Models for Multi-View Object Class Detection

by

Han-Pang Chiu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author ....
Department of Electrical Engineering and Computer Science
Feb 18, 2009

Certified by ...
Tomás Lozano-Pérez
Professor
Thesis Supervisor

Certified by ...
Leslie Pack Kaelbling
Professor
Thesis Supervisor

Accepted by ....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Models for Multi-View Object Class Detection

by

Han-Pang Chiu

Submitted to the Department of Electrical Engineering and Computer Science
on Feb 18, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

Learning how to detect objects from many classes in a wide variety of viewpoints is a key goal of computer vision. Existing approaches, however, require excessive amounts of training data. Implementors need to collect numerous training images not only to cover changes in the same object's shape due to the viewpoint variation, but also to accommodate the variability in appearance among instances of the same class.

We introduce the Potemkin model, which exploits the relationship between 3D objects and their 2D projections for efficient and effective learning. The Potemkin model can be constructed from a few views of an object of the target class. We use the Potemkin model to transform images of objects from one view to several other views, effectively multiplying their value for class detection. This approach can be coupled with any 2D image-based detection system. We show that automatically transformed images dramatically decrease the data requirements for multi-view object class detection.

The Potemkin model also allows detection systems to reconstruct the 3D shapes of detected objects automatically from a single 2D image. This reconstruction generates realistic views of 3D models, and also provides accurate 3D information for entire objects. We demonstrate its usefulness in three applications: robot manipulation, object detection using 2.5D data, and generating 3D 'pop-up' models from photos.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Professor

Thesis Supervisor: Leslie Pack Kaelbling
Title: Professor

Acknowledgments

I would like to thank Tomás Lozano-Pérez and Leslie Pack Kaelbling for being my advisors and mentors. They have been very inspiring and supportive throughout this research. Tomás gave me flexibility to pursue my ideas, and would always offer helpful suggestions at critical moment. Leslie was a reliable source of advice, and her encouraging personality also contributed to the accomplishment of this thesis. I could never complete this research without their guidance.

I would also like to thank Bill Freeman for being my thesis committee member. His careful reading of this thesis and insightful feedback have strengthened this research.

Members of the Learning and Intelligent Systems group also have contributed to this research. It has been great to work with them over the past few years. I am particularly thankful for the experimental help and discussions from Sam Davies, Kaijen Hsiao, Meg Aycinena Lippow, Sarah Finney, and Huan Liu. I would also like to thank all the other members of the group for being friendly and helpful colleagues, especially Ben Hong, Michael Ross, and Luke Zettlemoyer.

I want to thank people who have contributed a great deal to my experience as a graduate student at MIT. I am very happy to have become friends with Tom Yeh, Sybor Wang, Jinyang Li, David Huynh, Sayan Mitra, Rui Fan, Hung-An Chang, Vineet Sinha, Karun Bakshi, and Harold Fox. I am also thankful to have had the opportunity to share an office with Yushi Xu, Jingjing Liu, Che-Kuang Lin, Ming Tang, Alex Park, Luis Perez-Breva, and Claire Monteleoni. I have enjoyed every moment with them.

I am deeply grateful to my parents and my sister, for their love and support. Finally, my heartfelt appreciation goes to my wife, Ju-Hui. This thesis is dedicated to her.

Contents

1 Introduction **17**

1.1 Detecting Object Classes in Multiple Views 17

1.2 Thesis Overview 19

1.2.1 Contribution: The Potemkin Model 20

1.3 Road Map 23

2 Background and Related Work **25**

2.1 Learning Object Classes in Multiple Views 25

2.1.1 Background 26

2.1.2 Multiple 2D Models 27

2.1.3 Cross-View Constraints 27

2.1.4 Explicit 3D Models 29

2.1.5 Comparison 30

2.2 The Use of the Potemkin Model 32

2.2.1 Virtual Example Generation 32

2.2.2 3D Reconstruction from a Single Image 34

3 The Basic Potemkin Model **37**

3.1 The Basic Potemkin Model 37

3.2 Estimating the Model 39

3.2.1 Generic Phase: Learning Generic View Transformations 40

3.2.2 Class-Specific Phase: Refining the View Transformations . . . 42

3.2.3 Class-Specific Phase: Learning the Class Skeleton 43

3.3 View-Specific Phase: Using the Model to Generate Virtual Images . . 44

3.4 View-Specific 2D detection 46

3.4.1 EigenEdgeStar (EES) Model 47

3.4.2 Training 48

3.4.3 Detection 49

3.5 Experimental Results 50

3.5.1 Localization Performance 50

3.5.2 Localization Results on Previously Unseen Viewpoints 52

3.5.3 Detection Results 53

4 The Generalized Potemkin Model **57**

4.1 The Use of Multiple Primitive Shapes 57

4.1.1 Oriented Shape Primitives 58

4.1.2 Selecting Primitives 60

4.1.3 Learning and Using an Object Class Model 62

4.2 Self-Supervised Part Labeling 64

4.3 Experimental Results 66

4.3.1 Dataset 66

4.3.2 Basis Set of Primitives 68

4.3.3 Part Labeling 70

4.3.4 Single-View Detection 72

4.3.5 Multi-View Detection 77

4.3.6 Dalal-Triggs Detector 78

5 The 3D Potemkin Model **81**

5.1 3D Class Models 81

5.1.1 Estimating a 3DP Model from Data 83

5.2 Automatic Single-View Reconstruction 85

5.2.1 Detection and Segmentation 85

5.2.2 Part Registration 86

5.2.3 Creating the 3D Model 86

5.2.4 Estimating Locations of Occluded Parts 88

5.3 3D Object Localization 89

5.3.1 Localization Performance 89

5.3.2 Robot Manipulation 90

5.4 Object Detection Using 2.5D Data 93

5.5 3D Pop-Up Scenes from a Single Image 95

6 Conclusions **97**

List of Figures

1-1 Three specified tasks in object detection. For each task, systems are supposed to detect objects in the test set (right) after they learn models from the training set (left). 18

