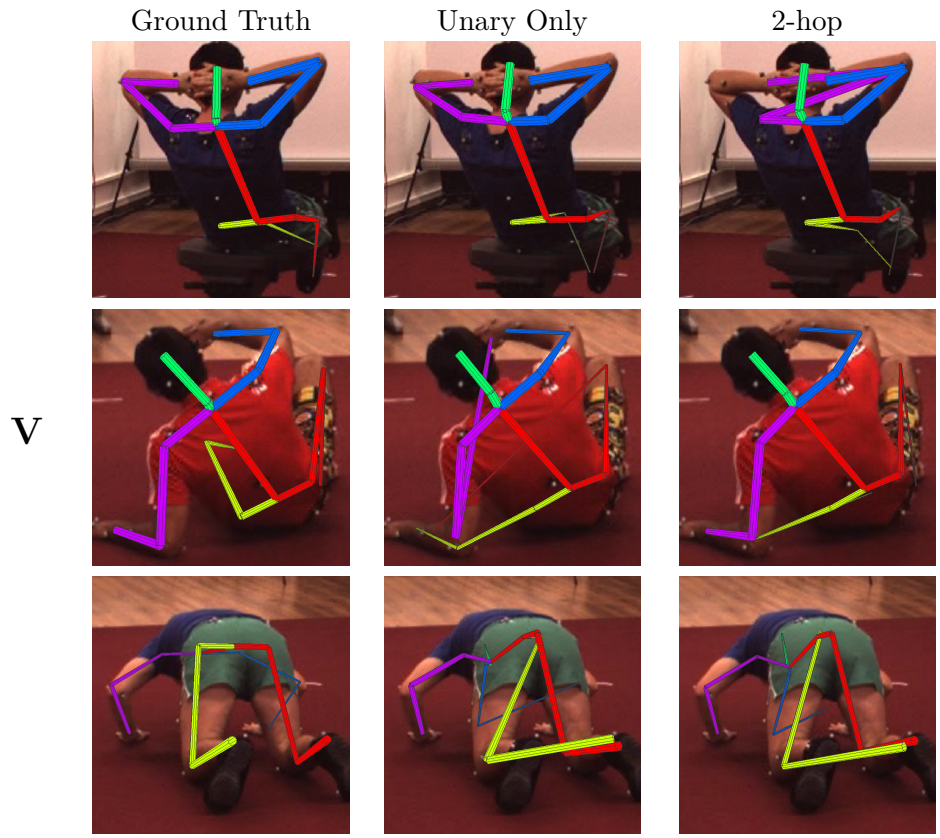
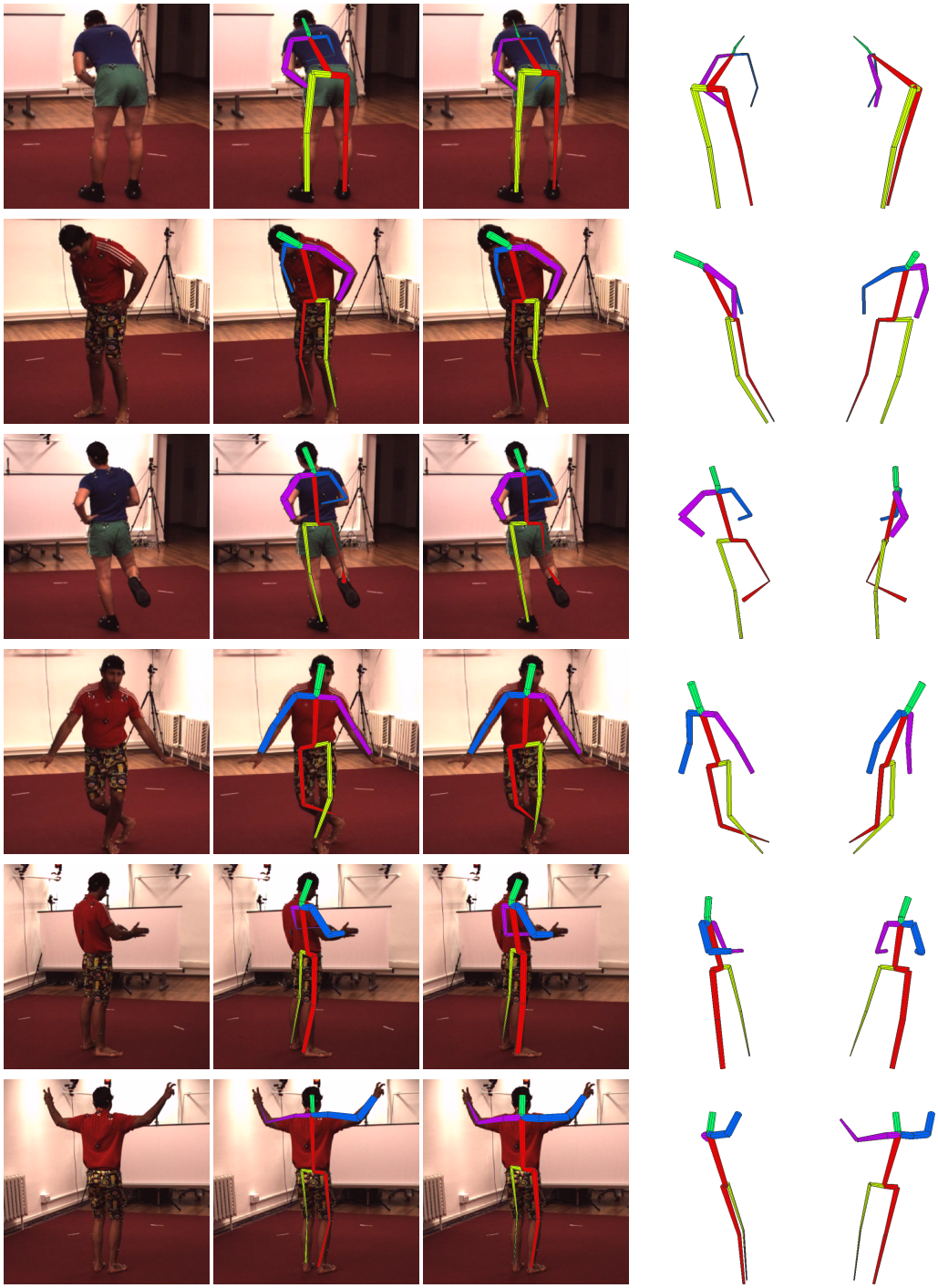


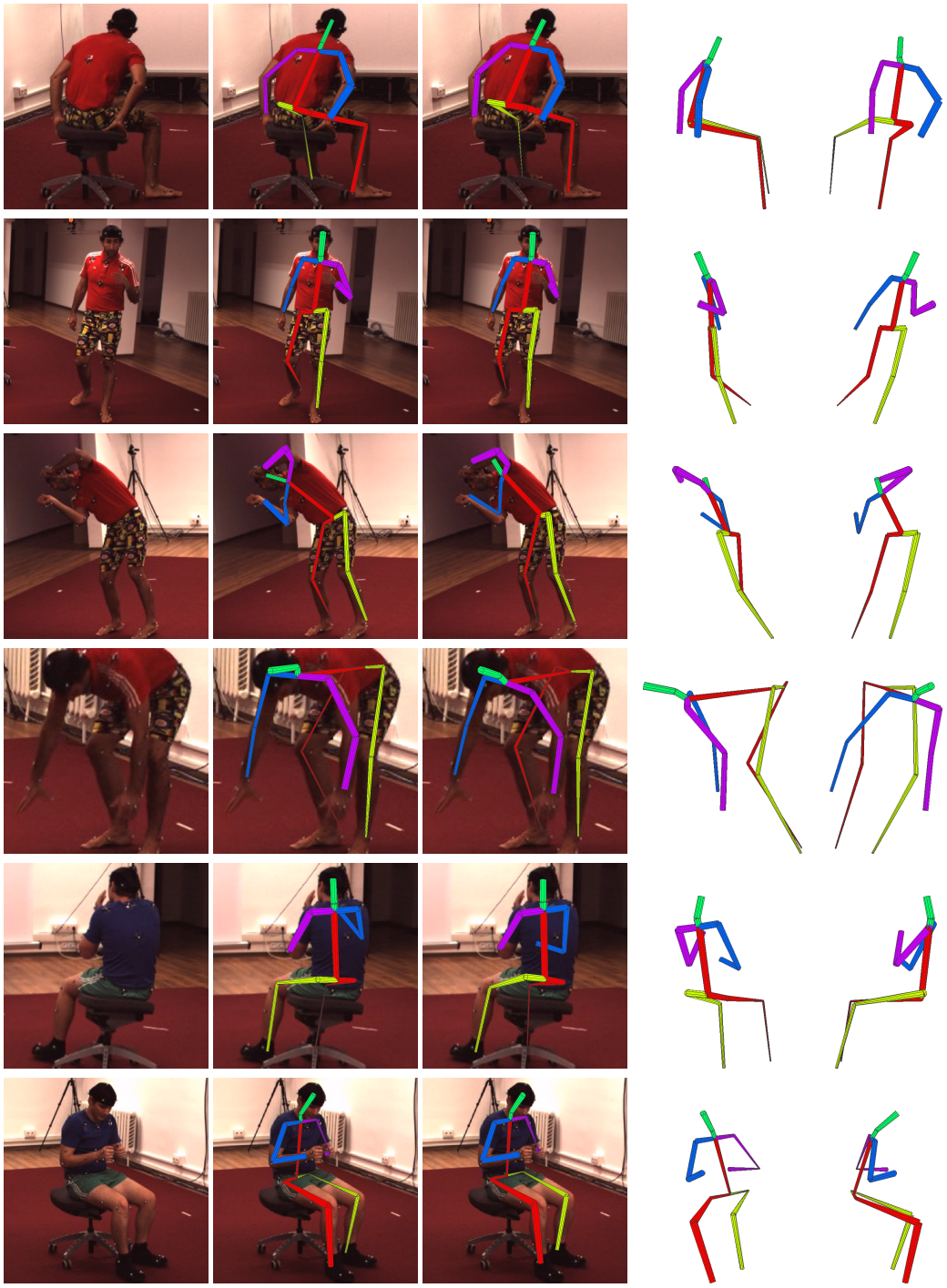
Table 4.5: Human pose estimation results. The first column shows the ground truth pose, the remaining two columns show the pose determined from the unary potential alone or by ADMM with one of the four analyzed graph configurations, "center star", "stick figure", "extended stick figure" and "2-hop".

In Table 4.6 we show the estimated poses of instances in different action categories and two additional views to visualize the 3D structure of the poses. The results have been obtained using the "2-hop" configuration.

