Dissertation

Object Class Detection Using Part-Based Models Trained From Synthetically Generated Images

Johannes Schels

Universität Augsburg
Fakultät für Angewandte Informatik

Faculty of Applied Computer Science
University of Augsburg

Advisor:	Prof. Dr. Rainer Lienhart
Reviewers:	Prof. Dr. Rainer Lienhart
	Prof. Dr. Bernhard Möller
	Prof. Dr. Eckehard Steinbach

Thesis Defense: March 5, 2013

Abstract

This thesis presents part-based approaches to object class detection in single 2D images, relying on pre-built CAD models as a source of synthetic training data.

Part-based models, representing an object class as a deformable constellation of object parts, have demonstrated state-of-the-art results with respect to object class detection. Typically, the majority of part-based approaches rely on real training images of publicly available image data sets and consequently, the positive output of those detectors is restricted to the viewpoints which are represented by those real training images. However, progress in the domain of computer graphics enables the generation of photo-realistic renderings on demand from a database of CAD models, which can serve as training source for learning an object class detection approach. In this thesis, we present part-based object class detection methods which are based on synthetically generated positive training images and real negative training images, thereby combining the advantages of the two domains described above. More specifically, photo-realistic object parts, representing the object class being trained, are learnt in an unsupervised way without requiring any manual bounding box, object part, or viewpoint annotations during the training process. The established object parts are efficiently combined into an object class detection framework relying on two part-based models with different learning paradigms. In addition, we outline an extension of our detection framework which is able to cope with multiple object classes.

The approaches to object class detection are evaluated on standard benchmark data sets and achieve state-of-the-art results with respect to object class detection in single 2D images.

Acknowledgements

At this point, I would like to thank all the people who advised, supported, and encouraged me throughout this thesis. First of all, I would like to thank Prof. Dr. Rainer Lienhart for supervising my thesis. I am grateful for his scientific guidance as well as for his encouragement and patience. This thesis was part of an industry cooperation with EADS Innovation Works in Munich. Special thanks go to my colleague and tutor Dr. Jörg Liebelt. This thesis would have been impossible without his encouragement and support. I want to thank Judith Andres, Helga Bayer, Peter Bernhardt, Prof. Dr. Alexander von Bodisco, Stefan Burger, Thilo Fath, Emanuel Heidinger, Olaf Heinzinger, Kevin Müller, Daniel Münch, Margot Neumeier, Jan Nowotsch, Thomas Reif, Fabian Richter, Christian Ries, Stefan Romberg, Dr. Klaus Schertler, and Eva-Maria Steindl, who directly or indirectly influenced this thesis. I am deeply grateful to my family, who always supported the way I chose. I am indebted to my parents, Johann Josef and Maria Anna Schels, who believe in me and support me in everything I do. Where I am today is due to their continuous support and encouragement. Andrea Schön was on my side throughout this adventure and was my link to the real world. She is my strength and inspiration more often than she knows.

Contents

Chapter 1

Introduction

In this chapter, we motivate our work on object class detection and explain the challenges in this research area. Further, we describe the contributions to object class detection provided by this thesis and outline its context.

1.1 Motivation

As outlined in (10), for a human a fraction of a second is enough to interpret a novel image. For example, a glance at Figure 1.1 is sufficient to detect all individual object instances and identify the way in which they might interact. In computer vision a long-term objective is to replicate this ability of the human vision system with respect to scene understanding. Replicating this ability would enable new possibilities for applications such as autonomous robots for domestic or rescue scenarios, driver assistance systems, or visual surveillance. In order to achieve this objective, the scene understanding problem can be divided into subtasks. Object class detection is one of the key requirements. It denotes the ability to detect all object instances of a predefined object class within an image.

In recent years remarkable progress has been achieved in the area of object class detection based on robust local image features (86) in conjunction with machine learning techniques (11). In contrast to holistic object class detection approaches, which model the entire appearance of object instances within an object class (21, 93), part-based approaches to object class detection represent object instances within an object class as a flexible constellation of individual object parts. They have already shown an

Figure 1.1: For a human a glance is enough to recognize all individual object instances in this image and identify their possible interactions (10). The image is taken from the PASCAL VOC2006 data set (28).

excellent detection performance for various object classes (3, 34, 40, 72). Typically, approaches to object class detection, including part-based approaches, mainly rely on publicly available data sets (27, 100, 129) for training. These data sets consist of manually annotated consumer photographs collected from photo-sharing web sites which usually reflect typical object views. A good example, for instance, is the object class car within the PASCAL VOC2006 data set (28). The majority of the cars within this data set is shown from the point of view of a person. Consequently, a detector for the object class car relying on those images as positive training set would not be able to detect cars from a bird's eye view, since the positive set of training images does not include this viewpoint. However, the domain of computer graphics enables the generation of photo-realistic renderings from pre-built CAD[1] models on demand from arbitrary viewpoints with varying lighting and background conditions, which can be used as positive training images for learning to detect an object class.

[1] Abbreviation for **C**omputer-**A**ided **D**esign

Consequently, the present thesis proposes methods for object class detection which combine the advantages of part-based object class detection approaches with the advantages of synthetically generated training images. The described part-based approaches rely on object class specific databases of pre-built CAD models to generate positive training images which avoid the need for any real positive training images in addition to a set of real negative training images.

1.2 Challenges in Object Class Detection

The main objective of this thesis is to develop algorithmic approaches to object class detection: given a 2D image from an optical camera, we intend to detect all object instances of a predefined object class. In this thesis, we follow the work of (68) and rely on the visual similarity of object instances in order to form object classes such as car, bicycle, or motorbike. In the following, we describe the main challenges in the area of object class detection and we briefly outline how the present thesis addresses these problems:

Figure 1.2: The appearance of object instances within an object class can vary significantly. The images are taken from the PASCAL VOC2006 data set (28).

- **Intra-class Variance:** One major problem of object class detection is that object instances within an object class can vary greatly in appearance. As illustrated

in Figure 1.2, these differences arise from a variability in geometry or texture. Although all cars in Figure 1.2 are shown from the same viewpoint, significant variations in appearance occur. An approach to object class detection must be able to deal with intra-class variation and should be able to detect all object instances of an object class. In this thesis, this problem is addressed by using part-based models which are established from a database of pre-built CAD models and a set of real negative images. A part-based model exploits the fact that within an object class the appearance of object parts, i.e., local image areas, is less variable and that these object parts occur in a consistent geometric configuration. In addition, CAD models are used to generate systematic variations of the object class being trained. Further details on part-based models and CAD models are given in Section 2.2 and Section 2.3. Furthermore, in order to model the appearance of an object class, all proposed approaches rely on machine learning techniques (11) in conjunction with local image features (86), which are invariant against small geometric variations.

Figure 1.3: Pose variance of an object instance within the object class car. The images are taken from the 3D Object Category data set (100).

- **Pose Variance:** As shown in Figure 1.3, one object instance of an object class can also cause significant variation in appearance when it is seen from different viewpoints. Object class detectors require the ability to deal with pose variation and should be able to detect all object instances of an object class from arbitrary viewpoints. However, object class detection approaches mainly rely on publicly available data sets of real images which often consist of images showing only some of the object views. Consequently, the resulting object class detectors are restricted to those viewpoints. To address this problem, we resort to CAD models as a source of synthetic training data which allows us to generate training

images of the object class being trained from arbitrary viewpoints. Based on these synthetically generated positive training images and a set of real negative images, part-based object class representations are built which are able to better deal with pose variation.

- **Occlusion and Background Clutter:** As shown in Figure 1.4, object instances are often occluded by other object instances, or some parts of an object instance stretch beyond the image border. In this work, this problem is addressed by part-based models which have an increased robustness to partial occlusion (104). As also shown in Figure 1.2 or Figure 1.4, an object instance can occur in different environments. In this thesis, we rely on CAD models which allows changing the background of the synthetically generated positive training images systematically in order to take into account those background variations during the training of an object class.

Figure 1.4: Occlusions in real world images are normally caused by other object instances or by parts of an object instance stretching beyond the image border. The images are taken from the PASCAL VOC2006 data set (28).

- **Supervision of Training Data:** Most of the existing object class detection approaches are trained on publicly available data sets (27, 100, 129). All training images of those data sets are provided at least with a rectangular bounding box, a weak pose annotation, and sometimes even with a segmentation mask. The generation of such annotations for a training image requires significant human effort. However, the training of an object class detection system should require as little manual intervention as possible. In this thesis, we circumvent this issue by resorting to pre-built CAD models as a source of synthetically generated positive training images. We argue that it is legitimate to access this resource for computer vision approaches in the same way as manually annotated large-scale image

databases are used in other approaches. All proposed approaches in this work rely on CAD models to generate positive training images and hence, no manual bounding box or viewpoint annotations within the synthetically generated positive images are necessary, since this information is automatically provided during the generation of an image.

- **Multiple Object Classes:** Different methods exist to detect object instances from multiple object classes within an image. One common strategy to multi-class object detection is termed 'one-versus-all': during the training and detection process of an object class detection framework each object class is treated independently from all other object classes. In contrast, in this thesis we present additional approaches to multi-class object detection relying on different sharing strategies of object parts to exploit the dependencies and similarities between the object classes being trained.

1.3 Contributions

In the following, we highlight the main contributions of this thesis:

- **Rendering Synthetic Training Data:** We present part-based approaches to object class detection which rely on pre-built CAD models as a source of synthetic training data and therefore avoid the need for any real positive training images of the object class under training. Based on an object class specific database of CAD models, positive training images are generated on demand including systematic variations in viewpoint, background, and lighting. Consequently, the proposed part-based approaches to object class detection do not require any manual bounding-box or viewpoint annotations, since this information is automatically provided by the rendering process.

- **Finding Suitable Object Parts:** Based on synthetically generated positive training images and a set of real negative training images, we present two different approaches for an unsupervised identification of object parts which are suitable to represent the object class being trained. In contrast to the first proposed approach, which is restricted to discrete viewpoints on the viewsphere, the second proposed approach makes use of viewpoint symmetries and part similarities within

an object class over the entire viewsphere to discover common object parts such that an optimal coverage of intra-class and viewpoint variation is guaranteed.

- **Combining Different Learning Paradigms:** The proposed approaches to object class detection combine two part-based models which rely on different learning paradigms into one common object class detection framework. These approaches exploit the benefits from these different learning paradigms by relying on generatively trained part-based models to produce initial object hypotheses, which are subsequently verified by a discriminatively trained part-based model to provide each initial object hypothesis with a final detection score.

- **Object Part Sharing Strategies for Multi-Class Object Detection:** We present and compare different learning methods for multi-class object detection which rely on different part sharing strategies to determine their suitability for learning multiple object classes on a larger scale.

1.4 Thesis Context

In this section, we describe how the present thesis is related to previous work. We identify links to part-based models and to the use of CAD models in computer vision which we briefly summarize in the following. Detailed surveys of related work are given in Chapter 3, Chapter 4, and Chapter 5.

1.4.1 Part-Based Models

The idea of representing an object class as a flexible arrangement of object parts was originally introduced by (44). Early approaches relying on this idea (14, 15, 38, 125, 126) used hand-labeled part locations and interest point detectors (85) for learning a part-based model. Further part-based approaches also built on interest point detectors to form a code book representation of an object class (1, 71). With the efficient matching scheme of (36) the focus has moved to tree-structured and star-structured models (36, 39) and their extensions with spatial priors (19, 20). The efficient matching scheme allows an exhaustive search for those part-based models over all possible part locations within an image by using the generalized distance transform (35) and dynamic programming. More recently, discriminatively trained part-based models have

been introduced by (8, 37, 66). The discriminative part-based model of (37) is extended in (34, 91) with the introduction of mixture models and part sharing among the components of such a mixture model. The present thesis takes up the original idea of (44) and represents an object class as a flexible constellation of object parts.

1.4.2 CAD Models

The use of CAD models in computer vision has a long history. First approaches used textureless CAD models in order to establish a recognition system for a specific object instance (55, 59, 60, 62, 63) relying on simple features such as surface normals, surface curvature, or edge features. Further approaches matched groups of line segments on an object model (81), established a relational graph representation of a CAD model (46), relied on indexing methods or geometric hashing (128), or resorted to a textured CAD model for the detection and the pose estimation of a specific object instance (2). More recently, the idea of using both textureless and textured CAD models for computer vision tasks has regained attention, notably in (77) aligning a CAD model to obtain the pose of a specific object instance, in (23, 119) recognizing 3D objects in an image or point clouds, or in (69, 114) using CAD models to establish a virtual scene for evaluating local image features or surveillance systems. There is also an increasing interest to use CAD models for object class detection approaches (78, 79, 82, 94, 97, 109). In the present thesis, we take up the idea to use CAD models for computer vision tasks and propose part-based approaches to object class detection relying on a database of prebuilt CAD models as a source of synthetically generated training data.

1.5 Thesis Overview

In Chapter 2 preliminary information is provided: an overview of the used terminology in this thesis and a brief introduction to part-based models are given. In addition, an introduction to the generation of synthetic training images, an overview of the benchmark data sets, and a description of the evaluation criterion in object class detection are provided. In Chapter 3 our first part-based approach to object class detection is presented which is based on a database of CAD models and real negative images, the *Multi-View Model*. The *Multi-View Model* establishes several viewpoint-specific part-based models and combines the responses of these models in a joint spatial pyramid encoding. The

detection performance of the *Multi-View Model* is evaluated on several benchmark data sets. Based on the analysis of the *Multi-View Model*, we present in Chapter 4 our second part-based approach to object class detection which also relies on a database of CAD models and real negative images, the *Viewsphere Model*. In contrast to the *Multi-View Model*, the *Viewsphere Model* establishes an object class representation which is based on an object part sharing procedure to densely cover the entire viewsphere. We compare the detection performance of the *Viewsphere Model* on several benchmark data sets. In Chapter 5 the *Viewsphere Model* is extended to cope with multiple object classes. We present and compare three different learning strategies to represent multiple object classes on the entire viewsphere. Chapter 6 concludes this thesis with a summary and provides an outlook on possible research directions in the area of object class detection.

Chapter 2

Preliminaries

This chapter gives a brief overview of the terminology and a general introduction to part-based models. In addition, detailed descriptions of the part-based models, which are used in this thesis, are provided. In this work, we rely on a database of pre-built CAD models to generate positive training images for our proposed object class detection approaches. A short introduction into this topic is provided by describing the properties of CAD models and the rendering process, which generates the synthetic training images. This chapter concludes with a description of the standardized evaluation criterion in the area of object class detection and an overview of the data sets, on which the proposed detection approaches of this thesis are evaluated.

2.1 Terminology

In the following, a brief explanation of the terminology used in the present work is provided:

- **Object Instance:** An individual object which is unique in the world (e.g. my mother's car).
- **Bounding Box:** The outer contour of an object instance within an image is marked by the smallest possible axis-aligned rectangle.
- **Object Class:** A set of object instances which share visually similar properties are grouped together into one object class (e.g. car, bicycle, or motorbike). Also termed object category.

- **Image Classification:** Image classification predicts whether an image contains one or more object instances of a specific object class.
- **Object Class Detection:** Object class detection answers the following question: are there any object instances of a specific object class in an image and if so, where are these object instances located? The position of an object instance within an image is marked by a bounding box.
- **CAD Model:** A CAD model is a collection of 3D points, also called vertices. Based on these vertices, polygons are defined which describe the 3D surface of an object. In addition, for a CAD model different materials are defined and each vertex is connected to a material. Some CAD models also rely on textures, in order to increase the level of detail. A CAD model is also termed 3D object model or 3D model.
- **Rendering Process:** A rendering process refers to a method, by which a computer creates a synthetic image from a CAD model.
- **Discriminative Model:** A model learnt by a machine learning approach, which establishes a decision boundary between a set of positive and negative training samples, is termed a discriminative model. In simple terms, a discriminative model describes directly what distinguishes an object class from other object classes.
- **Generative Model:** A generative model describes what all object instances of an object class have in common and builds an explicit model for each object class (while ignoring the other object classes).
- **Patch:** A local image area of an object instance is termed patch.
- **Local Image Feature:** A local image feature encodes a patch. To this purpose, different techniques are described in the literature.
- **Object Part:** Local image areas or patches which have a similar appearance over all object instances of an object class are termed object parts or simply parts.
- **Part-Based Model:** In general, a part-based model represents an object class based on object parts and their spatial relations.

- **SVM Classifier:** A Support Vector Machine (SVM) classifier is a discriminative learning technique. It models directly the decision boundary between a set of positive and negative training samples.

- **Object Part Detector:** The appearance of an object part is modeled by an SVM classifier in conjunction with a local image feature. We term such an SVM classifier object part detector or part detector.

2.2 Part-Based Models

The principle of a part-based model is shown in Figure 2.1. Within an object class, object instances can vary greatly in appearance which induces significant intra-class variance. However, within sufficiently small local image areas, indicated by the colored boxes in Figure 2.1, the appearance variations within an object class are less pronounced. As mentioned in Section 2.1, these local image areas are termed object parts or simply parts. In addition to having less appearance variation, these object parts often occur in a consistent geometric constellation within an object class. For example, for all object instances shown in Figure 2.1, the 'green part' is always above the 'red part' and the 'red part' is always to the left of the 'yellow part'. Consequently, the

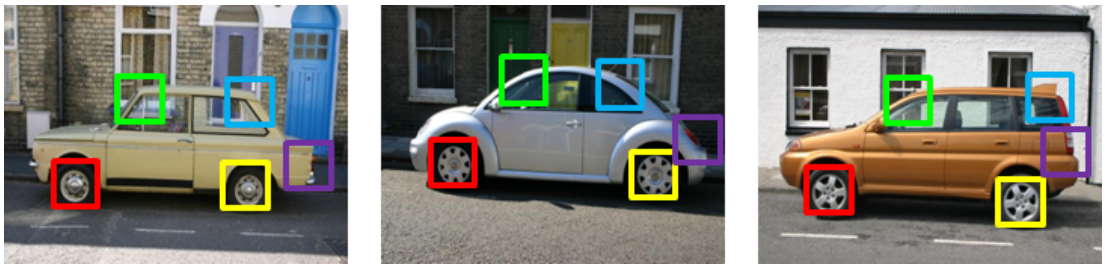

Figure 2.1: The basic idea of a part-based model: within an object class the appearance of object parts (colored boxes) is less variable and the spatial arrangement of these parts often follows a consistent geometric configuration. The images are taken from the PASCAL VOC2006 data set (28).

basic idea of a part-based model is to represent an object class based on the appearance of object parts and their geometric constellation. When using a part-based model in order to establish a representation of an object class, the following three questions have to be considered:

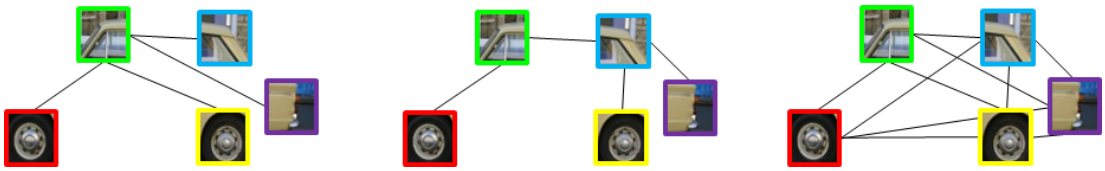

Figure 2.2: Examples of part-based models with different geometric structures: a star-structured model (left), where all object parts are connected to one reference part (green object part). A tree-structured model (center) has one root part (green object part) and each object part is only connected to its own fixed reference part. Within a fully connected model (right) each object part is connected to all other object parts.

- **Part Positions:** The first question is which local image areas within the object class under training are suitable to serve as parts for this object class. Various approaches to this problem are described in the literature. For example, in (19, 36, 103) object parts are manually defined or (53, 78) suggest a fixed grid-based subdivision of object views. The approaches of (38, 72, 84) rely on interest point detectors (85) to determine part positions. As shown in (34), it is also possible to treat the part positions as hidden or latent variables during the training process of a part-based model. In this thesis, we propose two different approaches to determining suitable part positions within the synthetically generated positive training images; see Chapter 3 and Chapter 4, respectively.

- **Part Appearance:** The second question in conjunction with a part-based model is how to represent the appearance of an object part. Early approaches model the appearance of an object part based on simple image patches, as described in (1, 73, 74). However, in recent years there is a growing consensus in the literature that local image features provide a robust and efficient way to represent the appearance of an object part (37, 72, 78, 87, 89, 109). In this thesis, we represent the appearance of an object part by using the Histograms of Oriented Gradient (HOG) descriptor of (21) in conjunction with an SVM classifier (105, 120).

- **Geometric Structure:** The third question of a part-based model concerns the geometric structure which encodes the spatial relations between the object parts. Figure 2.2 shows some common part-based models with different geometric structures: a star-structured model, where each object part is only connected to a fixed reference object part, a tree-structured model, where each object part is only con-

nected to its own fixed reference object part, and a fully connected model, where each object part is connected to all other object parts. The fully connected model has the most complex geometric structure which comprehends both, the star-structured model and the tree-structured model. A brief overview of further geometric structures, e.g., the k-fans of (19), is given in (16). Due to the high computational complexity of a fully connected model (39), a star-structured model or a tree-structured model is often used as an approximation of a fully connected model.

In the present work, we propose approaches to object class detection which combine two different part-based models: the first model is a star-structured model chosen due to its computational efficiency during training and detection. We term this star-structured model the *Star Model*. The second model establishes a representation for the simultaneous occurrence of object parts by relying on the spatial pyramid approach of (70). We term the second model the *Spatial Pyramid Model*. The *Star Model* and the *Spatial Pyramid Model* rely on different learning paradigms: originally, the *Star Model* is a generative part-based model (36, 50), while as described in Section 2.2.2, the *Spatial Pyramid Model* is a discriminatively trained part-based model. For a more detailed discussion on generative and discriminative learning methods, see (50, 67). The proposed object class detection approaches in Chapter 3 and Chapter 4 efficiently combine these two part-based models with different learning techniques. However, in the following these two part-based models are described independently of each other.

2.2.1 Star Model

As shown in Figure 2.2 (left), the *Star Model* is a star-structured model and consists of several object parts, which are all connected to the fixed reference part. In this section, we first describe a training procedure in order to establish a *Star Model* for a specific object class and then, we outline the detection procedure for the *Star Model*.

2.2.1.1 Training Procedure

For describing the training procedure of the *Star Model* we assume positive training images which represent the object class being trained at a predefined training scale[1]. Typically, this training scale represents the object class on the smallest size which should be detected, and the object instances within these training images differ in width and height. For example, on the same training scale the object instance of a compact car within a training image is smaller than the object instance of a limousine. In addition, we assume that within the positive training images all object parts are manually determined and annotated by bounding boxes[2]. These bounding boxes can also vary in width and height. For example, the wheel of a compact car normally is smaller than the wheel of a limousine.

Based on these assumptions, we outline a training procedure for the *Star Model* for a specific object class consisting of two sequentially performed steps: the first step is building an appearance model for each object part and the second step is modeling the spatial relation between each object part and the fixed reference part. An illustration of this sequential learning strategy is given in Figure 2.3.

2.2.1.1.1 Part Appearance

Based on the determined bounding boxes within the positive training images at the predefined training scale, a collection of part-specific training samples, which vary in width and height, is generated for each object part (see Figure 2.3). Subsequently, the training samples for each object part are rescaled to their corresponding average size (i.e. average width and average height) to obtain normalized and part-specific training samples for each object part (see Figure 2.3). In order to model the appearance of an object part at the predefined training scale, we rely on a discriminatively trained representation that combines the HOG descriptor of (21) with a linear SVM classifier. Discriminative training methods such as SVM classifiers (105, 120) or Adaboost (47) are based on a training set of positive and negative samples and establish a model by finding a decision boundary between the positive and the negative samples. As shown

[1] The training scales of the synthetically generated positive training images for the different object classes are given in the experimental sections of Chapter 3, Chapter 4, and Chapter 5.

[2] In Chapter 3 and Chapter 4 respectively, we propose two different approaches for automatically determining suitable part positions within the synthetically generated positive training images instead of choosing them manually.

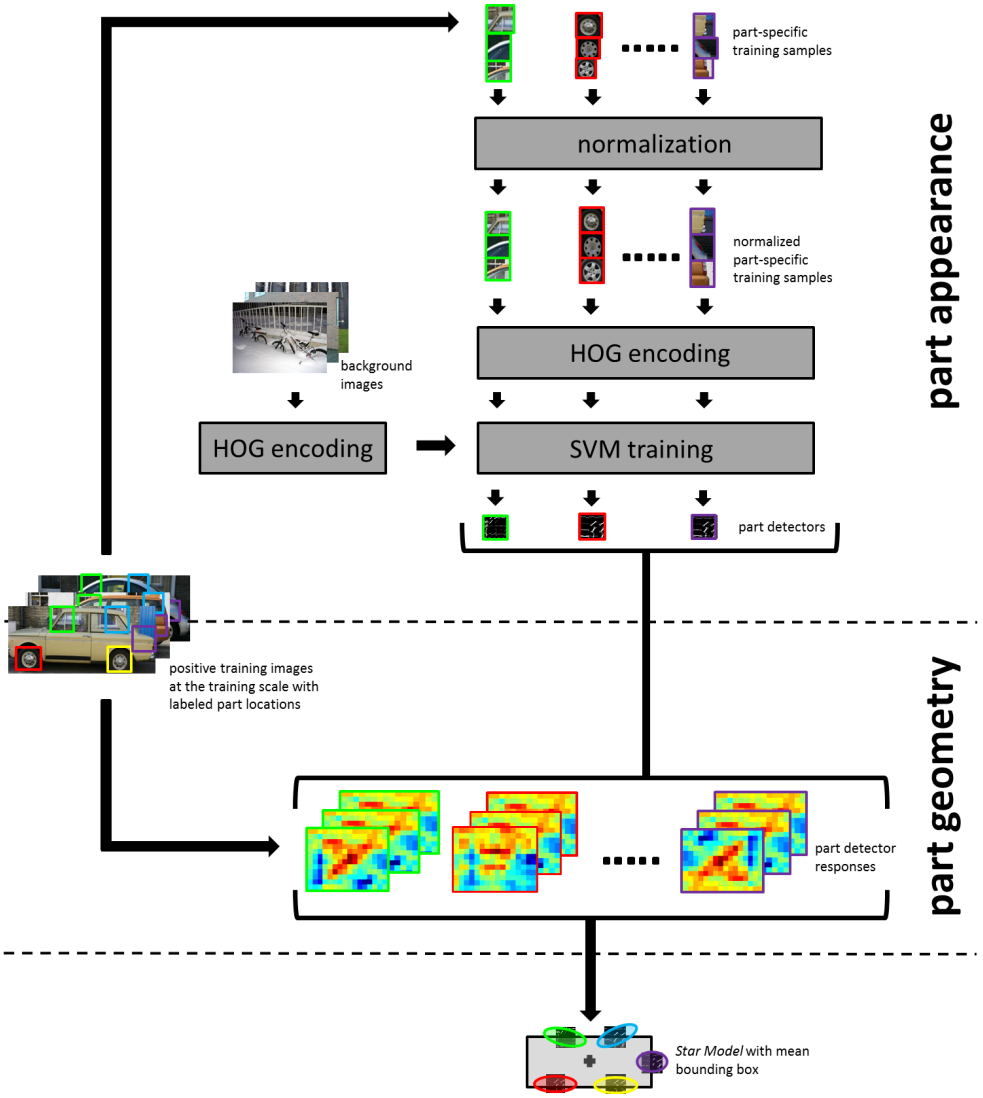

Figure 2.3: Sequential training procedure of the *Star Model* under the assumption that the part locations within the positive training images at the predefined training scale are given: first, an appearance model for each object part is established. The normalized part-specific training samples and background images are used in conjunction with the HOG descriptor to train a linear SVM classifier as an object part detector for each object part. Second, the spatial relation between an object part and the fixed reference part is modeled by a Gaussian mixture model. In this thesis, we use the center of the positive training images as reference and the average size of the positive training samples at the predefined training scale as mean bounding box.

in Figure 2.3, the normalized and part-specific training samples are encoded with the HOG descriptor and serve as positive samples for the SVM training. Background images, which do not contain any object instance of the object class being trained, are also encoded with the HOG descriptor and serve as negative samples during the SVM training. In general, the detection performance of an SVM classifier depends on the set of training samples. Learning an SVM classifier on a few and non-representative training samples normally results in a poor detection performance. As described in (113), the training of an SVM classifier should be based on a comprehensive but computationally feasible training set of positive and negative samples. For the positive samples this procedure is straight forward, since all normalized and part-specific training samples are given and can be encoded with the HOG descriptor in order to serve as positive samples. Considering all possible negative samples from a set of background images simultaneously during training is infeasible as every possible image patch within a background image can potentially serve as a negative sample. In order to keep the number of negative samples computational feasible and still obtain a comprehensive training set of negative samples we rely on a 'bootstrapping' procedure (24, 34, 99, 113) to train a linear SVM classifier as follows:

1. Create an initial training set consisting of all positive samples and randomly selected negative samples from a set of background images.

2. Train a linear SVM classifier based on the current training set.

3. Evaluate all background images with the current linear SVM classifier by using the detection procedure of Section 2.2.1.2 and collect all incorrectly classified negative samples, i.e., false positive samples.

4. Update the current training set by adding a random subset of the collected false positive samples as additional negative samples to the training set.

5. Repeat this procedure from step 2 until a predefined number of maximum bootstrapping iterations has been reached or no further false positives have been detected in the background images.

As shown in Figure 2.3, after a 'bootstrapping' procedure the appearance of an object

part at the predefined training scale of a *Star Model* is represented by a linear SVM classifier. We term such an SVM classifier part detector.

2.2.1.1.2 Part Geometry

The second step of the sequential learning strategy for the *Star Model* is to model the spatial uncertainty between each object part and the fixed reference part at the predefined training scale of the positive training images (see Figure 2.3). However, we do not choose an object part as reference part, instead we use the center of the positive training images at the predefined training scale as reference which actually represents the center of the object class being trained. The reason is that this choice enables a simple but effective method to predict a suitable bounding box during the detection procedure described in Section 2.2.1.2: the average size (i.e. the average width and the average height) of the positive training images at the predefined training scale is projected as mean bounding box into an input image. In order to model the spatial uncertainty between an object part and the center of the positive training images, all established part detectors, i.e., the linear SVM classifiers of Section 2.2.1.1.1, are applied to the positive training images at the predefined training scale. As illustrated in Figure 2.3, we obtain for each object part detector a part detector response on all positive training images. For each object part we measure in each positive training image the location x of the maximum detection response with respect to the image center[1]. We model for each part the spatial distribution pdf_X of all locations X by using a Gaussian mixture model $\theta = \{\alpha_c, \mu_c, \Sigma_c\}$ with C components $(1 \leq c \leq C, c \in \mathbb{N})$

$$pdf_X(x|\theta) = \sum_{c=1}^{C} \alpha_c \mathcal{N}(x|\mu_c, \Sigma_c) = \sum_{c=1}^{C} \alpha_c \frac{1}{2\pi|\Sigma_c|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_c)' \Sigma_c^{-1}(x-\mu_c)} \tag{2.1}$$

where α_c is the prior probability of a component c, μ_c is the mean vector of a component c, and Σ_c is the covariance matrix of a component c. The parameters of a Gaussian mixture model are automatically estimated by the approach of (13). By choosing a mixture of Gaussian distributions, we are able to perform an efficient detection approach which is described in Section 2.2.1.2.

Finally, we obtain a *Star Model* for a specific object class at the predefined training

[1]The position of the maximum detection response with respect to the center of a training image is measured in column direction u and row direction v, i.e., $x' = (\Delta u, \Delta v)$.

scale. The *Star Model* consists of a mean bounding box, i.e., the average size (the average width and the average height) of all positive training images at the predefined training scale, and several object part detectors which are represented by linear SVM classifiers. Each object part detector has its own Gaussian mixture model conditioned to the center of the object class being trained. An example for a *Star Model* is shown in Figure 2.3.

2.2.1.2 Detection Procedure

As described in (36), a part-based model in general can be represented by a collection of n object parts $P = \{p_1, \ldots, p_n\}$. For each pair of connected object parts p_i and p_j there is an edge (p_i, p_j) and Υ denotes the set of all connected object parts. An object instance within an image is given by a configuration $L = (l_1, \ldots, l_n)$, where each l_i specifies the location of an object part p_i. The process to detect an object instance in an image can be formulated in terms of energy minimization (36). The energy of a specific configuration in an image, i.e., of an object instance of a class at a specific location is defined by the energy which is necessary to place each object part onto its location and the energy necessary to deform each pair of connected object parts from their optimal relative arrangement. The energy necessary to place an object part in an image onto a location l_i is given by a function $a_i(l_i)$. The energy necessary to deform a given pair of connected object parts from their optimal relative arrangement and to place an object part p_i onto a location l_i and an object part p_j onto a location l_j is given by a function $s_{ij}(l_i, l_j)$. Then, an optimal configuration L_{opt} of a part-based model in an image is defined by

$$L_{opt} = \underset{L}{\operatorname{argmin}} \left(\sum_{(p_i,p_j)\in\Upsilon} s_{ij}(l_i, l_j) + \sum_{i=1}^{n} a_i(l_i) \right) . \tag{2.2}$$

In general form, i.e., for a fully connected model, the computational complexity to minimize Equation 2.2 is $O(m^n)$, where m is the number of possible locations for each object part in an image and n is the number of object parts. One possible approach to restrict the computational complexity during the detection process of a part-based model is to reduce the part locations to a small set of possibilities returned by an interest point detector (38, 39, 125). However, as shown in (39) such interest point detectors reduce the overall detection performance of a system in contrast to an exhaustive search over all

possible part locations. As mentioned in Section 2.2, the *Star Model* is a star-structured model. The restriction of a part-based model to this specific geometric structure enables a dynamic programming approach that takes $O(nm^2)$. In (36) it is shown that an additional restriction for s_{ij} in terms of a Gaussian distribution (i.e. Mahanalobis distance) yields a minimization of Equation 2.2 in $O(mn)$ by using the generalized distance transform (35). As described in Section 2.2.1.1.2, the spatial uncertainty for each object part of the *Star Model* is modeled by a mixture of Gaussian distributions. Consequently, we adapt the exhaustive detection approach of (36) over all possible part locations. In the following, we describe the resulting detection procedure for the *Star Model*. The equivalence of this detection procedure to (36) is shown in Appendix D.
Given an input image the overall detection process of the *Star Model*, i.e., minimization of Equation 2.2, is illustrated in Figure 2.4. Initially, the given input image is represented by an image pyramid in order to detect object instances of different sizes within the input image. Consequently, the following detection procedure is applied to each scale of the image pyramid: the image at a specific scale of the image pyramid is encoded by the HOG descriptor and for each defined object part of the *Star Model* we apply the corresponding linear SVM classifier to this encoded image to obtain a part detector response. Within a part detector response a high detection score indicates that in this area the input image contains the object part of the corresponding SVM classifier with a high probability. Consequently, we have to invert each part detector response, since the detection process of Equation 2.2 is defined in terms of a minimization problem. For each defined object part of the *Star Model*, a distance transform (35) on the corresponding inverse part detector response is performed for each component of the related Gaussian mixture model (see Section 2.2.1.1.2). The transformed responses of a defined object part are ranked with the corresponding prior probabilities of the related Gaussian mixture model (see Appendix D for details). Finally, the transformed and ranked part detector responses of all defined object parts are added and inverted to obtain a final score map, where the local maxima indicate object hypotheses at a specific scale of the image pyramid. By back-projecting the provided mean bounding box of the *Star Model* into the input image, we are able to predict for each local maximum a suitable bounding box with an associated detection score. Finally, across all defined

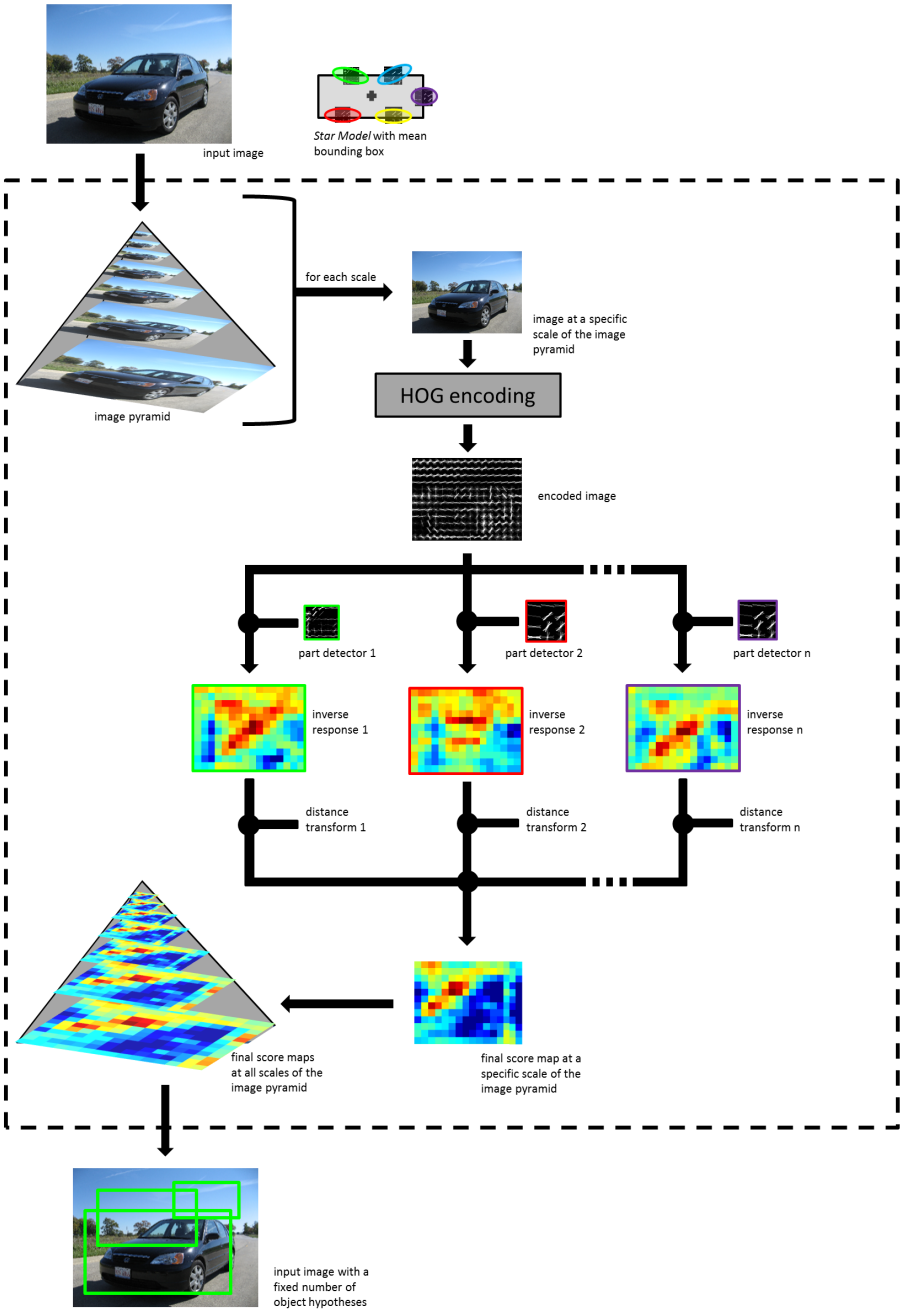

Figure 2.4: The detection procedure of the *Star Model* is performed at each scale of an image pyramid in order to detect object instances of different sizes: at each scale of the image pyramid part detector responses of all object parts are calculated by applying the corresponding linear SVM classifiers to the encoded image. We take the inverse part detector responses and perform a distance transform. The transformed responses are inverted and added in order to obtain a final score map at a specific scale of the image pyramid, where a local maximum indicates an object hypothesis. Across all scales of the image pyramid we keep a fixed number of object hypotheses.

scales within the image pyramid we keep a fixed number[1] of those object hypotheses which have the highest detection scores.

2.2.2 Spatial Pyramid Model

In contrast to the generative *Star Model* of Section 2.2.1, we introduce in this section a discriminatively trained part-based model, the *Spatial Pyramid Model*. In the following, we describe the training procedure in order to establish a *Spatial Pyramid Model* for a specific object class. As shown in Figure 2.5, we assume a set of positive training images which represents the object class at the predefined training scale, a set of negative training images, which does not contain any object instances of the object class being trained, and a set of already trained object part detectors at the predefined training scale (as described in Section 2.2.1.1.1).

Based on the training images and the object part detectors, the *Spatial Pyramid Model* encodes the simultaneous occurrence of all object parts by relying on the approach of (70) in conjunction with an SVM classifier. In the first training step of the *Spatial Pyramid Model*, the following procedure is performed on all training images: a training image is encoded by the HOG descriptor and the corresponding SVM classifiers of all object parts are applied to obtain part detector responses. We follow the approach of (70) and rely on a spatial pyramid representation, which partitions a given training image into increasingly fine sub windows, in order to encode the detector response of each defined object part on a training image (see Figure 2.5). In general, a spatial pyramid representation has the advantage that it imposes a regularly spaced grid subdivision which is relative to the area of a training image and thus independent of the aspect ratio and the dimension of a given training image. As illustrated in Figure 2.5, for each part detector response a spatial pyramid with K levels ($1 \leq k \leq K, k \in \mathbb{N}$) is defined, resulting in a fixed hierarchy of rectangular sub windows. A spatial pyramid, for example, can be defined by a linear subdivision, where the number of sub windows at a specific level k has k sub windows along the column and row direction, or a quadratic subdivision, where the number of sub windows at a specific level k has 2^{k-1} sub windows along the column and row direction. For each defined sub window of an object part detector, we sum up all the positive detection responses, i.e., all val-

[1] The number of object hypotheses per input image of the proposed object class detection approaches is given in Chapter 3 and Chapter 4.

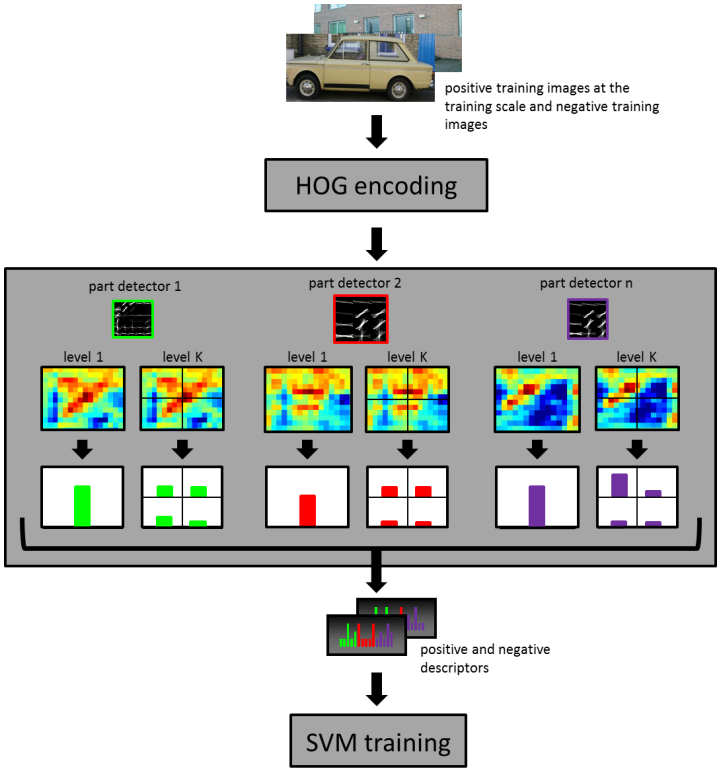

Figure 2.5: The principle of the *Spatial Pyramid Model*: the spatial layout of all part detector responses is encoded into a spatial pyramid. Based on positive and negative training images, we obtain histogram based descriptors which are used to train an SVM classifier.

ues above zero, of the corresponding object part detector and obtain one real-valued number. We concatenate the resulting real-valued numbers of all defined sub windows for all object part detectors in order to obtain a histogram based descriptor for a given training image. The dimension d of such a descriptor depends on the number of object parts n and the number of defined spatial pyramid levels K $(1 \leq k \leq K, k \in \mathbb{N})$. For a linear subdivision of rectangular sub windows we obtain a descriptor with

$$d = n \sum_{k=1}^{K} k^2 = n \frac{1}{6} K(K+1)(2K+1) \tag{2.3}$$

dimensions and with a quadratic subdivision of rectangular sub windows we obtain a descriptor with

$$d = n \sum_{k=1}^{K} 4^{k-1} = n \frac{1}{3}(4^K - 1) \tag{2.4}$$

dimensions.
Given the descriptors of the positive and negative training images, the second training step of the *Spatial Pyramid Model* is to train an SVM classifier by using the 'bootstrapping' procedure described in Section 2.2.1.1.1. For the *Spatial Pyramid Model* we rely on a nonlinear SVM classifier (105) with an intersection kernel (56). A nonlinear SVM classifier has an increased discriminative power at the cost of an increased computational complexity compared to a linear SVM classifier (122). However, our proposed approaches to object class detection limit the number of those nonlinear classifications to a few evaluations per image by pre-selecting object hypotheses using the *Star Model*. Consequently, the *Spatial Pyramid Model* represents an object class with a discriminatively trained nonlinear SVM classifier based on a spatial pyramid representation.