1-2 The two red arrows represent how the Potemkin model can be used with a multi-view object class detection system. One is to decrease the training-data requirement for effective detection. The other is to reconstruct 3D shapes of the detected objects from a 2D image. . . . 20

1-3 The architecture of the Potemkin model. The Potemkin model is trained in the first two phases, and used in the third phase. 22

2-1 The relationship between the Potemkin model and other learning approaches to multi-view object class detection. 30

3-1 The architecture of the basic Potemkin model. 39

3-2 A synthetic nearly-planar box object used for learning the generic view transformations. 40

3-3 The ellipses indicate the distribution of 2D positions of the centroids of each part in the training set of chair images for each viewpoint. The 3D spheres show the estimate of the skeleton positions S_j for each part. The size of the spheres is not significant. 44

3-4 The basic Potemkin model in the top two rows is constructed from only two real labeled images (left). Given the label images for two object instances, each in only one viewpoint bin (highlighted), the other virtual views are generated from the basic Potemkin model. The model in bottom two rows is constructed from a total of 10 real images, from 5 object instances over 6 views. 45

3-5 Full virtual images and the actual input images (bold border) of two object instances in the chair data set. 46

3-6 The 6 parts of a chair and the 5 parts of a bicycle. 50

3-7 Localization performance of EES for chairs and bicycles, using all the training data for all views. 51

3-8 Successful localizations of chairs and bicycles using EES. 52

3-9 Localization performance of EES for chairs as a function of the number of views in the training data. 53

3-10 ROC curves for single-view detection (top). ROC curves for multi-view bicycle detection (bottom). 54

4-1 Virtual views generated from the highlighted image using a basic Potemkin model that is minimally trained. 58

4-2 The top row of the figure shows the three 3D primitive shapes: cube, flat block, and stick. Each of these shapes is considered at several orientations with respect to the skeleton. The lower part of the figure shows, for the flat shape, several views of the shape at three example (azimuth, elevation) orientations. 59

4-3 Generative model for 2D part-specific transforms. There are plates for the M possible oriented primitives, for the N parts of the object model, and for the k^2 view bin pairs. $U^m_{\alpha,\beta}$ and $T^j_{\alpha,\beta}$ represent the view transform from view bin α to β for oriented primitive m and part j respectively. $R^j_{\alpha,\beta}$ represents an actual observed transform of part j from view bin α to β in real data. Z_j is an indicator variable that selects which primitive m is the appropriate shape for part j of the object. 60

4-4 Visualization of 3D skeleton and shape primitives. Top: two views of the four-legged chair model; bottom: two views of the airplane model. Note that these models were each constructed from two part-labeled images of the same object, knowing the view bins but with no further camera calibration. 63

4-5 First row: virtual images generated from highlighted instance without occlusion handling; second row: with occlusion handling. 64

4-6 The generalized Potemkin model (second and fourth rows) generates more realistic virtual images than the basic Potemkin model (first and third rows). 65

4-7 Given a model instance with labeled parts (blue), the parts of another instance (red) in the same view can be found by matching points along the boundaries of the instances (middle) and by deforming the model instance into the target instance (right). 66

4-8 Decomposition of object classes into parts. 67

4-9 Four selected view bins for bicycles, cars, and airplanes. 67

4-10 The quality of transforms as the number of available primitives (M) increases. Left: evaluated on training data; right: evaluated on held-out test data. 69

4-11 3D shape primitives selected for each part of each class. 70

4-12 Some part labeling results for four-legged chairs. First row: hand labeling; second row: self-supervised labeling. 71

4-13 Some part labeling results of bicycles. First row: hand labeling; second row: self-supervised labeling. 71

4-14 Some part labeling results of cars. First row: hand labeling; second row: self-supervised labeling. 72

4-15 Average single novel view detection results over four repetitions of each of four object classes: chairs (left-top), bicycles (right-top), airplanes (left-bottom), and cars (right-bottom). 74

4-16 The virtual examples (from left to right) are constructed by using the generalized Potemkin model, the basic Potemkin model, and a global transformation respectively. These virtual examples are generated by transforming the car in the highlighted image from side viewpoint to frontal viewpoint. 75

4-17 Six examples of real images (highlighted) and virtual images constructed by the generalized Potemkin model. 76

4-18 Multi-view detection performance (ROC curve) for chairs and cars. . 77

4-19 The precision-recall curves of the Dalal-Triggs detector using our training images. 79

5-1 Learned 3DP class model for four-legged chairs in the object-centered reference frame, and in each view reference frame. 84

5-2 The 3DP class model of airplanes (right), constructed from two images of a real airplane, looks like the paper cutout airplane (left). 85

5-3 The processing pipeline for automatic single-view 3D reconstruction. . 88

5-4 One failed example of our automatic single-view 3D reconstruction. . 88

5-5 A 3D reconstruction (left) from the green-circled car, which is in the image (right) of 20 model cars. 89

5-6 Some snapshots taken in the process (from left to right, from the first row to the second row) of the robot grasping a car. 91

5-7 Top (From left to right): Our 3D class model of baskets, the input image, and two views of the reconstructed 3D basket. The handle is recovered even it is occluded in the input image. Bottom: Some snapshots taken in the process (from left to right, from the first row to the second row) of the robot's grasp of a basket 92

5-8 Imaged objects with corresponding stereo depth maps (the second row) and depth maps generated by reconstruction from the 3DP model (the third row). 93

5-9 ROC curves for car detection. 94

5-10 Row 1: Original image. Row 2 and 3: 3D instance model generated from the 3DP class model. 95

Chapter 1

Introduction

1.1 Detecting Object Classes in Multiple Views

The problem we address in this dissertation is how to efficiently learn to detect object classes from a wide variety of views. This problem is crucial to the task of multi-view object class detection, which is to detect novel object instances from a wide range of viewpoints after training with some instances of the target class. In this chapter, we will first discuss the primary challenges of multi-view object class detection at a high level, and then we will overview the contributions of this research.