Input	Ground Truth	2-hop	-60°	60°
				

continued on the next page

Input	Ground Truth	2-hop	-60°	60°



continued on the next page

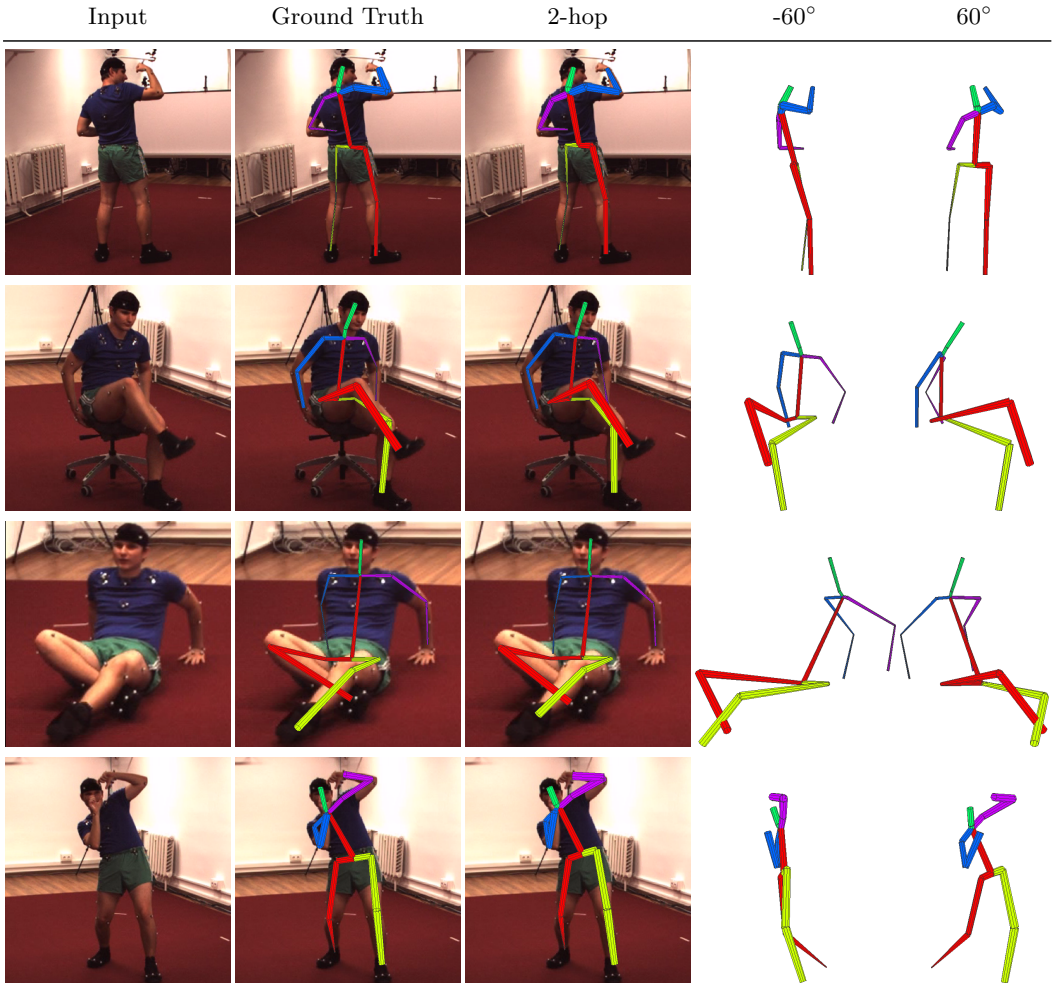


Table 4.6: Success cases on the Human3.6M dataset.

4.5.5 Improvement of 2D Joint Localizations

Our system is designed to jointly optimize image coordinates of joints and their respective depth. As elaborated in Section 4.1, this has the advantage that the depth can influence and potentially correct the 2D pose. We quantify the improvements by computing the 2D mean euclidean distance to the ground truth pose in 2D. The results are shown in Table 4.7. As expected, any optimization scheme improves the results of the "unary only" version. The best 2D error is achieved with the "2-hop" graph structure. The "center star" configuration performs slightly worse than the other graph structures, underlining that the other configurations are more faithful to the human body structure. Our approach clearly improves upon the results of [Zhou 2016] even though they use temporal information.

Approach	2D error (pixel)
Unary	6.06
Center Star	5.71
Stick Figure	5.65
Extended Stick Figure	5.65
2-hop	5.62
[Zhou 2016], no temporal smoothness	11.25
[Zhou 2016], with temporal smoothness	10.85

Table 4.7: 2D errors on the Human3.6M dataset. The error is computed as 2D mean euclidean distance to the ground truth on the testing set (Section 4.5.1).

4.5.6 Qualitative Evaluation on the Leeds Sports Dataset

So far we have analyzed the quantitative results on the Human3.6M dataset. We now augment the evaluation by a qualitative analysis on a second dataset, the Leeds Sports Dataset (LSP) [Johnson 2010]. This dataset is comprised of 2000 images of sport activities, ranging from Baseball to Badminton and Football. The dataset is challenging because of the wide range of poses and varying backgrounds, containing people and clutter of varying kinds. This distinguishes qualitatively the LSP from the Human3.6M dataset. To visualize the recovered 3D structure of the human pose we present several views of the same input image with superimposed stick figures. The underlying graph structure used for inference is the "2-hop"-configuration as presented above (Figure 4.5), but we chose the stick-figure representation for visualization as this serves the clarity of the illustration.

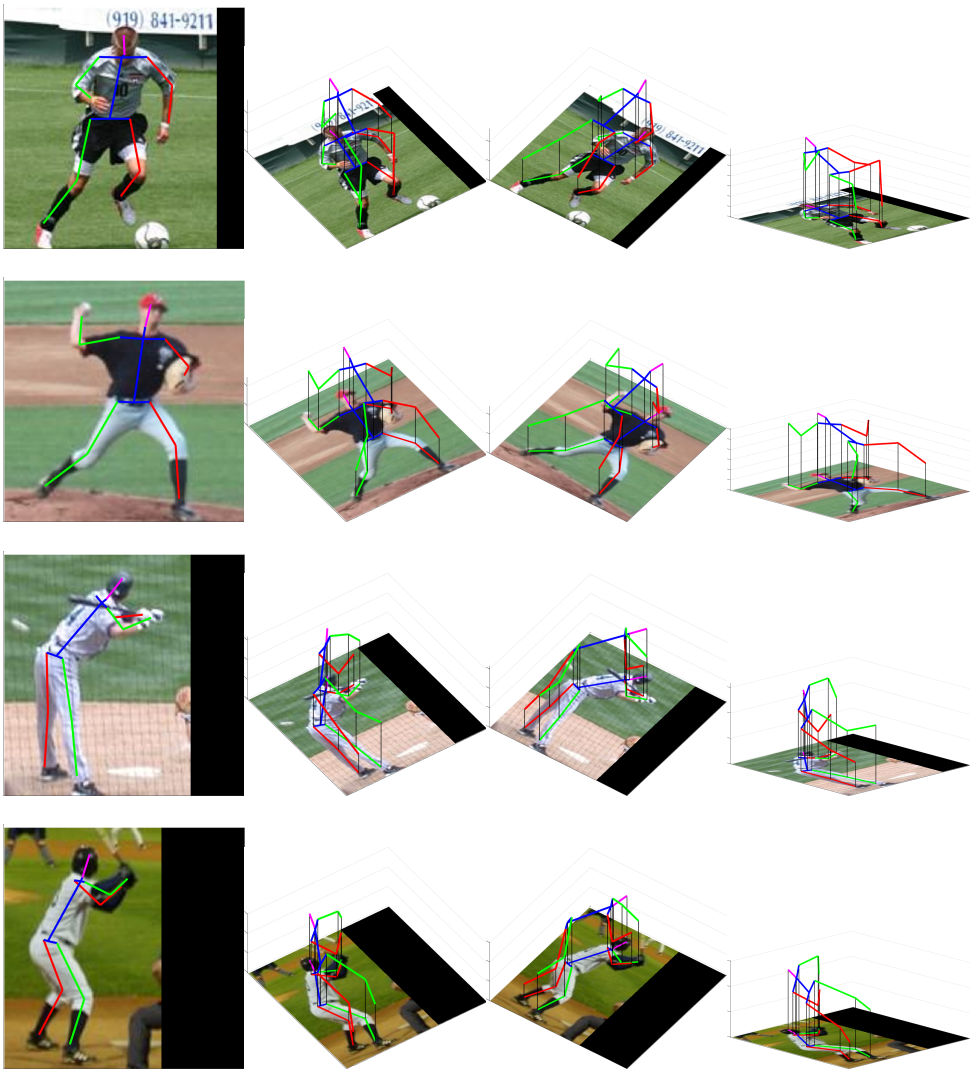


Figure 4.6: Monocular 3D pose estimation results on LSP dataset: we observe that our results transfer to unseen datasets, with highly different statistics from the Human3.6M dataset.

4.5.7 Runtime

We measure the inference runtime for different sizes of label space, Ω_{3D}. We measure separately the runtime of the two main components: creation of the Kd-Tree as necessary data structure and the tree traversal, the core part of the inference for which we implement the 3D Branch-and-Bound.

The Kd-Tree construction scales linearly in the number of voxels, since every voxel has to be touched at least once to determine the maximum over all voxels (Section 2.2.3). For a volume of $\Omega_{3D} = 20{\times}20{\times}20$ this takes on average 0.37 seconds. The trees need to be created only once in the first iteration of ADMM. In the remaining iterations the same unary trees can be reused without alteration. The ADMM term is treated separately, as described in Section 4.4.2. We note that we

use a single-threaded implementation for tree creation. A parallelization is possible to accelerate this task.

We view the time needed to solve one subproblem for one ADMM iteration, given the Kd-Trees for the unary potentials. In Section 4.4.2 we explain how the ADMM term can be bounded over intervals in its parametric form, instead of explicitly computing its values for each voxel. Due to the efficient bounding of the ADMM term, the computational overhead is marginal. The runtime of inference on the volume, $|\Omega_{3D}| = 20{*}20{*}20 = 8000$, takes on average 0.015 seconds.

Figure 4.7 relates the runtime of both subtasks, Kd-Tree creation and inference, with each other. As expected, we observe that the linear complexity for the Kd-Tree creation outweighs the logarithmic runtime for tree traversal even for small label spaces.

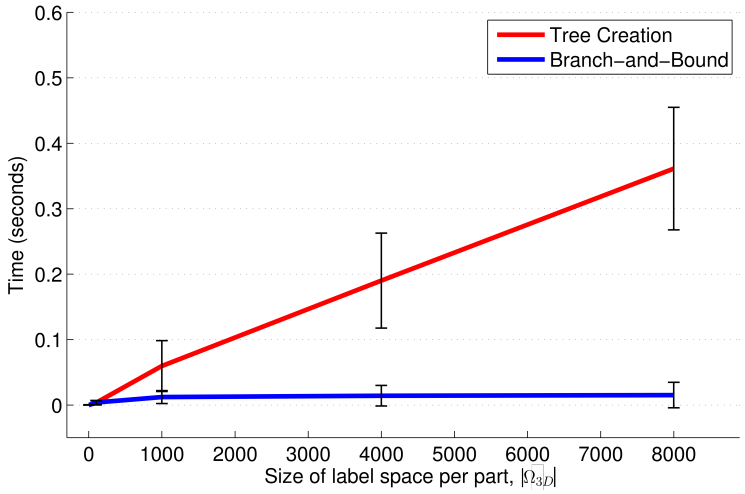


Figure 4.7: Runtime of Branch-and-Bound with varying size of label space, $|\Omega_{3D}|$. The time spent on building unary tree is displayed in red. The time spent in the Branch-and-Bound algorithm is depicted in blue.

Chapter 5

Future Work

Contents

5.1 Conclusion 103
5.2 Future Work 104
5.2.1 Applications 104
5.2.2 Optimization Algorithm 105

5.1 Conclusion

In this work we have presented two main tasks: (i) Efficient 3D object detection based 2D unaries; (ii) Full 3D inference as a deep network layer in combination with a graph decomposition strategy to deal with loopy graph structures;

(i) **Efficient 3D object detection based 2D unaries**
In Chapter 3 we advance the state of the art in 3D object detection. We capture viewpoint dependent variation by a combination of a small number of viewpoint-specific models. Each viewpoint-specific model can handle certain viewpoint variation by letting object parts move in relation to each other. To this end, we present a 3D Deformable Part Model, where every part node can move freely in space. The unary features are in 2D, so we take advantage of this structural particularity and devise a tailored inference algorithm. The proposed optimization algorithm does not trade accuracy for speed by decoupling the 2D and 3D components, but performs joint inference in 3D without materializing the unary potentials as 3D volumes. This enables inferece with 100 nodes and a volumetric labelspace within a fraction of a second.

(ii) **3D Inference as a deep network layer**
In Chapter 4 we integrate 3D Deformable Part Models into CNNs, enabling the neural network to perform inference on pairwise MRFs. MRF inference in 3D is exact for star-shape graphs and reaches close to real-time performance. We additionally extend the star-shaped graphs to more flexible, loopy graphs. This is achieved by applying ADMM, a graph decomposition algorithm that iteratively steers partial solutions towards an agreeable collective solution. We integrate the extra parametric term directly into our inference algorithm as to minimize computational overhead. A number of varying graph structures is

evaluated with regard to its performance for human pose estimation. Pairwise terms beyond the kinematic body structure offer additional merit.