2.3 Synthetic Training Data

In contrast to the majority of object class detection methods, the proposed object class detection approaches of this thesis rely on CAD models as a source of synthetic training data. As described by Liebelt in (76), both the CAD models and the rendering process suffer from imperfections which result in a gap between synthetically generated images and real world images. For example, in our case these imperfections are caused by the opaque rendering of originally transparent surfaces as illustrated in Figure 2.6 (left and center). However, in the present work we argue like Liebelt in (76) that these imperfections of synthetically generated training images are comparable with the imperfections resulting from a manually chosen database of real training images (27, 100, 129). In order to create a database of training images which should be representative for a given detection task, both processes rely on different assumptions and heuristics with respect to intra-class, pose, and background variations. But in this thesis we argue that the advantages of synthetically generated positive training images exceed their shortcomings: in contrast to real positive training images, a source of synthetic training data is able to create training images of different CAD models on demand from arbitrary viewpoints with varying lighting and background conditions. Meta data in terms of a binary segmentation mask, as illustrated in Figure 2.6 (right), and viewpoint annotations are automatically provided by the rendering process described in Section 2.3.2.
In this thesis we make use of techniques from the domain of computer graphics in order

Figure 2.6: Comparison between a real training image (left) and a synthetically generated training image (center). A byproduct of the rendering process is a binary segmentation mask (right) which is used to determine the bounding box of an object instance within the corresponding training image.

to generate positive training images for the proposed object class detection approaches. However, a detailed survey of the computer graphics domain is beyond the scope of the present thesis. For an introduction to computer graphics we refer to the book by Shirley (106). In the following sections only a brief introduction to this topic is provided by describing the properties of CAD models and the rendering process, which is used in this thesis to generate positive training images.

2.3.1 CAD Models

In the present work, we rely on object class specific databases which are established by downloading CAD models from commercial distributors such as turbosquid[1] or doschdesign[2]. We convert[3] these models into the COLLADA^{TM}[4] file format which is an XML-based scheme to represent a CAD model. Table 2.1 indicates that for each of the PASCAL VOC2006 object classes a sufficient number of CAD models is available from these commercial distributors.

In general, a CAD model is a collection of 3D points, also called vertices. Based on these vertices, polygons (normally triangles) are defined in order to describe the 3D surface of an object. In addition, for a CAD model different materials are defined and each vertex of a CAD model is connected to a material. Such a material, which consists of an ambient color coefficient c_a, a diffuse color coefficient c_d, a specular color coefficient c_s, and a specular color exponent n_s, determines how a local area of an object

[1] http://www.turbosquid.com/

[2] http://www.doschdesign.com/

[3] In this thesis, we rely on NuGraf® distributed by Okino Computer Graphics to convert or modify our CAD models (http://www.okino.com/nrs/nrs.htm/).

[4] http://www.khronos.org/collada/

reflects a light source. Some CAD models use textures in order to increase the level of detail of a rendered image. To this purpose each vertex is provided with 2D texture coordinates and during the rendering process a texture is mapped onto the 3D surface of an object based on these texture coordinates. Figure 2.7 illustrates some CAD models which are used to represent the object classes bicycle, car, and motorbike. Also see Appendix B for a visualization of all CAD models which are used in the present thesis. In Appendix B we also provide for each CAD model statistical information containing the number of vertices, the number of polygons, the number of materials, and the number of textures. Based on these statistics we assess the impact of the CAD models' quality on the detection performance (see Section 4.5.2.4).

object class	bike	bus	car	cat	cow	dog	horse	mbike	person	sheep
no. of CAD models	2115	1556	18744	785	389	953	1248	426	1850	141

Table 2.1: Number of available CAD models at turbosquid.com for the PASCAL VOC2006 object classes (accessed 2012-09-26).

Figure 2.7: Visualization of some CAD models to represent the object classes bicycle (top), car (center), and motorbike (bottom). See Appendix B for a visualization of all CAD models used in the present thesis.

2.3.2 Rendering Process

One objective of this thesis is to use techniques from the computer graphics domain in order to provide the proposed object class detection approaches with synthetically generated positive training images. To this purpose, we have implemented a rendering process based on OpenGL®[1] in conjunction with the OpenGL Shading Language in order to make use of a programmable graphics pipeline. In our case it is necessary to use a programmable graphics pipeline, since some of the described rendering techniques (e.g. shadow mapping) are not available in the fixed-function pipeline of OpenGL. In the implemented rendering process there remain differences between synthetically generated images and real world images. As shown in Figure 2.6 (left and center), these differences between a real world image and a synthetically generated image are, for example, due to the opaque rendering of originally transparent surfaces (e.g. glass panes). However, in this work we argue that it is not necessary to completely close this gap by improving the rendering quality (e.g. by using ray tracing). Instead we can bridge the gap between synthetic and real world images by suitably chosen object class representations.
In the following, we describe the techniques from the computer graphics domain implemented with OpenGL® and the OpenGL Shading Language to synthesize positive training images:

- **Blinn-Phong Lighting Model:** For our rendering process we have chosen the Blinn-Phong lighting model (12) which is a modification of the Phong lighting model (96) with a reduced computational complexity. The Blinn-Phong lighting model is an empirical model of local illumination, assuming a point light source at infinite distance. It has become the standard lighting model in most rendering processes and describes the reflection I of a light source as a combination of three components

$$I = I_{ambient} + I_{diffuse} + I_{specular}. \tag{2.5}$$

  The ambient component $I_{ambient}$ depends on the ambient color coefficient c_a of the material (see Section 2.3.1) and the ambient intensity I_a of the light source

$$I_{ambient} = c_a I_a. \tag{2.6}$$

[1] http://www.opengl.org/

The diffuse component $I_{diffuse}$ depends on the position of the light source $\vec{L}$, the surface normal $\vec{N}$ (see Figure 2.8 (left)), the diffuse color coefficient c_d of the material (see Section 2.3.1), and the diffuse intensity I_d of the light source

$$I_{diffuse} = c_d I_d \left(\frac{\vec{L} \cdot \vec{N}}{|\vec{L}| \cdot |\vec{N}|} \right). \tag{2.7}$$

The specular component $I_{specular}$ depends on the position of the light source $\vec{L}$, the position of the camera $\vec{V}$, the surface normal $\vec{N}$ (see Figure 2.8 (left)), the specular color coefficient c_s of the material (see Section 2.3.1), the specular color exponent n_s of the material (see Section 2.3.1), and the specular intensity I_s of the light source

$$I_{specular} = c_s I_s \left(\frac{\vec{N} \cdot (\vec{L} + \vec{V})}{|\vec{N}| \cdot (|\vec{L} + \vec{V}|)} \right)^{n_s} \tag{2.8}$$

By using the superposition principle the Blinn-Phong lighting model can also be extended to multiple light sources. In our implementation, the rendering process is able to handle four light sources where the number and the position of these light sources are randomly determined.

- **Antialiasing:** A main problem of synthetically generated images is that lines appear jagged because a synthesized line has to lie on the pixel grid which is just an approximation. This jaggedness is also called aliasing and different methods for reducing this detrimental effect are described in the literature. For our rendering process we use the multisampling technique which is a standard approach for antialiasing: multiple samples from neighboring pixels are used to calculate the final pixel value. The difference between an image with and without antialiasing is shown in Figure 2.8 (right). Further details on antialiasing are given in (106).

- **Shadow Mapping:** Shadows are an important element to improve the realism of a synthetically generated image. For our rendering process a shadow mapping algorithm is used which was introduced by (127). The first step of the shadow mapping algorithm is to render the 3D scene from the position of the light source and to store the depth information of the Z-buffer into a shadow map. The second step is to render the scene from the position of the camera and subsequently, the distance of the rendered points can be compared with the stored values in the shadow map. Based on this comparison, it is possible to decide if a scene point

Figure 2.8: The Blinn-Phong lighting model has become the standard in most rendering pipelines (left). The influence of antialiasing on a synthetically generated image (right).

is visible from a light source and therefore this point is illuminated or if a scene point is not visible from a light source and therefore this point has to be rendered shadowed. Examples of synthetically generated images with shadow mapping are illustrated in Figure 2.9. Details on shadow mapping are given in (106).

Figure 2.9: Example images of our rendering process for the object classes bicycle (top), car (center), and motorbike (bottom).

- **Background Variation:** In order to take into account the background variations in real world images (see Section 1.2), we render the CAD models in front of randomly selected background images which do not contain any object instances of a relevant object class. In this thesis, for the rendered images we use background images which stem from the TU Graz-02 data set (88). Examples of synthetically generated images with these real background images are shown in Figure 2.9. Note that for the proposed object class detection approaches we additionally resort to real negative training images which are provided by the benchmark data sets described in Section 2.5.

- **Binary Segmentation Mask:** As shown in Figure 2.6 (center and right), based on the OpenGL Shading Language it is possible to establish a binary segmentation mask for each generated training image. We render the CAD model from a given viewpoint without the Blinn-Phong lighting model, without the shadow mapping, and without all defined textures and materials of the CAD model. As a result, we obtain a binary segmentation mask where the white pixels indicate the object instance and the black pixels indicate the background within the corresponding training image. We use such a binary segmentation mask to determine a bounding box within the corresponding training image without any manual intervention.

2.4 Evaluation Criteria and Performance Measure

A standardized evaluation criterion and performance measure is essential to ensure that the comparison between different object class detection approaches is equitable. In this section, the naturally defined evaluation criterion in image classification (1) is explained. Subsequently, we describe how this criterion has been adapted in the past in order to obtain a standardized evaluation criterion and performance measure in the area of object class detection which is also used in the present thesis.

2.4.1 Evaluation Criterion in Image Classification

Typically, an image classification system assigns a real-valued classification score[1] to the image being classified. This classification score is compared with a defined classification

[1] In this thesis, we exclusively rely on SVM classifiers which output real-valued classification scores.

Figure 2.10: Possible evaluations in image classification based on the illustrated prediction of a classification system (from left to right): true positive (TP), false negative (FN), false positive (FP), and true negative (TN). In this case we assume that an image containing a car has a positive ground truth label (left) and an image which does not contain a car has a negative ground truth label (right). The images are taken from the PASCAL VOC2006 data set (28).

threshold which leads to a binary classification: an image is classified as positive if the assigned classification score is above the defined threshold and negative if the assigned classification score is below the defined threshold. Consequently, the comparison of a binary classification of an image against its corresponding ground truth label can have four results: true positive (TP), true negative (TN), false positive (FP), or false negative (FN). True positive means a positive labeled image is correctly classified as positive and true negative means a negative labeled image is correctly predicted as negative. False positive terms a negative labeled image which is incorrectly classified as positive and false negative describes a positive labeled image which is incorrectly predicted as negative. Figure 2.10 illustrates these four classification evaluations, based on the assumption that an image containing a car is labeled as positive and an image which does not contain a car is labeled as negative. With these definitions a true-positive-rate is defined by

$$\text{true-positive-rate} = \frac{nTP}{nTP + nFN} = \frac{nTP}{nP} \tag{2.9}$$

and a false-positive-rate is defined by

$$\text{false-positive-rate} = \frac{nFP}{nFP + nTN} = \frac{nFP}{nN}. \tag{2.10}$$

Here, nTP is the number of true positives, nFN is the number of false negatives, nFP is the number of false positives, and nTN is the number of true negatives. The number of all positive labeled images within a data set is denoted by nP and the number of all negative labeled images within a data set is denoted by nN. A receiver-operating-characteristics (ROC) curve now plots the true-positive-rate versus the false-positive-rate with a decreasing classification threshold, i.e., above this classification threshold

an image is classified as positive and below this threshold an image is classified as negative. From this curve the area-under-curve (AUC) is extracted in order to measure the overall performance of a classification system. Figure 2.11 illustrates a receiver-operating-characteristics curve with the area-under-curve as performance measure.

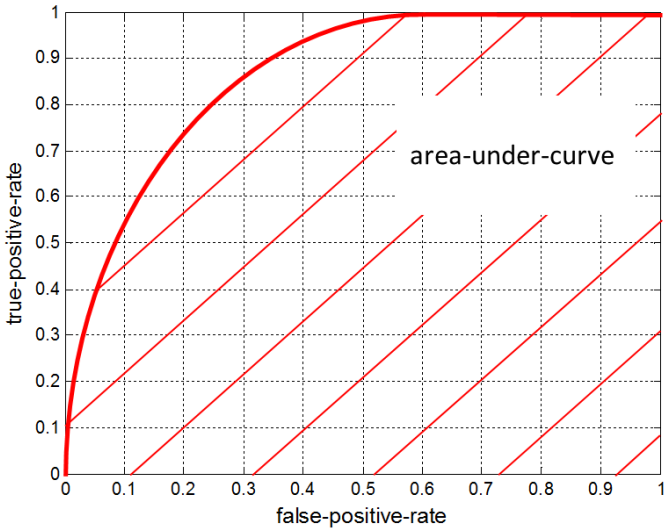

Figure 2.11: A receiver-operating-characteristics (ROC) curve is normally used in image classification. With the area-under-curve (AUC) the performance of an image classification approach is characterized.

2.4.2 Evaluation Criterion in Object Class Detection

In general, an object class detection system generates an object hypothesis, i.e., a bounding box within an image, if the corresponding detection score of this object hypothesis is above a defined detection threshold. However, in object class detection there is no natural evaluation criterion as in image classification. In contrast to image classification, we have to decide whether the bounding box of a predicted object hypothesis is close enough to a provided ground truth bounding box. For this purpose, Agarwal et al. (1) suggest an overlap criterion based on the centroid of a predicted location and the centroid of a ground truth bounding box. However, this criterion has been replaced in the past by the criterion suggested by (28): a predicted bounding box B_p is considered as correct (i.e. true positive) if the overlap o between this predicted bounding box B_p

and a ground truth bounding box B_{gt} exceeds 50%. The overlap o is defined by

$$o = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}}. \tag{2.11}$$

In contrast to image classification, multiple detections can occur, i.e., there could be several correctly predicted bounding boxes for one ground truth bounding box. As suggested in (28), such multiple detections are penalized: one detection is considered as correct (true positive) and the remaining detections are considered as false detections (false positive).

In contrast to image classification, a receiver-operating-characteristics curve is not suitable for the performance measure of an object class detection approach, since the total number of negatives nN, which is required in the definition of the false-positive-rate (see Equation 2.10), is not well defined. As described in (1), the number of negatives nN is not a property of the input but rather an internal property of the implementation of a detection system. A solution to this problem is given in (1) by the precision-recall (PR) curve. The precision of a detection system is defined by

$$\text{precision} = \frac{nTP}{nTP + nFP} \tag{2.12}$$

and the recall is given by

$$\text{recall} = \frac{nTP}{nTP + nFN} = \frac{nTP}{nP}. \tag{2.13}$$

Note that the recall is the same as the true-positive-rate as in Equation 2.9. A precision-recall curve now plots the precision versus the recall with a decreasing detection threshold. From this curve the average-precision (AP) is extracted in order to characterize the overall performance of an object class detection system. The average-precision is equal to the area under the graph. Figure 2.12 illustrates a precision-recall curve with the average-precision as performance measure.

2.5 Benchmark Data Sets

In the last few years, several benchmark data sets have been published in order to evaluate and compare different object class detection approaches. Each of these data sets has its own advantages and disadvantages, since each data set was designed for a specific detection task. In the following, we describe two state-of-the-art benchmark

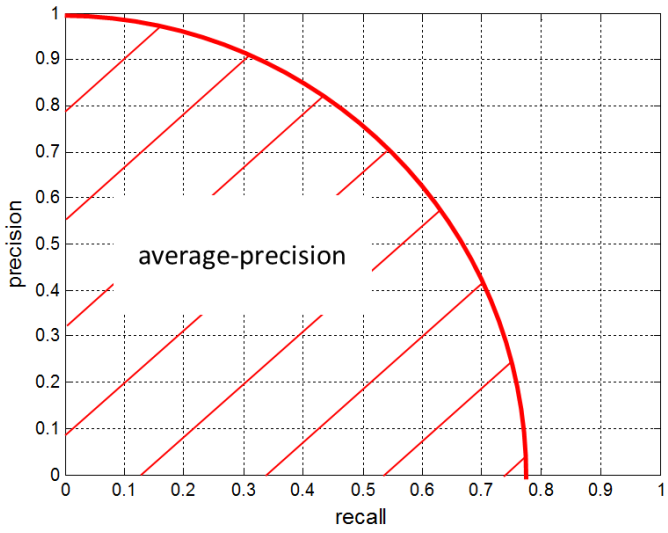

Figure 2.12: In object class detection a precision-recall (PR) curve is used. With the average-precision (AP) the performance of an object detection approach is characterized.

data sets for multi-view object class detection. Based on these two standardized data sets we introduce our Multi-Class data set.

2.5.1 3D Object Category Data Set

The 3D Object Category data set has been introduced by Savarese and Fei-Fei (100) in 2007 and contains altogether 10 different object classes: car, stapler, iron, shoe, monitor, computer mouse, head, bicycle, toaster, and cellphone. The data set was explicitly designed as a multi-view object class detection benchmark data set. The data set contains for each object class 10 different object instances. Each object instance is shown in front of a varying background from 8 different 45°-spaced azimuth angles (left, front-left, front, front-right, right, back-right, back, and back-left), 2 different elevation angles, and 3 different distances. Figure 2.13 shows for each of the 8 different azimuth angles an example image for the object classes car and bicycle. The 3D Object Category data set is the current state-of-the-art benchmark data set for multi-view object class detection and pose estimation.

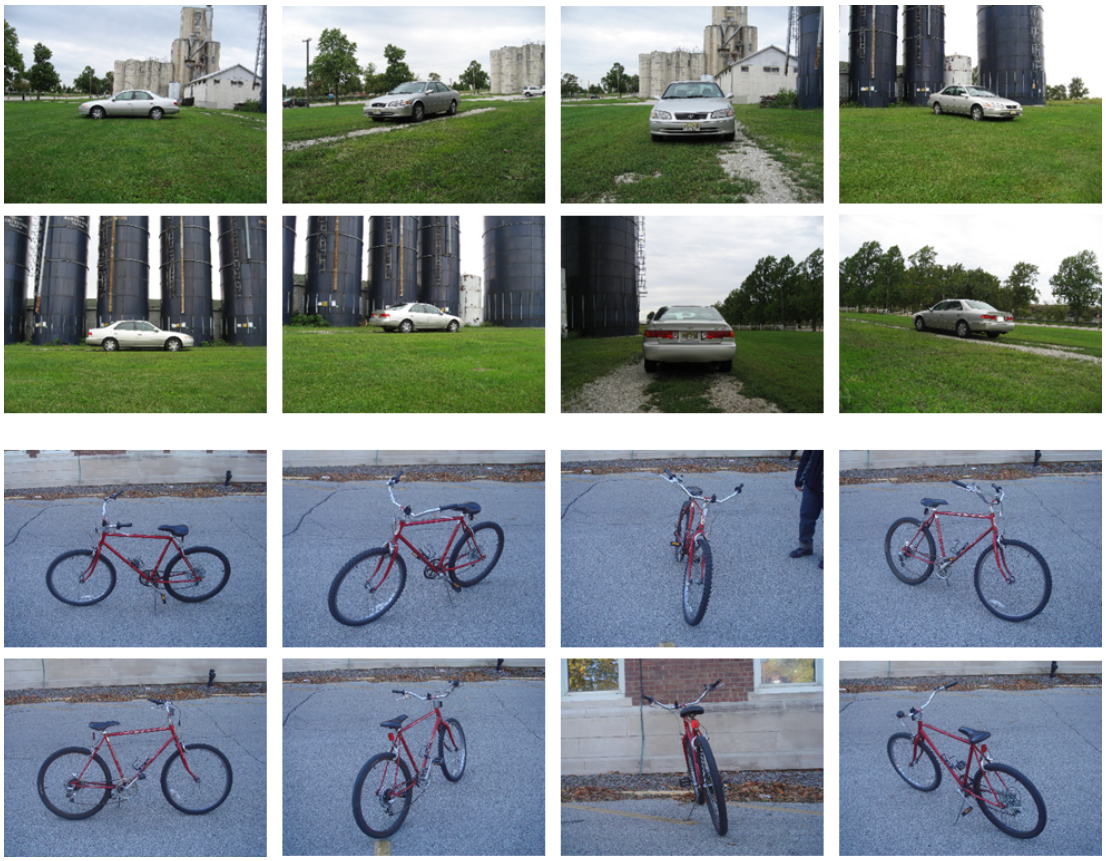

Figure 2.13: Example images from the 3D Object Category data set (100) for the classes car (top) and bicycle (bottom). Each object instance from the 3D Object Category data set is shown from 8 different 45°-spaced azimuth angles (from left to right and top to bottom): left, front-left, front, front-right, right, back-right, back, and back-left.

2.5.2 PASCAL VOC2006 Data Set

Since 2005 the PASCAL Visual Object Classes (VOC) challenge[1] is a benchmark in order to provide the computer vision community annually with a data set and standardized evaluation criterion (26). In this thesis, we rely on the PASCAL VOC2006 data set (28), since it is still a challenging data set with respect to intra-class variance, object pose and size, illumination, and occlusion. In addition, the majority of the reported detection approaches, which serve as reference to our proposed detection approaches, is evaluated on this data set up to now. The PASCAL VOC2006 data set contains 5304 real world images from the Microsoft Research Cambridge database (107) and from the

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/

photo-sharing web site flickr[1]. The data set is divided into 2618 training images and 2686 test images containing the 10 following object classes: bicycle, bus, car, cat, cow, dog, horse, motorbike, person, and sheep. Figure 2.14 shows some example images from the PASCAL VOC2006 data set.

Figure 2.14: Examples images from the PASCAL VOC2006 data set (28).

[1] http://www.flickr.com/

2.5.3 Multi-Class Data Set

In Chapter 5 of this thesis, we also address the problem of multi-class object detection. The 2D localization performance of the presented multi-class learning strategies is evaluated on three different test sets which consist of images from the 3D Object Category data set (see Section 2.5.1) and the PASCAL VOC2006 data set (see Section 2.5.2). Specifically, the following multi-class test sets are utilized:

- **Bicycle-Motorbike test set:** This test set contains 96 images from the 3D Object Category data set showing two bicycle instances from the 48 defined viewpoints of the 3D Object Category data set. In addition, we use the first 96 images from the VOC2006 motorbike test set which show only one motorbike (not labeled as 'truncated' or 'difficult'). See Appendix C for an illustration of these motorbike images. Altogether this test set contains 192 test images.
- **Bicycle-Car test set:** This test set contains 192 images from the 3D Object Category data set showing two bicycle and two car instances from the 48 defined viewpoints of the 3D Object Category data set.
- **Bicycle-Car-Motorbike test set:** This test set contains the 192 test images from the Bicycle-Car test set and the 96 motorbike images from the Bicycle-Motorbike test set. Altogether this test set contains 288 test images.

2.6 Summary

In this chapter preliminary information has been provided: an overview of the terminology and an introduction to the part-based models, which are used in this thesis, have been given. In addition, an explanation of CAD models, a description of our rendering process, an overview of standard benchmark data sets, and a description of the evaluation criterion have been provided. In the following chapters, we propose different approaches to object class detection, which exploit the advantages of part-based models in conjunction with CAD models as a source of synthetically generated training data.

Chapter 3

The Multi-View Model

In this chapter, we present our first part-based approach to multi-view object class detection which is based on a database of 3D object models and a set of real negative training images. This model allows us to detect object instances of a given object class from multiple viewpoints. In this thesis, we term this approach the *Multi-View Model*. We summarize previous work and focus on explaining the unsupervised training and detection procedure of the *Multi-View Model*. This chapter is concluded by providing the results of an experimental evaluation of the *Multi-View Model*.

3.1 Introduction

In the previous chapter, we have introduced the generative *Star Model* in Section 2.2.1 and the discriminative *Spatial Pyramid Model* in Section 2.2.2, two part-based models relying on two different learning paradigms, and we have outlined in Section 2.3.2 a rendering process in order to generate synthetic training images based on a database of CAD models. In this chapter, we propose with the *Multi-View Model* an object class detection approach which integrates these two different part-based models from Chapter 2 into one common object class detection framework. This framework is able to detect object instances of a given object class from multiple viewpoints. In addition, the *Multi-View Model* is learnt on synthetically generated positive training images and a set of real negative training images resulting in a training process which does not require any manual bounding box, object part, or viewpoint annotations.
More specifically, synthetically generated positive training images and real negative

training images are used as training sources for learning a set of viewpoint-specific object part detectors. This generated set of viewpoint-specific part detectors becomes the common component of the *Multi-View Model*. We then obtain an efficient object class detection framework as follows: based on the set of viewpoint-specific part detectors, we establish several viewpoint-specific *Star Models* and one *Spatial Pyramid Model*. During the detection procedure, the generative part of the *Multi-View Model* produces initial object hypotheses on a test image by relying on the viewpoint-specific *Star Models*. Subsequently, these initial object hypotheses are verified by the discriminative part of the *Multi-View Model*, the *Spatial Pyramid Model*.

The proposed object class detection framework of this chapter combines several advantages: first, the *Multi-View Model* does not require any manual bounding-box, viewpoint, or part annotation during the training process, since this training process relies on a database of CAD models with an automatic identification of suitable part positions. Second, the object class detection framework of the *Multi-View Model* combines the advantages of the generatively trained *Star Models* with the advantages of the discriminatively trained *Spatial Pyramid Model*. As outlined in (50, 51), a generative model such as the *Star Model* is able to deal with significant intra-class variation resulting in a high recall of an object class detection system. However, generatively trained models normally tend to produce a significant number of false-positives (51) decreasing the precision of a detection system. In contrast to a generative model, a discriminative model such as the *Spatial Pyramid Model* directly establishes a decision boundary between a set of positive and negative training samples. This leads to a classification performance which is often superior to those obtained by a generative model (51). The proposed *Multi-View Model* exploits the benefits from these different methods by relying on generative *Star Models* to produce initial object hypotheses, which are subsequently verified by the discriminative part of the *Multi-View Model*. This verification step of the *Multi-View Model* consists of one *Spatial Pyramid Model* combining all viewpoint-specific object parts into one spatial pyramid representation to fully exploit the information contained in the available set of viewpoint-specific part detectors. Consequently, the *Spatial Pyramid Model* is a multi-view representation of an object class which enables a consistent ranking of initial object hypotheses provided by the viewpoint-specific *Star Models* and additionally makes use of viewpoint symme-

tries[1] and part similarities[2] within an object class.
This chapter is structured as follows: Section 3.2 summarizes previous work on multi-view object class detection. A system overview of the proposed *Multi-View Model* is given in Section 3.3. Details on the training and the detection procedure are presented in Section 3.4 and in Section 3.5. Experimental results and a comparison of the *Multi-View Model* with other reported object class detection approaches are given in Section 3.6. This chapter is concluded with a summary in Section 3.7.

3.2 Related Work

A large amount of work has been published on object class detection. Consequently, in this section we focus on recent multi-view object class detection approaches which are related to our proposed detection approaches described in this chapter and in Chapter 4, respectively.
Most recent work on multi-view object class detection focuses on deriving a representation of an object class as a set of two-dimensional constellations of object parts for discrete viewpoints. Several authors propose sophisticated combinations of viewpoint-specific object class detectors, instead of running those discrete object class detectors independently from each other, as described in (17, 92). Thomas et al. (115), for example, suggest to link viewpoint-specific implicit shape models (74) with the approach of (41), thereby achieving an object class detection system over multiple viewpoints. In (3), part-based models for discrete viewpoints are combined by training an SVM classifier based on the detection scores of those viewpoint-specific detectors. Additionally, part-based models have been introduced and increasingly enriched with powerful machine learning techniques in conjunction with local image features in order to improve the detection performance of such a multi-view object class detection system. Originally described in (44), the idea to represent an object class as a flexible constellation of object parts is taken up by (36) who restricted the geometric structure of a part-based model to a tree (or star) instead of modeling all pairwise interactions. The approach of (36) is further extended in (34) with a discriminatively learnt object part appearance based on local image features and a mixture of multiple heuristically initialized

[1] For example, the front and back view of the object class car are symmetric viewpoints which might be visually similar.
[2] For example, the wheels within the object class car are similar object parts.

part-based models for different aspect ratios of the object class being trained, thereby increasing robustness. In (57), the approach of (34) is applied to build viewpoint-specific discriminative detectors with varying levels of supervision for viewpoint classification. The heuristic initialization of (34) is circumvented in (6) by relying on manually annotated object parts. In contrast to these part-based approaches which model a sparse set of distinct object parts, also a fixed grid-based subdivision of discrete object views into spatial regions has been suggested in (53, 78) to detect object instances within an image by the spatial consistency of those fixed defined object regions. Based on hand-segmented training images, Hoiem et al. (65) introduce a 3D layout conditional random field model to roughly align physical parts across object instances at different viewpoints. Savarese and Fei-Fei (100) establish discriminative object regions composed of local images features and homographic transformations between those regions in order to form a viewpoint-independent 3D object category model. Based on training images with known viewpoint labels, (112) describe a generative approach which represents an object class as a constellation of object parts and establishes correspondences of those parts among different viewpoints. A similar approach to (112) is proposed in (110). However, the training of (110) requires a video clip in order to establish an initial model. In (4), the implicit shape model from (74) is extended to obtain a 3D implicit shape model for 3D transformations and self occlusion relying on manually marked feature points. More recently, Mei et al. (83) propose a statistical manifold modeling approach which also represents an object class as a constellation of object parts and considers the trade-off between object class and viewpoint variation in a more principled way,although their approach requires video sequences for both training and testing. While being robust, the output of those described part-based approaches is mostly restricted to a few discrete viewpoints, due to the limited availability of annotated real training images or they require time-consuming video sequences for training. Alternatively, CAD models have been suggested as training data in order to circumvent this restriction of real training images, since synthetic training images can be generated on demand from arbitrary viewpoints. Early approaches relying on CAD models exclusively use textureless models in order to establish a detection system for a specific object instance (55, 59, 60, 62, 63). Recently, the idea of using CAD models for object class detection approaches has regained attention. Heisele et al. (61), for example, use textureless 3D models in order to systematically evaluate the influence of the generated

training set on the performance of a detection system. Yan et al. (130) establish a 3D feature model by relying on a homographic framework to map 2D features onto an existing CAD model. In (79), local image features are derived from synthetically rendered 3D models to evaluate the global consistency of a 2D detection with respect to a 3D geometry. Based on textureless renderings, (109) establish an object class representation consisting of viewpoint-specific shape models. The approach of (109) is extended in (132) with a global deformable 3D wireframe based on manual annotations of 3D correspondences. Liebelt and Schmid (78) propose a part-based approach for object class detection and pose estimation which builds for discrete viewpoints the 3D geometry of an object class from a database of 3D models and learns the part appearance from an annotated database of real training images. In (82, 97) CAD models are used to learn an object class detection system for pedestrians. More recently, (94) extend the deformable part-based models of (34) to include both pose estimation and 3D parts based with a heuristic initialization.
In this chapter and in Chapter 4, respectively, we propose part-based approaches to multi-view object class detection and build on the idea of using pre-built and textured CAD models, thereby combining the main advantages of the two domains described above. More specifically, photometric object parts are learnt in an unsupervised way from a database of CAD models and a set of real negative images. Subsequently, the established object parts are efficiently combined in a detection framework relying on part-based models with two different learning paradigms.

3.3 Overview

Figure 3.1 gives an overview of the *Multi-View Model* which is presented in this chapter. The *Multi-View Model* relies on a database of 3D object models for the object class that should be detected and a set of real negative images as training data source. See Figure 2.7 for some model examples of different object classes or Appendix B for a visualization of all CAD models which are used in this thesis. In contrast to other work (109, 132), the *Multi-View Model* does not require any semantically labeled 3D parts, which CAD designers sometimes assign to parts of the model geometry during the creation process (e.g. 'wheel' or 'car door'), since we have frequently found these manual labels to be inconsistent. By means of the rendering process, which is described

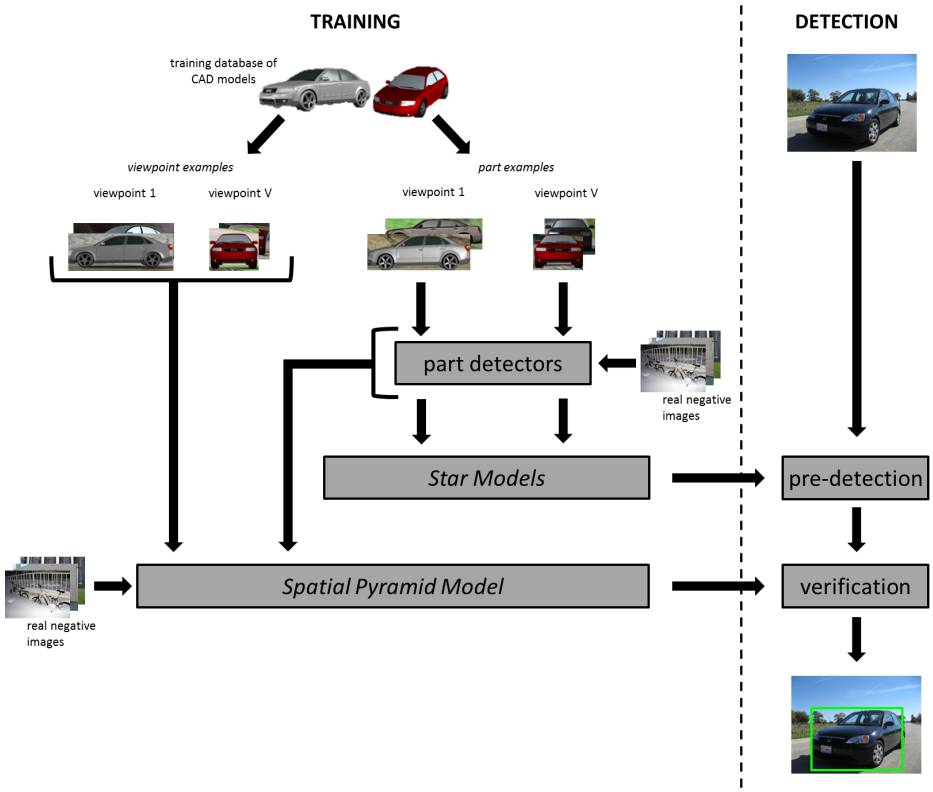

Figure 3.1: Overview of the *Multi-view Model* which integrates several generatively trained *Star Models* and one discriminatively trained *Spatial Pyramid Model* into one common object class detection framework. Training (left side): viewpoint-specific part detectors are trained by means of CAD models and a set of real negative images. These part detectors are used to establish a set of viewpoint-specific *Star Models* and one multi-view *Spatial Pyramid Model*. Detection (right side): Object hypotheses, which are generated by the viewpoint-specific *Star Models* in a pre-detection step, are verified by the multi-view *Spatial Pyramid Model*.

in Section 2.3.2, the CAD models from the training database are used to generate for each defined viewpoint v ($1 \leq v \leq V, v \in \mathbb{N}$) of the *Multi-View Model* two independent sets of positive training images: the *part examples* and the *viewpoint examples*.

As illustrated in Figure 3.1, the *part examples* in conjunction with a set of real negative images are used as training source for the discriminative learning of viewpoint-specific part detectors. The objective of this learning step is to automatically identify a spatial part layout, which describes the characteristic appearance of an object class under a given viewpoint v. For each defined viewpoint v of the *Multi-View Model*, individual

part locations are automatically chosen as those regions which consistently show dominant gradients across all 3D object models of an object class (see Figure 3.3). While the selected part regions do not necessarily have a semantic meaning, this process ensures that the appearance of the chosen object parts is sufficiently structured for the training process of an object part by capturing the dominant gradients across all object models of an object class. The determined part positions of the established spatial part layout are used in a subsequent step to train viewpoint-specific part detectors from the determined object part locations of the *part examples* (see Figure 3.5). As shown in Figure 3.1, the generated set of viewpoint-specific part detectors is the common component of the *Multi-View Model* for both, the generatively trained viewpoint-specific *Star Models* and the discriminatively trained *Spatial Pyramid Model*. More details on the generation of those viewpoint-specific part detectors are given in Section 3.4.2.

Based on the *part examples* and the viewpoint-specific part detectors, a *Star Model* is established for each defined viewpoint of the *Multi-View Model*. These viewpoint-specific *Star Models* are the generative part of the *Multi-View Model* and are able to determine object hypotheses on an image which have a high likelihood of containing an object instance of the trained object class. Further details on this training step are given in Section 3.4.3. However, as outlined in (51) and (50) generative models such as the viewpoint-specific *Star Models* tend to produce a significant number of false positives. In addition, due to the differences in layout and appearance discriminativity of the different defined viewpoints of the *Multi-View Model*, the scores of the *Star Models* do not allow for a comparison between the defined viewpoints which, however, is necessary to rank the generated set of initial object hypotheses.

In order to verify the initial object hypotheses which are detected by the above described viewpoint-specific *Star Models* and to establish a comparable ranking, we suggest to use the discriminative *Spatial Pyramid Model* as a joint encoding of the responses of all generated part detectors in a detected image region, i.e., in an initial object hypothesis. Consequently, the *Spatial Pyramid Model* is the discriminative part of the *Multi-View Model* and incorporates all the information contained in the generated set of viewpoint-specific part detectors. The training examples for the *Spatial Pyramid Model* are determined by applying the generated set of part detectors on the established *viewpoint examples* and real negative images (see Figure 3.8). On each training image we apply the generated set of all viewpoint-specific part detectors, resulting in a set of detector

responses that include real detections (e.g. responses of front view part detectors on an actual front view example) as well as 'hallucinated' detections (e.g. responses of front view part detectors on a side view example). The useful contribution of 'hallucinated' detections for the overall detection of objects relies on viewpoint symmetries and part similarities and is illustrated in Figure 3.7, where the consistent response of a front view detector (red bounding box) contributes to the evidence of a side view detection (green bounding box). Details on the *Spatial Pyramid Model* are presented in Section 3.4.4. As illustrated in Figure 3.1, during the detection process the generative part of the *Multi-View Model* generates in each test image a set of initial object hypotheses by relying on the viewpoint-specific *Star Models*. Subsequently, the full set of generated part detectors is applied to these object hypotheses. The resulting histogram based descriptors, encoding all the individual part detector responses, are then verified by the discriminatively trained multi-view *Spatial Pyramid Model*. A non-maxima suppression discards all those object hypotheses which overlap (see Equation 2.11) by more than 50% with a higher-scoring object hypothesis to obtain the final detections on a test image. More details on the detection process are given in Section 3.5.

3.4 Training

This section outlines the necessary training steps for the *Multi-View Model* and starts with the use of CAD models and a set of real negative images as training source. Afterwards, the unsupervised training approach for the viewpoint-specific part detectors is explained. These part detectors serve as the common component for both, the generative and the discriminative part of the *Multi-View Model*. The training steps of these two parts of the *Multi-View Model* are also described in this section.

3.4.1 Training Examples

As shown in Figure 3.1, the *Multi-View Model*, presented in this chapter, is trained on a database of 3D object models and a set of real negative images. See Figure 2.7 for some CAD models of different object classes or Appendix B for a visualization of all CAD models which are used in this thesis. As outlined in Section 2.3, the use of 3D object models as training source allows to generate positive training images of an object class from arbitrary viewpoints. In addition, for each synthetically generated

training image the bounding box of the object instance within the training image and its viewpoint label are automatically known without any manual intervention. Based on the described rendering process of Section 2.3.2, we generate for each defined viewpoint v $(1 \leq v \leq V, v \in \mathbb{N})$ of the *Multi-View Model* two independent sets of training images at a predefined training scale[1]: the *part examples* and the *viewpoint examples* (see Figure 3.1). In addition to the synthetically generated positive training images, we rely on a set of background images which does not contain any object instance of the object class being trained. To this purpose, we modify the PASCAL VOC2006 training data set to establish this set of negative training images[2].

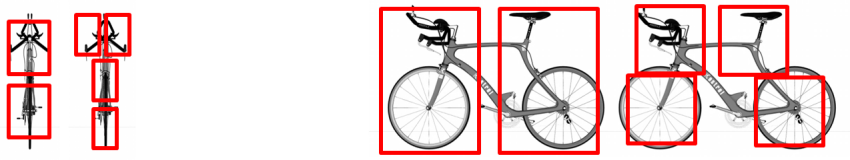

Figure 3.2: An adequate subdivision of an object class into object parts depends on the viewpoint. For example, for the front view of the bicycle class (left) a subdivision into two object parts seems to be adequate (in contrast to four object parts), whereas the side view of the bicycle class (right) might require a more fine-grained part subdivision with four parts (in contrast to two object parts).

3.4.2 Object Parts

Learning the appearance of an object class must take into account large intra-class and viewpoint variations as well as partial occlusions and background (see Section 1.2). In addition, when dealing with part-based object class detection, object parts have to be chosen such that they are suitable for the training of classifiers, i.e., object part detectors. A manual annotation of these part positions (19, 36, 103) is time consuming and additionally, there is no guarantee that the manually selected object parts are suitable, i.e., sufficiently discriminative, for the training process which is described in Section 2.2.1.1.1. As a consequence, some authors propose a fixed layout of object parts (53, 78) or suggest unsupervised approaches to localize suitable part positions.

[1]Note that the *part examples* and the *viewpoint examples* at a predefined training scale differ in width and height. The average width and the average height are given in experimental section of this chapter.
[2]The PASCAL VOC2006 training data set is modified such that it does not contain any object instance of the object class under training.

For example, in (37) an object class is decomposed into the fixed number of six object parts, selecting the part positions such that the resulting patches capture a maximum of the object structure. However, a multi-view object class detection approach requires the choice of part positions of different viewpoints with different sizes, aspect ratios, and appearance characteristics. The method of (37) results in a spatial part layout, where each object part has approximately the same size and each viewpoint is subdivided into the same number of object parts. However, this might not be a suitable approach for the multi-view representation of all object classes. For those viewpoints where an object class covers a smaller area the chosen patches could be too small and therefore might not contain sufficient structure to be suitable for a discriminative classifier, i.e., an SVM classifier. As shown in Figure 3.2, the front view of the bicycle class is a good example of a viewpoint for which a subdivision into one or two parts is adequate, whereas a bicycle side view may requires a more fine-grained part subdivision with four object parts.

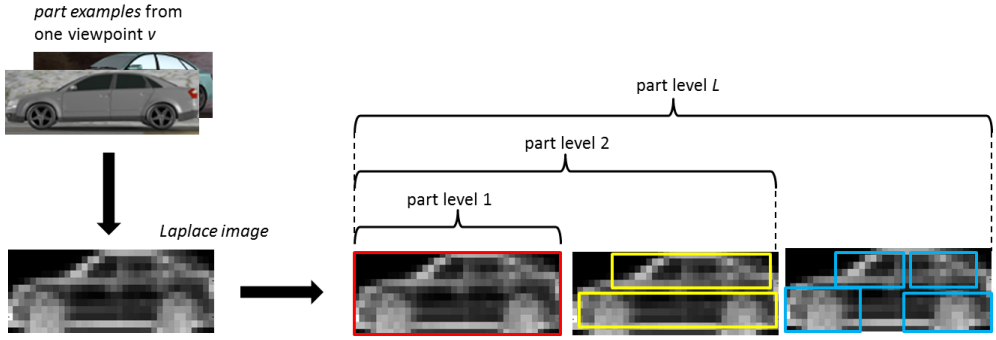

Figure 3.3: Based on the *part examples* for a specific viewpoint v (here side view) a *Laplace image* is established from which a spatial part layout with L part levels is derived.

3.4.2.1 Spatial Part Layout

For the *Multi-View Model* we rely on the approach of (37) in order to determine suitable part positions within the *part examples* for a defined viewpoint v. However, we circumvent the above described problem of an adequate subdivision for different viewpoints by deriving a spatial part layout which decomposes an object class for a given viewpoint into L ($1 \leq l \leq L, l \in \mathbb{N}$) part levels.
The concept of this object class decomposition for a given viewpoint v (e.g. side view)

is shown in Figure 3.3. We scale the *part examples* at the predefined training scale to their average width and their average height, apply a Laplacian filter mask, and average over the filtered examples. The resulting image is divided into small quadratic regions[1] and within each region we calculate the corresponding mean value. Finally, we obtain a *Laplace image* for a given viewpoint which indicates by an area of high values a structured region within the *part examples* (e.g. wheels). For establishing a *Laplace image*, it is necessary that the *part examples* for a given viewpoint have similar aspect ratios and that the structured regions of an object class for a given viewpoint occur in similar areas within the *part examples*. Based on the *Laplace image* for a given viewpoint, we employ the following procedure to determine a spatial part layout with altogether L part levels: for the first part level (i.e. $l = 1$) we define a rectangle which captures the entire *Laplace image* (see the red rectangle within Figure 3.3). As described in (37), for the subsequent part levels (i.e. $2 \leq l \leq L$) we define on each part level l an area a such that the object class is decomposed into 2^{l-1} object parts and $a2^{l-1}$ equals about 70% of the area of the *Laplace image*. By relying on the approach given in (32), we greedily select a rectangle with area a that captures the most structured region within the *Laplace image*; the selected region is masked out in the *Laplace image* and the procedure is repeated at each part level l until the positions of all 2^{l-1} object parts are determined. Decomposing an object class for a given viewpoint finally leads to a spatial part layout with N viewpoint-specific part positions

$$N = \sum_{l=1}^{L} 2^{l-1} = 2^L - 1 \tag{3.1}$$

on L part levels. Since the *Multi-View Model* is defined for V different viewpoints, we finally obtain V spatial part layouts with altogether VN normalized part positions.