Object detection, including several specialized tasks shown in Figure 1-1, is the classical problem in computer vision of determining whether a single image contains some pre-specified objects. The traditional task of object detection is to identify a particular object - such as George Bush's face or one specific Adidas sneaker - in a cluttered image under viewpoint changes, given only a few views of that object for learning. This task, which is called "multi-view specific object detection", has been studied by numerous researchers. However, systems designed for this task can only find a specific object shown in the model images.

Over the past five years, researchers have started to look at the task of detecting

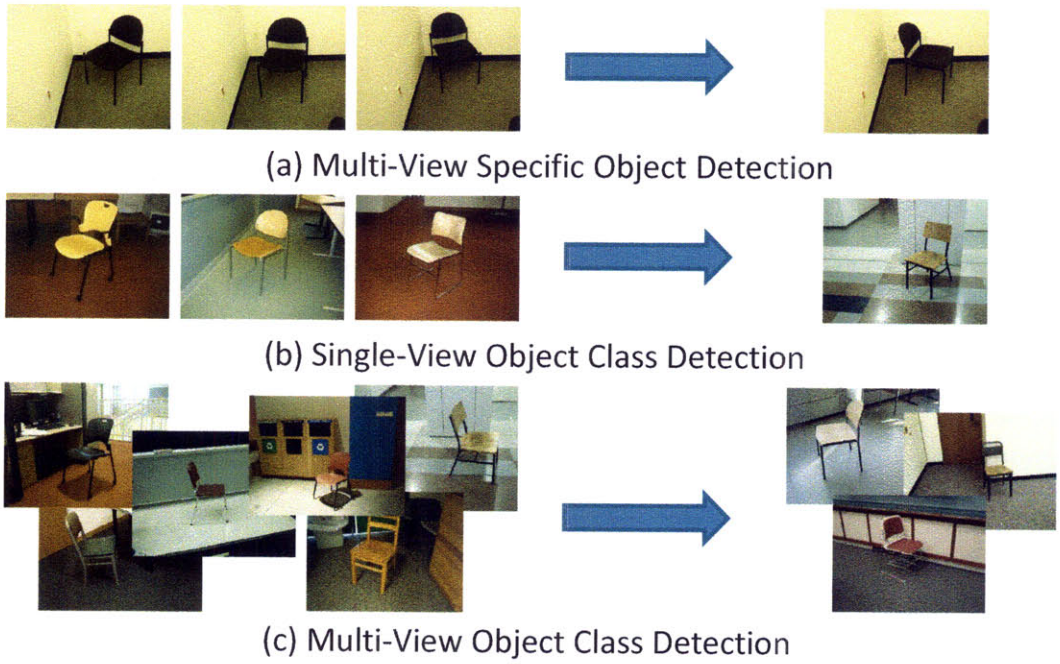

Figure 1-1: Three specified tasks in object detection. For each task, systems are supposed to detect objects in the test set (right) after they learn models from the training set (left).

"classes of objects", such as cars and airplanes, rather than specific objects that have already been memorized. This task seems to be more complicated, because there may be large variations in appearance and shape among different instances of the same class. Researchers have successfully enabled systems to recognize previously unseen object instances of a generic category after training these systems with a collection of object instances of that class. However, these systems have a major restriction: they are typically limited to a single viewpoint, e.g. side-views of airplanes. Thus we call this task "single-view object class detection" in order to differentiate it from the traditional task of detecting particular objects.

To better match humans' ability to perceive the objects, such as to identify novel instances from arbitrary viewpoints, we need to create a system that detects object classes in a wide range of viewpoints instead of just a single predefined aspect. Relying on recent advances in single-view object class detection, several researchers have begun to work on this topic. This newly emerging task, "multi-view object class

detection", is especially difficult because of the variations due to viewpoint transformations, as well as the variability among instances of the same class.

One of the key challenges for multi-view object class detection is learning. Suppose we plan to design a visual system that can quickly detect hundreds of object categories from arbitrary viewpoints, the training set, including images of object instances that are viewed under various poses, should be as small as possible. Human-supervised labeling tasks, such as eliminating the background in the training images, should be reduced as much as possible.

However, current approaches for multi-view object class detection require many real training images from many views for many object instances in the same class to achieve satisfactory detection results. They need to collect numerous training images for learning: not only to cover the various shapes of the same object due to viewpoint transformation, but also to accommodate the variability in appearance among instances of the same class. Moreover, these systems cannot deal with new viewpoints that have not been trained before.

The learning process in these systems is inefficient in three ways:

- The learning process is restricted to the viewpoints where training images of object instances are available.
- Learning procedures for each viewpoint of the same object class are performed without sharing any (or sufficient) information between viewpoints.
- Learning is handled as an independent procedure for each individual object class.