In our approach, one single network predicts the unary potentials as well as the parameters for the pairwise potentials. The results confirm that these different aspects of the same data can complement each other. We show that the combination of local appearance terms and far-fetching pairwise terms provides gains over a purely local approach. We demonstrate the potential of our approach by improving the state of the art in 3D human pose estimation.

5.2 Future Work

In the previous section we summarized the contributions of this thesis. For future investigation we strive to advance our work in terms of applications as well as in technical aspects.

5.2.1 Applications

Multi-view 3D Tracking It is desirable for research purposes to minimize conditions and requirements in setting and setup. For real-world applications the algorithm has to be tailored to the given data and conditions, even if this does not result in the most general algorithm. Consider a surveillance scenario, where a number of cameras is monitoring humans' actions. To optimize performance it is useful to include subsequent camera frames and the geometric setup of the recording cameras. The former equates to 3D tracking of human pose, the latter has ties to wide-baseline stereo vision.

Precursors in literature include the works [Lepetit 2004, Pilet 2005] who propose wide-baseline point matching, without regarding object structures. Closer in setup and objective is [Starck 2003], where a human model is animated to match frames recorded in a multi-view camera setup. This requires a studio setup, an elaborate matching procedure, and relies heavily on a computer created model to well represent the image data. A pictorial body model is used in [Knoop 2006], where 3D and 2D camera inputs are fused to find the best matching pose in an involved and heuristic pipeline.

In continuation of this work, one idea is to first create unary and pairwise terms as proposed in Chapter 4 for every camera and frame, and then find the best human pose over time and in several camera frames. Making use of our advances in 3D optimization, the pose can be optimized jointly for all frames. To this end, one can establish time-links, similarly to [Zhou 2016, Cherian 2014], across different time frames. To account for the cameras' geometric setup, a common coordinate system can be established. One can follow the ideas of [Newcombe 2011, Curless 1996] and overlay 3D unary volumes after projecting them into the common coordinate system. To avoid large 3D volumes, alternatively a sparse candidate representation can be implemented as in Chapter 4, where the estimated pose is penalized for its distance to candidates in individual frames.

Domain Specific Knowledge The energy formulation of DPMs provides a principled way to incorporate domain knowledge. Pairwise terms are not restricted to the 3D position of two joints, but can also include further appearance features that refer to material properties or color histograms. On the example of human pose estimation, we can model human limbs, namely the limb's end-points, instead of individual body joints. Then the pairwise term can enforce a certain ratio of limb lengths, for instance a ratio of $1:1$ between left and right femoral and a the model's dimensionality increases, but also facilitates to capture constraints of limb lenghts or joint angles. We note that additional information and features can be stored in variables with higher dimensions. In the context of object detection, the fourth dimension might be occupied by a color value, a material property or an occlusion flag. For example, the pairwise term could constrain the parts to have a similar color or specularity, potentially ruling out false positives.

3D Multi-person Pose Estimation In Chapter 4 we focus solely on 3D human pose estimation with *one* person. Many current works deal with the arguably more difficult problem of multiple person pose estimation in 2D [Cao 2016, Pishchulin 2016, Insafutdinov 2016]. In 3D this problem has been investigated only in recent years [Andriluka 2010, Belagiannis 2014]. Our approach for inference with loopy graph structures allows to solve similar energies as proposed in [Andriluka 2010] or [Pishchulin 2016, Insafutdinov 2016] in 2D. The main idea is to introduce pairwise terms between joints of different people that act like a repellent force, separating detected people in the image.

5.2.2 Optimization Algorithm

As last point we want to emphasize that the algorithms presented in this thesis are not limited to the presented applications, but have the potential to be used as general optimization algorithm. Its guarantee to find the global optimum and best-case logarithmic runtime are intriguing properties for discrete optimization problems. In the light of this thesis we envision to extend the Branch-and-Bound algorithm into higher dimensions, extended it with novel energy terms and higher level terms.

Appendix A

DPMs for Viewpoint Invariant Detection in Robotics

Contents

A.1 Object Detection for Identity Anchoring 107
A.1.1 Identity Anchoring 108
A.1.2 Exigence of Practical Applications in Robotics 108
A.1.3 Previous Approaches to Object Detection in Robotics 110
A.2 Efficient Implementation 112
A.2.1 DPM Training 112
A.2.2 Data Collection 113
A.2.3 Inference Algorithms for Object Detection 114

In this chapter we will describe the object detection component we have implemented in the course of the RECONFIG project. As of the beginning of the RECONFIG project, utilizing DPMs was the state of the art in object detection. We implemented this as modular block in the Robot Operating System (ROS), a widely used framework in robotics. In such a way we provide easy access to efficient object detection with DPMs to the hands of the robotics community. The focus lies on the challenges of implementing for a real-world setup and practical usability than on theoretical advances in the field of computer vision. This chapter is thus different to the following chapters in that it does not extend the state of the art in computer vision, but sheds light on the applicability of computer vision algorithms in a practical robotics setting. We pursue two objectives with this chapter: Firstly, we emphasize the importance of putting theory to practice, discuss its specific challenges and make these methods available to the research community. Secondly, we describe an implementation of an object detection system with DPMs and illustrate the capacities and limitations of object detection with DPMs of [Felzenszwalb 2010b, Dubout 2012]. We advance upon these methods in Chapter 3 and 4.

A.1 Object Detection for Identity Anchoring

In robotics, object detection is a fundamental element of many applications. Either an object needs to be grasped or avoided, any kind of interaction requires either

perfect predetermined knowledge of all objects in the scenario or an object detection method to determine the existence and location, eventually also the state of objects in the scenario. For research purposes and in controlled environments, easily recognized visual markers are often used as tool to simplify object detection. In a more general, real-world scenario this is often not possible, but additional properties are desired. We argue that current solutions do not fulfill these needs and propose the use of DPMs in the next section.

A.1.1 Identity Anchoring

Identity anchoring is a fundamental element of communication of any kind, namely agreeing on a common symbol to describe a certain object. To cooperate on a given task, two agents need to have the same understanding of the physical object of interest. Assume that agent A wants to transmit the identity of "car" to agent B, together with the symbol "car", in order to establish the relationship between symbol and identity. In order to minimize hardware requirements and mutual understanding, we transmit the identity of the object to agent B by indicating a real-world instance in its surroundings. Agent B is then free to create a suitable representation of this object by itself, such as a 3D model, anchoring the corresponding symbol "car".

For agents with different means of perception, identity anchoring can be achieved via implicit communication. We propose to mimic human communication by letting the robot agent A perform a pointing gesture towards the object. Agent B perceives the pointing gesture and infers the correct object identity and connects it with the transmitted symbol. Using pointing gestures removes the need of having a common coordinate system established, something that can be difficult to set up, for example in a natural desaster scenario. Additionally, pointing gestures possess the advantage that robot-to-robot communication becomes perceivable and interpretable by humans. In the RECONFIG project an agent detects an object in order to carify the meaning of a symbol for the other agents by pointing to it.

A.1.2 Exigence of Practical Applications in Robotics

There is a number of premises that have to be taken into consideration especially in a realistic, cooperative robotics scenario. From here we identify a number of desired properties of object detection systems that constitute a demanding practical challenge.

Uncontrolled Environment The environment may be uncontrolled, as is for many practical applications. Therefore the background can be cluttered, which leads to strong gradients that might distract classifiers. Additionally the background is dynamic in a real-world scenario, negating any possibilities of explicitly taking a certain difficult background into account during training. Hence the detection algorithm has to deal with general backgrounds during test time.

Moderate Occlusions It can not be guaranteed that the sought object is com-

pletely visible due to other objects in the scene obstructing the view of a robotic agent. An object detection system needs to be able to handle at least slight occlusions with other objects.

Varying Viewpoints A typical characteristic of robots is to be able to move around in the environment. Thus the camera records images of objects from any viewing angle of the viewing sphere. One approach is to move around in the scene to obtain a more opportune viewing angle, but this costs time and energy and secondly might not be possible to physical constraints of the scene. Being able to detect objects from any potential viewpoint is important in uncontrolled environments.

Interactive Framerates In an interactive environment, robots have to be aware of changes in their environment. Specifically, the robot might have to observe permanently its environment while moving around in order to detect new objects of interest. A standard camera delivers a constant stream of video frames of 24 frames per second. Assuming that changes happen in a speed comparably to human interactions and robotic movements, it seems reasonable to expect a result at least at 1-2 frames per second. This involves that the vision module responds in fractions of a second, so that the robot can react in interactive frame rates. This is required when cooperating with humans, for example a household robot, or dynamic processes, like robots playing football.

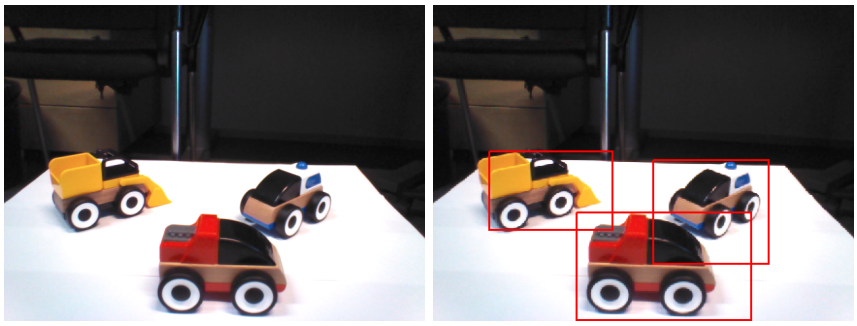
Sensory Modalities The quality of and equipment with sensors may vary considerably. In context of the RECONFIG project, we consider two different sensors: A Kinect camera recording RGB-D and a simple RGB camera. As to provide most general solutions, we opt to work on minimal means, specifically with an RGB image without depth channel. The resolution of the cameras is limited to a VGA-sized images (480×360 pixels).

Limited Computational Resources A fully independent robotic agent is naturally limited in the resources available to it to process sensor data and make decisions. This is especially true for graphics cards, not only because of the additional weight of the GPU and the required cooling, but also due to the higher power consumption. If the algorithm is to be deployed on the robot itself, the available computing resources are severely diminished in comparison to a modern working station with a GPU and large main memory. Although we run all computations on separate hardware, we consider this a viable point to increase the range of applications where resources are limited, and refrain from using GPUs and memory intensive computations.

Trainable Models for Intra-Class Variation Typically a robot's action is not defined by the color or texture of an object, but its identity and member of an object class. Therefore we want to create an object detector that does not only determine the presence of a specific object instance, but detects the instance of an object class. The visual appearance may vary considerably between instances of this class. For the RECONFIG project we train a single model to detect a set of toy cars (Figure A.1). These cars can be put together in varying configurations, but shall still be detected by the object detector.

Multiple Objects For every-day scenes it is common to encounter several

object instances of the same class, such as several cars on the street. Likewise it is possible that there are several objects of the same class in a robotics scenario, for example a number of toy cars on the floor. If the robot's task is to pick up all toy cars, all of them are relevant and should to be detected.



(a) (b)

Figure A.1: Left: Example of multiple toy cars. Right: Toy car bounding boxes obtained by our DPM object detector. We train a single model that covers the visual variability of all three car configurations. The objects are detected within one pass of the detection algorithm.

A.1.3 Previous Approaches to Object Detection in Robotics

There is a number of approaches implemented for ROS that enable object detection "out of the box". We characterize these and show that they can not cope with the challenges identified above. They all pose certain conditions either on the camera's modalities or on the scene and object, for example a dominant plane and depth sensors [ROS 2012b] or strongly textured objects [ROS 2012a].

The ROS `tabletop_object_detector` [ROS 2012b] provides simple object detection for objects on a flat supporting surface. This method subdivides 3D object detection into object segmentation and object recognition. Segmentation is performed by determining the dominant plane with a RANSAC scheme. Hereby subsequent segmentation of the remaining objects is simplified. The object is recognized by fitting the segmentation to available object models in a ICP-motivated procedure. This method requires the existence of a dominant plane in the image, additionally objects can not overlap to not disturb segmentation and finally the available models have to fit the extracted point cloud accurately.