3.4.2.2 Training of Part Detectors

Once the spatial part layout for each defined viewpoint of the *Multi-View Model* is identified with the method described in Section 3.4.2.1, it is possible to determine the corresponding part positions within each image of the *part examples*. The concept of this projection is illustrated in Figure 3.4. We assume that a normalized part position

[1] The size of those quadratic regions is equivalent to the HOG cell size which is used for the *Multi-View Model*; see Section 3.6.

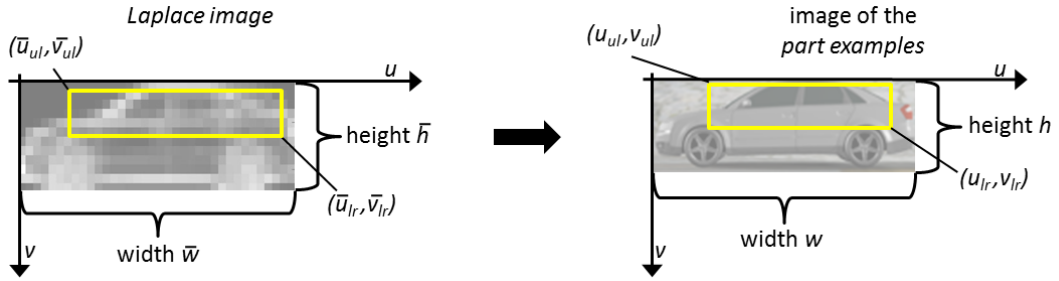

Figure 3.4: We project the normalized part position, i.e., $\bar{u}_{ul}$, $\bar{v}_{ul}$, $\bar{u}_{lr}$, and $\bar{v}_{lr}$, within the *Laplace image* into the corresponding part position, i.e., u_{ul}, v_{ul}, u_{lr}, and v_{lr}, within an image of the *part examples*.

within the *Laplace image* for a given viewpoint is specified by the upper left corner and the lower right corner of the bounding box, i.e., $\bar{u}_{ul}$, $\bar{v}_{ul}$, $\bar{u}_{lr}$, and $\bar{v}_{lr}$. We further assume that the *Laplace image* has a width $\bar{w}$ and a height $\bar{h}$ and an image of the *part examples* has a specific width w and a specific height h. Then, the position of an object part within an image of the *part examples*, i.e., u_{ul}, v_{ul}, u_{lr}, and v_{lr}, is given by

$$\begin{aligned} u_{ul} &= \bar{u}_{ul}\frac{w}{\bar{w}} \\ v_{ul} &= \bar{v}_{ul}\frac{h}{\bar{h}} \\ u_{lr} &= \bar{u}_{lr}\frac{w}{\bar{w}} \\ v_{lr} &= \bar{v}_{lr}\frac{h}{\bar{h}}. \end{aligned} \tag{3.2}$$

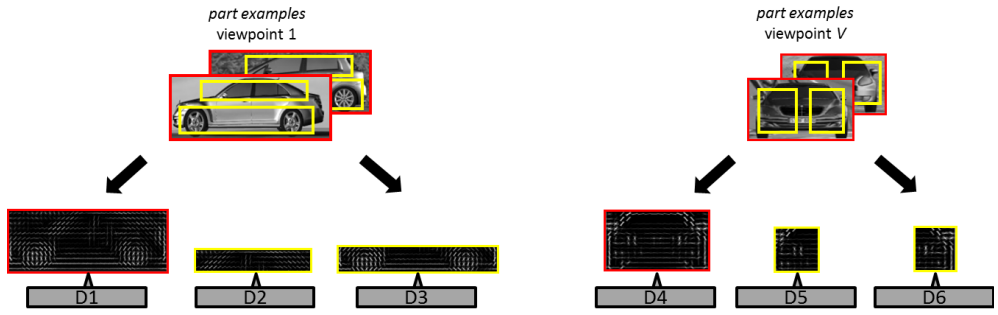

Figure 3.5: The determined object part positions (here red and yellow bounding boxes) within the *part examples* at the predefined training scale are used to train viewpoint-specific part detectors (here D1 to D3 and D4 to D6).

Based on Equation 3.2 it is possible to determine for each spatial part layout the part positions within the corresponding *part examples* at the predefined training scale. Those determined part positions are subsequently used to train viewpoint-specific part detectors by relying on the procedure described in Section 2.2.1.1.1. We generate for each object part specific training samples and resize those positive training samples of an object part to their corresponding average width and average height. We resort to the HOG descriptor of (21) to encode the appearance of an object part. For each object part a separate linear SVM classifier is learnt by relying on the 'bootstrapping' procedure described in Section 2.2.1.1.1: the normalized and part-specific training samples serve as positive training examples and the negative examples are initially chosen randomly from the negative training images. After the initial training run, the SVM classifier is refined on an extended training set which has been augmented with the false positives of the initial SVM classifier. Finally, VN discriminatively learnt object part detectors are obtained, each representing the appearance of an object part for a defined viewpoint; for an example see D1 to D3 and D4 to D6 in Figure 3.5.

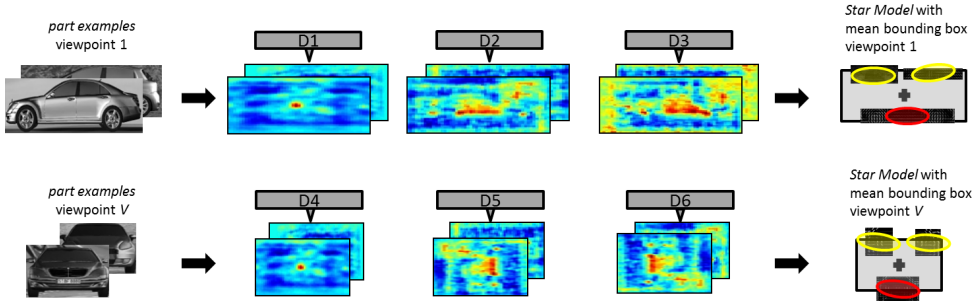

Figure 3.6: The viewpoint-specific *part examples* at the predefined training scale and the corresponding part detectors (here D1 to D3 and D4 to D6) are used to train viewpoint-specific *Star Models*.

3.4.3 Viewpoint-Specific Star Models

For the generative part of the *Multi-View Model*, which is used to produce a set of initial object hypotheses in a test image, we establish for each defined viewpoint of the *Multi-View Model* a generatively trained *Star Model*. As shown in Figure 3.6, we rely on the viewpoint-specific part detectors of Section 3.4.2 and the *part examples* at the predefined training scale to establish those viewpoint-specific *Star Models*. As described

in Section 2.2.1.1.2, we apply all generated part detectors of a specific viewpoint to the *part examples* at the predefined training scale which represent the same viewpoint. We model the spatial uncertainty for each object part detector, i.e., the location of an object part, with a Gaussian mixture model. According to Section 2.2.1.1.2, we choose the center of the *part examples* as reference in order to predict a suitable bounding box during the detection procedure, by projecting the average size of the *part examples* (i.e. the average width and the average height at the predefined training scale) as mean bounding box into an input image.

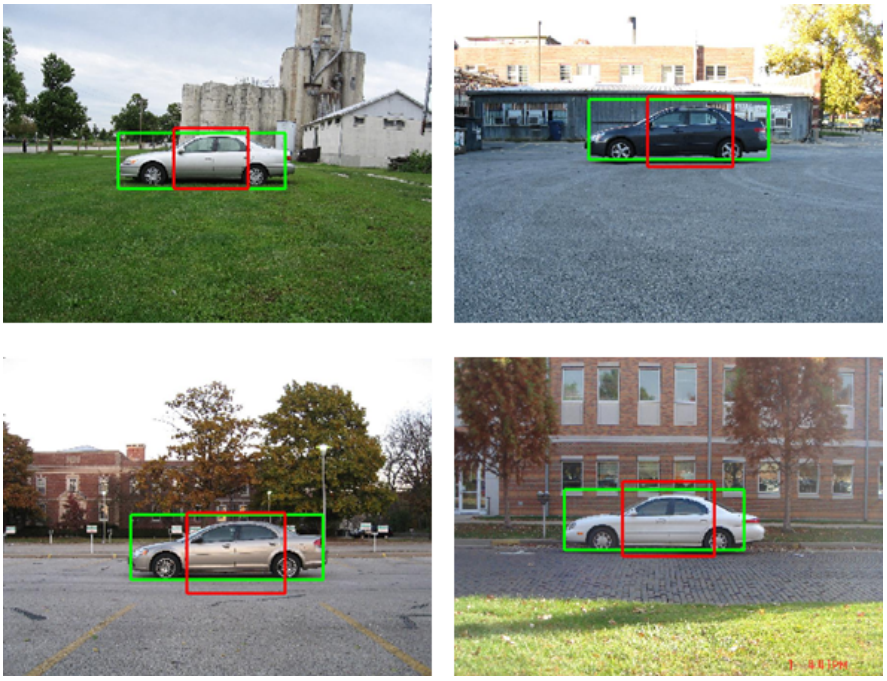

Figure 3.7: The *Spatial Pyramid Model* builds on the combination of all viewpoint-specifically trained object part detectors. For example, detector D4 (trained on front view images) also provides a consistent response on side view images (red bounding box); instead of discarding these detector responses, the *Spatial Pyramid Model* exploits their information content in a joint encoding.

3.4.4 Spatial Pyramid Model

As described in Section 3.4.2, viewpoint-specific part detectors are derived in an unsupervised way. In the following, we introduce the *Spatial Pyramid Model*, a multi-view object class representation which jointly encodes the responses of all individual object

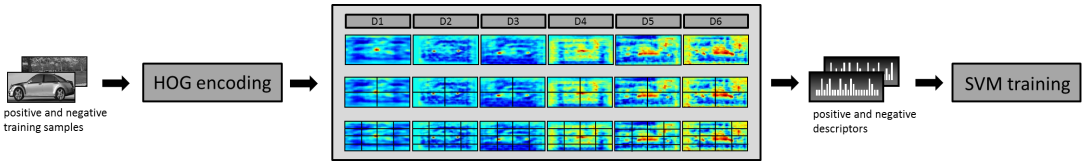

Figure 3.8: The *Spatial Pyramid Model* builds on all viewpoint-specific part detectors (here D1 to D6) and encodes the spatial layout of their responses into a spatial pyramid representation. Based on a set of positive and negative training samples a non-linear SVM classifier with an intersection kernel is trained.

part detectors. The *Spatial Pyramid Model* allows to consistently rank detections for the different defined viewpoints of the *Multi-View Model*. Note that the scores of the viewpoint-specific *Star Models* of Section 3.4.3 alone do not allow for such a ranking, mainly due to their differences in layout and appearance discriminativity of the different defined viewpoints.

Due to viewpoint symmetries and part similarities, the trained object part detectors of Section 3.4.2 sometimes locate object parts at wrong positions or in viewpoints where these object parts are not actually visible. Still, these 'hallucinated' part detections often appear consistently within an object class. An example of such a 'hallucinated' part detection is given in Figure 3.7: as expected, an object part detector, which has been trained on example images for cars from a side view, provides consistent 'true' responses on images showing the object class from this viewpoint (green bounding box); however, another object part detector, which has been trained on example images for cars from a front view, also provides consistent 'hallucinated' (false positive) responses on these images (red bounding box). We suggest exploiting this kind of readily available additional information with the *Spatial Pyramid Model* in order to combine the 'true' as well as the 'hallucinated' detector responses for a more discriminative multi-view object class representation.

The idea of the *Spatial Pyramid Model* is shown in Figure 3.8. For each training instance HOG features with the same layout as in Section 3.4.2 are computed densely. Each encoded training image is classified by all viewpoint-specific object part detectors. As described in Section 2.2.2, we follow the approach of (70) and rely on a spatial pyramid representation to encode the responses for each object part detector within the area of a training instance. For each part detector response a spatial pyramid with K $(1 \leq k \leq K, k \in \mathbb{N})$ levels is defined which results in a fixed hierarchy of rectangular

sub windows. Within each defined sub window we sum up all the positive detection responses, i.e., all detection responses above zero, to obtain one real-valued number. We concatenate those real-valued numbers of all defined sub windows for all object part detectors and obtain a descriptor for a given training image. We choose a spatial pyramid with a quadratic subdivision resulting in an object class representation with

$$d = VN \sum_{k=1}^{K} 4^{k-1} = VN \frac{1}{3}(4^K - 1) \tag{3.3}$$

dimensions for altogether VN object part detectors. Given the *viewpoint examples* for all defined viewpoints at the predefined training scale as positive training examples and a set of negative training examples, a nonlinear SVM classifier with an intersection kernel (56) is trained based on the 'bootstrapping' procedure described in Section 2.2.1.1.1.

3.5 Detection

This section describes the two detection steps of the *Multi-View Model*, the pre-detection step based on the generative part of the *Multi-View Model*, i.e., the viewpoint-specific *Star Models* to obtain initial object hypotheses, and the discriminative part of the *Multi-View Model*, i.e., the *Spatial Pyramid Model*, in order to verify those initial object hypotheses and to establish a comparable ranking.

3.5.1 Pre-Detection

In order to identify regions of interest, i.e., initial object hypotheses which potentially contain an object instance of the trained object class, we rely on the viewpoint-specific *Star Models* of Section 3.4.3 in conjunction with the detection procedure of Section 2.2.1.2. The generatively trained *Star Models* provide on a test image a fixed number[1] of object hypotheses (see Figure 3.9 (left)) with a detection score and a viewpoint label for each generated object hypothesis. Although the *Star Models* are able to deal with significant intra-class variation, resulting in a high recall on a benchmark data set, the detection scores of the *Star Models* alone do not allow a consistent ranking of the generated object hypotheses. The reason is that for a given viewpoint each *Star Model* is trained independently of all other viewpoints and relies on different layout and

[1] In our experiments each *Star Model* generates 8 object hypotheses on a test image.

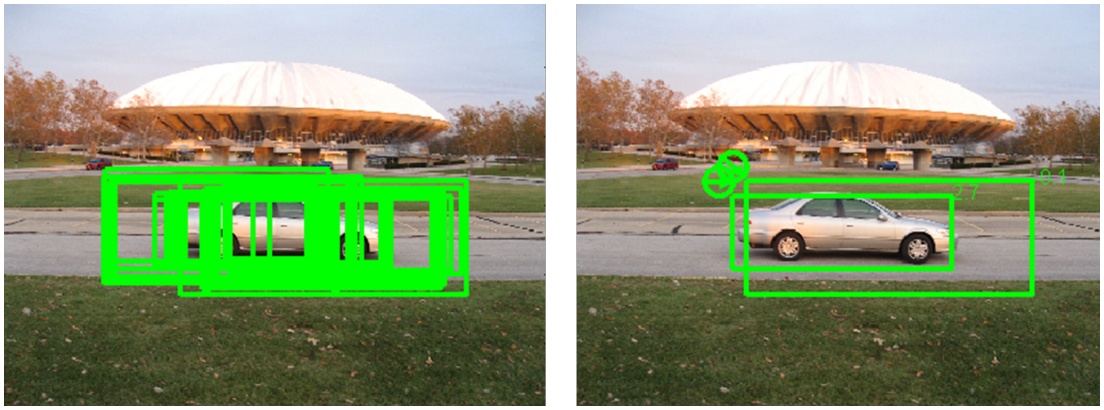

Figure 3.9: The detection procedure of the *Multi-View Model* consists of two steps: the pre-detection step is based on the generative and viewpoint-specific *Star Models* in order to establish a set of initial object hypotheses (left). The verification step relies on the discriminative part of the *Multi-View Model*, the *Spatial Pyramid Model*. Consequently, each initial object hypothesis is classified by the *Spatial Pyramid Model*. Based on the resulting classification scores of the *Spatial Pyramid Model*, we apply a non-maximum suppression to avoid multiple and overlapping detections. Subsequently, we obtain the final detections on a test image (right) which are provided with a detection score based on the *Spatial Pyramid Model* and an approximate pose label based on the viewpoint-specific pre-detection step.

appearance characteristics; see Section 3.6.2.1 for experimental results. Consequently, in the following verification step we build on the classification performance of the *Spatial Pyramid Model*, which relies on the object part detectors of all defined viewpoints of the *Multi-View Model*, in order to obtain a normalized and comparable detection score for all generated object hypotheses within a test image.

3.5.2 Verification

The final detection result consists of a consistent and comparable scoring of the obtained initial object hypotheses based on the *Spatial Pyramid Model* of Section 3.4.4. To this purpose, each initial object hypothesis, which is generated by a viewpoint-specific *Star Model*, is scaled to the average size (i.e. the average width and the average height) of the corresponding *viewpoint examples* at the predefined training scale. The responses of all object part detectors in the scaled area of an object hypothesis are encoded by the spatial pyramid representation described in Section 3.4.4 and subsequently classified by the *Spatial Pyramid Model* to obtain a final detection score for the corresponding

object hypothesis. Since the detection process of the *Multi-View Model* can result in multiple overlapping object hypotheses a non-maximum suppression retains only high scoring bounding boxes and discards those bounding boxes covered by a higher-scoring bounding box with an overlap (see Equation 2.11) of more than 50%.
As a result, each final detection on a test image is provided with a detection score based on the *Spatial Pyramid Model* and an approximate pose label based on the viewpoint-specific pre-detection step of the *Multi-View Model* (see Figure 3.9 (right)).

3.6 Evaluation

In this section, we outline the experimental results we achieve with the proposed *Multi-View Model*. We evaluate the *Multi-View Model* on the 3D Object Category data set of Section 2.5.1 and on the PASCAL VOC2006 data set of Section 2.5.2 for the object classes car and bicycle.

3.6.1 Training Setup

For generating a set of training images in order to learn the viewpoint-specific part detectors, the corresponding *Star Models*, and the *Spatial Pyramid Model*, we rely on positive training images rendered from CAD models which are available from commercial distributors, notably turbosquid.com or doschdesign.com (see Appendix B for a visualization of all CAD models which are used in the present thesis). We use 24 car models and all 8 bicycle models from the available databases for the respective object classes. The reason for taking 24 car models instead of all 25 is that it is essential that the corresponding *part examples* have a comparable aspect ratio for a specific viewpoint (see Section 3.4.2.1). Therefore, model 25 of the car database (see Appendix B) is not considered in the following experiments due to its clearly different aspect ratio compared to the remaining car models.
For the *Multi-View model* we define the following five different viewpoints (i.e. $V = 5$): left (i.e. azimuth=0° and elevation=0°), front-left (i.e. azimuth=45° and elevation=0°), back-left (i.e. azimuth=315° and elevation=0°), front (i.e. azimuth=90° and elevation=0°), and back (i.e. azimuth=270° and elevation=0°). Example images of these defined viewpoints are given in Figure 3.10. The respective symmetric views are covered by applying the approach to the horizontally mirrored images. Based on these five viewpoints we

Figure 3.10: Example images for the different defined viewpoints (from left to right): left, front-left, back-left, front, and back.

generate the *part examples* and the *viewpoint examples* for both object classes. The *part examples* are generated from the fixed azimuth angle and the fixed elevation angle for each defined viewpoint. The *viewpoint examples* are generated by a uniform variation in azimuth direction (i.e. $\pm 9°$) and a uniform variation in elevation direction (i.e. $+9°$) in order to increase the robustness towards small viewpoint variations. We also vary the light conditions for each defined viewpoint within the *part examples* and *viewpoint examples* in order to take into account this variation in real world images. Altogether 2640 training images (i.e. 1200 *part examples* and 1440 *viewpoint examples*) for the object class car and 880 training images (i.e. 400 *part examples* and 480 *viewpoint examples*) for the object class bicycle are generated; details on the generated positive training images are given in Table 3.1 for the object class car and in Table 3.2 for the object class bicycle.

car (24 CAD models)	left	front-left	back-left	front	back
average width	147 pixels	126 pixels	130 pixels	67 pixels	66 pixels
average height	51 pixels	51 pixels	55 pixels	50 pixels	54 pixels
azimuth	0°	45°	315°	90°	270°
elevation	0°	0°	0°	0°	0°
no. of light variations	10	10	10	10	10
no. of all *part examples*	1200				
average width	168 pixels	143 pixels	148 pixels	82 pixels	85 pixels
average height	64 pixels	71 pixels	71 pixels	72 pixels	71 pixels
azimuth	$[-9°, 0°, 9°]$	$[36°, 45°, 54°]$	$[306°, 315°, 324°]$	$[81°, 90°, 99°]$	$[261°, 270°, 279°]$
elevation	$[0°, 9°]$	$[0°, 9°]$	$[0°, 9°]$	$[0°, 9°]$	$[0°, 9°]$
no. of light variations	2	2	2	2	2
no. of all *viewpoint examples*	1440				

Table 3.1: Details on the generated positive training images for the object class car.

In order to train the linear SVM classifiers for the object parts as well as the nonlinear SVM classifier for the *Spatial Pyramid Model*, negative training examples are drawn

bicycle (8 CAD models)	left	front-left	back-left	front	back
average width	96 pixels	72 pixels	69 pixels	31 pixels	25 pixels
average height	59 pixels	65 pixels	61 pixels	67 pixels	63 pixels
azimuth	$0°$	$45°$	$315°$	$90°$	$270°$
elevation	$0°$	$0°$	$0°$	$0°$	$0°$
no. of light variations	10	10	10	10	10
no. of all *part examples*	400				
average width	95 pixels	70 pixels	68 pixels	32 pixels	27 pixels
average height	59 pixels	66 pixels	65 pixels	71 pixels	68 pixels
azimuth	$[-9°, 0°, 9°]$	$[36°, 45°, 54°]$	$[306°, 315°, 324°]$	$[81°, 90°, 99°]$	$[261°, 270°, 279°]$
elevation	$[0°, 9°]$	$[0°, 9°]$	$[0°, 9°]$	$[0°, 9°]$	$[0°, 9°]$
no. of light variations	2	2	2	2	2
no. of all *viewpoint examples*	480				

Table 3.2: Details on the generated positive training images for the object class bicycle.

from the PASCAL VOC2006 training data set[1]. The part appearance is built on the HOG implementation of (37) with a HOG cell size of 4 pixels; we choose a spatial part layout with $L = 4$ part levels resulting in $N = 15$ part detectors per viewpoint. For the *Spatial Pyramid Model* we choose a spatial pyramid representation with $K = 3$ levels and a quadratic subdivision. For testing we choose an image pyramid with 10 levels in an octave.

As described in Section 2.4.2, we evaluate the performance of our *Multi-View Model* with respect to 2D ground truth bounding boxes by relying on the detection quality criterion suggested by (28): a predicted bounding box is considered as correct if the overlap between the predicted bounding box and a ground truth bounding box exceeds 50%. If several bounding boxes are predicted in the same image area, only one detection is considered as correct and the remaining detections are considered as false positives.

3.6.2 3D Object Category Data Set

On the 3D Object Category data set we assess the contribution of the generative part of the *Multi-View Model*, i.e., the viewpoint-specific *Star Models*, and the discriminative part of the *Multi-View Model*, i.e., the *Spatial Pyramid Model*. In addition, we evaluate the *Multi-View Model* with respect to 2D localization, pose estimation, and robustness to partial occlusions.

[1]The PASCAL VOC2006 training data set is modified such that it does not contain any object instance of the object class under training.

3.6.2.1 Contribution of the Star Models and the Spatial Pyramid Model

In order to demonstrate the contribution of the generative part of the *Multi-View Model*, we exemplarily apply the pre-detection step of Section 3.5.1, which consists of the viewpoint-specific *Star Models*, on the entire car test set (i.e. containing all 480 test images) and the entire bicycle test set (i.e. containing all 480 test images). Based on the generated object hypotheses for each test image, we calculate the corresponding recall for both test sets. We determine how often on each test set the overlap between one generated object hypothesis, i.e., a predicted bounding box, and the ground truth bounding box within a test image exceeds 50%. As mentioned above, the generative and viewpoint-specific *Star Models* are able to deal with significant intra-class variation which results in a high recall of the detection system. As shown in Table 3.3, the generated object hypotheses of the *Star Models* achieve with 99.0% on the entire 3D Object Category car data set and 97.3% on the entire 3D Object Category bicycle data set a significantly high recall on these two test sets.

	3D Object Category car data set	**3D Object Category bicycle data set**
recall	99.0%	97.3%

Table 3.3: The generative part of the *Multi-View Model* achieves with 99.0% on the entire 3D Object Category car data set and with 97.3% on the entire 3D Object Category bicycle data set a high recall by relying on the generative and viewpoint-specific *Star Models* of the pre-detection step.

In order to demonstrate the contribution of the proposed *Spatial Pyramid Model* of Section 3.4.4, we evaluate the *Multi-View Model* exemplarily on the entire car test set (i.e. containing all 480 test images) and the entire bicycle test set (i.e. containing all 480 test images), once with and once without the discriminative *Spatial Pyramid Model* as the verification step of the detection process. Without the *Spatial Pyramid Model* we exclusively rely on the detection score provided by the viewpoint-specific *Star Models* of Section 3.5.1 in conjunction with the non-maximum suppression described in Section 3.5.2. The results are given in Figure 3.11. Omitting the *Spatial Pyramid Model* as verification step of the *Multi-View Model* (blue curves with 72.2% for the object class car and 31.0% for the object class bicycle) results in an average-precision, which is significantly below the precision obtained with the proposed *Spatial Pyramid Model* (red curves with 82.0% for the object class car and 79.4% for the object class bicycle), since

the generative *Star Models* tend to produce a large number of false-positives (especially the *Star Models* for the front view and the back view of the object class bicycle). These two experiments indicate that the proposed *Multi-View Model* exploits the advantages from both the generative *Star Models*, i.e., dealing with significant intra-class variation which results in a high recall[1], and the discriminative *Spatial Pyramid Model*, i.e., superior classification performance, by integrating these two part-based model with different learning paradigms into one common object class detection framework.

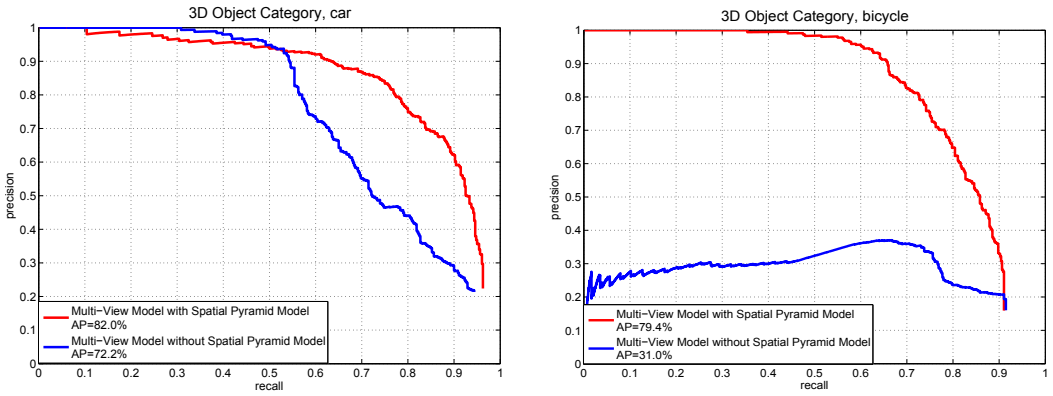

Figure 3.11: The *Spatial Pyramid Model* increases the average-precision of the *Multi-View Model*. For example, on the entire 3D Object Category car data set (left) the detection performance of the *Multi-View Model* increases from 72.2% without the *Spatial Pyramid Model* (blue curve) to 82.0% with the *Spatial Pyramid Model* (red curve) and on the entire 3D Object Category bicycle data set (right) the detection performance increases from 31.0% (blue curve) to 79.4% (red curve).

3.6.2.2 2D Localization

In order to compare the 2D localization performance of the *Multi-View Model* on the 3D Object Category data set to other reported detection approaches, we follow the test

[1]In addition to dealing with significant intra-class variation, the generative *Star Models* reduce the number of all possible object hypotheses to a few object hypotheses per image without compromising the recall of the *Multi-View Model* (see Table 3.3). For example, a test image of the *3D Object Category data set* with a size of 400x300 pixels, an image pyramid with 10 levels in an octave, and a HOG cell size of 4 pixels result in more than $2{\cdot}10^5$ possible object hypotheses for a specific viewpoint of an object class. In our experiments each viewpoint-specific *Star Model* reduces those $2 \cdot 10^5$ possible object hypotheses to 8 object hypotheses which are subsequently verified by the computational more expensive *Spatial Pyramid Model*.

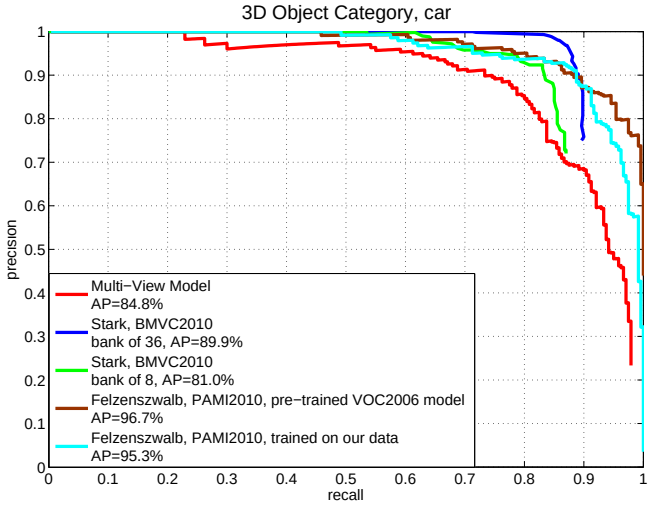

Figure 3.12: Precision-recall curves of the *Multi-View Model* (red curve) on the 3D Object Category car data set compared to other reported results and the state-of-the-art detection approach of (34).

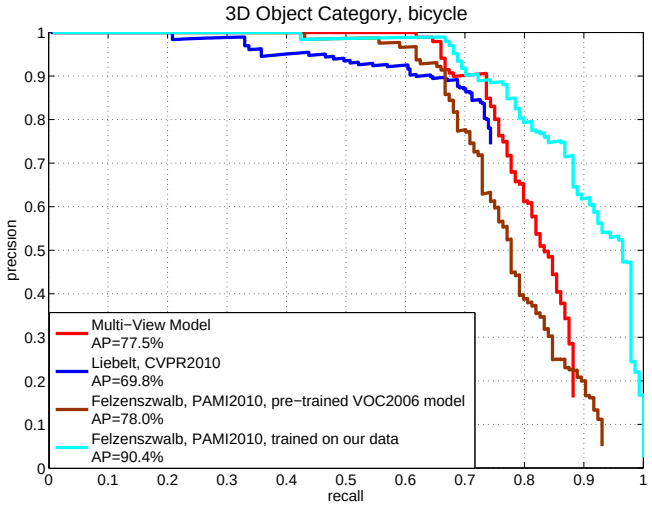

Figure 3.13: Precision-recall curves of the *Multi-View Model* (red curve) on the 3D Object Category bicycle data set compared to other reported results and the state-of-the-art detection approach of (34).

protocol of (109) for the object class car and the test protocol of (78) for the object class bicycle. Note that the test protocol of (109) for the object class car and the test protocol of (78) for the object class bicycle define a subset of test images and therefore

differ from the test setup for the experiments in Section 3.6.2.1, which are evaluated on the entire car test set and the entire bicycle test set. In Figure 3.12, we compare the result of the *Multi-View Model* on the 3D Category car data set to the approach of (109). As can be seen, with 84.8% the proposed *Multi-View Model* (red curve) is comparable to the results achieved by (109). Note that the best result of (109) (blue curve with 89.9%) is based on a set of 36 viewpoint-specific models with more than 400 trained object parts; when using only a comparable number of 8 viewpoint models, we outperform their average-precision (green curve with 81.0%) due to a higher recall. The precision-recall curve obtained with the *Multi-View Model* on the 3D Category bicycle data set is given in Figure 3.13. On this test set we compare to the approach of (78). The *Multi-View Model* achieves with 77.5% (red curve) on the bicycle test set a higher average-precision than the multi-view approach of (78) (blue curve with 69.8%).
We also compare the *Multi-View Model* against the current state-of-the-art object class detection approach of (34) using their pre-trained VOC2006 model provided as part of *voc-release4* (33). As shown in Figure 3.12 and in Figure 3.13, the approach of (34) achieves with 96.7% (brown curve) on the car test set a higher average-precision than the *Multi-View Model* and with 78.0% (brown curve) a comparable average-precision on the bicycle test set. We also evaluate the method of (34) based on our synthetically generated positive training images. To this purpose, we train and evaluate for both object classes a detection model with the recommended settings of 3 components and 8 parts per component. We use the approach of (34), which is provided as part of *voc-release4* (33), in conjunction with the synthetically generated *viewpoint examples*[1] as positive training images and the PASCAL VOC2006 training data set as negative training images. The results indicate with 95.3% (cyan curve in Figure 3.12) on the car test set and with 90.4% (cyan curve in Figure 3.13) on the bicycle test set a gap between the *Multi-View Model* and the current state-of-the-art object class detection approach of (34). Some successful detection results of the *Multi-View Model* on the 3D Object Category data set are shown for the object class car in Figure 3.20 and for the object class bicycle in Figure 3.21. Figure 3.24 depicts some failed detection results of

[1]When we try to use the *part examples* and the *viewpoint examples* jointly as positive training images for learning a detection model with the approach of (34), we get the error 'Not enough space' on a 64bit-machine with 24GB RAM during the training process. Consequently, we use only the *viewpoint examples* as positive training images.

the *Multi-View Model* for the object class car and the object class bicycle. The failed detections on the 3D Object Category data set are mainly caused by sub detections within an object instance or a nonrigid geometry.

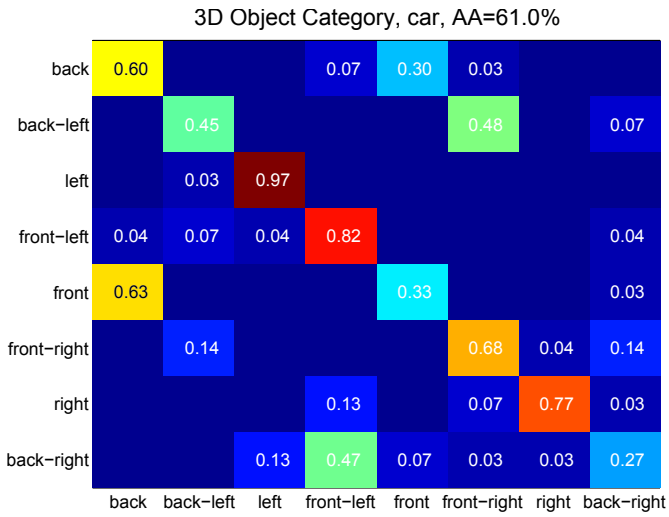

Figure 3.14: Confusion matrix (rows: ground truth, columns: estimates) for the 3D Object Category car data set. Based on the viewpoint-specific *Star Models* the *Multi-View Model* is able to predict an approximate pose label.

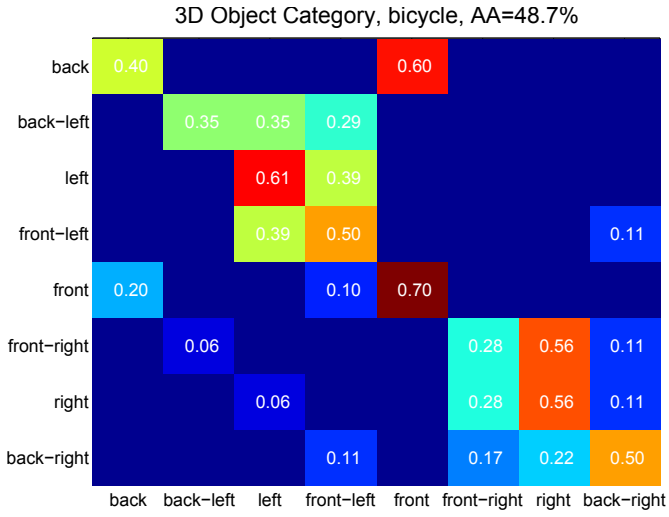

Figure 3.15: Confusion matrix (rows: ground truth, columns: estimates) for the 3D Object Category bicycle data set. Based on the viewpoint-specific *Star Models* the *Multi-View Model* is able to predict an approximate pose label.

3.6.2.3 Pose Estimation

In Section 3.5.1 we mention that the *Multi-View Model* is able to provide an approximate pose label based on the pre-detection step, i.e., the viewpoint-specific *Star Models*. Figure 3.14 and Figure 3.15, respectively, show the resulting confusion matrices on the car test set and the bicycle test set for classifying all true positive detections of Section 3.6.2.2 into the 8 azimuth angles defined by the 3D Object Category data set (see Section 2.5.1). For cars, we observe that neighboring viewpoints are rarely confused. Confusion for cars is more pronounced for opposing views due to the viewpoint symmetries inherent in the car class. For example, 30.0% of the back views are classified as front views. For bicycles, we observe that confusion is primarily pronounced for neighboring viewpoints and for the front and back views. For example, 39.0% of the left views are classified as front-left views. With an average-accuracy of 61.0% for the object class car and an average-accuracy of 48.7% for the object class bicycle, the pose estimation of the *Multi-View Model* performs worse than the reported results of (109) with an average-accuracy of 80.5% for cars and of (78) with an average-accuracy of 75.0% for bicycles. However, the pose estimation of the *Multi-View Model* relies exclusively on the pose label provided by the viewpoint-specific *Star Models* of the pre-detection step. Since the object parts of these *Star Models* are selected in an unsupervised way which does not take into account the inter-viewpoint discriminativity of the object parts such a behavior of the *Multi-View Model*, i.e., confusion of neighboring and opposing viewpoints, cannot be avoided.

3.6.2.4 Occlusion

In this experiment we assess the quality of the *Multi-View Model* in the presence of partial occlusion. To this purpose, we modify the entire 3D Object Category car data set containing all 480 test images and generate a test set with artificial partial occlusions: 30% of the annotated ground truth for each test image is replaced by a white area. Some example images of this modified test set are given in Figure 3.16. In this experiment the object class detector with the same settings as in Section 3.6.2.2 is applied to the modified data set, i.e., without any retraining or adaptation. In order to compare the performance of our part-based approach, i.e., the *Multi-View Model*, to detection methods which do not use object parts, we have implemented a baseline approach which

is based on the method of (21). This baseline approach is trained on the *viewpoint examples* which are described in Section 3.6.1; it consists of viewpoint-specific HOG descriptors, each representing the entire object class for a defined viewpoint, which are classified by linear SVM classifiers[1]. We use a sliding window approach and rely on the non-maximum suppression step of Section 3.5.2 to combine these viewpoint-specific classifier responses.

Figure 3.16: Example images of the modified 3D Object Category car data set in order to assess the quality of the *Multi-View Model* in the presence of partial occlusions. In each image 30% of the annotated ground truth is replaced by a white area.

As can be seen in Figure 3.17, with 57.2% compared to 20.0%, *the Multi-View Model* (red curve) outperforms the baseline approach (green curve) which relies on a global description of the object class. Since the object class representation of the *Multi-View Model* is based on several object parts with different sizes and due to the chosen spatial part layout with $L = 4$ part levels, partial occlusions have less effect on the overall description of the object class than on the baseline approach.

We also assess the influence of the chosen spatial part layout on the overall detection performance of the *Multi-View Model*. To this purpose, we exemplarily evaluate a *Multi-View Model* with a spatial part layout consisting of three part levels ($L = 3$) and a *Multi-View Model* with a spatial part layout consisting of two part levels ($L = 2$). For training these models we rely on the same training images and the same training settings of Section 3.6.1, i.e., a HOG cell size with 4 pixels and a spatial pyramid

[1]Note that this baseline approach is inherently included in the *Multi-View Model* by the trained object part detectors which cover for each defined viewpoint at the part level $l = 1$ the entire object class.

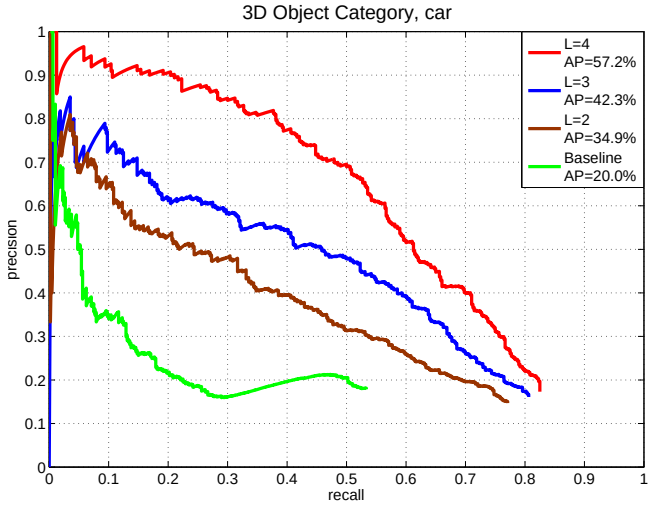

Figure 3.17: The quality of the *Multi-View Model* (red curve) under partial occlusion is shown in comparison to a baseline approach (green curve) which is based on the method of (21). A *Multi-View Model* based a spatial part layout with fewer part levels results in a lower average-precision (blue and brown curves).

representation with $K = 3$ levels and quadratic subdivision. The detection performance of these *Multi-View Models* are also shown in Figure 3.17 (blue and brown curve). As expected, the average-precision is reduced when decreasing the part levels due to the lack of information of the small object parts. Note that with two part levels the corresponding *Multi-View Model* still performs better than the baseline approach, although the reduction in performance is considerable when compared to a spatial part layout with $L = 4$ part levels. Consequently, for the *Multi-View Model* a spatial part layout with sufficiently many part levels is necessary to achieve increased robustness against partial occlusions.

3.6.3 PASCAL VOC2006 Data Set

The precision-recall curves obtained with the *Multi-View Model* on the PASCAL VOC2006 data set for the object classes car and bicycle are given in Figure 3.18 (red curve) and Figure 3.19 (red curve), respectively. For both data sets we provide the best performing approaches of the PASCAL challenge 2006 (28) (blue curves), the best performing approaches of the PASCAL challenge 2007 on the 2006 test set (29) (cyan curves), and the most recent multi-view approaches of (110) (green curves) and (95) (brown curves).

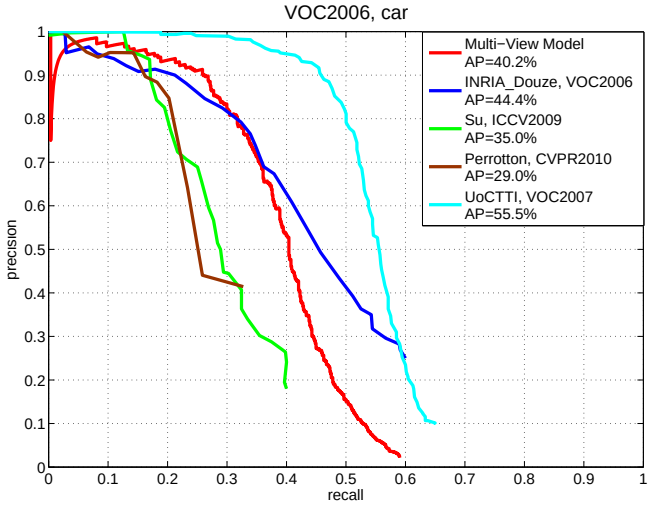

Figure 3.18: Precision-recall curves of the *Multi-View Model* (red curve) for the object class car on the PASCAL VOC2006 data set compared to other reported results.

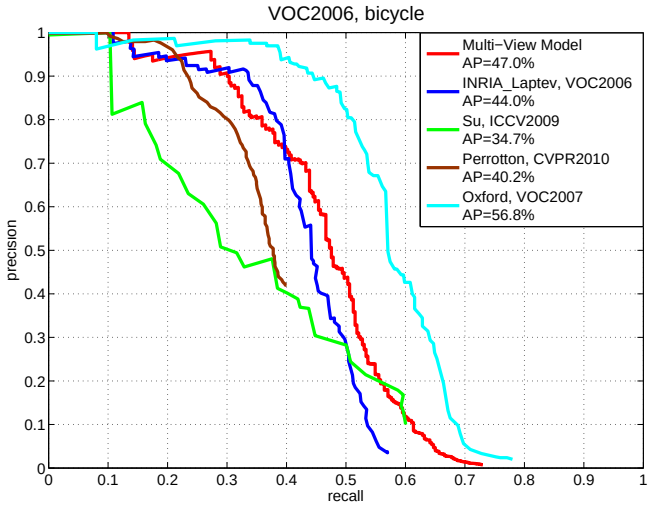

Figure 3.19: Precision-recall curves of the *Multi-View Model* (red curve) for the object class bicycle on the PASCAL VOC2006 data set compared to other reported results.

With 40.2% on the car test set and 47.0% on the bicycle test set the *Multi-View Model* achieves a higher average-precision than these two multi-view approaches, despite being trained on a different (i.e. synthetically generated positive training images) data set. We observe that the appearance variations within the car test set are more pronounced

than those within the bicycle class, which might be the reason for the observed performance difference of the *Multi-View Model*: while the chosen bicycle CAD models and the corresponding *part examples* and *viewpoint examples* are sufficient to represent these appearance variations of the bicycle test set, the car CAD models and the corresponding *part examples* and *viewpoint examples* seem to be not representative enough. Some successful detection results of the *Multi-View Model* on the PASCAL VOC2006 data set are shown for the object class car in Figure 3.22 and for the object class bicycle in Figure 3.23. Figure 3.25 depicts some failed detection results of the *Multi-View Model* for both object classes. The failed detections on the PASCAL VOC2006 data set are mainly caused by sub detections within other object instances or a nonrigid geometry.

