ABSTRACT

Title of dissertation: DEEP NEURAL NETWORKS AND REGRESSION MODELS FOR OBJECT DETECTION AND POSE ESTIMATION

Kota Hara, Doctor of Philosophy, 2016

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

Estimating the pose, orientation and the location of objects has been a central problem addressed by the computer vision community for decades. In this dissertation, we propose new approaches for these important problems using deep neural networks as well as tree-based regression models.

For the first topic, we look at the human body pose estimation problem and propose a novel regression-based approach. The goal of human body pose estimation is to predict the locations of body joints, given an image of a person. Due to significant variations introduced by pose, clothing and body styles, it is extremely difficult to address this task by a standard application of the regression method. Thus, we address this task by dividing the whole body pose estimation problem into a set of local pose estimation problems by introducing a dependency graph which describes the dependency among different body joints. For each local pose estimation problem, we train a boosted regression tree model and estimate the pose by progressively applying the regression along the paths in a dependency graph starting from the root node.

Our next work is on improving the traditional regression tree method and demonstrate its effectiveness for pose/orientation estimation tasks. The main issues of the traditional regression training are, 1) the node splitting is limited to binary splitting, 2) the form of the splitting function is limited to thresholding on a single dimension of the input vector and 3) the best splitting function is found by exhaustive search. We propose a novel node splitting algorithm for regression tree training which does not have the issues mentioned above. The algorithm proceeds by first applying k-means clustering in the output space, conducting multi-class classification by support vector machine (SVM) and determining the constant estimate at each leaf node. We apply the regression forest that includes our regression tree models to head pose estimation, car orientation estimation and pedestrian orientation estimation tasks and demonstrate its superiority over various standard regression methods.

Next, we turn our attention to the role of pose information for the object detection task. In particular, we focus on the detection of fashion items a person is wearing or carrying. It is clear that the locations of these items are strongly correlated with the pose of the person. To address this task, we first generate a set of candidate bounding boxes by using an object proposal algorithm. For each candidate bounding box, image features are extracted by a deep convolutional neural network pre-trained on a large image dataset and the detection scores are generated by SVMs. We introduce a pose-dependent prior on the geometry of the bounding boxes and combine it with the SVM scores. We demonstrate that the proposed algorithm achieves significant improvement in the detection performance.

Lastly, we address the object detection task by exploring a way to incorporate an

attention mechanism into the detection algorithm. Humans have the capability of allocating multiple fixation points, each of which attends to different locations and scales of the scene. However, such a mechanism is missing in the current state-of-the-art object detection methods. Inspired by the human vision system, we propose a novel deep network architecture that imitates this attention mechanism. For detecting objects in an image, the network adaptively places a sequence of glimpses at different locations in the image. Evidences of the presence of an object and its location are extracted from these glimpses, which are then fused for estimating the object class and bounding box coordinates. Due to the lack of ground truth annotations for the visual attention mechanism, we train our network using a reinforcement learning algorithm. Experiment results on standard object detection benchmarks show that the proposed network consistently outperforms the baseline networks that do not employ the attention mechanism.

DEEP NEURAL NETWORKS AND REGRESSION MODELS FOR OBJECT DETECTION AND POSE ESTIMATION

by

Kota Hara

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Dedication

To my parents.

Acknowledgments

The work presented in this dissertation would not have been possible without the support of several individuals to whom I owe my gratitude.

First, I would like to thank my advisor, Professor Rama Chellappa for giving me an opportunity to work on challenging and exciting projects over the past five years. Throughout my studies, he has been always supporting, encouraging and directing me, especially when I had hard times. It has been such a privilege to work with and learn from such an extraordinary individual and I am truly honored to be among those who call him their mentor.

I would like to thank Dr. Robinson Piramuthu and Dr. Vignesh Jagadeesh who helped and guided me throughout my unforgettable internship at eBay Inc in the summer of 2014, Dr. Ming-Yu Liu, Dr. Oncel Tuzel and Dr. Amir-massoud Farahmand who provided enormous supports during my exciting internship at Mitsubishi Electric Research Labs in 2015-2016. These two internships helped me to expand my knowledge and skills, which will be enormously useful for my future career. I would also like to thank Professor Larry Davis, Professor Min Wu, Professor Uzi Vishkin and Professor Amitabh Varshney for agreeing to serve on my dissertation committee and for sparing their invaluable time reviewing the work presented here.

During my studies, I was fortunate enough to have many brilliant colleagues at the University of Maryland who have enriched my graduate life in many ways. A special mention goes to Dr. Vishal Patel who have been always willing to help me and have been a great role model for me, and Raviteja Vemulapalli who has been the best labmate one could have and I would like to thank his various comments and feedback through many discussions we had for the last five years.

I owe my deepest thanks to my parents who have always stood by me and motivated me through my career. My special gratitude goes to my wife Eriko for her constant encouragement and support, without which the pages of this dissertation would be blank.

Finally, this dissertation was supported by a MURI Grant N00014-10-1-0934 from the Office of Naval Research.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

In this dissertation, we address a variety of computer vision tasks involving object poses, orientations and locations. First, we propose an efficient regression-based algorithm for the task of human body pose estimation. The algorithm consists of a series of regressions, each of which is responsible only for the local pose estimation task. Secondly, we point out several issues in the standard regression tree training algorithm and propose a novel node splitting method for regression tree training based on k-means clustering and SVM. We then apply this method to several object pose estimation tasks. Next, we study the role of human pose for detecting the fashion items. We introduce a pose-dependent prior on the geometry of the object bounding boxes and integrate it with a state-of-the-art object detector trained on our dataset. Finally, in order to incorporate an attention mechanism into an object detection method, we propose a deep recurrent neural network model trained by a reinforcement learning technique. We briefly describe these topics below.

1.1 Human Body Pose Estimation by Regression on a Dependency Graph

We present a hierarchical method for human pose estimation from a single still image. In our approach, a dependency graph representing relationships between reference points such as body joints is constructed and the positions of these reference points are sequentially estimated by the successive application of multidimensional output regressions along the dependency paths, starting from the root node. Each regressor takes image features computed from an image patch centered on the current node's position estimated by the previous regressor and is specialized for estimating its child nodes' positions. The use of the dependency graph allows us to decompose a complex pose estimation problem into a set of local pose estimation problems that are less complex. We design a dependency graph for two commonly used human pose estimation datasets, the Buffy Stickmen dataset and the ETHZ PASCAL Stickmen dataset, and demonstrate that our method achieves accuracy comparable to state-of-the-art results on both datasets with significantly lower computation time. Furthermore, we propose an importance weighted boosted regression trees method for transductive learning settings and demonstrate the resulting improved performance for pose estimation tasks.

1.2 Growing Regression Tree Forests by Classification for Continuous Object Pose Estimation

In this work, we propose a novel node splitting method for regression trees and incorporate it into the random regression forest framework. Unlike traditional binary splitting, where the splitting rule is selected from a predefined set of binary splitting rules via trial-and-error, the proposed node splitting method first finds clusters of the training data which at least locally minimize the empirical loss without considering the input space. Then splitting rules which preserve the found clusters as much as possible, are determined by casting the problem as a classification problem. Consequently, our new node splitting method enjoys more freedom in choosing the splitting rules, resulting in more efficient tree structures. In addition to the algorithm for the ordinary Euclidean target space, we present a variant which can naturally deal with a circular target space by the proper use of circular statistics. In order to deal with challenging, ambiguous image-based pose estimation problems, we also present a voting-based ensemble method using the mean shift algorithm. Furthermore, to address the data imbalance problems present in some of the datasets, we propose a bootstrap sampling method using a sample weighting technique. We apply the proposed random regression forest algorithm to head pose estimation, car direction estimation and pedestrian orientation estimation tasks, and demonstrate its competitive performance.

1.3 Fashion Apparel Detection: the Role of Deep Convolutional Neural Network and Pose-dependent Priors

In this work, we propose and address a new computer vision task, which we call fashion item detection, where the aim is to detect various fashion items a person in the image is wearing or carrying. The types of fashion items we consider in this work include hat, glasses, bag, pants, shoes and so on. The detection of fashion items can be an important first step in various e-commerce applications in fashion industry. Our method is based on a state-of-the-art object detection method which combines object proposal methods with a Deep Convolutional Neural Network. Since the locations of fashion items are in strong correlation with the locations of body joints positions, we propose a hybrid discriminative-generative model to incorporate contextual information from body poses in order to improve the detection performance. Through experiments, we demonstrate that our algorithm outperforms baseline methods by a large margin.

1.4 Attentional Network for Visual Object Detection

We propose augmenting deep neural networks with an attention mechanism for the visual object detection task. It is believed that humans have the capability of analyzing scene contents from multiple fixation points. However, such a mechanism is missing in the current state-of-the-art object detection methods although some efforts have been made for the object classification task. In order to achieve an improved performance, we propose a recurrent neural network to imitate this mechanism. The algorithm adaptively places a sequence of glimpses around a potential object and accumulates the visual evidences from the glimpses to make a final decision, where the glimpse placement is learned using a reinforcement learning algorithm. Experiment results on benchmark datasets show that the proposed algorithm outperforms the baseline method that does not model the attention mechanism.

1.5 Dissertation Organization

The rest of the dissertation is organized as follows. In chapter 2, we present a method for human body pose estimation. In chapter 3, a new node splitting method for regression tree training and its applications to computer vision problems are presented. Then, we discuss in chapter 4 a fashion item detection method utilizing a pose-dependent prior. Chapter 5 presents an object detection method incorporating attention mechanism. Finally, in chapter 6 we conclude this dissertation with a brief summary and directions for future work.

Chapter 2: Human Body Pose Estimation by Regression on a Dependency Graph

Human pose estimation has been a widely studied topic in the computer vision community. Most of the early methods work on silhouettes extracted by background subtraction to reduce the complexity of the problem. However, reliably extracting silhouettes is itself a difficult task in practical settings and requires background images. Recently, the focus of the community has shifted toward pose estimation from a single still image in cluttered backgrounds. Although some of the silhouette-based algorithms can be applied, the task is significantly more difficult, generating new challenges to address.

Most of the existing methods for pose estimation from a single image, including many state-of-the-art methods, are based on a pictorial structure model, which was first proposed in A. Fischler and A. Elschlager [1973] for general computer vision problems and later applied to the pose estimation problem in F. Felzenszwalb and P. Huttenlocher [2000]. The pictorial structure model represents a human body by a combination of body parts with spring-like constrains between those parts to enforce kinematically plausible spatial configurations. The inference is done by first evaluating the likelihood of each body part's locations on the image and then finding the most plausible configuration. If the model forms a tree structure, the globally optimum solution is efficiently found by dynamic programming.

Despite their successes, pictorial structure models have some problems. First, detecting body parts such as limbs, torso and head is challenging in a real-world scenario due to noisy backgrounds, occlusion and variation in appearances and poses. Most of the efforts have been devoted to building reliable body part detectors; however, they tend to be finely tuned to a specific dataset. Second, it is apparent that a simple pictorial structure model does not produce sufficiently good results and thus many efforts have concentrated on extending the basic pictorial structure model to more complex ones, requiring extensive computations.

In this chapter, we propose a novel solution for the human pose estimation problem, which we call Regression on a Dependency Graph (RoDG). RoDG does not rely on detectors for each body part nor requires computationally expensive optimization methods. In RoDG, a dependency graph representing relationships among reference points such as body joints is specified and the positions of these reference points are sequentially estimated by a successive application of multidimensional output regression along the dependency paths, starting from the root node. Each regressor takes image features computed from an image patch centered on the current node's position estimated by the previous regressor and is specialized for estimating its child nodes' positions. The use of the dependency graph allows us to decompose a complex pose estimation problem into a set of local pose estimation problems that are much simpler. In the training phase, those regressors are independently trained using images of people with ground-truth joint locations.

Most regression methods for the human pose estimation task Bissacco et al. [2007], Agarwal and Triggs [2006], Yamada et al. [2012] learn a single regressor mapping an image patch containing an entire human body region to all of the pose parameters. A drawback of this approach is that image patches have to be large enough to cover all possible poses and thus are dominated by a lot of background regions, making regression problems complex. In contrast, the size of the image patches in our approach is designed to contain mostly foreground regions that are sufficient to estimate local poses, reducing the complexity of the mapping problems.

RoDG is simple, versatile and significantly faster than existing approaches, yet achieves accuracy comparable to state-of-the-art on two popular benchmarks, the Buffy Stickmen dataset[1] and the ETHZ PASCAL Stickmen dataset[2]. We also propose an importance weighted variant of boosted regression trees for transductive learning settings and demonstrate its effectiveness for the human pose estimation task.

2.1 Related work

Many existing approaches to human pose estimation from a still image are based on a pictorial structure model. The focus of current research has been in 1) extending the models to a non-tree structures with efficient inference procedures and 2) improving body part detectors. Ren et al.Ren et al. [2005] introduced pair-wise constraints between parts and use Integer Quadratic Programming to find the most probable configuration, however, their part detectors relied on simple line features. Andriluka et al.Andriluka et al. [2011] used discriminatively trained part detectors to detect parts from images with

[1]`http://www.robots.ox.ac.uk/~vgg/data/stickmen/`

[2]`http://groups.inf.ed.ac.uk/calvin/ethz_pascal_stickmen/`

complex backgrounds.

Instead of relying on a single model, Sapp et al.Sapp et al. [2010] proposed a coarse-to-fine cascade of pictorial structure models. In this approach, the coarser models are trained to efficiently prune implausible poses as much as possible while preserving the true poses for the finer level of pictorial structure models that are more accurate but computationally expensive. Sun et al.Sun et al. [2012a] extended the tree models of Sapp et al. [2010] to loopy models and presented an efficient and exact inference algorithm based on branch-and-bound.

Yang and Ramanan Yang and Ramanan [2011] proposed a mixture of templates for each part. They introduced a score term for representing the co-occurrence relations between the mixtures of parts in a scoring function of the pictorial structure model and achieved impressive results. Ukita Ukita [2012] extended Yang and Ramanan [2011] by introducing contour-based features to evaluate connectivities among parts and achieved state-of-the-art results with at most four times the computation time of Yang and Ramanan [2011].

Several approaches to human pose estimation from cluttered images that do not use pictorial structure models W. Lee and Cohen [2004], Hara and Kurokawa [2011], Müller and Arens [2010], Agarwal and Triggs [2006] have been developed. W. Lee and Cohen [2004] applied the MCMC technique to find the MAP estimate of the 3-dimensional pose. Hara and Kurokawa [2011], Müller and Arens [2010] extended the Implicit Shape Model of Leibe et al. [2008] to the human pose estimation task by allowing voting in a pose parameter space.

Transductive learning was first applied to human pose estimation in Yamada et al.

[2012] where the authors proposed importance weighted variants of kernel regression and the twin Gaussian process model to remove the biases in the training set.

2.2 Method - Regression on a Dependency Graph

Let us denote I for an image, $p_i = (x, y)$ for a pixel location of the i-th key point in the image, where $i \in \{1, \ldots, K\}$. The key points may correspond to anatomically defined points of a human body or arbitrarily defined reference points. A dependency graph on the key points is manually designed based on the anatomical structure of the human body. For notational simplicity, we assume p_1 corresponds to the root node. Each adjacent pair of nodes (i, j) in the graph has the following dependency:

$$p_j = s \cdot f_{i,j}(p_i, I, s) + p_i \tag{2.1}$$

where i and j are a parent and child node, respectively, s is the scale parameter and $f_{i,j}$ is a function that outputs a vector. Given a root node position p_1, scale s and an image I, we can determine subsequent $\{p_2, \ldots, p_K\}$ by successively applying Eq.(2.1) along all the graph paths.

Each function $f_{i,j}$ is defined as follows:

$$f_{i,j}(p_i, I, s) = g_{i,j}(h(p_i, I, s)) \tag{2.2}$$

where $g_{i,j}$ is a regressor and $h(p_i, I, s)$ is a predefined function which computes the image features from an image patch centered on p_i at scale s. The size of the image patches is designed to be sufficiently large to contain all possible p_j, however, it should not be larger than necessary.

Each regressor $g_{i,j}$ is independently trained from a set of images with ground-truth annotations of $\{p_1, \dots, p_K\}$ and s. Input features for each regressor are computed by the same h. A target vector for each regressor is the relative location of p_j with respect to p_i normalized by s and can be computed by solving Eqs.(2.1) and (2.2) for $g_{i,j}$:

$$g_{i,j}(h(p_i, I, s)) = (p_j - p_i)/s \tag{2.3}$$

Note that each regressor $g_{i,j}$ is a multidimensional output regressor as the output is a 2-dimensional vector. Furthermore, for a parent node i that has more than two child nodes $\{j_1, \dots, j_L\}$, we define a single multidimensional output regressor that computes an output for each child node at once from the same input:

$$g_i(\cdot) = (g_{i,j_1}(\cdot), \dots, g_{i,j_L}(\cdot)) \in \mathbb{R}^{2L} \tag{2.4}$$

In Fig.2.1 left, we show an instance of the dependency graph designed for the datasets used in the experiments. The non-root nodes of the graph correspond to a set of body joints used to represent a human body pose in the dataset. In Fig.2.1 right, the red box represents a detection window given by an upper body detector. The root node corresponds to the center of the detection window while the other nodes correspond to endpoints of sticks representing a head, torso, upper and lower arms. The scale s is determined by the ratio between the size of the detection window and a predefined canonical window size.

The dependency graph is designed by taking into account the anatomical structure of the human body and also the pose representation adopted by the target datasets. For instance, we make both nodes 7 and 8 depend on node 6 in the graph as they represent body points that are close to each other and thus are contained by the image patch centered on p_6. Similarly, we make nodes 2,3,4,5,6,10 depend on node 1 as their positions do not vary significantly with respect to p_1. Designing an optimum dependency graph for a given task is an interesting topic which will be considered in future.

The details of the training and testing steps on this structure are presented in Section 2.4. Note that RoDG is quite general and applicable to other tasks such as the localization of facial points localization and estimation of hand pose by properly designing the dependency graphs.

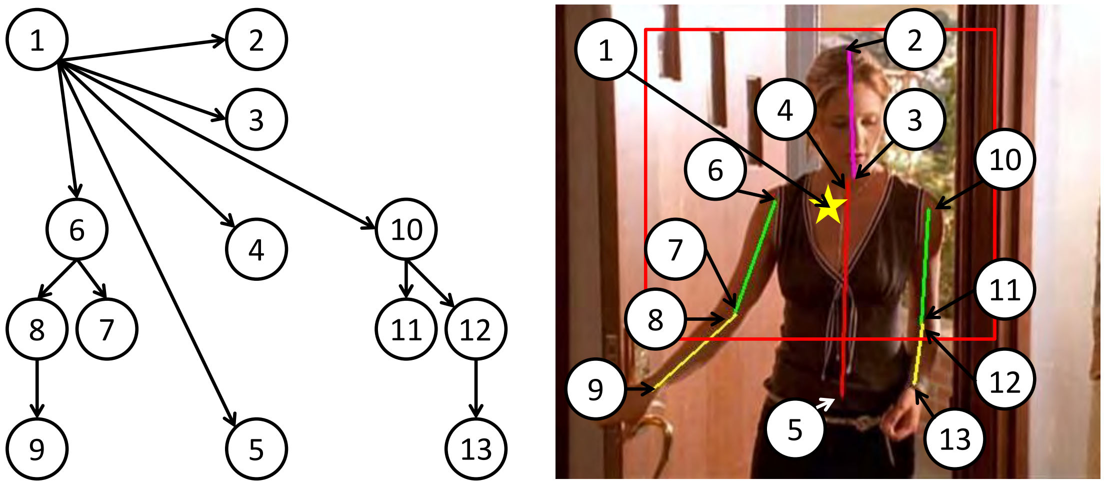


Figure 2.1: Left: Dependency graph, Right: Semantics of the nodes. The red box is a detection window and the yellow star is the center of the detection window.

2.3 Multidimensional Output Regression on Weighted Training Samples

Multidimensional output regression allows us to train a single model that outputs target vectors instead of independently training a single model for each output dimension. We denote a set of training samples by $\{\mathbf{t}_i, \mathbf{x}_i\}_{i=1}^{N}$, where $\mathbf{t}$ is a target vector and $\mathbf{x}$ is an input vector. Furthermore, we denote the weight of the i-th training sample as w_i. All the weights are set to 1 except in the transductive learning setting (Section 2.3.3).

The goal of regression is to learn a function $F^*(\mathbf{x})$ such that the expected value of a certain loss function $\Psi(\mathbf{t}, F(\mathbf{x}))$ is minimized:

$$F^*(\mathbf{x}) = \underset{F(\mathbf{x})}{\operatorname{argmin}} \operatorname{E}[\Psi(\mathbf{t}, F(\mathbf{x})] \tag{2.5}$$

By approximating the above expected loss by empirical loss, we obtain

$$F^*(\mathbf{x}) = \underset{F(\mathbf{x})}{\operatorname{argmin}} \sum_{i=1}^{N} w_i \Psi(\mathbf{t}_i, F(\mathbf{x}_i)). \tag{2.6}$$

2.3.1 Multidimensional Output Regression Tree on Weighted Training Samples

We propose a multidimensional output regression tree on weighted training samples and use it as a building block for the gradient boosting procedure which is presented in Section 2.3.2. The multidimensional output regression tree is a non-linear regression model represented as follows:

$$H(\mathbf{x}; \mathcal{A}, \mathcal{R}) = \sum_{k=1}^{K} \mathbf{a}_k \mathbb{1}(\mathbf{x} \in r_k) \tag{2.7}$$

where $\mathbb{1}$ is an indicator function, $\mathcal{R} = \{r_1, \ldots, r_K\}$ is a set of disjoint partitions of the input space and $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_K\}$ is a set of vectors. Each $\mathbf{a}_k$ is computed as the *weighted* mean of the target vectors of the training samples that fall into r_k.

In the training phase, the regression tree is grown by recursively partitioning the input space, starting from a root node which corresponds to the entire input space. Subsequent partitions are applied to one of the leaves. Throughout the growth of the tree,

$\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_{K'}\}$, where K' is the number of leaves at the time and the weighted sum of squared error for each leaf node k is computed as follows:

$$S_k = \sum_{i \in r_k} w_i ||\mathbf{t}_i - \mathbf{a}_k||_2^2 \tag{2.8}$$

Then the weighted sum of squared error on the entire training data is given by $S = \sum_{k=1}^{K'} S_k$.

At each partitioning stage, the leaf with the largest weighted sum of squared error is selected for partitioning. A binary split rule defined by an index of the input dimension and a threshold is selected among all possible split rules such that the reduction in S is maximized. When computing the weighted means and the sum of squared errors, an efficient incremental algorithm such as West [1979] is used. The recursive partitioning stops when K leaves are generated, where K is a predefined parameter.

2.3.2 Multidimensional Output Boosted Regression Trees on Weighted Training Samples

A gradient boosting machine H. Friedman [2001] is an algorithm to construct a strong regressor from an ensemble of weak regressor. In this chapter, we use the proposed weighted variant of multidimensional output regression tree as a weak regressor. The strong regressor $F(\mathbf{x})$ is expressed as an ensemble of regression trees H:

$$F(\mathbf{x}; P) = \sum_{m=0}^{M} H(\mathbf{x}; \mathcal{A}_m, \mathcal{R}_m) \tag{2.9}$$

where $P = \{\mathcal{A}_m, \mathcal{R}_m\}_{m=0}^{M}$ represents the set of regression trees' parameters.

In the training phase, the gradient boosting algorithm tries to minimize the function in Eq.(2.6) by sequentially adding a new regression tree H at each stage m, where $m = 0$

to M. At each stage except for $m = 0$, a set of the parameters of the tree is determined such that the updated model maximally reduces the loss:

$$(\mathcal{A}_m, \mathcal{R}_m) = \underset{\mathcal{A},\mathcal{R}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i \Psi(\mathbf{t}_i, F_{m-1}(\mathbf{x}_i) + H(\mathbf{x}_i; \mathcal{A}, \mathcal{R})) \tag{2.10}$$

Then the learned regression tree is added to the current model,

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + H(\mathbf{x}; \mathcal{A}_m, \mathcal{R}_m). \tag{2.11}$$

For $m = 0$, $F_0(\mathbf{x})$ is the *weighted* mean target vector of all training samples.

Choosing the squared error loss function $\Psi(\mathbf{t}, F(x)) = ||\mathbf{t} - F(x)||_2^2$ and the weighted regression trees as the weak regressor, we obtain Algorithm 1, where ν is a shrinkage parameter to prevent overfitting. Each tree H is trained using residual $\tilde{\mathbf{t}}$ of each training sample recomputed at each iteration as target vectors. A non-weighted version of the algorithm is also described in Bissacco et al. [2007].

Algorithm 1 Multidimensional Output Boosted Regression Trees on Weighted Training Samples

```
1: $F_0(\mathbf{x}) = \bar{\mathbf{t}}$                                   ▷ weighted mean
2: for $m = 1$ to $M$ do
3:     $\tilde{\mathbf{t}}_i = \mathbf{t}_i - F_{m-1}(\mathbf{x}_i), i = 1, \ldots, N$
4:     $(\mathcal{A}_m, \mathcal{R}_m) = \underset{\mathcal{A},\mathcal{R}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i ||\tilde{\mathbf{t}}_i - H(\mathbf{x}_i; \mathcal{A}, \mathcal{R})||_2^2$
5:     $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu H(\mathbf{x}; \mathcal{A}_m, \mathcal{R}_m)$
6: end for
```

2.3.3 Importance Weighted Boosted Regression Trees

In a transductive learning setting, (unlabeled) testing samples are available during the training phase along with labeled training samples. When the test samples and training samples are drawn from different probability distributions, the regressor trained solely on the training samples is not optimal for the given test samples. One of the possible remedies to this problem is realized by weighting each training sample by an *importance weight* w such that the new distribution formed by the weighted training samples resembles the distribution of testing samples. This is accomplished by setting the importance weight of the i-th training sample as $w_i = p_{te}(\mathbf{x}_i)/p_{tr}(\mathbf{x}_i)$, where p_{te} and p_{tr} are probability density functions of the testing samples and training samples respectively. The proposed weighted variant of the boosted regression trees can work with any method that estimate importance weights. In our work, we adopt RuLSIF Yamada et al. [2011] owing to its impressive performance.

Instead of working on the entire test samples at once, we first cluster the test samples into several clusters by the k-means algorithm and for each cluster we independently estimate the importance weights and train a regressor. This would make the probability density of each cluster simpler and ease the estimation of the importance weights. Furthermore, we transform the test samples to N_{tr} dimensional vectors by computing a kernel matrix $K = (k(\mathbf{x}_i^{te}, \mathbf{x}_j^{tr}))_{i,j}, i = 1, \ldots, N_{te}, j = 1, \ldots, N_{tr}$ where N_{te} and N_{tr} are the number of the testing and training samples respectively. This feature transformation and clustering was found to improve the accuracy.

2.4 Experiments

We tested our algorithm on publicly available datasets for the upper body pose estimation task. The performance is measured by the Percentage of Correctly estimated body Parts (PCP). A comparison with existing works reveals the advantages of our method.

2.4.1 Datasets

We use the Buffy Stickmen dataset and the ETHZ PASCAL Stickmen dataset to evaluate our method. Both datasets have the same representation of poses and provide the same protocol to measure the performance. A body pose is represented by 6 sticks representing the torso, head, upper arms and lower arms (see Fig. 2.1). Each stick is represented by the locations of two endpoints. Both datasets come with detection windows containing upper bodies obtained by an upper body detector. The performance is measured only on images with detection windows, allowing the separation of the human detection task from the pose estimation task. As two endpoints of each stick are annotated without consistent ordering, we manually swap two endpoints if necessary.

The Buffy Stickmen dataset has 748 images taken from the TV show *Buffy the Vampire Slayer* and it is very challenging due to highly cluttered backgrounds. However, the same subjects with same clothing occasionally appear in both training and testing sets which makes the task easier. Among 748 images, 276 images are specified as test data while 472 images are used for training. In the first release of the dataset, 85.1% of the images in the test set come with detection windows while 95.3% come with detection windows obtained by an improved detector in the latest release.

The PASCAL Stickmen dataset contains images taken from the PASCAL VOC 2008 trainval release. Unlike the Buffy Stickmen dataset, it consists mainly of 549 amateur photographs with unconstrained illumination, severe occlusion and low image quality making this dataset more challenging than the Buffy dataset. In the first release, 65.6% of the images come with detection windows while 75.1% in the latest release with the improved detector. Note that the PASCAL dataset is used only for testing.

The performance of pose estimation algorithms is measured using PCP. Each body part is represented as a stick and its estimate is considered correct if its endpoints lie within $100t\%$ of the length of the ground-truth stick from their ground-truth locations. We denote PCP with $t = 0.5$ by $\mathrm{PCP}_{0.5}$.