1.2 Thesis Overview

To efficiently learn object classes in multiple views, the learning procedures for each viewpoint of the same object class, as well as each instance of each object class, should

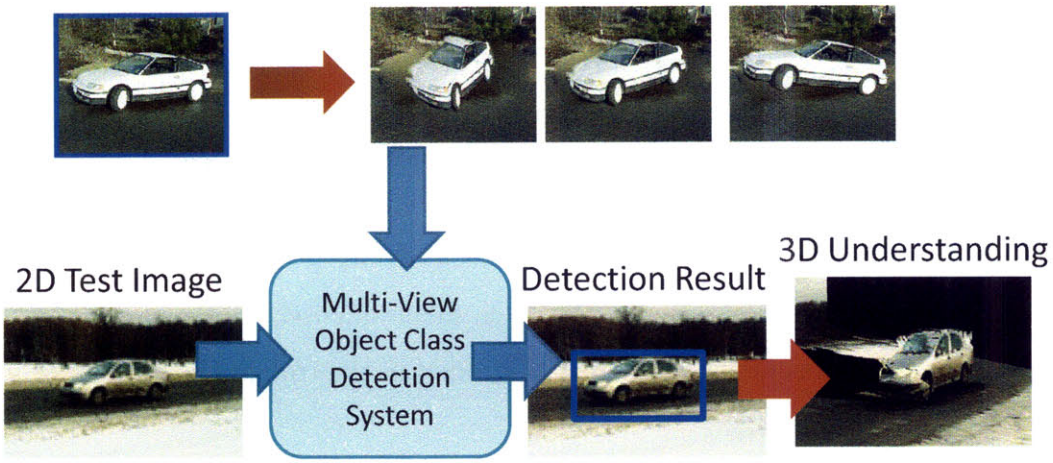

Figure 1-2: The two red arrows represent how the Potemkin model can be used with a multi-view object class detection system. One is to decrease the training-data requirement for effective detection. The other is to reconstruct 3D shapes of the detected objects from a 2D image.

be related and coherent. During the learning process, information should be gathered from and shared with all trained views, and generalized into viewpoints that have not been seen before. Our research embraces these ideas: We have developed a model that transfers information from one view to another, and this model can be learned for each multi-view object class without undue effort. The learned model can also be extended to represent the approximate 3D shape of the target class.

1.2.1 Contribution: The Potemkin Model

In this research, we examine what information can be transferred across views, and generalized for the learning of novel viewpoints of the same class as well as new object classes. The primary goal of this dissertation is to design a model that can employ this information in the learning process, that can be efficiently constructed from a few training images, and that then can be used with a multi-view object class detection system in two ways, as shown in Figure 1-2.

We develop the *Potemkin*[1] *model*, which facilitates information transfer to help

[1]So-called "Potemkin villages' were artificial villages, constructed only of facades. Our models,

the learning of target object classes in multiple views. The Potemkin model exploits the fact that there is a fundamental relationship between multiple views of an object class, and allows subsequent unlabeled views of new objects from the same class to be transformed into novel views. This ability to transform training examples into new views can generate "virtual" training data to circumvent the excessive training data requirement for multi-view object class detection.

The Potemkin model of an object class is a collection of parts, which are oriented 3D primitive shapes. The model allows the parts to have an arbitrary arrangement in 3D, but assumes that, from any viewpoint, the parts themselves can be treated as being nearly planar. We will refer to the arrangement of the part centroids in 3D as the *skeleton* of the class. As one moves the viewpoint around the object, the part centroids move as dictated by the 3D skeleton; but rather than having the detailed 3D shape model that would be necessary for predicting each part's shape in each view, we model the change in views of each part's image as a 2D perspective transformation.

The Potemkin model is trained and used in three phases (see Figure 1-3). The first phase is class-independent and carried out once only. In it, the system learns, for each element of a set of simple oriented 3D primitive shapes, what 2D image transformations are induced by changes of viewpoint of the shape primitive. The necessary data can be simply acquired from synthetic image sequences of a few objects rotating through the desired space of views. After the first phase is complete, the model can be used for new object classes with very little overhead. This process might be seen as learning basic view transformations of primitive 3D shapes by "watching the world go by", and then transferring that knowledge to accelerating the learning of new object types which are constituted of these primitives.

The second phase is class-specific: a few images from a target class (typically a single object from two views) are used to estimate the 3D skeleton of the object class, to select an oriented primitive 3D shape for each of the parts, and to initialize part transforms between pairs of views, for each part.

---

too, are constructed of facades.