3.7 Summary

In this chapter, we have presented the *Multi-View Model* which is a part-based approach to multi-view object class detection based on a database of 3D object models and a set of real negative training images. The main contribution of this approach is the integration of the generative *Star Model* and the discriminative *Spatial Pyramid Model* into one common object class detection framework. In addition, the *Multi-View Model* does not require any manual bounding box, object part, or viewpoint annotations during the training process. The training process relies on a database of 3D object models to generate positive training images with an automatic identification of suitable part positions within these positive training images. However, this identification of suitable part positions is independent for each defined viewpoint of the *Multi-View Model* and is based on a simple heuristic: object parts are chosen as those regions which consistently show dominant gradients across all 3D object models of an object class. It is assumed that the aspect ratios of the CAD models from the training database are similar for a given viewpoint and that the structured regions within an object class occur in similar areas within the synthetically generated positive training images. In addition, the pose estimation of the *Multi-View Model* relies exclusively on the pose label provided by the viewpoint-specific *Star Models* of the pre-detection step. As a result, the 2D localization performance of the *Multi-View Model* can compete with other reported object class detection approaches, however, there is a performance gap to the current

state-of-the-art detection approach of (34). Also the pose estimation performance of the *Multi-View Model* is not comparable to other reported results. In the next chapter, we outline an approach which addresses the shortcomings of the *Multi-View Model* and thus is able to close the gap to the current state-of-the-art detection approach of (34).

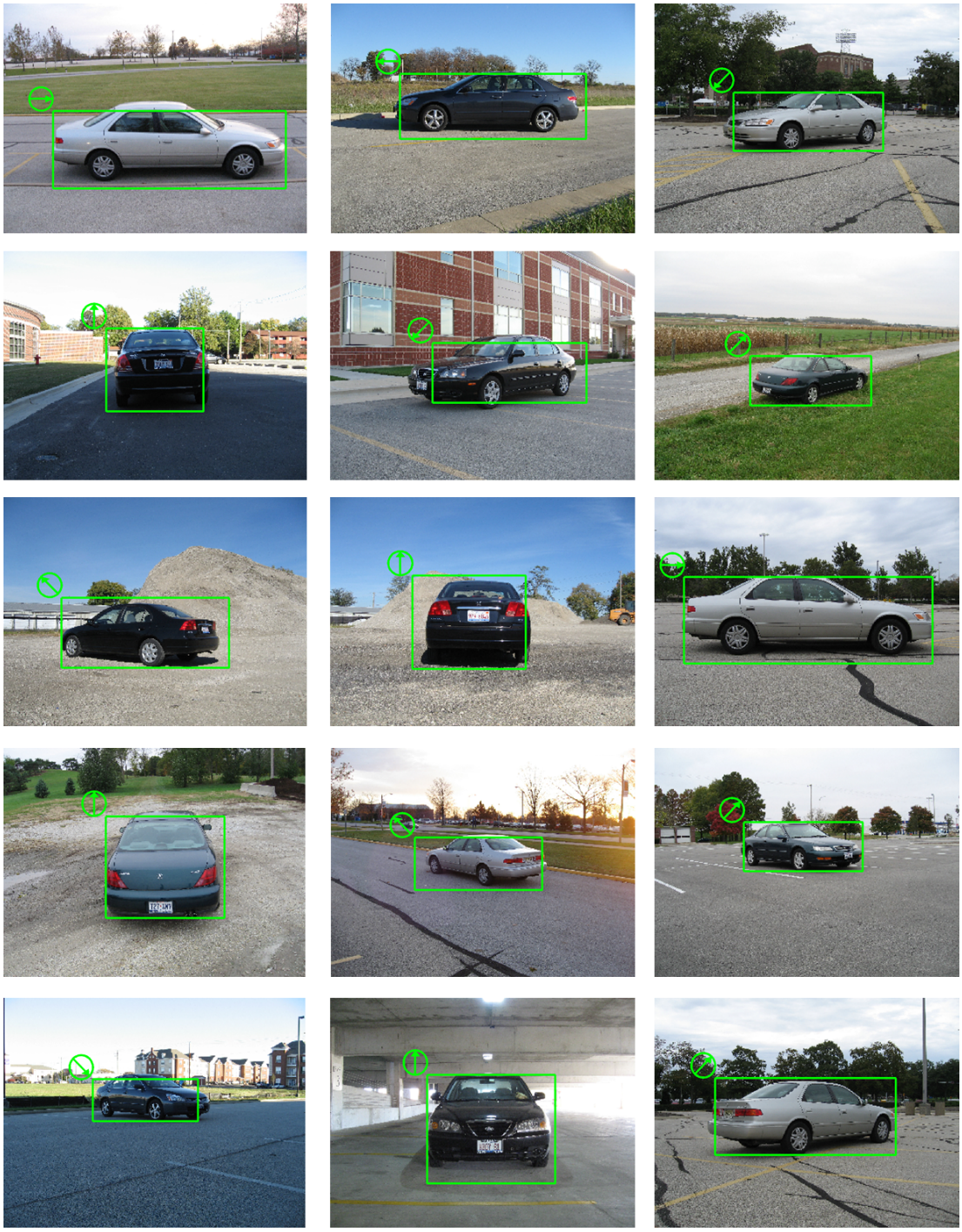

Figure 3.20: Some successful detection results of the *Multi-View Model* on the 3D Object Category car data set. Note that the *Multi-View Model* also provides an approximate pose label based on the viewpoint-specific *Star Models*. This pose label is indicated for each detection by an arrow. We show only the highest scored detection per image.

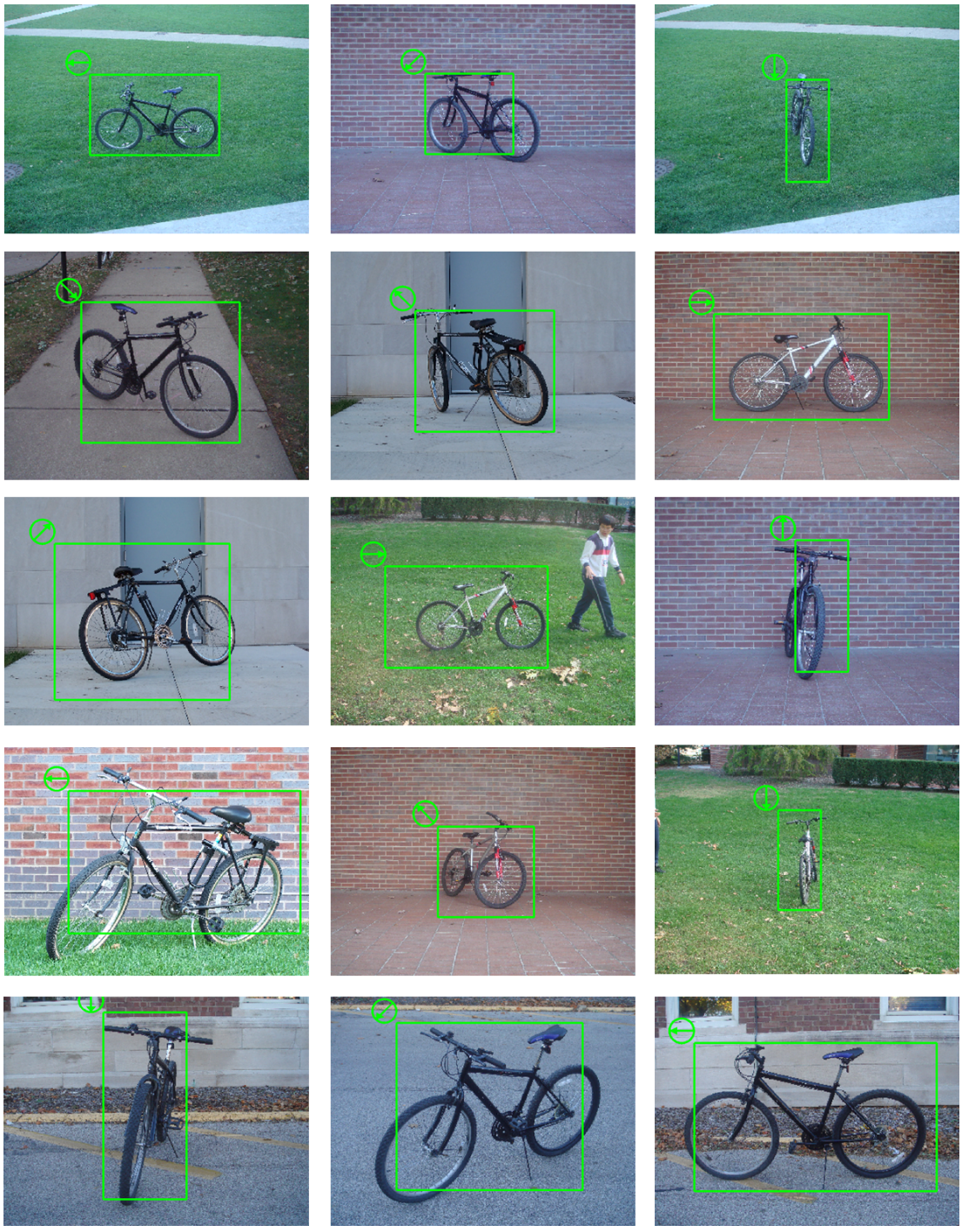

Figure 3.21: Some successful detection results of the *Multi-View Model* on the 3D Object Category bicycle data set. Note that the *Multi-View Model* also provides an approximate pose label based on the viewpoint-specific *Star Models*. This pose label is indicated for each detection by an arrow. We show only the highest scored detection per image.

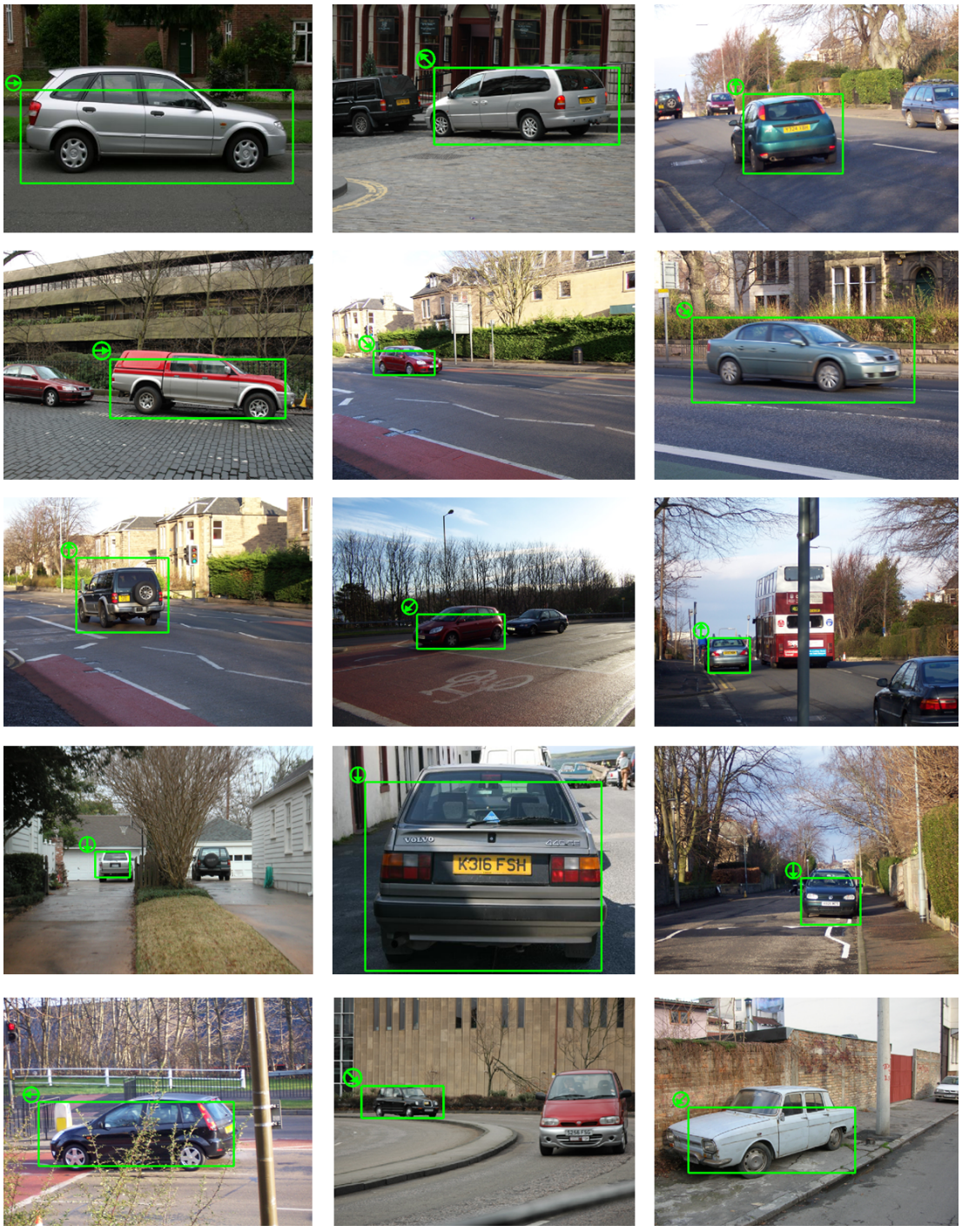

Figure 3.22: Some successful detection results of the *Multi-View Model* for the object class car on the PASCAL VOC2006 data set. Note that the *Multi-View Model* also provides an approximate pose label based on the viewpoint-specific *Star Models*. This pose label is indicated for each detection by an arrow. We show only the highest scored detection per image.

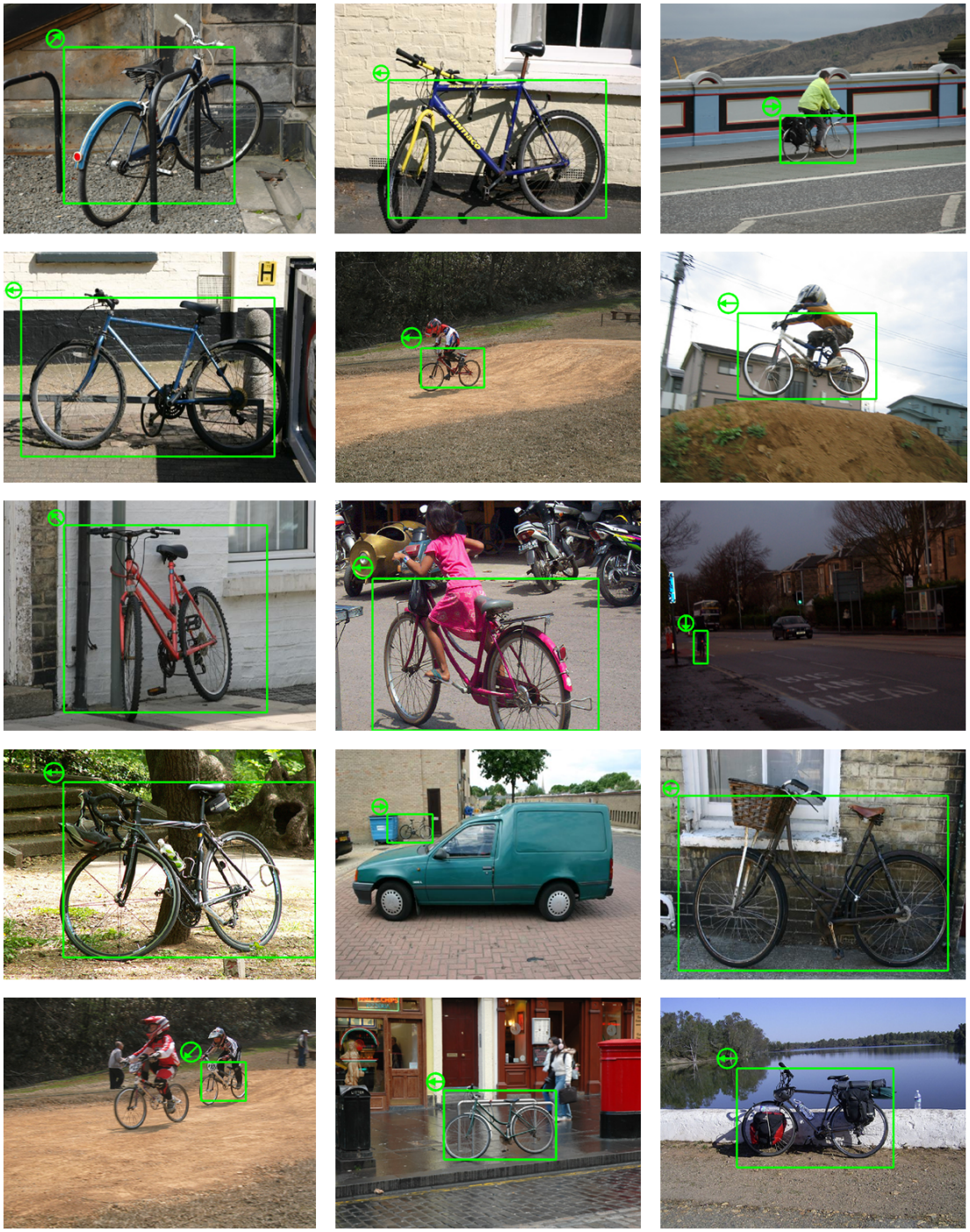

Figure 3.23: Some successful detection results of the *Multi-View Model* for the object class bicycle on the PASCAL VOC2006 data set. Note that the *Multi-View Model* also provides an approximate pose label based on the viewpoint-specific *Star Models*. This pose label is indicated for each detection by an arrow. We show only the highest scored detection per image.

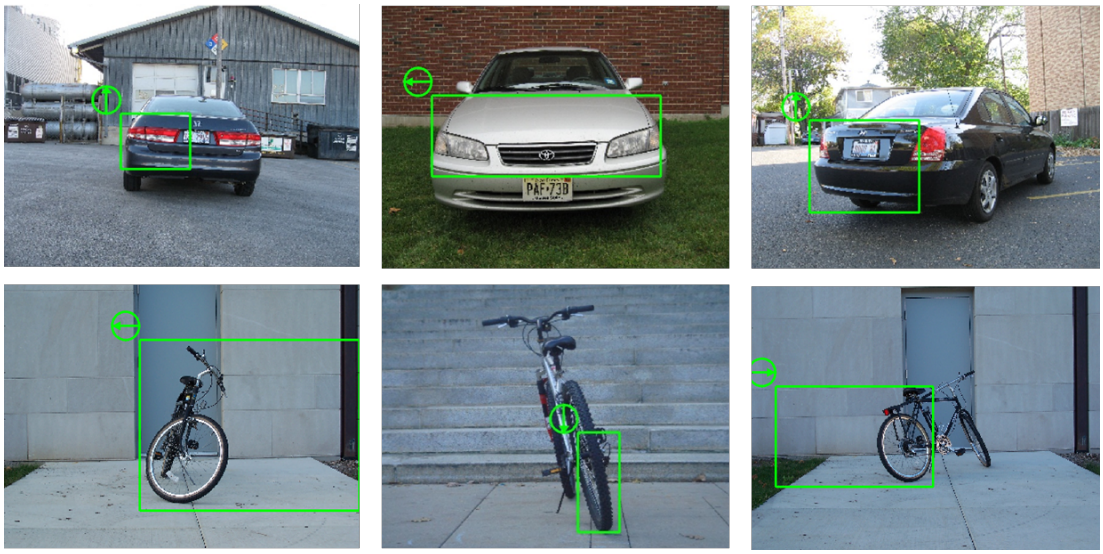

Figure 3.24: Some failed detection results of the *Multi-View Model* on the 3D Object Category car data set (top) and on the 3D Object Category bicycle data set (bottom). The failed detections are mainly caused by sub detections within an object instance or a nonrigid geometry.

Figure 3.25: Some failed detection results of the *Multi-View Model* on the PASCAL VOC2006 data set for the object class car (top) and for the object class bicycle (bottom). The failed detections are mainly caused by sub detections within other object instances or a nonrigid geometry.

Chapter 4

The Viewsphere Model

In this chapter, we describe our second part-based approach to object class detection which relies on a database of 3D object models and a set of real negative training images. In this thesis, we term this approach the *Viewsphere Model*. We relate the *Viewsphere Model* to the *Multi-View Model*, summarize previous work, and focus on explaining the unsupervised training procedure of the *Viewsphere Model*. This chapter concludes with an experimental evaluation which also includes a comparison of the *Viewsphere Model* with the *Multi-View Model*.

4.1 Introduction

In the previous chapter, we have introduced the *Multi-View Model*, our first approach to multi-view object class detection which relies on a database of CAD models and real negative training images. The *Multi-View Model* combines the advantages of the generative *Star Model* and the discriminative *Spatial Pyramid Model* by combining these two different part-based models into one common object class detection framework. However, the *Multi-View Model* suffers from the following limitations which will be addressed by our second part-based approach, the *Viewsphere Model*:

- For each defined viewpoint of the *Multi-View Model* a set of viewpoint-specific part detectors is established independently of all other viewpoints. Consequently, the *Multi-View Model* generates similar or redundant object part detectors due to

viewpoint symmetries[1] or part similarities[2]. The *Viewsphere Model* addresses this issue by discovering common object parts in an unsupervised way. These common object parts are intrinsic within an object class over the entire viewsphere and, consequently such an object part can contribute to an object class representation for several defined viewpoints on the viewsphere.

- The described method of the *Multi-View Model* for generating a set of viewpoint-specific part detectors relies on a simple heuristic: based on the synthetically generated positive training images of an object class a *Laplace image* for a specific viewpoint is established. A spatial part layout with different part levels, i.e., different part sizes, is derived by decomposing this *Laplace image* into several part locations which capture the highest gradient over all training examples. Subsequently, these part locations are used to train viewpoint-specific part detectors. The first shortcoming of this method is that there is no verification step in order to ensure that the resulting part detectors are suitable to represent the object class being trained. The second shortcoming is the assumption that the aspect ratios of the CAD models for a specific object class are similar for a given viewpoint and that the structured regions within an object class occur in similar areas within the synthetically generated positive training images. In this chapter, we circumvent these two shortcomings by relying on a two-stage approach which determines suitable object parts for a given object class: first, we use affinity propagation (48) in conjunction with the HOG descriptor (21) to decompose a given object class by an unsupervised approach into a pool of potential object parts. In the second step, a subset of these generated object parts is selected by an entropy-based measure (45, 123) in order to obtain those object parts which are most informative with respect to the object class being trained.

- The pose estimation of the *Multi-View Model* simply relies on the pose label which is provided by the viewpoint-specific *Star Models* of the pre-detection step. The corresponding object parts of those *Star Models* are established in an unsupervised way which does not take into account the inter-viewpoint discriminativity.

[1] For example, the front view and the back view of the object class car are symmetric viewpoints which might be visually similar.

[2] For example, the wheels of a car are similar object parts.

As a result the performance of the *Multi-View Model* with respect to pose estimation cannot compete with other reported results due to viewpoint confusion which is mainly pronounced between opposing and neighboring viewpoints. The *Viewsphere Model* addresses this issue by dividing the defined viewsphere into a set of suitable subspaces. Subsequently, we select for each defined subspace a subset of most informative object parts and establish for each defined subspace a discriminative *Spatial Pyramid Model* based on these selected sets of object parts. In addition, within the subspace with the highest classification score, the pose estimation can be further refined by modeling the locations of the corresponding object parts with Gaussian mixture models.

In summary, we introduce the *Viewsphere Model* which is an approach to object class detection and approximate pose estimation based on a database of 3D object models and real negative training images. The *Viewsphere Model* avoids the described limitations of the *Multi-View Model* but retains the advantage of the *Multi-View Model*, i.e., the integration of the generative *Star Model* and the discriminative *Spatial Pyramid Model* into one common object class detection framework. This chapter is structured as follows: Section 4.2 summarizes previous work on object part sharing and the unsupervised selection of object parts. Details on the training and the detection procedure of the *Viewsphere Model* are presented in Section 4.3 and in Section 4.4. Experimental results and a comparison of the *Viewsphere Model* with the *Multi-View Model* are given in Section 4.5. This chapter is concluded in Section 4.6.

4.2 Related Work

A survey of related work on multi-view object class detection has already been given in Section 3.2. Consequently, in this section we focus on the selection of informative object parts[1] and the sharing of object parts for the task of multi-view object class detection and pose estimation. Both the sharing of object parts and the selection of informative object parts reduce the computational complexity of an object class detection system. In addition, the selection of informative object parts overcomes overfitting (58).
Recent work on feature selection with respect to object class detection and pose estimation can mainly be divided into two groups, either by using AdaBoost (47) or

[1]Note that object parts are also referred to as features in related work.

by relying on the mutual information (18) as the selection criterion. AdaBoost was originally introduced as a machine learning algorithm by Freund and Schapire (47). It iteratively selects a 'weak' classifier from a pool of 'weak' classifiers to establish a 'strong' classifier as linear combination of those selected 'weak' classifiers. The idea of using AdaBoost for object class detection, especially for face detection, is proposed by (124). It relies on a boosted cascade of simple haar-like features and was extended for example by (80). In (117) Gentle AdaBoost (49) is adapted in order to select simple image patches that can be shared across different object classes and different object poses as well.

The second group of approaches to object part selection relies on the mutual information as a criterion by measuring the mutual dependence between random variables. The advantages of an entropy-based part selection are discussed in (121). In (118) class-specific image patches are selected for classification by maximizing the mutual information between the image patches and the object class they represent. Dorko and Schmid (22) also maximize the mutual information in order to select object parts which are based on manually labeled local image features for the task of object class detection. In (123) the maximization of the mutual information is extended by a greedy search, taking into account the previously selected features and thus avoiding redundancy among the selected features. This feature selection procedure is also termed conditional mutual information and in (45) it is shown that this selection procedure achieves equivalent results compared to AdaBoost. The conditional mutual information criterion is used in (25) to build a hierarchy of simple image patches for classification and in (92) to select binary features for a spatial pyramid in order to perform a pose estimation.

In contrast to the selection of informative object parts, procedures for object part sharing intend to establish common object parts that can be shared within an object class and/or among different object classes and thus reduce the computational complexity of an object class detection system. In (84) one common hierarchical codebook of edge based features is established by using agglomerative clustering in order to represent several object classes and (111) propose hierarchical models based on shared object parts and interest point detectors for learning multiple object classes. Fidler and Leonardis (43) also represent several object classes by a hierarchy of parts based on a set of oriented Gabor filters, where the parts of a specific layer are the statistically

most significant compositions of parts from the previous layer. A further multi-level hierarchical model for part sharing is proposed in (131), relying on both appearance and shape based features in order to simultaneously perform multi-view and multi-object detection.

Regarding the proposed *Viewsphere Model*, we reconsider the work of (123) in order to select informative object parts for the task of multi-view object class detection and pose estimation. However, we rely on a discriminative learning of object parts and unlike the object class detection approaches of (91, 94), these object parts are initialized by an unsupervised clustering step. In further contrast to previous work (43, 84, 111, 131), the *Viewsphere Model* is based on a flat hierarchy which can be efficiently evaluated. Furthermore, we exploit the advantages of CAD models for training in order to perform a fully unsupervised selection of photo-realistic object parts over the entire viewsphere. By avoiding manually chosen semantic object part correspondences (132) or viewpoint-specific object parts (109), common geometry and appearance, which are intrinsic within an object class over the entire viewsphere, are discovered. Consequently, an established object part can contribute to the representation of an object class for several defined viewpoints on the viewsphere, resulting in an object class representation which performs on par with the current state-of-the-art detection approach of (34).

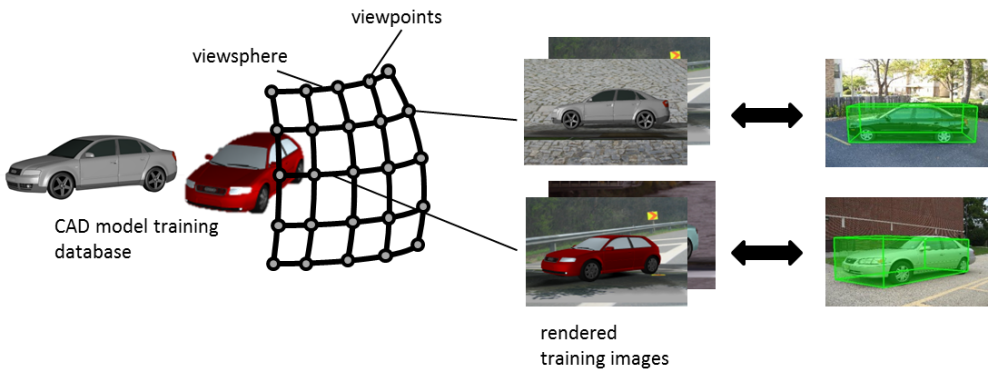

Figure 4.1: With the *Viewsphere Model* an object class representation covering the defined viewsphere is built from a database of 3D object models and a set of real negative images. To this purpose, common object parts are discovered from the rendered training images such that intra-class and viewpoint variation is covered. The *Viewsphere Model* allows for a 2D localization and an approximate pose estimation on unseen test images.

4.3 Training

This section outlines the necessary training steps for the *Viewsphere Model* and starts with the use of an object class specific database of CAD models and a set of real negative images as training source. Afterwards, the unsupervised approach for decomposing an object class into a pool of potential object parts is explained. This generated pool of potential object parts serves as input for both the 2D localization step and the pose estimation step of the *Viewsphere Model*, which are also described in this section.

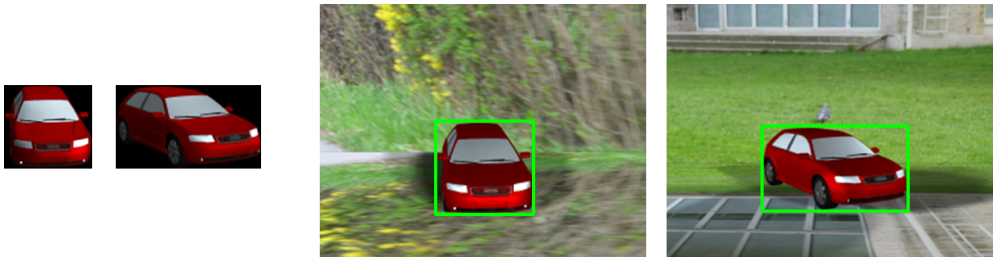

Figure 4.2: Example images for the *pure examples* (left) and the *validation images* (right). The *pure examples* exclusively show the object instances of a specific object class in front of a black background. In contrast, the *validation images* show the object instances (indicated by the bounding boxes which are automatically determined during the rendering process) of a specific object class in front of randomly selected images from the background images described in Section 2.3.2. Note that the viewpoint label for each synthetically generated training image is provided by the rendering process without any manual intervention.

4.3.1 Training Examples

Similar to the *Multi-View Model*, the *Viewsphere Model* derives its positive training examples for all subsequent training steps entirely from an object class specific database of CAD models (see Figure 4.1). See Figure 2.7 for some model examples of different object classes or Appendix B for a visualization of all CAD models which are used in this thesis. When applied to a test set, the *Viewsphere Model* does not need to be retrained or adapted to the data set characteristics. This is a key advantage of reducing training set dependencies in favor of a better generalization. The CAD models of the database for a specific object class are rendered from a dense grid of viewpoints $\omega = (\alpha, \epsilon)$ which is defined on the viewsphere in steps of 5° in azimuth direction α and in steps of 5° in elevation direction ϵ. Based on this defined grid of viewpoints and the rendering process described in Section 2.3.2, we generate for the *Viewsphere Model*

two independent sets of training images: the *pure examples* and the *validation images*. For the *pure examples*, which are used to decompose an object class into a pool of potential object parts, the CAD models are rendered in front of a black background (see Figure 4.2 (left)). As a result, the influence of any background on the part generation process for a specific object class can be avoided. In order to define a fixed training scale for the *Viewsphere Model* we rescale (by retaining the aspect ratio of each training image) the *pure examples* to a fixed height[1]. The *validation images* are generated by rendering the CAD models in front of randomly selected images from the background images described in Section 2.3.2. The object instances within the *validation images* vary in width and height (see Figure 4.2 (right)). Further details on the *pure examples* and the *validation images* are given in the experimental evaluation of Section 4.5. In addition to the synthetically generated positive training images, i.e., the *pure examples* and the *validation images*, the *Viewsphere Model* relies on a set of real negative training images which does not contain any object instance of the object class being trained. To this purpose, we modify the PASCAL VOC2006 training data set to establish this set of negative training images[2].

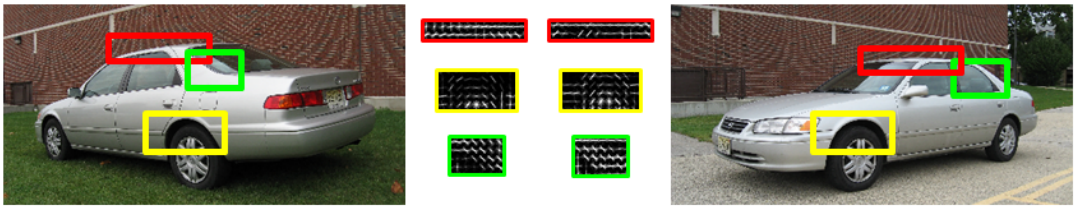

Figure 4.3: Object parts from different viewpoints might display similar appearance characteristics in HOG space. The *Viewsphere Model* exploits these similarities in an adaptive way for 2D localization and pose estimation.

4.3.2 Generating a Pool of Object Parts

Initially, for the object class being trained, a pool P of potential object parts is generated as input for the subsequent higher-level training steps of the *Viewsphere Model*,

[1] For our experiments we use a fixed height of 64 pixels for all object classes. Note that the *pure examples* vary in width, since we retain the aspect ratio of each training image.

[2] The PASCAL VOC2006 training data set is modified such that it does not contain any object instance of the object class under training.

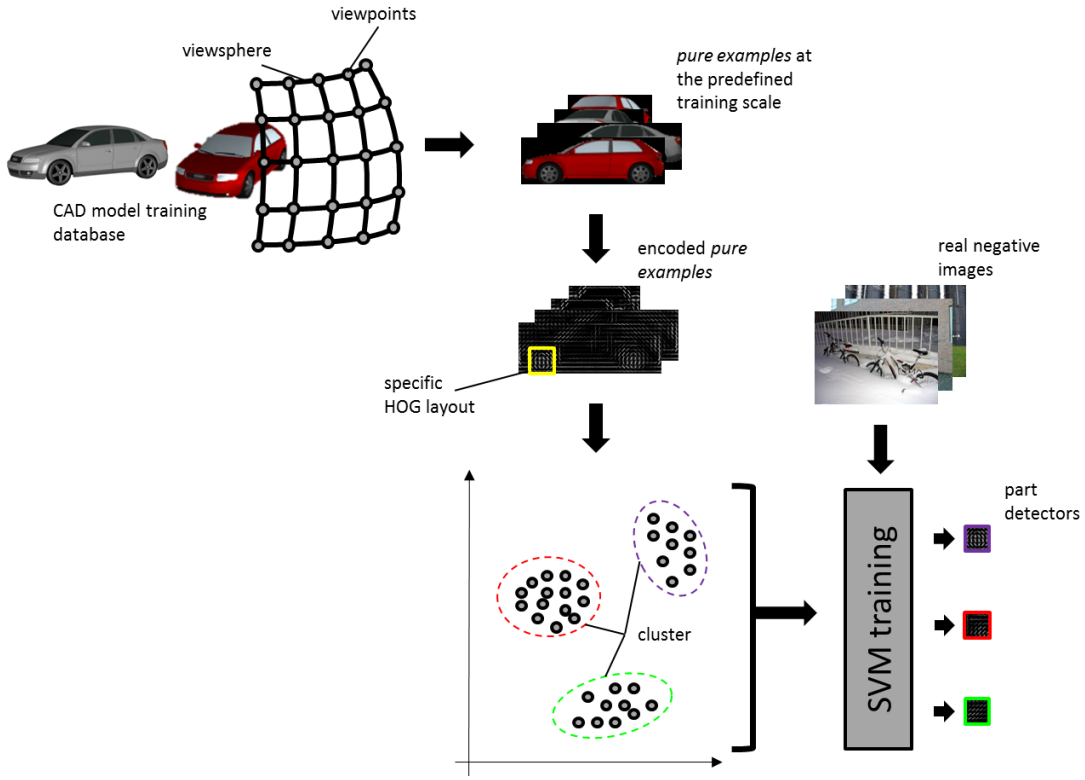

Figure 4.4: The *pure examples* for all defined viewpoints on the viewsphere at the predefined training scale are encoded with the HOG descriptor of (21). We apply affinity propagation (48), which is an unsupervised clustering algorithm, to all features which are collected from the *pure examples* for a specific HOG layout. The features which are assigned to a cluster serve as positive training examples and subsequently, we train for each generated cluster a linear SVM classifier, i.e., an object part detector, by using the 'bootstrapping' procedure of Section 2.2.1.1.1 in conjunction with real negative training images. We repeat this procedure for different HOG layouts in order to obtain a pool of potential object parts.

i.e., for training the 2D localization step which is described in Section 4.3.3 and for training the pose estimation step which is described in Section 4.3.4. The objective of this training step is to discover common object parts which are intrinsic within an object class over parts of the viewsphere due to viewpoint symmetries and/or part similarities (see Figure 4.3). The unsupervised approach to establish a pool P of potential object parts and to discover common object parts is illustrated in Figure 4.4.

For generating a pool of potential object parts for an object class, we rely on the generated *pure examples* at the predefined training scale for all defined viewpoints and real

negative images which do not contain any object instance of the object class under training. The T *pure examples* are encoded with the HOG descriptor of (21), since we intend to discover common object parts within an object class in HOG space (see Figure 4.3). Let E be an encoded *pure example* with the fixed training height of h_{train} 'HOG-cells' and a width of w_t 'HOG-cells'. The maximum width within all T encoded *pure examples* is denoted by w_{max}. Let $l = (h, w)$ be a HOG layout with a height of h 'HOG-cells' and a width of w 'HOG-cells'. $L = \{(h, w) \mid 4 \leq h \leq h_{train},\ 4 \leq w \leq w_{max}, h \in \mathbb{N}, w \in \mathbb{N}\}$ denotes a set of different HOG layouts. A HOG feature f of a corresponding HOG layout $l = (h, w)$ is obtained by concatenating the 'HOG-cells' within a $h \times w$ sub window of an encoded *pure example* E.

As shown in Figure 4.4, for a specific HOG layout in L we collect HOG features from the encoded *pure examples* of all defined viewpoints using a 'sliding window' approach. We apply affinity propagation (48), which is an unsupervised clustering process, to these collected HOG features and consequently, similar HOG features (i.e. patches with a similar appearance) within an object class are assigned to the same cluster. For each established cluster an object part detector is built in order to model the appearance of the corresponding patches within a cluster. We use the HOG features assigned to a cluster as positive training examples and rely on the 'bootstrapping' procedure of Section 2.2.1.1.1 in conjunction with real negative training examples from the modified PASCAL VOC2006 training data set to train an object part detector, i.e., a linear SVM classifier (see Figure 4.4). For learning those object part detectors the size of the established clusters is essential: the size of the established clusters must not be too large (e.g. all collected HOG features are assigned to one cluster) and must not be too small (e.g. each collected HOG feature represents a cluster on its own) to learn suitable object part detectors. To address this issue, we measure the average distance s of all established clusters in Euclidean space by using the following equation

$$s = \frac{1}{Q} \sum_{q=1}^{Q} \sum_{f_j \in G_q} (||f_j - m_q||_2)^2. \tag{4.1}$$

Q is the total number of all established clusters, f_j are the HOG features which are assigned to the cluster G_q, and m_q is the representative training example, i.e., a specific HOG feature, for the cluster G_q chosen by affinity propagation (48). By setting the 'preference' values (48) of affinity propagation, we are able to pre-specify the number of established clusters: high 'preference' values cause a large number of clusters (i.e.

a small average distance s of the established clusters) and small 'preference' values induce a small number of clusters (i.e. a high average distance s of the established clusters). Consequently, we apply affinity propagation with high 'preference' values to the collected HOG features for a specific HOG layout, calculate the average distance s of all established clusters with Equation 4.1, and reapply affinity propagation with smaller 'preference' values to the collected HOG features until a predefined average distance s_{pre} of the established clusters is reached[1]. The described procedure above, i.e., collecting HOG features for a specific HOG layout from the encoded *pure examples* of all defined viewpoints, clustering the resulting HOG features until a predefined average distance s_{pre} is reached, and training a linear SVM classifier for each established cluster, is repeated for all defined HOG layouts in L. Finally, we obtain a pool P of potential object parts, in which each object part is represented by a linear SVM classifier.

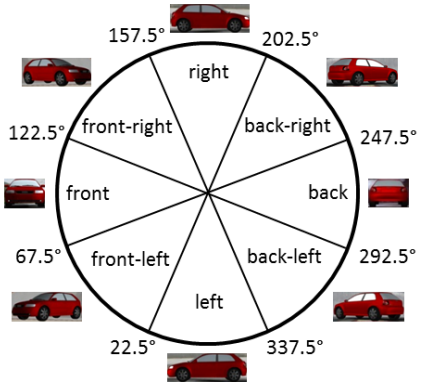

Figure 4.5: The defined viewsphere of the *Viewsphere Model* is divided into eight equally spaced viewsphere subspaces in azimuth direction and for each subspace a potential pool of object parts is generated. Finally, the generated object parts for all defined subspaces form the final pool P of potential object parts.

During initial experiments it turned out that it is advantageous for the detection performance of the *Viewsphere Model* to divide the defined viewsphere of the *Viewsphere Model* into smaller subspaces and to generate for each of these defined subspaces a pool of potential object parts by applying the described procedure above to the corresponding *pure examples* for each defined viewsphere subspace. To this purpose, we divide the defined viewsphere of the *Viewsphere Model* in azimuth direction into eight equally

[1]Based on initial experiments we have identified $s_{pre} = 500$ as an adequate average distance of the established clusters resulting in suitable object part detectors.

spaced viewsphere subspaces (see Figure 4.5) and establish for each of those viewsphere subspaces a pool of potential object parts. Finally, the generated object parts for all defined viewsphere subspaces form the final pool P of potential object parts for the subsequent training steps; the influence of this viewsphere subdivision on the detection performance of the *Viewsphere Model* is reported during the experimental results in Section 4.5.2.1.

4.3.3 2D Localization

In this section, we describe the training procedure for the 2D localization step of the *Viewsphere Model*: a subset of N_{2D} object parts, which are most informative with respect to the object class being trained, is selected from the generated pool P of potential object parts. Here, we rely on an entropy-based measure (45, 123) which is described in this section. Similar to the proposed *Multi-View Model*, the *Viewsphere Model* combines the generative *Star Model* and the discriminative *Spatial Pyramid Model* into one common object class detection framework. Consequently, the selected subset of N_{2D} informative object parts serves as input for both the generative part of the *Viewsphere Model*, which consists of a dense grid of *Star Models*, and the discriminative part of the *Viewsphere Model*, which consists of one *Spatial Pyramid Model*. The training steps of these two parts for the 2D localization step of the *Viewsphere Model* are also described in this section.

4.3.3.1 Selecting the Most Informative Object Parts

The generated pool P of potential object parts (see Section 4.3.2) contains a large number of redundant and/or non-informative object parts due to viewpoint symmetries and part similarities. In this training step a subset of N_{2D} ($1 \leq n \leq N_{2D}, n \in \mathbb{N}$) object parts is selected from the established object part pool P, by ranking the informativeness of each potential object part with respect to a defined positive and a negative image set based on an entropy-based measure. For the 2D localization of an entire object class, we define that an object part is informative if it appears on as many object instances under as many viewpoints as possible. Consequently, the positive image set is chosen to contain the generated *pure examples* at the predefined training scale of all defined viewpoints on the viewsphere. The negative image set is chosen to contain real negative training examples from the modified PASCAL VOC2006 training data set (see Section 4.3.1).

Based on the defined positive and negative image set, we select altogether a subset of N_{2D} object parts from the object part pool P for the 2D localization step of the *Viewsphere Model* by relying on the entropy-based selection procedure which is described in the following two subsections. We rely on this entropy-based selection procedure also for training the pose estimation step of the *Viewsphere Model* which is described in Section 4.3.4. Consequently, in the following we describe this selection procedure consisting of two steps in general form, i.e., independent of the defined positive and negative image set and the number of selected object parts.
The first step of this entropy-based measure determines independently for each object part from the established pool P of potential object parts (i.e. for the corresponding linear SVM classifier) an optimal detection threshold. This detection threshold maximizes the information content of an object part regarding the defined positive and negative image set. The second step of this selection procedure is a greedy search algorithm based on the conditional mutual information criterion (45) to avoid redundancy among the selected object parts.