Both datasets come with a tool to compute the PCP, however, it was recently pointed out in Pishchulin et al. [2012] that the tool does not exactly compute the above defined PCP, leading to erroneously higher PCP. As most of the existing works report PCP on the detection windows in the first releases of the dataset using this tool, we also report PCP using the same tool. To facilitate future comparison, we also report the correct PCP computed by a fixed version of the tool[3] on the updated detection windows provided in recent releases. To eliminate any confusion, we precisely define a condition that an estimated part (i.e. stick) has to satisfy to be considered as correctly localized:

$$\begin{gathered}(||E_1 - G_1||_2 \leq t \cdot L \wedge ||E_2 - G_2||_2 \leq t \cdot L) \\ \vee \\ (||E_1 - G_2||_2 \leq t \cdot L \wedge ||E_2 - G_1||_2 \leq t \cdot L)\end{gathered} \tag{2.12}$$

[3]The fixed tool is available on the author's website.

where (E_1, E_2) and (G_1, G_2) are the locations of two endpoints of the estimated and ground-truth stick, respectively, and $L = ||G_1 - G_2||_2$.

2.4.2 Implementation Details

In order to obtain the ground-truth of the root node, a set of detection windows containing the annotated upper bodies in the training images is first obtained by running the same upper body detector used to obtain the detection windows for the test set. Each image has exactly one annotated human. Detection windows are obtained for 345 out of 472 training images in the Buffy training set[4]. The scale s for each sample is determined by the width of the detection window divided by 64. The ground-truth for the other nodes is included in the dataset.

The image patches from which $h(p_i, I, s = 1)$ computes image features is set to 64×64 pixel rectangular region whose center is located at p_i. From each patch, we compute multiscale HOG Dalal and Triggs [2005] with cell size 8, 16, 32 and 2×2 cell blocks. The orientation histogram for each cell is computed with unsigned gradients with 9 orientation bins. The dimensionality of the resultant HOG feature is 2124. For an arbitrary s, the image patch size is scaled by s while keeping the center location unchanged.

In its original form, the dependency graph (Fig.2.1) requires 5 regressors, namely, $g_{1,\{2,3,4,5,6,10\}}$, $g_{6,\{7,8\}}$, $g_{10,\{11,12\}}$, $g_{8,9}$ and $g_{12,13}$. In order to exploit the symmetric structure of the human body, we train a shared regressor for $g_{6,\{7,8\}}$ and $g_{10,\{11,12\}}$ by horizontally flipping the training samples for the key points on the right side of the body. In testing time, the same regressor is used for both sides but for the right side both the input patch

[4]We thank Marcin Eichner for providing the results.

and output vector need to be horizontally flipped. We do the same for $g_{8,9}$ and $g_{12,13}$. This procedure practically doubles the number of the training samples. For $g_{1,\{2,3,4,5,6,10\}}$, we also double the number of the training samples by appropriately mirroring each training sample.

For boosted regression trees, the number of leaves in the regression trees K is set to 5 and the shrinkage parameter ν is set to 0.1 following the suggestion in Hastie et al.. Through cross-validation on the training set, it is observed that the error keeps decreasing as the number of trees increases. Thus, we empirically set the number of trees M to 2000 for $g_{1,\{2,3,4,5,6,10\}}$ and 1000 for the rest. The regressors are trained on the Buffy training set and the same regressors are used for testing on both Buffy testing set and PASCAL dataset.

2.4.3 Results

As our RoDG works with any multidimensional output regression methods, we also test RoDG with Kernel Partial Least Squares (KPLS) Rosipal and Trejo [2001], Partial Least Squares (PLS) de Jong [1993], Lasso Efron et al. [2004] and Multivariate RVM (MRVM) Thayananthan et al. [2006]. The parameters of these regression methods are determined by 5-fold cross validation.

In Table 2.1, we show the results on the Buffy dataset evaluated with the PCP tool provided in the dataset and the detection windows in the initial release of the dataset, while in Table 2.2, we show the results with the fixed PCP tool and the updated detection windows in the latest release.

As can be seen from Table 2.1, the RoDG-Boost achieves the second best total $\mathrm{PCP}_{0.5}$ next to Ukita [2012] with significantly lower computation time (Table 2.5). Note that unlike some of the previous works, RoDG does not require external training data nor exploit color information. For reference, we also compare our methods with Ukita [2012][5] using a stricter criteria (total $\mathrm{PCP}_{0.2}$) and found out that RoDG-Boost outperforms Ukita [2012] with a large margin (RoDG-Boost:63.0, Ukita [2012]:58.2). This result indicates that the ranking of performance varies depending on the PCP threshold, thus comparisons should also be made by PCP-curves obtained by varying the PCP threshold. Table 2.2 shows that RoDG-Boost and RoDG-KPLS outperform existing methods by a large margin.

The PCP values on the first setting are higher than those on the second setting due to the flaw in the original PCP tool, mentioned in 2.4.1. The correct PCP scores reveal that there is still much room for improvement, especially for lower arms. In Fig.2.2(a), we plot the PCP curves on the Buffy testing set with the second setting. RoDG-Boost consistently outperforms RoDG-KPLS when PCP threshold is less than 0.47 and both methods significantly outperform the state-of-the-art. We encourage future comparisons on this new setting with PCP curves.

In Tables 2.3 and 2.4, we show the results on the PASCAL dataset under the two settings. We achieve state-of-the-art results on both settings. The PCPs on the PASCAL are much lower than that on Buffy. We believe that the reasons are 1) that the PASCAL dataset is more difficult due to more complex poses, more challenging occlusions and blur, 2) the similarity between the test and training sets in the Buffy dataset favors PCP

[5]We thank Norimichi Ukita for providing the results.

Table 2.1: $\mathrm{PCP}_{0.5}$ on Buffy with the original PCP tool and detection windows

	total	torso	u.arms	l.arms	head
RoDG-Boost	89.8	99.6	96.8	73.0	99.6
RoDG-KPLS	88.9	100	97.0	69.8	99.6
RoDG-MRVM	87.5	99.6	97.2	67.0	97.0
RoDG-LASSO	86.7	100	96.7	63.6	99.6
RoDG-PLS	87.2	100	97.5	65.3	97.9
Ukita Ukita [2012]	90.3	100	97.5	73.9	98.9
Yang Yang and Ramanan [2011]	89.1	100	96.6	70.9	99.6
Zuffi Zuffi et al. [2012]	85.6	99.6	94.7	62.8	99.2
Sun Sun et al. [2012a]	85.7	99.6	93.8	63.9	99.2
Sapp Sapp et al. [2010]	85.5	100	95.3	63.0	96.2
Andriluka Andriluka et al. [2011]	83.1	97.5	92.7	59.6	95.7

on the Buffy dataset. In Fig.2.2(b), we plot the PCP curves on the PASCAL dataset with the second setting. RoDG-KPLS consistently outperforms RoDG-Boost, however, RoDG-KPLS is much more computationally expensive due to KPLS execution (Table 2.5).

Table 2.5 presents approximate computation times of each method to process one image. Note that the computation time of previous methods are taken from their original papers or websites and thus are not obtained by running on the same computer, however, they give a rough idea on the computational requirements of each method. All RoDGs

Table 2.2: $\mathrm{PCP}_{0.5}$ on Buffy with the updated PCP tool and detection windows

	total	torso	u.arms	l.arms	head
RoDG-Boost	81.1	98.5	92.8	51.5	99.2
RoDG-KPLS	79.6	98.9	92.0	47.7	99.2
RoDG-MRVM	76.9	98.9	91.8	40.5	97.7
RoDG-LASSO	74.6	98.5	89.7	35.4	98.9
RoDG-PLS	74.2	99.6	90.5	33.5	97.7
Eichner Eichner et al. [2012]	76.7	99.6	81.9	50.0	96.6

are run on Xeon 3.6GHz CPU machine. All RoDGs run significantly faster than all the previous methods.

Representative results of RoDG-Boost on Buffy and PASCAL are shown in Fig.2.3 and Fig.2.4, respectively.

Transductive learning results

We evaluate the performance of RoDG with our importance weighted boosted regression trees in transductive settings. As the fixed PCP tool is more adequate to compare the performance of the methods, we conduct experiments only using the second setting. For RuLSIF, we use the same parameter settings employed in Yamada et al. [2011]. We use a Gaussian kernel with $\sigma = 10$ for feature transformation and set the number of clusters to 10 and 20 for Buffy and PASCAL, respectively. The parameters of the gradient boosting are kept the same.

Table 2.3: $\mathrm{PCP}_{0.5}$ on PASCAL with the original PCP tool and detection windows

	total	torso	u.arms	l.arms	head
RoDG-Boost	79.2	100	87.8	50.4	98.9
RoDG-KPLS	79.1	99.7	87.5	51.0	97.8
RoDG-MRVM	77.5	99.7	86.0	47.5	98.1
RoDG-LASSO	76.4	100	86.7	44.4	96.1
RoDG-PLS	76.3	99.7	87.0	43.8	96.9
Sun Sun et al. [2012a]	78.8	99.7	81.4	55.4	99.4
Sapp Sapp et al. [2010]	77.2	100	87.1	49.4	90.0
Andriluka Andriluka et al. [2011]	71.8	96.4	77.8	47.0	85.0

Tables 2.6 and 2.7 show the results on the Buffy and PASCAL dataset, respectively. The first row presents the results of non-transductive settings, the second row, the results of transductive settings without clustering and the third row presents the results with clustering. On the Buffy dataset, the PCP clearly improves while on the PASCAL dataset, RuLSIF degrades the performance but RuLSIF-cluster recovers the loss.

2.5 Conclusion

In this chapter, we presented an algorithm for human pose estimation from a still image based on successive application of multidimensional output regressions on a dependency graph. The pose estimation problem was divided into a set of local pose estimation problems and solved sequentially from the root node of the graph. The method is a com-

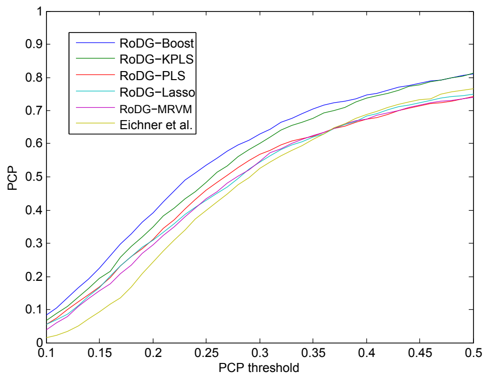
Table 2.4: $\mathrm{PCP}_{0.5}$ on PASCAL with the updated PCP tool and detection windows

	total	torso	u.arms	l.arms	head
RoDG-Boost	63.3	91.5	75.1	27.8	82.3
RoDG-KPLS	62.9	90.3	74.5	28.9	80.3
RoDG-MRVM	59.6	87.1	71.5	26.1	75.5
RoDG-LASSO	57.4	89.6	69.4	22.1	71.6
RoDG-PLS	56.5	88.8	72.1	18.0	69.9
Eichner Eichner et al. [2012]	55.7	96.6	60.6	27.3	61.9

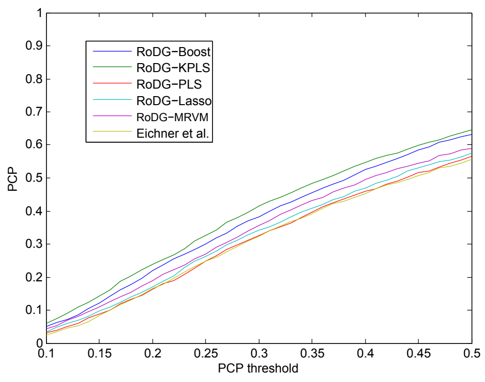
petitive alternative to pictorial structure-based methods for human pose estimation. On the two popular benchmarks, Buffy Stickmen and ETHZ PASCAL Stickmen, our method achieves comparable accuracy to state-of-the-art result with significantly lower computation time. Furthermore, we proposed boosted regression trees for importance weighted samples and applied it to transductive learning settings for human pose estimation.

2.6 Acknowledgments

The work presented in this chapter was conducted under the support by a MURI grant from the US Office of Naval Research under N00014-10-1-0934.



(a) Buffy stickmen



(b) PASCAL Stickmen

Figure 2.2: PCP curves with the second setting (best viewed in color)

Table 2.5: Computation time per image. Left: our methods, Right: existing methods

method	time	method	time
RoDG-Boost	23 msec.	Ukita Ukita [2012]	4 sec.
RoDG-KPLS	193 msec.	Yang Yang and Ramanan [2011]	1 sec.
RoDG-PLS	13 msec.	Zuffi Zuffi et al. [2012]	a few min.
RoDG-LASSO	13 msec.	Sun Sun et al. [2012a]	300 sec.
RoDG-MRVM	15 msec.	Sapp Sapp et al. [2010]	300 sec.
		Andriluka Andriluka et al. [2011]	50 sec.
		Eichner Eichner et al. [2012]	6.6 sec.

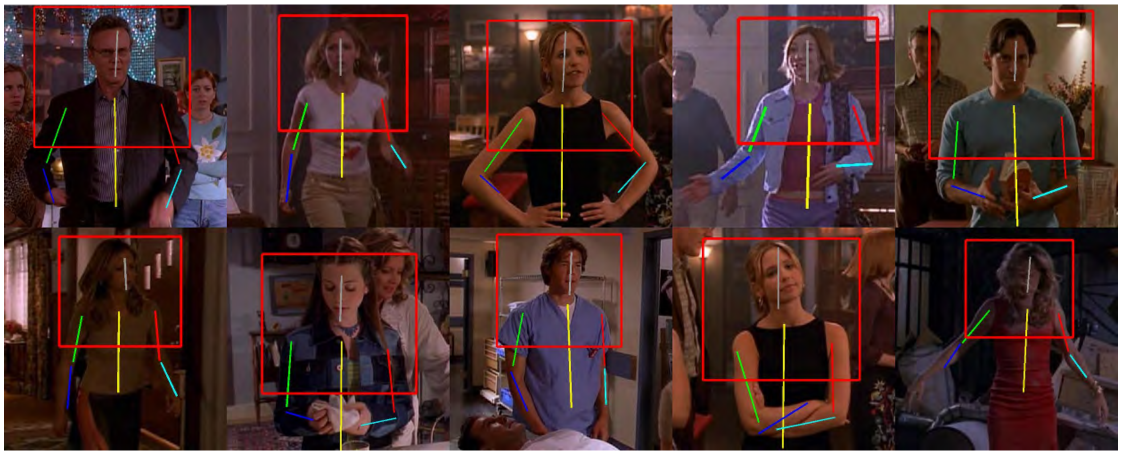


Figure 2.3: Representative results of RoDG-Boost on Buffy Stickmen dataset.

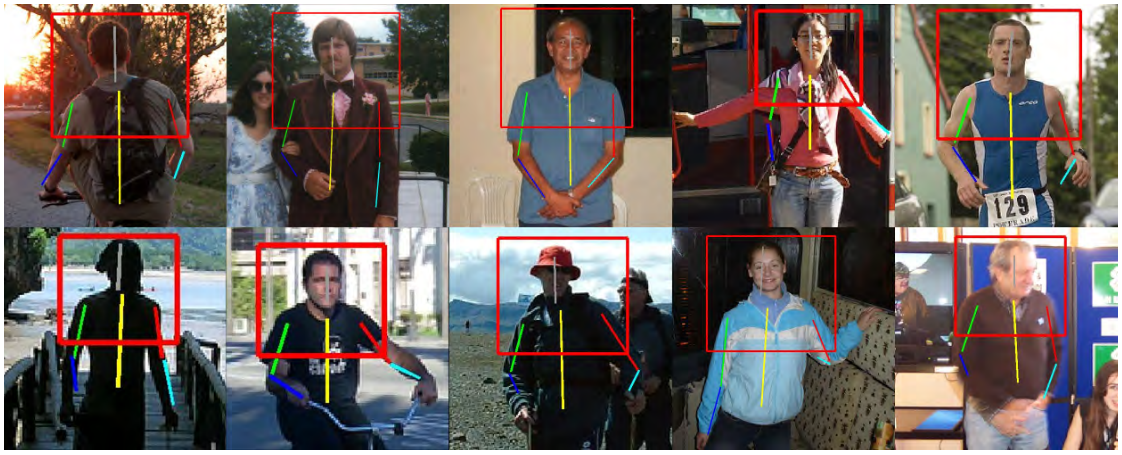


Figure 2.4: Representative results of RoDG-Boost on PASCAL Stickmen dataset. The last two columns show failure cases.

Table 2.6: $\mathrm{PCP}_{0.5}$ of importance weighted boosted regression trees on Buffy

	total	torso	u.arms	l.arms	head
Base	81.1	98.5	92.8	51.5	99.2
RuLSIF	81.6	98.9	92.6	53.2	99.2
RuLSIF-clstrs	82.5	98.9	93.5	54.9	99.2

Table 2.7: $\mathrm{PCP}_{0.5}$ of importance weighted boosted regression trees on PASCAL

	total	torso	u.arms	l.arms	head
Base	63.3	91.5	75.1	27.8	82.3
RuLSIF	63.0	90.3	75.2	28.8	79.9
RuLSIF-clstrs	63.4	90.3	75.5	27.9	83.0

Chapter 3: Growing Regression Tree Forests by Classification for Continuous Object Pose Estimation

Regression has been successfully applied to various computer vision tasks such as head pose estimation [Haj et al., 2012, Fenzi et al., 2013], object direction estimation [Fenzi et al., 2013, Torki and Elgammal, 2011], human body pose estimation [Bissacco et al., 2007, Sun et al., 2012b, Hara and Chellappa, 2013] and facial point localization [Dantone et al., 2012, Cao et al., 2012], which require continuous outputs. In regression, a mapping from an input space to a target space is learned from the training data. The learned mapping function is used to predict the target values for new data. In computer vision, the input space is typically the high-dimensional image feature space and the target space is a space which represents some high level concepts present in the given image. Due to the complex input-target relationship, non-linear regression methods are usually employed for computer vision tasks.

Among several non-linear regression methods, random regression forests [Breiman, 2001] have been shown to be effective for various computer vision problems [Sun et al., 2012b, Criminisi et al., 2010, Dantone et al., 2012, A. Criminisi, 2013]. The random regression forest is an ensemble learning method which combines several regression trees [Breiman et al., 1984] into a strong regressor. The regression trees define recursive partitioning of the input space and each leaf node contains a model for the predictor. In the training stage, the trees are grown in order to reduce the empirical loss over the training data. In the random regression forest, each regression tree is independently trained using a random subset of training data (bootstrap samples) and prediction is done by finding the average/mode of outputs from all the trees.

In computer vision, it is often the case that a target space is multidimensional. A common approach is to independently train a regressor for each of the target dimensions. However, this approach is cumbersome if the dimensionality of the target space is high. Also, the training algorithms do not take into account possibly existing correlations among the different target dimensions. Multi-dimensional target regression allows us to train a single model which can output vector values. During training, a single empirical loss defined over all the target dimensions is minimized. With regression trees, the extension from scalar outputs to vector outputs is trivially achieved and thus the same is true with the random regression forest.

As a node splitting algorithm, binary splitting is commonly employed for regression trees; however, it has limitations regarding how it partitions the input space. The biggest limitation of the standard binary splitting is that a splitting rule at each node is selected by trial-and-error from a predefined set of splitting rules. To manage the search space, simple thresholding operations on a single dimension of the input are typically chosen. Due to these limitations, the resulting trees are not necessarily efficient in reducing the empirical loss.

3.0.1 K-clusters Regression Forest

To overcome the above drawbacks of the standard binary splitting scheme, we propose a novel node splitting method and incorporate it into the regression forest framework. In our node splitting method, clusters of the training data which at least locally minimize the empirical loss are first found without being restricted to a predefined set of splitting rules. Then splitting rules which preserve the found clusters as much as possible, are determined by casting the problem as a classification problem. As a by-product, our procedure allows each node in the tree to have more than two child nodes, adding one more level of flexibility to the model. We also propose a way to adaptively determine the number of child nodes at each splitting using the Bayesian Information Criterion (BIC) [Kashyap, 1977, Schwarz, 1978]. Thus, the number of leaf nodes of each regression tree is adjusted based on the complexity of the distribution of the data. Unlike the standard binary splitting method, our splitting procedure enjoys more freedom in choosing the partitioning rules, resulting in more efficient regression tree structures. In addition to the method for the Euclidean target space, we present a variant which can naturally deal with a circular target space by the proper use of circular statistics.

We refer to regression forests (RF) employing our node splitting algorithm as K-clusters Regression Forest (KRF) and those employing the adaptive determination of the number of child nodes as Adaptive KRF (AKRF).

3.0.2 Voting-based ensemble

Some of the image-based continuous prediction tasks are challenging as similar images can have completely different target values. For instance, in car direction estimation and pedestrian orientation estimation tasks, appearances of some samples are very similar to their 180° flipped versions, making the prediction difficult. On those challenging samples, predictions from multiple trees in the forest tend to form multiple peaks. Thus, the final prediction based on the mean, as in standard regression forest ensemble, results in inaccurate predictions.

To alleviate this problem, we propose a new voting-based ensemble method. In the prediction stage, we allow each training sample in leaf nodes to cast a probabilistic vote in the target space. We then find the highest mode using the mean shift algorithm [Fukunaga and Hostetler, 1975, Cheng, 1995, Comaniciu and Meer, 2002]). By choosing the highest mode, only trees with the largest agreement contribute to the final prediction and those with less agreement are ignored, making the prediction more reliable. For the circular target space, we model each vote as a weighted von Mises distribution and apply the mean shift algorithm derived for the circular space.

3.0.3 Bootstrap sampling for data imbalanceness problem

Another challenge present in some pose estimation tasks is a discrepancy between target variable's distributions of training data and test data. The discrepancy between them can lead to suboptimal performance for any supervised learning method. A particular case we consider in this work is when the target variable distribution of the test data is likely to be uniform but that of the training data is highly imbalanced. For instance, in an orientation estimation problem where object poses range from $0°$ to $360°$, it is natural to assume that each orientation is equally likely: however, if the training data distribution is highly imbalanced, a model trained on this training data would not perform well in the operation stage. To address this issue, we propose to weigh each training data point such that the target variable distribution computed from the weighted training data is uniform. Based on those weights, we then select bootstrap samples for the regression forest training, i.e., samples with larger weights are more likely to be selected. We compute the weights as the reciprocal of the probability density obtained by the kernel density estimation. The likelihood cross-validation is used to determine the parameters of the kernel function, thus, no additional parameters are introduced in the method.

3.0.4 Object pose estimation tasks

In this work, we demonstrate the effectiveness of the proposed approach on three different object pose estimation tasks. The first task is the head pose estimation task which has been a standard computer vision task used to show the effectiveness of various regression methods. In typical head pose estimation testbeds, head poses are represented by one to three dimensional vectors in the Euclidean space. Thus, it is a suitable application to test our methods for the Euclidean target space. Among many existing datasets, we employ Pointing'04 dataset [Gourier et al., 2004] due to its popularity.

The second task is a car direction estimation task which has gained more and more attention due to its practical importance. In this task, car directions are represented by the 1D continuous circular space, making this task suitable for our methods for a circular target space. For this task, we employ the EPFL Multi-view Car Dataset [Ozuysal et al., 2009].

In addition to the above two tasks, we evaluate our methods on a continuous pedestrian orientation estimation task which we introduce to the community. A body orientation of a pedestrian can provide valuable cues for many applications. For 3D pose estimation tasks, accurate orientation estimates significantly reduce the ambiguity of the poses. From a person's orientation, we can infer a potential moving direction which may help to improve tracking accuracy. Person re-identification benefits from the orientation information by modeling color distribution in the orientation space. Interactions between humans and crowd behaviors can be more precisely recognized if their orientations are known. A person's attention can be inferred by his/her body orientation.

Traditionally, body orientation estimation has been addressed as a multi-class classification problem by representing orientations by four or eight representative discrete orientations. Although this is partially justified as obtaining ground truth of continuous orientations is difficult, such a coarse representation may not be sufficient for subsequent applications. Moreover, since the body orientation is continuous by nature, artificial discretization of orientation may result in a suboptimal performance. Therefore, we collected continuous annotations of the body orientations using Amazon Mechanical Turk for an existing orientation estimation dataset which has only discrete annotations (TUD Multiview Pedestrians Dataset [Andriluka et al., 2010]). The user interface used for the annotation is shown in Fig. 3.1). Visualization of the annotation (Fig. 3.9) reveals that the obtained continuous annotations for body orientations capture the smooth transitions of body orientations. Due to various pose and appearance variations and the size of the dataset, this new testbed is much more challenging and realistic than Pointing'04 and EPFL Multi-view Car Dataset and can be used to evaluate the effectiveness of various regression methods. These newly collected annotations will be made publicly available in order to promote more research in this field.

One may argue that the orientation estimation is just a simpler version of popular body pose estimation problems. Although it is true for 3D pose estimation where the aim is to locate body joints in a 3D space, most of the recent pose estimation works focus on localizing body joints on 2D images and in general it is difficult to infer the body orientation from 2D joint locations. Thus, we believe that orientation estimation and pose estimation are complementary to each other and the body orientation estimation task from 2D still images deserve special attention.

To the best of our knowledge, our work is the first regression-based method applied for continuous body orientation estimation from still images. We believe that the introduction of the continuous pedestrian orientation estimation task will facilitate further research in the field of regression for computer vision.

3.0.5 Summary of the results

Through experiments, we demonstrate that the proposed methods, KRF and AKRF, achieve competitive results. Also, they significantly outperform other general regression methods including regression forests with the standard binary splitting. We observe that the proposed two extensions, the sample weighting technique and the voting-based en-

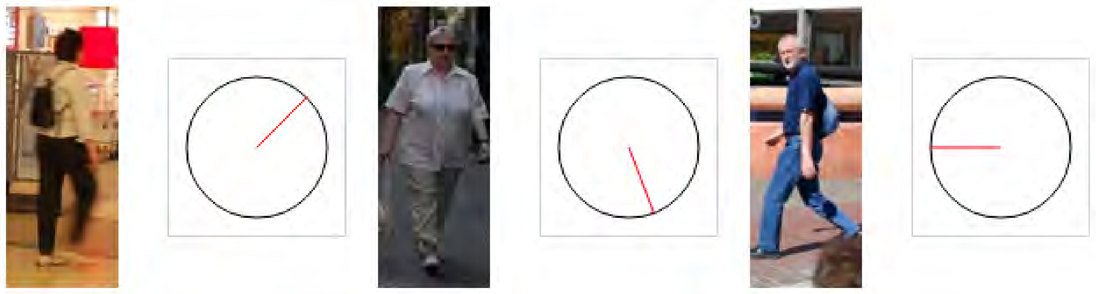


Figure 3.1: User interface for the continuous orientation annotation. Each annotator is requested to specify the body orientation of pedestrians by moving a line segment in a circle.

semble method, further improve the performance for the car direction estimation task (12.5% reduction in an error) and the pedestrian orientation estimation task (8.7% improvement in accuracy) compared to AKRF.

Throughout the chapter, we suffixes W and V to represent methods that use sample weighting and voting-based ensemble, respectively. For instance, AKRF with voting-based ensemble is referred to as AKRF-V and AKRF with both voting-based ensemble and sample weighting is referred to as AKRF-VW.

3.0.6 Organization

In Sec. 3.1, we review related works. In Sec. 3.2, we describe the details of the proposed methods. Sec. 3.3 reports experimental results and Sec. 3.4 concludes the chapter.

3.1 Related work

3.1.1 Regression

Several regression problems such as head pose estimation and body orientation estimation have been addressed by classification methods by assigning a different pseudo-class label to each of roughly discretized target value (e.g., Yan et al. [2013], Huang et al. [2010], Orozco et al. [2009], Baltieri et al. [2012], Ozuysal et al. [2009]). Increasing the number of pseudo-classes allows more precise prediction, however, the classification problem becomes more difficult. This becomes more problematic as the dimensionality of target space increases. In general, discretization is conducted experimentally to balance the desired classification accuracy and precision.

Weiss and Indurkhya [1995], Torgo and Gama [1996] apply k-means clustering to the target space to automatically discretize the target space and assign pseudo-classes. They then solve the classification problem by rule induction algorithms for classification. Though somewhat more sophisticated, these approaches still suffer from problems due to discretization. Our method differs from approaches discussed above in that in these approaches, pseudo-classes are fixed once determined either by a human or clustering algorithms while in our approach, pseudo-classes are *adaptively* redetermined at each node splitting of regression tree training. Furthermore, instead of trying to find possibly complex decision boundaries at once, our method *recursively* partitions the input space such that training samples in each partition have similar target values. Thus, nodes at higher levels of the tree are responsible only for coarse partitioning while those at lower levels focus on finer partitioning, making further partitioning easier. Since each leaf node has a constant estimate for the corresponding partition, the number of possible outputs is equal to the number of leaf nodes in the tree, making detailed prediction possible. When combined with the regression forest framework, the number of possible outputs is further increased.

Similar to our method, Dobra and Gehrke [2002] converted node splitting tasks into local classification tasks by applying the EM algorithm to the joint input-output space. Since clustering is applied to the joint space, this method is not suitable for tasks with high dimensional input space. In fact these experiments are limited to tasks with upto 20 dimensional input space.