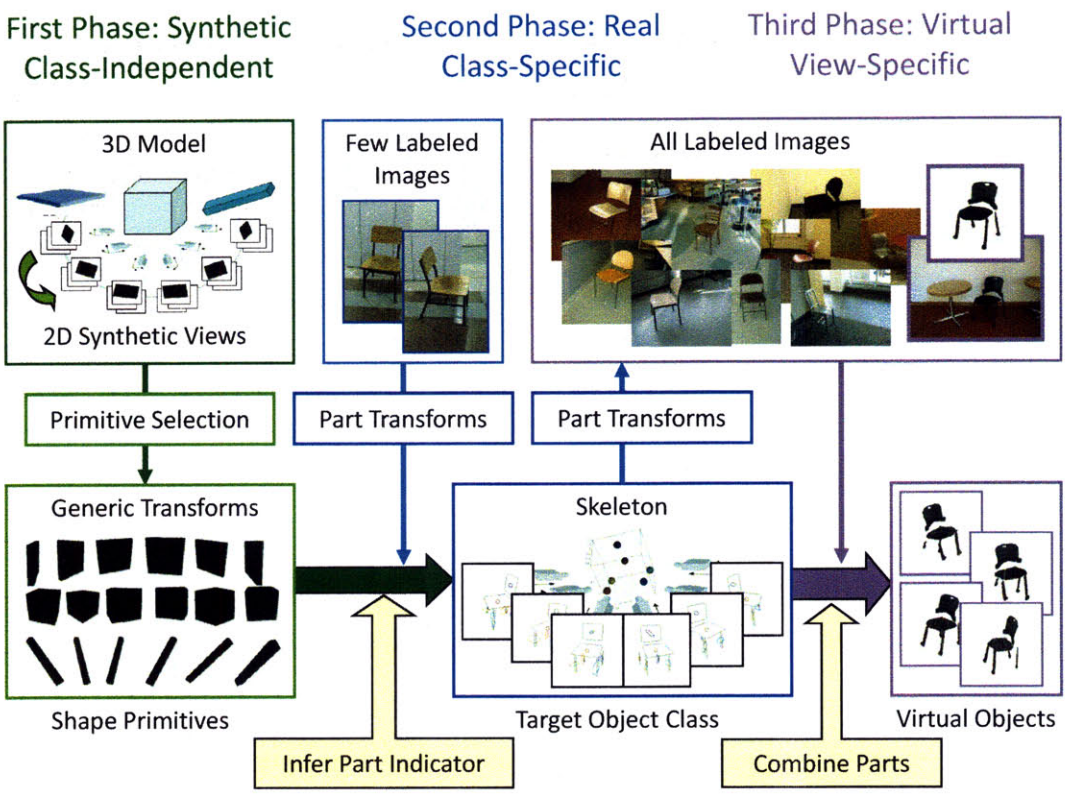

Figure 1-3: The architecture of the Potemkin model. The Potemkin model is trained in the first two phases, and used in the third phase.

The third phase is view-specific: given a new view of interest, the class skeleton is projected into that view, specifying the 2D centroids of the parts. Then, all available 2D training images, from any view, are transformed into this view using the image transforms selected in the second phase. These new images are "virtual training examples" of previously seen objects from novel views. These virtual training examples, along with any real training examples that are available, can then be used as training data for any view-dependent 2D detection system, which can then be used to detect instances of the object class in the novel viewpoint.

The Potemkin model therefore provides a data-efficient learning approach to multi-view object class detection. Using the Potemkin model, systems are able to generate virtual training examples in many views from every example in any view. This dramatically decreases the training-data requirement for multiple-view class detection

by maximally utilizing the value of each real training image.

The Potemkin model also presents an approximate 3D shape model of the target object class, by estimating a rough extent and planar model of each part in 3D. When transforming a real training object to a novel view and then forming a virtual object, this easily-obtained 3D class model provides important information about self-occlusion and increases the quality of virtual training examples. The virtual examples generated by the Potemkin model are thus comparable to real examples for training class-specific detectors.

This 3D class model presented by the Potemkin model can also be used to reconstruct the 3D shape of a detected instance automatically from a single 2D image. Once this 3D class model is constructed, this reconstruction mechanism can be built on top of any multi-view object class detection system. The resulting single-view reconstructions allow artificial systems to perceive the 3D shapes of detected objects and to perform mechanical actions on them, such as moving a robotic arm to grasp objects.

1.3 Road Map

In the following chapter we give a more detailed overview of the problem setting and a discussion of related work. We describe current learning approaches for multi-view object class detection, and draw a comparison between these approaches and the Potemkin model. The Potemkin model can be used either to generate virtual training examples, or to reconstruct 3D shapes of detected instances from a single image; therefore, the related work in virtual data generation and 3D reconstruction is also covered in this chapter.

Then in Chapter 3 we sketch the basic version of the Potemkin model, which emphasizes how to transfer information from one viewpoint to another. In the first phase of the basic Potemkin model, there is only a single oriented primitive used for

learning the transforms from view to view. These transforms are then used as the basis for transforms of all parts in the model. We show how the model can be trained from only a few labeled images for the target object class, and how the constructed model can be used to transform images of objects from one view to several other views, effectively multiplying their value of class detection.

Chapter 4 extends the basic Potemkin model to use a basis set of multiple oriented primitives in the first phase, and to select among them to represent each of the parts of the target class based on a pair of initial training images of the class. For each part, the transforms learned from the selected primitive are used to initialize image part transforms between pairs of views. We also show how the part labeling process can be self-supervised and demonstrate that these extensions significantly outperform the basic model.

Once the details of how to decrease the training-data requirement have been presented in previous chapters, we then describe an other way to use the Potemkin model in Chapter 5. We show how to use an extension of the Potemkin model to support reconstruction of the 3D shapes of object instances from a single image. To achieve complete automation of the reconstruction process, we describe an approach involving several steps: detection, segmentation, part registration, and model creation. Then we demonstrate its usefulness in three applications: robot manipulation, object detection, and generating 3D 'pop-up' models from photos.

Chapter 2

Background and Related Work

In this chapter we frame the problem setting for this research and provide an overview of the related work. We describe current approaches for learning object classes in multiple views, and compare these approaches with the Potemkin model (Section 2.1). Since the Potemkin model can be used either to generate virtual training examples, or to reconstruct 3D instances from a single image, we also review related work in virtual example generation (Section 2.2.1) and 3D single-view reconstruction (Section 2.2.2).

2.1 Learning Object Classes in Multiple Views

The task of learning to identify novel object instances from a wide variety of viewpoints is highly challenging due to two main factors. First, a single object instance can look dramatically different depending on its pose relative to the camera. Second, different instances of objects from the same category can exhibit significant variations in appearance and shape.