4.3.3.1.1 Detection Threshold

The first step of the entropy-based selection process is to determine for each object part from the pool P an optimal detection threshold θ. This detection threshold maximizes the mutual information (18) between an object part and the defined positive and negative image set. To this purpose, we treat an object part F_i in association with a detection threshold θ_i as a binary random variable

$$F_i(\theta_i) = \begin{cases} 1, & \text{if } sc_{max}(img, svm_i) \geq \theta_i \\ 0, & \text{otherwise} \end{cases} . \tag{4.2}$$

Here, sc_{max} is the maximum score of the corresponding object part detector svm_i (i.e. the linear SVM classifier) on an image img. In addition, a binary random variable C is defined, where $C = 1$ if the image img belongs to the defined positive image set and $C = 0$ if the image img belongs to the defined negative image set. Between these

two binary random variables the mutual information $MI(F_i(\theta_i); C)$[1] is defined as

$$MI(F_i(\theta_i); C) = H(C) + H(F_i(\theta_i)) - H(C, F_i(\theta_i)) \tag{4.3}$$

As shown in Equation 4.3, the mutual information between an object part F_i and the binary random variable C depends on the detection threshold θ_i. Consequently, the optimal detection threshold θ_i^{opt} for an object part F_i can be determined from

$$\theta_i^{opt} = \underset{\theta_i}{\operatorname{argmax}}[MI(F_i(\theta_i); C)] \tag{4.4}$$

resulting in the maximal mutual information MI_i^{max} for an object part F_i

$$MI_i^{max} = \max_{\theta_i}[MI(F_i(\theta_i); C)] = MI(F_i(\theta_i^{opt}); C). \tag{4.5}$$

An example for the mutual information of an object part F_i as a function of the detection threshold θ_i is given in Figure 4.6. If the detection threshold is set too low, the mutual information score will also be low since the object part is detected frequently in the defined negative image set. A high detection threshold will likewise result in a low mutual information since the object part now is too sparsely detected in the defined positive image set. At some intermediate value of the detection threshold the mutual information reaches a maximum and the object part delivers a maximum amount of information.

4.3.3.1.2 Greedy Search

After the optimal detection threshold θ_i for each object part F_i in the pool P is independently determined, we can iteratively select an optimal subset of N ($1 \leq n \leq N, n \in \mathbb{N}$) object parts from the pool P of potential object parts. We rely on a greedy search algorithm, which is based on the conditional mutual information criterion (45), to avoid redundancy among the selected object parts.

We assume that the initial pool of potential object parts, which contains all generated object parts from P, is denoted by P_0. The initial pool of selected object parts, which is an empty set, is denoted by S_0. The greedy search algorithm is initialized (i.e.

[1] With $H(U) = -\sum_{u \in U} p(u) \log_2 p(u)$ and $H(U, V) = -\sum_{u \in U} \sum_{v \in V} p(u, v) \log_2 p(u, v)$ being Shannon's entropy and Shannon's joint entropy based on the binary random variables $U = \{0, 1\}$ and $V = \{0, 1\}$. The estimation of the probabilities (i.e. p(u) and p(u,v)) can be done accurately by counting the numbers of occurrences of the specific patterns within the defined positive and negative image set.

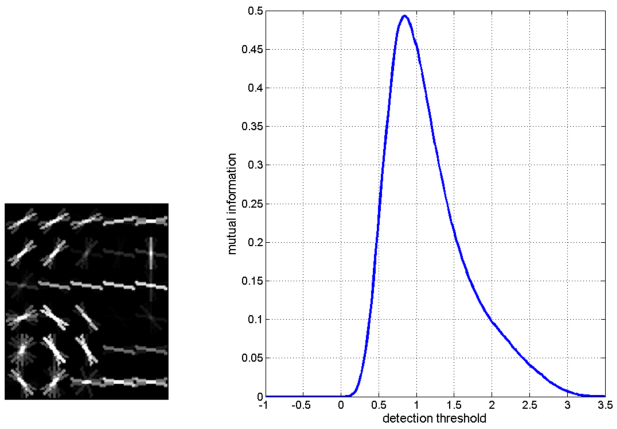

Figure 4.6: Mutual information as a function of varying the detection threshold of an example object part detector.

$n = 1$) by moving the object part F_1^{opt} with the highest mutual information, obtained by Equation 4.5, from the initial pool of potential object parts P_0 to the initial pool of selected object parts S_0. After the first iteration of the greedy search algorithm, the pool of potential object parts is denoted by P_1 and the pool of selected object parts is denoted by S_1. At the second iteration of the greedy search algorithm (i.e. $n = 2$) we want to select the second object part F_2^{opt} from the pool of potential object parts P_1. However, at the second iteration we do not rely on the maximal mutual information as the selection criterion. Instead, we rely on the conditional mutual information criterion (45) in order to select as second object part F_2^{opt} from the pool P_1 the object part which has the highest additional information regarding the previously selected object part F_1^{opt}

$$MI(C; F_i | F_1^{opt}) = H(C, F_1^{opt}) - H(C, F_i, F_1^{opt}) - H(F_1^{opt}) + H(F_i, F_1^{opt}). \tag{4.6}$$

As described in (45), the conditional mutual information of Equation 4.6[1] is an estimate of the quantity of information shared between an object part F_i from the potential pool of object parts P_1 and the binary random variable C, given the first selected object part F_1^{opt}. If F_i and F_1^{opt} contain the same information about C the terms on the

[1] With $H(U, V, W) = -\sum_{u \in U} \sum_{v \in V} \sum_{w \in W} p(u, v, w) \log_2 p(u, v, w)$ being Shannon's joint entropy based on the binary random variables $U = \{0, 1\}$, $V = \{0, 1\}$, and $W = \{0, 1\}$. Similar to Equation 4.3, the probabilities of Equation 4.6 can be accurately estimated by counting the numbers of occurrences of the specific patterns within the defined positive and negative image set.

right side of Equation 4.6 compensate each other and consequently, the conditional mutual information is zero. If F_i brings additional information about C, which is not already contained in F_1^{opt}, there is a difference on the right side of Equation 4.6 and the conditional mutual information is not zero. Consequently, at the second iteration of the greedy search algorithm, we select as second object part F_2^{opt} from the pool P_1 of potential object parts the object part with the highest conditional mutual information obtained by Equation 4.6. For the selection of the third object part F_3^{opt} (i.e. $n = 3$) we then have to consider the previously selected object parts F_1^{opt} and F_2^{opt}. Formally, the selection process of an object part F_n^{opt} at iteration n can be described as

$$F_n^{opt} = \underset{F_i \in P_{n-1}}{\operatorname{argmax}} \left[\min_{F_j \in S_{n-1}} \left[MI(C; F_i | F_j) \right] \right] \quad 2 \leq n \leq N. \tag{4.7}$$

P_{n-1} is the pool of potential object parts and S_{n-1} is the pool of selected object parts at iteration n. First, we have to calculate between a potential object part F_i and all previously selected object parts F_j the conditional mutual information by using Equation 4.6. Afterwards, we store for this potential object part F_i the minimum value obtained for the conditional mutual information (see Equation 4.7). This value will be small if the potential object part F_i is similar to a previously selected object part F_j and consequently, redundancy among the selected object parts will be avoided. This calculation is done for all potential object parts from the pool P_{n-1}. Subsequently, we select the object part F_n^{opt} from the pool P_{n-1} which yields the maximum increase of additional information about the defined positive and negative image set (see Equation 4.7). The update rules for the pool of potential object parts and the pool of selected object parts are defined by

$$P_n = P_{n-1} \setminus \{F_n^{opt}\} \quad S_n = S_{n-1} \cup \{F_n^{opt}\} \quad 1 \leq n \leq N. \tag{4.8}$$

After N iterations we have selected N object parts from the pool P of potential object parts that contain a maximum of information regarding the defined positive and negative image set.

4.3.3.2 Modeling a Dense Grid of Star Models

As shown in Figure 4.7, the generative part of the *Viewsphere Model* consists of a dense grid of *Star Models*. To this purpose, we establish for each defined viewpoint on the

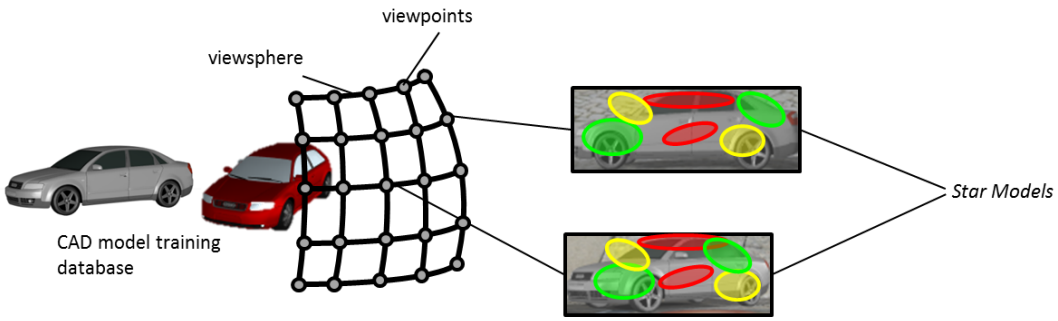

Figure 4.7: The pre-detection step of the *Viewsphere Model* consists of a *Star Model* for each defined viewpoint on the viewsphere. Each generative *Star Model* describes the spatial uncertainty, i.e., the locations of the most informative object parts, with Gaussian mixture models.

viewsphere a generative *Star Model* (see Section 2.2.1) based on the selected subset of N_{2D} object parts of Section 4.3.3.1. As shown in Figure 4.8, each linear SVM classifier associated with a selected object part is applied densely to the *pure examples* at the predefined training scale of a defined viewpoint leading to part detector responses. According to Section 2.2.1.1.2, we collect for each object part the location of the maximum detection response in each *pure example* and model the spatial distribution of these locations with respect to the image center by using a Gaussian mixture model. Finally, we obtain for each defined viewpoint a generative *Star Model* which is able to predict a suitable bounding box during the detection procedure, by projecting the average size of the corresponding *pure examples* (i.e. the fixed height and the average width at the predefined training scale) as mean bounding box into a test image (see Figure 4.8).

The dense grid of *Star Models* as the generative part of the *Viewsphere Model* is used to identify regions of interest, i.e., initial object hypotheses, which potentially contain an object instance of the trained object class. However, in order to establish such a dense grid of *Star Models*, which should be able to deal with significant intra-class variation in order to achieve a high recall on a benchmark data set, it is not necessary to use all N_{2D} selected object parts of Section 4.3.3.1. On the *validation images*, we optimize the trade-off between minimizing the number of object parts M_{2D} ($M_{2D} \leq N_{2D}$) used by the dense grid of *Star Models* in favor of a lean description and maximizing the recall of the dense grid of *Star Models* on the expected intra-class and intra-viewpoint variation

within the object class being trained. In practise, we determine the optimal number of M_{2D} object parts for the dense grid of *Star Models* in a validation step; see Section 4.5 for experimental results.

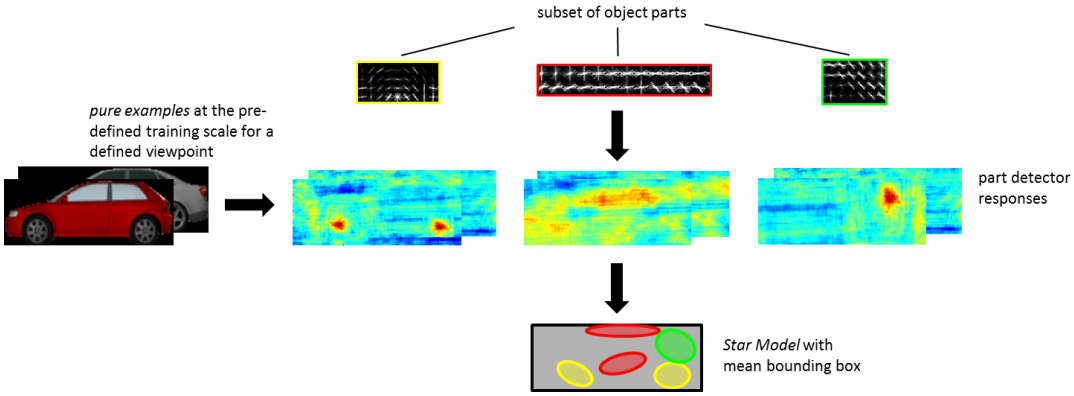

Figure 4.8: For each defined viewpoint on the viewsphere, we apply the SVM classifiers associated with the selected subset of object parts to the corresponding *pure examples* at the predefined training scale. We model the location of these object parts with respect to the image center by using Gaussian mixture models. The average size (i.e. the fixed height and the average width) of the *pure examples* is used as mean bounding box to predict a suitable bounding box during the detection procedure.

4.3.3.3 Learning the Spatial Pyramid Model

During testing, the established dense grid of *Star Models* (see Section 4.3.3.2) will allow generating a set of initial object hypotheses. However, depending on each *Star Model*, these generated object hypotheses have different score ranges and varying aspect ratios. In the following, we introduce, similar to the *Multi-View Model*, a verification step which is based on the *Spatial Pyramid Model* of Section 2.2.2 in order to rank the generated object hypotheses in a consistent way. We apply the established dense grid of *Star Models* to the *validation images* and resize (while retaining the aspect ratio) the generated object hypotheses to the predefined training scale, i.e., the fixed height of the *pure examples*. We convert the resized object hypotheses into spatial pyramid representations (70) by relying on the selected subset of N_{2D} object parts of Section 4.3.3.1. By using a spatial pyramid representation we can impose a regularly spaced grid subdivision which is relative to the area covered by an object hypothesis and thus independent of its aspect ratio and dimension. As a result, this spatial

pyramid representation encodes the part detector responses of all N_{2D} selected object parts within the area covered by an object hypothesis. Based on the ground truth, which is automatically provided for each *validation image*, we are able to divide these spatial pyramid representations into a set of positive and negative training examples. Subsequently, we train one common nonlinear SVM classifier with an intersection kernel (56) based on this training set of spatial pyramid representations. Consequently, the established *Spatial Pyramid Model* is an object class representation which models a given object class for all defined viewpoints on the viewsphere and enables a consistent ranking of the initial object hypotheses provided by the dense grid of *Star Models*.

4.3.4 Pose Estimation

In this section, we describe the training procedure for the pose estimation step of the *Viewsphere Model*. In contrast to the 2D localization step of the *Viewsphere Model*, where we rely on one common *Spatial Pyramid Model* for the entire viewsphere, we divide the viewsphere into equally spaced viewsphere subspaces. Subsequently, we select for each defined subspace the most informative object parts from the pool P of potential object parts. Based on these selected subsets of object parts a *Spatial Pyramid Model* is established for each defined subspace enabling an initial pose estimation for an object hypothesis. In addition, this initial pose estimation for an object hypothesis can be further refined by using Gaussian mixture models.

4.3.4.1 Selecting the Most Informative Object Parts

Unlike the 2D localization step of the *Viewsphere Model*, the pose estimation step of the *Viewsphere Model* requires a suitable discretization of the defined viewsphere into V $(1 \leq v \leq V, v \in \mathbb{N})$ equally spaced subspaces. This discretization can be freely adapted to the task setting and is not inherent to or imposed by our training procedure[1]. Since the pose estimation step of the *Viewsphere Model* is based on the object hypotheses, which are provided by the pre-detection step of the *Viewsphere Model*, it is not necessary to select object parts which are suitable to discriminate the entire object class from the background. In fact, for the pose estimation step of the *Viewsphere Model* it is necessary to select object parts which are suitable to discriminate a specific object

[1] In our experiments we rely on the same viewsphere discretization (see Figure 4.5) which is used for the part pool generation described in Section 4.3.2.

pose from other object poses. To this purpose, we define the *pure examples* which correspond to a specific viewsphere subspace v as the positive image set and we define the remaining *pure examples*, i.e., which correspond to the remaining viewsphere subspaces, as the negative image set. Based on those defined positive and negative image sets, we are able to select for each defined viewsphere subspace v a subset of N_{3D}^{v} informative object parts from the generated pool P of potential object parts by using the entropy-based selection process which is described in Section 4.3.3.1.

4.3.4.2 Pose Initialization

Based on the discretization of the defined viewsphere of the *Viewsphere Model* into V $(1 \leq v \leq V, v \in \mathbb{N})$ equally spaced subspaces and the V selected subsets of N_{3D}^{v} informative object parts, we establish for each defined viewsphere subspace v a *Spatial Pyramid Model*. As described in Section 4.3.3.3, the object hypotheses which are generated on the *validation images* (by applying the established dense grid of *Star Models*) are resized to the predefined training scale of the *Viewsphere Model*, i.e., the fixed height of the *pure examples*. For each defined subspace v we rely on the corresponding subset of N_{3D}^{v} selected object parts, convert all rescaled object hypotheses into spatial pyramid representations (70), and divide these spatial pyramid representations into a set of positive and negative training examples based on the ground truth which is provided for each *validation image*. Given these training sets, a nonlinear SVM with an intersection kernel (56) is trained for each defined subspace v by relying on the 'bootstrapping' procedure described in Section 2.2.1.1.1. During detection, the *Viewsphere Model* is able to provide an object hypothesis with an initial pose estimation by relying on the corresponding subspace of the *Spatial Pyramid Model* with the highest classification score.

4.3.4.3 Pose Refinement

As described in Section 4.3.4.2, the pose initialization of the *Viewsphere Model* is based on several *Spatial Pyramid Models* where each *Spatial Pyramid Model* is directly linked to a discretized subspace of the viewsphere. Figure 4.9 shows how the pose estimation precision of an object hypothesis can be further refined for the subspace of the corresponding *Spatial Pyramid Model* with the highest classification score. Each defined subspace v $(1 \leq v \leq V, v \in \mathbb{N})$ has its subset of N_{3D}^{v} $(1 \leq n \leq N_{3D}^{v}, n \in \mathbb{N})$ informative

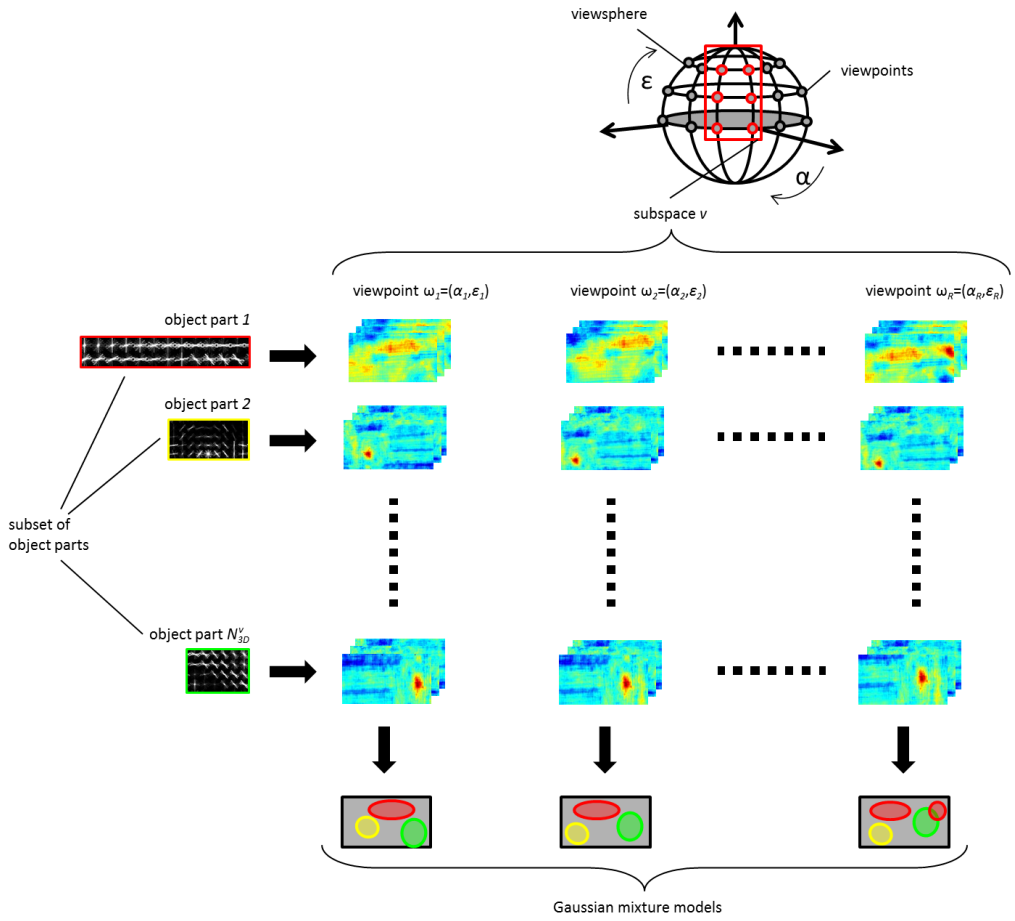

Figure 4.9: The pose refinement step of the *Viewsphere Model* is based on several Gaussian mixture models. Each Gaussian mixture model captures the spatial arrangement of an object part for a defined viewpoint on the viewsphere.

object parts and consists of R ($1 \leq r \leq R, r \in \mathbb{N}$) defined viewpoints $\omega_r = (\alpha_r, \epsilon_r)$. During the training of the pose estimation step, the corresponding linear SVM classifiers of the N_{3D}^v selected object parts are applied densely to the *pure examples* at the predefined training scale of the *Viewsphere Model*, i.e., the fixed height of the *pure examples*, within the corresponding subspace v. Subsequently, we measure for each object part n in each *pure example* for a given viewpoint $\omega_r = (\alpha_r, \epsilon_r)$ the location x_{n,ω_r} of the maximum detection response with respect to the image center[1]. We model for each object

[1]The position of the maximum detection response with respect to the center of a training image is measured in column direction u and row direction v, i.e., $x'_{n,\omega_r} = (\Delta u, \Delta v)$.

part n and each defined viewpoint ω_r the spatial distribution $pdf_{X_{n,\omega_r}}$ of all locations X_{n,ω_r} by using a Gaussian mixture model $\theta_{n,\omega_r} = \{\alpha_k, \mu_k, \Sigma_k\}$ with K components $(1 \leq k \leq K, k \in \mathbb{N})$

$$pdf_{X_{n,\omega_r}}(x_{n,\omega_r}|\theta_{n,\omega_r}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x_{n,\omega_r}|\mu_k, \Sigma_k). \tag{4.9}$$

α_k is the prior probability of a component k, μ_k is the mean vector of a component k, and Σ_k is the covariance matrix of a component k. The parameters of a Gaussian mixture model θ_{n,ω_r} are automatically estimated by the approach of (13). Assuming conditional independence of the N_{3D}^v object parts, the initially estimated subspace v can be refined to the viewpoint ω_r^{est}

$$\omega_r^{est} = \underset{\omega_r}{\operatorname{argmax}} \prod_{n=1}^{N_{3D}^v} pdf_{X_{n,\omega_r}}(x_{n,\omega_r}|\theta_{n,\omega_r}). \tag{4.10}$$

Note that the refined viewpoint of Equation 4.10 is relative to the virtual camera parameters used to generate the synthetic training images. With an estimated viewpoint ω_r^{est} and a given bounding box of the 2D localization step, we are able to project for each object hypothesis a mean 3D bounding box, which is computed from the 3D object model database, into a tested image.

4.4 Detection

This section describes the two necessary detection steps of the *Viewsphere Model* with respect to 2D localization and approximate pose estimation.

4.4.1 2D Localization

As mentioned in the introduction of this chapter, the *Viewsphere Model* retains the advantage of the *Multi-View Model* and integrates the generative *Star Model* and the discriminative *Spatial Pyramid Model* into one common object class detection framework. In the following, we describe the two detection steps of the *Viewsphere Model* with respect to 2D localization. First, the generative pre-detection step based on the dense grid of *Star Models* to obtain initial object hypotheses and second, the verification step based on the discriminative *Spatial Pyramid Model* to verify the initial object hypotheses and to establish a comparable ranking.

4.4.1.1 Pre-Detection

In order to identify regions of interest, i.e., initial object hypotheses, which potentially contain an object instance of the object class being detected, we rely on the dense grid of generative *Star Models*[1] described in Section 4.3.3.2 in conjunction with the detection procedure of the *Star Model* described in Section 2.2.1.2. The detection scores of these *Star Models* alone do not allow a consistent ranking of the initial object hypotheses, since the corresponding object hypotheses have varying aspect ratios and different score ranges. Consequently, in the following verification step of the *Viewsphere Model* we build on the classification performance of the *Spatial Pyramid Model* in order to obtain a normalized and comparable detection score for each initial object hypothesis.

4.4.1.2 Verification

The final detection result consists of a consistent and comparable scoring of the initial object hypotheses based on the *Spatial Pyramid Model* of Section 4.3.3.3. To this purpose, each initial object hypothesis, which is generated by the dense grid of *Star Models*, is resized (while retaining the aspect ratio) to the predefined training size, i.e., the fixed height of the *pure examples*. The responses of all N_{2D} selected object parts in the scaled area of an object hypothesis are encoded in a spatial pyramid representation as described in Section 4.3.3.3, which is then classified by the *Spatial Pyramid Model* to obtain the final detection score for the corresponding object hypothesis. Since the detection process of the *Viewsphere Model* can result in multiple overlapping object hypotheses, finally, a non-maximum suppression is performed which retains only high scoring bounding boxes and discards those bounding boxes covered by a higher-scoring bounding box.
Similar to the verification step of the *Multi-View Model*, the verification step of the *Viewsphere Model* results in a significant gain in average-precision of up to 40% in our experiments when compared to the detection scores provided by the dense grid of *Star Models*.

[1] In our experiments each *Star Model* generates 5 object hypotheses on a test image.

4.4.2 Pose Estimation

In addition to 2D localization, the *Viewsphere Model* is able to perform an approximate pose estimation for each final object hypothesis. As described in Section 4.3.4, this pose estimation step is based on several *Spatial Pyramid Models* where each *Spatial Pyramid Model* is directly linked to a discretized subspace of the defined viewsphere. To this purpose, each object hypothesis remaining after the non-maxima suppression of the verification step described in Section 4.4.1.2 is classified by the V *Spatial Pyramid Models* for all defined subspaces. Within the discrete subspace with the highest classification score the pose estimation precision of an object hypothesis is further refined by applying the pose refinement step of the *Viewsphere Model* which is described in Section 4.3.4.3.

4.5 Evaluation

In this section, we outline the experimental results we achieve with the proposed *Viewsphere Model*. We evaluate the *Viewsphere Model* with respect to different tasks on the 3D Object Category data set of Section 2.5.1 and on the PASCAL VOC2006 data set of Section 2.5.2 for the classes car and bicycle and compare its performance with the *Multi-View Model*.

4.5.1 Training Setup

Like the *Multi-View Model*, the *Viewsphere Model* relies on synthetically generated positive training images rendered from CAD models which are available from commercial distributors, notably turbosquid.com and doschdesign.com. See Appendix B for a visualization of the CAD models which are used in the present thesis. In contrast to the *Multi-View Model*, we use all 25 car models and all 8 bicycle models for training, as the object part generation process of the *Viewsphere Model* (see Section 4.3.2) is not based on the assumption that the aspect ratios of the CAD models for a specific object class should be similar for a given viewpoint. As illustrated in Figure 4.10, for the bicycle class we modify[1] 6 bicycle CAD models of the 8 basic CAD models in a similar way to cover typical variations within the bicycle object class resulting in altogether

[1]We rely on NuGraf® distributed by Okino Computer Graphics to modify the CAD models (http://www.okino.com/nrs/nrs.htm/).

14 CAD models, i.e., 8 basic CAD models and 6 modified CAD models. In order to define a dense grid of viewpoints on the viewsphere, azimuth α is uniformly sampled from 0° to 360° in 5° steps and elevation ϵ is uniformly sampled from 0° to 20° in 5° steps. This viewpoint setup is used to generate the *pure examples* and the *validation images*; details on the generated positive training images for the *Viewsphere Model* are presented in Table 4.1. We draw all negative training images from the PASCAL VOC2006 training data set, which excluded the training images for the object classes car and bicycle. For our experiments we rely on the HOG implementation of (37) with a HOG cell size of 8 pixels and for testing we choose an image pyramid with 10 levels in an octave; for the *Spatial Pyramid Models* we choose a spatial pyramid representation with three levels of linear subdivision since a linear subdivision results in a more compact object class representation compared to a quadratic subdivision[1].

Figure 4.10: In order to take into account the typical variations within the object class bicycle (left) 6 of the 8 basic bicycle CAD models (center) are modified to generate 6 additional bicycle CAD models (right).

In order to evaluate the 2D localization performance of the *Viewsphere Model* and to be comparable with the results of the *Multi-View Model*, we again use the overlap criterion which is described in Section 2.4.2: a predicted bounding box is considered as correct if the overlap between a predicted bounding box and a ground truth bounding box exceeds 50%. If several bounding boxes are predicted in the same image area, only one detection is considered as correct, while the remaining detections are considered as false positives.

In Section 4.3.3.2 we outline that the number of selected object parts for the dense grid of *Star Models* is chosen in a validation step to optimize the tradeoff between recall and model complexity. To this purpose, we increase the number of object parts M_{2D} for

[1]For a spatial pyramid representation with three levels a linear subdivision reduces the descriptor length to 66% compared to a quadratic subdivision, independently of the number of object parts.

building a dense grid of *Star Models* until the recall[1] on the *validation images* exceeds a predefined threshold or the limit of the N_{2D} selected informative object parts has been reached (i.e. $M_{2D} \leq N_{2D}$). Based on the results for the pre-detection step of the *Multi-View Model* (see Section 3.6.2.1), we decide to use a threshold of 98% for the predefined recall. Figure 4.11 shows the impact of changing the object part number M_{2D} with respect to the achieved recall on the *validation images* for the object class bicycle. In this example, saturation is reached when selecting $M_{2D} = 50$ object parts for building the dense grid of *Star Models*.

	car (25 CAD models)		**bicycle (8+6 CAD models)**	
	pure examples	*validation images*	*pure examples*	*validation images*
average height of the object instances	64 pixels	64 pixels	64 pixels	109 pixels
no. of defined viewpoints	360	360	360	360
no. of light variations	1	1	1	1
no. of all training images	9000	360	5040	360

Table 4.1: Details on the generated positive training images for the *Viewsphere Model*. The *pure examples* at the predefined training scale have the same fixed height and vary only in width. The object instances within the generated *validation images* vary in width and height. Note that for each defined viewpoint one CAD model is randomly selected to generate a viewpoint-specific training image for the *validation images*.

4.5.2 3D Object Category Data Set

On the 3D Object Category data set we assess the influence of the viewsphere subdivision and the number of selected object parts on the detection performance of the *Viewsphere Model*. In addition, the performance of the *Viewsphere Model* with respect to 2D localization and pose estimation is evaluated and compared to other reported results. Finally, we assess the influence of the CAD models' quality and the influence of using real images for training on the detection performance of the *Viewsphere Model*.

4.5.2.1 Viewsphere Subdivision and Number of Object Parts

In the first experiment, we evaluate the impact of two different parameters on the training procedure of the proposed *Viewsphere Model*: the influence of the viewsphere

[1] We determine how often on the *validation images* the overlap between one generated object hypothesis, i.e., a predicted bounding box, and the ground truth bounding box within a *validation image* exceeds 50%.

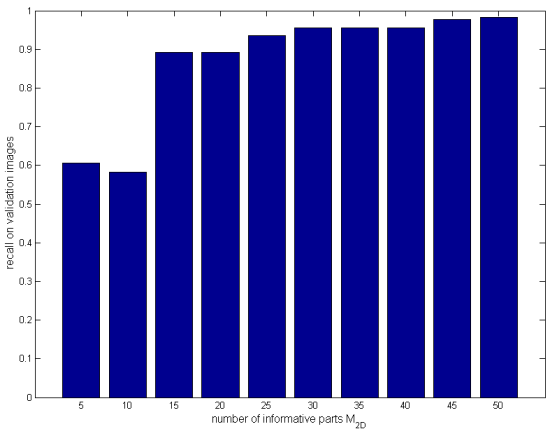

Figure 4.11: Tradeoff between recall on the *validation images* and number of selected object parts M_{2D} for the dense grid of *Star Models* of the object class bicycle.

car	left	front-left	front	front-right	right	back-right	back	back-left
no. of object parts	734	512	225	547	773	622	248	527
no. of all object parts	4188							

Table 4.2: Number of generated object parts for the object class car when dividing the viewsphere of the *Viewsphere Model* into the eight equally spaced viewsphere subspaces given in Figure 4.5.

bicycle	left	front-left	front	front-right	right	back-right	back	back-left
no. of object parts	681	364	38	404	691	390	41	394
no. of all object parts	3003							

Table 4.3: Number of generated object parts for the object class bicycle when dividing the viewsphere of the *Viewsphere Model* into the eight equally spaced viewsphere subspaces given in Figure 4.5.

	car	bicycle
no. of all object parts	1597	936

Table 4.4: Number of generated object parts for the object classes car and bicycle when the object parts are generated on the entire viewsphere of the *Viewsphere Model*.

subdivision during the generation of potential object parts which is described in Section 4.3.2 and the influence of the number of selected object parts N_{2D} which is described in Section 4.3.3.1. For both object classes (i.e. cars and bicycles) we compare the 2D localization performance of the *Viewsphere Model* on the 3D Object Category data set when generating informative object parts from the entire viewsphere or when

dividing the entire viewsphere into the eight equally spaced viewsphere subspaces which are given in Figure 4.5. The numbers of generated object parts for the final pool of potential object parts are given in Table 4.2, Table 4.3, and Table 4.4. The numbers indicate that the viewsphere subdivision results in a significantly increased number of generated object parts. Based on these generated pools P of potential object parts, we train for both object classes a *Viewsphere Model* with $N_{2D} = 50$ and $N_{2D} = 100$ selected object parts. We apply the resulting *Viewsphere Models* to the entire 3D Object Category data set, i.e., to all 480 test images per object class. The precision-recall curves we obtain are shown in Figure 4.12 for the object class car and in Figure 4.13 for the object class bicycle. For cars, the viewsphere subdivision has little effect on the overall detection accuracy. The reason is that for both settings suitable object parts are generated and selected. Consequently, the resulting object class detectors do not suffer from low recall. In addition, the number of selected object parts N_{2D} has little effect on the performance of the corresponding car detector. For the object class bicycle the behavior is different. If the object parts for the bicycle class are generated from the entire viewsphere the corresponding bicycle detector suffers from low recall. The bicycle detector is not able to detect the front and the back views of a bicycle which cover relatively small areas and contain delicate structures. If we choose the eight equally spaced viewsphere subspaces given in Figure 4.5, specific object parts for those viewpoints, i.e., the front and the back view of a bicycle, are established and selected. This results in a significantly higher recall of the final bicycle detector. Furthermore, increasing the number of selected object parts from $N_{2D} = 50$ to $N_{2D} = 100$ object parts improves the overall precision of the bicycle detector.

Based on these results, for the subsequent experiments we choose a viewsphere division into the eight subspaces given in Figure 4.5. For the car detector we select $N_{2D} = 50$ object parts which results in $M_{2D} = 25$ object parts for building the dense grid of *Star Models*. For the bicycle detector we also choose the same subdivision and select $N_{2D} = 100$ object parts which results in $M_{2D} = 50$ object parts for building the dense grid of *Star Models*. For training the pose estimation step, we also rely on the eight subspaces given in Figure 4.5. For each defined subspace v ($1 \leq v \leq 8, v \in \mathbb{N}$), we select a subset with $N_{3D}^v = 50$ object parts for the object class car and a subset with $N_{3D}^v = 100$ object parts for the object class bicycle in order to train the *Spatial Pyramid Models* and to establish the Gaussian mixture models.

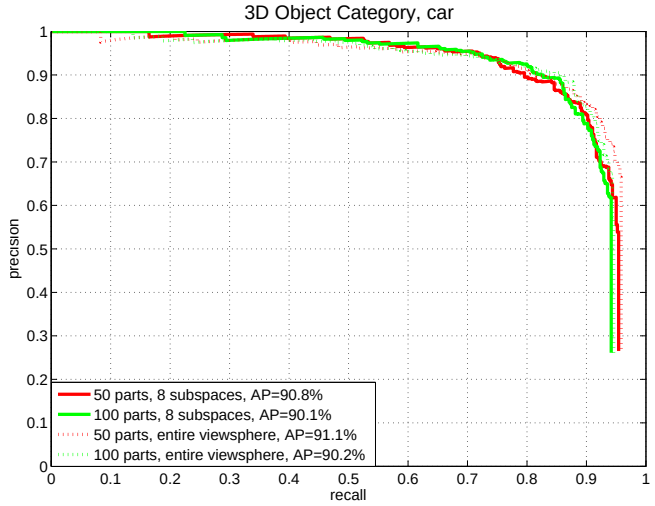

Figure 4.12: Precision-recall curves for the 3D Object Category car data set when using different viewsphere subspaces for generating the final pool of potential object parts (solid/dashed curves) and different numbers N_{2D} of object parts (red/green curves).

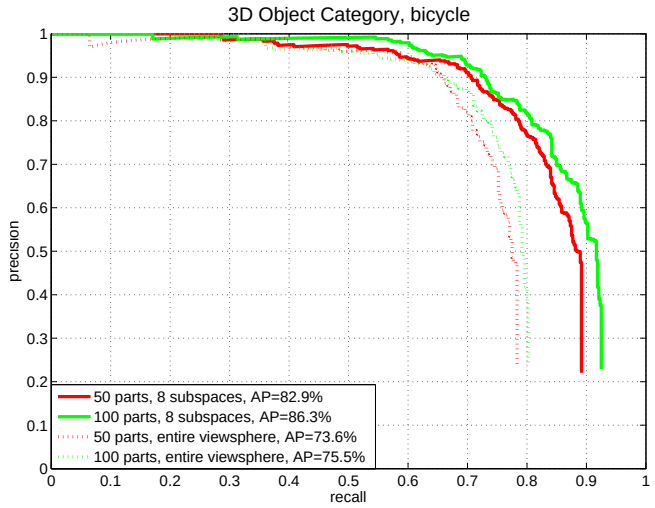

Figure 4.13: Precision-recall curves for the 3D Object Category bicycle data set when using different viewsphere subspaces for generating the final pool of potential object parts (solid/dashed curves) and different numbers N_{2D} of object parts (red/green curves).

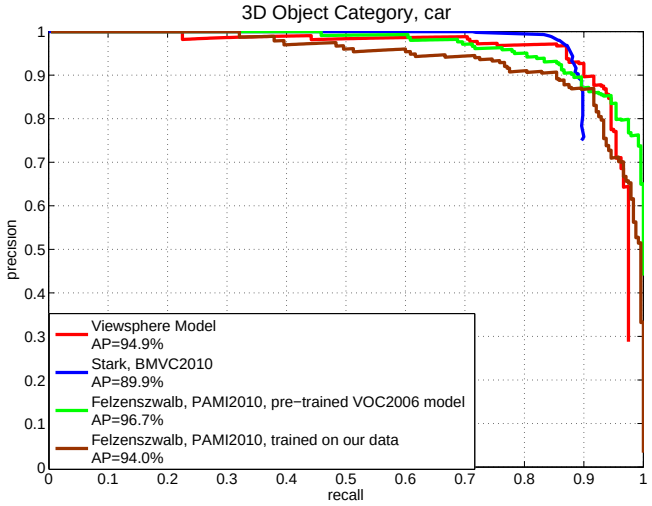

Figure 4.14: Precision-recall curves of the *Viewsphere Model* (red curve) on the 3D Object Category car data set compared to other reported results and the state-of-the-art detection approach of (34).

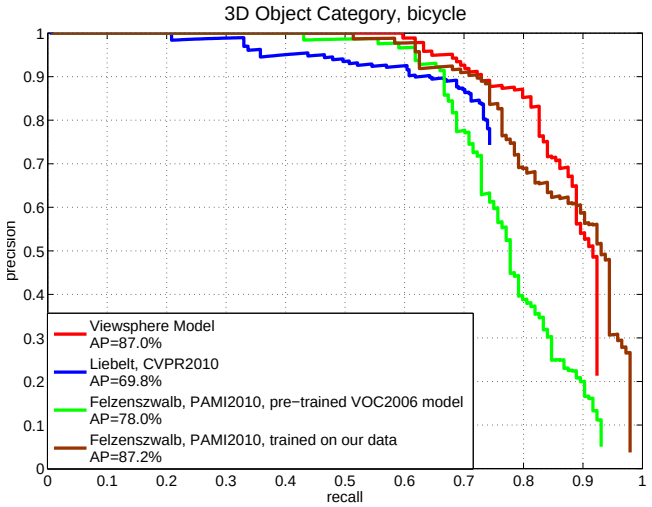

Figure 4.15: Precision-recall curves of the *Viewsphere Model* (red curve) on the 3D Object Category bicycle data set compared to other reported results and the state-of-the-art detection approach of (34).

4.5.2.2 2D Localization

In order to compare the 2D localization performance of the *Viewsphere Model* on the 3D Object Category data set to previous work, we follow the test protocol of (109)

for the object class car and the test protocol of (78) for the object class bicycle. Note that these two test protocols define test subsets and therefore differ from the test setup for the experiments in Section 4.5.2.1 which are evaluated on the entire data set, i.e., containing all 480 test images per object class. In Figure 4.14 we compare the detection performance of the *Viewsphere Model* to the approach of (109). With 94.9% on the car data set the proposed *Viewsphere Model* (red curve) outperforms the approach of (109) (blue curve with 89.9%) due to a higher recall on the test set. Note that the approach of (109), which is also trained on synthetic data, uses a bank of 36 viewpoint-specific models with more than 400 trained object parts. In contrast, the *Viewsphere Model*, which is able to exploit appearance co-occurrences across different defined viewpoints, requires only 50 object parts. The precision-recall curve for the *Viewsphere Model* on the 3D Object Category bicycle data set is given in Figure 4.15. With an average-precision of 87.0% on the bicycle data set the *Viewsphere Model* (red curve) outperforms the approach of (78) (blue curve with 69.8%) due to a significantly higher recall.

For both object classes, we compare the *Viewsphere Model* against the current state-of-the-art approach of (34) using their pre-trained object class models on the PASCAL VOC2006 data set provided as part of *voc-release4* (33). As shown in Figure 4.14, with an average-precision of 94.9% on the car data set the proposed *Viewsphere Model* (red curve) can compete with the state-of-art detector of (34) (green curve with 96.7%) despite being trained on synthetically generated positive training images. As shown in Figure 4.15, on the bicycle test set the *Viewsphere Model* (red curve with 87.0%) outperforms the state-of-the-art detection approach of (34) (green curve with 78.0%) due to a higher precision at a comparable recall. We also evaluate the method of (34) based on our synthetically generated positive training images. To this purpose, we train and evaluate for both object classes a detection model with the recommended settings of 3 components and 8 parts per component. We use the approach of (34) which is provided as part of *voc-release4* (33) in conjunction with the synthetically generated *validation images*[1] as positive training images and the PASCAL VOC2006 training data set as negative training images. The results indicate with an average-precision of 94.0% (brown

[1]We only use the synthetically generated *validation images* as positive training images, since the *pure examples* with a black background might not be suitable as positive training images for the method of (34).

curve in Figure 4.14) on the car test set and with an average-precision of 87.2% (brown curve in Figure 4.15) on the bicycle test set that the proposed *Viewsphere Model* performs on par with the current state-of-the-art object class detection approach of (34). Numerous approaches have been evaluated on the 3D Object Category data set with respect to 2D localization. However, different test configurations have been used which makes an objective and comprehensive benchmarking difficult. To compare to each approach, we evaluate the *Viewsphere Model* with respect to 2D localization using each of the test configurations reported by the different authors on the car data set. The results are shown in Table 4.5. Note that the *Viewsphere Model* performs on par or better than most of these reported detectors with respect to 2D detection, despite being trained on synthetically generated positive training images.

Approach	Reported Test Configuration	AP_{2D}	Own AP_{2D}
Glasner (54)	5 inst./3 scales	**99.2%**	94.9%
Liebelt (78)	3 inst./3 scales	76.7%	**97.2%**
Stark (109)	5 inst./3 scales	89.9%	**94.9%**
Su (110)	5 inst./2 scales	55.3%	**94.9%**
Sun (112)	5 inst./2 scales	—	**94.9%**
Zia (132)	5 inst./3 scales	90.4%	**94.9%**

Table 4.5: We evaluate the *Viewsphere Model* with respect to 2D localization following the previously reported test protocols on the 3D Object Category car data set in order to achieve an objective comparison (abbreviation: inst.=object instance, AP_{2D}=average-precision for 2D localization).

4.5.2.3 Pose Estimation

In order to benchmark the pose estimation performance of the *Viewsphere Model* on the 3D Object Category data set, we bin the estimated viewpoints of the pose estimation step (see Section 4.4.2) in 45° steps to match to the ground truth annotations defined by (100). Here, we follow the test protocol of (109) for the object class car and the test protocol of (78) for the object class bicycle to compare the pose estimation performance of the *Viewsphere Model* to existing approaches. The confusion matrices obtained by classifying all positive detections of Section 4.5.2.2 are shown in Figure 4.16 for the object class car and in Figure 4.17 for the object class bicycle. For the object class car we observe that confusion is more pronounced for opposing views due to the symmetries inherent in the car class. For example, 27.0% of back-right views

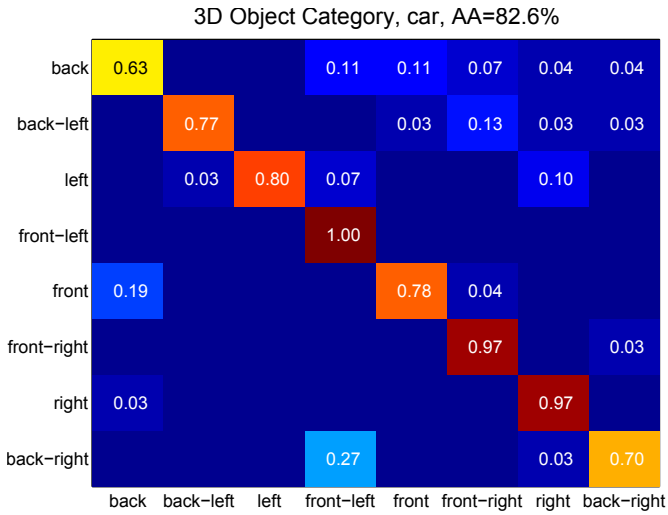

Figure 4.16: Confusion matrix (rows: ground truth, columns: estimates) for the 3D Object Category car data set.