The work most similar to our method was proposed by Chou [1991] who applied k-means like algorithm to the target space to find a locally optimal set of partitions for regression tree learning. However, this method is limited to the case where the input is a categorical variable. Although we limit ourselves to continuous inputs, our formulation is more general and can be applied to any type of inputs by choosing appropriate classification methods. Furthermore, incorporating such regression trees into a regression forest framework has not been explored.

3.1.2 Decision trees with multiway splitting

Many multiway splitting methods have been proposed in the literature for classification purpose. Fayyad and Irani [1993] proposed a multiway splitting based on a single input dimension where the number of child nodes is determined by Minimum Description Length (MDL). Berzal et al. [2004] designed a hierarchy of intervals on each input dimension by a hierarchical clustering method which also takes into account the class distributions and selects a set of intervals which minimizes an impurity measure. Loh and Vanichsetakul [1988] used linear discriminant analysis as a multiway splitting function which can naturally exploit all the input dimensions at once and does not rely on exhaustive search for the best splitting function.

For regression, a variant of regression trees called regression ferns realize multiway splitting. Dollár et al. [2010] proposed random regression ferns which partition the input space into 2^S regions based on the results of randomly selected S binary splitting functions. In the training phase, multiple regression ferns are evaluated and the one which has the lowest error is selected. Cao et al. [2012] employed a fern model in the boosted regression framework. Instead of randomly generating binary splitting functions, they selected a set of feature dimensions based on correlations between features and the targets.

3.1.3 Sample weighting for data imbalanceness problem

The issue of imbalanced data has been a major research topic for many years, however, most existing works focus on classification tasks. In Chen et al. [2004], two approaches for addressing the imbalanced training data for classification are discussed for random forests. The first approach is to incorporate sample weights in a cost function to be minimized. The second approach is to use a sampling technique to artificially make the training data balanced by either over-sample minority classes or down-sample majority classes. The other approaches which fall into the first class of approaches are Domingos

[1999], Pazzani et al. [1994] and those falling into the second one are Kubat et al. [1997], Chawla et al. [2002], Drummond and Holte [2003], although they are not designed for random forests. For regression tasks, Torgo et al. [2013] proposed a sampling-based method by extending Chawla et al. [2002], however, the proposed algorithm is not specifically designed for random forests.

3.1.4 Mean shift for a circular space

Several mean shift algorithms for a circular space have been proposed in Chang-Chien et al. [2012], Kobayashi and Otsu [2010], Kafai et al. [2010], Wu and Yang [2007]. In Kobayashi and Otsu [2010], the mean shift algorithm for a unit hypersphere is proposed. Chang-Chien et al. [2012] proposed a mean shift-based clustering algorithm for circular data by extending the algorithm originally developed for Euclidean space [Wu and Yang, 2007]. Kafai et al. [2010] introduced a directional mean shift algorithm based on the shortest path between two points on the directional space and applied it to 3D medical structure topology classification.

3.1.5 Applications

3.1.5.1 Head pose estimation

Regression has been widely applied for head pose estimation tasks. Haj et al. [2012] used kernel partial least squares regression to learn a mapping from HOG features to head poses. Fenzi et al. [2013] learned a set of local feature generative model using RBF networks and estimated poses using MAP inference. Fanelli et al. [2011] applied random regression forest to a head pose estimation task from depth images. Ho and Chellappa [2012] used a multi-class SVM to obtain a coarse estimate of the head pose and then used SVR to refine the estimate. Bailly et al. [2009] proposed a feature selection method based on the boosting technique and combined it with a generalized regression neural network.

Hough Forests [Gall and Lempitsky, 2009] and its extension [Girshick et al., 2011] can be considered as a regression framework based on random forests, where each decision tree on a local patch casts a vote on the location of the object and/or the pose of the object. Redondo-cabrera et al. [2014] extended the Hough Forests to a joint object detection and continuous pose estimation task.

3.1.5.2 Car direction estimation

Several works considered the car direction estimation task where the direction ranges from $0°$ and $360°$. Herdtweck and Curio [2013] modified regression forests so that the binary splitting minimizes a cost function specifically designed for direction estimation tasks. Torki and Elgammal [2011] applied supervised manifold learning and used RBF networks to learn a mapping from a point on the learnt manifold to the target space. Yang et al. [2014] proposed a special convolutional neural network referred to as an Automasking Neural Network (ANN) to jointly detect an object and estimate its pose as a continuous value. ANNs can automatically learn to select the most discriminative object parts across different viewpoints from training images. Fenzi and Ostermann [2014] proposed a method which combines continuous pose estimation for object categories based on feature regression and a graph matching strategy that disambiguates the pose solution.

Fenzi et al. [2015] proposed a regression method to perform continuous pose estimation of object categories on the basis of a spatially arranged ensemble of Fisher regressors.

In He et al. [2014], a joint object detection and pose estimation approach based on structured SVM is proposed. To achieve efficient inference, authors propose to first prune the search space and then refine the prediction. In the pruning stage, for each representative pose, a set of candidate bounding boxes is generated and in the refining stage, for each candidate, pose is optimized and a candidate with the highest score is returned as a final prediction.

3.1.5.3 Pedestrian orientation estimation

The pedestrian orientation problem has been studied mostly as a multiclass classification problem where the orientation is discretized into four or eight evenly spaced orientations, each $45°$ or $90°$ apart from each other. Then the performance evaluation is done using classification accuracy as the metric.

In Andriluka et al. [2010], eight orientation-specific pedestrian detectors based on a pictorial structured model are trained and the scores from the SVM-based detectors are combined to produce the final estimate of the orientation. Approaches based on classifiers but with holistic image features have also been proposed in Shimizu and Poggio [2004], Gandhi and Trivedi [2008], Nakajima et al. [2003], Chen et al. [2011], Zhao et al. [2012].

Enzweiler and Gavrila [2010] proposed an integrated framework for pedestrian classification and orientation estimation where view-specific pedestrian classifiers trained on positive and negative samples are used for orientation estimation. Similar to Baltieri et al. [2012], their method produces continuous prediction by modeling the distribution over the orientation as a mixture of Gaussian. Joint pedestrian classification and orientation estimation is conducted also by Tao and Klette [2013] and Goto et al. [2011].

Our voting-based ensemble technique is similar to the one proposed by Baltieri et al. [2012], where a set of Extremely Randomized Trees is adopted as classifiers and multi-scale HOG features are used as image features. They define a probability density function over the orientation space by treating outputs from the classifiers as continuous values. The final estimate is obtained by finding the mode of the probability density function by the mean shift algorithm. It is experimentally shown that by combining the outputs from the classifiers using a mixture of approximated Gaussian distributions, one can obtain significant improvements over methods which select the orientation with the highest classifier score. Similar to our work, this method allows continuous prediction, however, both training and evaluation are still done on discretized orientations. In addition, they use an approximated version of wrapped Gaussian distribution for the mean-shift while we use the von Mises distribution without any approximation in the mean-shift algorithm specifically derived for the von Mises distribution.

3.2 Methods

Throughout the paper, we denote a set of training data by $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^N$, where $\mathbf{x} \in \mathbb{R}^p$ is an input vector and $\mathbf{t} \in \mathbb{R}^q$ is a target vector. The goal of regression is to learn a function $F^*(\mathbf{x})$ such that the expected value of a certain loss function $\Psi(\mathbf{t}, F(\mathbf{x}))$ is minimized:

$$F^*(\mathbf{x}) = \operatorname*{argmin}_{F(\mathbf{x})} \mathrm{E}[\Psi(\mathbf{t}, F(\mathbf{x})]. \tag{3.1}$$

By approximating the above expected loss by an empirical loss and using the squared loss function, Eq. 3.1 is reformulated as minimizing the sum of squared errors (SSE):

$$F^*(\mathbf{x}) = \operatorname*{argmin}_{F(\mathbf{x})} \sum_{i=1}^{N} ||\mathbf{t}_i - F(\mathbf{x}_i)||_2^2. \tag{3.2}$$

However, other loss functions can also be used. In this chapter, we employ a specialized loss function to deal with tasks with a circular target space (Sec. 3.2.5).

In the following subsections, we first present an abstracted regression tree algorithm, followed by the presentation of a standard binary splitting method normally employed for regression tree training. We then describe the details of our splitting method. An algorithm to adaptively determine the number of child nodes is presented, followed by a modification of our method for the circular target space, which is necessary for orientation estimation tasks. Then the standard regression forest framework for combining regression trees is presented. Finally, we introduce two extensions of AKRF.

3.2.1 Abstracted Regression Tree Model

Regression trees are grown by recursively partitioning the input space into a set of disjoint partitions, starting from a root node which corresponds to the entire input space. At each node splitting stage, a set of splitting rules and prediction models for each partition are determined so as to minimize the certain loss (error). A typical choice for a prediction model is a constant model which is determined as a mean target value of training samples in the partition. However, higher order models such as linear regression can also be used. Throughout this work, we employ the constant model. After each partitioning, the corresponding child nodes are created and each training sample is forwarded to one of the child nodes. Each child node is further split if the number of the training samples belonging to that node is larger than a predefined number.

The essential component of regression tree training is an algorithm for splitting the nodes. Due to the recursive nature of the training stage, it suffices to discuss the splitting of the root node where all the training data are available. Subsequent splitting is done with a subset of the training data belonging to each node in exactly the same manner.

Formally, we denote a set of K disjoint partitions of the input space by $\mathcal{R} = \{r_1, r_2, \ldots, r_K\}$, a set of constant estimates associated with each partition by $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_K\}$ and the K clusters of the training data by $\mathbf{S} = \{S_1, S_2, \cdots, S_K\}$ where

$$S_k = \{i : \mathbf{x}_i \in r_k\}. \tag{3.3}$$

In the squared loss case, a constant estimate, $\mathbf{a}_k$, for the k-th partition is computed as the mean target vector of the training samples that fall into r_k:

$$\mathbf{a}_k = \frac{1}{|S_k|} \sum_{i \in S_k} \mathbf{t}_i. \tag{3.4}$$

The sum of squared errors (SSE) associated with each child node is computed as:

$$\mathrm{SSE}_k = \sum_{i \in S_k} ||\mathbf{t}_i - \mathbf{a}_k||_2^2, \tag{3.5}$$

where SSE_k is the SSE for the k-th child node. Then the sum of squared errors on the entire training data is computed as:

$$\mathrm{SSE} = \sum_{k=1}^{K} \mathrm{SSE}_k = \sum_{k=1}^{K} \sum_{i \in S_k} ||\mathbf{t}_i - \mathbf{a}_k||_2^2. \tag{3.6}$$

The aim of training is to find a set of splitting rules defining the input partitions which minimizes the SSE.

Assuming there is no further splitting, the regression tree is formally represented as

$$H(\mathbf{x}; \mathcal{A}, \mathcal{R}) = \sum_{k=1}^{K} \mathbf{a}_k \mathbb{1}(\mathbf{x} \in r_k), \tag{3.7}$$

where $\mathbb{1}$ is an indicator function. The regression tree outputs one of the elements of $\mathcal{A}$ depending on to which of the $\mathcal{R} = \{r_1, \ldots, r_K\}$, the new data $\mathbf{x}$ belongs. As mentioned earlier, the child nodes are further split as long as the number of the training samples belonging to the node is larger than a predefined number.

3.2.2 Standard Binary Node Splitting

In standard binary regression trees [Breiman et al., 1984], K is fixed at two. Each splitting rule is defined as a pair of the index of the input dimension and a threshold. Thus, each binary splitting rule corresponds to a hyperplane that is perpendicular to one of the axes. Among a predefined set of such splitting rules, the one that minimizes the overall SSE, as defined in Eq. 3.6, is selected by trial-and-error.

The major drawback of the splitting procedure presented above is that the splitting rules are determined by exhaustively searching the best splitting rule among the predefined set of candidate rules. Essentially, this is the reason why only simple binary splitting rules defined as thresholding on a single dimension are considered in the training stage. Since the candidate rules are severely limited, the selected rules are not necessarily the best among all possible ways to partition the input space.

3.2.3 Proposed Node Splitting

In order to overcome the drawbacks of the standard binary splitting procedure, we propose a new splitting procedure which does not rely on trial-and-error. A graphical illustration of the algorithm is given in Fig. 3.2. At each node splitting stage, we first find ideal clusters $\mathbf{T} = \{T_1, T_2, \cdots, T_K\}$ of the training data associated with the node, those that at least locally minimize the following objective function:

$$\min_{\mathbf{T}} \sum_{k=1}^{K} \sum_{i \in T_k} ||\mathbf{t}_i - \mathbf{a}_k||_2^2 \tag{3.8}$$

where $T_k = \{i : ||\mathbf{t}_i - \mathbf{a}_k||_2 \leq ||\mathbf{t}_i - \mathbf{a}_j||_2, \forall\, 1 \leq j \leq K\}$ and $\mathbf{a}_k = \frac{1}{|T_k|} \sum_{i \in T_k} \mathbf{t}_i$. This minimization can be done by applying the k-means clustering algorithm in the target space with K as the number of clusters. Note the similarity between the objective functions in Eq. 3.8 and Eq. 3.6. The difference is that in Eq. 3.6, clusters in $\mathbf{S}$ are indirectly determined by the splitting rules defined in the input space while the clusters in $\mathbf{T}$ are directly determined by the k-means algorithm without taking into account the input space.

After finding $\mathbf{T}$, we find partitions $\mathcal{R} = \{r_1, \ldots, r_K\}$ of the input space which preserve $\mathbf{T}$ as much as possible. This task is equivalent to a K-class classification problem which aims at determining a cluster ID of each data point based on $\mathbf{x}$. Note that here, what we truly care is the generalization ability of the classifier on unseen data points. Among existing classification methods, we employ the L2-regularized L2-loss linear SVM with a one-versus-rest approach due to its proven generalization ability and low computational time for both training and testing. Formally, we solve the following optimization problem for each cluster using LIBLINEAR [Fan et al., 2008]:

$$\min_{\mathbf{w}_k} ||\mathbf{w}_k||_2 + C \sum_{i=1}^{N} (\max(0, 1 - l_i^k \mathbf{w}_k^T \mathbf{x}_i))^2, \tag{3.9}$$

where $\mathbf{w}_k$ is the weight vector for the k-th cluster, $l_i^k = 1$ if $i \in T_k$ and -1 otherwise and $C > 0$ is a penalty parameter. We set $C = 1$ throughout the paper. Each training sample is forwarded to one of the K child nodes by

$$k^* = \underset{k \in \{1, \cdots, K\}}{\text{argmax}} \ \mathbf{w}_k^T \mathbf{x}. \tag{3.10}$$

At the last stage of the node splitting procedure, we compute $\mathbf{S}$ defined in Eq. 3.3 and $\mathcal{A}$ defined in Eq. 3.4 based on the constructed splitting rules in Eq. 3.10.

Unlike standard binary splitting, the proposed splitting rules are not limited to hyperplanes that are perpendicular to one of the axes and the clusters are found without being restricted to a set of predefined splitting rules in the input space. Furthermore, our splitting strategy allows each node to have more than two child nodes by employing $K > 2$, adding one more level of flexibility to the model. Note that larger K generally results in smaller value for Eq. 3.8, however, since the subsequent classification problem becomes more difficult, a larger K does not necessarily lead to an improved performance.

3.2.4 Adaptive determination of K

Since K is a parameter, we need to determine the value for K by a time consuming cross-validation step. In order to avoid the cross-validation step while achieving comparative performance, we propose a method to adaptively determine K at each node based on the sample distribution.

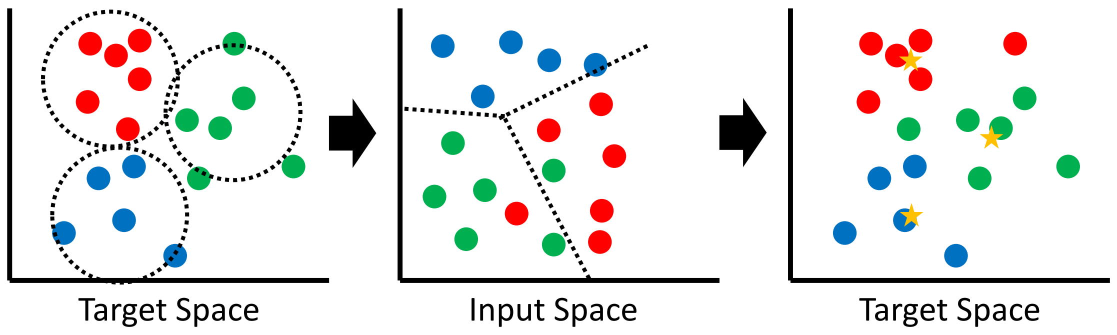


Figure 3.2: An illustration of the proposed splitting method ($K = 3$). A set of clusters of the training data is found in the target space by k-means (left). The input partitions preserving the found clusters as much as possible are determined by an SVM (middle). If no more splitting is needed, a mean is computed as a constant estimate for each set of colored samples. The yellow stars represent the means (right). Note that the color of some points change due to misclassification. If additional splitting is needed, clusterling is applied to each set of colored samples separately in the target space.

In this work we adopt a criterion proposed in x-means clustering algorithm [Pelleg and Moore, 2000], an extension of the k-means, where the number of clusters is adaptively determined by the Bayesian Information Criterion (BIC) [Kashyap, 1977, Schwarz, 1978]. The BIC is designed to balance the model complexity and likelihood. In the x-means algorithm, the number of clusters is increased by splitting initial clusters until the BIC does not improve. Although we use the same criterion, unlike x-means, we determine K by running k-means independently with each candidate value of K, and select one which achieves the lowest BIC value. As a result, when a target distribution is complex, a larger value of K is selected and when the target distribution is simple, a smaller value of K is selected. This is in contrast to the non-adaptive method where a fixed number of

K is used regardless of the complexity of the distributions.

To compute the BIC, a probabilistic interpretation of the k-means algorithm is necessary. As in Pelleg and Moore [2000], we assume that the data are generated from a mixture of isotropic weighted Gaussians with a shared variance. The unbiased estimate for the shared variance is computed as[1]

$$\hat{\sigma}^2 = \frac{1}{q(N-K)} \sum_{k=1}^{K} \sum_{i \in T_k} ||\mathbf{t}_i - \mathbf{a}_k||_2^2. \tag{3.11}$$

A point probability density for a data point $\mathbf{t}$ belonging to the k-th cluster is computed as follows:

$$p(\mathbf{t}) = \frac{|T_k|}{N} \frac{1}{\sqrt{2\pi\hat{\sigma}^2}^q} \exp(-\frac{||\mathbf{t} - \mathbf{a}_k||_2^2}{2\hat{\sigma}^2}). \tag{3.12}$$

After simple calculations, the log-likelihood of the data is obtained as

$$\ln \mathcal{L}(\{\mathbf{t}_i\}_{i=1}^N) = \ln \mathbf{\Pi}_{i=1}^N p(\mathbf{t}_i) = \sum_{k=1}^{K} \sum_{i \in T_k} \ln p(\mathbf{t}_i) =$$

$$-\frac{qN}{2} \ln(2\pi\hat{\sigma}^2) - \frac{q(N-K)}{2} + \sum_{k=1}^{K} |T_k| \ln |T_k| - N \ln N \tag{3.13}$$

Finally, the BIC for a particular value of K is computed as

$$\text{BIC}_K = -2 \ln \mathcal{L}(\{\mathbf{t}_i\}_{i=1}^N) + F_k \ln N. \tag{3.14}$$

where $F_k = (K - 1 + qK + 1)$ is the number of free parameters ($K - 1$ cluster priors, K q-dimensional centroids and 1 shared variance).

At each node splitting stage, we run the k-means algorithm for each value of K in a manually specified range and select K with the smallest BIC. Since SVM training is done

[1]In Hara and Chellappa [2014] and Pelleg and Moore [2000], the variance is incorrectly estimated by missing q in the denominator.

only once with the selected K, the computation time is not significant. Throughout this work, we select K from $\{2, 3, \ldots, 40\}$.

3.2.5 Modification for a Circular Target Space

1D orientation estimation of objects such as cars and pedestrians is unique in that the target variable is periodic, namely, 0° and 360° represent the same direction angle. Thus, the target space can be naturally represented as a unit circle, which is a 1D Riemannian manifold in R^2. To deal with such a target space, special treatments are needed since the Euclidean distance is inappropriate. For instance, the distance between 10° and 350° should be shorter than that between 10° and 50° on this manifold.

In our method, such orientation estimation problems are naturally addressed by modifying the k-means algorithm and the computation of BIC. The remaining steps are kept unchanged. The k-means clustering method consists of computing the cluster centroids and hard assignment of the training samples to the closest centroid. Finding the closest centroid on a circle is trivially done by using the length of the shorter arc as a distance. Due to the periodic nature of the variable, the arithmetic mean is not appropriate for computing the centroids. A typical way to compute the mean of angles is to first convert each angle to a 2D point on a unit circle. The arithmetic mean is then computed on a 2D plane and converted back to the angular value. More specifically, given a set of orientation angles $t, \ldots, t_N$, the mean orientation a is computed by

$$a = \mathrm{atan2}(\frac{1}{N}\sum_{i=1}^{N}\sin t_i, \frac{1}{N}\sum_{i=1}^{N}\cos t_i). \tag{3.15}$$

It is known [Gaile and Burt, 1980] that a minimizes the sum of a certain distance defined on a circle,

$$a = \underset{s}{\operatorname{argmin}} \sum_{i=1}^{N} d(t_i, s) \tag{3.16}$$

where $d(q, s) = 1 - \cos(q - s) \in [0, 2]$. Thus, k-means clustering using the above definition of means finds clusters $\mathbf{T} = \{T_1, T_2, \cdots, T_K\}$ of the training data that at least locally minimize the following objective function,

$$\min_{\mathbf{T}} \sum_{k=1}^{K} \sum_{i \in T_k} (1 - \cos(t_i - a_k)) \tag{3.17}$$

where $T_k = \{i : 1 - \cos(t_i - a_k) \leq 1 - \cos(t_i - a_j), \forall\, 1 \leq j \leq K\}$.

Using the k-means algorithm presented above in splitting a node essentially means that we employ distance $d(q, s)$ as a loss function in Eq. 3.1. Although squared shorter arc length might be more appropriate for the orientation estimation task, there is no constant time algorithm to find the mean which minimizes it. Also, as will be explained shortly, the above definition of the mean coincides with the maximum likelihood estimate of the mean of a certain probability distribution defined on a circle.

As in the Euclidean target case, we can also adaptively determine the value for K at each node using BIC. As a density function, the Gaussian distribution is not appropriate. A suitable choice is the von Mises distribution, which is a periodic continuous probability distribution defined on a circle,

$$p(t|a, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp\left(\kappa \cdot \cos(t - a)\right) \tag{3.18}$$

where a and κ are the mean angle and concentration parameter, respectively, analogous to the mean and variance of the Gaussian distribution, and I_λ is the modified Bessel function of order λ. It is known [Fisher, 1996] that the maximum likelihood estimate of

a is computed by Eq. 3.15 and that of κ satisfies

$$\frac{I_1(\kappa)}{I_0(\kappa)} = \sqrt{(\frac{1}{N}\sum_{i=1}^{N}\sin t_i)^2 + (\frac{1}{N}\sum_{i=1}^{N}\cos t_i)^2} \tag{3.19}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\cos(t_i - a). \tag{3.20}$$

Note that, from the second term, the above quantity is the Euclidean norm of the mean vector obtained by converting each angle to a 2D point on a unit circle.

Similar to the derivation for the Euclidean case, we assume that the data are generated from a mixture of weighted von Mises distributions with a shared κ. The mean a_k of k-th von Mises distribution is same as the mean of the k-th cluster obtained by k-means clustering. The shared value for κ is obtained by solving the following equation

$$\frac{I_1(\kappa)}{I_0(\kappa)} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i\in T_k}\cos(t_i - a_k). \tag{3.21}$$

Since there is no closed form solution for the above equation, we use the following approximation proposed in Mardia and Jupp [2000],

$$\kappa \approx \frac{1}{2(1-\frac{I_1(\kappa)}{I_0(\kappa)})}. \tag{3.22}$$

Then, a point probability density for a data point t belonging to the k-th cluster is computed as:

$$p(t|a_k,\kappa) = \frac{|T_k|}{N}\frac{\exp(\kappa\cdot\cos(t-a_k))}{2\pi I_0(\kappa)}. \tag{3.23}$$

After simple calculations, the log-likelihood of the data is obtained as

$$\ln \mathcal{L}(\{t_i\}_{i=1}^N) = \ln \mathbf{\Pi}_{i=1}^N p(t_i) = \sum_{k=1}^K \sum_{i \in T_k} \ln p(t_i) =$$
$$-N \ln(2\pi I_0(\kappa)) + \kappa \sum_{k=1}^K \sum_{i \in T_k} \cos(t_i - a_k)$$
$$+ \sum_{k=1}^K |T_k| \ln |T_k| - N \ln N. \tag{3.24}$$

Finally, the BIC for a particular value of K is computed as

$$\mathrm{BIC}_K = -2 \ln \mathcal{L}(\{t_i\}_{i=1}^N) + 2K \ln N. \tag{3.25}$$

where the last term is obtained by putting $q = 1$ into the last term of Eq. 3.14.

3.2.6 Random Regression Forest

We use the regression forest [Breiman, 2001] as the final regression model. The regression forest is an ensemble learning method for regression which first constructs multiple regression trees from random subsets of training data. In a standard regression forest, testing is done by computing the mean of the outputs from each regression tree. We denote the ratio of random samples as $\beta \in (0, 1.0]$. For the Euclidean target case, the arithmetic mean is used to obtain the final estimate and for the circular target case, the mean defined in Eq. 3.15 is used.

For the regression forest with standard binary regression trees, an additional randomness is typically injected. In finding the best splitting function at each node, only a randomly selected subset of the feature dimensions is considered. We denote the ratio of randomly chosen feature dimensions as $\gamma \in (0, 1.0]$. For the regression forest with our regression trees, we always consider all the feature dimensions. However, another form of randomness is naturally injected by randomly selecting the data points as the initial cluster centroids in the k-means algorithm.

3.2.7 Further extensions

We propose two extensions to our method in order to handle more challenging tasks such as the pedestrian orientation estimation task. The first extension is done to address the data imbalance issue in the training data. We compute weights of the training data as a reciprocal of the density and then construct random subset of the training data for regression forest training considering those weights.

The second extension is done to address the multiple peak issue of the predictions which causes flipping errors in orientation estimation tasks. Unlike the standard regression trees where a single value is attached to each leaf node, we retain all the target values of the training samples at each leaf node during training and then in testing stage, we allow those multiple samples at leaf nodes to cast probabilistic votes in the target space. We then find the highest mode of the distribution using a mean shift algorithm [Fukunaga and Hostetler, 1975, Cheng, 1995, Comaniciu and Meer, 2002]. For tasks with a circular target space, we apply the newly derived mean shift algorithm for a circular space presented in Sec. 3.2.7.3.

3.2.7.1 Sample weighting technique for imbalanced training data

In regression training, typically each training sample is treated equally, i.e., the weight of each training sample is equal, by assuming that the training data and testing data are generated from the same distribution. However, in many practical settings, we collect training data from various scenes and deploy the trained model in some unknown scenes. Thus, in many cases it is no longer valid to assume that the training data and testing data are generated from the same distribution. Since, in general, we do not know the distribution in test scenes, in applications such as orientation estimation tasks, it is best to assume that the distribution of the target values is uniform in testing time. On the other hand, labels of training data may not follow a uniform distribution, leading to unbalanced number of training samples across orientations/poses. To alleviate this problem, we assign a different weight to each training sample in order to bring the underlying distribution of the target values closer to uniform distribution. The proposed technique is intended to be used when A) the training data distribution and test data distribution are largely different AND B) target variable's distribution of the test data is believed to be close to uniform. If the condition A is not satisfied, we can just train without the sample weighting technique and use it for testing. If the condition B is not satisfied, the performance could be worse since the sample weighting technique trains a model on weighted training data whose weights are determined to bring the target variable's distribution to uniform.

In this work, we employ the standard approach where the weight w_i for the i-th training sample is computed by $w_i = \frac{1}{\hat{p}(t_i)}$. We compute the probability density estimate $\hat{p}(t)$ using kernel density estimation. For tasks with the Euclidean target space, we use a

Gaussian kernel. For tasks with a circular target space, we use the von Mises distribution as a kernel function. We denote the concentration parameter of the von Mises kernel by η.

The choice of the concentration parameter η of the von Mises kernel (as well as the standard deviation of the Gaussian kernel) are extremely important as large values of η lead to highly variable estimates whereas small values lead to oversmoothed density estimates. To determine η, we use the likelihood cross-validation method [Habbema and Hermans, 1977, Duin, 1976] modified for the circular space. The standard deviation of the Gaussian kernel can be determined in a similar manner.