2.1.1 Background

Most learning methods for object detection are designed to identify a particular object. There has been a great deal of work on modeling and recognition for single 3D objects (e.g., [70, 25, 45, 58]), which is to use multiple views of a specific object instance to construct a 3D model for the object. Such a model allows detection of the particular object from many viewpoints, including entirely novel views. However, these methods are not designed to cover the variability in shape and appearance within instances of a class and therefore cannot be readily generalized for learning object classes. There is also a long tradition of representing objects as arrangements of 3D building blocks (e.g., [16, 56, 47, 46, 4, 5, 51, 55]). In particular, Marr and Nishihara [47] describe objects as a collection of 3D cylinders. Biederman [4] defines an object as relationships between 3D volumetric components. Decoret et al. [16] simplify 3D objects onto a set of 3D polygons with texture and transparency maps. However, it has been difficult to achieve robust detection performance in real images using these approaches.

Most recent advances in learning methods attempt to train a 2D model for detecting examples of a target class from only a single specific viewpoint [23, 24, 37, 10, 11, 3, 15, 28, 40, 69, 72]. This 2D model represents a single-view object class as a collection of distinct features, and learns the spatial relationships among these features. In principle, these single-view learning methods can be extended to cover the range of likely views of a set of object classes. However, training a detector for even a single view of a class already requires a large amount of training data. Multiplying these requirements by the number of views makes this learning approach expensive.

There is already an existing body of work on learning specific object classes, such as cars and faces, for multi-view detection [1, 21, 43, 50, 63, 71]. However, these methods are not readily generalized for learning other object classes. There are also learning approaches for class detection using cartoon images [18] or synthetic images from 3D CAD models [44] of objects to avoid collecting real training images. However,

cartoon images or 3D CAD models with realistic textures are not easily obtained for some object classes. Below we organize our discussion of current learning approaches for multi-view object class detection into three groups: multiple 2D models, cross-view constraints, and 3D explicit models. We also draw a detailed comparison between these approaches and the Potemkin model.

2.1.2 Multiple 2D Models

In most current approaches to object class detection, the problem of learning multiple views of the same object class is treated as learning multiple independent object classes, with a separate 2D model trained for each. Based on this independent-view approach, there are already a number of effective multi-view object class detection systems.

For example, Crandall et al. [12] treat ranges of poses as separate classes, with a separate model learned for each. Torralba et al. [68] show that sharing features among models for multiple views of the same object class improves both running time and detection performance. Leibe et al.'s detection system [39] for cars combines a set of seven view-dependent detectors [41], each of which requires hundreds of training examples to be effective.

Systems based on this learning approach require a large number of training images for each viewpoint, which is expensive. Our work aims to alleviate this burden and to enable existing detection systems to be applied to multi-view detection without excessive training-data requirements.

2.1.3 Cross-View Constraints

The learning approach based on multiple 2D models is inefficient because the learning procedures for each viewpoint of the same object class are performed without sharing any (or sufficient) information between viewpoints. Recently, there have been a range

of approaches that make deeper use of the relationship between views of the same class. These approaches attempt to exploit the fact that 2D images of an object class in multiple views are generated by similar 3D objects. They aim to find a suitable representation that cover the variability among members of the object classes, and they use their representation to learn object classes in multiple views.

One approach to exploiting the relationship between views is to link regions of the same or similar training instances across different viewpoints, forming a multi-view class model. This multi-view class model can then be used to detect object instances from a wide variety of viewpoints. During recognition, linked regions in the multi-view model can work together to determine the location and pose of the object. Recently, there have been three systems using this approach.

Thomas et al. [67] construct separate view-dependent detectors for each of the viewing directions, and track regions of the same training instance across different viewpoints. In their recognition process, these view-dependent detectors act together by linking tracked features across viewpoints to share image evidence coming from different training views. Thus, they can use these integrated view-dependent detectors to improve detection, by transferring the tracked regions from one viewpoint to another during the recognition stage.

Savarese and Fei-Fei [60] also relate multiple views by finding correspondences among 2D regions, formed by feature grouping, on the same training instance across different views. The 2D transformation between each pair of 2D regions from the same viewpoint is also estimated as the geometric relation in their models. Compared to the work of Thomas et al, their models are more compact and can be learned with very weak supervision. Using their models, they can detect the object and decide its pose, by finding a globally consistent combination of multiple 2D regions learned from the same viewpoint.

Rather than constructing multiple view-dependent detectors, Kushal et al. [38] construct a single model that relates partial surfaces of the target object class among

multiple viewpoints. Each partial surface is formed by locally dense rigid assemblies of image features on training instances across different views. The detection is achieved by matching each partial surface in the model to the test image independently, and then using a greedy algorithm to find a most likely global correspondence from their model to the image.

Each of these three methods requires many training images from each viewpoint and the first two methods require many views of the same training instances, which are relatively difficult to obtain.

2.1.4 Explicit 3D Models

An alternative approach to exploiting the relationship between views is to construct explicit 3D shape models of object classes. An ideal scenario is to use multiple views of multiple instances to construct a model of the distribution of 3-dimensional shapes of the object class. Such a model would allow detection of many entirely novel views. However, this is a very difficult problem, which is yet unsolved. Instead of modeling the distribution of 3D shapes of the class, there are two recently developed methods which construct a rough 3D model to represent the target object class.

Hoiem et al. [34] create a coarse 3D model from mean segmentations of two views of training instances, avoiding the need for many views of the same training object instance during learning. They exploit this 3D model to learn physically localized patch appearances and their layout on the training instances. Then they use a conditional random field algorithm on the layout to assign an object label, which defines an object's position and viewpoint, to each of the local patches on the test image. Therefore, they can simultaneously detect and segment objects in a large range of viewpoints on the test image.