A more flexible approach is implemented in the ROS `object_recognition_tod` package [ROS 2012a]. In a bag-of-features approach, 2D features are extracted from the input image and a RANSAC scheme is applied to find the object's position according to the model's features. A similar approach is taken by the ROS `Find-Object` package [Labbé, M. 2011], basing detection on recognizing individual features of the model within the input image. To avoid mismatches, matching features with a Nearest Neighbor Distance Ratio below a certain threshold are

discarded. The Nearest Neighbour Distance Ratio is computed as the distance between the two closest matches. Both approaches do not require a special geometry of the scene, but depend on texture on objects as features. As is typical for bag-of-feature approaches, there no structure between keypoints is enforced, simplifying the computation at the price of reduced precision.

Finally there are approaches that require depth as input. In recent years, the Kinect camera has become almost commodity hardware. A depth value is calculated for every pixel from the distortion of an infrared pattern, emitted by the Kinect itself. While this provides additional clues for the object detector, the additional hardware is an additional requirement. For a re-configurable environment, the Kinect camera can not be assumed. Thus approaches like [ROS 2012b, Lysenkov 2013b, Lysenkov 2013a, Prankl 2015, Astua 2014] can not or can only partially be applied.

In the attempt to amend challenge of detecting transparent or highly translucent objects, [Lysenkov 2013b, Lysenkov 2013a] search consequently "holes" in the depth channel of a certain shape. Therefore during training, a 3D model of the object is built and template silhouettes specific for each considered viewpoint are created from this model. Since this method capitalizes specifically on the properties of the Kinect camera, this approach is not applicable with RGB input alone. Additionally, this approach will break down for overlapping objects or for objects in front of non-responsive backgrounds, for instance a highly specular surface, not allowing to extract the silhouette of the object.

The `v4r_ros_wrappers`-package is a ROS wrapper for the V4R library based on [Prankl 2015]. They form models of 3D point clouds by merging at least partially overlapping 3D images of Kinect cameras. They advocate to use the method of [Aldoma 2013] to perform inference with the created models. Herein a number of recognition pipelines is applied to the same input data, producing a number of detection hypotheses. Each pipeline is based on different features, highlighting either local or global, 2D or 3D features. These detection hypotheses are fused in the final verification stage, designed to assert geometrical consistency with the scene. A similar approach is proposed in [Astua 2014], where a method is devised to combine and profit specifically from 2D and 3D cues. From depth images, contours are extracted and a relation is established to the model's contours over correlations. From RGB images, sparse SURF descriptors [Bay 2006] are extracted and matched with the model's features using an approximate Nearest Neighbours search [Muja 2009]. A third approach of combining different modalities like depth and 2D images is implemented in the ROS `object_recognition_linemod` package. Based on [Hinterstoisser 2011], features of multiple modalities are combined to yield an improved detection result. They argue that RGB template matching by means of image gradients breaks down under heavy background clutter and 3D surface normals can heal this weakness by providing complementary information. To combine both kinds of gradients, a similarity measure between object templates and the gradients of the RGB-D input is computed in a sliding window fashion. Although these methods are applicable on 2D images alone, their detection performance would

suffer from the lack of 3D complements.

We notice that all of above presented 3D methods rely on creating a number of templates of the sought object to deal with viewpoint variation. This always necessitates the trade-off between speed and detection rate. Inference in 3D is avoided to retain close real time performance, as it is often required in robotics. Our proposed method also avoids 3D inference, but our models are robust towards viewpoint changes, requiring a small number of viewpoint-specific models to achieve high detection rates.

A.2 Efficient Implementation

In this section we describe the practical realization of our object detection system. It fulfills the identified requirements to be trainable for varying object classes, differing viewpoints and deformable objects, detection is based on RGB images and runtime is close to real time. We first describe the training process in the robotics environment. Then, for inference, we improve the baseline OpenCV implementation with the accelerated approach of [Dubout 2012].

The goal is to identify a sought object in the current camera frame. The location of the object is returned as bounding box around the object in the image plane. Using the center of the bounding box, together with the robot's geometry and viewing direction, the intersection with the floor plane is computed, which gives the approximate 3D position of the object (Figure A.2). Based on this, further actions can be performed on the object by the robot, for instance to indicate the objects position to other robots.

A.2.1 DPM Training

We train the DPM mixture model discriminatively for every object class using the code by [Girshick 2012, Felzenszwalb 2008]. A mixture model is constituted by a set of individual object models, called mixtures. Each mixture reflects a part of the variability of the object that a single DPM would not be able to capture. This variability may come from different viewpoints, deformation states or instance variability. We follow this approach and train a mixture model using latent SVM, assigning training samples to mixtures based on a heuristic. The mixture assignments are modelled by latent variables and are updated alternating with the models' parameters. The algorithm is initialized by matching the aspect ratio of annotated bounding boxes. After updating the models parameters, the instance assignments are remade to lie in the best matching mixture. By inspecting the resulting models, we find that this roughly approximates a viewpoint classification (compare Figure A.1, rows 4 and 5 versus rows 6 and 7). We set the number of mixtures and object parts according to the visual complexity and variability of the object class. The simplest object, a single red ball without noticeable texture (Figure A.1, first row), can be well represented by a single mixture with only 1 center or root node. The object is almost invariant to viewpoint variations and there is no intra-class

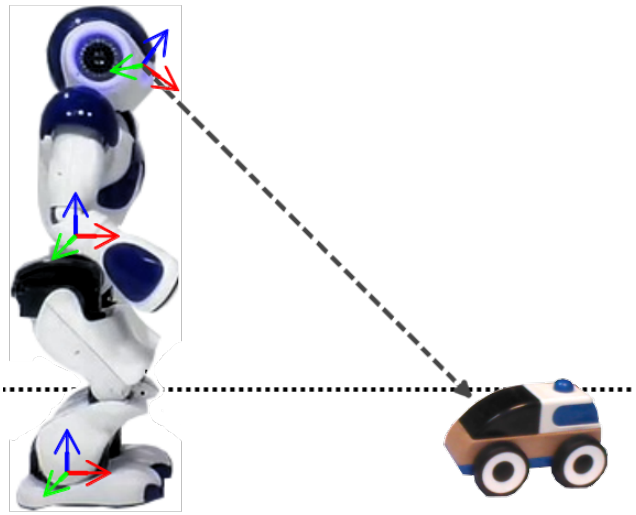


Figure A.2: Determination of 3D coordinates based on 2D image coordinates and the robot's geometry. The robot's viewing ray to the object, determined by the robot's joints and the center of the object's bounding box, is intersected with the floor plane to determine the object's 3D position.

variation requiring several mixtures. In contrary, the toy cars (Figure A.1, third column) consist of a number of configurable parts, from which we build three prototypical instances in the training set. This model is trained with 6 mixtures and 8 object parts each, to reflect the instance variability and effect of viewpoint changes. We report a training time of up to 10 hours on a single machine for models with several mixtures and parts, but less than 1 hour for the simple ball model.

A.2.2 Data Collection

For each object we train a DPM [Felzenszwalb 2008, Girshick 2012] from training data we collect in a laboratory environment. Object instances spanning the intra-class variability are photographed from varying viewpoints. We annotate these images by hand with a 2D bounding box. As such we create a training set of positive samples. We exemplify our dataset created for various toys in Figure A.1, rows 1 to 3. Depending on the object's complexity and intra-class variability we create between 60 and 674 positive instances with annotations. As negative training set we use the PASCAL VOC dataset [Everingham 2010, Everingham b], as none of their images contain the toy objects we use. This dataset contains over 11k images, presenting a wide variety of natural scenes and objects. Empirically this yields high recall with few false positives in our application setup. We demonstrate that augmenting the set of negative samples with images in the laboratory environment diminishes false positives in Figure A.3.

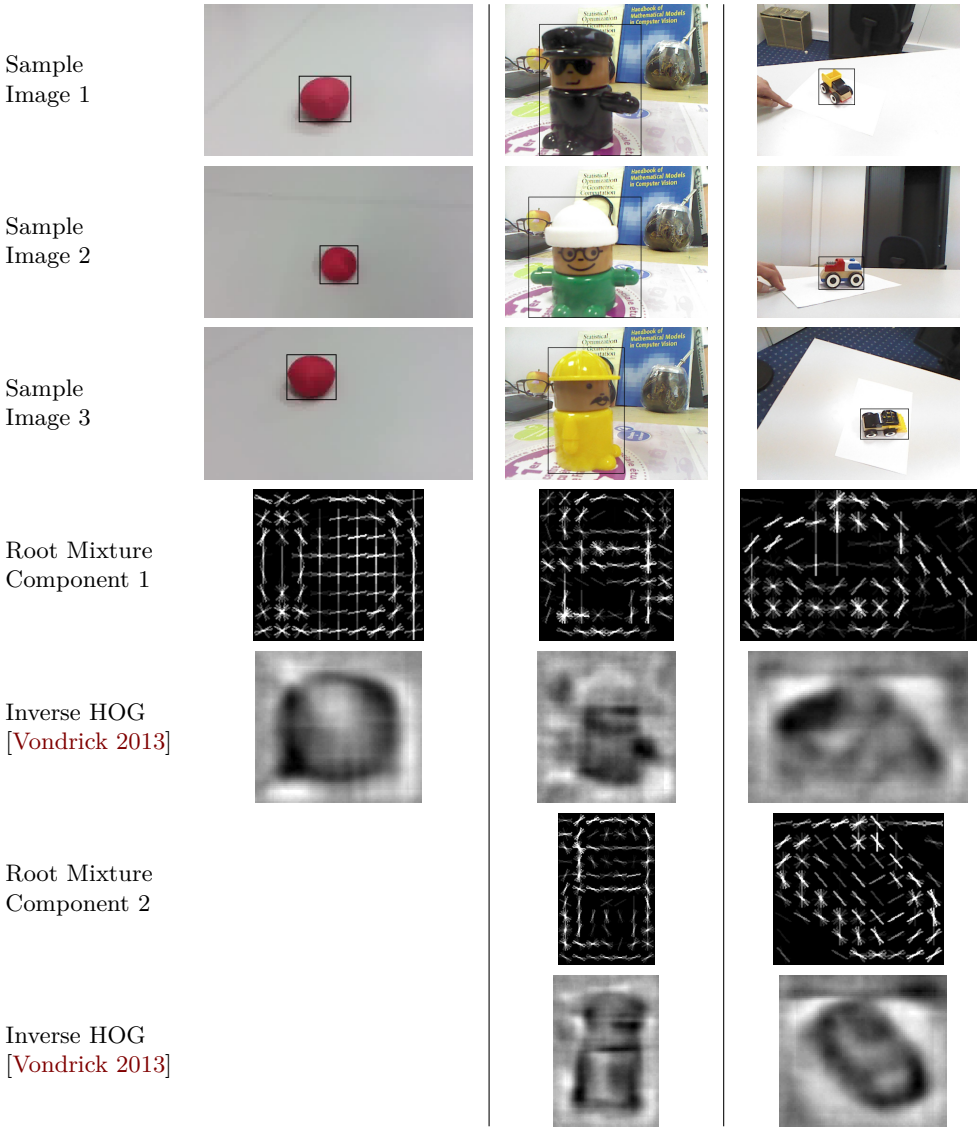
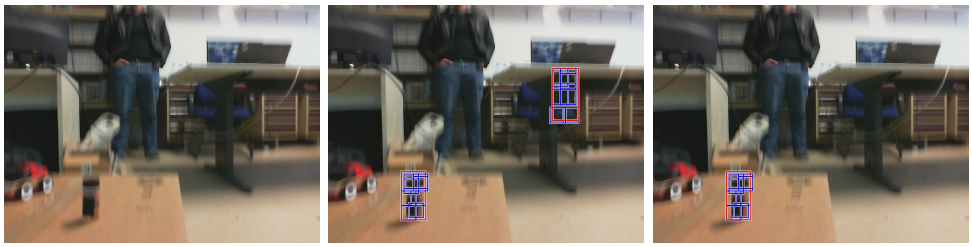


Table A.1: Visualization of our dataset of toys and the trained deformable part models, ordered by columns. Column one: The simple ball model; Row two: Various colored toy figures. Row three: Configurable toy cars. For each we show three samples from our dataset. The resulting model is illustrated in the rows 4-7. Row 4 show the HOG features of one mixture component's root node, row 5 displays Hoggles [Vondrick 2013] as an alternative visualization. Rows 5 and 6 mirror this for a second out of six mixture components. The ball model consists of only one component.