Let D_i denote the observations with t_i excluded, i.e., $D_i = \{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_N\}$. Then η is selected as the solution to the following maximization problem:

$$\eta^* = \underset{\eta}{\operatorname{argmax}} \, \mathbf{\Pi}_{i=1}^{N} \hat{p}(t_i; \eta, D_i) \tag{3.26}$$

The solution is found by exhaustive search.

The random subsets of training data for regression forest training are constructed by weighted random sampling with replacement, i.e., each sample is randomly chosen with probability $w_i / \sum_{i=1}^{N} w_i$.

3.2.7.2 Voting-based ensemble using the mean shift algorithm

Our voting-based ensemble method is invoked only in the testing stage. In the testing stage, for each regression tree in the forest, an unseen data point x_{new} is directed to one of the leaf nodes. A set of weighted samples retained in the leaf node is then used for casting probabilistic votes in the output space. Before voting, for each leaf node, we normalize the weights to make the contribution from each tree equal. Note that this is necessary as the number of training samples at each leaf node varies.

Below we discuss a subsequent procedure for the circular target space. Assume we obtain V weighted votes from M trees. We model each vote as a weighted von Mises distribution and find the highest mode using the mean shift algorithm which is derived for a circular space in Sec. 3.2.7.3. The highest mode found is used as the predicted value.

We determine the value for ν of the von Mises distribution by model validation using held-out validation data. Note that since ν is a parameter used only in test time, the model verification for ν does not involve training a model with each candidate value of ν. Thus, the model validation process is computationally simple. In a preliminary experiment, we also tried ν estimated by the weighted version of the likelihood cross-validation as in Eq. 3.26, however, the performance is not satisfactory.

For tasks with the Euclidean target space, we use the mean shift algorithm with the Gaussian distribution instead of the von Mises distribution. The standard deviation of the Gaussian distribution is determined in a similar manner.

3.2.7.3 Mean shift algorithm for a circular space

The mean shift algorithm was originally proposed for the Euclidean space. Here, we derive the mean shift algorithm for a circular space with weighted data. Note that the data here is a set of weighted votes from regression trees in our context. The derivation of the mean shift algorithm starts by assuming that the underlying distribution is obtained by the kernel density estimation. Since the space is a circular space, we assume that the distribution is computed with a von Mises kernel.

Given V weighted votes $\Gamma = \{(\theta_1, \mathrm{w}_1), \dots, (\theta_V, \mathrm{w}_V)\}$, where θ is an angle in the circular space and w is a weight, the density is defined as

$$\hat{p}(\theta; \nu, \Gamma) = \frac{1}{Z} \sum_{i=1}^{V} \mathrm{w}_i K(\theta - \theta_i) \tag{3.27}$$

$$= \frac{1}{2\pi Z I_0(\nu)} \sum_{i=1}^{V} \mathrm{w}_i \exp\{\nu \cos(\theta - \theta_i)\}. \tag{3.28}$$

where $Z = \sum_{i=1}^{V} \mathrm{w}_i$ and ν is a concentration parameter.

Let $k(\theta) = \exp(\nu \cos\sqrt{\theta}), \;\; (\theta \geq 0)$. Then Eq. 3.28 is redefined as

$$\hat{p}(\theta; \nu, \Gamma) = \frac{1}{2\pi Z I_0(\nu)} \sum_{i=1}^{V} \mathrm{w}_i k(|\theta - \theta_i|^2). \tag{3.29}$$

By defining $g(\theta) = -k'(\theta)$, we obtain

$$g(\theta) = -k'(\theta) \tag{3.30}$$

$$= \frac{\nu}{2\sqrt{\theta}} \sin\sqrt{\theta} \exp(\nu \cos\sqrt{\theta}), \;\; (\theta \geq 0) \tag{3.31}$$

Note that $g(0) = \frac{\nu}{2}\exp(\nu)$ since $\lim_{\theta \to 0} \frac{\sin\sqrt{\theta}}{\sqrt{\theta}} = 1$

The derivative of $\hat{p}(\theta; \nu, \Gamma)$ with respect to θ is

$$\frac{\delta}{\delta\theta}\hat{p}(\theta; \nu, \Gamma) \tag{3.32}$$

$$= \frac{2}{2\pi Z I_0(\nu)} \sum_{i=1}^{V} \mathrm{w}_i(\theta - \theta_i) k'(|\theta - \theta_i|^2) \tag{3.33}$$

$$= \frac{1}{\pi Z I_0(\nu)} \sum_{i=1}^{V} \mathrm{w}_i(\theta_i - \theta) g(|\theta - \theta_i|^2) \tag{3.34}$$

$$= \frac{1}{\pi Z I_0(\nu)} (\sum_{i=1}^{V} \mathrm{w}_i g(|\theta - \theta_i|^2)) (\frac{\sum_{i=1}^{V} \mathrm{w}_i \theta_i g(|\theta - \theta_i|^2)}{\sum_{i=1}^{V} \mathrm{w}_i g(|\theta - \theta_i|^2)} - \theta) \tag{3.35}$$

$$= \frac{1}{\pi Z I_0(\nu)} (\sum_{i=1}^{V} \mathrm{w}_i g(|\theta - \theta_i|^2)) \mathbf{m}(\theta; \nu, \Gamma), \tag{3.36}$$

where $\mathbf{m}(\theta;\nu,\Gamma)$ is the mean shift and

$$g(|\theta-\theta_i|^2) \tag{3.37}$$

$$= \frac{\nu}{2\sqrt{|\theta-\theta_i|^2}} \sin\sqrt{|\theta-\theta_i|^2} \exp\left(\nu\cos\sqrt{|\theta-\theta_i|^2}\right) \tag{3.38}$$

$$= \frac{\nu}{2|\theta-\theta_i|} \sin|\theta-\theta_i| \exp\left(\nu\cos|\theta-\theta_i|\right). \tag{3.39}$$

Given the current estimate of the mode, $\theta^{(s)}$, the updated estimate is computed by

$$\theta^{(s+1)} = \theta^{(s)} + \mathbf{m}(\theta^{(s)};\nu,\Gamma). \tag{3.40}$$

The process is started from each data point and repeated until $d(\theta^{(s+1)},\theta^{(s)})$ becomes small. Each convergence point is a mode in the distribution. The height of the mode is computed as the density at the mode (See Eq. 3.28). To reduce the computation time, we keep track of a path until convergence and assume that all the data points in that path converge to the same point.

Note that due to the periodic nature of the orientation, θ_i and $\theta_i \pm 2k\pi\,(k=1,2,\dots)$ represent the same orientation. However, in computing $\mathbf{m}(\theta;\nu,\Gamma)$, it is important to use θ_i which has the smallest value for $|\theta-\theta_i|$. Thus, $0 \le |\theta-\theta_i| \le \pi$ for all i. This is to ensure that the algorithm finds the nearest mode to the current estimate.

3.3 Experiments

3.3.1 Head Pose Estimation

3.3.1.1 Dataset and image features

We test the effectiveness of the proposed methods for the head pose estimation task on the Euclidean target space. We adopt Pointing'04 dataset [Gourier et al., 2004]. The dataset contains head images of 15 subjects and for each subject there are two series of 93 images with different poses represented by pitch and yaw.

The dataset comes with manually specified bounding boxes indicating the head regions. Based on the bounding boxes, we crop and resize the image patches to 64×64 pixels image patches and compute multiscale HOG [Dalal and Triggs, 2005] from each image patch with cell size 8, 16, 32 and 2×2 cell blocks. The orientation histogram for each cell is computed with signed gradients for 9 orientation bins. The resulting HOG feature is 2124 dimensional.

In the sample weighting step of AKRF-W, the likelihood cross-validation is used to determine the bandwidth of the Gaussian kernel, however, since in Pointing'04 dataset there are multiple samples whose target values are exactly the same, the obtained bandwidth becomes infinite. Thus, we conduct likelihood cross-validation after removing the duplicate samples.

3.3.1.2 Results

First, we compare the proposed methods with other general regression methods using the same image features. We choose standard binary regression forest (BRF) [Breiman, 2001], Boosted Binary Regression Tree (BBRT) [H. Friedman, 2001], kernel PLS [Rosipal and Trejo, 2001] and ϵ-SVR with Radial Basis Function (RBF) kernels [Vapnik, 1998], all of which have been widely used for various computer vision tasks. The first series of images from all subjects are used as the training set and the second series of images are used for testing. The performance is measured by the Mean Absolute Error in degree. For our methods as well as BRF, we terminate node splitting once the number of training data associated with each leaf node is less than 5. The number of trees combined is set to 20. K for KRF, β for KRF, AKRF, AKRF with the proposed extensions and BRF, and γ for BRF are all determined by 5-fold cross-validation on the training set. For BBRT, we use the implementation by Hara and Chellappa [2013] and the number of leaf nodes of regression trees is set to 35 as a result of cross-validation and the number of trees is set to 1000. Other parameters are set to the default values. For kernel PLS, we use the implementation provided by the author of Rosipal and Trejo [2001] and for ϵ-SVR, we use the LIBSVM package [Chang and Lin, 2011]. All the parameters for kernel PLS and ϵ-SVR are also determined by 5-fold cross-validation.

As can been seen in Table 3.1, all of the proposed methods work significantly better than other regression methods. The AKRF performs worse than the KRF, however, the AKRF with the voting-based ensemble (AKRF-V) improves the performance of the AKRF by 13.2%, surpassing the KRF. On the other hand, the AKRF with the sample weighting technique (AKRF-W) deteriorates the performance of the AKRF. This result is expected since the target value distributions are not uniform in the testing set.

Our methods are computationally efficient (Table 3.1). KRF and AKRF take only 7.7 msec and 8.7 msec, respectively, to process one image including feature computation with a single thread. AKRF-W does not increase the computation time for testing while AKRF-V slightly increases the computation time (9.6 msec) due to the mean shift procedure being invoked in testing time.

Table 3.1: MAE in degree of different regression methods on the Pointing'04 dataset (even train/test split). Time to process one image including HOG computation is also shown.

Methods	yaw	pitch	average	testing time (msec)
AKRF-V	4.98	3.43	4.20	9.6
AKRF-W	6.02	4.61	5.31	8.7
AKRF	5.57	4.11	4.84	8.7
KRF	5.32	3.52	4.42	7.7
BRF [Breiman, 2001]	7.77	8.01	7.89	4.5
BBRT [H. Friedman, 2001]	7.74	7.82	7.78	112.7
Kernel PLS [Rosipal and Trejo, 2001]	7.35	7.02	7.18	86.2
ϵ-SVR [Vapnik, 1998]	7.34	7.02	7.18	189.2

Table 3.2 compares the proposed methods with prior art. Since the previous works report the 5-fold cross-validation estimate on the whole dataset, we also follow the same

protocol. KRF, AKRF, and AKRF-V advance state-of-the-art with 26.8%, 16.9% and 21.4% reduction in the average MAE, respectively. As in the previous experimental setting, AKRF-W deteriorates the performance of AKRF.

Table 3.2: Head pose estimation results on the Pointing'04 dataset (5-fold cross-validation)

Methods	yaw	pitch	average
AKRF-V	5.53	2.86	4.19
AKRF-W	5.71	4.19	4.95
AKRF	5.43	3.43	4.43
KRF	5.29	2.51	3.90
He et al. [2014]	5.71	4.95	5.33
Fenzi et al. [2013]	5.94	6.73	6.34
Haj et al. [2012] Kernel PLS	6.56	6.61	6.59
Haj et al. [2012] PLS	11.29	10.52	10.91

A recent work by Zhen et al. [2015] proposed a supervised feature learning method for multidimensional target regression and compared their features with various feature learning techniques on Pointing'04 Dataset. For all the experiments, they use our AKRF as a regression method and achieve significant improvement over the multiscale HOG we use in this paper (3.81 average MAE using even split and 3.11 using 5-fold cross-validation setting.) Please refer to Zhen et al. [2015] for the full comparisons.

Fig. 3.3 shows the effect of K of KRF on the average MAE along with the average

MAE of AKRF. In this experiment, the cross-validation process successfully selects K with the best performance. AKRF works better than KRF with the second best K. The overall training time is much faster with AKRF since the cross-validation step for determining the value of K is not necessary. To train a single regression tree with $\beta = 1$, AKRF takes only 6.8 sec while KRF takes 331.4 sec for the cross-validation and 4.4 sec for training a final model. As a reference, BRF takes 1.7 sec to train a single tree with $\beta = 1$ and $\gamma = 0.4$. Finally, some estimation results by AKRF on the second sequence of person 13 are shown in Fig. 3.4.

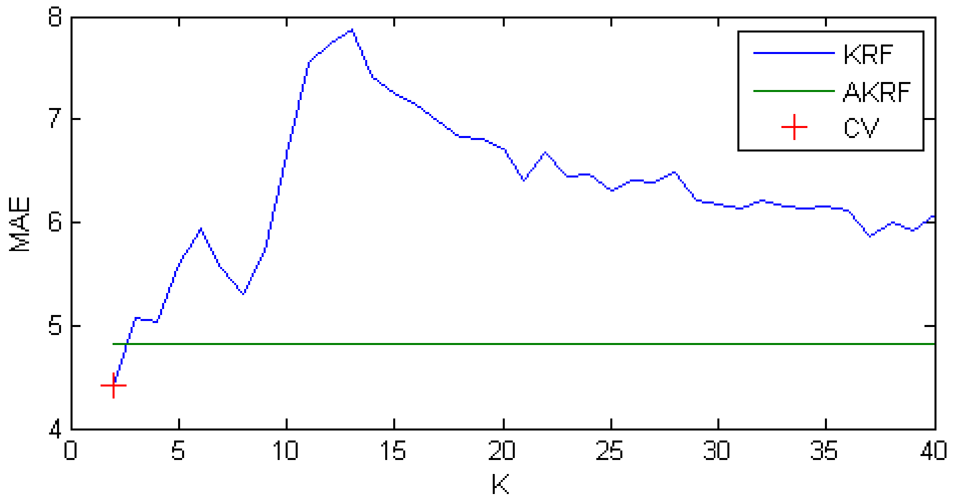


Figure 3.3: Pointing'04: The effect of K of KRF on the average MAE. "CV" indicates the value of KRF selected by cross-validation.

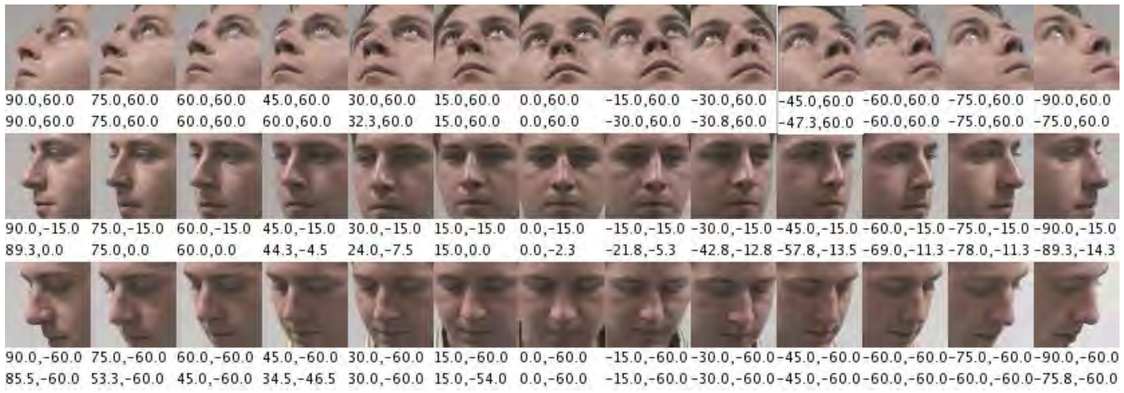


Figure 3.4: Some estimation results of the second sequence of person 13. The top numbers are the ground truth yaw and pitch and the bottom numbers are the estimated yaw and pitch.

3.3.2 Car Direction Estimation

3.3.2.1 Dataset and image features

We test KRF, AKRF and its extensions, AKRF-W (AKRF with the sample weighting) and AKRF-VW (AKRF with the sample weighting and voting-based ensemble), for a circular target space on the EPFL Multi-view Car Dataset [Ozuysal et al., 2009]. The dataset contains 20 sequences of images of cars with various directions. Each sequence contains images of only one instance of car. In total, there are 2299 images in the dataset. Each image comes with a bounding box specifying the location of the car and ground truth for the direction of the car. The direction ranges from 0° to 360°. In Fig. 3.5, we show a histogram of the car directions computed from the training data. The car directions are not uniformly distributed. As input features, multiscale HOG features [Dalal and Triggs, 2005] with the same parameters as in the previous experiment are extracted from 64×64

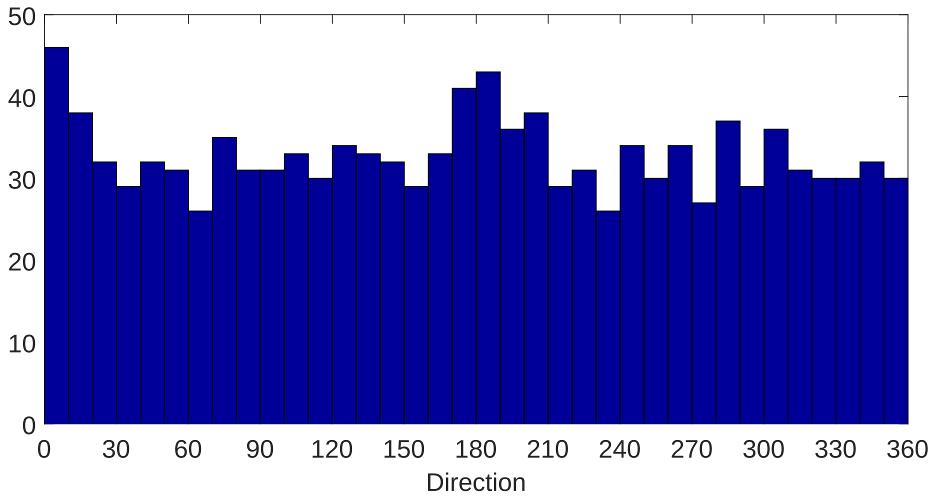


Figure 3.5: EPFL Multi-view Car Dataset: a histogram obtained from the directions on the training data. The car directions are not uniformly distributed.

pixels image patches obtained by resizing the given bounding boxes.

3.3.2.2 Results

The algorithm is evaluated by using the first 10 sequences for training and the remaining 10 sequences for testing. In Table 3.3, we compare the proposed algorithms with BRF, Kernel PLS and ϵ-SVR with RBF kernels using the same HOG features. We also include the performance of previous works. For BRF, we extend it to directly minimize the same loss function ($d(q, s) = 1 - \cos(q - s)$) as with our methods. For Kernel PLS and ϵ-SVR, we first map direction angles to 2D points on a unit circle and train regressors using the mapped points as target values. In testing phase, a 2D point coordinate (x, y) is first estimated and then mapped back to the angle by $\mathrm{atan2}(y, x)$. All the parameters are determined by leave-one-sequence-out cross-validation on the training set. The performance is evaluated by the Mean Absolute Error (MAE) measured in degrees. In addition, the MAE of 90-th percentile of the absolute errors and that of 95-th percentile are reported, following the convention from prior works.

As can be seen from Table 3.3, all of our proposed methods work much better than existing regression methods we have compared with. In particular, the improvement over BRF is noteworthy. Compared to AKRF, AKRF-W works slightly better (4.0% reduction in MAE 90-th percentile). The use of the voting-based ensemble (AKRF-VW) further improves the performance (in total, 12.5% reduction in MAE 90-th percentile). In Fig. 3.6, we show the MAE of AKRF computed on each sequence in the testing set. The performance varies significantly among different sequences (car instances). Fig. 3.7 shows some representative results from the *worst* three sequences in the testing set (seq 16, 20 and 15). We notice that most of the failure cases are still due to the flipping errors ($\approx 180^\circ$) which mostly occur at particular intervals of directions. Fig. 3.8 shows the effect of K of KRF. The performance of AKRF is comparable to that of KRF with K selected by the cross-validation.

3.3.3 Continuous Pedestrian Orientation Estimation

3.3.3.1 Dataset

We conducted experiments on the TUD Multiview Pedestrians Dataset [Andriluka et al., 2010] which consists of 5,228 images of pedestrians with bounding box annotations as well as orientation annotations. Most of the training images are gray scale images. In total, there are 4,732 pedestrians for training, 290 for validation and 309 for testing. Note

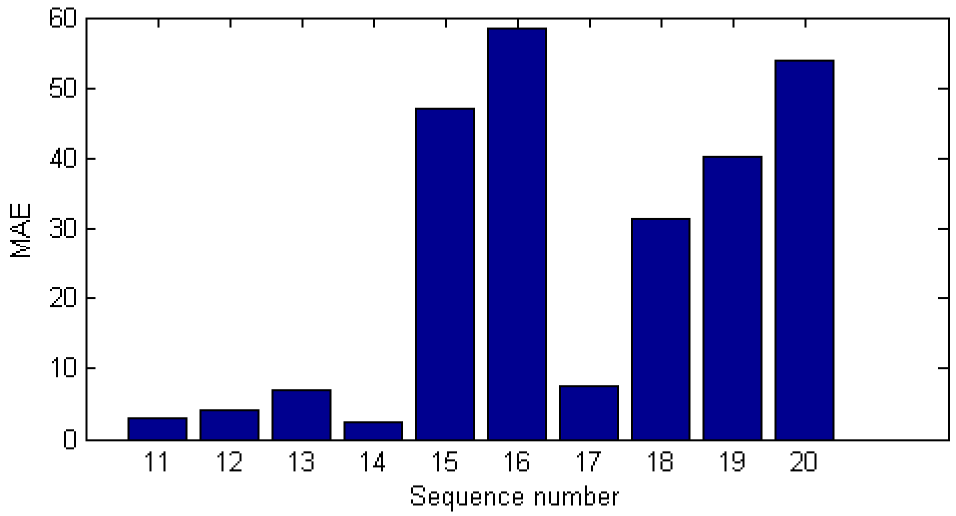


Figure 3.6: MAE of AKRF computed on each sequence in the testing set of the EPFL Multiview Car Dataset

that the size of the dataset is more than two times larger than that of EPFL Multi-view Car Dataset and slightly smaller than two times of Pointing'04 Dataset. Unlike those two datasets, all the images in this dataset are captured "in the wild" and images contain a large variety of poses and clothing, making this dataset much more challenging.

3.3.3.2 Annotation of continuous orientations

Since it is difficult to measure the accurate orientations of pedestrians captured in a real-life setting, the original annotations for orientations are given in a discrete form. Specifically, each pedestrian is labeled as one of the eight orientation classes (Right, Right-Back, Back, Left-Back, Left, Left-Front, Front, Right-Front). Thus, all the previous works using this dataset treat the problem as a mutli-class classification problem.

In this work, we annotate the orientations of the pedestrians in a continuous form

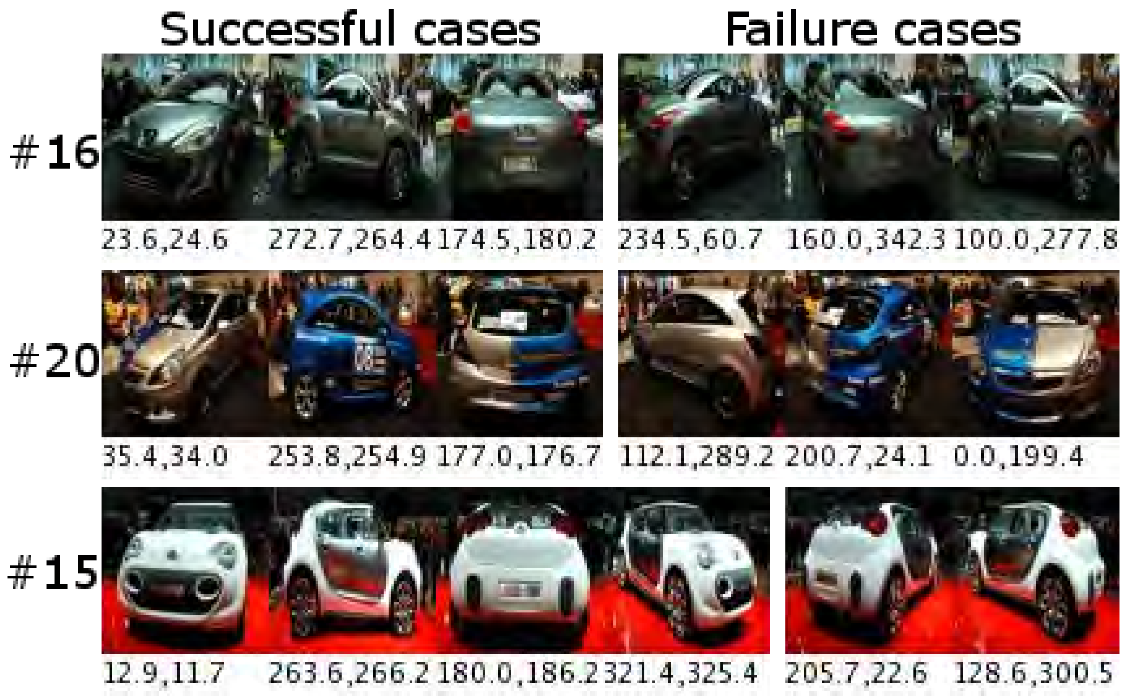


Figure 3.7: Representative results from the *worst* three sequences in the testing set. The numbers under each image are the ground truth direction (left) and the estimated direction (right). Most of the failure cases are due to the flipping error.

using the Amazon Mechanical Turk. For each pedestrian, annotators specify the orientation of the pedestrian by moving a line segment in a circle (Fig. 3.1). The orientation of the pedestrian is defined as body orientation. We obtain 5 annotations for each pedestrian from 5 unique annotators. We then compute the mean orientation of the annotations by Eq. 3.15 and use it as a ground truth continuous annotation. The mean absolute deviation of the annotations from the mean is $9.6°$. We believe that the effect of perspective errors is small in the annotations since most of the pedestrians are photographed from a sufficiently large distance compared to the thickness of the human body.

To confirm the usefulness of continuous annotations, for each angle in $\{0°, 10°, \ldots, 350°\}$,

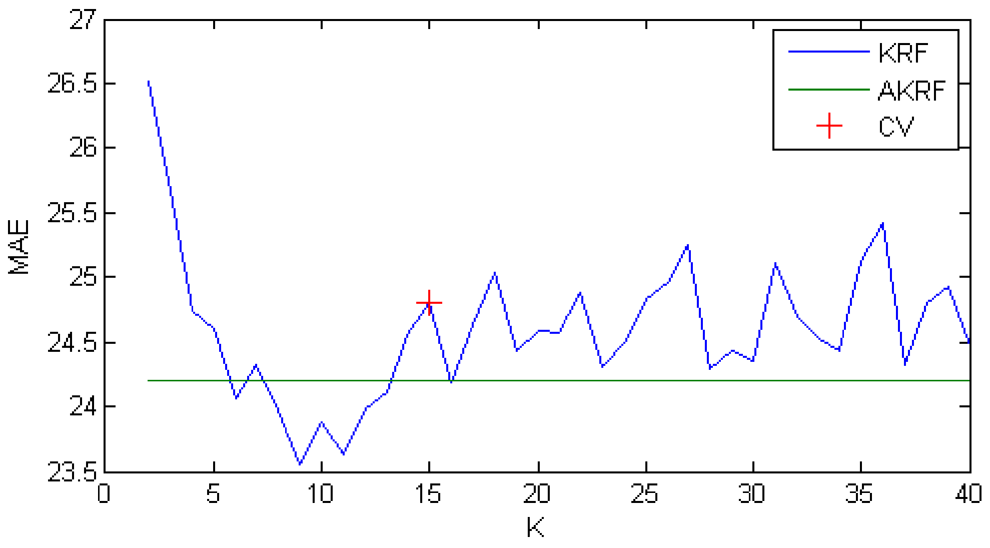


Figure 3.8: EPFL Multi-view Car: The effect of K of KRF on MAE. "CV" indicates the value of KRF selected by cross-validation.

we pick a training sample with the closest ground truth orientation and show them in Fig. 3.9 in order of the orientation. As can be seen, the continuous annotations capture smooth transitions of the orientations even though the annotations are done solely from 2D images.

In Fig. 3.10, we show the histogram of the orientations on the training data. The orientations are highly imbalanced, thus the sample weighting method discussed in Sec. 3.2.7 is needed.

3.3.3.3 Image features

Since many of the images in the dataset are gray scale images, we first convert all the color images to gray scale images. Then for each image, we extract the HOG features from three different scales and reduce the dimensionality to 2,000 by PCA, preserving

98.8% of the energy.