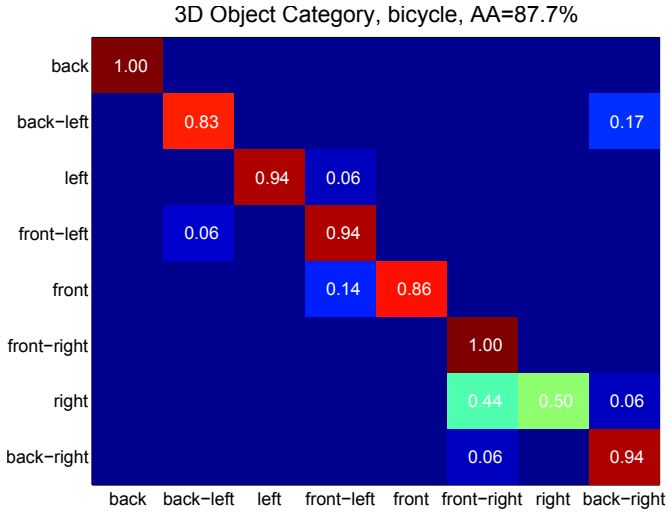

Figure 4.17: Confusion matrix (rows: ground truth, columns: estimates) for the 3D Object Category bicycle data set.

are classified as front-left views. Still, with an average-accuracy of 82.6% the pose estimation step of the *Viewsphere Model* compares favorably to the reported result of (109) with 80.5%. For the object class bicycle we observe that confusion is more pronounced between neighboring viewpoints. For example, 44.0% of the right views are classified as front-right views. On the bicycle data set, the achieved result of 87.7% significantly outperforms the approach of (78) with 75.0%. Some successful

detection results with the full detection process of the *Viewsphere Model* on the 3D Object Category data set are shown for the object class car in Figure 4.27 and for the object class bicycle in Figure 4.28. Figure 4.31 depicts some failed detection results of the *Viewsphere Model* for the object class car and the object class bicycle. The failed detections on the 3D Object Category data set are mainly caused by sub detections within an object instance or when there is insufficient evidence in the image for a correct pose initialization.

Similar to Section 4.5.2.2, we compare the *Viewsphere Model* with respect to pose estimation to other reported results. Table 4.6 shows that the *Viewsphere Model* also performs on par or better than most of these reported detection approaches with respect to pose estimation.

Approach	Reported Test Configuration	AA_{3D}	Own AA_{3D}
Glasner (54)	5 inst./3 scales	**84.9%**	82.6%
Liebelt (78)	3 inst./3 scales	70.0%	**81.5%**
Stark (109)	5 inst./3 scales	80.5%	**82.6%**
Su (110)	5 inst./2 scales	≈69.4%	**83.6%**
Sun (112)	5 inst./2 scales	66.6%	**83.6%**
Zia (132)	5 inst./3 scales	**84.0%**	82.6%

Table 4.6: We evaluate the *Viewsphere Model* with respect to pose estimation following the previously reported test protocols on the 3D Object Category car data set in order to achieve an objective comparison (abbreviation: inst.=object instance, AA_{3D}=average-accuracy for pose estimation).

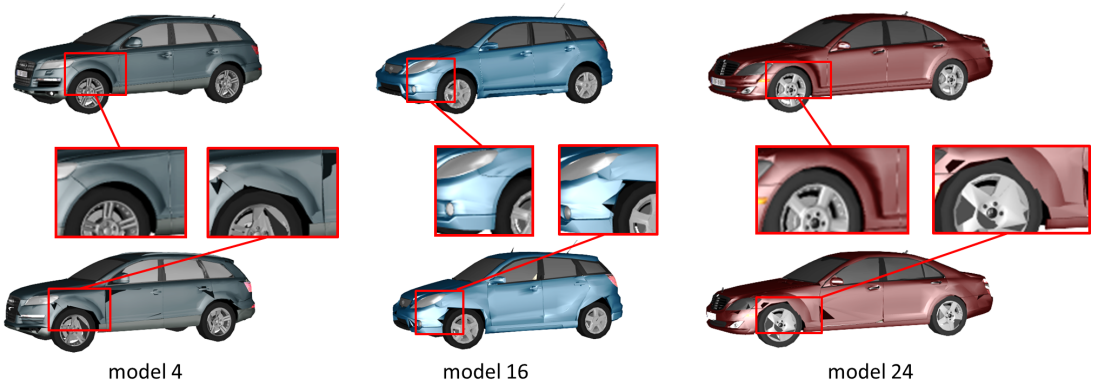

Figure 4.18: Some examples of CAD models with a reduced quality (bottom) from the car database (see Appendix B) compared to their original CAD models (top). The number of vertices is reduced by 25% resulting in visible defects of the CAD models.

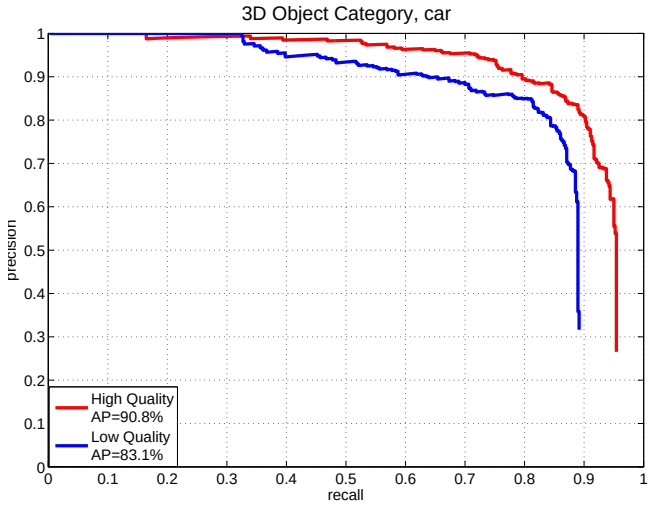

Figure 4.19: The influence of the CAD models' quality on the detection performance of the *Viewsphere Model*. A database of CAD models with a reduced quality, i.e., a small number of vertices, results in a significantly lower 2D localization performance of the proposed *Viewsphere Model*.

4.5.2.4 Influence of CAD Models' Quality

In this experiment we assess the influence of the CAD models' quality on the 2D localization performance of the proposed *Viewsphere Model*. To this purpose, we reduce the number of vertices for all CAD models of our car database (see Appendix B) by 25% resulting in a database of car models with visible defects[1]. Some examples of car models with such a reduced quality are shown in Figure 4.18. Based on this database of CAD models with a reduced quality, we train a *Viewsphere Model* for the object class car with the 8 subspaces given in Figure 4.5 and $N_{2D} = 50$ object parts to be comparable with the previous results of the *Viewsphere Model*. We follow the test protocol of Section 4.5.2.1 and apply the resulting *Viewsphere Model* to the entire 3D Object Category car data set, i.e., to all 480 test images. As shown in Figure 4.19, the *Viewsphere Model*, relying on a database with a reduced quality of the CAD models, achieves a 2D localization performance of 83.1% (blue curve) which is below the detection performance of the *Viewsphere Model* relying on the original CAD

[1]To this purpose, we rely on the polygon reduction system of NuGraf® distributed by Okino Computer Graphics which automatically adapts the textures and the materials of a CAD model when reducing the number of vertices. Details on this polygon reduction system are given in (52).

model database without any visible defects (red curve with 90.8%). Consequently, a database of CAD models with sufficiently many vertices, i.e., CAD models with a high quality without any visible defects, is necessary to establish a powerful *Viewsphere Model* for a specific object class.

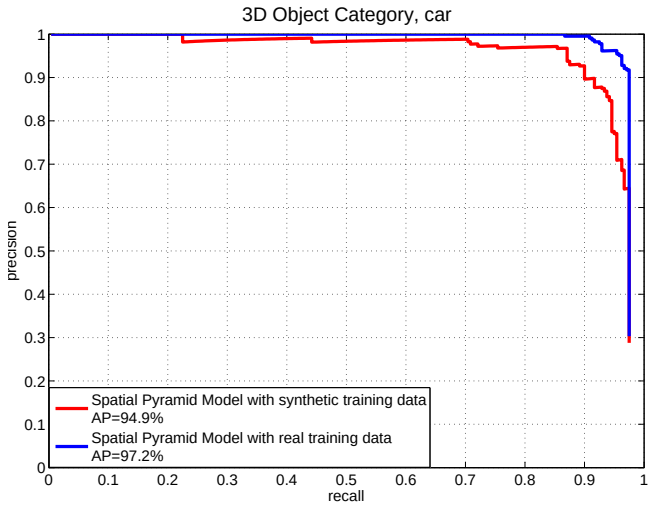

Figure 4.20: Using real training images for the *Spatial Pyramid Model* of the *Viewsphere Model* results in a slightly better 2D localization performance. However, this performance gain comes with the shortcoming of using real training images, i.e., the supervision of training data.

4.5.2.5 Influence of Real Training Images

In this experiment we evaluate the influence of real training images on the proposed *Viewsphere Model*. To this purpose, we adapt the trained *Viewsphere Model* for the object class car of Section 4.5.2.1 and learn the corresponding *Spatial Pyramid Model* with real training examples instead of synthetically generated *validation images*. In order to be comparable with the 2D localization results of Section 4.5.2.2, we follow the test protocol of (109) by dividing the 3D Object Category car data set into 240 training images (i.e. 5 cars in total) and 240 test images (i.e. 5 cars in total). We use the training images as training source for learning the *Spatial Pyramid Model* of the *Viewsphere Model* as described in Section 4.3.3.3. As a result, we obtain a car detector which consists of a synthetically trained pre-detection step and a verification step based on real training images. Consequently, the resulting *Viewsphere Model* is based on a mix-

ture of real and synthetic training images, similar to the approach of (78). As shown in Figure 4.20, the *Viewsphere Model*, consisting on both synthetic and real training images (blue curve with 97.2%), achieves a sightly better performance due to a higher precision, compared to the *Viewsphere Model* which relies only on synthetic training images (red curve with 94.9%). Consequently, training the *Viewsphere Model* with both synthetic and data set specific images results in a slightly better 2D localization performance. However, this small performance gain has the disadvantage of using real training images (see Section 1.2), which in this case do not seem to justify the use of real training images for the *Viewsphere Model*. In addition, this small performance gain could also be the result of an adaption of the *Viewsphere Model* to the data set characteristics of the 3D Object Category data set at the expense of its generalization abilities.

4.5.3 PASCAL VOC2006 Data Set

The precision-recall curves on the PASCAL VOC2006 data set obtained with the *Viewsphere Model* are given in Figure 4.21 (red curve) for the object class car and in Figure 4.22 (red curve) for the object class bicycle. For both object classes we provide the best performing approaches of the PASCAL challenge 2006 (28) (blue curves), the best performing approaches of the PASCAL challenge 2007 on the 2006 test set (29) (cyan curves), and the most recent multi-view approaches of (110) (green curves) and (95) (brown curves). With 40.4% on the car test set and 44.5% on the bicycle test set, the *Viewsphere Model* achieves a higher average-precision than these two multi-view approaches, despite being trained on synthetically generated positive training images. Similar to the *Multi-View Model* (see Section 3.6.3), we observe that the appearance variations within the car test set are more pronounced than those within the bicycle class which might be the reason for the observed performance difference of the *Viewsphere Model* on these two object classes: while the chosen synthetic bicycle models and the corresponding *validation images* and *pure examples* are sufficient to represent the appearance variations of the bicycle test set, the synthetic car models and the corresponding *validation images* and *pure examples* seem to be not representative enough. Some successful detection results of the *Viewsphere Model* on the PASCAL VOC2006 data set are shown for the class car in Figure 4.29 and for the class bicycle in Figure 4.30. Figure 4.32 depicts some failed detections of the *Viewsphere Model* for

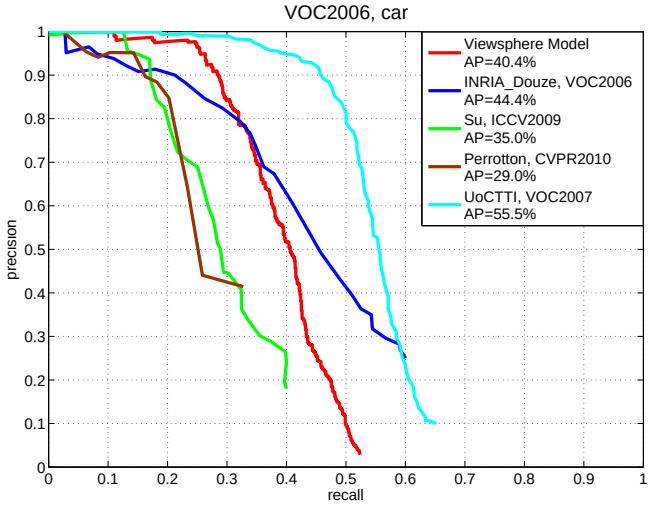

Figure 4.21: Precision-recall curves of the *Viewsphere Model* (red curve) for the object class car on the PASCAL VOC2006 data set compared to other reported results.

both object classes. The failed detections are mainly caused by sub detections within other object instances or nonrigid geometries. Note that in contrast to the 3D Object Category data set of Section 4.5.2, we do not perform the pose estimation step of Section 4.4.2 on the PASCAL VOC2006 data set, instead providing only a 2D bounding box.

4.5.4 Comparison to the Multi-View Model

In this section, we compare the achieved results of the *Viewsphere Model* to those of the *Multi-View Model* on the 3D Object Category data set and PASCAL VOC2006 data set.

In order to compare the results of the *Viewsphere Model* with the results of the *Multi-View Model* on the 3D Object Category data set for the object classes car and bicycle, we follow the test protocol of Section 4.5.2.1 and apply the trained detectors to the entire 3D Object Category data set, i.e., to all 480 test images per object class. Note that the test protocol of Section 4.5.2.1 differs from the two test protocols used in Section 4.5.2.2 and in Section 4.5.2.3, since these two test protocols define a subset of test images. The results of the two different approaches, i.e., the *Multi-View Model* and the *Viewsphere Model*, with respect to 2D localization are given in Figure 4.23. For both object classes,

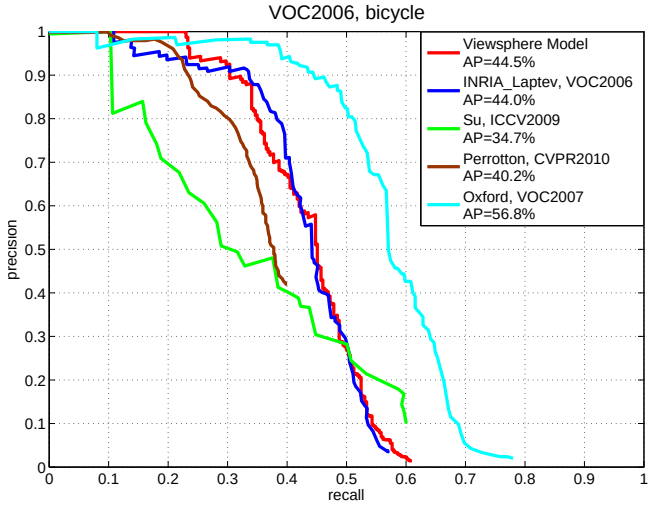

Figure 4.22: Precision-recall curves of the *Viewsphere Model* (red curve) for the object class bicycle on the PASCAL VOC2006 data set compared to other reported results.

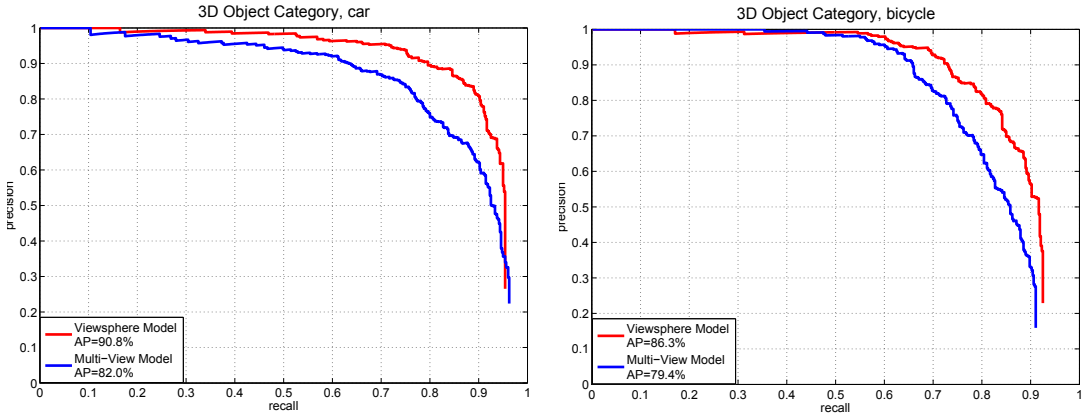

Figure 4.23: Comparison of the *Viewsphere Model* (red curves) with the *Multi-View Model* (blue curves) with respect to 2D localization on the 3D Object Category car data set (left) and the 3D Object Category bicycle data set (right).

the *Viewsphere Model* outperforms the *Multi-View Model* due to a higher precision. The confusion matrices obtained by classifying the corresponding positive detections are shown in Figure 4.24 for the object class car and in Figure 4.25 for the object class bicycle. In the case of pose estimation the *Viewsphere Model* significantly outperforms the *Multi-View Model* due to a set of discriminatively trained *Spatial Pyramid Models* where each *Spatial Pyramid Model* is directly linked to a discretized subspace of the

defined viewsphere. We observe that for the object class car the confusion for opposing views and for the object class bicycle the confusion for neighboring views is significantly reduced by the *Viewsphere Model* compared to the *Multi-View Model*. The comparison of the *Viewsphere Model* with the *Multi-View Model* on PASCAL VOC2006 data set for the object classes car and bicycle is given in Figure 4.26. For both proposed approaches, we observe a similar detection performance on both test sets.

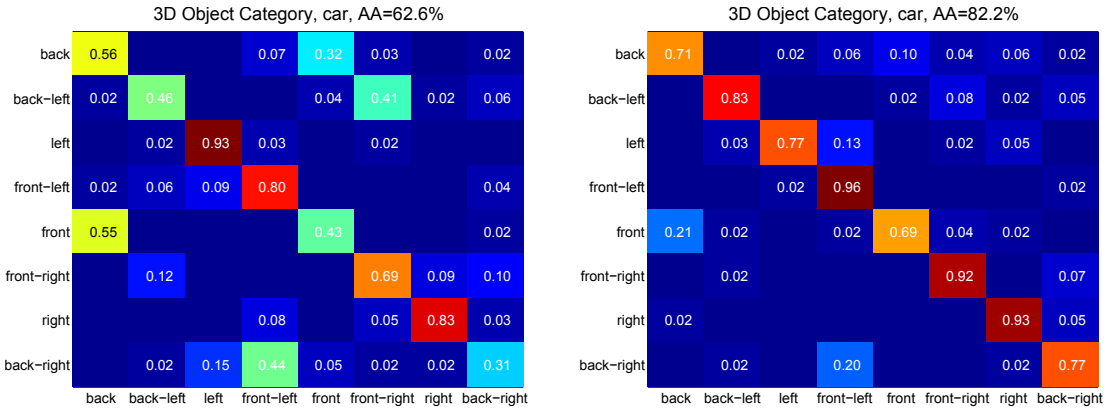

Figure 4.24: Comparison of the *Multi-View Model* (left) with the *Viewsphere Model* (right) with respect to pose estimation on the 3D Object Category car data set.

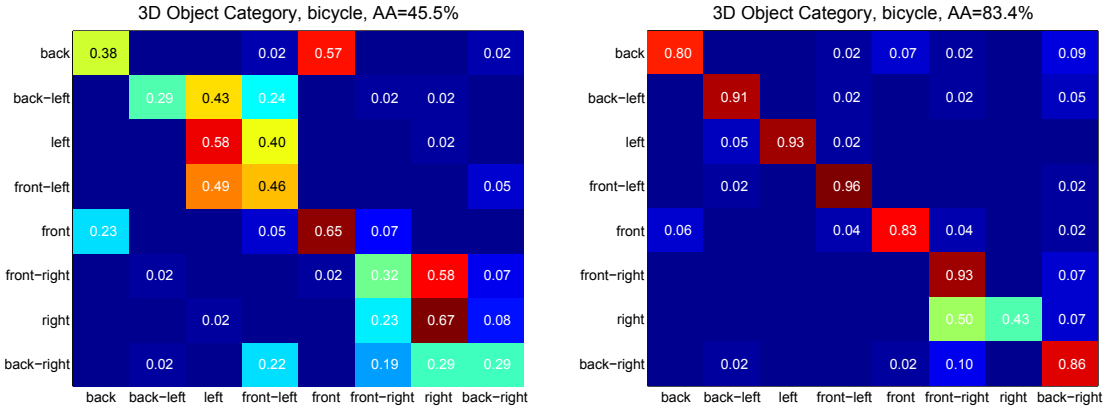

Figure 4.25: Comparison of the *Multi-View Model* (left) with the *Viewsphere Model* (right) with respect to pose estimation on the 3D Object Category bicycle data set.

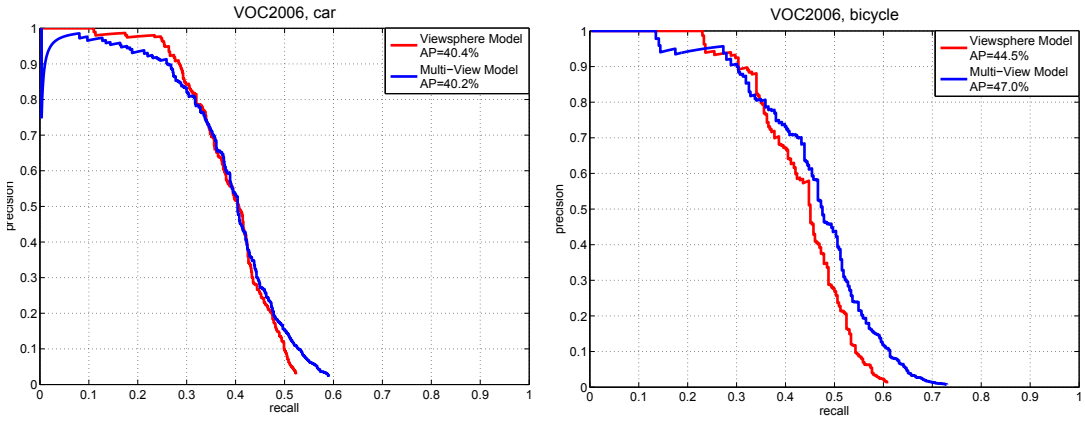

Figure 4.26: Comparison of the *Viewsphere Model* (red curves) with the *Multi-View Model* (blue curves) with respect to 2D localization on the PASCAL VOC2006 data set for the object class car (left) and the object class bicycle (right).

4.6 Summary

In this chapter, we have presented the *Viewsphere Model* which is an approach to 2D localization and pose estimation. The *Viewsphere Model* learns an object class representation from a database of CAD models and a set of real negative training images. Similar to the *Multi-View Model* of the previous chapter, the *Viewsphere Model* integrates the generative *Star Model* and the discriminative *Spatial Pyramid Model* in one common object class detection framework. However, in contrast to the *Multi-View Model*, the *Viewsphere Model* exploits appearance co-occurrences due to viewpoint symmetries and part similarities by choosing non-semantic object parts. The 2D localization performance of the *Viewsphere Model* is on par with the current state-of-the-art detection approach of (34) and we show an increased robustness of the *Viewsphere Model* with respect to pose estimation compared to the *Multi-View Model*. In the following chapter, we extend the *Viewsphere Model* to cope with multiple object classes simultaneously.

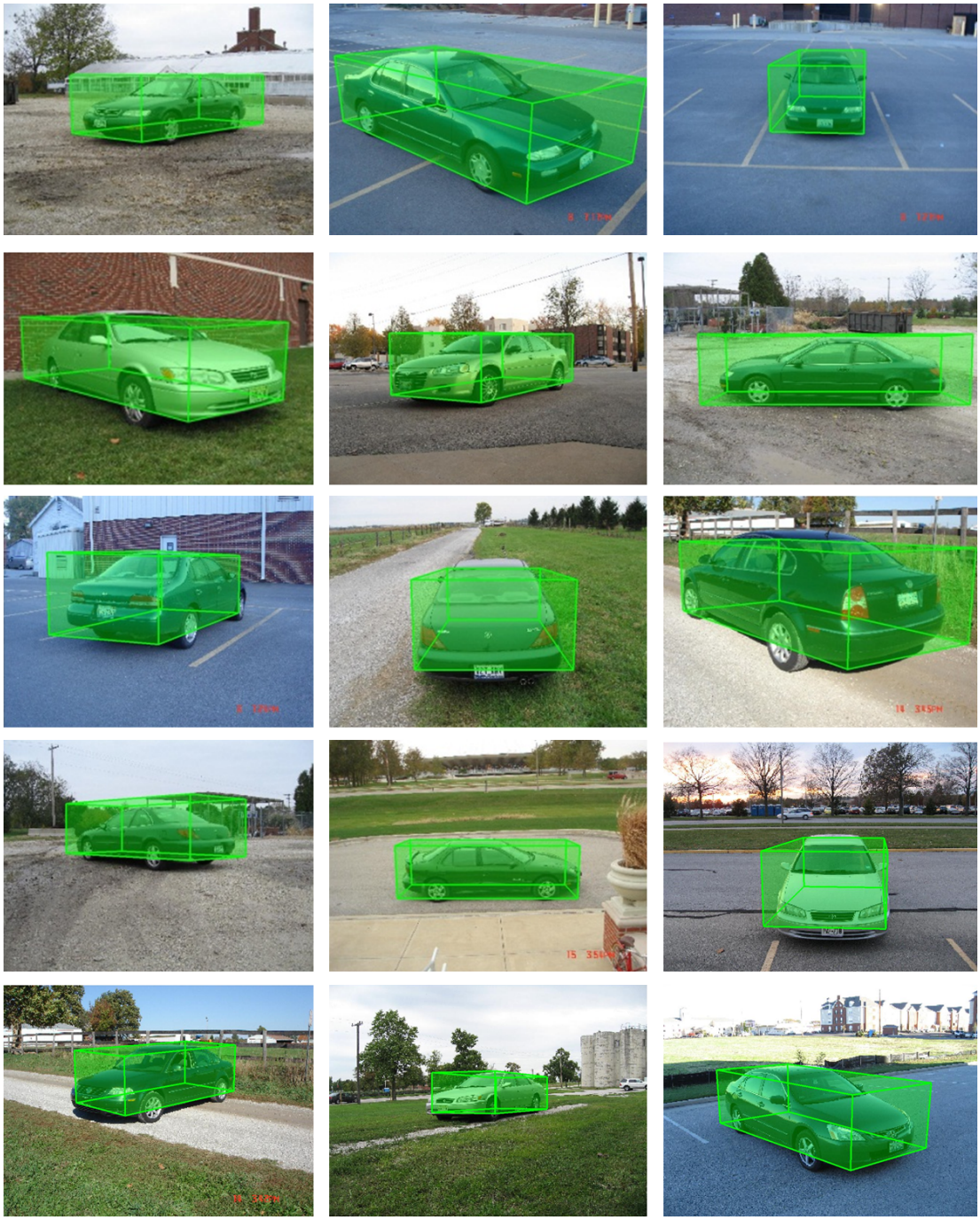

Figure 4.27: Some successful detection results with the full detection process of the *Viewsphere Model* on the 3D Object Category car data set. We show only the highest scored detection per image.

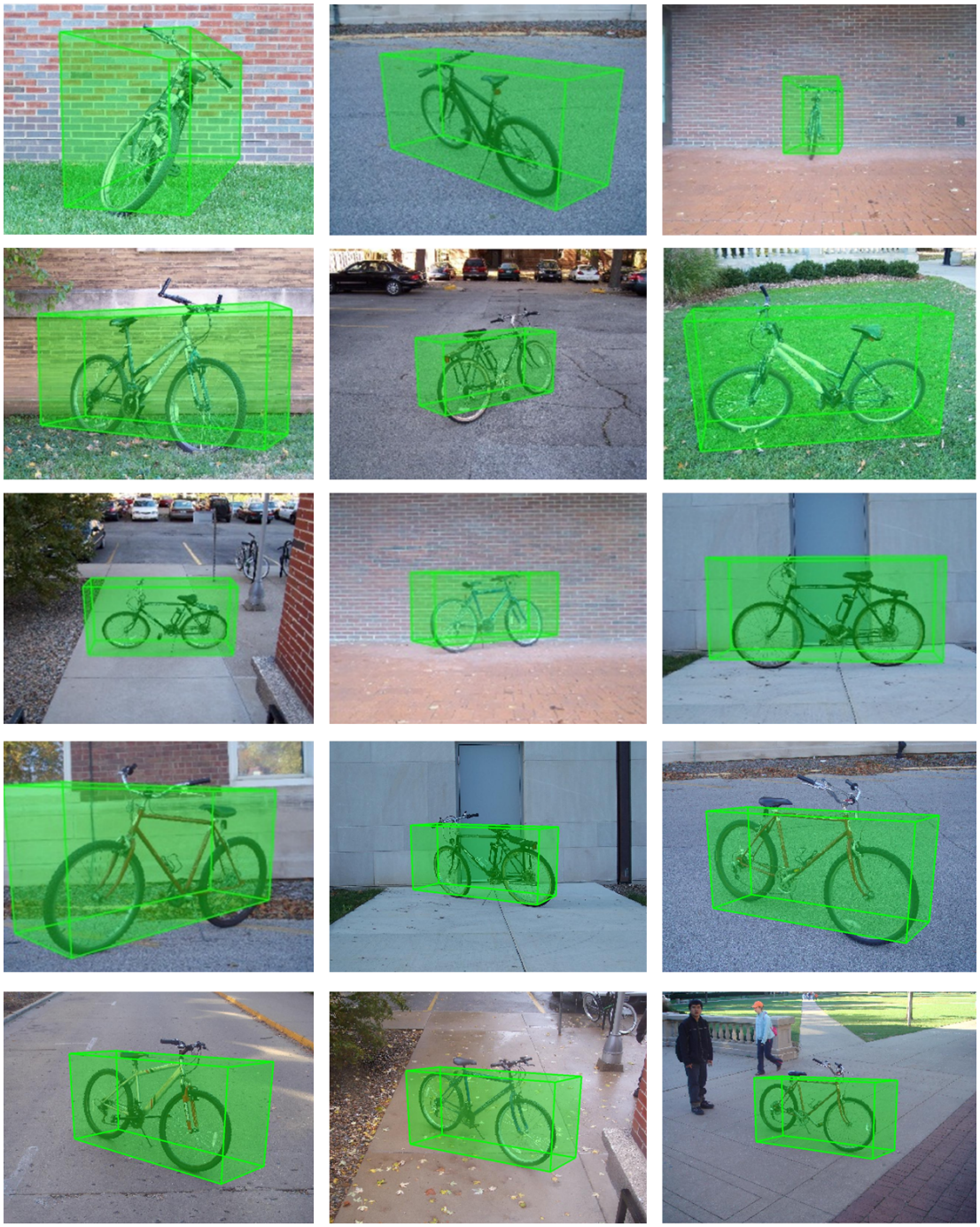

Figure 4.28: Some successful detection results with the full detection process of the *Viewsphere Model* on the 3D Object Category bicycle data set. We show only the highest scored detection per image.

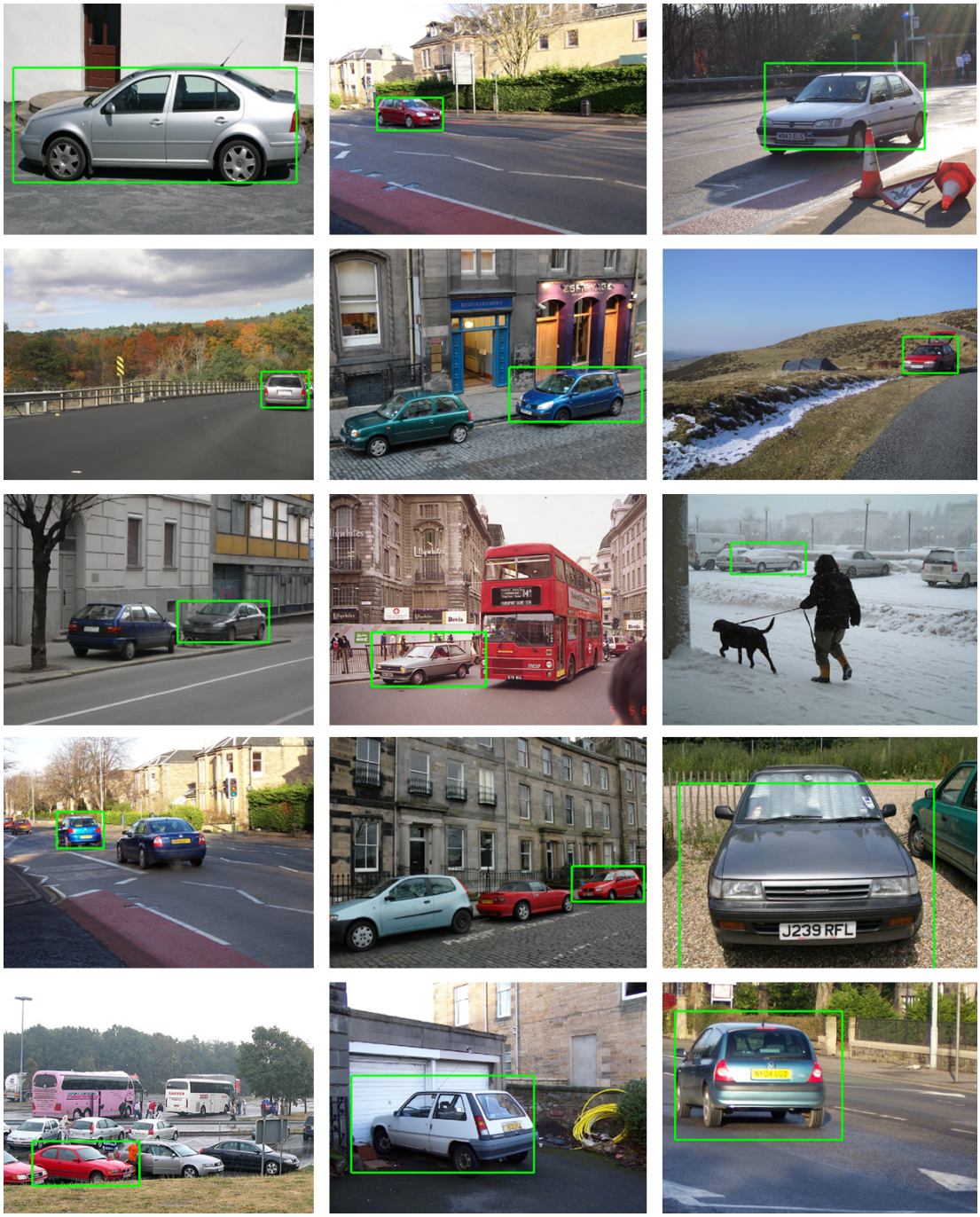

Figure 4.29: Some successful detection results of the *Viewsphere Model* for the object class car on the PASCAL VOC2006 data set. Note that the PASCAL VOC2006 data set is only evaluated with respect to 2D localization resulting in a 2D bounding box. We show only the highest scored detection per image.

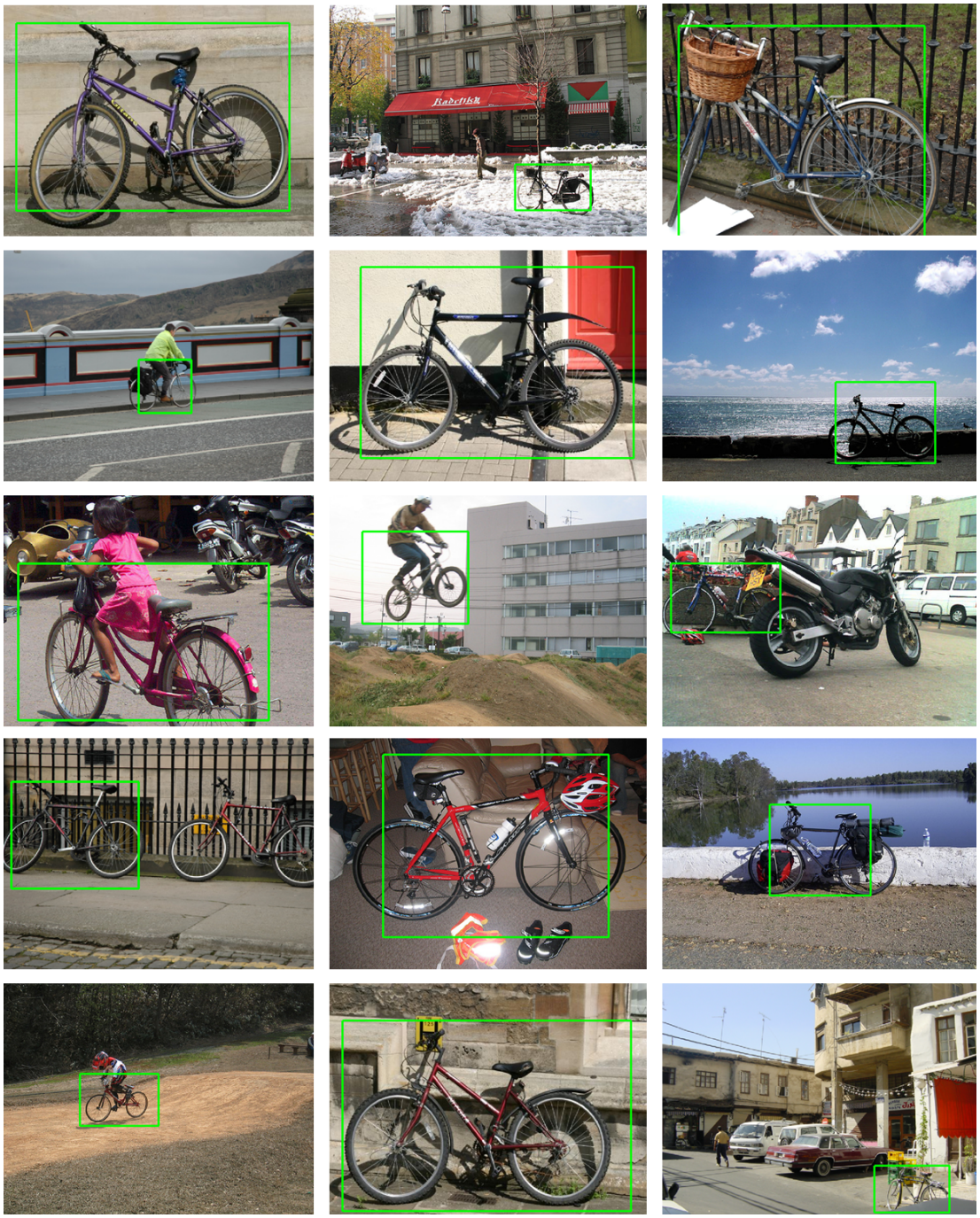

Figure 4.30: Some successful detection results of the *Viewsphere Model* for the object class bicycle on the PASCAL VOC2006 data set. Note that the PASCAL VOC2006 data set is only evaluated with respect to 2D localization resulting in a 2D bounding box. We show only the highest scored detection per image.

Figure 4.31: Some failed detection results of the *Viewsphere Model* on the 3D Object Category car data set (top) and on the 3D Object Category bicycle data set (bottom). The failed detections are mainly caused by sub detections within an object instance or when there is insufficient evidence in the image for a correct pose initialization.

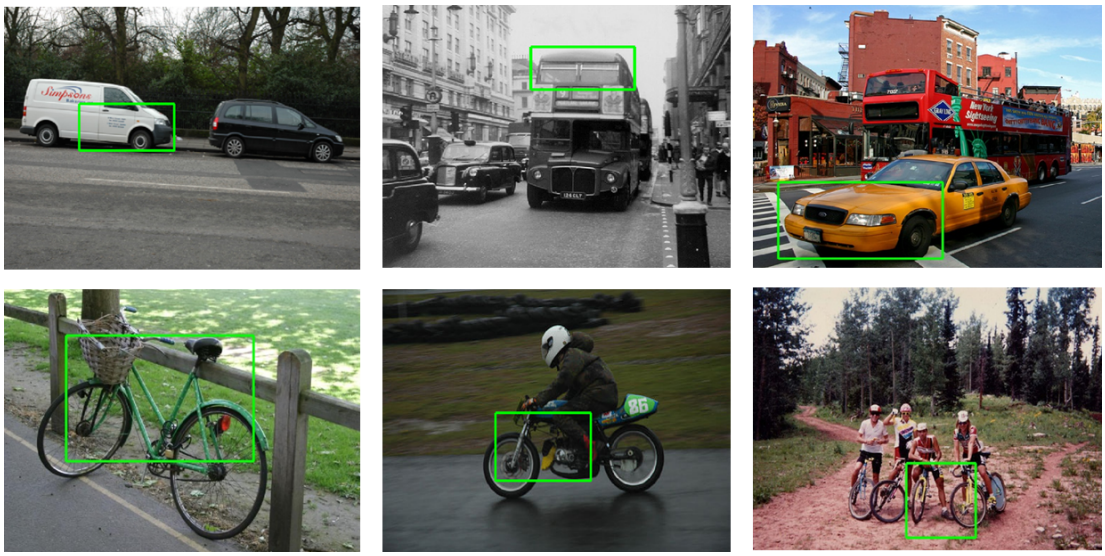

Figure 4.32: Some failed detection results of the *Viewsphere Model* on the PASCAL VOC2006 data set for the object class car (top) and for the object class bicycle (bottom). The failed detections are mainly caused by sub detections within other object instances or a nonrigid geometry. Note that the PASCAL VOC2006 data set is only evaluated with respect to 2D localization resulting in a 2D bounding box.

Chapter 5

The Viewsphere Model for Multiple Object Classes

In this chapter, we extend the *Viewsphere Model* of Chapter 4 in order to cope with multiple object classes. First, we summarize previous work on multi-class object detection and second, we present three learning approaches based on the training steps of the *Viewsphere Model* with different part sharing strategies. This chapter concludes with an experimental evaluation and a comparison of these three learning methods.

5.1 Introduction

Learning and recognizing multiple object classes from arbitrary viewpoints is still in its infancy. Several approaches address the problem of viewpoint-independent object class detection (54, 79, 100, 102, 110) or multi-class object detection (31, 42, 90, 98, 108, 117). Most of these approaches consider these two problems in isolation, i.e., either a viewpoint-independent representation of an object class is built (54, 102, 110) or multiple object classes are trained from discrete viewpoints (42, 90, 117). In contrast to these approaches, we present in this chapter three learning strategies based on the *Viewsphere Model* in order to represent multiple object classes on a densely sampled viewsphere. Altogether we describe, evaluate, and compare the following learning strategies: first, an independent learning strategy, which trains each object class separately from all other object classes, second, a joint learning strategy, which trains all object classes simultaneously, and third, a sequential learning strategy, which learns one object class

after another. All presented learning strategies rely on the part-based object class representation of the *Viewsphere Model* which derives its positive training examples from a database of textured CAD models. The 2D localization performance of these learning strategies is evaluated on the Multi-Class data set of Section 2.5.3 consisting of images from the 3D Object Category data set and the PASCAL VOC2006 data set. Our experiments indicate that the sequential learning strategy achieves the best result with respect to 2D localization performance and flexibility during the training process and thus could be suitable for learning multiple object classes from arbitrary viewpoints on a larger scale.

This chapter is structured as follows: Section 5.2 gives an overview of previous work on multi-class object detection. In Section 5.3 we present the three different learning strategies based on the training steps of the *Viewsphere Model* and their detection procedures are outlined in Section 5.4. Experimental results and a comparison of these learning strategies are given in Section 5.5. This chapter is concluded in Section 5.6.

5.2 Related Work

In general, there are three main strategies for learning to represent multiple object classes: first, an independent learning which trains each object class separately from all other object classes. Second, a joint learning (117) which trains all object classes simultaneously and third, a sequential learning which trains one object class after another (90). In (117) multiple object classes are trained jointly based on boosted decision stumps to find common features. A variation of (117) is proposed in (90) which enables a sequential addition of a new object class without a retraining of the previously learnt object classes. In the context of learning object classes from a small number of training samples (5, 9, 31, 75, 108) the sequential learning is also termed knowledge transfer or one-shot learning. In (31) the priors of probabilistic models are adapted by a few training samples to represent new object classes and in (5) a template from a previously trained object class is used to regularize the training of a novel object class. Bart and Ullman (9) replace features from known object classes with ones from a new but visually similar object class. Levi et al. (75) use prior information about a novel object class in order to assist a feature selection process. Stark et al. (108) propose a shape-based model which enables full or partial knowledge transfer. All mentioned

approaches have in common that they either learn the object classes from just a few discrete viewpoints (31, 108) or they perform knowledge transfer within visually similar object classes (5, 9, 75).
In contrast, we present in this chapter three learning strategies to represent multiple (also visually dissimilar) object classes on a densely sampled viewsphere. To this purpose, we rely on the training steps of the *Viewsphere Model* where an object class representation is learnt by relying on a database of CAD models and a set of real negative images. Our work is related to (42) where a hierarchical framework is used to compare different types of multi-class learning strategies. However, in contrast to our work, (42) restrict possible synergies among the object classes to a few discrete viewpoints.