A.2.3 Inference Algorithms for Object Detection

Having trained a DPM for the sought object, we compare two implementations for inference: one relying on the OpenCV implementation [(OpenCV) 2014] for object detection and one relying on the implementation by [Dubout 2012]. We realize



(a) (b) (c)

Figure A.3: The effect of retraining DPMs in a specific environment. The model is a shampoo-bottle. On the left is the input image. In the middle a true positive and a false positive detection is found. On the right we show the result after retraining the model with additional negative samples (Figure A.4).

(a) (b) (c)

Figure A.4: Additional negative samples in the laboratory setup we add to the training set in order to diminish false detections of the shampoo-bottle.

both variants as ROS-nodes in the ROS framework to facilitate integration within robotics projects. The object detection modules take an input image from a ROS-topic and infers object locations within the image, based on the previously trained object model.

A.2.3.1 OpenCV Baseline Method

The OpenCV implementation is based directly on [Felzenszwalb 2010b]. The unary terms are calculated as convolution of part features with the features computed from the input image. GDTs with DP is used to compute potential detections (see Section 2.2.1). Often there are multiple, heavily overlapping detections around the location of an object. To yield only one detection per object, non-maximum suppression is applied as post-processing step. For this implementation we measure a runtime of approximately 2 seconds per frame in VGA resolution (480×360), 10 pyramid levels and 6 mixture components.

A.2.3.2 Accelerated Object Detection

The model trained in the previous section (Section A.2.1) scores high for objects within the model's specific capacity of visual variation. However, the model is trained on cropped and scaled image instances, simplifying training, but defining the object's scale. To extend the model's capabilities to instances of varying scale, the established approach is to compute feature pyramid. The input image is re-sized over various scales, the features are extracted on each scale and constitute thus the feature pyramid. Then inference is done on all levels of the pyramid.

The computational bottleneck is the computation of the convolutions between the image features and the model's weight vectors. Recognizing this, [Dubout 2012] replace convolutions by multiplications in frequency domain using Fourier transforms. Their work contributes in three main ways to a higher performance, but yield the same optimum, up to numerical precision. Firstly, the convolution via Fast Fourier Transform is efficient and is known to speed up convolutions. The larger are the filters, the greater is the benefit of using the Fourier Transformation for convolution. The DPM is calculated with rectangular filters with side length between 7 and 11, depending on the object's aspect ratio. Secondly, the Fourier Transforms of the model, the transformed weight vectors in the model, are cached and therefore do not have to be recomputed in subsequent inferences. This turns this computation into a static overhead like training, such that during test time the cached Fourier transforms are available to rapidly compute the unary potentials. Thirdly, [Dubout 2012] avoid saving the Fourier Transforms at all pyramid levels, but propose a patchwork approach. Specifically, input images of varying scales are composed into one patchwork image of a fixed size. The Fourier Transforms of the model are pre-computed for this patchwork size. This also reduces the computational expenses to transform the input image features. The implementation of [Dubout 2012] runs in 0.4 seconds in average. This is about 4-5 times faster than the OpenCV implementation with equal detection performance.

Bibliography

[Adelson 1984] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt and Joan M Ogden. *Pyramid methods in image processing.* RCA engineer, vol. 29, no. 6, pages 33–41, 1984. (Cited on page 15.)

[Akhter 2015] Ijaz Akhter and Michael J Black. *Pose-conditioned joint angle limits for 3D human pose reconstruction.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1446–1455, 2015. (Cited on page 71.)

[Aldoma 2013] Aitor Aldoma, Federico Tombari, Johann Prankl, Andreas Richtsfeld, Luigi Di Stefano and Markus Vincze. *Multimodal cue integration through hypotheses verification for RGB-D object recognition and 6DOF pose estimation.* In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 2104–2111. IEEE, 2013. (Cited on page 111.)

[Andriluka 2010] Mykhaylo Andriluka, Stefan Roth and Bernt Schiele. *Monocular 3d pose estimation and tracking by detection.* In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 623–630. IEEE, 2010. (Cited on page 105.)

[Andriluka 2014] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler and Bernt Schiele. *2d human pose estimation: New benchmark and state of the art analysis.* In Proceedings of the IEEE Conference on computer Vision and Pattern Recognition, pages 3686–3693, 2014. (Cited on pages 70, 71 and 90.)

[Astua 2014] Carlos Astua, Ramon Barber, Jonathan Crespo and Alberto Jardon. *Object detection techniques applied on mobile robot semantic navigation.* Sensors, vol. 14, no. 4, pages 6734–6757, 2014. (Cited on page 111.)

[Aubry 2014] Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell and Josef Sivic. *Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3762–3769, 2014. (Cited on pages 43 and 44.)

[Barron 2012] Jonathan T Barron and Jitendra Malik. *Color constancy, intrinsic images, and shape estimation.* In European Conference on Computer Vision, pages 57–70. Springer, 2012. (Cited on page 44.)

[Barron 2015] Jonathan T Barron and Jitendra Malik. *Shape, illumination, and reflectance from shading.* IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 8, pages 1670–1687, 2015. (Cited on page 44.)

[Batra 2010] Dhruv Batra, Andrew C Gallagher, Devi Parikh and Tsuhan Chen. *Beyond trees: MRF inference via outer-planar decomposition.* In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 2496–2503. IEEE, 2010. (Cited on page 78.)

[Bay 2006] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. *Surf: Speeded up robust features.* Computer vision–ECCV 2006, pages 404–417, 2006. (Cited on page 111.)

[Belagiannis 2014] Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab and Slobodan Ilic. *3D pictorial structures for multiple human pose estimation.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1669–1676, 2014. (Cited on page 105.)

[Bellman 1952] Richard Bellman. *On the theory of dynamic programming.* Proceedings of the National Academy of Sciences, vol. 38, no. 8, pages 716–719, 1952. (Cited on page 27.)

[Bellman 2015] Richard E Bellman and Stuart E Dreyfus. Applied dynamic programming. Princeton university press, 2015. (Cited on page 27.)

[Blanz 1999] Volker Blanz and Thomas Vetter. *A morphable model for the synthesis of 3D faces.* In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999. (Cited on pages 44 and 45.)

[Blaschko 2016] Matthew B Blaschko. *Slack and Margin Rescaling as Convex Extensions of Supermodular Functions.* arXiv preprint arXiv:1606.05918, 2016. (Cited on page 36.)

[Bogo 2016] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero and Michael J Black. *Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image.* In European Conference on Computer Vision, pages 561–578. Springer, 2016. (Cited on page 70.)

[Booth 2017] James Booth, Epameinondas Antonakos, Stylianos Ploumpis, George Trigeorgis, Yannis Panagakis, Stefanos Zafeiriou*et al. 3D Face Morphable Models "In-the-Wild".* In Proceedings of the IEEE Conference on ComputerVision and Pattern Recognition, 2017. (Cited on page 44.)

[Bosch 2007] Anna Bosch, Andrew Zisserman and Xavier Munoz. *Representing shape with a spatial pyramid kernel.* In Proceedings of the 6th ACM international conference on Image and video retrieval, pages 401–408. ACM, 2007. (Cited on page 76.)

[Bourdev 2009] Lubomir Bourdev and Jitendra Malik. *Poselets: Body part detectors trained using 3d human pose annotations.* In Computer Vision,

2009 IEEE 12th International Conference on, pages 1365–1372. IEEE, 2009. (Cited on page 71.)

[Bourdev 2010] Lubomir Bourdev, Subhransu Maji, Thomas Brox and Jitendra Malik. *Detecting people using mutually consistent poselet activations.* In European conference on computer vision, pages 168–181. Springer, 2010. (Cited on page 18.)

[Boussaid 2014] Haithem Boussaid and Iasonas Kokkinos. *Fast and exact: ADMM-based discriminative shape segmentation with loopy part models.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4058–4065, 2014. (Cited on page 78.)

[Boyd 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato and Jonathan Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers.* Foundations and Trends® in Machine Learning, vol. 3, no. 1, pages 1–122, 2011. (Cited on pages 78, 79 and 80.)

[Bregler 2000] Christoph Bregler, Aaron Hertzmann and Henning Biermann. *Recovering non-rigid 3D shape from image streams.* In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 2, pages 690–696. IEEE, 2000. (Cited on pages 43, 48 and 63.)

[Burl 1998] Michael C Burl, Markus Weber and Pietro Perona. *A probabilistic approach to object recognition using local photometry and global geometry.* In European conference on computer vision, pages 628–641. Springer, 1998. (Cited on page 19.)

[Cao 2016] Zhe Cao, Tomas Simon, Shih-En Wei and Yaser Sheikh. *Realtime multi-person 2d pose estimation using part affinity fields.* arXiv preprint arXiv:1611.08050, 2016. (Cited on pages 14, 70 and 105.)

[Chang 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su*et al.* *Shapenet: An information-rich 3d model repository.* arXiv preprint arXiv:1512.03012, 2015. (Cited on page 43.)

[Chen 2014] Xianjie Chen and Alan L Yuille. *Articulated pose estimation by a graphical model with image dependent pairwise relations.* In Advances in Neural Information Processing Systems, pages 1736–1744, 2014. (Cited on page 70.)

[Chen 2016a] Ching-Hang Chen and Deva Ramanan. *3D Human Pose Estimation= 2D Pose Estimation+ Matching.* arXiv preprint arXiv:1612.06524, 2016. (Cited on page 71.)

[Chen 2016b] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L Yuille. *Deeplab: Semantic image segmentation with*

deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915, 2016. (Cited on pages 22 and 55.)

[Cherian 2014] Anoop Cherian, Julien Mairal, Karteek Alahari and Cordelia Schmid. *Mixing body-part sequences for human pose estimation.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2353–2360, 2014. (Cited on page 104.)

[Commons 2007] Wikimedia Commons. *Hausdorff distance.* `https://commons.wikimedia.org/wiki/File:Hausdorff_distance_sample.svg`, 2007. Accessed: 21/10/2017 under GNU Free Documentation License, Version 1.2 or later. (Cited on pages 4 and 63.)

[Cortes 1995] Corinna Cortes and Vladimir Vapnik. *Support-vector networks.* Machine learning, vol. 20, no. 3, pages 273–297, 1995. (Cited on page 36.)

[Crandall 2005] David Crandall, Pedro Felzenszwalb and Daniel Huttenlocher. *Spatial priors for part-based recognition using statistical models.* In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 10–17. IEEE, 2005. (Cited on page 19.)

[Curless 1996] Brian Curless and Marc Levoy. *A volumetric method for building complex models from range images.* In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 303–312. ACM, 1996. (Cited on page 104.)

[Dalal 2005] Navneet Dalal and Bill Triggs. *Histograms of oriented gradients for human detection.* In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005. (Cited on pages 12, 14, 15, 16 and 76.)

[Dantone 2013] Matthias Dantone, Juergen Gall, Christian Leistner and Luc Van Gool. *Human pose estimation using body parts dependent joint regressors.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3041–3048, 2013. (Cited on page 71.)

[Divvala 2012] Santosh K Divvala, Alexei A Efros and Martial Hebert. *How important are "deformable parts" in the deformable parts model?* In European Conference on Computer Vision, pages 31–40. Springer, 2012. (Cited on page 13.)

[Dollár 2009] Piotr Dollár, Zhuowen Tu, Pietro Perona and Serge Belongie. *Integral channel features.* 2009. (Cited on page 15.)

[Dubout 2012] Charles Dubout and François Fleuret. *Exact acceleration of linear object detectors.* Computer Vision–ECCV 2012, pages 301–311, 2012. (Cited on pages 107, 112, 114 and 116.)

[Eichner 2012] Marcin Eichner, Manuel Marin-Jimenez, Andrew Zisserman and Vittorio Ferrari. *2d articulated human pose estimation and retrieval in (almost) unconstrained still images*. International journal of computer vision, vol. 99, no. 2, pages 190–214, 2012. (Cited on page 77.)