3.3.3.4 Performance measure

We evaluate the performance of the proposed methods by three measures. The first one is Mean Absolute Error (MAE) of angular distance, $d_{\text{angle}}((t_1, t_2) = \min_{k\in\{0,\pm1,\dots\}} |t_1 - t_2 + 360k|$. The second and third measures, Accuracy-22.5° and Accuracy-$45°$, are defined as the ratio of samples whose predicted orientation is within $22.5°$ and $45°$ from the ground truth, respectively. We argue that Accuracy-22.5° and Accuracy-$45°$ are more practical measure than MAE as oftentimes we have an acceptable error, depending on the applications, and would like to know how likely the predictor can produce the acceptable predictions. Errors larger than the acceptable error are penalized equally. On the other hand, by definition, MAE is strongly influenced by large errors. In the experiments, we observe that many of the failure cases are due to the flipping errors ($\simeq 180°$) which makes MAE less reliable. Thus, our primary evaluation criterion in this work is Accuracy-$22.5°$. We also use Accuracy-$22.5°$ as a criterion for parameter determination.

3.3.3.5 Evaluated methods

We evaluate the performance of AKRF, AKRF-W and AKRF-VW. We also compare the regression forest with BRF and Extremely Randomized Trees algorithm [Geurts et al., 2006] (referred to as ERT) optimizing the same objective function (Eq. 3.17). Instead of using a random subset of the training data, ERT always uses all the training samples but chooses the threshold completely randomly. Note that in Baltieri et al. [2012], a classification version of the ERT achieves the best performance.

For all the tree-based methods, the number of the trees in the forest is set to 100. The ratio of randomly selected samples, $\beta \in (0, 1.0)$, is determined based on the Accuracy-$22.5°$ on the validation set, except for ERT. For BRF and ERT, additional randomness is enforced by considering only a subset of input feature indexes at each node splitting. We determine the ratio of randomly selected feature indexes based on the Accuracy-$22.5°$ on the validation set.

3.3.3.6 Additional baseline methods

As additional baseline methods, we train the ϵ-Support Vector Regression (ϵ-SVR) [Drucker et al., 1996] with Gaussian kernel and Kernel Partial Least Squares Regression (Kernel PLS) [Rosipal and Trejo, 2001] with Gaussian kernel. Since both methods cannot directly handle circular outputs, as was done for the car direction estimation task, we convert each orientation to a point on a unit circle before training and convert them back to the angle during testing time. All the parameters are determined based on Accuracy-$22.5°$ on the validation set.

3.3.3.7 Results

In Table 3.4, we show the results of the proposed methods as well as the baseline methods. We also show the performance of humans computed from all the annotations. The proposed methods significantly outperforms BRF, ERT, Kernel PLS and ϵ-SVR. The AKRF-W improves Accuracy-$22.5°$ of the AKRF by 4.1%. The AKRF-VW improves

Accuracy-22.5° of AKRF by 8.7% and that of AKRF-W by 4.4%. In Fig. 3.11, we plot the change of the accuracy by varying the threshold. Since none of the methods perform as well as humans, there is still a large room for improvement.

It is worth noting that BRF and ERT perform very poorly on this task, even though ERT is the best performing method for the discrete orientation estimation task on the same dataset as reported in Baltieri et al. [2012]. This result indicates that node splitting based on a single feature dimension is not efficient for regression tasks. A similar observation can be made in previous two experiments (Tables.3.1 and 3.3).

In Table 3.5, we summarize previously reported results on the same dataset; however, the original discrete annotations are used by all the previous results for both training and evaluation, thus the performances are measured differently. Accuracy8 is a percentage of correctly predicted samples using the original 8 discrete orientation classes. Accuracy4 also uses the same 8 orientation classes but consider the two adjacent orientations as being correct. Essentially, Accuracy-22.5° and Accuracy-45° will be equivalent to Accuracy8 and Accuracy4 respectively if using the the discrete annotations.

Finally, Fig. 3.12 shows some qualitative results from the AKRF-VW. The results in the last row are failure cases. Note that color information is not used in computing the image features since many of the images in the training set are gray scale images. It would be interesting to see if the use of color information helps to resolve some of the confusion, provided a set of color images for training.

[1]Although numbers are reported in Tao and Klette [2013], we omit them from the table for the following reason. Their method is not capable of predicting eight orientations. In computing the accuracy for the eight orientation setting, they assume that NE, SE, SW, and NW orientations are correctly estimated if the predicted orientation is their adjacent orientations. Thus, the number reported are not comparable to other

3.4 Conclusions

In this chapter, we proposed a novel node splitting algorithm for regression tree training. Unlike previous works, the proposed method does not rely on a trial-and-error process to find the best splitting rules from a predefined set of rules, providing more flexibility to the model. Combined with the regression forest framework, our methods achieve competitive results on head pose estimation, car direction estimation and newly introduced continuous pedestrian orientation estimation tasks. Further improvement is achieved by the proposed sample weighting technique and voting-based ensemble method based on the mean shift algorithm.

3.5 Acknowledgments

The work presented in this chapter was conducted under the support by a MURI grant from the US Office of Naval Research under N00014-10-1-0934.

results.

Table 3.3: Car direction estimation results on the EPFL Multi-view Car Dataset

Method	MAE (°) 90-th percentile	MAE (°) 95-th percentile	MAE (°)
AKRF-VW	6.76	15.65	23.81
AKRF-W	7.42	15.94	24.06
AKRF	7.73	16.18	24.24
KRF	8.32	16.76	24.80
BRF	23.97	30.95	38.13
Kernel PLS	16.86	21.20	27.65
ϵ-SVR	17.38	22.70	29.41
Fenzi et al. [2015]	N/A	N/A	13.6
He et al. [2014]	N/A	N/A	15.8
Fenzi and Ostermann [2014]	12.67	17.77	23.38
Yang et al. [2014]	N/A	N/A	24.1
Zhang et al. [2013]	N/A	N/A	24.0
Fenzi et al. [2013]	14.51	22.83	31.27
Torki and Elgammal [2011]	19.4	26.7	33.98
Ozuysal et al. [2009]	N/A	N/A	46.48

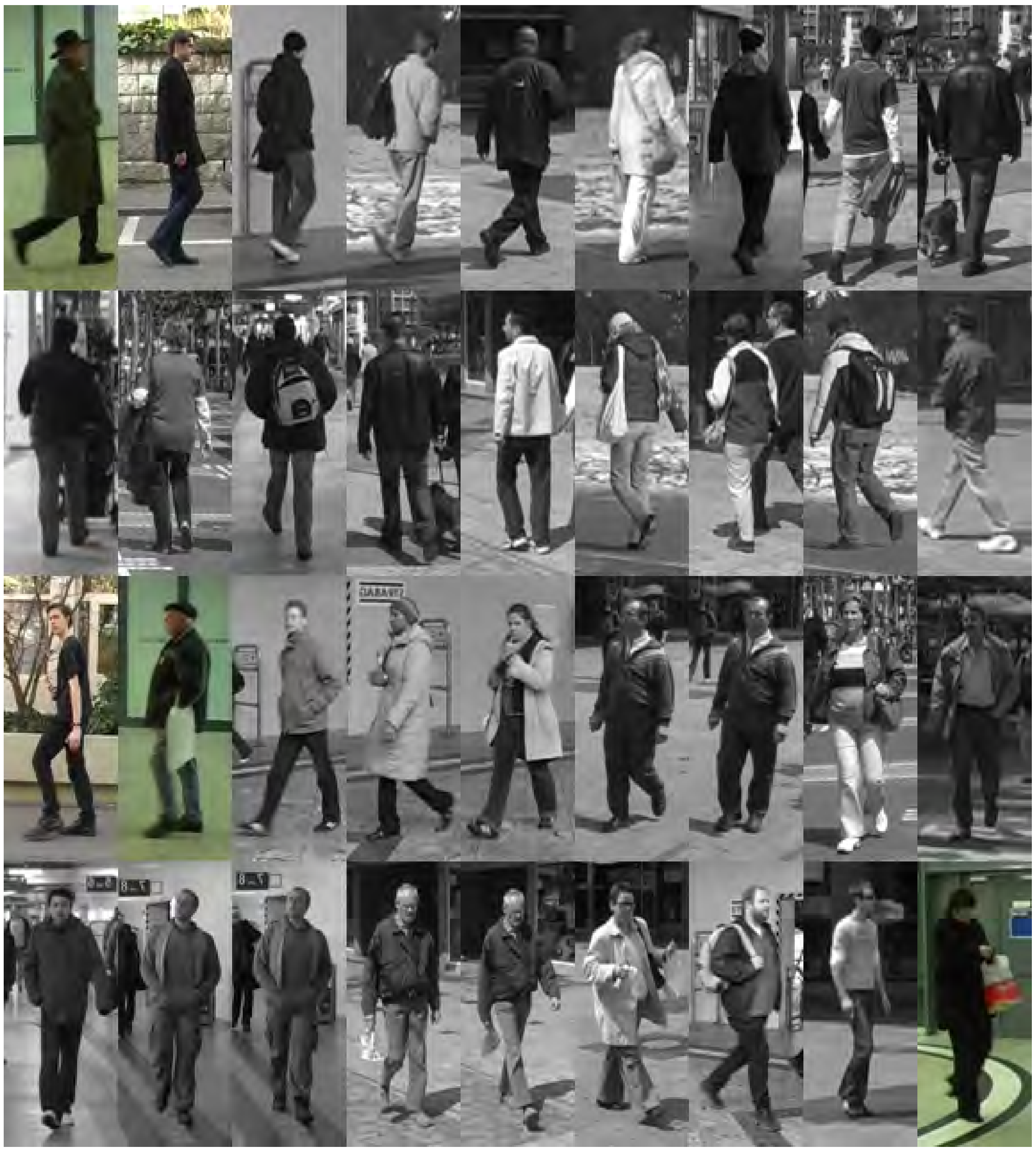


Figure 3.9: Training samples for representative orientation angles are shown. For each angle in $\{0°, 10°, \ldots, 350°\}$, a training sample with the closest ground truth is selected. The left-top image corresponds to $0°$ and the right-bottom one corresponds to $350°$. The continuous annotations capture smooth transition of the body orientations even though the annotations are done solely from the 2D images.

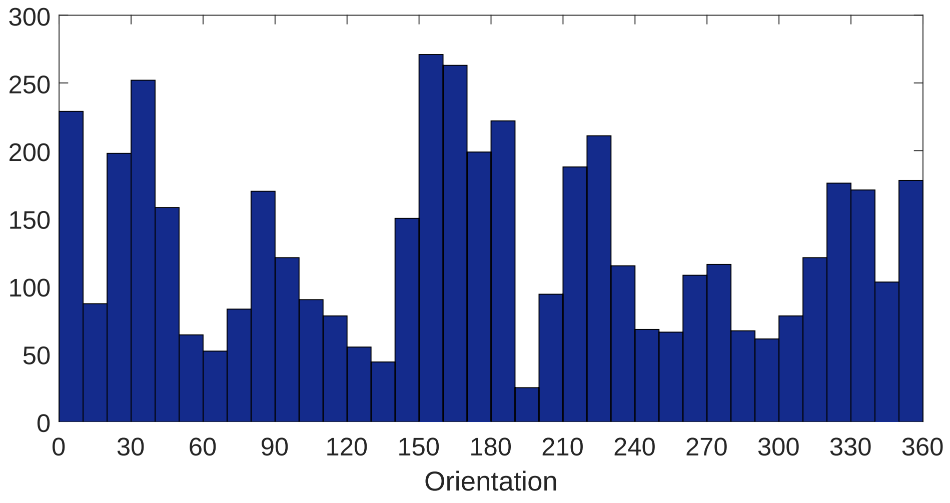


Figure 3.10: TUD Multiview Pedestrians Dataset: a histogram obtained from the orientations on the training data. The orientations are higly imbalanced.

Table 3.4: Continuous pedestrian orientation estimation: Accuracy-$22.5°$, Accuracy-$45°$ and Mean Absolute Error in degree are shown for AKRF-VW and all baseline methods.

Method	Accuracy-$22.5°$	Accuracy-$45°$	MAE ($°$)
AKRF-VW	68.6	78.0	34.7
AKRF-W	65.7	76.1	35.9
AKRF	63.1	76.1	36.1
Kernel PLS	49.8	71.5	36.5
ϵ-SVR	48.2	69.6	39.1
BRF	32.4	55.3	54.7
ERT	31.1	56.0	50.3
Human	90.7	99.3	9.1

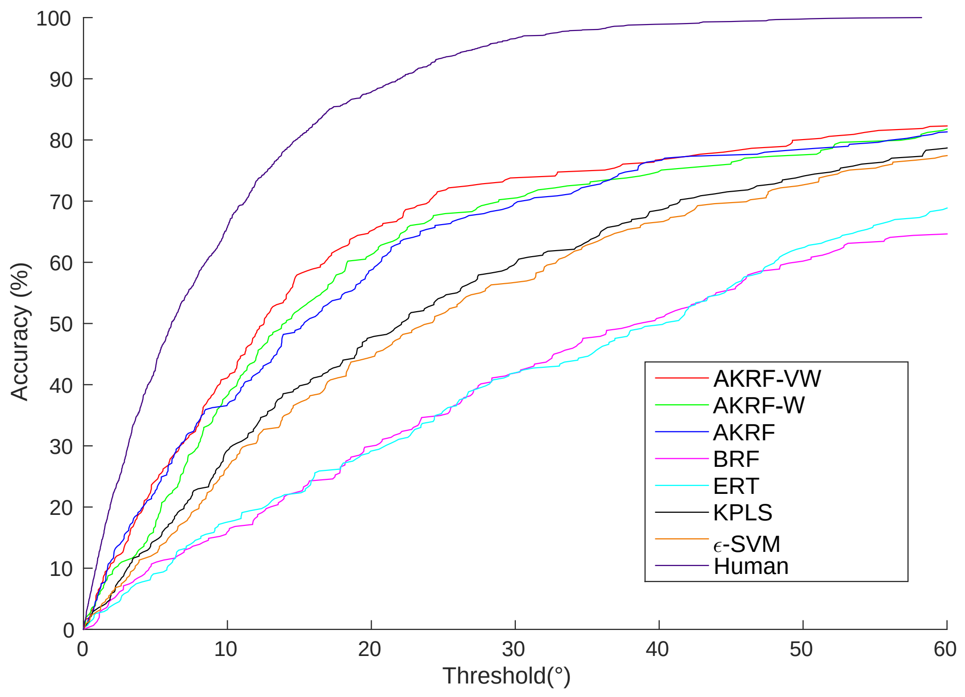


Figure 3.11: Change of the accuracy by varying threshold on the pedestrian orientation estimation task

Table 3.5: Results of previously proposed approaches. Note that the performance is measured differently for the previous approaches as the original discrete annotations are used. See text for the details. All the results are on the TUD Multiview Pedestrians Dataset [Andriluka et al., 2010]

Method	Accuracy8	Accuracy4
Baltieri et al. [2012] - AWG	65	83
Baltieri et al. [2012] - Max	58	76
Chen et al. [2011]	55	76
Tao and Klette [2013] - FourD1	N/A[1]	69
Tao and Klette [2013] - FourPedRD2	N/A[1]	71
Andriluka et al. [2010] - Max	31	N/A
Andriluka et al. [2010] - SVM	42	70
Andriluka et al. [2010] - SVM-adj	35	76

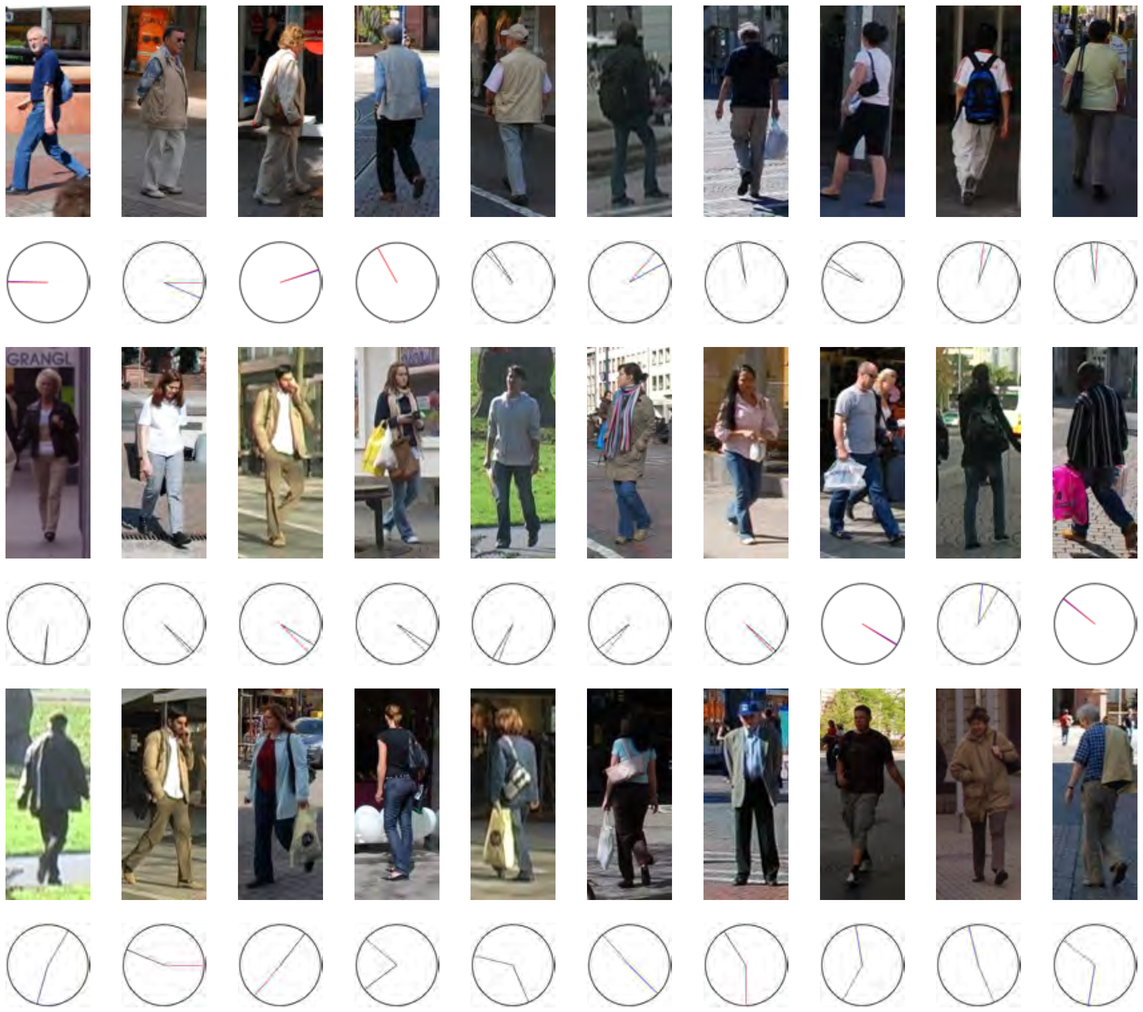


Figure 3.12: Example results from AKRF-VW. Red lines indicate ground truth orientations. Blue lines indicate predicted orientations. The first two rows show successful cases while the last row shows failure cases. Note that many of the failure cases are due to the flipping errors.

Chapter 4: Fashion Apparel Detection: the Role of Deep Convolutional Neural Network and Pose-dependent Priors

In this work, we propose a method to detect fashion apparels a person in an image is wearing or holding. The types of fashion apparels include hat, bag, skirt, etc. Fashion apparel spotting has recently gained considerable traction. A major reason is due to a variety of applications that a reliable fashion item spotter can enable. For instance, spotted fashion items can be used to retrieve similar or identical fashion items from an online inventory.

Unlike most prior works on fashion apparel spotting which address the task as a specialization of the semantic segmentation to the fashion domain, we address the problem as an object detection task where the detection results are given in the form of bounding boxes. Detection-based spotters are more suitable as (a) bounding boxes suffice to construct queries for the subsequent visual search, (b) it is generally faster and have lower memory footprint than semantic segmentation, (c) large scale pixel-accurate training data is extremely hard to obtain, while it is much easier to get training data as bounding boxes, and (d) detection is done at instance-level while semantic segmentation does not differentiate multiple instances belonging to the same class. To the best of our knowledge, our work is the first detection-based (as opposed to segmentation-based) fashion item spotting method.

Although any existing object detection method can possibly be applied, the fashion apparel detection task poses its own challenges such as (a) deformation of clothing is large, (b) some fashion items classes are extremely similar to each other in appearance (e.g., skirt and bottom of short dress), (c) the definition of fashion item classes can be ambiguous (e.g., pants and tights), and (d) some fashion items are very small (e.g., belt, jewelry). In this work, we address some of these challenges by incorporating state-of-the-art object detectors with various domain specific priors such as pose, object shape and size.

The state-of-the-art object detector we employ in this work is R-CNN Girshick et al. [2014], which combines object proposals with a Convolutional Neural Network Fukushima [1980], Lecun et al. [1998]. The R-CNN starts by generating a set of object proposals in the form of bounding boxes. Then image patches are extracted from the generated bounding boxes and resized to a fixed size. The Convolutional Neural Network pretrained on a large image database for the image classification task is used to extract features from each image patch. SVM classifiers are then applied to each image patch to determine if the patch belongs to a particular class. The R-CNN is suitable for our task as it can detect objects with various aspect ratios and scales without running a scanning-window search, reducing the computational complexity as well as false positives.

It is evident that there are rich priors that can be exploited in the fashion domain. For instance, handbag is more likely to appear around the wrist or hand of the person holding them, while shoes typically occur near feet. The size of items are typically proportional to the size of a person. Belts are generally elongated. One of our contributions is to integrate these domain-specific priors with the object proposal-based detection method. These priors are learned automatically from the training data.

We evaluate the detection performance of our algorithm on the previously introduced Fashionista dataset Yamaguchi et al. [2012] using a newly created set of bounding box annotations. We convert the segmentation results of state-of-the-art fashion item spotter into bounding box results and compare with the results of the proposed method. The experiments demonstrate that our detection-based approach outperforms the state-of-the art segmentation-based approaches in mean Average Precision criteria.

The rest of the chapter is organized as follows. Section 4.1 summarizes related work in fashion item localization. Our proposed method is detailed in Section 4.2 where we start with object proposal, followed by classification of these proposals using a combination of generative and discriminative approaches. Section 4.3 validates our approach on the popular Fashionista Dataset Yamaguchi et al. [2012] by providing both qualitative and quantitative evaluations. Finally, Section 4.4 contains closing remarks.

4.1 Related Work

The first segmentation-based fashion spotting algorithm for general fashion items was proposed by Yamaguchi et al. [2012] where they introduce the Fashionista Dataset and utilize a combination of local features and pose estimation to perform semantic segmentation of a fashion image. In Yamaguchi et al. [2013], the same authors followed up this work by augmenting the existing approach with data driven model learning, where a model for semantic segmentation was learned only from nearest neighbor images from an

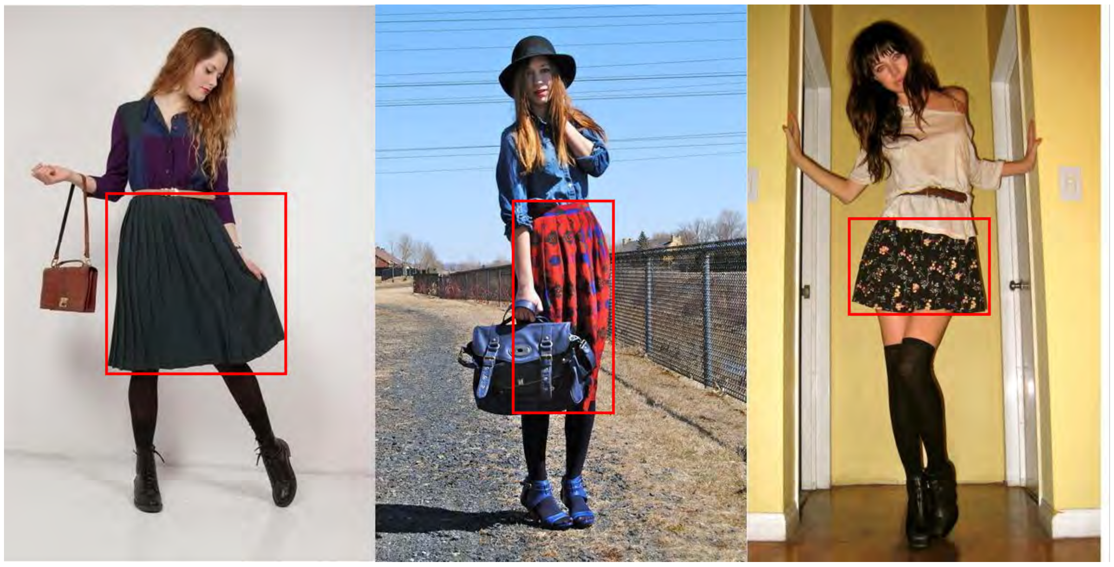


Figure 4.1: Bounding boxes of three different instances of "skirt" class. The aspect ratios vary significantly even though they are from the same object class.

external database. Further, this work utilizes textual content along with image information. The follow up work reported considerably better performance than the initial work. We report numbers by comparing to the results accompanying these two papers.

Apart from the above two works, Hasan and Hogg [2010] also proposed a segmentation-based approach aimed at assigning a unique label from "Shirt", "Jacket", "Tie" and "Face and skin" classes to each pixel in the image. Their method is focused on people wearing suits.

There exist several clothing segmentation methods Gallagher and Chen [2008], Hu et al. [2008], Wang and Ai [2011] whose main goal is to segment out the clothing area in the image and types of clothing are not dealt with. In Gallagher and Chen [2008], a clothing segmentation method based on graph-cut was proposed for the purpose of identity recognition. In Hu et al. [2008], similar to Gallagher and Chen [2008], a graph-cut based method was proposed to segment out upper body clothing. Wang and Ai [2011] presented a method for clothing segmentation of multiple people. They propose to model and utilize the blocking relationship among people.

Several works exist for classifying types of upper body clothing Bossard et al. [2012], Shen et al. [2014], Chen et al. [2012]. In Shen et al. [2014], a structured learning technique for simultaneous human pose estimation and garment attribute classification is proposed. The focus of this work is on detecting attributes associated with upper body clothing, such as collar types, color, types of sleeves, etc. Similarly, an approach for detecting apparel types and attributes associated with upper bodies was proposed in Bossard et al. [2012], Chen et al. [2012]. Since localization of upper body clothing is essentially solved by upper body detectors and detecting upper body is relatively easy, the focus of the above methods has been on subsequent classification stage. On the other hand, we focus on a variety of fashion items with various sizes which cannot be easily detected even with perfect pose information.

Yang and Yu [2011] proposed a real-time clothing recognition method in surveillance settings. They first obtain foreground segmentation and classify upper bodies and lower bodies separately into a fashion item class. In Bourdev et al. [2011], a poselet-based approach for human attribute classification is proposed. In their work, a set of poselet detectors is trained and for each poselet detection, attribute classification is done using SVM. The final results are then obtained by considering the dependencies between different attributes. In Wang and Cottrell [2015], recognition of social styles of people in an image is addressed by Convolutional Neural Network applied to each person in the image as well as the entire image.

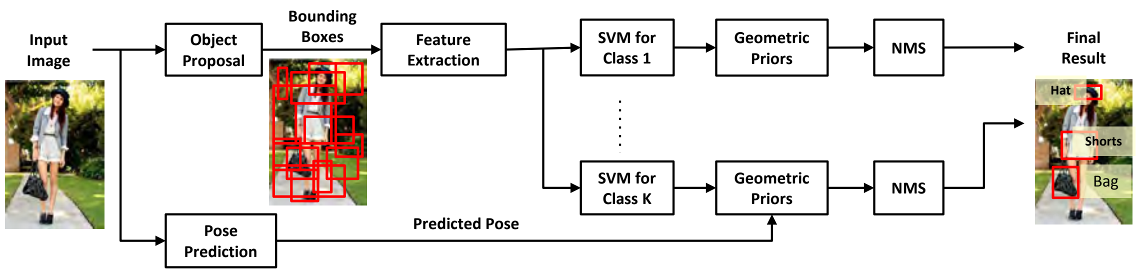


Figure 4.2: Overview of the proposed algorithm for testing stage. Object proposals are generated and features are extracted using Deep CNN from each object proposal. An array of 1-vs-rest SVMs are used to generate appearance-based posteriors for each class. Geometric priors are tailored based on pose estimation and used to modify the class probability. Non-maximum suppression is used to arbitrate overlapping detections with appreciable class probability.

4.2 Proposed Method

The aim of the proposed method is to detect fashion items in a given image, worn or carried by a single person. The proposed method can be considered as an extension of the recently proposed R-CNN framework Girshick et al. [2014], where we utilize various priors on location, size and aspect ratios of fashion apparels, which we refer to as geometric priors. Specifically for location prior, we exploit strong correlations between the pose of the person and location of fashion items. We refer to this as pose context. We combine these priors with an appearance-based posterior given by SVM to obtain the final posterior density function. Thus, the model we propose is a hybrid of discriminative and generative models.