Yan et al. [74] use training images, taken from a dense sampling of viewpoints around a specific object, to reconstruct a 3D model for the target class. The 3D

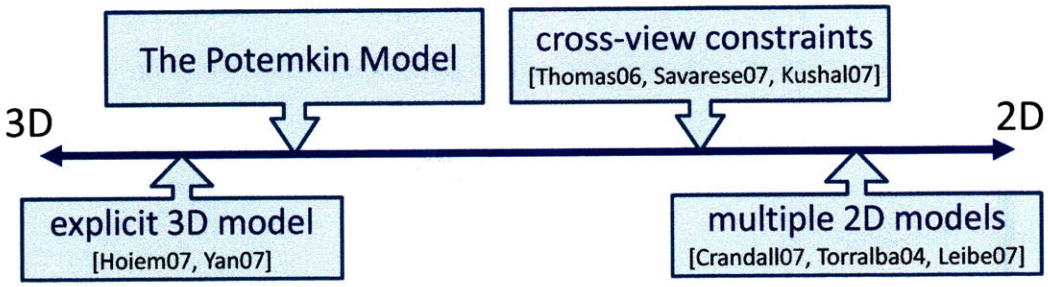

Figure 2-1: The relationship between the Potemkin model and other learning approaches to multi-view object class detection.

object model is reconstructed by using a homographic framework from the set of viewpoints around the object, and is represented by a volume consisting of binary slices. Given a test image, they find matches between the detected 2D image features and the 3D features stored in the model, and propose several hypotheses based on the 3D locations of the matched 3D features in the model. The object on the test image is then detected by evaluating these hypotheses.

Both methods form correspondences between 2D features across training instances at arbitrary viewpoints by projecting the surfaces of the 3D model onto each instance. The learned features thus can be shared across viewpoints through 3D correspondence, which is more flexible than the methods based on cross-view constraints. However, these two systems still require many training instances at many viewpoints to achieve satisfactory detection results. It is crucial to have a data-efficient approach for learning object classes.

2.1.5 Comparison

The Potemkin model we develop in this dissertation is somewhere between the approaches based on cross-view constraints and those based on explicit 3D models, as shown in Figure 2-1. Methods based on either cross-view constraints or explicit 3D models can be difficult to train. Cross-view constraints, which are formed by linking 2D regions of instances from one viewpoint to another, require many training images

in each of the modeled viewpoints. Detailed 3D models of variable object classes can be difficult to learn and use. Therefore, we construct a relatively weak 3D model, which can be learned from a few labeled images, but which is powerful enough to generate virtual examples in novel viewpoints and to support reconstruction of the 3D shapes of object instances from a single image.

There are two major differences between the Potemkin model and other approaches for multi-view object class detection: data-efficiency and compatibility. The Potemkin model facilitates information transfer to help the learning of target object classes in multiple views. Unlike most approaches which require many training images from a wide variety of viewpoints to build their models, the Potemkin model needs a minimum of only two images of one particular object instance to construct a 3D class model. This 3D class model can then be used to generate credible virtual examples and to recover 3D shapes of detected instances from a single image.

The learned Potemkin model allows subsequent unlabeled views of new objects from the same class to be transformed into novel views. Compared to [38, 34], the Potemkin model requires relatively few total training images in each viewpoint for multi-view detection, since images from different viewpoints can be transformed to provide virtual training images for all of the viewpoints. Furthermore, several of these methods [67, 60, 74] make heavy use of multiple views of particular object instances to construct their models, which are generally more difficult to acquire than images of diverse instances each from a different view. On the contrary, the Potemkin model can generate virtual training examples for multiple viewpoints without the need of more than one image for each particular instance.

Unlike other methods, which learn models that are directly used for their own detection systems, the Potemkin model provides a data-efficient approach for any method in multi-view object class detection. For instance, there are a large number of effective single-view detection systems available, and it is possible, in principle, to train enough instances of them to cover the range of likely views of a set of object classes. The Potemkin model thus enables these single-view detection systems to be

applied to multi-view detection without excessive training-data requirements.

And, because the Potemkin model can generate multiple virtual views of a single instance, the resulting virtual training data could potentially be used as input to any of the approaches based on either cross-view constraints or explicit 3D models. The Potemkin model uses information transfer to maximally utilize the value of each training image for the purpose of multi-view object detection, and amplifies the effectiveness of any detection method that needs images from multiple views.

The Potemkin model also allows detection systems to perceive the approximate 3D shapes of detected objects automatically from a single 2D image. This 3D reconstruction is valuable for many applications such as robot manipulation. Most multi-view object class detection systems use learning approaches based on either multiple 2D models or cross-view constraints. Thus, they are only able to obtain 2D shapes from the detected objects, which are not sufficient for artificial systems to interact with the external object in 3D space.

2.2 The Use of the Potemkin Model

Below we review the related work to the two uses of the Potemkin model respectively.

2.2.1 Virtual Example Generation

There has been a long tradition of synthesizing novel views of objects without constructing 3D models of objects [2, 7, 66]. These methods morph a set of trained views of an object into a novel view using principles of projective geometry. The set of source images of the target object needs to contain two views [64], three views [73] or more than three views [31] of the object. However, these methods are designed for single objects with known feature correspondences across views. In contrast, the Potemkin model works at the categorical level rather than the single object level,

generating novel views of an object from only one given view of a particular object.