[Everingham a] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. (Cited on page 26.)

[Everingham b] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. (Cited on page 113.)

[Everingham 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn and Andrew Zisserman. *The pascal visual object classes (voc) challenge*. International journal of computer vision, vol. 88, no. 2, pages 303–338, 2010. (Cited on pages 22, 38, 43, 62 and 113.)

[Fan 2005] Zhi-Gang Fan and Bao-Liang Lu. *Fast recognition of multi-view faces with feature selection*. In Computer vision, 2005. ICCV 2005. Tenth IEEE international conference on, volume 1, pages 76–81. IEEE, 2005. (Cited on page 46.)

[Felzenszwalb 2004] Pedro Felzenszwalb and Daniel Huttenlocher. *Distance transforms of sampled functions*. Rapport technique, Cornell University, 2004. (Cited on pages 3, 8, 28, 29 and 77.)

[Felzenszwalb 2005] Pedro F Felzenszwalb and Daniel P Huttenlocher. *Pictorial structures for object recognition*. International journal of computer vision, vol. 61, no. 1, pages 55–79, 2005. (Cited on pages 19, 27 and 77.)

[Felzenszwalb 2008] Pedro Felzenszwalb, David McAllester and Deva Ramanan. *A discriminatively trained, multiscale, deformable part model*. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008. (Cited on pages 13, 18, 19, 21, 38, 39, 44, 46, 54, 112 and 113.)

[Felzenszwalb 2010a] Pedro F Felzenszwalb, Ross B Girshick and David McAllester. *Cascade object detection with deformable part models*. In Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, pages 2241–2248. IEEE, 2010. (Cited on pages 3, 8, 27, 30 and 31.)

[Felzenszwalb 2010b] Pedro F Felzenszwalb, Ross B Girshick, David McAllester and Deva Ramanan. *Object detection with discriminatively trained part-based*

models. IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 9, pages 1627–1645, 2010. (Cited on pages 15, 16, 19, 20, 22, 27, 45, 66, 76, 107 and 115.)

[Fergus 2003] Robert Fergus, Pietro Perona and Andrew Zisserman. *Object class recognition by unsupervised scale-invariant learning*. In Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, volume 2, pages II–II. IEEE, 2003. (Cited on page 19.)

[Fergus 2005] Robert Fergus, Pietro Perona and Andrew Zisserman. *A sparse object category model for efficient learning and exhaustive recognition*. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 380–387. IEEE, 2005. (Cited on page 19.)

[Ferrari 2008] Vittorio Ferrari, Manuel Marin-Jimenez and Andrew Zisserman. *Progressive search space reduction for human pose estimation*. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008. (Cited on pages 71 and 77.)

[Fidler 2012] Sanja Fidler, Sven Dickinson and Raquel Urtasun. *3d object detection and viewpoint estimation with a deformable 3d cuboid model*. In Advances in neural information processing systems, pages 611–619, 2012. (Cited on page 45.)

[Fischler 1973] Martin A Fischler and Robert A Elschlager. *The representation and matching of pictorial structures*. IEEE Transactions on computers, vol. 100, no. 1, pages 67–92, 1973. (Cited on page 19.)

[Frie 1998] Thilo-Thomas Frie, Nello Cristianini and Colin Campbell. *The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines*. In Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98), pages 188–196, 1998. (Cited on page 36.)

[Girshick 2012] R. B. Girshick, P. F. Felzenszwalb and D. McAllester. *Discriminatively Trained Deformable Part Models, Release 5*. http://people.cs.uchicago.edu/ rbg/latent-release5/, 2012. (Cited on pages 1, 9, 13, 15, 16, 18, 54, 112 and 113.)

[Girshick 2014] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014. (Cited on pages 14, 18, 24 and 45.)

[Girshick 2015a] Ross Girshick. *Fast r-cnn*. In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015. (Cited on pages 24 and 72.)

[Girshick 2015b] Ross Girshick, Forrest Iandola, Trevor Darrell and Jitendra Malik. *Deformable part models are convolutional neural networks.* In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 437–446, 2015. (Cited on pages 3, 14, 23, 25 and 26.)

[Goodfellow 2009] Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe and Andrew Y Ng. *Measuring invariances in deep networks.* In Advances in neural information processing systems, pages 646–654, 2009. (Cited on page 54.)

[Güler 2016] Rıza Alp Güler, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou and Iasonas Kokkinos. *Densereg: Fully convolutional dense shape regression in-the-wild.* arXiv preprint arXiv:1612.01202, 2016. (Cited on pages 70 and 72.)

[Hammersley 1971] John M Hammersley and Peter Clifford. *Markov fields on finite graphs and lattices.* 1971. (Cited on page 20.)

[Harris 1988] Chris Harris and Mike Stephens. *A combined corner and edge detector.* In Alvey vision conference, volume 15, pages 10–5244. Manchester, UK, 1988. (Cited on page 14.)

[Hartley 2003] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003. (Cited on pages 14 and 41.)

[He 2000] BS He, Hai Yang and SL Wang. *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities.* Journal of Optimization Theory and applications, vol. 106, no. 2, pages 337–356, 2000. (Cited on page 82.)

[He 2014] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Spatial pyramid pooling in deep convolutional networks for visual recognition.* In European Conference on Computer Vision, pages 346–361. Springer, 2014. (Cited on page 15.)

[He 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep residual learning for image recognition.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. (Cited on page 75.)

[Hejrati 2014] Mohsen Hejrati and Deva Ramanan. *Analysis by synthesis: 3d object recognition by object reconstruction.* In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2449–2456. IEEE, 2014. (Cited on page 44.)

[Hinterstoisser 2011] S. Hinterstoisser S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab and V. Lepetit. *Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes.* 2011. (Cited on page 111.)

[Hornung 2013] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss and Wolfram Burgard. *OctoMap: An efficient probabilistic 3D mapping framework based on octrees.* Autonomous Robots, vol. 34, no. 3, pages 189–206, 2013. (Cited on page 41.)

[Huttenlocher 1993] Daniel P. Huttenlocher, Gregory A. Klanderman and William J Rucklidge. *Comparing images using the Hausdorff distance.* IEEE Transactions on pattern analysis and machine intelligence, vol. 15, no. 9, pages 850–863, 1993. (Cited on page 62.)

[Insafutdinov 2016] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka and Bernt Schiele. *Deepercut: A deeper, stronger, and faster multi-person pose estimation model.* In European Conference on Computer Vision, pages 34–50. Springer, 2016. (Cited on pages 70 and 105.)

[Ioffe 2001] Sergey Ioffe and David A. Forsyth. *Probabilistic methods for finding people.* International Journal of Computer Vision, vol. 43, no. 1, pages 45–68, 2001. (Cited on page 19.)

[Ionescu 2011] Catalin Ionescu, Fuxin Li and Cristian Sminchisescu. *Latent structured models for human pose estimation.* In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 2220–2227. IEEE, 2011. (Cited on page 71.)

[Ionescu 2014] Catalin Ionescu, Dragos Papava, Vlad Olaru and Cristian Sminchisescu. *Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments.* IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 7, pages 1325–1339, 2014. (Cited on pages 71, 90 and 91.)

[Jiang 2010] Hao Jiang. *3D human pose reconstruction using millions of exemplars.* In Pattern Recognition (ICPR), 2010 20th International Conference on, pages 1674–1677. IEEE, 2010. (Cited on page 71.)

[Joachims 2009] Thorsten Joachims, Thomas Finley and Chun-Nam John Yu. *Cutting-plane training of structural SVMs.* Machine Learning, vol. 77, no. 1, pages 27–59, 2009. (Cited on page 37.)

[Johnson 2010] Sam Johnson and Mark Everingham. *Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation.* In Proceedings of the British Machine Vision Conference, 2010. doi:10.5244/C.24.12. (Cited on pages 1, 7, 70, 71 and 100.)

[Kar 2015] Abhishek Kar, Shubham Tulsiani, Joao Carreira and Jitendra Malik. *Category-specific object reconstruction from a single image.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1966–1974, 2015. (Cited on pages 43, 45, 48, 54, 62, 63 and 64.)

[Ke 2018] Lipeng Ke, Ming-Ching Chang, Honggang Qi and Siwei Lyu. *Multi-Scale Structure-Aware Network for Human Pose Estimation*. arXiv preprint arXiv:1803.09894, 2018. (Cited on page 70.)

[Kinauer 2016] Stefan Kinauer, Maxim Berman and Iasonas Kokkinos. *Monocular Surface Reconstruction using 3D Deformable Part Models*. In Computer Vision–ECCV 2016 Workshops, pages 296–308. Springer, 2016. (Cited on pages 1, 8 and 9.)

[Kinauer 2017] Stefan Kinauer, Riza Alp Güler, Siddhartha Chandra and Iasonas Kokkinos. *Structured output prediction and learning for deep monocular 3d human pose estimation*. In International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 34–48. Springer, 2017. (Cited on pages 9 and 69.)

[Knoop 2006] Steffen Knoop, Stefan Vacek and Rüdiger Dillmann. *Sensor fusion for 3D human body tracking with an articulated 3D body model*. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 1686–1691. IEEE, 2006. (Cited on page 104.)

[Kokkinos 2011] Iasonas Kokkinos. *Rapid deformable object detection using dual-tree branch-and-bound*. In Advances in Neural Information Processing Systems, pages 2681–2689, 2011. (Cited on pages 3, 31, 33 and 59.)

[Kokkinos 2016] Iasonas Kokkinos. *Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory*. arXiv preprint arXiv:1609.02132, vol. 2, 2016. (Cited on page 22.)

[Koller 2009] Daphne Koller and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009. (Cited on page 19.)

[Kontaxakis 2018] Polychronis Kontaxakis, Khurram Gulzar, Stefan Kinauer, Iasonas Kokkinos and Ville Kyrki. *Robot-Robot Gesturing for Anchoring Representations*. IEEE Transactions on Robotics, 2018. To be published. (Cited on page 9.)

[Krizhevsky 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in neural information processing systems, pages 1097–1105, 2012. (Cited on page 24.)

[Kushal 2007] Akash Kushal, Cordelia Schmid and Jean Ponce. *Flexible object models for category-level 3d object recognition*. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. IEEE, 2007. (Cited on page 46.)

[Labbé, M. 2011] Labbé, M. *Find-Object.* http://introlab.github.io/find-object, 2011. accessed 2017-11-15. (Cited on page 110.)

[Ladicky 2013] Lubor Ladicky, Philip HS Torr and Andrew Zisserman. *Human pose estimation using a joint pixel-wise and part-wise formulation.* In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3578–3585, 2013. (Cited on page 70.)

[Lassner 2017] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black and Peter V Gehler. *Unite the people: Closing the loop between 3d and 2d human representations.* In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on page 70.)

[Le 2013] Quoc V Le. *Building high-level features using large scale unsupervised learning.* In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 8595–8598. IEEE, 2013. (Cited on page 54.)

[LeCun 1989] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard and Lawrence D Jackel. *Backpropagation applied to handwritten zip code recognition.* Neural computation, vol. 1, no. 4, pages 541–551, 1989. (Cited on page 23.)

[LeCun 1995] Yann LeCun, Yoshua Bengio*et al. Convolutional networks for images, speech, and time series.* The handbook of brain theory and neural networks, vol. 3361, no. 10, page 1995, 1995. (Cited on page 23.)

[LeCun 1998] Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, vol. 86, no. 11, pages 2278–2324, 1998. (Cited on page 23.)

[Lepetit 2004] Vincent Lepetit, Julien Pilet and Pascal Fua. *Point matching as a classification problem for fast and robust object pose estimation.* In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–II. IEEE, 2004. (Cited on page 104.)

[Li 2014] Sijin Li and Antoni B Chan. *3d human pose estimation from monocular images with deep convolutional neural network.* In Asian Conference on Computer Vision, pages 332–347. Springer, 2014. (Cited on pages 71 and 72.)

[Li 2015] Sijin Li, Weichen Zhang and Antoni B Chan. *Maximum-margin structured learning with deep networks for 3d human pose estimation.* In Proceedings of the IEEE International Conference on Computer Vision, pages 2848–2856, 2015. (Cited on page 90.)