The recognition pipeline of the proposed algorithm for the testing stage is shown in

Figure 4.2. First, the pose of the person is estimated by an off-the-shelf pose estimator. Then, a set of candidate bounding boxes is generated by an object proposal algorithm. Image features are extracted from the contents of each bounding box. These image features are then fed into a set of SVMs with a sigmoid function to obtain an appearance-based posterior for each class. By utilizing the geometric priors, the final posterior probability for each class is computed for each bounding box. The results are then filtered by a standard non-maximum suppression method by Felzenszwalb et al. [2010]. We present the details of each component below.

4.2.1 Object Proposal

Object detection based on a sliding window strategy has been a standard approach Felzenszwalb et al. [2010], Dalal and Triggs [2005], Viola and Jones [2001], Bourdev and Malik [2009] where object detectors are exhaustively run on all possible locations and scales of the image. To accommodate the deformation of objects, most recent works detect a single object by a set of part-specific detectors and allow the configurations of the parts to vary. Although a certain amount of deformation is accommodated, possible aspect ratios considered are still limited and the computation time increases linearly as the number of part detectors increases.

In our task, the intra-class shape variation is large. For instance, as shown in Figure 4.1, bounding boxes of three instances from the same “skirt” class have very different aspect ratios. Thus, for practical use, detection methods which can accommodate various deformations without significant increase in computation time are required.

In order to address these issues, we use object proposal algorithms Uijlings et al. [2013], Arbelaez et al. [2014] employed by state-of-the-art object detectors (i.e., R-CNN Girshick et al. [2014]). The object proposal algorithm generates a set of candidate bounding boxes with various aspect ratios and scales. Each bounding box is expected to contain a single object and the classifier is applied only at those candidate bounding boxes, reducing the number of false positives. For the classification step, an image patch within a bounding box is resized to a predefined size and image features are extracted. Since feature computation is done only at the generated bounding boxes, the computation time is significantly reduced while allowing various aspect ratios and scales. In this work, we employ Selective Search (SS) Uijlings et al. [2013] as the object proposal method.

4.2.2 Image Features by CNN

Our framework is general in terms of the choice of image features. However, recent results in the community indicate that features extracted by Convolutional Neural Network (CNN) Fukushima [1980], Lecun et al. [1998] with many layers perform significantly better than the traditional hand-crafted features such as HOG and LBP on various computer vision tasks Farabet et al. [2012], Krizhevsky et al. [2012], Sermanet et al. [2013], Zhang et al. [2014]. However, in general, to train a good CNN, a large amount of training data is required.

Several papers have shown that features extracted by CNN pre-trained on a large image dataset are also effective for other vision tasks. Specifically, a CNN trained on ImageNet database Deng et al. [2009] is used for various related tasks as a feature extractor and achieve impressive performance Donahue et al. [2014], Razavian et al. [2014]. In this work, we use CaffeNet Jia et al. [2014] trained on ImageNet dataset as the feature extractor. We use a 4096 dimensional output vector from the second last layer (fc7) of CaffeNet as a feature vector.

4.2.3 SVM training

For each object class, we train a linear SVM to classify an image patch as positive or negative. The training patches are extracted from the training data with ground-truth bounding boxes. The details of the procedure are described in Section 4.3.2.

4.2.4 Probabilistic formulation

We formulate a probabilistic model to combine outputs from the SVM and the priors on the object location, size and aspect ratio (geometric priors) into the final posterior for each object proposal. The computed posterior is used as a score for each detection.

Let $B = (x_1, y_1, x_2, y_2)$ denote the bounding box coordinates of an object proposal. Let f denote the image features extracted from B. We denote by $c = (l_x, l_y)$ the location of the bounding box center, where $l_x = (x_1 + x_2)/2$ and $l_y = (y_1 + y_2)/2$. We denote by $\mathrm{a} = \log((y_2 - y_1)/(x_2 - x_1))$, the log aspect ratio of the bounding box and by $\mathrm{r} = \log((y_2 - y_1) + (x_2 - x_1))$ the log of half the length of the perimeter of the bounding box. We refer to c, a and r as geometric features.

Let Y denote a set of fashion item classes and $y_{\mathrm{z}} \in \{+1, -1\}$ where $z \in Y$, denote a binary variable indicating whether or not B contains an object belonging to z. Let

$\mathbf{t} = (t_1, \ldots, t_K) \in \mathrm{R}^{2\times K}$ denote pose information, which is a set of K 2D joint locations on the image. The pose information serves as additional contextual information for the detection.

We introduce a graphical model describing the relationship between the above variables and define a posterior of y_z given f, $\mathbf{t}$, c, a and r as follows:

$$p(y_z|f, c, \mathrm{a}, \mathrm{r}, \mathbf{t}) \propto p(y_z|f)p(c|y_z, \mathbf{t})p(\mathrm{a}|y_z)p(\mathrm{r}|y_z, \mathbf{t}) \tag{4.1}$$

Here we assume that $p(\mathbf{t})$ and $p(f)$ are constant. The first term on the RHS of Eq. 4.1 defines the appearance-based posterior and the following terms are the priors on the geometric features. For each object proposal, we compute $p(y_z = 1|f, c, \mathrm{a}, \mathrm{r}, \mathbf{t})$ and use it as a detection score. The introduced model can be seen as a hybrid of discriminative and generative models. In the following sections, we give the details of each component.

4.2.5 Appearance-based Posterior

We define an appearance based posterior $p(y_z = 1|f)$ as

$$p(y_z = 1|f) = \mathrm{Sig}(\mathrm{w}_z^T f; \lambda_z) \tag{4.2}$$

where w_z is an SVM weight vector for the class z and λ_z is a parameter of the sigmoid function $\mathrm{Sig}(\mathrm{x}; \lambda_z) = 1/(1 + \exp(-\lambda_z \mathrm{x}))$. The parameter λ_z controls the shape of the sigmoid function. We empirically find that the value of λ_z largely affects the performance. We optimize λ_z based on the final detection performance on the validation set.

4.2.6 Geometric Priors

Priors on Aspect Ratio and Perimeter

The term $p(\mathrm{r}|y_z = 1, \mathbf{t})$ is the prior on perimeter conditioned on the existence of an object from class z and pose $\mathbf{t}$. Intuitively, the length of perimeter r, which captures the object size, is useful for most of the items as there is a typical size for each item. Also, r is generally proportional to the size of a person. The size of the person can be defined using $\mathbf{t}$ in various ways. However, in this work, since the images in the dataset we use for experiments are already normalized such that the size of the person is roughly same, we assume $p(\mathrm{r}|y_z = 1, \mathbf{t}) = p(\mathrm{r}|y_z = 1)$.

The term $p(\mathrm{a}|y_z = 1)$ is the prior on the aspect ratio of object bounding box conditioned on the existence of an object from class z. Intuitively, the aspect ratio a is useful for detecting items which have a distinct aspect ratio. For instance, the width of waist belt and glasses are most likely larger than their height. To model both $p(\mathrm{a}|y_z = 1)$ and $p(\mathrm{r}|y_z = 1)$, we use a 1-D Gaussian fitted by standard maximum likelihood estimation.

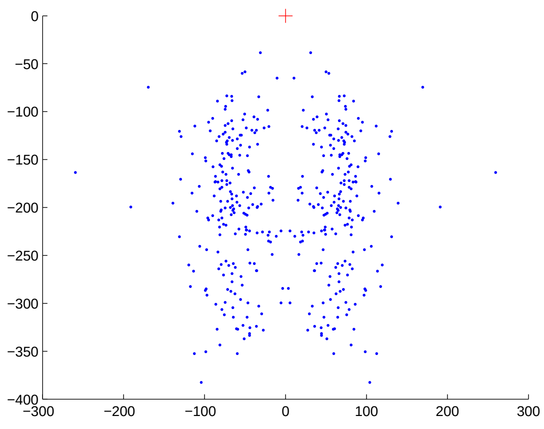
Pose dependent prior on the bounding box center

We define a pose dependent prior on the bounding box center as

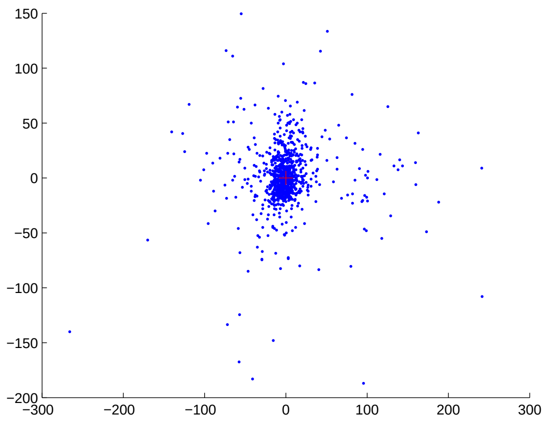
$$p(c|y_z = 1, \mathbf{t}) = \Pi_{k \in T_z} p(l_x, l_y | y_z = 1, t_k) \tag{4.3}$$

$$= \Pi_{k \in T_z} p((l_x, l_y) - t_k | y_z = 1) \tag{4.4}$$

where T_z is a set of joints that are informative about the bounding box center location of the object belonging to the class z. The algorithm to determine T_z for each fashion item



(a) Bag - Neck



(b) Left Shoe - Left Ankle

Figure 4.3: Distributions of relative location of item with respect to location of key joint. Key joint location is depicted as a red cross. (a) distribution of relative location of bag with respect to neck is multi-modal. (b) locations of left shoe and left ankle are strongly correlated and the distribution of their relative location has a single mode. See Section 4.2.6 for details.

class z will be described shortly. Each $p((l_x, l_y) - t_k | y_z = 1)$ models the relative location of the bounding box center with respect to the k-th joint location.

Intuitively, the locations of fashion items and those of body joints have strong correlations. For instance, the location of hat should be close to the location of head and thus, the distribution of their offset vector, $p((l_x, l_y) - t_{\text{Head}} | y_{\text{Hat}} = 1)$ should have a strong peak around t_{Head} and relatively easy to model. On the other hand, the location of left hand is less informative about the location of the hat and thus, $p((l_x, l_y) - t_{\text{Lefthand}} | y_{\text{Hat}} = 1)$ has a complex distribution which is difficult to model accurately. Thus, it is beneficial to use for each fashion item only a subset of body joints that have strong correlations with the location of that item.

The relative location of the objects with respect to the joints can be most faithfully modeled as a multimodal distribution. For instance, bags, purses and wallets are typically carried on either left or right hand side of the body, thus generating multimodal distributions. To confirm this claim, In Figure 4.3, we show a plot of $(l_x, l_y) - t_{\text{Neck}}$ of “Bag” and a plot of $(l_x, l_y) - t_{\text{LeftAnkle}}$ of “Left Shoe” obtained from the dataset used in our experiments. As can be seen, $p((l_x, l_y) - t_{\text{Neck}}|y_{\text{Bag}} = 1)$ clearly follows a multimodal distribution while $p((l_x, l_y) - t_{\text{LeftAnkle}}|y_{\text{LeftShoe}} = 1)$ has a unimodal distribution. Depending on the joint-item pair, it is necessary to automatically choose the number of modes.

To address the challenges raised above, we propose an algorithm to automatically identify the subset of body joints T_z and learn a model. For each pair of a fashion item z and a body joint k, we model $p((l_x, l_y) - t_k|y_z = 1)$ by a Gaussian mixture model (GMM) and estimate the parameters by the EM-algorithm. We determine the number of GMM components based on the Bayesian Information Criteria Kashyap [1977], Schwarz [1978] to balance the complexity of the model and fit to the data. To obtain T_z for item z, we pick the top 2 joints whose associated GMM has larger likelihood. This way, for each item, body joints which have less scattered offsets are automatically chosen. The selected joints for each item will be shown in the next section.

4.3 Experiments

4.3.1 Dataset

To evaluate the proposed algorithm, we use the Fashionista Dataset which was introduced by Yamaguchi et al. [2012] for pixel-level clothing segmentation. Each image in this dataset is fully annotated at pixel level, *i.e.* a class label is assigned to each pixel. In addition to pixel-level annotations, each image is tagged with fashion items presented in the images. In Yamaguchi et al. [2013], another dataset called Paper Doll Dataset including 339,797 tagged images is introduced and utilized to boost performance on the Fashionista Dataset. Our method does not use either associated tags or the Paper Doll Dataset. We use the predefined training and testing split for the evaluation (456 images for training and 229 images for testing) and take out 20% of the training set as the validation set for parameter tuning.

In the Fashionista Dataset, there are 56 classes including 53 fashion item classes and three additional non-fashion item classes (hair, skin and background.) We first remove some classes that do not appear often in the images and those whose average pixel size is too small to detect. We then merge some classes which look very similar. For instance, there are “bag”, “Purse” and “Wallet” classes but the distinction between those classes is visually vague, thus we merge those three classes into a single ”Bag” class. We also discard all the classes related to footwear such as “sandal” and “heel’ and instead add “left shoe” and “right shoe” classes which include all types of footwear. It is intended that, if needed by a specific application, a sophisticated fine-grained classification method can be applied as a post-processing step once we detect the items. Eventually, we obtain 10 new classes where the occurrence of each class is large enough for training the detector and the appearance of items in the same class is similar. The complete definition of the new ten classes and some statistics are shown in Table 4.1.

We create ground-truth bounding boxes based on pixel-level annotations under the new definition of classes. For classes other than “Left shoe” and “Right shoe”, we define a ground-truth bounding box as the one that tightly surrounds the region having the corresponding class label. For “Left shoe” and “Right shoe” classes, since there is no distinction between right and left shoes in the original pixel-level annotations, this automatic procedure cannot be applied. Thus, we manually annotate bounding boxes for “Right shoe” and “Left shoe” classes. These bounding box annotation are available at the author’s website to facilitate further research on fashion apparel detection.

Our framework is general in the choice of pose estimators. In this work, we use pose estimation results provided in the Fashionista Dataset, which is based on Yang and Ramanan [2011]. There are 14 key joints namely head, neck, left/right shoulder, left/right elbow, left/right wrist, left/right hip, left/right knee and left/right foot.

In Table 4.1, we show the first and second key body joints that are selected by the proposed algorithm. Interestingly, for “Pants”, “Shorts” and “Skirt”, left hip and right hip are selected but for “Tights”, left knee and right knee are selected instead.

4.3.2 Detector Training

We create image patches for training the detector by cropping the training images based on corresponding ground-truth bounding box. Before cropping, we enlarge the bounding boxes by a scale factor of 1.8 to include the surrounding regions, thus providing contextual information. Note that we intentionally make the contextual regions larger than Girshick et al. [2014] as contextual information would be more important when detecting small objects like fashion items we consider in this work. The cropped image patches are then resized to the size of the first layer of CaffeNet (227 by 227 pixels). To increase the number of training patches, we run the object proposal algorithm on the training images and for each generated bounding box, we compute the intersection over union (IoU) with the ground-truth bounding boxes. If the IoU is larger than 0.5 for a particular class, we use the patch as an additional training instance for that class. If IoU is smaller than 0.1 with ground-truth bounding boxes of all the classes, we use it as a training instance for the background class. We also obtain the training patches for the background class by including image patches from ground-truth bounding boxes of the classes which we do not include in our new ten classes.

The number of training patches for each class obtained is shown in Table 4.3. From the obtained training patches, we train a set of linear SVMs, each of which is trained by using instances in a particular class as positive samples and all instances in the remaining classes as negative samples. The parameters of SVMs are determined using the validation set.

4.3.3 Baseline Methods

Since fashion apparel detection has not been previously addressed, there is no existing work proposed specifically for this task. Thus, we convert the pixel-level segmentation results of Yamaguchi et al. [2012] and Yamaguchi et al. [2013] to bounding boxes and use their performance as baselines. To obtain bounding boxes from segmentation results, we use the same procedure we use to generate ground-truth bounding boxes from the ground-truth pixel-level annotations. Note that we exclude "Left shoe" and "Right shoe" from the comparison since in their results, there is no distinction between left and right shoes.

4.3.4 Results

We first evaluate the performance of the object proposal methods in terms of precision and recall. Here, precision is defined as the number of object proposals which match the ground-truth bounding boxes regardless of class, divided by the total number of object proposals. Specifically, we consider each object proposal as correct if $\mathrm{IoU} \geq 0.5$ for at least one ground-truth bounding box. We compute recall for each class by the number of ground-truth bounding boxes which have at least one corresponding object proposal, divided by the total number of ground-truth bounding boxes.

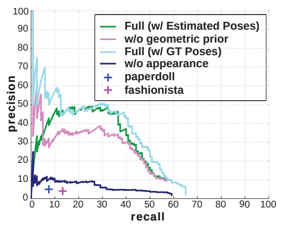
In Table 4.4, we show the precision, recall and the average number of object proposals per image. We tune the parameters of both object proposal algorithms to retain high recall so that it will not miss too many true objects. Although it results in low precision, false positives are reduced in the subsequent classification stage.

We evaluate the performance of the detection methods using the average precision (AP) computed from the Precision-Recall curves. In Table 4.2, we report the performance of the proposed framework with three different settings, "Full" represents our complete method using both geometric priors and appearance-based posterior, "w/o geometric prior" represents a method which excludes the geometric priors from "Full" and "w/o appearance" is a method which excludes appearance-based posterior from "Full".

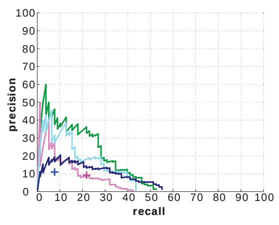
From the comparison between "Full" and "w/o geometric prior", it is clear that incorporating geometric priors significantly improves the performance (35.8% improvement for mAP). This result indicates the effectiveness of the geometric priors in the fashion item detection task.

In Figure 4.4 we show the precision-recall curves of the proposed methods with various settings as well as precision-recall points of the baseline methods. In the figures, "paperdoll" refers to the results of Yamaguchi et al. [2013] and "fashionista" refers to Yamaguchi et al. [2012]. Except for "Pants", our complete method outperforms the baselines with a large margin. Note that "paperdoll" Yamaguchi et al. [2013] uses a large database of tagged fashion images as additional training data.

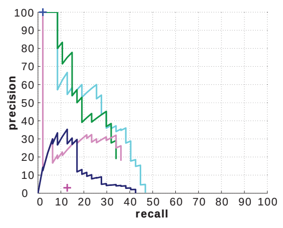
In Figure 4.5, we show some qualitative results. Figure 4.6 shows sample images where our approach makes mistakes. We see that fashion apparel detection has its own unique challenges. First of all, even with our new fashion item classes, some fashion items are visually very similar to each other. For example, "Tights" and "Pants" can look very similar since both items can have a variety of colors. The only distinguishable cue might be how tight it is, which is quite challenging to capture. Another example is "Skirt" and bottom half of a dress. Both items have extremely similar appearance. The only difference



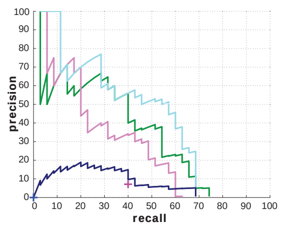
(a) Bag



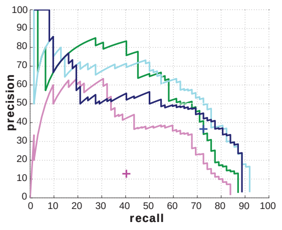
(b) Belt



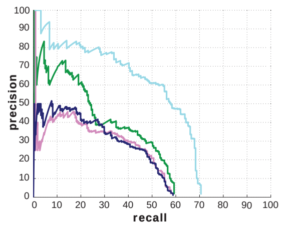
(c) Glasses



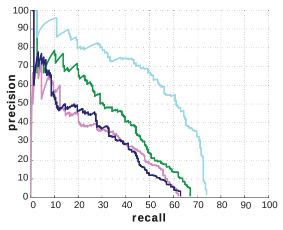
(d) Hat



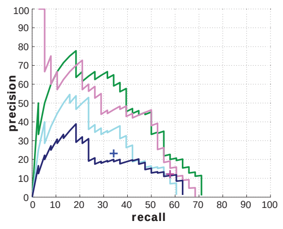
(e) Pants



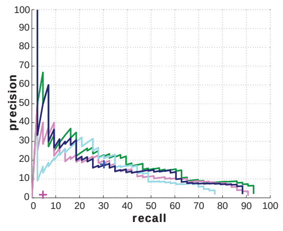
(f) Left shoe



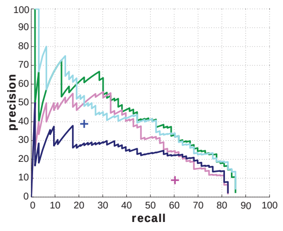
(g) Right shoe



(h) Shorts



(i) Skirt



(j) Tights

Figure 4.4: Precision-Recall curves for each fashion category. Our full method outperforms the baseline method (shown by cross) with a large margin (sometimes up to 10 times in precision for the same recall), except for "Pants". Note that we do not have results from the baseline methods for "Left shoe" and "Right shoe" as they are newly defined in this work.

is that a dress is a piece of cloth which covers both upper body and lower body and this difference is difficult to detect. Furthermore, “Belt” and “Glasses” are difficult to detect as they are usually very small.

4.4 Conclusion

In this work, we reformulate fashion apparel parsing, traditionally treated as a semantic segmentation task, as an object detection task and propose a probabilistic model which incorporates state-of-the-art object detectors with various geometric priors of the object classes. Since the locations of fashion items are strongly correlated with the pose of a person, we propose a pose-dependent prior model which can automatically select the most informative joints for each fashion item and learn the distributions from the data. Through experimental evaluations, we observe the effectiveness of the proposed priors for fashion apparel detection.

4.5 Acknowledgments

The work presented in this chapter was conducted as part of the summer internship at eBay Research Labs.

New Class	Original Classes	Average Size in Pixel	Average Occurrence per Image	First and Second Key Joints
Bag	Bag, Purse, Wallet	5,644	0.45	Left hip, Right hip
Belt	Belt	1,068	0.23	Right hip, Left hip
Glasses	Glasses, Sunglasses	541	0.16	Head, Neck
Hat	Hat	2,630	0.14	Neck, Right shoulder
Pants	Pants, Jeans	16,201	0.24	Right hip, Left hip
Left Shoe	Shoes, Boots, Heels, Wedges, Flats, Loafers, Clogs, Sneakers, Sandals, Pumps	3,261	0.95	Left ankle, Left knee
Right Shoe		2,827	0.93	Right ankle, Right knee
Shorts	Shorts	6,138	0.16	Right hip, Left hip
Skirt	Skirt	14,232	0.18	Left hip, Right hip
Tights	Tights, Leggings, Stocking	10,226	0.32	Right knee, Left knee

Table 4.1: The definition of new classes, their average size and the average number of occurrence per image are shown. The top 2 key body joints for each item as selected by the proposed algorithm are also shown. See Section 4.3.1 for details.

Methods	**mAP**	**Bag**	**Belt**	**Glasses**	**Hat**	**Pants**	**Left Shoe**	**Right Shoe**	**Shorts**	**Skirt**	**Tights**
Full	**31.1**	**22.5**	**14.2**	**22.2**	**36.1**	**57.0**	**28.5**	**32.5**	**37.4**	**20.3**	**40.6**
w/o geometric priors	22.9	19.4	6.0	13.0	28.9	37.2	20.2	23.1	34.7	15.2	31.7
w/o appearance	17.8	4.3	7.1	7.5	8.9	50.7	20.5	23.4	15.6	18.0	22.3

Table 4.2: Average Precision of each method. “Full” achieves better mAP and APs for all the items than “w/o geometric priors” and “w/o appearance”.

Bag	**Belt**	**Glasses**	**Hat**	**Pants**	**Left shoe**	**Right shoe**	**Shorts**	**Skirt**	**Tights**	**Background**
1,254	318	177	306	853	1,799	1,598	473	683	986	225,508

Table 4.3: The number of training patches generated for each class with Selective Search Uijlings et al. [2013].

Precision (%)	**Recall (%)**											**Avg. # of bounding boxes**
	Avg.	**Bag**	**Belt**	**Glasses**	**Hat**	**Pants**	**Left shoe**	**Right shoe**	**Shorts**	**Skirt**	**Tights**	
1.36	86.7	93.6	69.2	62.5	95.3	93.6	86.6	82.4	93.2	98.8	91.2	1073.4

Table 4.4: Precision, recall and the average number of generated bounding boxes per image. Note that it is important to have high recall and not necessarily precision so that we will not miss too many true objects. Precision is controlled later by the classification stage.

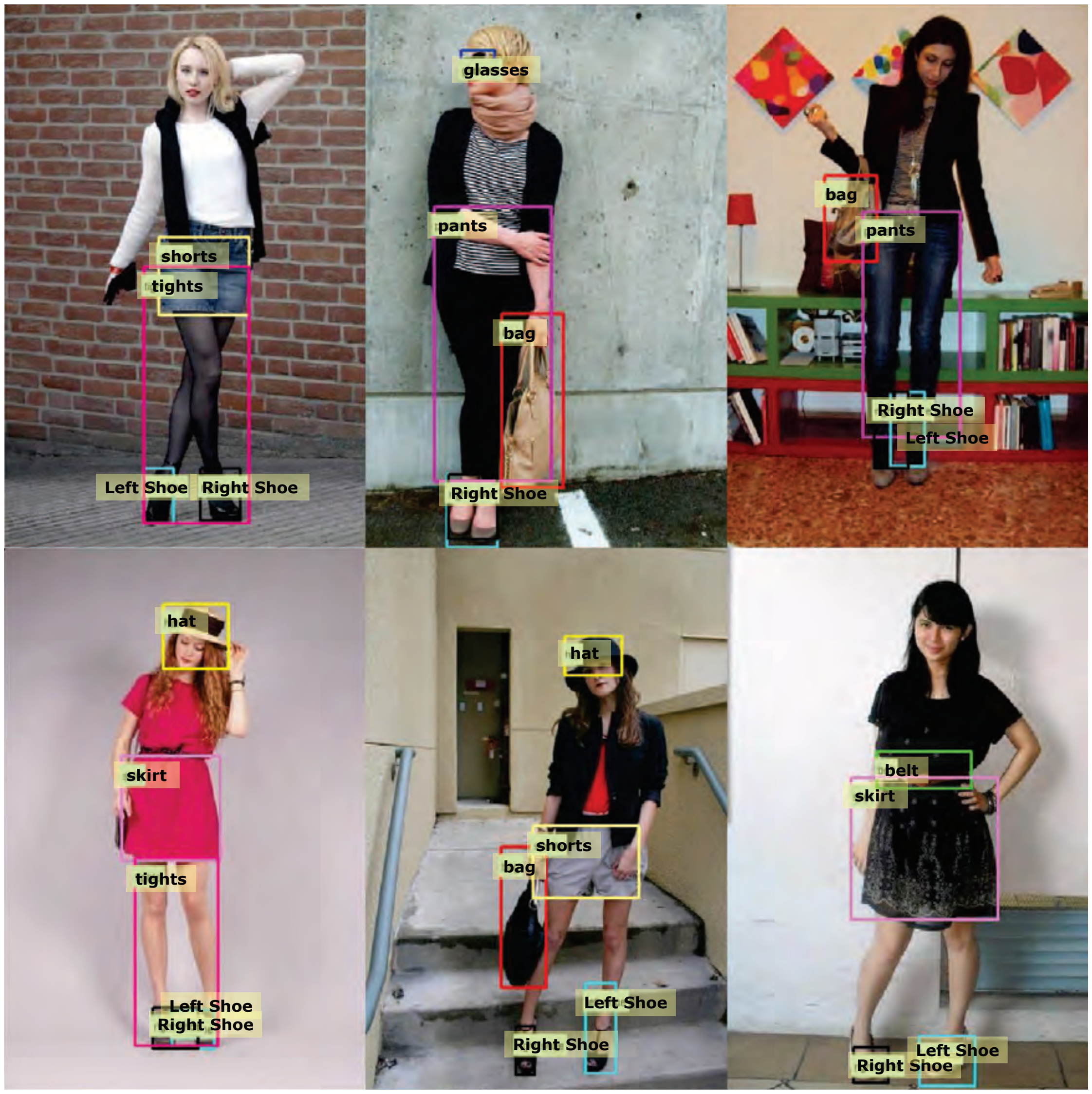


Figure 4.5: Example detection results obtained by the proposed method. Note that we manually overlaid the text labels to improve legibility.

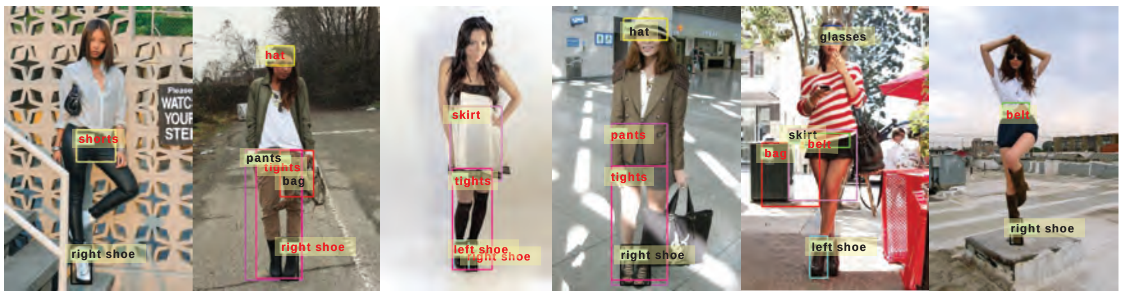


Figure 4.6: Examples of failed detection results obtained by the proposed method. Note that we overlaid text labels manually to improve legibility. Incorrect labels are shown in red.