5.3 Multi-Class and Multi-View Learning Strategies

In this section, we present three learning methods with different part sharing strategies, which rely on the training steps of the *Viewsphere Model* to represent C object classes on a densely sampled viewsphere: an independent (I), a joint (J), and a sequential (S) learning strategy. By relying on the part-based representation of the *Viewsphere Model*, we follow common multi-class approaches which also decompose each object class into object parts (9, 98, 117).

5.3.1 Independent Learning

The first learning strategy is an independent learning of all C $(1 \leq c \leq C, c \in \mathbb{N})$ object classes (see Table 5.1 for pseudo code). Based on the *pure examples* E^c_{pure} and the *validation images* E^c_{val}, each object class c is trained independently from all other object classes. For each object class c a pool P^c_I of independent and object class specific parts is generated (see Section 4.3.2) and for each object class the N^c_I most informative object parts from the established pool P^c_I are selected (see Section 4.3.3.1). Based on a subset of M^c_I object parts and the *pure examples* E^c_{pure}, a dense grid of *Star Models* is established for each object class c (see Section 4.3.3.2). Finally, for each object class c a *Spatial Pyramid Model* with the N^c_I selected object parts is learnt on the corresponding *validation images* E^c_{val} (see Section 4.3.3.3).
Training each object class independently from all other object classes has the advantage

Independent Learning of C Object Classes:
for $c := 1$ **to** C
- generate a pool P_I^c of object parts based on the *pure examples* E_{pure}^c
- select the N_I^c most informative object parts from P_I^c with an entropy-based measure
- model a dense grid of *Star Models* based on the *pure examples* E_{pure}^c
- learn a *Spatial Pyramid Model* with all N_I^c object parts on the *validation images* E_{val}^c

end

Table 5.1: An independent learning strategy based on the training steps of the *Viewsphere Model*. For a single object class ($C = 1$) this strategy reduces to the *Viewsphere Model* described in Chapter 4.

that a new object class can easily be added without retraining the previously learnt object classes (42). However, object parts are not shared among object classes which implying that the computational complexity of the overall representation grows linearly with the number of object classes, as shown in (117).

5.3.2 Joint Learning

The second learning strategy is a joint learning of all C ($1 \leq c \leq C, c \in \mathbb{N}$) object classes. An overview of the joint learning strategy is outlined in Figure 5.1 and the corresponding pseudo code is given in Table 5.2. Based on the *pure examples* of all object classes $E_{pure} = \bigcup_{c=1}^{C} E_{pure}^c$ a common pool P_J of object parts is generated (see Section 4.3.2) and the N_J most informative object parts, which cover all object classes at once, are selected from the established pool P_J (see Section 4.3.3.1). Subsequently, a dense grid of *Star Models* is established for each object class c by using the corresponding *pure examples* E_{pure}^c and a common subset of M_J object parts (see Section 4.3.3.2). Finally, one common *Spatial Pyramid Model*, covering all object classes at once, with all N_J selected object parts is learnt on the *validation images* of all object classes $E_{val} = \bigcup_{c=1}^{C} E_{val}^c$ (see Section 4.3.3.3).

The properties of the joint learning strategy are opposed to the properties of the independent learning strategy: for joint learning, adding a new object class to an already ex-

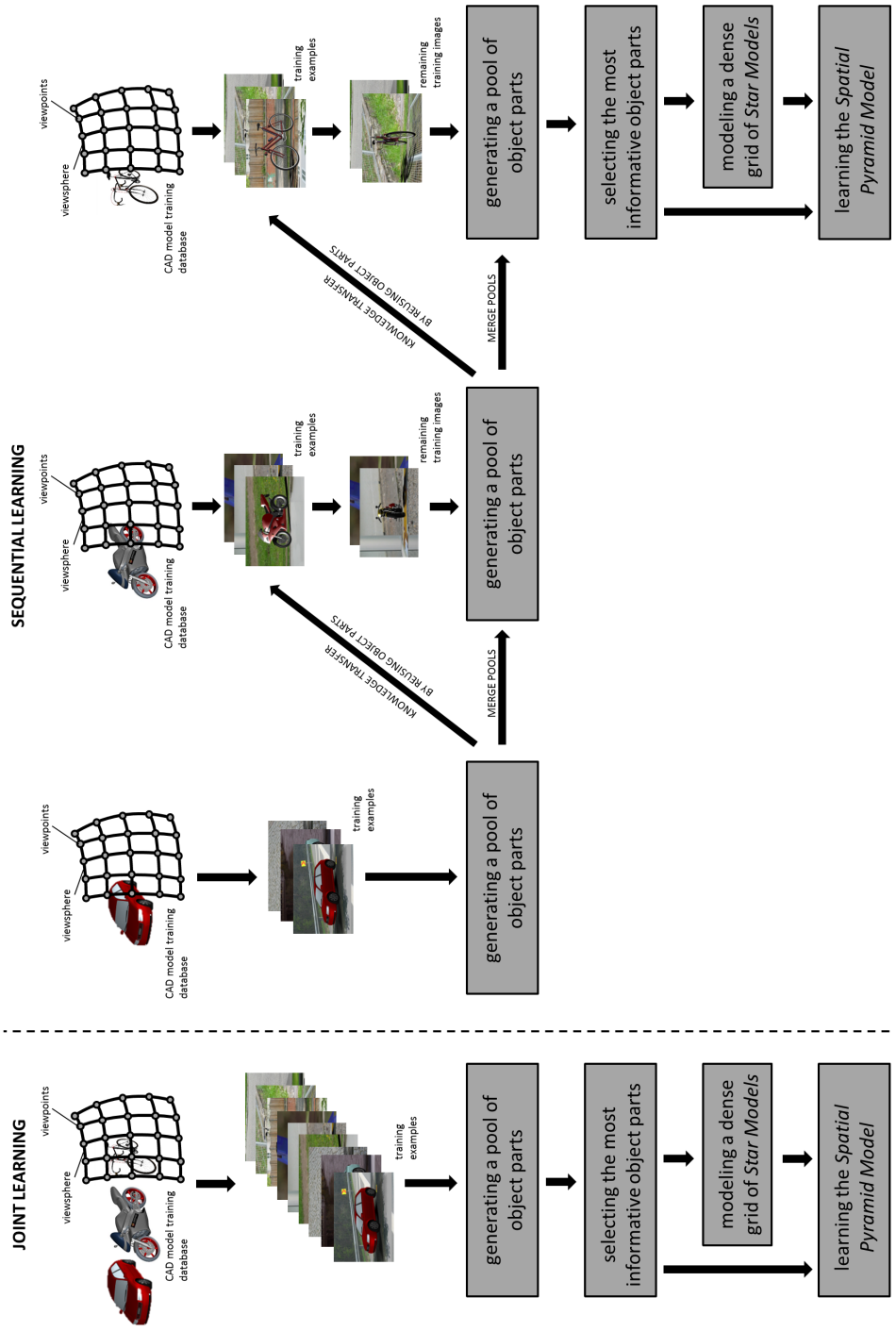

Figure 5.1: Overview of the joint and sequential learning strategy based on the training steps of the *Viewsphere Model*. The corresponding pseudo code is given in Table 5.2 for the joint learning and in Table 5.3 for the sequential learning. The term knowledge transfer stems from machine learning literature (30) and is described with respect to this work in Section 5.3.3.

Joint Learning of C Object Classes:
- generate a common pool P_J of object parts based on the *pure examples* $E_{pure} = \bigcup_{c=1}^{C} E^c_{pure}$ of all object classes
- select the N_J most informative object parts from P_J with an entropy-based measure

for $c := 1$ **to** C
- model a dense grid of *Star Models* based on the *pure examples* E^c_{pure}

end
- learn one common *Spatial Pyramid Model* with all N_J object parts on the *validation images* $E_{val} = \bigcup_{c=1}^{C} E^c_{val}$ of all object classes

Table 5.2: A joint learning strategy based on the training steps of the *Viewsphere Model*. For a single object class ($C = 1$) this strategy reduces to the *Viewsphere Model* described in Chapter 4.

isting multi-class representation is not possible without retraining all previously trained object classes from scratch. As shown in (117), a joint learning of multiple object classes normally reduces the computational complexity of the overall representation by finding common object parts that can be shared across different object classes. In the following section, we describe a sequential learning strategy which combines the advantages of both the independent and the joint learning strategy (42).

Figure 5.2: Examples for *transferable object parts*: from the bicycle class to the motorbike class (left) and from the bicycle class to the car class (right).

5.3.3 Sequential Learning

In this thesis, the knowledge of an object class is defined as the appearance of its corresponding object parts. When learning one object class after another, we are able to transfer object parts (i.e. knowledge) from previously trained object classes to novel

Sequential Learning of C Object Classes:
- generate an initial pool P_S^1 of object parts based on the *pure examples* E_{pure}^1 of the first (c=1) object class

for $c := 2$ **to** C
- determine the *transferable object parts* from the pool P_S^{c-1} to the novel object class c in order to obtain the *remaining examples* E_{remain}^c
- generate a pool $\overline{P}_S^c$ of object parts based on the *remaining examples* E_{remain}^c of the novel object class c
- merge the resulting object part pools $P_S^c = P_S^{c-1} \cup \overline{P}_S^c$

end
- select the N_S most informative object parts from P_S^C with an entropy-based measure

for $c := 1$ **to** C
- model a dense grid of *Star Models* based on the *pure examples* E_{pure}^c

end
- learn one common *Spatial Pyramid Model* with all N_S object parts on the *validation images* $E_{val} = \bigcup_{c=1}^{C} E_{val}^c$ of all object classes

Table 5.3: A sequential learning strategy based on the training steps of the *Viewsphere Model*. For a single object class ($C = 1$) this strategy reduces to the *Viewsphere Model* described in Chapter 4.

object classes. In this work, we term those object parts which are transferred from previously trained object classes to novel object classes the *transferable object parts*. Examples for *transferable object parts* are shown in Figure 5.2. By finding *transferable object parts* across different object classes we reduce the computational complexity of the overall representation. However, similar to the independent learning strategy it is possible to learn a novel object class without retraining all previously learnt object classes from scratch. In the following, we describe a sequential learning strategy of C ($1 \leq c \leq C, c \in \mathbb{N}$) object classes. An overview of this sequential learning strategy is outlined in Figure 5.1 and the corresponding pseudo code is given in Table 5.3.
We assume an initial pool of potential object parts P_S^1 based on all generated *pure examples* E_S^1 (see Section 4.3.2) of the first object class (c=1). However, in contrast to the independent learning strategy, for a novel (subsequent) object class c ($2 \leq c \leq C$) it

might not be necessary to establish a pool of potential object parts based on all *pure examples* E^c_{pure} of the corresponding object class c. Instead, we intend to determine the *transferable object parts* which can be transferred from a previously trained object class to a novel object class c. Based on the *transferable object parts*, we are able to reduce the number of the *pure examples* E^c_{pure} and obtain the *remaining examples* E^c_{remain} ($E^c_{remain} \subseteq E^c_{pure}$) for a novel object class c. For example, if a previously trained object class represents bicycles and a novel object class represents motorbikes it is not necessary to establish a pool of object parts based on all *pure examples* of the motorbike class. In both object classes visually very similar object parts occur (e.g. the wheels). Consequently, those *transferable object parts* can be transferred from the bicycle class to the motorbike class without generating those object parts again from the corresponding *pure examples* of the motorbike class (see Figure 5.2 (left)). In order to determine the *transferable object parts* which can be transferred from a previously trained object class to a novel object class c, we calculate for each object part from the established pool P_S^{c-1} of object parts a joint mutual information MI^{joint} as follows

$$MI^{joint}(c) = \frac{1}{c-1} MI^{max}_{all} + \frac{c-2}{c-1} MI^{max}_{novel} \qquad 2 \leq c \leq C. \tag{5.1}$$

MI^{max}_{all} is the maximal mutual information for an object part on all *pure examples*. Consequently, the positive image set consists of both the *pure examples* of all previously trained object classes and the *pure examples* of a novel object class, i.e., $\bigcup_{i=1}^{c} E^i_{pure}$. MI^{max}_{novel} is the maximal mutual information for an object part on the *pure examples* of a novel class. Consequently, the positive image set consists of the *pure examples* of a novel object class E^c_{pure}. See Equation 4.2 to Equation 4.5 for calculating MI^{max}_{all} and MI^{max}_{novel} of an object part from the pool P_S^{c-1} in conjunction with an optimal detection threshold[1]. The joint mutual information of Equation 5.1 for an object part takes into account with the first term on the right side that with an increasing number of previously trained object classes an object part is less likely to contain information about all object classes simultaneously. However, with an increasing number of previously trained object classes, Equation 5.1 requires with the second term on the right side that an object part must provide at least information about a novel object class for

[1]The negative image sets are chosen to contain real negative training images from the modified PASCAL VOC2006 training data set which does not contain any object instance of the object classes under training.

being a *transferable object part*. Finally, an object part from the established pool P_S^{c-1} with a joint mutual information MI^{joint} (obtained by Equation 5.1) above a predefined information threshold α

$$MI^{joint}(c) \geq \alpha \tag{5.2}$$

is considered as *transferable object part*, which can be transferred from a previously trained object class to a novel object class c ($2 \leq c \leq C$). Based on the determined *transferable object parts*, we are able to reduce the number of the *pure examples* E_{pure}^c for a novel object class c. To this purpose, we determine for each *transferable object part* its visibility in the *pure examples* of a novel object class E_{pure}^c and consequently, we remove those images from the *pure examples* E_{pure}^c of a novel object class c. We rely on the calculated maximal mutual information on all *pure examples* MI_{all}^{max} of Equation 5.1 in conjunction with Equation 4.2 to determine if the maximum detection score of the corresponding linear SVM classifier of a *transferable object part* on a *pure example* is above the optimal detection threshold. We require that at least L[1] *transferable object parts* are visible in a *pure example* to remove this image from all *pure examples* E_{pure}^c. Finally, we obtain the *remaining examples* E_{remain}^c of a novel object class c. Based on the *remaining examples* E_{remain}^c of a novel object class c, a pool $\overline{P}_S^c$ of object parts is established and subsequently, the pools P_S^{c-1} and $\overline{P}_S^c$ are merged into one common pool P_S^c of object parts which forms the basis for further novel object classes.

After repeating the procedure described above for all C defined object classes we obtain a final pool of object parts P_S^C. Based on this final pool P_S^C of object parts, the N_S most informative object parts, which cover all C object classes at once, are selected with an entropy-based measure (see Section 4.3.3.1). Subsequently, for each object class c a dense grid of *Star Models* is established by using a common subset of M_S object parts and the corresponding *pure examples* E_{pure}^c (see Section 4.3.3.2). Finally, one common *Spatial Pyramid Model* for all object classes is learnt by using the *validation images* of all C object classes $E_{val} = \bigcup_{c=1}^{C} E_{val}^c$ and the selected N_S object parts (see Section 4.3.3.3).

Note that Equation 5.1 and Equation 5.2 rely on heuristics and consequently, the sequential learning strategy cannot be generalized to any number of object classes C (see Section 6.2). However, in Section 5.5 we present experimental results which indicate

[1] We choose $L = 3$, since this value performed best in our experiments.

that the sequential learning strategy works well for two and three object classes.

5.4 Detection

In this section, we outline for the described learning strategies of Section 5.3 the necessary detection steps with respect to 2D localization. In addition, we present for the joint and sequential learning strategy a procedure which enables an estimation of the object class label for a predicted object hypothesis.

5.4.1 Independent Learning

The independent learning strategy of Section 5.3.1 results in a set of object class specific *Viewsphere Models*. Consequently, during the detection process the 2D localization procedure described in Section 4.4.1 is applied to a test image for each established *Viewsphere Model*. Since this process can result in multiple overlapping object hypotheses of different object classes, finally, a non-maximum suppression for all generated object hypotheses of all object classes is performed in order to keep only high scoring bounding boxes. Note that the predicted object hypotheses can be provided with an object class label based on the object class specific *Viewsphere Models*.

5.4.2 Joint Learning and Sequential Learning

Both the joint learning strategy of Section 5.3.2 and the sequential learning strategy of Section 5.3.3 establish object class specific grids of *Star Models* and one common *Spatial Pyramid Model*. This results in the following detection procedure for these two learning strategies: during the detection process we rely on the established object class specific grids of *Star Models* in order to identify object hypotheses based on the detection procedure described in Section 4.4.1.1. These initial object hypotheses potentially contain an object instance of the object classes being trained. Subsequently, all these initial object hypotheses are verified by the common *Spatial Pyramid Model* to obtain a final detection score for each object hypothesis (see Section 4.4.1.2). Finally, we perform a non-maximum suppression in order to retain only high scoring object hypotheses.
The joint learning strategy and the sequential learning strategy are not able to provide an object hypothesis with an object class label, since these learning strategies rely on a common *Spatial Pyramid Model* which covers all object classes at once. However,

based on a set of object class specific *Spatial Pyramid Models* it is possible to provide a predicted object hypothesis with an object class label. To this purpose, it is necessary to adapt the selection criterion described in Section 4.3.3.1 for selecting the most informative object parts: for each defined object class c ($1 \leq c \leq C, c \in \mathbb{N}$), we select a subset of N_J (or N_S) object parts from the final pool P_J (or P_S^C) which contains a maximum amount of information about an object class. The *pure examples* of a specific object class serve as positive image set and the *pure examples* of the remaining object classes serve as negative image set. Based on those selected subsets of object class specific parts and the *validation images* of all object classes, for each object class c a *Spatial Pyramid Model* is learnt by relying on the procedure described in Section 4.3.3.3. Finally, a predicted object hypothesis of the joint or sequential learning strategy obtains the object class label from the corresponding *Spatial Pyramid Model* with the highest classification score; see Section 5.5.4 for experimental results.

5.5 Evaluation

In this section, we outline the experimental results we achieve with the three different learning strategies described in Section 5.3. First, we evaluate the performance of these learning strategies with respect to 2D localization and object class estimation on the Multi-Class data set of Section 2.5.3 and second, we outline and evaluate for the sequential and joint learning strategy the pose estimation approach of the *Viewsphere Model* described in Section 4.3.4.

5.5.1 Training Setup

The described learning strategies of Section 5.3 rely on the training steps of the *Viewsphere Model*, where a database of CAD models and a set of real negative images[1] serve as training source. For our experiments we use all 25 car models, all 14 bicycle models (i.e. the 8 basic CAD models and the 6 modified CAD models of Section 4.5.1), and all 13 motorbike models. See Appendix B for a visualization of these CAD models. In order to define a dense grid of viewpoints on the viewsphere, azimuth α is uniformly sampled from 0° to 360° in 5° steps and elevation ϵ is uniformly sampled from 0° to 20° in 5°

[1]The PASCAL VOC2006 training data set is modified such that it does not contain any object instance of the object classes under training.

steps. This viewpoint setup is used to generate the *pure examples* and the *validation images* once for each object class to make sure that the training images for the different learning strategies are identical; details are given in Table 5.4. For a fair comparison of the three different learning strategies with respect to the 2D localization performance two possibilities exist: either we keep the 2D localization performance constant and compare the computational complexity (which is measured by the number of object parts to detect) of the overall representation or we keep the computational complexity constant and compare the 2D localization performance. In our case, we choose to keep the computational complexity for the different learning strategies constant and compare the 2D localization performance. Consequently, the following settings are used for all experiments: $M_I^c = \frac{M_J}{C} = \frac{M_S}{C} = 10$ object parts for modeling a dense grid of *Star Models* and $N_I^c = \frac{N_J}{C} = \frac{N_S}{C} = 25$ object parts for learning the *Spatial Pyramid Model* where C is the number of object classes. We rely on the HOG implementation of (37) with a HOG cell size of 8 pixels and for testing we choose an image pyramid with 10 levels per octave; for the *Spatial Pyramid Model* we choose a spatial pyramid representation with three levels of linear subdivision. In order to generate the pools of object parts for the different learning strategies (see Section 4.3.2 for further details), we choose to establish the object parts from the entire viewsphere, i.e., without a viewsphere subdivision, since the joint learning strategy does not allow assigning viewsphere subspaces of different object classes.

	car (25 CAD models)		**bicycle (8+6 CAD models)**		**mbike (13 CAD models)**	
	pure examples	*validation images*	*pure examples*	*validation images*	*pure examples*	*validation images*
average height of the object instances	56 pixels	64 pixels	56 pixels	109 pixels	56 pixels	63 pixels
no. of defined viewpoints	360	360	360	360	360	360
no. of light variations	1	1	1	1	1	1
no. of all training images	9000	360	5040	360	4680	360

Table 5.4: Details on the generated positive training images for the object classes car, bicycle, and motorbike. The *pure examples* at the predefined training scale have the same fixed height and vary only in width. The object instances within the generated *validation images* vary in width and height. For each defined viewpoint one CAD model is randomly selected to generate a viewpoint-specific training image for the *validation images*. Note that for the object class bicycle we take the 8 basic CAD models and the 6 modified CAD models of Section 4.5.1.

To compare the presented learning strategies with respect to 2D localization we use the evaluation criterion described in Section 2.4.2: an object hypothesis is considered as correct if the overlap between its corresponding bounding box and a ground truth bounding box exceeds 50%. For multiple overlapping object hypotheses, only one object hypothesis is considered as correct and the remaining object hypotheses are penalized as false detections.

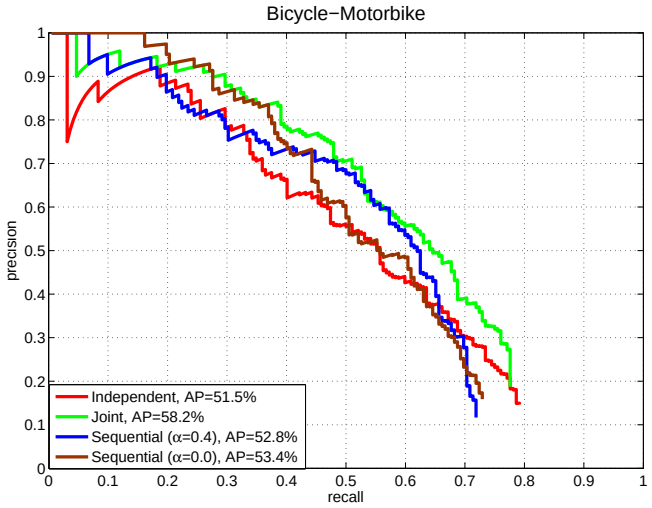

Figure 5.3: Precision-recall curves for the different multi-class learning strategies on the Bicycle-Motorbike test set.

5.5.2 Two Object Classes

We perform the different learning strategies on two visually similar object classes (i.e. on the Bicycle-Motorbike test set) and on two visually dissimilar object classes (i.e. on the Bicycle-Car test set). Figure 5.3 and Figure 5.4 illustrate the corresponding precision-recall curves. We observe for both cases that the joint learning strategy (green curves) outperforms the independent learning strategy (red curves) and the sequential learning strategy (blue and brown curves) due to a higher precision. In order to assess the influence of the *transferable object parts* (see Section 5.3.3), the sequential learning strategy from the bicycle class to the motorbike class and from the bicycle class to the car class is performed for two different information thresholds α (see Equation 5.2). For $\alpha = 0.0$ (brown curves) all object parts from the previously trained bicycle class are

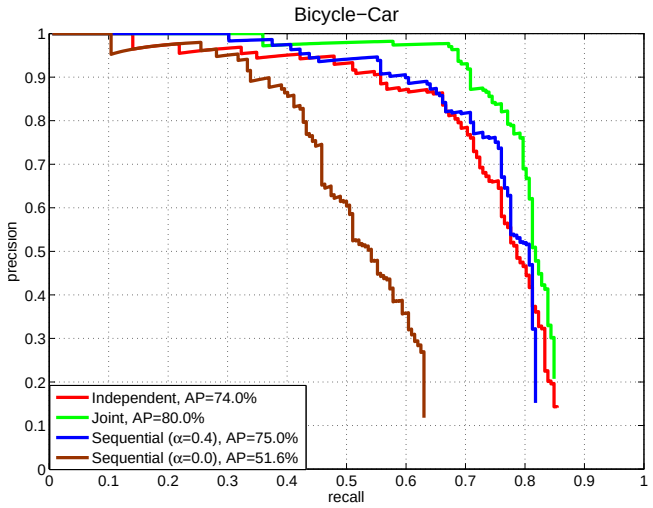

Figure 5.4: Precision-recall curves for the different multi-class learning strategies on the Bicycle-Car test set.

considered as *transferable object parts* with the result that for both cases (bicycle to car and bicycle to motorbike) the set of the *remaining examples* E_{remain} for the novel object class is an empty set and consequently, no further object parts for the novel object class (i.e. motorbike or car) are generated. As a result, for visually similar object classes (i.e. from bicycle to motorbike) the detection result for the sequential learning strategy (brown curve with 53.4%) is still on par with the independent learning strategy (red curve with 51.5%) and worse than the joint learning strategy (green curve with 58.2%). For visually dissimilar object classes (i.e. from bicycle to car) the detection result for the sequential learning strategy (brown curve with 51.6%) is worse than both the independent learning strategy (red curve with 74.0%) and the joint learning strategy (green curve with 80.0%), since no further object parts for the visually dissimilar object class are generated. With an increased information threshold of $\alpha = 0.4$ (blue curves) the situation is different. For visually dissimilar object classes (i.e. from bicycle to car) none of the bicycle object parts is considered as *transferable object part* which results in a non-empty set for the *remaining examples* E_{remain} and consequently, additional object parts for the object class car are generated. The increased detection performance (blue curve with 75.0%) is now on par with the independent learning strategy (red curve with 74.0%). For visually similar object classes (i.e. from bicycle to motorbike)

three of the bicycle object parts are still considered as *transferable object parts*. This results in a non-empty set for the *remaining examples* E_{remain}, additionally generated object parts for the object class motorbike, and a detection result (blue curve with 52.8%) which is on par with the result of the independent learning strategy (red curve with 51.5%). These results indicate that in both cases (i.e. visually similar and visually dissimilar object classes) an information threshold of $\alpha = 0.4$ for the sequential learning strategy achieves a good trade-off between transferring knowledge (i.e. object parts) from previously trained object classes to novel object classes and generating additional knowledge (i.e. object parts) from a novel object class. Then, the sequential learning strategy which combines the advantages of the joint and the independent learning strategy results in a detection performance which is on par with or better than the detection performance of the independent and the joint learning strategy. Therefore, for the subsequent tests the information threshold for the sequential learning strategy is set to $\alpha = 0.4$. Examples for *transferable object parts* on both data sets are shown in Figure 5.2.

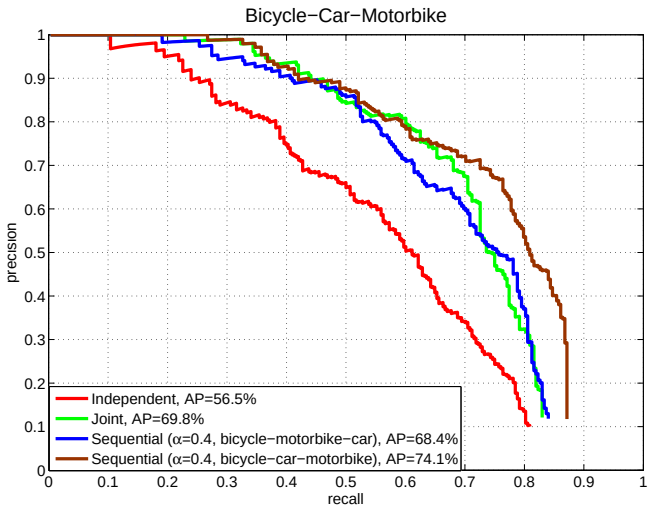

Figure 5.5: Precision-recall curves for the different multi-class learning strategies on the Bicycle-Car-Motorbike test set.

5.5.3 Three Object Classes

The precision-recall curves on the Bicycle-Car-Motorbike test set are shown in Figure 5.5. In this case, the joint learning strategy (green curve with 69.8%) clearly outperforms the independent learning strategy (red curve with 56.5%) due to a higher precision. We observe that the order in which the object classes are learnt during the sequential learning strategy affects the detection performance. The detection results indicate that it might be advantageous to learn visually dissimilar object classes at first. However, both detection results (blue curve with 68.4% and brown curve with 74.1%) of the sequential learning strategy significantly outperform the independent learning strategy and perform on par with or even better than the joint learning strategy. In addition, with the sequential learning strategy it is possible to learn a novel object class without retraining the previously trained object classes from scratch (in contrast to the joint learning strategy).

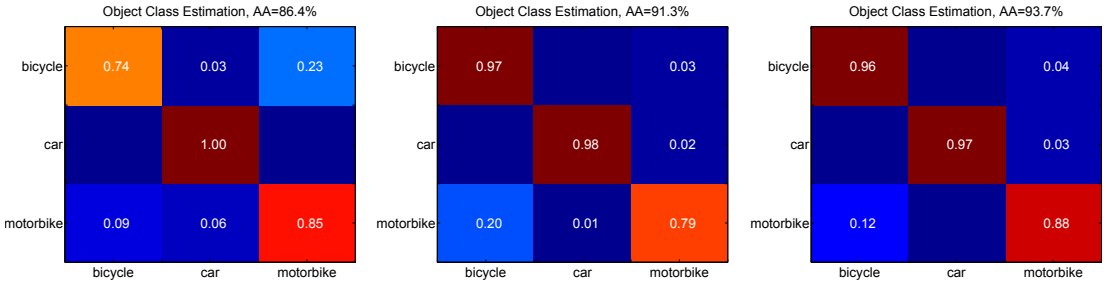

Figure 5.6: Comparison of the independent learning strategy (left), the joint learning strategy (center), and the sequential learning strategy (right) with respect to object class estimation showing confusion matrices (rows: ground truth, columns: estimates) for the Bicycle-Car-Motorbike test set.

5.5.4 Object Class and Pose Estimation

As described in Section 5.4, the three learning strategies are able to provide a predicted object hypothesis with an object class label. The confusion matrices obtained by classifying all positive detections of Section 5.5.3 on the Bicycle-Car-Motorbike test set are shown in Figure 5.6 (left) for the independent learning strategy, in Figure 5.6 (center) for the joint learning strategy, and in Figure 5.6 (right) for the sequential learning strategy (learning order: bicycle-car-motorbike). For all three cases, confusion is more pronounced between the bicycle class and the motorbike class. The joint learning

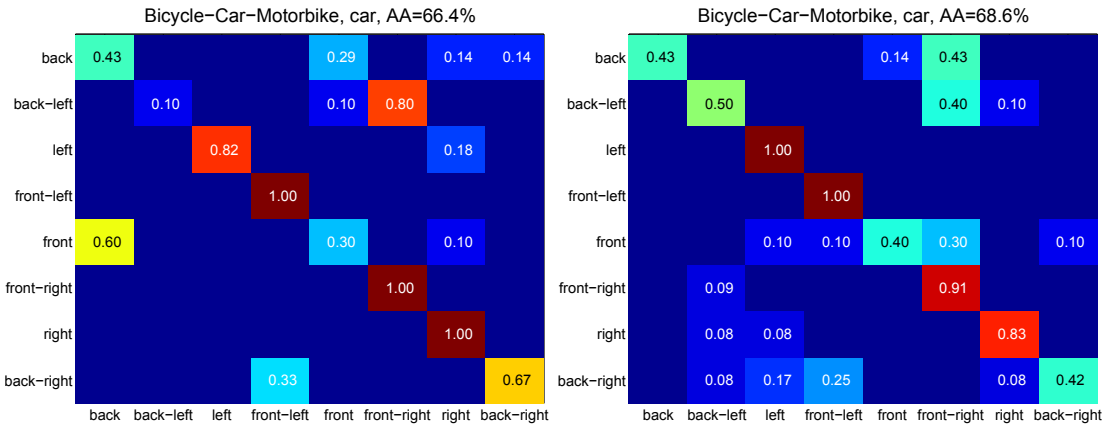

Figure 5.7: Comparison of the joint learning strategy (left) with the sequential learning strategy (right) with respect to pose estimation showing confusion matrices (rows: ground truth, columns: estimates) on the Bicycle-Car-Motorbike test set for the object class car.

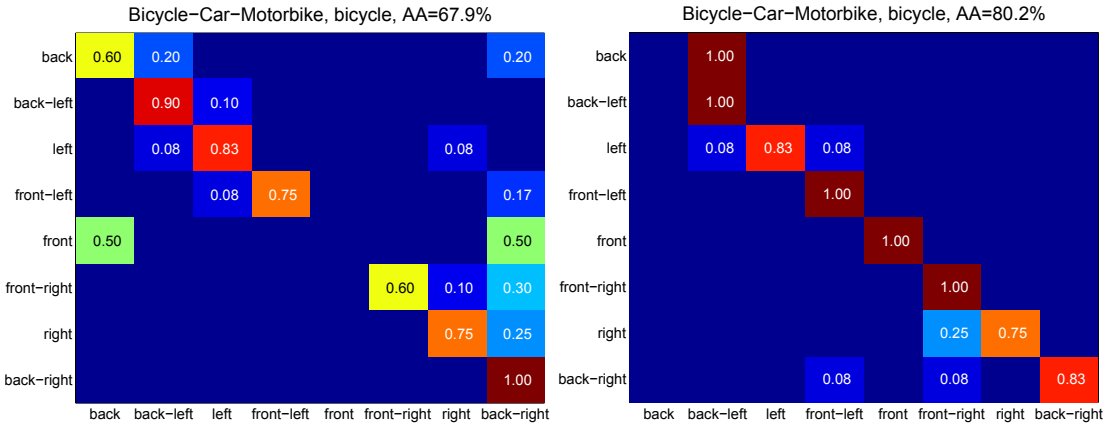

Figure 5.8: Comparison of the joint learning strategy (left) with the sequential learning strategy (right) with respect to pose estimation showing confusion matrices (rows: ground truth, columns: estimates) on the Bicycle-Car-Motorbike test set for the object class bicycle.

strategy with an average classification accuracy of 91.3% and the sequential learning strategy with an average classification accuracy of 93.7% achieve a better performance with respect to object class estimation than the independent learning strategy with an average classification accuracy of 86.4%, due to the established sets of discriminatively trained object class specific *Spatial Pyramid Models* described in Section 5.4.2.

An advantage of a part-based representation for multiple object classes resides in the spatial co-occurrence of object parts which can be used for the pose estimation step of

the *Viewsphere Model* described in Section 4.3.4. The following experiment shows that this advantage is retained even when the object parts are shared over several object classes, as done for the joint learning strategy and the sequential learning strategy. To this purpose, for each object class being trained we rely on the eight subspaces given in Figure 4.5 and for each defined subspace we draw a new subset of N_S (or N_J) object parts from the final pool P_S^C (or P_J) of object parts for training the pose estimation step of the *Viewsphere Model*. The confusion matrices obtained by classifying all positive detections of Section 5.5.3 on the Bicycle-Car-Motorbike test set are shown in Figure 5.7 for the sequential learning strategy (right) and the joint learning strategy (left) for the object class car and in Figure 5.8 for the sequential learning strategy (right) and the joint learning strategy (left) for the object class bicycle[1]. For the joint learning strategy we observe that for the object class car confusion is more pronounced in opposing views and for the object class bicycle confusion is more pronounced in neighboring views. However, the pose estimation of the joint learning strategy relies on the final pool P_J which consists of common object parts of all object classes (i.e. bicycle, car, and motorbike) with a reduced viewpoint discriminativity. For the sequential learning (with the learning order bicycle-car-motorbike) the situation is different. For the object class bicycle we obtain a pose estimation result which is comparable to the result of the *Viewsphere Model* (see Section 4.5.2.3) and for the object class car we obtain a confusion matrix which is similar to the confusion matrix of the joint learning strategy. The reason for this behavior is that the pose estimation of the sequential learning is based on the final pool P_S^C of object parts which mainly consists of an initial pool P_S^1 of bicycle object parts and a few additionally generated object parts for the object classes car and motorbike. While the initially generated object parts for the bicycle class are sufficient to discriminate between the viewpoints of the bicycle class, the few additionally generated object parts for the object class car seem to be not sufficient to discriminate between the viewpoints of the object class car. Some successful results of the full detection process with 2D localization, object class, and pose estimation on the Bicycle-Car-Motorbike test set are shown in Figure 5.9 for the sequential learning strategy and in Figure 5.10 for the joint learning strategy. Figure 5.11 (top) depicts some

[1]Note that for the object class motorbike it is not possible to establish a confusion matrix, since no ground truth with respect to the object pose for the motorbike class is provided within the Bicycle-Car-Motorbike test set.

failed detection results of the sequential learning strategy and Figure 5.11 (bottom) depicts some failed detection results of the joint learning strategy. The failed detections on the Bicycle-Car-Motorbike test set are mainly caused by sub detections within an object instance or when there is insufficient evidence in the image for a correct pose initialization.

5.6 Summary

In this chapter, we have presented three learning methods with different part sharing strategies: an independent learning strategy which trains each object class independently from all other object classes and does not share object parts among the object classes, a joint learning strategy which trains all object classes simultaneously and shares object parts among the object classes, and a sequential learning strategy which learns one object class after another and transfers object parts from previously trained object classes to novel object classes. All these learning strategies rely on the training steps of the proposed *Viewsphere Model* where a database of CAD models and real negative images serve as training source. Our experiments indicate that the sequential learning achieves the best result with respect to flexibility during the training process and recognition performance and thus could be suitable for learning multiple object classes on a densely sampled viewsphere on a larger scale.

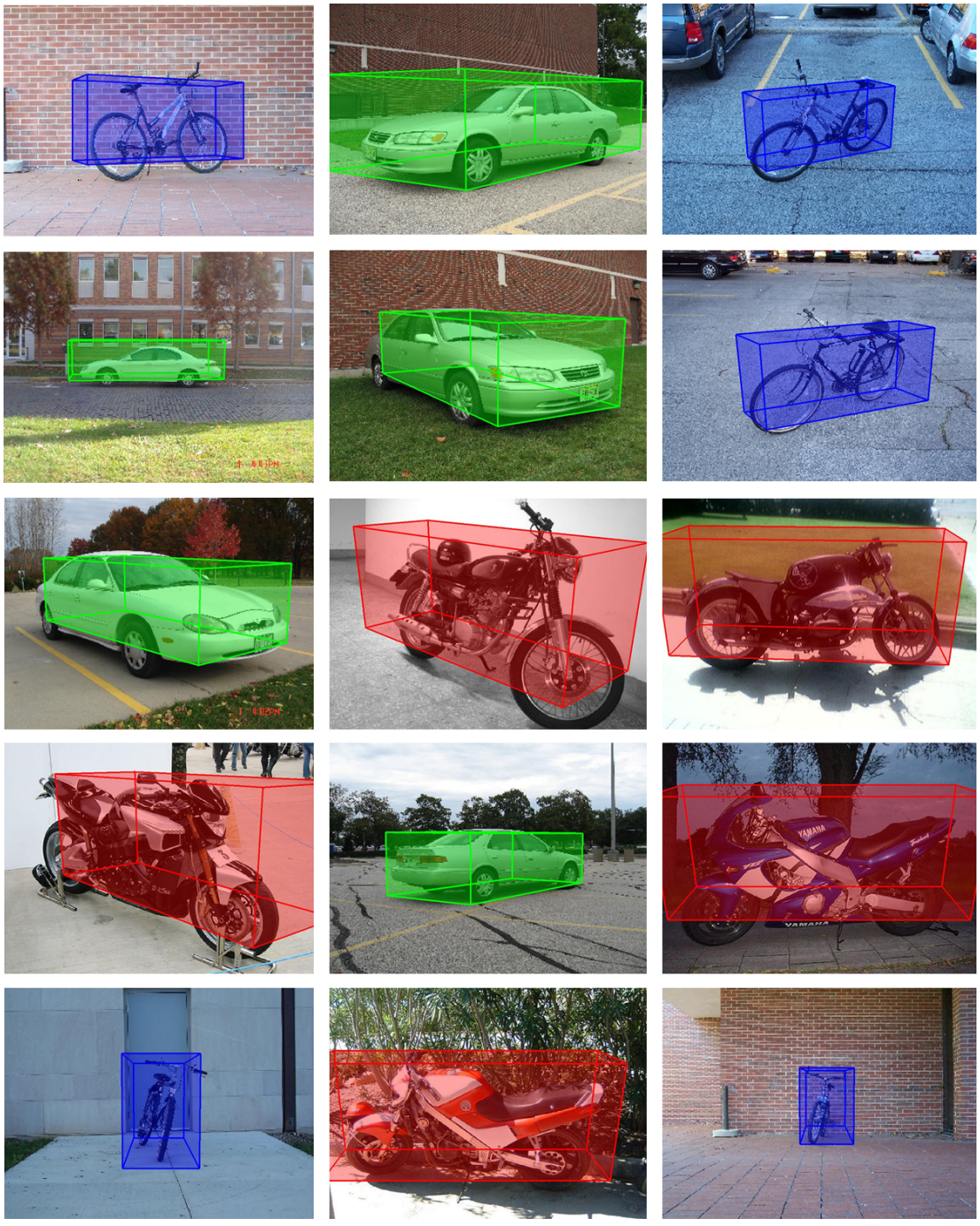

Figure 5.9: Some successful detection results of the full detection process consisting of 2D localization, object class, and pose estimation on the Bicycle-Car-Motorbike test set for the sequential learning strategy. The estimated object class and object pose are indicated with a colored 3D bounding box (car: green, bicycle: blue, motorbike: red). We show only the highest scored detection per image.

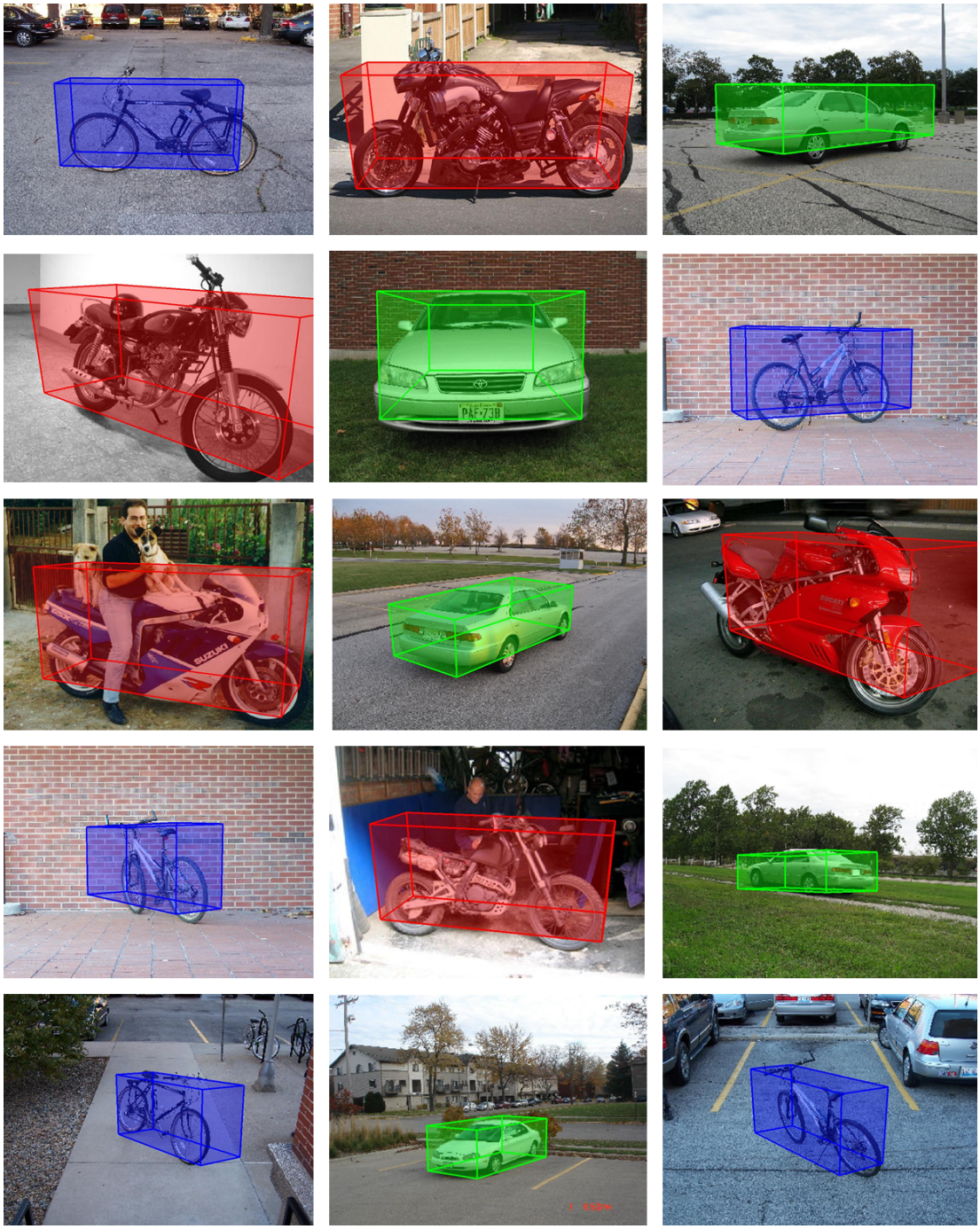

Figure 5.10: Some successful detection results of the full detection process consisting of 2D localization, object class, and pose estimation on the Bicycle-Car-Motorbike test set for the joint learning strategy. The estimated object class and object pose are indicated with a colored 3D bounding box (car: green, bicycle: blue, motorbike: red). We show only the highest scored detection per image.