[Liebelt 2010] Joerg Liebelt and Cordelia Schmid. *Multi-view object class detection with a 3d geometric model.* In Computer Vision and Pattern Recognition

(CVPR), 2010 IEEE Conference on, pages 1688–1695. IEEE, 2010. (Cited on page 43.)

[Lim 2014] Joseph J Lim, Aditya Khosla and Antonio Torralba. *Fpm: Fine pose parts-based model with 3d cad models*. In European Conference on Computer Vision, pages 478–493. Springer, 2014. (Cited on pages 43 and 44.)

[Lin 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C Lawrence Zitnick. *Microsoft coco: Common objects in context*. In European conference on computer vision, pages 740–755. Springer, 2014. (Cited on page 43.)

[Liu 2016] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C Berg. *Ssd: Single shot multibox detector*. In European conference on computer vision, pages 21–37. Springer, 2016. (Cited on page 18.)

[Long 2015] Jonathan Long, Evan Shelhamer and Trevor Darrell. *Fully convolutional networks for semantic segmentation*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015. (Cited on page 22.)

[Lowe 2004] David G Lowe. *Distinctive image features from scale-invariant keypoints*. International journal of computer vision, vol. 60, no. 2, pages 91–110, 2004. (Cited on page 14.)

[Lysenkov 2013a] Ilya Lysenkov, Victor Eruhimov and Gary Bradski. *Recognition and pose estimation of rigid transparent objects with a kinect sensor*. Robotics, page 273, 2013. (Cited on page 111.)

[Lysenkov 2013b] Ilya Lysenkov and Vincent Rabaud. *Pose estimation of rigid transparent objects in transparent clutter*. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 162–169. IEEE, 2013. (Cited on page 111.)

[Malisiewicz 2011] Tomasz Malisiewicz, Abhinav Gupta and Alexei A Efros. *Ensemble of exemplar-svms for object detection and beyond*. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 89–96. IEEE, 2011. (Cited on page 44.)

[Mangasarian 1999] Olvi L Mangasarian and David R Musicant. *Successive overrelaxation for support vector machines*. IEEE Transactions on Neural Networks, vol. 10, no. 5, pages 1032–1037, 1999. (Cited on page 36.)

[Massa 2016] Francisco Massa, Bryan C Russell and Mathieu Aubry. *Deep exemplar 2d-3d detection by adapting from real to rendered views*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6024–6033, 2016. (Cited on pages 43 and 44.)

[McKenna 1999] Stephen J McKenna, Yogesh Raja and Shaogang Gong. *Tracking colour objects using adaptive mixture models*. Image and vision computing, vol. 17, no. 3-4, pages 225–231, 1999. (Cited on page 16.)

[Mehta 2017] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu and Christian Theobalt. *Monocular 3d human pose estimation in the wild using improved cnn supervision*. In 3D Vision (3DV), 2017 Fifth International Conference on, 2017. (Cited on page 71.)

[Michels 2005] Jeff Michels, Ashutosh Saxena and Andrew Y Ng. *High speed obstacle avoidance using monocular vision and reinforcement learning*. In Proceedings of the 22nd international conference on Machine learning, pages 593–600. ACM, 2005. (Cited on page 43.)

[Moghaddam 1995] Baback Moghaddam and Alex Pentland. *Probabilistic visual learning for object detection*. In Computer Vision, 1995. Proceedings., Fifth International Conference on, pages 786–793. IEEE, 1995. (Cited on page 16.)

[Muja 2009] Marius Muja and David Lowe. *Flann-fast library for approximate nearest neighbors user manual*. Computer Science Department, University of British Columbia, Vancouver, BC, Canada, 2009. (Cited on page 111.)

[Murphy 1999] Kevin P Murphy, Yair Weiss and Michael I Jordan. *Loopy belief propagation for approximate inference: An empirical study*. In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pages 467–475. Morgan Kaufmann Publishers Inc., 1999. (Cited on page 78.)

[Newcombe 2011] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges and Andrew Fitzgibbon. *KinectFusion: Real-time dense surface mapping and tracking*. In Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on, pages 127–136. IEEE, 2011. (Cited on pages 41 and 104.)

[Newell 2016] Alejandro Newell, Kaiyu Yang and Jia Deng. *Stacked hourglass networks for human pose estimation*. In European Conference on Computer Vision, pages 483–499. Springer, 2016. (Cited on pages 22 and 70.)

[(OpenCV) 2014] Open Source Computer Vision (OpenCV). *LatentSVMDetector*. `https://opencv.org/`, Documentation under `https://docs.opencv.org/2.4.10/modules/objdetect/doc/latent_svm.html`, 2014. (Cited on page 114.)

[Ouyang 2015] Wanli Ouyang, Xiaogang Wang, Xingyu Zeng, Shi Qiu, Ping Luo, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Chen-Change Loy*et al. Deepid-net: Deformable deep convolutional neural networks for object detection*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2403–2412, 2015. (Cited on page 14.)

[Pavlakos 2016] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis and Kostas Daniilidis. *Coarse-to-fine volumetric prediction for single-image 3D human pose*. arXiv preprint arXiv:1611.07828, 2016. (Cited on pages 71, 72, 90 and 91.)

[Pearl 2014] Judea Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, 2014. (Cited on pages 28 and 77.)

[Pepik 2012a] Bojan Pepik, Peter Gehler, Michael Stark and Bernt Schiele. *3d2pm–3d deformable part models*. In European Conference on Computer Vision, pages 356–370. Springer, 2012. (Cited on pages 45 and 46.)

[Pepik 2012b] Bojan Pepik, Michael Stark, Peter Gehler and Bernt Schiele. *Teaching 3d geometry to deformable part models*. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3362–3369. IEEE, 2012. (Cited on page 45.)

[Pepik 2015a] Bojan Pepik, Rodrigo Benenson, Tobias Ritschel and Bernt Schiele. *What is holding back convnets for detection?* In German Conference on Pattern Recognition, pages 517–528. Springer, 2015. (Cited on page 43.)

[Pepik 2015b] Bojan Pepik, Michael Stark, Peter Gehler, Tobias Ritschel and Bernt Schiele. *3d object class detection in the wild*. In Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on, pages 1–10. IEEE, 2015. (Cited on page 45.)

[Pepik 2015c] Bojan Pepik, Michael Stark, Peter Gehler and Bernt Schiele. *Multi-view and 3d deformable part models*. IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 11, pages 2232–2245, 2015. (Cited on pages 45 and 46.)

[Peyré 2006] Gabriel Peyré and Laurent D Cohen. *Geodesic remeshing using front propagation*. International Journal of Computer Vision, vol. 69, no. 1, page 145, 2006. (Cited on page 48.)

[Pilet 2005] Julien Pilet, Vincent Lepetit and Pascal Fua. *Real-time nonrigid surface detection*. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 822–828. IEEE, 2005. (Cited on page 104.)

[Pishchulin 2012] Leonid Pishchulin, Arjun Jain, Mykhaylo Andriluka, Thorsten Thormählen and Bernt Schiele. *Articulated people detection and pose estimation: Reshaping the future*. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3178–3185. IEEE, 2012. (Cited on page 70.)

[Pishchulin 2016] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler and Bernt Schiele. *Deepcut: Joint*

subset partition and labeling for multi person pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4929–4937, 2016. (Cited on pages 70 and 105.)

[Pons-Moll 2011] Gerard Pons-Moll, Andreas Baak, Juergen Gall, Laura Leal-Taixe, Meinard Mueller, Hans-Peter Seidel and Bodo Rosenhahn. *Outdoor human motion capture using inverse kinematics and von mises-fisher sampling.* In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 1243–1250. IEEE, 2011. (Cited on page 71.)

[Prankl 2015] Johann Prankl, Aitor Aldoma, Alexander Svejda and Markus Vincze. *RGB-D object modelling for object recognition and tracking.* In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 96–103. IEEE, 2015. (Cited on page 111.)

[Prasad 2010] Mukta Prasad, Andrew Fitzgibbon, Andrew Zisserman and Luc Van Gool. *Finding nemo: Deformable object class modelling using curve matching.* In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1720–1727. IEEE, 2010. (Cited on pages 44 and 45.)

[Ramakrishna 2012] Varun Ramakrishna, Takeo Kanade and Yaser Sheikh. *Reconstructing 3d human pose from 2d image landmarks.* Computer Vision–ECCV 2012, pages 573–586, 2012. (Cited on page 71.)

[Ramanan 2007] Deva Ramanan. *Learning to parse images of articulated bodies.* In Advances in neural information processing systems, pages 1129–1136, 2007. (Cited on pages 70 and 77.)

[Ranjan 2015] Rajeev Ranjan, Vishal M Patel and Rama Chellappa. *A deep pyramid deformable part model for face detection.* In Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on, pages 1–8. IEEE, 2015. (Cited on page 14.)

[Ren 2015] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. *Faster R-CNN: Towards real-time object detection with region proposal networks.* In Advances in neural information processing systems, pages 91–99, 2015. (Cited on pages 22 and 24.)

[Ren 2017] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang and Jian Sun. *Object detection networks on convolutional feature maps.* IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 7, pages 1476–1481, 2017. (Cited on page 15.)

[Rockafellar 1976] R Tyrrell Rockafellar. *Monotone operators and the proximal point algorithm.* SIAM journal on control and optimization, vol. 14, no. 5, pages 877–898, 1976. (Cited on page 82.)

[Rogez 2016] Grégory Rogez and Cordelia Schmid. *MoCap-guided data augmentation for 3D pose estimation in the wild*. In Advances in Neural Information Processing Systems, pages 3108–3116, 2016. (Cited on page 91.)

[Ronfard 2002] Rémi Ronfard, Cordelia Schmid and Bill Triggs. *Learning to parse pictures of people*. In European Conference on Computer Vision, pages 700–714. Springer, 2002. (Cited on page 77.)

[ROS 2012a] ROS. *ros object_recognition_tod*. `http://wg-perception.github.io/tod/index.html#tod`, 2012. Accessed: 25/10/2017 under BSD license. (Cited on page 110.)

[ROS 2012b] ROS. *ros tabletop_object_detector*. `http://wiki.ros.org/tabletop_object_detector`, 2012. Accessed: 24/10/2017 under BSD license. (Cited on pages 110 and 111.)

[Russakovsky 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. International Journal of Computer Vision (IJCV), vol. 115, no. 3, pages 211–252, 2015. (Cited on pages 25 and 43.)

[Savalle 2014] Pierre-André Savalle, Stavros Tsogkas, George Papandreou and Iasonas Kokkinos. *Deformable part models with cnn features*. In European Conference on Computer Vision, Parts and Attributes Workshop, 2014. (Cited on pages 14, 25 and 26.)

[Schneiderman 2000] Henry Schneiderman and Takeo Kanade. *A statistical method for 3D object detection applied to faces and cars*. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 1, pages 746–751. IEEE, 2000. (Cited on page 16.)

[Seitz 2006] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein and Richard Szeliski. *A comparison and evaluation of multi-view stereo reconstruction algorithms*. In Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 519–528. IEEE, 2006. (Cited on page 41.)

[Sermanet 2013] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus and Yann LeCun. *Overfeat: Integrated recognition, localization and detection using convolutional networks*. arXiv preprint arXiv:1312.6229, 2013. (Cited on page 22.)

[Shalev-Shwartz 2011] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro and Andrew Cotter. *Pegasos: Primal estimated sub-gradient solver for svm*. Mathematical programming, vol. 127, no. 1, pages 3–30, 2011. (Cited on page 36.)

[Sheikh 2005] Yaser Sheikh and Mubarak Shah. *Bayesian modeling of dynamic scenes for object detection.* IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 11, pages 1778–1792, 2005. (Cited on page 16.)

[Shimony 1994] Solomon Eyal Shimony. *Finding MAPs for belief networks is NP-hard.* Artificial Intelligence, vol. 68, no. 2, pages 399–410, 1994. (Cited on page 78.)

[Shneier 1980] Michael Shneier. *Extracting linear features from images using pyramids.* Rapport technique, MARYLAND UNIV COLLEGE PARK COMPUTER VISION LAB, 1980. (Cited on page 15.)