Chapter 5: Attentional Network for Visual Object Detection

Object detection is one of the most fundamental problems in computer vision. Given an image, the goal of an object detection algorithm is to detect and localize all instances of pre-defined object classes, typically in the form of bounding boxes with confidence values. Although an object detection problem can be converted to many object classification problems by a scanning window technique [Viola and Jones, 2004], it is inefficient since a classifier has to be applied to all potential image regions at various locations, scales, and aspect ratios. The region-based convolution neural network (R-CNN) [Girshick et al., 2014] algorithm suggested recently a two-stage approach. It first generates a set of object proposals, called regions of interest (ROI), using a proposal generator and then determines the existence of an object and its classes in the ROI using a deep neural network. The R-CNN algorithm has achieved impressive performance on public benchmarks and has become the backbone of many recent object detection methods.

In detecting an object, the R-CNN algorithm and its extensions look at the ROI (and sometimes its neighborhood) given by the proposal generator only once. This is in contrast to humans' capability of multiple fixations of visual attention as depicted in Fig. 5.1. We propose to imitate such an attention mechanism for improving the object detection performance of the R-CNN algorithm. To this end, we develop an algorithm that adaptively places a sequence of glimpses for accumulating visual evidence for determining the object class and its precise location from ROIs. We use a recurrent neural network for learning the glimpse placement as well as for summarizing the visual evidence extracted from the multiple glimpse. Due to lacks of ground truth annotations of visual attention for the object detection task, we use a reinforcement algorithm to train the proposed network.

Our work is largely inspired by Mnih et al. [2014a], which uses a visual attention mechanism for the handwritten digit classification task. In this chapter, we study a more challenging task, detecting visual objects in images. Due to large variation in object appearances, it is more difficult to learn a reliable attention mechanism. The glimpse has to vary both in shapes and scales for finding most relevant information. We investigate the network structure that can facilitate the learning of a reliable visual attention strategy for the object detection task. We provide detailed performance analysis in the experiment section. We evaluate the proposed algorithm, which we refer to as Attention-based visual Object Detection network (AOD), on the PASCAL VOC detection benchmarks and demonstrate its advantage over the baseline R-CNN algorithm that does not have the visual attention mechanism.

5.1 Related Work

The attention mechanism has been proposed for different applications including speech recognition [Chorowski et al., 2015], machine translation [Bahdanau et al., 2015] and question–answering [Sukhbaatar et al., 2015]. Among many previous attempts, our work is inspired by Mnih et al. [2014b] who present a recurrent neural network model that sequentially selects and processes sub-regions of an image and combines the information from those regions to obtain a better representation for a targeted task. The method is applied to classify handwritten digit images with clutters, and excellent results are demonstrated. The proposed method is an extension of Mnih et al. [2014b] to the more difficult visual object detection task for dealing with variations in appearances deformations and scales. Unlike the classification task, the object detection task requires algorithms to localize multiple objects from various classes present in an input image. While in Mnih et al. [2014b], glimpse size and shape are fixed, in our method, glimpse can change its shape adaptively to detect objects in various size and shapes.

A lot of works have been done on various vision tasks after Mnih et al. [2014b]. To handle multiple objects in a more realistic image, Ba et al. [2015] extends Mnih et al. [2014b] by allowing a model to predict one object at each time step, making the glimpse network deeper and introducing a context network. The method is applied to the task of transcribing multi-digit house numbers from Google Street View images. Sermanet et al. [2015] applies a recurrent neural network almost identical to the one in Ba et al. [2015] to a more challenging fine-grained categorization task. Yeung et al. [2015] also uses a recurrent neural network trained by the REINFORCE algorithm to address a video-based action detection task and achieves state-of-the-art results. Unlike Mnih et al. [2014b], Ba et al. [2015], Sermanet et al. [2015] where a spatial attention policy is learned, a model in Yeung et al. [2015] learns to output temporal attentions. Beside the aforementioned works, a recurrent neural network with an attention mechanism trained by a reinforcement learning algorithm has been applied to tasks such as image caption generation [Xu et al., 2015], image generation [Gregor et al., 2015] and action recognition [Sharma et al., 2015].

A few attention-based methods have been proposed for the object detection task. Caicedo and Lazebnik [2015] train a class specific object localization model using a reinforcement learning technique and utilize the model for a detection task by evaluating all the regions generated over the course of localization. Yoo et al. [2015] also propose a class specific model which iteratively modifies the initial ROI until it declares the existence of an object. Unlike these works, the proposed method is class agnostic, which scales better with the number of object classes as we do not need to train a separate detector for each object class.

Most of the recent object detection methods are following the R-CNN style approach. There are currently two main directions explored for further improvement. The first direction is to make the underlying CNN deeper [He et al., 2015]. The second one is incorporating semantic segmentation [Dai et al., 2015], which typically require additional training data for semantic segmentation. Other works focus on speeding up the computation time [Girshick, 2015, Ren et al., 2015, Redmon et al., 2015, Lenc and Vedaldi, 2015, Najibi et al., 2015].

An attempt to extract features from multiple regions is made by a few works. In Gidaris and Komodakis [2015], in addition to the proposal bounding box, visual features are extracted from a set of hand-crafted regions and used for the recognition. Bell et al. [2015] also explore the use of multiple regions by extracting features from the entire image in addition to the proposal bounding boxes in order to capture the context. Unlike ours, the additional regions are manually predefined and not adaptively selected.

5.2 Attention-based Object Detection Network

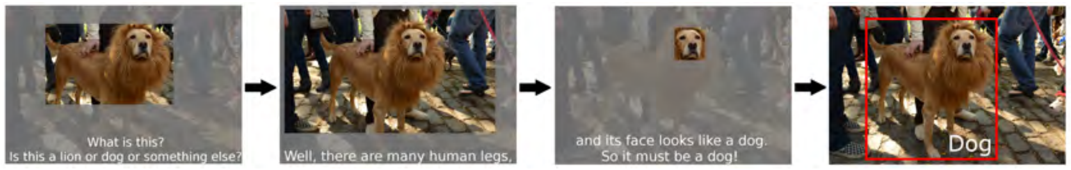

What is this?
Is this a lion or dog or something else?
Well, there are many human legs,
and its face looks like a dog.
So it must be a dog!
Dog

Figure 5.1: Humans have the capability of using multiple fixation points to accumulate evidences for detecting objects in a scene.

We describe the AOD network in details. The network is a deep recurrent neural network designed to detect objects in an image by placing a sequence of glimpses of different sizes and aspect ratios, and make a final decision based on features extracted from these glimpses. Each sequence starts from an object proposal bounding box given by the proposal generator and at the end of the sequence, the network produces scores and bounding boxes for all of the pre-defined classes. With the help of a reinforcement learning algorithm, the network is trained to generate glimpses that lead to better detection performance. In the following, we first describe our network behavior in the test time, then we briefly introduce the reinforcement algorithm and the training procedure for the proposed network.

5.2.1 Network Architecture

The AOD network illustrated in Fig. 5.2 is an active recurrent neural network that decides the attention areas by itself. Given an image, the detection process starts by first applying a deep convolutional neural network to the whole image to obtain a set of feature

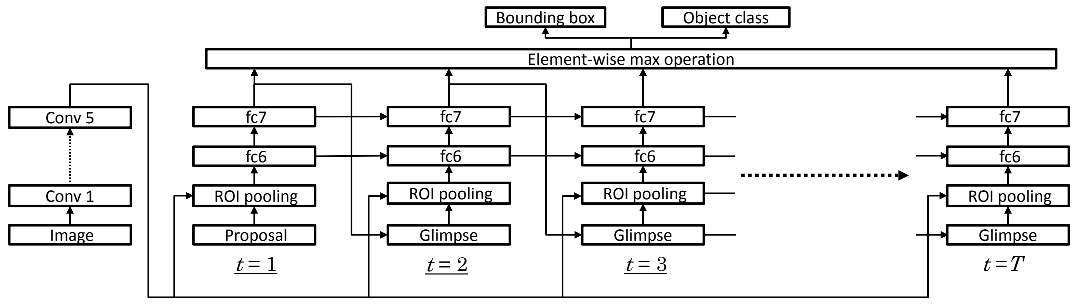


Figure 5.2: Illustration of the AOD network: the network consists a stacked recurrent module designed for object class recognition, bounding box regression and glimpse generation. The classification and bounding box regression are done only at the final time step while the glimpse generation is done at all time steps except the last time step. Given an input image, first, a set of feature maps are computed by the Deep Convolutional Neural Network. Given a proposal bounding box at $t = 1$, a fixed dimensional feature vector is extracted from the proposal bounding box on the last feature map by the ROI pooling layer [Girshick, 2015]. A few fully connected layers (fc6 and fc7 in the figure), each followed by a ReLU and dropout layers, are then applied to the extracted feature vector. From the resultant features, a next glimpse bounding box is determined by applying a fully connected layer. At $t = 2$, a feature vector is extracted from the glimpse bounding box region using the ROI pooling layer. The process is repeated until the last time step $t = T$. At the last time step, an element-wise max operation is applied to the final feature vectors at all time steps and then softmax classification and bounding box regression are conducted.

maps as in the Fast R-CNN algorithm [Girshick, 2015]. In the case of utilizing pre-trained networks such as the AlexNet [Krizhevsky et al., 2012] or the VGG-Net [Simonyan and Zisserman, 2015], the feature maps are computed from the last convolutional layers. Concurrently, a set of proposal bounding boxes is obtained by running a proposal generator. The AOD processes each proposal bounding box separately by extracting the features from the computed feature maps within the bounding box regions. In the following, we describe a procedure applied to each proposal bounding box.

We denote a glimpse at each time step t by $G_t \in \mathbb{R}^4$. The first ROI, G_1, is a proposal bounding box given by the proposal generator and the subsequent G_t are dynamically determined by the network by aggregating information acquired so far. As in Girshick et al. [2014], we employ the scale-invariant and height/width normalized shift parameterization for G_t by using the proposal bounding box as an anchor bounding box. Specifically,

$$G = (\delta_x, \delta_y, \delta_w, \delta_h) = (\frac{g_x - p_x}{p_w}, \frac{g_y - p_y}{p_h}, \log \frac{g_w}{p_w}, \log \frac{g_h}{p_h})$$

where (g_x, g_y, g_w, g_h) is the center coordinate, width and height of the glimpse bounding box. Similarly, (p_x, p_y, p_w, p_h) represents the proposal bounding box. The glimpse layer generates $(\delta_x, \delta_y, \delta_w, \delta_h)$ for determining the glimpse bounding box, which is considered as the glimpse at the next time step. Note that the glimpse bounding boxes are not necessarily the object bounding boxes which indicate the locations of the objects.

From each G_t, a fixed dimensional feature vector is extracted by applying the ROI pooling [Girshick, 2015] to the computed feature maps region within G_t. The ROI pooling works by dividing a given ROI into a predefined grid of sub-windows and then max-pool the feature values in each sub-window. The pooled features are fed into a stacked recurrent neural network of two layers, which are termed as fc6 and fc7 respectively.

At the last time step $t = T$, an element-wise max operation is applied to the last feature vectors at all time steps to compute the final feature vector. The final feature vector is fed into a softmax classification layer and a bounding box regression layer for computing the object class and its location. The softmax classification layer outputs class probabilities over K foreground object classes and one background class. The bounding box prediction layer outputs bounding box prediction for each of the K foreground classes.

We argue that the element-wise max operation retains the most strong signal across time steps independent of the order of the time steps. The stacked recurrent network allows alternative paths of information propagation. They are used because of empirical evidence of superior performance as will be discussed in the experiment section.

5.2.2 Reinforcement learning

The glimpse generation problem can be seen as a reinforcement learning (RL) problem [Sutton and Barto, 1998, Szepesvári, 2010]. In RL, an agent continually interacts with an environment by observing the state $x \in \mathcal{X}$ of the environment and then choosing an action $a \in \mathring{A}$ according to its policy $\pi(a|x)$, a probabilistic mapping from the state to actions. Depending on the current state and the chosen action, the agent's state in the environment changes to $X' \sim \mathcal{P}(\cdot|x, a)$. The agent also receives a real-valued reward signal $r \sim \mathcal{R}(\cdot|x, a)$.

This interaction might continue for a finite or infinite number of steps. In this work, we consider a finite number of steps T. The outcome of each T step of interactions is called an episode, which we denote by ξ.

The goal of an RL agent is to maximize a function of the rewards that it receives. An example of such a function is the sum of rewards in the episode, which can be written as

$$R(\xi) = \sum_{t=1}^{T} r_t.$$

$R(\xi)$ is called the *return* of ξ. The goal of an RL can be stated as finding a policy π which maximize the expected return $J(\pi) \overset{\text{def}}{=\!=} \mathbb{E}_\pi\left[R(\xi)\right]$.

What differentiates RL from supervised learning is that there is no training data consisting of correct input-output pairs. Instead, the policy should be learned based only on the scalar reward signal that the agent receives at each time step. This is very appropriate for our problem as there is no dataset providing us with proper glimpse proposal, but on the other hand it is relatively easy to specify whether the new glimpse proposal is useful for the task of object detection or not.

Among many different approaches to solve an RL problem, in this work we use the REINFORCE algorithm [Wil, 1992], which is a policy gradient approach [Deisenroth et al., 2013, Sutton et al., 2000]. Suppose π is parameterized by θ. The policy gradient algorithm, in its simplest form, changes the policy parameters in the direction of gradient of $J(\pi_\theta)$:

$$\theta_{i+1} \leftarrow \theta_i + \alpha_i \nabla J(\pi_{\theta_i}), \tag{5.1}$$

for some choice of step size $\alpha_i > 0$.

By using the Gaussian distribution as π_θ, the approximate gradients are computed by generating multiple episodes under the current policy (refer to Wil [1992] for the derivation):

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{n}\sum_{i=1}^{n} R(\xi^{(i)}) \sum_{t=1}^{T} \frac{(a_t^{(i)} - \theta x_t^{(i)}) x_t^{(i)\top}}{\sigma^2}. \tag{5.2}$$

Since this is a gradient ascent algorithm, it can easily be incorporated into the standard back propagation neural network training. In fact, our network is trained by back propagating both gradient from reinforcement learning and those from supervised training.

5.2.3 Network Training

The training data fed to our network is constructed in the same way as that in the R-CNN algorithm. Each generated proposal bounding box is assigned a class label c^* among one background class and K foreground object classes according to the overlaps with the ground-truth object bounding boxes. The background class is anything not belonging to any of the foreground classes. Also, given to each of the proposal bounding boxes is a bounding box target vector encoding the scale-invariant translation and log-space height/width shift relative to the object proposal as in Eq. 5.1. Note that the bounding box target vectors for background proposal boxes are not defined and thus not used for training. The details of the training data construction is presented in Sec. 5.2.4.2.

The final outputs from our network are softmax classification scores and bounding boxes for all of the pre-defined foreground classes. During training, ground-truth for them are provided, thus the standard Backpropagation Through Time (BPTT) algorithm [Werbos, 1990] can be used for training. However, since the locations and shapes of the glimpses which lead to a higher detection performance are unknown, the BPTT algorithm cannot be applied to train the glimpse generation layer (an arrow from fc7 to Glimpse in the figure). To train the glimpse generation layer, we use the REINFORCE algorithm presented in Sec. 5.2.2.

In our model, the state is the input given to the glimpse module (i.e., the output of fc7 in Figure 5.2); the action is a new glimpse region described by a G_t at time t. During training, we generate multiple episodes from each sample (a proposal bounding box). All episodes start from the same proposal bounding box and at each time step, i.i.d. Gaussian noise is added to the current glimpse representation computed by the glimpse generation layer. For each episode, the network outputs the class probabilities and object bounding boxes at the last time step. From these outputs, we compute a reinforcement reward for each episode as follows:

$$r_t = \begin{cases} \mathrm{P}(c^*) \times \mathrm{IoU}(B_{c^*}, B^*_{c^*}) & (t = T) \\ 0 & (otherwise) \end{cases} \tag{5.3}$$

where $\mathrm{P}(c^*)$ is the predicted probability of the true class c^* and IoU is the area of intersection of the predicted bounding box for c^* and the ground-truth bounding box, divided by the union of them. Intuitively, if the glimpse bounding box after adding a Gaussian noise leads to a higher class probability and a larger IoU, then a higher return is assigned to the corresponding episode. The REINFORCE algorithm updates the model such that the generated glimpses lead to higher returns. In Mnih et al. [2014b], a 0-1 reward based on the classification success is used. We also evaluated a similar 0-1 reward and found that the proposed continuous reward function performs better for the object detection problem.

The AOD network is trained end-to-end by back propagating an expected gradient of the return along with other gradients computed from the standard classification and bounding box regression losses. The gradients from the REINFORCE algorithm affect all network parameters except those in the classification and bounding box regression layers. The gradients from the classification and localization layers affect all the parameters except those in the glimpse generation layer.

We use the stochastic gradient descent with a mini-batch. To reduce the memory footprint, one mini-batch contains samples from only a few images. Since the number of proposal boxes generated by a proposal generator such as the selective search algorithm [Uijlings et al., 2013] from a single image is large, only a predefined number of foreground samples and background samples are randomly selected and used for training. The detail is provided in Sec. 5.2.4.3.

The policy gradients are only computed for foreground samples because the appearance variations of the background class is larger than those of the foreground classes and it is difficult for a reinforcement agent to learn a good glimpse placement policy. The net effect is that the glimpse generation is optimized only for better discrimination among foreground objects and more accurate bounding box regression. The benefit of excluding background samples for the REINFORCE algorithm is evaluated in Sec. 5.4.

5.2.3.1 Return Normalization

In the REINFORCE algorithm, typically a baseline is subtracted from the return in order to reduce the variance of the expected gradient. A common approach to obtain the baseline is to use exponential moving average of the return before subtracting the baseline [Wil, 1992]. Another approach is to learn a value function $V(x_t) = \mathbb{E}\left[\sum_{l=t}^{T} r_l | x_t\right]$ and use it as a baseline.

We find out that in our setting, computing reliable baselines is challenging. The main reason is that our environment is a space of natural images whose variations are significantly large, and the agent is placed into a variety of different image subregions with different level of difficulties for making accurate decisions. Therefore, it is possible that all the episodes generated from a proposal bounding box A get higher returns than those generated from a proposal bounding box B. In this case, all the episodes from A are prioritized than those from B, which leads to an undesirable training behavior.

To deal with this problem, we convert the original return in Eq. 5.3 by making the mean and variance of the returns computed from all episodes generated from one sample to 0 and 1, respectively, and use the converted return in the REINFORCE algorithm. This way, the new return reflects how well a particular episode works compared to other episodes from the same sample. Also the new return value is less dependent from the samples since it is normalized per sample. We find this approach works well in practice (Sec. 5.4). Note that the proposed return normalization scheme keeps the expected gradients unbiased as the computed baseline is the expectation over the rewards, which becomes a constant as computing the expected gradient.

5.2.4 Implementation Details

In this section, we present some of the implementation details.

5.2.4.1 Glimpse features

At each time step, visual features are computed by the ROI pooling based on the glimpse vector generated in the previous time step. In addition to the visual features, we use the glimpse vector as an additional feature for the current time step. This is to ensure that the network explicitly knows the glimpses it has produced. One fully connected layer followed by ReLU is applied to the glimpse vector and concatenated with the last visual feature vector (i.e., fc7 in Fig. 5.2). Similarly to fc6 and fc7, a recurrent connection is applied. Note that for $t = 1$, the zero vector is fed as glimpse features.

5.2.4.2 Training sample construction

The training sample construction follows the procedure described in the Fast R-CNN algorithm [Girshick, 2015]. For each sample, i.e., proposal bounding box B, we compute the IoU with all the ground-truth bounding boxes and select one with the highest IoU. Let α denote the highest IoU and c denote the class label of the selected ground-truth bounding box. If $\alpha \geq 0.5$, we assign c to B and if $0.5 > \alpha \geq 0.1$, we assign the background class label to B. We ignore all other proposal bounding boxes for training. The whole process is done once before the start of the training stage.

5.2.4.3 SGD hyper parameters

For each mini-batch, we randomly pick two training images and from each image, we randomly select 16 foreground samples and 48 background samples, resulting in 128 samples in one mini-batch. The glimpse generation layer is initialized from zero-mean Gaussian distributions with standard deviations 0.0001. The glimpse generation layer does not have a bias term. All the recurrent layers are initialized from zero-mean Gaussian with standard deviations 0.01 and the biases are set to 0. The fully connected layer applied to the glimpse vectors have 32 output neurons. We multiply the return by 0.1 to control the balance against the classification loss and regression loss.

The initial learning rate is set to 0.001. We run SGD for 30k mini-batch iterations, reduce the learning rate to 0.0001 and then train for another 10K iterations. A momentum of 0.9 and parameter decay of 0.0005 (on weights and biases) are used.

5.2.4.4 Underlying Convolutional Network

Our AOD uses a deep convolutional network (DCNN) to convert an input image into a set of feature maps. We evaluate AOD with two renowned DCNN architectures, CaffeNet Jia et al. [2014] (essentially AlexNet Krizhevsky et al. [2012]) and VGG16 Simonyan and Zisserman [2015] proposed for an image classification task. The CaffeNet has 5 convolution layers, 2 fully connected layers and 1 softmax classification layer while VGG16 has 13 convolution layers, 2 fully connected layers and 1 softmax classification layer. Before the training of AOD, we first train a Fast R-CNN model using the above DCNN pre-trained on the ImageNet Classification task, following Girshick et al. [2014]. We then initialize all the convolution layers and 2 fully connected layers (fc6 and fc7 in Fig. 5.2) of the AOD by the corresponding layers in the trained Fast R-CNN model.

5.2.4.5 Other default settings

Here we summarize some of the important parameters and design choices in our default network architecture. We set $T = 3$ if not specifically mentioned. We set the standard deviations of the Gaussian random perturbation added to the generated glimpse representation to 0.2. The number of episodes generated from one sample is 8. Unlike a standard recurrent neural network, we have separate weights for a glimpse prediction layer at each time step. We empirically found this rendered a better performance.

5.3 Main Results

We evaluate the AOD algorithm on 2007 and 2012 PASCAL VOC detection tasks [Everingham et al., 2010]. In these tasks, there are 20 object classes and detectors are expected to produce a set of bounding boxes with scores. A detection is considered correct if the output bounding box overlaps with a ground-truth object bounding box significantly. The performance of a detector for a single object class is given by the average of the precision rates obtained at different recall rates. The performance of a detector for the entire task is computed by the mean of the average precisions (mAP) of the 20 classes. We compare different object detectors based on the mAP measure. For more details about the performance metric and evaluation protocol, please refer to [Everingham et al., 2010].

In this work, we focus on validating the use of the attention mechanism for object detection. Hence, we only compare our results with those obtained by the baseline algorithm—the Fast R-CNN algorithm. Since the DCNN architecture employed has a significant impact on the final performance, we show performance results separately based on the DCNN used. We also use the same proposal bounding boxes and the same pre-trained DCNN used in the Fast-RCNN work for a fair comparison.

We present experiment results obtained under four different settings, which use different combinations of training and testing data as in Girshick [2015]. The VOC 2007 and VOC 2012 settings are the official settings, and the VOC 2007+2012 and VOC

2007++2012 are additional settings used to show the effect of augmented training data. The training data in the VOC 2007+2012 consists of the training data from VOC 2007 and 2012 as well as the test data from VOC 2012. The training data in the VOC 2007++2012 consists of the training data from VOC 2007 and 2012 as well as the test data from VOC 2007. These settings are summarized in Table. 5.1.

Table 5.1: The experimental settings

Experimental setting ID	Testing data	Training data
VOC 2007	VOC 2007 test	VOC 2007 trainval
VOC 2012	VOC 2012 test	VOC 2012 trainval
VOC 2007+2012	VOC 2007 test	The union of VOC 2007 trainval, VOC 2012 trainval and VOC 2012 test
VOC 2007++2012	VOC 2012 test	The union of VOC 2007 trainval, VOC 2007 test and VOC 2012 trainval

Table 5.2 compare the performance of the proposed algorithm with the baseline algorithm when both are based on the CaffeNet [Jia et al., 2014] in the VOC 2007 setting. We find that the proposed AOD method achieves an mAP of 58.1 when $T = 3$ and an mAP of 57.8 when $T = 2$, both outperforming the mAP of 57.1 obtained by the Fast R-CNN baseline. This validates the effectiveness of the use of the proposed attention mechanism.

In Table 5.3, we show that the proposed method improves the mAP from 58.1 to 67.5 by using a stronger VGG16 net presented in Simonyan and Zisserman [2015]. It again outperforms the Fast R-CNN baseline, which obtains 66.9 by using VGG16 net. The consistent improvements over the baseline by using a stronger DCNN suggests that better performance can be obtained by applying the proposed algorithm with better DCNN.

Table 5.4 shows the detection results on the VOC 2012 setting. Again, the proposed algorithm outperforms the baseline algorithm, improving the mAP from 65.7 to 66.7.

In Table. 5.5 and Table 5.6, we present the performance when trained with a large training set. We observe that all the methods improve with a larger training set. The benefit of the attention mechanism is not downgraded by the use of additional training data.

In Fig. 5.3, we show some example detection results using VGG16 under 2007+2012 setting. We first observe that AOD detects objects well. In the figure, we also visualize the learned glimpse. We find that the reinforcement agent first tries to capture the context around the proposal bounding box and then looks at smaller regions.

5.4 Design Evaluation

We conduct a set of design evaluations to understand the impact of design choices in the AOD. The evaluations are conducted under the VOC 2007 setting with the CaffeNet.

Number of episodes: We evaluate the impact of the number of episodes generated from one sample in a mini-batch (Table 5.7). As can be seen, the larger number of episodes tends to lead better performance. Since the computation time and the amount of memory also increase with the larger number of episodes, we pick 8 as the default number of episodes.

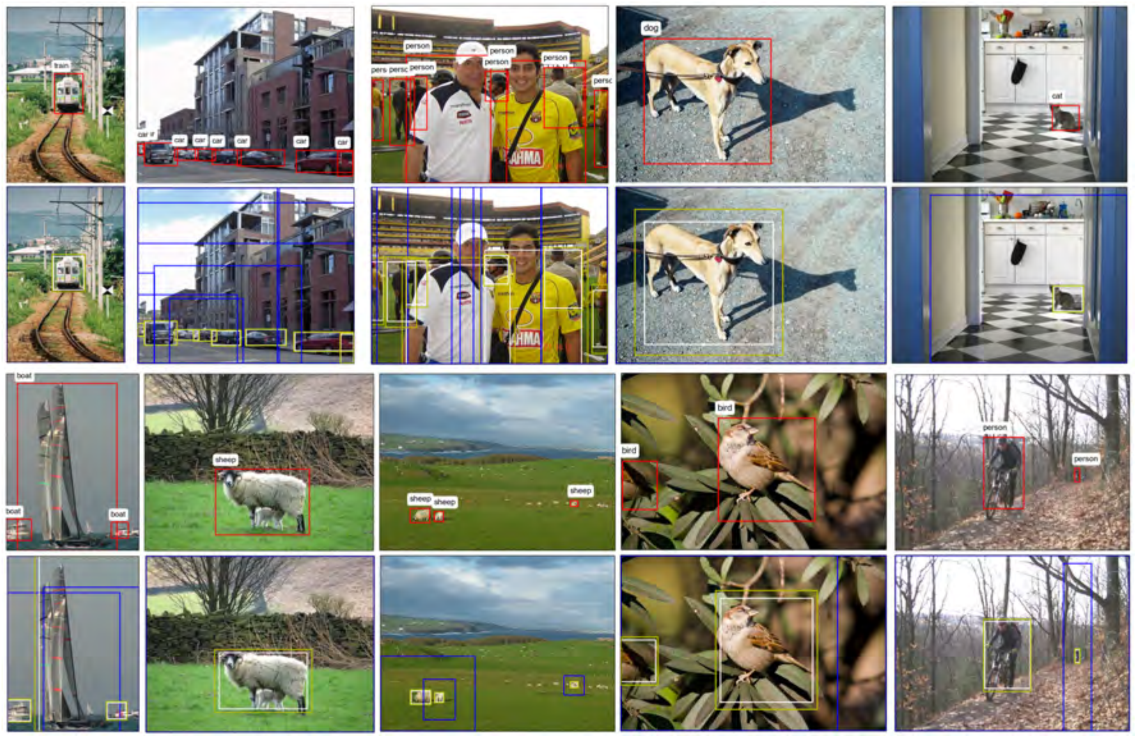


Figure 5.3: Representative detection results. White, blue, yellow and red bounding boxes represent object proposals, the first glimpses, the second glimpses and the final localization results, respectively.