Figure 5.11: Some failed detection results on the Bicycle-Car-Motorbike test set for the sequential learning strategy (top) and for the joint learning strategy (bottom). The failed detections are mainly caused by sub detections within an object instance or when there is insufficient evidence in the image for a correct pose initialization.

Chapter 6

Conclusion

In this chapter, we summarize the work presented in this thesis and provide an outlook on possible research directions in the area of object class detection.

6.1 Summary

In this thesis, we have presented three different, but related approaches to object class detection which learn a part-based object class representation from a database of CAD models and a set of real negative training images.
First, we have proposed the *Multi-View Model*, an approach which is able to detect object instances from multiple viewpoints. One advantage of the *Multi-View Model* is that it integrates the generative *Star Model* with the discriminative *Spatial Pyramid Model* (i.e. two different learning paradigms) into one common object class detection framework. In addition, the *Multi-View Model* derives its positive training examples exclusively from a database of pre-built CAD models and its negative training examples from an arbitrary set of background images. Consequently, during training the *Multi-View Model* does not require any viewpoint or bounding box annotations. Furthermore, it relies on an unsupervised approach to determine suitable object part positions within the positive training examples for a given viewpoint such that during training a set of viewpoint-specific part detectors can be established without any manual intervention. Based on this set of part detectors several viewpoint-specific *Star Models* and one common *Spatial Pyramid Model* are established. The *Spatial Pyramid Model* combines all viewpoint-specific part detectors into one spatial pyramid representation to exploit all

the information contained in the set of part detectors and to make use of viewpoint symmetries and part similarities within the object class being trained. During detection, the *Multi-View Model* relies on a pre-detection step where the viewpoint-specific *Star Models* identify regions of interest which potentially contain an object instance of the object class being tested. Subsequently, these initial object hypotheses are verified by the *Spatial Pyramid Model* in order to obtain a final detection score for each object hypothesis. In addition to the final detection score, each object hypothesis is also provided with an approximate viewpoint label based on the viewpoint-specific *Star Models* of the pre-detection step. When applying the *Multi-View Model* on different benchmark data sets it performs on par with other reported object class detection approaches with respect to 2D localization. However, there is a performance gap to the current state-of-the-art detection method of (34). In addition, the performance of the *Multi-View Model* with respect to pose estimation is not comparable with other reported results, due to the unsupervised object part generation which does not take into account the inter-viewpoint discriminativity.

In order to obtain 2D localization and pose estimation results which are comparable to the current state-of-the-art results we have introduced the *Viewsphere Model*, our second part-based approach to object class detection. The *Viewsphere Model* retains the advantage of the *Multi-View Model* by combining the generative *Star Model* and the discriminative *Spatial Pyramid Model* into one common object class detection framework and also derives its positive training examples exclusively from a database of CAD models. However, the *Viewsphere Model* takes into account the shortcomings of the *Multi-View Model*, notably a viewpoint-specific generation of object parts resulting in redundant or non-informative object parts and a poor pose estimation due to a lack of inter-viewpoint discriminativity. The *Viewsphere Model* addresses these shortcomings and establishes an object class representation which densely covers the entire viewsphere and efficiently exploits co-occurrences of object parts within the object class being trained due to viewpoint symmetries and part similarities. To this purpose, the object class being trained is decomposed by using affinity propagation (48) in conjunction with the HOG descriptor of (21) resulting in a pool of potential object parts. Subsequently, we rely on an information-theoretic selection criterion (123) in order to obtain a subset of object parts containing a maximum of information about the object class with respect to object class detection. Based on this selected subset of object parts,

a dense grid of *Star Models* is established providing initial object hypotheses and one common *Spatial Pyramid Model* is learnt to verify these initial object hypotheses. For the pose estimation step of the *Viewsphere Model*, we first divide the viewsphere into a set of suitable subspaces and second, for each defined subspace we draw a new subset of object parts from the pool of potential object parts and learn a *Spatial Pyramid Model*. During detection, an object hypothesis is provided with the corresponding viewpoint label of the *Spatial Pyramid Model* with the highest classification score. In addition, the pose estimation within an estimated subspace can be further refined by using Gaussian mixture models (13) which model the spatial arrangement of the corresponding object parts. The *Viewsphere Model*, which densely covers the entire viewsphere by sharing object parts over several defined viewpoints on the viewsphere, achieves detection and pose estimation results which are superior to those obtained by the *Multi-View Model* and on par with the current state-of-the-art results.

Finally, we have extended the *Viewsphere Model* for a single object class to cope with multiple object classes simultaneously. To this purpose, we have presented three learning methods with different object part sharing strategies relying on the training steps of the *Viewsphere Model*: an independent learning strategy which learns each object class independently from all other object classes, a joint learning strategy which learns all object classes simultaneously, and a sequential learning strategy which learns one object class after another. An independent learning strategy has the advantage that a novel object class can easily be added to an already existing multi-class representation (42). However, the complexity of this representation grows linearly with the number of object classes (117). The properties of a joint learning are opposed to the properties of an independent learning strategy. Adding a novel object class to an existing multi-class representation is not possible without retraining all previously trained object classes from scratch. However, a joint learning strategy normally reduces the computational complexity of a multi-class representation by finding common object parts across several object classes (117). A sequential learning strategy now combines the advantages of a joint and an independent learning strategy. When learning one object class after another it is possible to add a novel object class to an existing multi-class representation without retraining the previously learnt object classes from scratch. In addition, it is possible to reduce the overall complexity of the multi-class representation by transferring knowledge (30) from previously trained object classes to a novel object class.

In this work, the knowledge transfer of the described sequential learning strategy relies on an entropy based measure (123). We determine which training images of a novel object class are already covered by the knowledge (i.e. object parts) of a previously trained object class and thus can be removed from the training set of the novel object class. Our experiments show that the sequential learning strategy achieves the best result with respect to flexibility during the training process and detection performance and thus could be used for training multiple object classes on a larger scale.

6.2 Future Work

This thesis presents part-based approaches to object class detection which rely on a database of CAD models and a set of real negative images. Based on the described methods and concepts, the following directions for future work are possible:

- **Sequential Learning on a Larger Scale:** In Chapter 5 we have presented three different learning strategies to learn multiple object classes on a densely sampled viewsphere. Considering the flexibility of the training process and the detection performance the sequential learning strategy achieves the most promising results. However, in our experiments we use only three object classes and thus, the scalability and the detection performance of the described sequential learning strategy have to be investigated for more object classes. Also the proposed concept for knowledge transfer could be revisited to improve the pose estimation performance by also considering the viewpoint discriminativity of the *transferable object parts*.

- **Geometric and Scene Context:** All the described object class detection approaches of this thesis consider a predicted object hypothesis in isolation and do not take into account geometric or scene context, although contextual information plays an important role for object detection (116) and several successful approaches to this topic have been proposed such as (7, 64, 101). For example, the estimated pose of an object instance could be used as geometric prior and the estimated object class label of an object hypothesis could be used as scene prior for other object hypotheses. Consequently, an extension of the approaches presented in this thesis should make use of this context information to further improve their detection performance.

- **Extension to Nonrigid Object Classes:** The evaluation of the object class detection approaches presented in this thesis is restricted to predominantly rigid object classes. An extension of the proposed methods to nonrigid object classes has to be investigated. As shown in (97) for people detection, the use of CAD models as training source could be an advantage, since it is possible to generate an arbitrary amount of training images which better represent the shape and pose variations within the nonrigid object class being detected.

Appendix A

Publications

Parts of the work presented in this thesis have been published in the following conference papers and technical reports:

- Johannes Schels, Joerg Liebelt, and Rainer Lienhart. Learning an object class representation on a continuous viewsphere. In *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, 2012.
- Johannes Schels, Joerg Liebelt, and Rainer Lienhart. Self-calibrating 3D context for retrieving people with luggage. In *IEEE International Conference on Computer Vision Workshops*, Barcelona, 2011.
- Johannes Schels, Joerg Liebelt, Klaus Schertler, and Rainer Lienhart. Synthetically trained multi-view object class and viewpoint detection for advanced image retrieval. In *ACM International Conference on Multimedia Retrieval*, Trento, 2011.
- Johannes Schels, Joerg Liebelt, Klaus Schertler, and Rainer Lienhart. Building a semantic part-based object class detector from synthetic 3D object models. In *IEEE International Conference on Multimedia and Expo*, Barcelona, 2011.
- Johannes Schels, Joerg Liebelt, and Rainer Lienhart. Learning to represent multiple object classes on a continuous viewsphere. Technical report, University of Augsburg, 2012.

Appendix B

3D Model Database

In the following, all CAD models for the three different object classes bicycle, car, and motorbike are visualized. All CAD models stem from commercial distributors, notably turbosquid.com and doschdesign.com. In addition, we provide for each CAD model a statistic which contains the number of vertices, the number of polygons, the number of materials, and the number of textures. See Section 2.3.1 for details.

Figure B.1: All CAD models to represent the object class bicycle.

Figure B.2: All CAD models to represent the object class car (1/2).

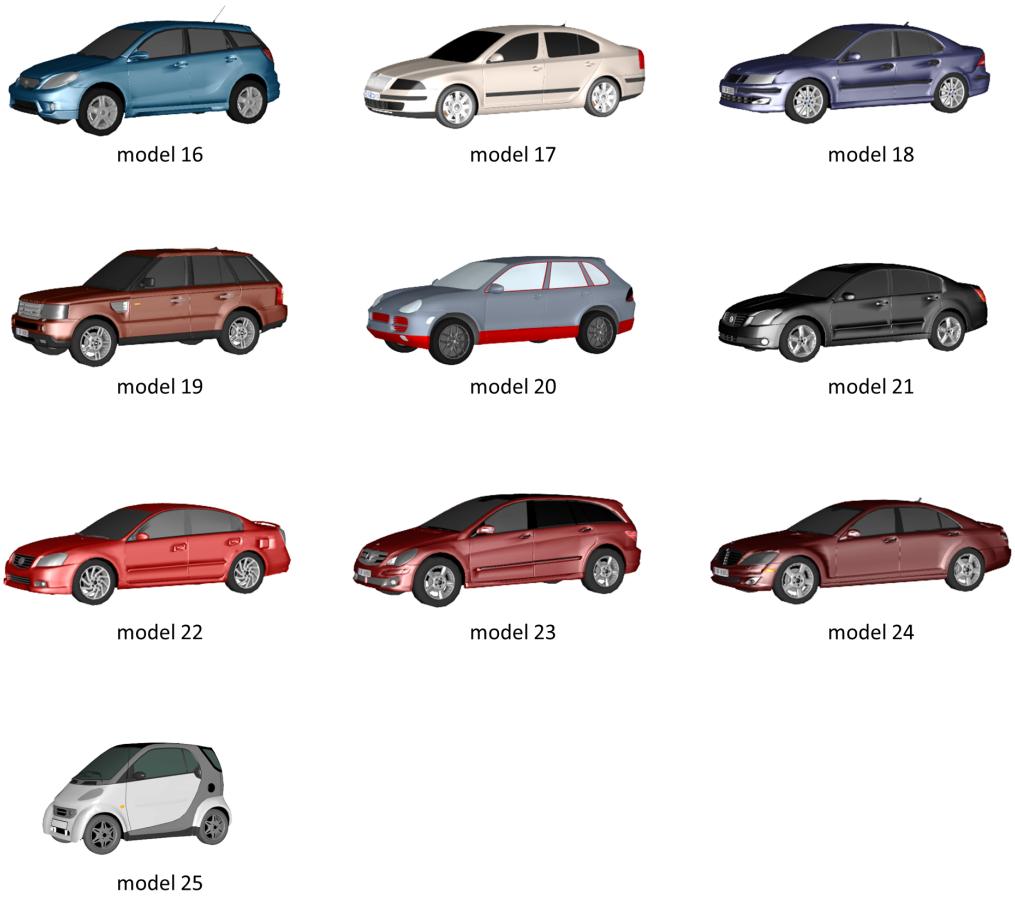

Figure B.3: All CAD models to represent the object class car (2/2).

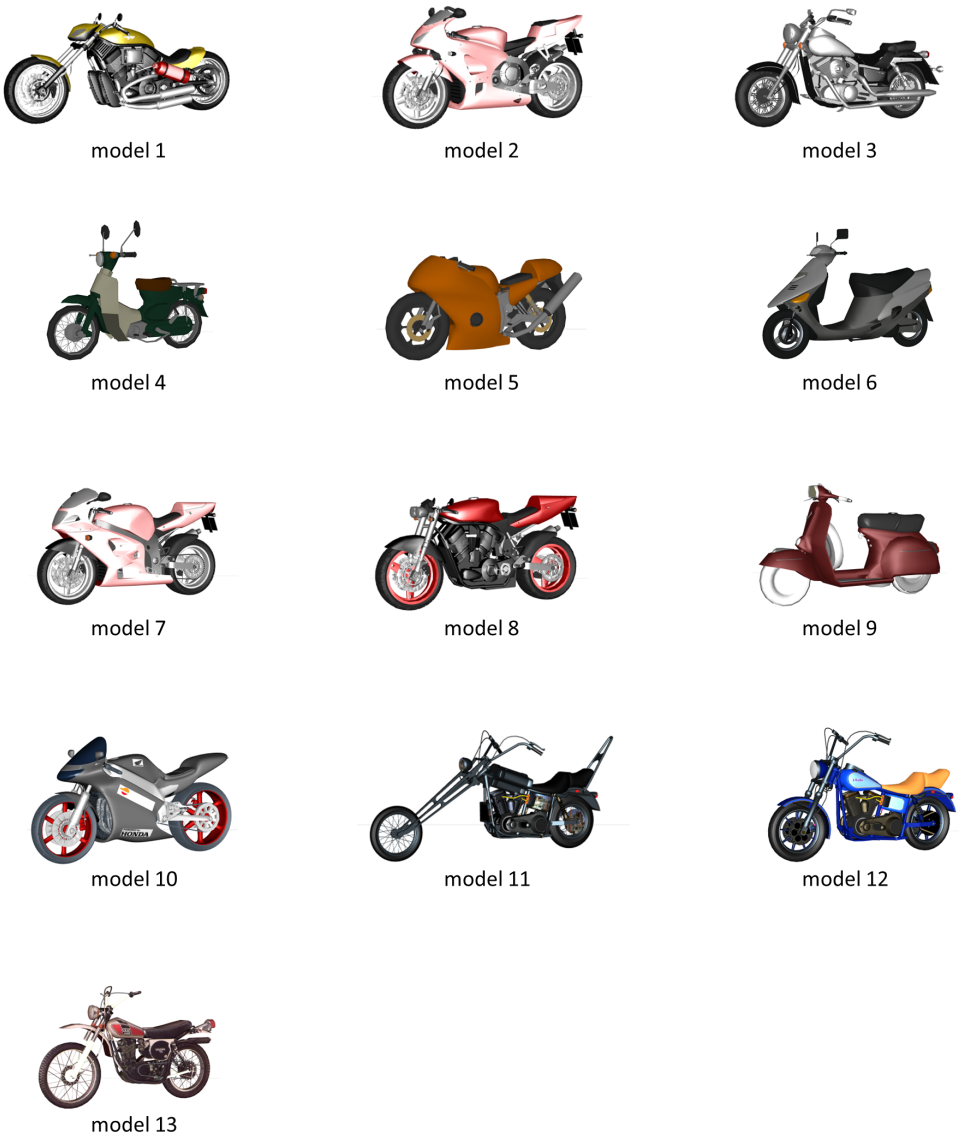

Figure B.4: All CAD models to represent the object class motorbike.

	no. of vertices	no. of polygons	no. of materials	no. of textures
model 1	55442	101943	19	17
model 2	63166	116397	11	6
model 3	50507	94954	18	20
model 4	46824	58360	14	14
model 5	27142	26164	19	19
model 6	32445	53272	6	6
model 7	49565	62468	20	7
model 8	51256	62228	9	8
mean	47034	73223	15	12

Table B.1: Statistics of all CAD models used to represent the object class bicycle.

	no. of vertices	no. of polygons	no. of materials	no. of textures
model 1	67145	44702	13	6
model 2	45132	30158	11	5
model 3	80608	86841	11	6
model 4	3438	3051	8	0
model 5	6529	6556	5	0
model 6	10535	11431	12	0
model 7	58995	43167	12	6
model 8	42061	26954	12	4
model 9	4178	6819	14	3
model 10	118755	212177	26	3
model 11	30524	54958	19	8
model 12	25600	45801	27	22
model 13	29657	52808	4	1
mean	40243	48109	13	5

Table B.2: Statistics of all CAD models used to represent the object class motorbike.

	no. of vertices	no. of polygons	no. of materials	no. of textures
model 1	62542	118071	13	2
model 2	20124	36232	16	0
model 3	87416	170156	12	2
model 4	42344	78358	26	5
model 5	23032	37776	16	0
model 6	8739	11795	14	0
model 7	90506	171326	15	2
model 8	34050	63170	25	8
model 9	35897	66482	20	4
model 10	9686	14553	13	0
model 11	32952	60820	24	6
model 12	9004	13790	19	0
model 13	27285	50704	23	9
model 14	32497	59998	23	3
model 15	30136	55882	23	5
model 16	30587	56430	22	1
model 17	304969	585568	123	17
model 18	35257	65630	26	6
model 19	40051	74378	28	4
model 20	116997	225295	15	3
model 21	33111	61768	22	2
model 22	37070	69516	21	1
model 23	40541	74890	30	6
model 24	36873	67818	30	8
model 25	18790	31663	18	0
mean	40243	92882	23	4

Table B.3: Statistics of all CAD models used to represent the object class car.

Appendix C

Multi-Class Data Set

In the following, the 96 images from the VOC2006 motorbike test set which are used for the Multi-Class data set are visualized. See Section 2.5.3 for further details.

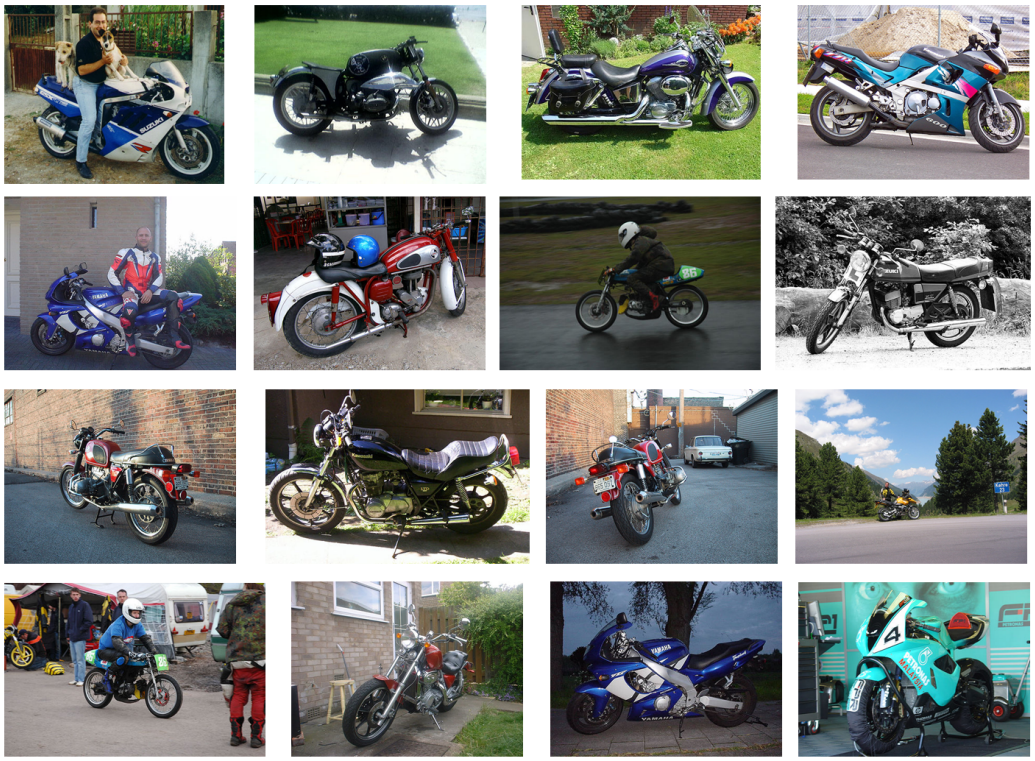

Figure C.1: All images from the VOC2006 motorbike test set which are used for the Multi-Class data set (1/4).

Figure C.2: All images from the VOC2006 motorbike test set which are used for the Multi-Class data set (2/4).

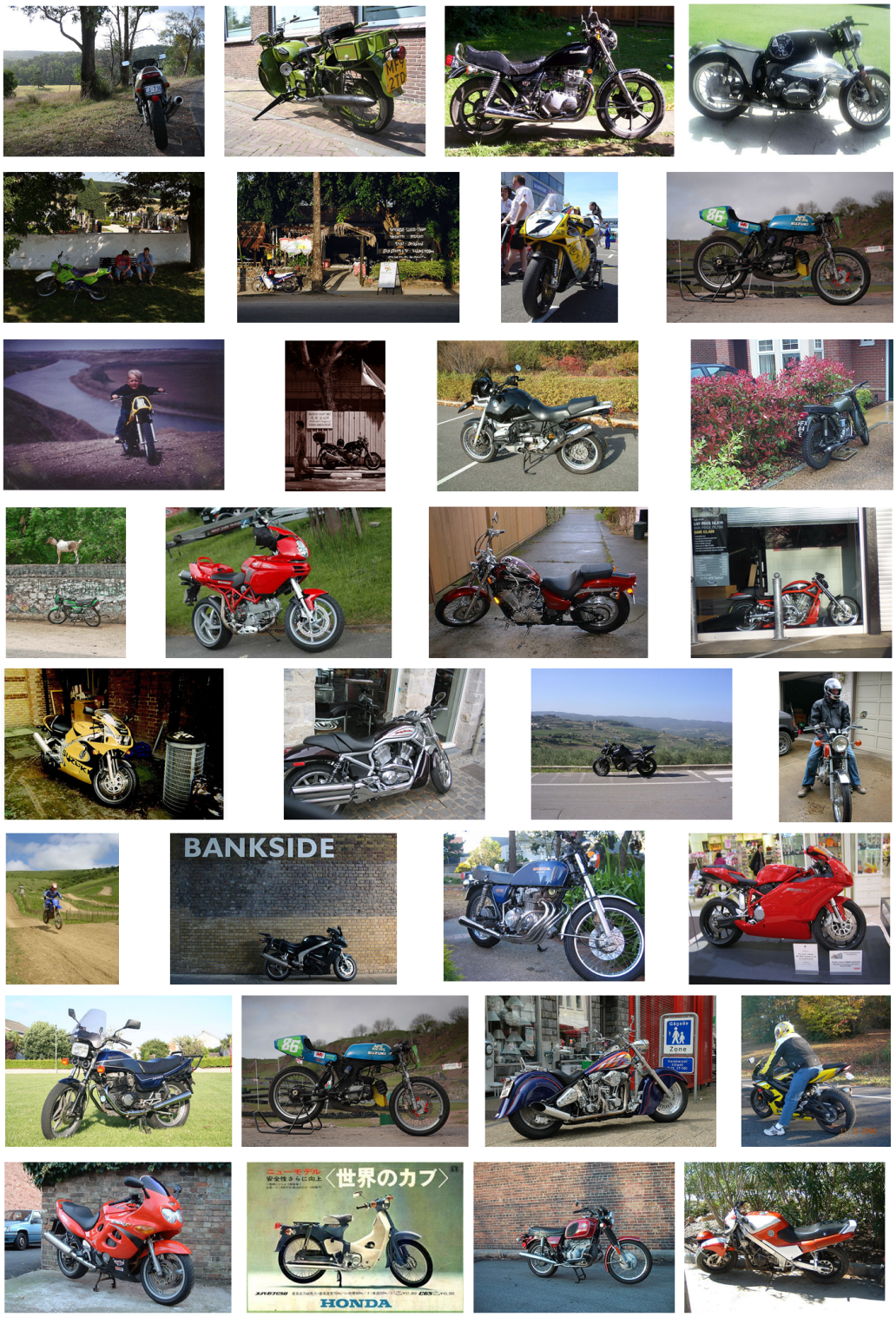

Figure C.3: All images from the VOC2006 motorbike test set which are used for the Multi-Class data set (3/4).

Figure C.4: All images from the VOC2006 motorbike test set which are used for the Multi-Class data set (4/4).

Appendix D

Star Model with C Components per Object Part

Felzenszwalb and Huttenlocher describe in (36) a dynamic programming approach in order to minimize Equation 2.2 for a star-structured or tree-structured model. In the following, we briefly summarize the principle of this dynamic programming approach and describe the adaption of this method with respect to the detection procedure of the *Star Model* described in Section 2.2.1.2.

As outlined in (36), for any object part p_j with no children[1], the best location for the object part p_j can be computed as a function of the location of its own fixed reference part p_i. The only edge incident on p_j is (p_i, p_j), thus the only contribution of l_j to the energy of Equation 2.2 is $a_j(l_j) + s_{ij}(l_i, l_j)$. The quality of the best location for the object part p_j given a location l_i for its own fixed reference part p_i is

$$B_j(l_i) = \min_{l_j} \left(a_j(l_j) + s_{ij}(l_i, l_j) \right). \tag{D.1}$$

For any object part p_j with T children, we assume that the function $B_t(l_j)$ is known for each child p_t. Then the quality of the best location for the object part p_j given a location l_i for its own fixed reference part p_i is

$$B_j(l_i) = \min_{l_j} \left(a_j(l_j) + s_{ij}(l_i, l_j) + \sum_{t=1}^{T} B_t(l_j) \right). \tag{D.2}$$

[1] A child refers any leaf of a tree-structured model or any object part (excepting the reference part) of a star-structured model.

Finally, for the root or reference part p_r with T children, we assume that the function $B_t(l_r)$ is known for each child p_t. Then the quality for the root or reference part p_r is

$$B_r(l_r) = a_r(l_r) + \sum_{t=1}^{T} B_t(l_r). \tag{D.3}$$

Consequently, the minimization of Equation 2.2 can be recursively expressed in terms of the function $B_j(l_i)$ (see Equation D.1). The function $B_j(l_i)$ can be efficiently computed by using the generalized distance transform (35) when s_{ij} is restricted to a Mahalanobis distance.

The *Star Model* described in Section 2.2.1 consists of N object parts p_n and one reference part p_r. The reference part p_r represents the center of the training images and in contrast to the object parts p_n, the appearance of the reference part p_r is not represented by a linear SVM classifier, i.e., $a_r(l_r) = 0$. Each object part p_n is connected to the reference part p_r by its own Gaussian mixture model $\theta = \{\alpha_c, \mu_c, \Sigma_c\}$ with C components (see Equation 2.1). As described in Section 2.2.1.2, for the inverted part detector response of an object part p_n we perform a distance transform for each component c of the related Gaussian mixture model. For an object part with C components, the transformed responses of all components are ranked with corresponding prior probabilities $prob_c$ and added to compute the quality for the reference part p_r

$$\begin{aligned} B_r(l_r) &= a_r(l_r) + \sum_{c=1}^{C} prob_c B_c(l_r) \\ &= \sum_{c=1}^{C} prob_c B_c(l_r) \end{aligned} \tag{D.4}$$

with

$$prob_c = \frac{\alpha_c}{2\pi |\Sigma_c|^{\frac{1}{2}}}. \tag{D.5}$$

By using Equation D.1 the term $prob_c B_c(l_r)$ in Equation D.4 can be reformulated as

follows

$$
\begin{aligned}
prob_c B_c(l_r) &= prob_c \min_{l_n} \left(a_n(l_n) + s^c_{rn}(l_r, l_n) \right) \\
&= \min_{l_n} \left(prob_c a_n(l_n) + prob_c s^c_{rn}(l_r, l_n) \right) \\
&= \min_{l_n} \left(\bar{a}^c_n(l_n) + \bar{s}^c_{rn}(l_r, l_n) \right) \\
&= \bar{B}_c(l_r)
\end{aligned}
\tag{D.6}
$$

where $a_n(l_n)$ denotes the inverse part detector response of an object part p_n. $s^c_{rn}(l_r, l_n)$ denotes the Mahanalobis distance of an object part p_n to the reference part p_r which is defined by the mean vector μ_c and the covariance matrix Σ_c of a component c. $\bar{a}^c_n(l_n)$ denotes the inverse part detector response of an object part p_n and $\bar{s}^c_{rn}(l_r, l_n)$ denotes the Mahanalobis distance of an object part p_n. Both $\bar{a}^c_n(l_n)$ and $\bar{s}^c_{rn}(l_r, l_n)$ are specialized for a component c of the related Gaussian mixture model. By combining Equation D.4 and Equation D.6 we obtain

$$
B_r(l_r) = \sum_{c=1}^{C} \bar{B}_c(l_r) \tag{D.7}
$$

which is equivalent to the dynamic programming approach of (36) for C object parts in a star-structured model with flat hierarchy (see Equation D.3). The above described procedure can be repeated for each of the N object parts of the *Star Model* with C components per object part, resulting in a total of NC separate object parts represented by a flat hierarchy.

List of Figures

LIST OF FIGURES

LIST OF FIGURES

List of Tables

Bibliography

[1] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**:1475–1490, 2004.

[2] Nicolas Allezard, Michel Dhome, and Frederic Jurie. Recognition of 3D textured objects by mixing view-based and model-based representations. In *IEEE International Conference on Pattern Recognition*, pages 960–963, Istanbul, 2000.

[3] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Monocular 3D pose estimation and tracking by detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 623–630, San Francisco, 2010.

[4] Mica Arie-Nachimson and Ronen Basri. Constructing implicit 3D shape models for pose estimation. In *IEEE International Conference on Computer Vision*, pages 1341–1348, Kyoto, 2009.

[5] Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *IEEE International Conference on Computer Vision*, pages 2252–2259, Barcelona, 2011.

[6] Hossein Azizpour and Ivan Laptev. Object detection using strongly-supervised deformable part models. In *European Conference on Computer Vision*, pages 836–849, Florence, 2012.

[7] Sid Ying-Ze Bao, Min Sun, and Silvio Savarese. Toward coherent object detection and scene layout understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72, San Francisco, 2010.

[8] Aharon Bar-Hillel and Daphna Weinshall. Efficient learning of relational object class models. *International Journal of Computer Vision*, **77**:175–198, 2008.

[9] Evgeniy Bart and Shimon Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 672–679, San Diego, 2005.

[10] Irving Biederman. *An Invitation to Cognitive Science*, chapter Visual Object Recognition, pages 121–165. MIT Press, 1995.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, New York, 2006.

[12] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Conference on Computer Graphics and Interactive Techniques*, pages 192–198, San Jose, 1977.

[13] Charles A. Bouman. Cluster: An unsupervised algorithm for modeling gaussian mixtures. https://engineering.purdue.edu/~bouman/software/cluster/, 1997.

[14] Michael C. Burl and Pietro Perona. Recognition of planar object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 223–230, San Francisco, 1996.

[15] Michael C. Burl, Markus Weber, and Pietro Perona. A probabilistic approach to object recognition using local photometry and global. In *European Conference on Computer Vision*, pages 628–641, Freiburg, 1998.

[16] Gustavo Carneiro and David Lowe. Sparse flexible models of local features. In *European Conference on Computer Vision*, pages 29–43, Graz, 2006.

[17] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, 2007.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

[19] David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. Spatial priors for part-based recognition using statistical models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10–17, San Diego, 2005.

[20] David J. Crandall and Daniel P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *European Conference on Computer Vision*, pages 16–29, Graz, 2006.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, San Diego, 2005.

[22] Gyuri Dorko and Cordelia Schmid. Selection of scale-invariant parts for object class recognition. In *IEEE International Conference on Computer Vision*, pages 634–639, Nice, 2003.

[23] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 998–1005, San Francisco, 2010.

[24] Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, New York, 1993.

[25] Boris Epshtein and Shimon Ullman. Feature hierarchies for object classification. In *IEEE International Conference on Computer Vision*, pages 220–227, Beijing, 2005.

[26] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, **88**:303–338, 2010.

[27] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html, 2011.

[28] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, and Luc Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf, 2006.

[29] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, John Winn, and Luc Van Gool. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[30] Li Fei-Fei. Knowledge transfer in learning to recognize visual object classes. In *International Conference on Development and Learning*, Bloomington, 2006.

[31] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**:594–611, 2006.

[32] Pedro F. Felzenszwalb, Ross B. Girshick, and David McAllester. Discriminatively trained deformable part models, release 3. http://people.cs.uchicago.edu/~pff/latent-release3/, 2009.

[33] Pedro F. Felzenszwalb, Ross B. Girshick, and David McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/~pff/latent-release4/, 2010.

[34] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**:1627–1645, 2010.

[35] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.

[36] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, **61**:55–79, 2005.

[37] Pedro F. Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, 2008.

[38] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–271, Madison, 2003.

[39] Rob Fergus, Pietro Perona, and Andrew Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 380–387, San Diego, 2005.

[40] Rob Fergus, Pietro Perona, and Andrew Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, **71**:273–303, 2007.

[41] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Integrating multiple model views for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 105–112, Washington, 2004.

[42] Sanja Fidler, Marko Boben, and Aleš Leonardis. Evaluating multi-class learning strategies in a hierarchical framework for object detection. In *Advances in Neural Information Processing Systems*, pages 531–539, Vancouver, 2009.

[43] Sanja Fidler and Aleš Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, 2007.

[44] Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, **22**:67–92, 1973.

[45] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, **5**:1531–1555, 2004.

[46] Patrick J. Flynn and Anil K. Jain. CAD-based computer vision: From CAD models to relational graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**:114–132, 1991.

[47] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**:119–139, 1997.

[48] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, **315**:972–976, 2007.

[49] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, **28**:337–374, 2000.

[50] Mario Fritz. *Modeling, Representing and Learning of Visual Categories*. PhD thesis, Technische Universität Darmstadt, 2008.

[51] Mario Fritz, Bastian Leibe, Barbara Caputo, and Bernt Schiele. Integrating representative and discriminant models for object category detection. In *IEEE International Conference on Computer Vision*, pages 1363–1370, Beijing, 2005.

[52] Michael Garland. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University, 1999.

[53] Gurman Gill and Martin Levine. Multi-view object detection based on spatial consistency in a low dimensional space. In *Symposium of the German Association for Pattern Recognition*, pages 211–220, Jena, 2009.

[54] Daniel Glasner, Meirav Galun, Sharon Alpert, Ronen Basri, and Gregory Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *IEEE International Conference on Computer Vision*, pages 1275–1282, Barcelona, 2011.

[55] Chris Goad. Special purpose automatic programming for 3D model-based vision. In *DARPA Image Understanding Workshop*, pages 94–104, Arlington, 1983.

[56] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, **8**:725–760, 2007.

[57] Chunhui Gu and Xiaofeng Ren. Discriminative mixture-of-templates for viewpoint classification. In *European Conference on Computer Vision*, pages 408–421, Heraklion, 2010.

[58] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**:1157–1182, 2003.

[59] Chuck Hansen and Thomas C. Henderson. CAGD-based computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**:1181–1193, 1989.

[60] Chuck Hansen, Thomas C. Henderson, and Rod Grupen. CAD-based 3-D object recognition. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 168–172, Cambridge, 1989.

[61] Bernd Heisele, Gunhee Kim, and Andrew J. Meyer. Object recognition with 3D models. In *British Machine Vision Conference*, pages 29.1–29.11, London, 2009.

[62] Thomas C. Henderson, Eliot Weitz, Chuck Hansen, Rod Grupen, C. C. Ho, and Bir Bhanu. CAD-based robotics. In *IEEE International Conference on Robotics and Automation*, pages 631–635, Raleigh, 1987.

[63] Tom Henderson, Chuck Hansen, Ashok Samal, C. C. Ho, and Bir Bhanu. CAGD-based 3-D visual recognition. In *IEEE International Conference on Pattern Recognition*, pages 230–232, Paris, 1986.

[64] Derek Hoiem, Alexei Alyosha Efros, and Martial Hebert. Putting objects in perspective. *International Journal of Computer Vision*, **80**:3–15, 2008.

[65] Derek Hoiem, Carsten Rother, and John Winn. 3D layoutCRF for multi-view object class recognition and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, 2007.

[66] Alex Holub and Pietro Perona. A discriminative framework for modelling object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 664–671, San Diego, 2005.

[67] Tony Jebara. *Discriminative, Generative and Imitative Learning.* PhD thesis, Massachusetts Institute of Technology, 2002.

[68] Pierre Jolicoeur, Mark A. Gluck, and Stephen M. Kosslyn. Pictures and names: Making the connection. *Cognitive Psychology*, **16**:243–275, 1984.

[69] Biliana Kaneva, Antonio Torralba, and William T. Freeman. Evaluation of image features using a photorealistic virtual world. In *IEEE International Conference on Computer Vision*, pages 2282–2289, Barcelona, 2011.

[70] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, New York, 2006.

[71] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *European Conference on Computer Vision Workshops (ECCV Workshops)*, pages 17–32, Prague, 2004.

[72] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, **77**:259–289, 2008.

[73] Bastian Leibe and Bernt Schiele. Interleaved object categorization and segmentation. In *British Machine Vision Conference*, pages 759–768, Norwich, 2003.

[74] Bastian Leibe and Bernt Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. In *Symposium of the German Association for Pattern Recognition*, pages 145–153, Tübingen, 2004.

[75] Kobi Levi, Michael Fink, and Yair Weiss. Learning from a small number of training examples by exploiting object categories. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pages 96–102, Washington, 2004.

[76] Joerg Liebelt. *Detection de Classes d'Objets et Estimation de leurs Poses a partir de Modeles 3D Synthetiques.* PhD thesis, L'Universite de Grenoble, 2010.

[77] Joerg Liebelt and Klaus Schertler. Precise registration of 3D models to images by swarming particles. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, 2007.

[78] Joerg Liebelt and Cordelia Schmid. Multi-view object class detection with a 3D geometric model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1688–1695, San Francisco, 2010.

[79] Joerg Liebelt, Cordelia Schmid, and Klaus Schertler. Viewpoint-independent object class detection using 3D feature maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, 2008.

[80] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Symposium of the German Association for Pattern Recognition*, pages 297–304, Magdeburg, 2003.

[81] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, **31**:355–395, 1987.

[82] Javier Marin, David Vazquez, David Geronimo, and Antonio M. Lopez. Learning appearance in virtual scenarios for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 137–144, San Francisco, 2010.

[83] Liang Mei, Jingen Liu, Alfred Hero, and Silvio Savarese. Robust object pose estimation via statistical manifold modeling. In *IEEE International Conference on Computer Vision*, pages 967–974, Barcelona, 2011.

[84] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Multiple object class detection with a generative model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–36, New York, 2006.

[85] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, **60**:63–86, 2004.

[86] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**:1615–1630, 2005.

[87] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**:349–361, 2001.

[88] Andreas Opelt and Axel Pinz. Object localization with boosting and weak supervision for generic object recognition. In *Scandinavian Conference on Image Analysis*, pages 862–871, Joensuu, 2005.

[89] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**:416–431, 2006.

[90] Andreas Opelt, Axel Pinz, and Andrew Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3–10, New York, 2006.

[91] Patrick Ott and Mark Everingham. Shared parts for deformable part-based models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1513–1520, Colorado Springs, 2011.

[92] Mustafa Ozuysal, Vincent Lepetit, and Pascal Fua. Pose estimation for category specific multiview object localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785, Miami, 2009.

[93] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, **38**:15–33, 2000.

[94] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3D geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3362–3369, Providence, 2012.

[95] Xavier Perrotton, Marc Sturzel, and Michel Roux. Implicit hierarchical boosting for multi-view object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 958–965, San Francisco, 2010.

[96] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, **18**:311–317, 1975.

[97] Leonid Pishchulin, Arjun Jain, Christian Wojek, Mykhaylo Andriluka, Thorsten Thormaehlen, and Bernt Schiele. Learning people detection models from few training samples. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1480, Colorado Springs, 2011.

[98] Nima Razavi, Juergen Gall, and Luc Van Gool. Scalable multi-class object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1505–1512, Colorado Springs, 2011.

[99] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. Technical report, Carnegie Mellon University, 1995.

[100] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, 2007.

[101] Johannes Schels, Joerg Liebelt, and Rainer Lienhart. Self-calibrating 3D context for retrieving people with luggage. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1920–1927, Barcelona, 2011.

[102] Johannes Schels, Joerg Liebelt, and Rainer Lienhart. Learning an object class representation on a continuous viewsphere. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3170–3177, Providence, 2012.

[103] Johannes Schels, Joerg Liebelt, Klaus Schertler, and Rainer Lienhart. Building a semantic part-based object class detector from synthetic 3D models. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, Barcelona, 2011.

[104] Johannes Schels, Joerg Liebelt, Klaus Schertler, and Rainer Lienhart. Synthetically trained multi-view object class and viewpoint detection for advanced image retrieval. In *ACM International Conference on Multimedia Retrieval*, pages 3:1–3:8, Trento, 2011.

[105] Bernhard Schoelkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, 2001.

[106] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. A K Peters/CRC Press, 2009.

[107] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, pages 1–15, Graz, 2006.

[108] Michael Stark, Michael Goesele, and Bernt Schiele. A shape-based object class model for knowledge transfer. In *IEEE International Conference on Computer Vision*, pages 373–380, Kyoto, 2009.

[109] Michael Stark, Michael Goesele, and Bernt Schiele. Back to the future: Learning shape models from 3D CAD data. In *British Machine Vision Conference*, pages 106.1–106.11, Aberystwyth, 2010.

[110] Hao Su, Min Sun, Li Fei-Fei, and Silvio Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *IEEE International Conference on Computer Vision*, pages 213–220, Kyoto, 2009.

[111] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision*, pages 1331–1338, Beijing, 2005.

[112] Min Sun, Hao Su, Silvio Savarese, and Li Fei-Fei. A multi-view probabilistic model for 3D object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1247–1254, Miami, 2009.

[113] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. Technical report, Massachusetts Institute of Technology, 1994.

[114] Geoffrey R. Taylor, Andrew J. Chosak, and Paul C. Brewer. Ovvv: Using virtual worlds to design and evaluate surveillance systems. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, 2007.

[115] Alexander Thomas, Vittorio Ferrari, Bastian Leibe, Tinne Tuytelaars, Bernt Schiele, and Luc Van Gool. Towards multi-view object class detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1589–1596, New York, 2006.

[116] Antonio Torralba. *Neurobiology of Attention*, chapter Contextual Influences on Saliency, pages 586–593. Academic Press / Elsevier, 2005.

[117] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**:854–869, 2007.

[118] Shimon Ullman, Erez Sali, and Michel Vidal-Naquet. A fragment-based approach to object representation and classification. In *International Workshop on Visual Form*, pages 85–100, Capri, 2001.

[119] Markus Ulrich, Christian Wiedemann, and Carsten Steger. CAD-based recognition of 3D objects in monocular images. In *IEEE International Conference on Robotics and Automation*, pages 1191–1198, Kobe, 2009.

[120] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[121] Nuno Vasconcelos and Manuela Vasconcelos. Scalable discriminant feature selection for image retrieval and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages II–770–II–775, Washington, 2004.

[122] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *IEEE International Conference on Computer Vision*, pages 606–613, Kyoto, 2009.

[123] Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *IEEE International Conference on Computer Vision*, pages 281–288, Nice, 2003.

[124] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, Kauai, 2001.

[125] Markus Weber, Max Welling, and Pietro Perona. Towards automatic discovery of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 101–108, Hilton Head Island, 2000.

[126] Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, pages 18–32, Dublin, 2000.

[127] Lance Williams. Casting curved shadows on curved surfaces. In *Conference on Computer Graphics and Interactive Techniques*, pages 270–274, Atlanta, 1978.

[128] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science & Engineering*, **4**:10–21, 1997.

[129] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, San Francisco, 2010.

[130] Pingkun Yan, Saad M. Khan, and Mubarak Shah. 3D model based object class detection in an arbitrary view. In *IEEE International Conference on Computer Vision*, pages 1–6, Rio de Janeiro, 2007.

[131] Long Zhu, Yuanhao Chen, Antonio Torralba, William Freeman, and Alan Yuille. Part and appearance sharing: Recursive compositional models for multi-view multi-object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1919–1926, San Francisco, 2010.

[132] Muhammad Zeeshan Zia, Michael Stark, Bernt Schiele, and Konrad Schindler. Revisiting 3D geometric models for accurate object shape and pose. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 569–576, Barcelona, 2011.

Curriculum Vitae

Personal Data

Name:	Johannes Schels
Date of birth:	June 21, 1983
Place of birth:	Regensburg, Germany
Citizenship:	German

Education

07/2007 - 01/2009	Studies of Electrical Engineering and Information Technology at the Technical University of Munich, Germany Graduation with the degree *M.Sc. (TU)*
10/2002 - 03/2007	Studies of Electrical Engineering at the University of Applied Sciences Regensburg, Germany Graduation with the degree *Dipl.-Ing. (FH)*