[Sigal 2006] Leonid Sigal and Michael J Black. *Measure locally, reason globally: Occlusion-sensitive articulated pose estimation.* In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 2, pages 2041–2048. IEEE, 2006. (Cited on page 77.)

[Sigal 2010] Leonid Sigal, Alexandru O Balan and Michael J Black. *Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion.* International journal of computer vision, vol. 87, no. 1, pages 4–27, 2010. (Cited on page 71.)

[Simonyan 2014] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014. (Cited on pages 22 and 24.)

[Starck 2003] Jonathan Starck and Adrian Hilton. *Model-based multiple view reconstruction of people.* In null, page 915. IEEE, 2003. (Cited on page 104.)

[Steinwart 2008] Ingo Steinwart and Andreas Christmann. Support vector machines. Springer Science & Business Media, 2008. (Cited on page 16.)

[Stockman 1987] George Stockman. *Object recognition and localization via pose clustering.* Computer vision, graphics, and image processing, vol. 40, no. 3, pages 361–387, 1987. (Cited on page 46.)

[Sturm 2012] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard and Daniel Cremers. *A benchmark for the evaluation of RGB-D SLAM systems.* In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 573–580. IEEE, 2012. (Cited on page 41.)

[Sun 2017] Xiao Sun, Jiaxiang Shang, Shuang Liang and Yichen Wei. *Compositional Human Pose Regression.* arXiv preprint arXiv:1704.00159, 2017. (Cited on page 90.)

[Swain 1991] Michael J Swain and Dana H Ballard. *Color indexing.* International journal of computer vision, vol. 7, no. 1, pages 11–32, 1991. (Cited on page 14.)

[Szeliski 2006] Richard Szeliski. *Image alignment and stitching: A tutorial.* Foundations and Trends® in Computer Graphics and Vision, vol. 2, no. 1, pages 1–104, 2006. (Cited on page 14.)

[Tian 2010] Tai-Peng Tian and Stan Sclaroff. *Fast globally optimal 2d human detection with loopy graph models.* In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 81–88. IEEE, 2010. (Cited on page 77.)

[Tola 2010] Engin Tola, Vincent Lepetit and Pascal Fua. *Daisy: An efficient dense descriptor applied to wide-baseline stereo.* IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 5, pages 815–830, 2010. (Cited on page 14.)

[Tome 2017] Denis Tome, Chris Russell and Lourdes Agapito. *Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image.* arXiv preprint arXiv:1701.00295, 2017. (Cited on pages 71 and 91.)

[Tompson 2014] Jonathan J Tompson, Arjun Jain, Yann LeCun and Christoph Bregler. *Joint training of a convolutional network and a graphical model for human pose estimation.* In Advances in neural information processing systems, pages 1799–1807, 2014. (Cited on pages 22 and 72.)

[Tompson 2015] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun and Christoph Bregler. *Efficient object localization using convolutional networks.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 648–656, 2015. (Cited on page 70.)

[Torresani 2008] Lorenzo Torresani, Aaron Hertzmann and Chris Bregler. *Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors.* IEEE transactions on pattern analysis and machine intelligence, vol. 30, no. 5, pages 878–892, 2008. (Cited on page 43.)

[Toshev 2014] Alexander Toshev and Christian Szegedy. *Deeppose: Human pose estimation via deep neural networks.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1653–1660, 2014. (Cited on pages 22 and 70.)

[Trigeorgis 2016] George Trigeorgis, Patrick Snape, Mihalis A Nicolaou, Epameinondas Antonakos and Stefanos Zafeiriou. *Mnemonic descent method: A recurrent process applied for end-to-end face alignment.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4177–4187, 2016. (Cited on page 44.)

[Tsochantaridis 2004] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims and Yasemin Altun. *Support vector machine learning for interdependent and structured output spaces.* In Proceedings of the twenty-first

international conference on Machine learning, page 104. ACM, 2004. (Cited on pages 36 and 37.)

[Vapnik 1998] Vladimir Naumovich Vapnik and Vlamimir Vapnik. Statistical learning theory, volume 1. Wiley New York, 1998. (Cited on page 36.)

[Viola 2001] Paul Viola and Michael Jones. *Rapid object detection using a boosted cascade of simple features*. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I–I. IEEE, 2001. (Cited on page 14.)

[Vlasic 2007] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik and Jovan Popović. *Practical motion capture in everyday surroundings*. In ACM transactions on graphics (TOG), volume 26, page 35. Acm, 2007. (Cited on page 71.)

[Vondrick 2013] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz and Antonio Torralba. *Hoggles: Visualizing object detection features*. In Proceedings of the IEEE International Conference on Computer Vision, pages 1–8, 2013. (Cited on page 114.)

[Wang 2008] Yang Wang and Greg Mori. *Multiple tree models for occlusion and spatial constraints in human pose estimation*. In European Conference on Computer Vision, pages 710–724. Springer, 2008. (Cited on page 77.)

[Weber 2000] Markus Weber, Max Welling and Pietro Perona. *Towards automatic discovery of object categories*. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 2, pages 101–108. IEEE, 2000. (Cited on page 16.)

[Wei 2016] Shih-En Wei, Varun Ramakrishna, Takeo Kanade and Yaser Sheikh. *Convolutional pose machines*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4724–4732, 2016. (Cited on pages 14 and 70.)

[Weiss 2001] Yair Weiss and William T Freeman. *On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs*. IEEE Transactions on Information Theory, vol. 47, no. 2, pages 736–744, 2001. (Cited on page 28.)

[Whelan 2012] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard and John McDonald. *Kintinuous: Spatially extended kinectfusion*. 2012. (Cited on page 41.)

[Xiang 2014] Yu Xiang, Roozbeh Mottaghi and Silvio Savarese. *Beyond pascal: A benchmark for 3d object detection in the wild*. In Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on, pages 75–82. IEEE, 2014. (Cited on pages 43 and 62.)

[Yan 2007] Pingkun Yan, Saad M Khan and Mubarak Shah. *3d model based object class detection in an arbitrary view.* In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–6. IEEE, 2007. (Cited on page 46.)

[Yang 2011] Yi Yang and Deva Ramanan. *Articulated pose estimation with flexible mixtures-of-parts.* In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1385–1392. IEEE, 2011. (Cited on pages 70 and 77.)

[Yang 2013] Yi Yang and Deva Ramanan. *Articulated human detection with flexible mixtures of parts.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 12, pages 2878–2890, 2013. (Cited on page 70.)

[Yang 2016] Wei Yang, Wanli Ouyang, Hongsheng Li and Xiaogang Wang. *End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3073–3082, 2016. (Cited on pages 72 and 77.)

[Yasin 2016] Hashim Yasin, Umar Iqbal, Bjorn Kruger, Andreas Weber and Juergen Gall. *A dual-source approach for 3D pose estimation from a single image.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4948–4956, 2016. (Cited on pages 71 and 91.)

[Yu 2010] Jie Yu, Dirk Farin, Christof Krüger and Bernt Schiele. *Improving person detection using synthetic training data.* In Image Processing (ICIP), 2010 17th IEEE International Conference on, pages 3477–3480. IEEE, 2010. (Cited on page 43.)

[Yuille 1992] Alan L Yuille, Peter W Hallinan and David S Cohen. *Feature extraction from faces using deformable templates.* International journal of computer vision, vol. 8, no. 2, pages 99–111, 1992. (Cited on page 19.)

[Yuille 2003] Alan L Yuille and Anand Rangarajan. *The concave-convex procedure.* Neural computation, vol. 15, no. 4, pages 915–936, 2003. (Cited on page 39.)

[Zeiler 2014] Matthew D Zeiler and Rob Fergus. *Visualizing and understanding convolutional networks.* In European conference on computer vision, pages 818–833. Springer, 2014. (Cited on pages 2 and 25.)

[Zhang 2004] Tong Zhang. *Solving large scale linear prediction problems using stochastic gradient descent algorithms.* In Proceedings of the twenty-first international conference on Machine learning, page 116. ACM, 2004. (Cited on page 36.)

[Zhou 2015] Xiaowei Zhou, Spyridon Leonardos, Xiaoyan Hu and Kostas Daniilidis. *3D shape estimation from 2D landmarks: A convex relaxation approach.*

In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4447–4455, 2015. (Cited on page 71.)

[Zhou 2016] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis and Kostas Daniilidis. *Sparseness meets deepness: 3D human pose estimation from monocular video.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4966–4975, 2016. (Cited on pages 71, 90, 99, 100 and 104.)

[Zhu 2006] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng and Shai Avidan. *Fast human detection using a cascade of histograms of oriented gradients.* In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 2, pages 1491–1498. IEEE, 2006. (Cited on page 76.)

[Zhu 2015] Menglong Zhu, Xiaowei Zhou and Kostas Daniilidis. *Single image pop-up from discriminatively learned parts.* In Proceedings of the IEEE International Conference on Computer Vision, pages 927–935, 2015. (Cited on page 46.)

[Zitnick 2014] C Lawrence Zitnick and Piotr Dollár. *Edge boxes: Locating object proposals from edges.* In European Conference on Computer Vision, pages 391–405. Springer, 2014. (Cited on page 18.)

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Représentations à base de parties pour la vision 3D de haut niveau

Mots clés: vision par ordinateur, optimisation, intelligence artificielle

Résumé: Dans cette thèse, nous utilisons des modèles de parties déformables (Deformable Part Models – DPMs) pour apprendre à détecter des parties d'objets. Pour une image d'un objet, l'objectif est de déterminer l'emplacement des parties de cet objet dans l'image. Le problème d'optimisation qui en résulte est non-convexe et difficile en raison de son grand espace de recherche.
Notre première contribution consiste à étendre les DPMs à la troisième dimension, grâce à un algorithme par séparation et évaluation (Branch-and-Bound). Nous élaborons un algorithme personnalisé qui est deux fois plus rapide qu'une approche naïve et garantit l'optimalité globale. Nous dérivons pour le modèle 3-dimensionnel une structure 3-dimensionnel. Cependant, nous entrainons un algorithme prenant en compte chaque sous point de vue de l'apparence. Nous démontrons notre approche sur la tache de l'estimation 3-dimensionnel de la posture, en déterminant la posture de l'objet dans une fraction de second.
Notre deuxième contribution nous permet d'effectuer une inférence efficace sur des modèles où les connexions des parties forment un graphe avec des boucles, étendant ainsi des modèles plus riches. Pour cela, nous utilisons l'algorithme des directions alternées (Alternating Direction Method of Multipliers – ADMM) pour découpler le problème et résoudre itérativement un ensemble de sous-problèmes plus faciles. Nous calculons les paramètres du modèle via un Réseaux Neuronal Convolutif pour la determination de la posture 3-dimensionnel. L'inference développée est utilisée comme dernière couche du réseau neural. Cela permet d'obtenir une performance à l'état de l'art pour la tâche d'estimation de pose humaine en 3D.

Title: Part-Based Representations for High-Level 3D Vision

Keywords: computer vision, optimization, artificial intelligence

Abstract: In this work we use Deformable Part Models (DPMs) to learn and detect object parts in 3 dimensions. Given a single RGB image of an object, the objective is to determine the location of the object's parts. The resulting optimization problem is non-convex and challenging due to its large solution space.
Our first contribution consists in extending DPMs into the third dimension through an efficient Branch-and-Bound algorithm. We devise a customized algorithm that is two orders of magnitude faster than a naive approach and guarantees global-optimality. We derive the model's 3-dimensional geometry from one 3-dimensional structure, but train viewpoint-specific part appearance terms based on deep learning features. We demonstrate our approach on the task of 3D object pose estimation, determining the object pose within a fraction of a second.
Our second contribution allows us to perform efficient inference with part-based models where the part connections form a graph with loops, thereby allowing for richer models. For this, we use the Alternating Direction Method of Multipliers (ADMM) to decouple the problem and solve iteratively a set of easier sub-problems. We compute 3-dimensional model parameters in a Convolutional Neural Network for 3D human pose estimation. Then we append the developed inference algorithm as final layer to this neural network. This yields state of the art performance in the 3D human pose estimation task.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France