Network architecture: We employ a stacked recurrent neural network, which has recurrent connections at both fc6 and fc7. We compare the default network architecture with a standard recurrent neural network, which has a recurrent connection only at fc7. In addition, we evaluate versions which directly perform the final classification and regression using the recurrent features at the last time step—without conducting the element-wise max operation. As shown in 5.8, the stacked RNN with the element-wise max perform significantly better than the other architectures.

Reinforcement baseline: We evaluate the effect of the reinforcement baselines. We compare our return normalization method presented in Sec. 5.2.3 with the exponential

Table 5.2: Average Precision of methods using CaffeNet under the VOC 2007 setting

Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fast R-CNN	66.4	71.6	53.8	43.3	24.7	69.2	69.7	71.5	31.1	63.4	59.8	62.2	73.1	65.9	57.0	26.0	52.0	56.4	67.8	57.7	57.1
AOD T=2	66.4	72.9	51.1	44.4	24.8	66.5	71.2	72.5	30.2	66.3	63.0	65.0	74.1	68.5	58.3	25.5	50.5	55.8	71.2	56.9	57.8
AOD T=3	67.3	72.5	51.3	45.5	26.5	67.5	71.0	71.5	30.4	65.6	64.2	66.4	74.1	69.0	58.2	24.4	53.7	55.3	69.8	58.5	58.1

Table 5.3: Average Precision of methods using VGG16 under the VOC 2007 setting

Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fast R-CNN	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
AOD T=2	74.9	78.1	64.9	51.3	40.8	80.1	78.5	80.6	42.9	74.1	68.4	78.2	79.9	76.5	69.4	32.1	64.4	67.1	74.7	65.5	67.1
AOD T=3	76.4	78.2	67.6	51.3	41.0	79.6	78.2	83.0	42.1	73.8	68.0	79.7	79.7	75.2	69.2	34.0	66.0	66.4	75.0	66.2	67.5

Table 5.4: Average Precision of methods using VGG16 under the VOC 2012 setting

Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fast R-CNN	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
AOD T=2	81.6	78.0	69.1	50.1	37.0	74.2	68.5	87.4	41.3	71.6	52.7	86.1	79.0	79.7	71.0	32.0	67.6	63.5	78.7	61.9	66.5
AOD T=3	82.5	77.6	69.7	50.0	37.4	74.2	68.7	87.0	41.8	71.4	52.8	85.7	78.9	79.6	70.9	32.8	67.6	63.9	78.9	61.8	66.7

Table 5.5: Average Precision of methods using VGG16 under the VOC 2007+2012 setting

Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fast R-CNN	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4	70.0
AOD T=2	77.6	78.6	70.1	59.7	38.2	83.3	79.3	87.6	48.3	78.9	71.8	83.5	84.0	78.8	71.7	33.1	73.3	74.3	80.0	70.2	71.1
AOD T=3	77.2	79.7	69.5	60.2	38.5	83.8	79.5	86.2	48.9	81.2	72.2	83.5	83.0	77.9	72.1	33.9	73.7	74.7	79.1	70.4	71.3

Table 5.6: Average Precision of methods using VGG16 under VOC 2007++2012 setting

Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fast R-CNN	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2	68.4
AOD T=2	82.6	79.5	70.2	52.5	40.9	78.1	72.8	89.7	46.3	75.3	58.3	87.6	82.9	81.5	73.3	35.6	69.3	68.3	81.7	64.6	69.5
AOD T=3	82.2	79.6	70.5	52.7	40.5	78.5	72.8	88.9	45.8	75.6	57.7	87.5	82.5	80.9	73.6	35.3	69.6	67.5	80.8	64.6	69.4

moving average baseline. For the exponential moving average baseline, the result with the best smoothing parameter value obtained through a grid search is shown.

Table 5.7: The effect of the number of episodes generated from one sample in a mini-batch

# of episodes	2	4	8	16
mAP	57.4	57.5	58.1	57.8

Table 5.8: The effect of the network architecture

Network architecture	mAP
Stacked RNN with element-wise max	58.1
RNN with element-wise max	57.4
Stacked RNN without element-wise max	57.0
RNN without element-wise max	57.2

Table 5.9: The effect of the reinforcement baseline methods

Reinforcement baseline	mAP
Return normalization (ours)	58.1
Moving average	57.8

Continuous return vs. discrete return: Our return is continuous (Eq.5.3), ranging from 0 to 1. In Mnih et al. [2014b], a discrete return is employed: a return is 1 if the highest scoring class is the ground-truth label and 0 otherwise. For validating the use of the continuous return, we adopt a similar discrete return computation where we assign 1 if the highest scoring class is the ground-truth label AND an IoU between a predicted bounding box and the ground-truth bounding box is greater than or equal to the IoU threshold used in the evaluation. The results demonstrate the superiority of the continuous return over the discrete return (Table. 5.10).

Table 5.10: The effect of the choice between continuous return and discrete return

Continuous return vs. discrete return	mAP
Continuous	58.1
Discrete	57.8

Table 5.11: The effect of excluding background samples

With background samples?	mAP
without background samples	58.1
with background samples	57.6

Table 5.12: The effect of the glimpse representation

Glimpse representation	mAP
x-shifting, y-shifting, x-scaling and y-scaling,	58.1
x-shifting, y-shifting	57.3

Effect of excluding background samples: We evaluate the effect of excluding background samples from the REINFORCE algorithm. Since there are no ground-truth bounding boxes for background samples, we always set IOU in Eq. 5.3 to 1 for background samples. As can be seen in Table. 5.11, excluding background samples yields a better performance.

Glimpse representation: Our glimpse is represented as a four dimensional vector encoding x-shifting, y-shifting, x-scaling and y-scaling, enabling to generate an arbitrary glimpse bounding box. To evaluate the effect of different level of flexibility in representing glimpses, we conduct an experiment with a model employing two dimensional glimpse representation encoding only x-shifting and y-shifting (Table. 5.12). The experimental results clearly show that allowing the network to produce arbitrary-shaped glimpse bounding boxes is important for achieving a good performance.

5.5 Conclusion

We propose an attentional network for visual object detection. It sequentially generates glimpse regions of various sizes and aspect ratios, extracted features from these regions, and makes a final decision based on the information it has acquired. The key advantage of the proposed method is that the glimpses are adaptively generated in order to make more accurate decision. Since there are no ground truth annotations for glimpse locations and shapes, we train the network using a reinforcement learning algorithm. The consistent performance improvement over the baseline method verifies the benefit of incorporating the attention mechanism into the deep neural networks for the object detection task.

5.6 Acknowledgments

The work presented in this chapter was conducted as part of the internship at Mitsubishi Electric Research Laboratories.

Chapter 6: Summary and Directions for Future Work

In Chapter 2, we presented an efficient regression-based approach for the 2D human body joint estimation task. Our strategy is to decompose the full pose estimation problem into a set of local pose estimation problems and progressively estimate joint locations along the paths in a dependency graph representing dependency structure of the body joints. Through the experiments on widely used datasets, we demonstrated that the proposed algorithm is simple, yet effective.

In Chapter 3, we turned our focus toward the regression method and proposed a novel node splitting method for regression tree training. The method is based on the observation that the objective function of the regression tree training is similar to the objective function of the k-means clustering method. Unlike traditional binary node splitting method, the proposed algorithm allows K-ary node splitting and splitting based on multiple input dimensions while not relying on the exhaustive search. We applied the proposed method on head pose estimation, car orientation estimation and pedestrian orientation estimation tasks and demonstrate significant improvements.

In Chapter 4, we proposed an algorithm for detecting fashion items a person in the image is wearing or carrying. Since the locations of the fashion items are strongly correlated with the body joint locations, we model their relationship using a mixture of Gaussian model and use it as an additional cue to determine the detection score. Combined with the state-of-the-art object detection method utilizing the object proposal scheme, we show that the use of pose information significantly improve the detection performance.

In Chapter 5, we presented our work on object detection using an attention mechanism. We proposed a deep recurrent neural network architecture which sequentially explores potential regions of object instances in order to make better detection decisions. We successfully trained the proposed network using a technique from reinforcement learning. The proposed method consistently outperforms the baseline method, which does not have the attention mechanism, on several public benchmarks.

6.1 Directions for Future Research

We believe that each part of the work presented above has an interesting direction for future research.

6.1.1 Human Body Pose Estimation by Regression on a Dependency Graph

The key contribution of this work is to sequentially predict joint locations, from more stable joints to more varying ones. In this work, we use boosted regression trees as a regression method, however, nothing prevent us from using other regression methods. Since DCNN is the most promising learning method, it would be interesting to see if the performance improves by using the DCNN regression in stead of the boosted regression trees.

6.1.2 Growing Regression Tree Forests by Classification for Continuous Object Pose Estimation

The regression tree training algorithm presented in this dissertation is a general regression method and thus can be applied to a variety of regression tasks, including non-computer vision tasks. In fact, it has been applied to cardiac four-Chamber volume estimation [Xia et al., 2015] and semantic sentence similarity measurement [Lev et al., 2015]. It would be also interesting to see the performance of the proposed method on tasks such as facial point localization task, human body joint localization, hand pose estimation, age estimation and so on.

Due to the advent of deep learning, may of the image recognition tasks are now addressed by the deep learning-based methods with great successes. It is also observed that deep convolutional neural networks pre-trained on a large image classification dataset serve as a good feature extractor. In the experiments we conducted, we used the classic HOG features. We are interested to know if the standard use of the deep learning techniques outperform the proposed regression forest approach.

6.1.3 Fashion Apparel Detection: the Role of Deep Convolutional Neural Network and Pose-dependent Priors

The current approach trains the appearance-based detector and geometric priors separately, and combine them using a probabilistic formulation whose parameters are determined by cross-validation. A better approach would be to train both components jointly in a deep learning framework by putting both images and pose information into the network and allowing the interaction between them in the network. This strategy should be achieved by using more training data. Furthermore, it might be possible to also incorporate the pose prediction component into the deep network with two loss functions, one for pose estimation and the other for fashion apparel detection.

6.1.4 Attentional Network for Visual Object Detection

The proposed network uses a simple recurrent layer to allow sequential processing. The recurrent property can also be achieved by more complex Long Short-Term Memory (LSTM) layers [Hochreiter and Schmidhuber, 1997] which allow the model to have long-term memories. In fact, many of the recent works demonstrate superior performance by using LSTM. The main reason why we employed a simple recurrent layer is that we thought that the long-term memory which can store memories of hundreds of time steps is not necessary for our task and also training LSTM requires more training data.

The biggest problem of the proposed method is a limited scalability. The REINFORCE algorithm requires generation multiple episodes from each training sample, significantly increasing the training time as the number of episodes increases. The spatial transformer Networks [Jaderberg et al., 2015] is a trainable module which explicitly allows spatial transformation of the feature maps. It is possible to use the spatial transformer layer to transform proposal bounding boxes into glimpse bounding boxes without resorting to the REINFORCE algorithm.

Bibliography

Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Monocular 3D Pose Estimation and Tracking by Detection. In *CVPR*, 2010.

J R R Uijlings, K.E.A. Van De Sande, T Gevers, and A.W.M. Smeulders. Selective Search for Object Recognition. *IJCV*, 2013.

R Girshick. Fast r-cnn. In *ICCV*, 2015.

Martin A. Fischler and Robert A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, 1973.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Matching of Pictorial Structures. *CVPR*, 2000.

Alessandro Bissacco, Ming-Hsuan Yang, and Stefano Soatto. Fast Human Pose Estimation using Appearance and Motion via Multi-dimensional Boosting Regression. In *CVPR*, 2007.

Ankur Agarwal and Bill Triggs. A Local Basis Representation for Estimating Human Pose from Cluttered Images. *ACCV*, 2006.

Makoto Yamada, Leonid Sigal, and Michalis Raptis. No Bias Left Behind: Covariate Shift Adaptation for Discriminative 3D Pose Estimation. *ECCV*, 2012.

Xiaofeng Ren, Alexander C Berg, and Jitendra Malik. Recovering Human Body Configurations Using Pairwise Constraints Between Parts. *ICCV*, 2005.

Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Discriminative Appearance Models for Pictorial Structures. *IJCV*, 2011.

Benjamin Sapp, Alexander Toshev, and Ben Taskar. Cascaded Models for Articulated Pose Estimation. *ECCV*, 2010.

Min Sun, Murali Telaprolu, Honglak Lee, and Silvio Savarese. An Efficient Branch-and-bound Algorithm for Optimal Human Pose Estimation. *CVPR*, 2012a.

Yi Yang and Deva Ramanan. Articulated Pose Estimation with Flexible Mixtures-of-Parts. *CVPR*, 2011.

Norimichi Ukita. Articulated Pose Estimation with Parts Connectivity Using Discriminative Local Oriented Contours. *CVPR*, 2012.

Mun W. Lee and Isaac Cohen. Proposal Maps Driven MCMC for Estimating Human Body Pose in Static Images. *CVPR*, 2004.

Kota Hara and Takaharu Kurokawa. Human Pose Estimation Using Patch-based Candidate Generation and Model-based Verification. In *Face and Gesture (FG)*, 2011.

Jürgen Müller and Michael Arens. Human Pose Estimation with Implicit Shape Models. *ARTEMIS*, 2010.

Bastian Leibe, A Leonardis, and B Schiele. Robust Object Detection with Interleaved Categorization and Segmentation. *IJCV*, 2008.

D. H. D. West. Updating Mean and Variance Estimates: An Improved Method. *Communications of the ACM*, 1979.

Jerome H. Friedman. Greedy Function Approximation : A Gradient Boosting Machine. *The Annals of Statistics*, 2001.

Makoto Yamada, T Suzuki, T Kanamori, H Hachiya, and M Sugiyama. Relative Density-ratio Estimation for Robust Distribution Comparison. *NIPS*, 2011.

Leonid Pishchulin, A Jain, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele. Articulated People Detection and Pose Estimation: Reshaping the Future. *CVPR*, 2012.

Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Second Edition.

Roman Rosipal and Leonard J. Trejo. Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space. *JMLR*, 2, 2001.

Sijmen de Jong. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 1993.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least Angle Regression. *The Annals of Statistics*, 2004.

Arasanathan Thayananthan, Ramanan Navaratnam, B Stenger, Philip H.S. Torr, and Roberto Cipolla. Multivariate Relevance Vector Machines for Tracking. *ECCV*, 2006.

Silvia Zuffi, Oren Freifeld, and Michael J Black. From Pictorial Structures to Deformable Structures. *CVPR*, 2012.

M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari. 2D Articulated Human Pose Estimation and Retrieval in (Almost) Unconstrained Still Images. *IJCV*, 2012.

Murad Al Haj, J. Gonzalez, and Larry S Davis. On Partial Least Squares in Head Pose Estimation: How to simultaneously deal with misalignment. In *CVPR*, 2012.

Michele Fenzi, Laura Leal-taixé, Bodo Rosenhahn, and Jörn Ostermann. Class Generative Models based on Feature Regression for Pose Estimation of Object Categories. In *CVPR*, 2013.

Marwan Torki and Ahmed Elgammal. Regression from Local Features for Viewpoint and Pose Estimation. In *ICCV*. Ieee, 2011.

Min Sun, Pushmeet Kohli, and Jamie Shotton. Conditional Regression Forests for Human Pose Estimation. In *CVPR*, 2012b.

Kota Hara and Rama Chellappa. Computationally Efficient Regression on a Dependency Graph for Human Pose Estimation. In *CVPR*, 2013.

Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time Facial Feature Detection using Conditional Regression Forests. In *CVPR*, 2012.

Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. Face Alignment by Explicit Shape Regression. In *CVPR*, 2012.

Leo Breiman. Random Forest. *Machine Learning*, 45(1), 2001.

Antonio Criminisi, Jamie Shotton, Duncan Robertson, and Ender Konukoglu. Regression Forests for Efficient Anatomy Detection and Localization in CT Studies. In *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging*, volume 6533, 2010.

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

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

R. L. Kashyap. A Bayesian Comparison of Different Classes of Dynamic Models Using Empirical Data. *IEEE Transactions on Automatic Control*, 22(5), 1977.

Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2), 1978.

Keinosuke Fukunaga and Larry D. Hostetler. The Estimation of the Gradient of a Density Function , with Applications in Pattern Recognition. *IEEE Transactions on Information Theory*, 21(1), 1975.

Yizong Cheng. Mean Shift , Mode Seeking , and Clustering. *PAMI*, 17(8), 1995.

Dorin Comaniciu and Peter Meer. Mean Shift : A Robust Approach Toward Feature Space Analysis. *PAMI*, 24(5), 2002.

Nicolas Gourier, Daniela Hall, and James L. Crowley. Estimating Face orientation from Robust Detection of Salient Facial Structures. In *ICPR*, 2004.

Mustafa Ozuysal, Vincent Lepetit, and Pascal Fua. Pose Estimation for Category Specific Multiview Object Localization. In *CVPR*, 2009.

Yan Yan, Elisa Ricci, Ramanathan Subramanian, Oswald Lanz, and Nicu Sebe. No Matter Where You Are: Flexible Graph-guided Multi-task Learning for Multi-view Head Pose Classification under Target Motion. In *ICCV*, 2013.

Chen Huang, Xiaoqing Ding, and Chi Fang. Head Pose Estimation Based on Random Forests for Multiclass Classification. In *ICPR*. Ieee, aug 2010.

Javier Orozco, Shaogang Gong, and Tao Xiang. Head Pose Classification in Crowded Scenes. In *BMVC*. British Machine Vision Association, 2009.

Davide Baltieri, Roberto Vezzani, and Rita Cucchiara. People Orientation Recognition by Mixtures of Wrapped Distributions on Random Trees. In *ECCV*, 2012.

Sholom M. Weiss and Nitin Indurkhya. Rule-based Machine Learning Methods for Functional Prediction. *Journal of Artificial Intelligence Research*, 3, 1995.

Luís Torgo and João Gama. Regression by Classification. In *Brazilian Symposium on Artificial Intelligence*, 1996.

Alin Dobra and Johannes Gehrke. SECRET: A Scalable Linear Regression Tree Algorithm. *ACM SIGKDD*, 2002.

Philip A. Chou. Optimal Partitioning for Classification and Regression Trees. *PAMI*, 13 (4), 1991.

Usama M. Fayyad and Keki B. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *Proceedings of the International Joint Conference on Uncertainty in AI*, 1993.

Fernando Berzal, Juan-Carlos Cubero, Nicolás Marn, and Daniel Sánchez. Building Multi-way Decision Trees with Numerical Attributes. *Information Sciences*, 165(1-2), 2004.

Wei-Yin Loh and Nunta Vanichsetakul. Tree-structured Classification Via Generalized Discriminant Analysis. *Journal of the American Statistical Association*, 83(403), 1988.

Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded Pose Regression. In *CVPR*, 2010.

Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. Technical report, Department of Statistics,UC Berkeley, 2004.

Pedro Domingos. MetaCost: A General Method for Making Classifiers Cost-Sensitive. In *SIGKDD*, 1999.

Michael Pazzani, Christoper Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing Misclassification Costs. In *ICML*, 1994.

Miroslav Kubat, Robert Holte, and Stan Matwin. Learning when Negative Examples Abount. In *ECML*, 1997.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique Nitesh. *Journal of Artificial Intelligence Research*, 16, 2002.

Chris Drummond and Robert C. Holte. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling. In *ICML Workshop on Learning from Imbalanced Datasets II*, 2003.

Luís Torgo, Rita P. Ribeiro, Bernhard Pfahringer, and Paula Branc. SMOTE for Regression. In *Portuguese Conference on Artificial Intelligence*, 2013.

Shou-Jen Chang-Chien, Wen-Liang Hung, and Miin-Shen Yang. On Mean Shift-based Clustering for Circular Data. *Soft Computing*, 16(6), jan 2012.

Takumi Kobayashi and Nobuyuki Otsu. Von Mises-Fisher Mean Shift for Clustering on a Hypersphere. In *ICPR*. Ieee, aug 2010.

Mehran Kafai, Yiyi Miao, and Kazunori Okada. Directional Mean Shift and its Application for Topology Classification of Local 3D Structures. In *CVPR Workshop*. Ieee, jun 2010.

Kuo-Lung Wu and Miin-Shen Yang. Mean shift-based clustering. *Pattern Recognition*, 40(11), nov 2007.

Gabriele Fanelli, Juergen Gall, and Luc Van Gool. Real Time Head Pose Estimation with Random Regression Forests. In *CVPR*, 2011.

Huy Tho Ho and Rama Chellappa. Automatic Head Pose Estimation Using Randomly Projected Dense SIFT Descriptors. In *ICIP*, 2012.

Kevin Bailly, Maurice Milgram, and Philippe Phothisane. Head Pose Estimation by a Stepwise Nonlinear Regression. In *International Conference on Computer Analysis of Images and Patterns*, 2009.

Juergen Gall and Victor Lempitsky. Class-Specific Hough Forests for Object Detection. In *CVPR*, 2009.

Ross Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. Efficient regression of general-activity human poses from depth images. *ICCV*, nov 2011.

Carolina Redondo-cabrera, Roberto López-Sastre, and Tinne Tuytelaars. All together now : Simultaneous Object Detection and Continuous Pose Estimation using a Hough Forest with Probabilistic Locally Enhanced Voting. In *BMVC*, 2014.

Christian Herdtweck and C Curio. Monocular Car Viewpoint Estimation with Circular Regression Forests. In *Intelligent Vehicles Symposium (IVS)*, 2013.

Linjie Yang, Jianzhuang Liu, and Xiaoou Tang. Object Detection and Viewpoint Estimation with Auto-masking Neural Network. *ECCV*, 8691, 2014.

Michele Fenzi and Jörn Ostermann. Embedding Geometry in Generative Models for Pose Estimation of Object Categories. *BMVC*, 2014.

Michele Fenzi, Laura Leal-taixé, Jörn Ostermann, and Tinne Tuytelaars. Continuous Pose Estimation with a Spatial Ensemble of Fisher Regressors. In *ICCV*, 2015.

Kun He, Leonid Sigal, and Stan Sclaroff. Parameterizing Object Detectors in the Continuous Pose Space. In *ECCV*, 2014.

Hiroaki Shimizu and Tomaso Poggio. Direction Estimation of Pedestrian from Multiple Still Images. In *Intelligent Vehicles Symposium (IVS)*, 2004.

Tarak Gandhi and Mohan Manubhai Trivedi. Image Based Estimation of Pedestrian Orientation for Improving Path Prediction. In *Intelligent Vehicles Symposium*. Ieee, jun 2008.

Chikahito Nakajima, Massimiliano Pontil, Bernd Heisele, and Tomaso Poggio. Full-body person recognition system. *Pattern Recognition*, 36(9), sep 2003.

Cheng Chen, Alexandre Heili, and Jean-Marc Odobez. Combined Estimation of Location and Body Pose in Surveillance Video. In *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Ieee, 2011.

Guangzhe Zhao, Mrutani Takafumi, Kajita Shoji, and Mase Kenji. Video Based Estimation of Pedestrian Walking Direction for Pedestrian Protection System. *Journal of Electronics (China)*, 29(1-2), jun 2012.

Markus Enzweiler and Dariu M. Gavrila. Integrated Pedestrian Classification and Orientation Estimation. In *CVPR*, 2010.

Junli Tao and Reinhard Klette. Integrated Pedestrian and Direction Classification using a Random Decision Forest. In *ICCV Workshop*, 2013.

Kunihiro Goto, Kiyosumi Kidono, Yoshikatsu Kimura, and Takashi Naito. Pedestrian Detection and Direction Estimation by Cascade Detector with Multi-classifiers Utilizing Feature Interaction Descriptor. In *IEEE Intelligent Vehicles Symposium (IV)*. Ieee, jun 2011.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9, 2008.

Dan Pelleg and Andrew Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML*, 2000.

Kota Hara and Rama Chellappa. Growing Regression Forests by Classification: Applications to Object Pose Estimation. In *ECCV*, 2014.

Gary L. Gaile and James E. Burt. *Directional Statistics*. Geo Abstracts Ltd., 1980.

N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1996.

K. V. Mardia and P. Jupp. *Directional Statistics, 2nd edition*. John Wiley and Sons Ltd., 2000.

J. D. F. Habbema and J Hermans. Selection of Variables in Discriminant Analysis by F-statistic and Error Rate. *Technometrics*, 19(4), 1977.

Robert P. W. Duin. On the Choice of Smoothing Parameters for Parzen Estimators of Probability Density Functions. *IEEE Transactions on Computers*, C-25(11), 1976.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 2011.

Xiantong Zhen, Zhijie Wang, Mengyang Yu, and Shuo Li. Supervised Descriptor Learning for Multi-Output Regression. In *CVPR*, 2015.

Haopeng Zhang, Tarek El-gaaly, Ahmed Elgammal, and Zhiguo Jiang. Joint Object and Pose Recognition using Homeomorphic Manifold Analysis. In *AAAI*, 2013.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1), mar 2006.

Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support Vector Regression Machines. In *NIPS*, 1996.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

Kunihiko Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 202, 1980.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

Kota Yamaguchi, M. Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Parsing clothing in fashion photographs. *CVPR*, 2012.

Kota Yamaguchi, M. Hadi Kiapour, and Tamara L Berg. Paper Doll Parsing : Retrieving Similar Styles to Parse Clothing Items. *ICCV*, 2013.

Basela S Hasan and David. C Hogg. Segmentation using Deformable Spatial Priors with Application to Clothing. *BMVC*, 2010.

Andrew C. Gallagher and Tsuhan Chen. Clothing cosegmentation for recognizing people. *CVPR*, jun 2008.

Zhilan Hu, Hong Yan, and Xinggang Lin. Clothing segmentation using foreground and background estimation based on the constrained Delaunay triangulation. *Pattern Recognition*, 41(5), may 2008.

Nan Wang and Haizhou Ai. Who Blocks Who: Simultaneous clothing segmentation for grouping images. *ICCV*, nov 2011.

Lukas Bossard, Matthias Dantone, Christian Leistner, Christian Wengert, Till Quack, and Luc Van Gool. Apparel classification with style. *ACCV*, 2012.

Jie Shen, Guangcan Liu, Jia Chen, Yuqiang Fang, Jianbin Xie, Yong Yu, and Shuicheng Yan. Unified Structured Learning for Simultaneous Human Pose Estimation and Garment Attribute Classification. *IEEE Transactions on Image Processing*, 2014.

Huizhong Chen, Andrew Gallagher, and Bernd Girod. Describing clothing by semantic attributes. *ECCV*, 2012.

Ming Yang and Kai Yu. Real-time clothing recognition in surveillance videos. *ICIP*, 2011.

Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Describing people: A poselet-based approach to attribute classification. *ICCV*, 2011.

Yufei Wang and Garrison W. Cottrell. Bikers are like tobacco shops, formal dressers are like suits: Recognizing urban tribes with caffe. *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, 2015.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, sep 2010.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

Lubomir Bourdev and Jitendra Malik. Poselets : Body Part Detectors Trained Using 3D Human Pose Annotations . *CVPR*, 2009.

Pablo Arbelaez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale Combinatorial Grouping. *CVPR*, 2014.

Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers. *ICML*, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian Detection with Unsupervised Multi-stage Feature Learning. *CVPR*, jun 2013.

Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. PANDA: Pose Aligned Networks for Deep Attribute Modeling. *CVPR*, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-jia Li, Kai Li, and Li Fei-fei. ImageNet : A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF : A Deep Convolutional Activation Feature for Generic Visual Recognition. *ICML*, 2014.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CVPR Workshop*, mar 2014.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv*, 2014.

Paul Viola and Michael J. Jones. Robust real-time face detection. *IJCV*, 2004.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014a.

Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *NIPS*, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. *NIPS*, 2014b.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2015.

Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. *arXiv:1511.06984*, 2015.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv:1502.04623*, 2015.

Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv:1511.04119*, 2015.

Juan C. Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015.

Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S. Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *ICCV*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *arXiv:1512.04412*, 2015.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv:1506.02640*, 2015.

Karel Lenc and Andrea Vedaldi. R-cnn minus r. *arXiv:1506.06981*, 2015.

Mahyar Najibi, Mohammad Rastegari, and Larry S. Davis. G-cnn: an iterative grid based object detector. *arXiv:1512.07729*, 2015.

Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. *arXiv:1505.01749v3*, 2015.

Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *arXiv:1512.04143*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010.

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8, 1992.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 2000.

P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Mark Everingham, Luc Van Gool, ChristopherK.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Wenfeng Xia, Efthymios Maneas, Daniil I Nikitichev, Charles A Mosse, Gustavo Sato dos Santos, Tom Vercauteren, Anna L David, Jan Deprest, Sebastien Ourselin, Paul C. Beard, and Adrien E. Desjardins. Medical Image Computing and Computer-Assisted Intervention MICCAI 2015. *MICCAI 2015, Part I, Lecture Notes in Computer Science*, 9349(October), 2015.

Guy Lev, Benjamin Klein, and Lior Wolf. In defense of word embedding for generic text representation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9103, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural Computation*, 9(8), 1997.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. *NIPS*, 2015.