Novel views of an object can also be generated from a 3D visual hull (Shape-From-Silhouette). Grauman et al. [29] learn a non-parametric density model of object shape for the target class by collecting many multi-view silhouette examples from calibrated cameras. Given a single input silhouette of a novel object, they search for 3D shapes with maximal posterior probability to the input contour, and they infer the visual hull of this object. Virtual views of this input object can then be generated from the interpolation between neighboring trained virtual hulls of the class.

There are a few methods that generate novel views of real training objects for specific object classes. For example, Everingham and Zisserman [19] generate virtual training data for face recognition from multiple viewpoints using a reconstructed 3D model of the head. Compared to their work, the Potemkin model provides a more compact but less accurate approach to generating virtual training data for a broad range of object classes.

The closest work to the Potemkin model's virtual example generation is Savarese and Fei-Fei's method [61]. They extend traditional methods in image-based rendering [2, 7, 66], which synthesize novel views of objects without constructing 3D models of objects, to the category level. Therefore, they can synthesize novel views of trained objects of the target class, for detecting object instances from the viewpoints that have not been trained before.

While the work of Savarese and Fei-Fei [61] and the Potemkin model agree in the general spirit, there are three major differences between these two approaches. First, as opposed to [61] where novel views are synthesized at detection time, the Potemkin model constructs virtual examples to augment real training data during the learning stage. Second, Savarese and Fei-Fei need at least two views of an object instance to morph novel views of that object instance. In contrast, the Potemkin model generates virtual views of an object instance from only one training image. Third, the Potemkin model synthesizes the novel views of the "whole" object as opposed to several partial

patches of the object [61]. The virtual examples generated by the Potemkin model can thus be placed into the background from the original training image to construct a realistic image suitable for use in any existing recognition system.

2.2.2 3D Reconstruction from a Single Image

There are many methods for single-view reconstruction that compute 3D models with assistance from a human. Taking advantage of manual specification of information such as surface normals and discontinuities, these methods can reconstruct either planar scenes [13, 35, 65] or special classes of curved surfaces [53, 75]. One recent work is the 3D extension of the LabelMe system developed by Torralba et al [59]. After users label the objects and the ground region in an image, the system is able to construct a 3D "pop-up" visualization by treating each labeled object as a single 3D plane perpendicular to the ground. There is also a long history of bottom-up methods for general 3D reconstruction of images based on shading, texture, etc. [26].

Recent work in single-view reconstruction focuses on automatic reconstruction of generic scene elements such as the ground plane, vertical building facades, etc. For instance, Hoiem et al. [32] learn segmentation cues to classify image regions into geometric classes (ground, vertical surfaces, and sky), and then create a coarse 3D scene model from the source image based on this classification. Hoiem et al. [33] also use geometric classes to improve the performance of existing object detection systems. They model the contextual relationships between three elements (object detections, rough 3D scene geometry, and approximate camera position/orientation), and then use a statistical framework that simultaneously infers the probabilities of these three elements in a single image. Saxena et al. [62] train a Markov Random Field (MRF) to predict the relationships between various image patches, and the relation between the image features and the 3D location/orientation of the planes. Then they use the MRF to infer a depth map from a 2D intensity image.

There is relatively little work on automatic single-view reconstruction for spe-

cific object types. Prasad et al. [54] adapt an object-specific 2D segmentation technique [36] for curved objects, such as oranges or bananas, and then reconstruct a smooth 3D parametric surface from the segmented region by energy minimization. Romdhani and Vetter [57] collect a database of 3D face models, then form a linear basis to instantiate a 3D face from a single image using MAP estimation.

The Potemkin model allows automatic single-view reconstruction of general object classes, and works well with object classes that can be approximated by a collection of planar patches. In contrast to gathering a large number of 3D instance models for a particular class [57], the Potemkin model constructs an easily-learned 3D model, composed of oriented planar shapes, to represent the target class. This class-specific 3D representation enables the reconstruction of a wide variety of realistic 3D classes, albeit within a limited range of shape variation. Moreover, this 3D class model can predict the 3D shape of the occluded parts of the object. The recovered 3D models provide accurate enough information for applications such as robot manipulation.

Chapter 3

The Basic Potemkin Model

In this chapter we describe the basic version of the Potemkin model in detail, including how to estimate the parameters from a small number of training images (Section 3.2), and how to use it to generate virtual training images in novel views (Section 3.3). These virtual training images, along with any real training images available, can then be fed into any view-dependent 2D detection system. The detection system thus can leverage sparse real training data to achieve multi-view class detection, and detect objects at previously unseen views. We show results in Section 3.5 using both Crandall et al's [10] detection system in a detection task and a detection system we describe in Section 3.4 in a localization task. The basic Potemkin model emphasizes how to transfer information from one view to another, but is ultimately too weak to describe many classes of objects well. We will extend it in Chapter 4.

3.1 The Basic Potemkin Model

Informally, a basic Potemkin model can be viewed as a collection of planar "facades," one for each part, which are arranged in three dimensions. In order to model the transformation of an object from one view to another, the 3D structure is used to specify the location of each part in the new view, and a set of learned view transforms

is used to specify how the pixels belonging to that part in the first view should be transformed into the new view. To achieve this, the view space is divided into a discrete set of *view bins*, and an explicit 2D projective transform between each view bin pair is represented for each part. Different transforms are necessary because the parts have different shapes, depths, and orientations, so their pixels move differently from view to view.

More formally, a Potemkin model for an object class with N parts is defined by:

- k *view bins*, which are contiguous regions of the view sphere;
- k *projection matrices*, $P_\alpha \in R^{2\times 3}$, from normalized 3D coordinates to image coordinates for each view bin α;
- a *class skeleton*, $\langle S_1, \ldots, S_N \rangle$, specifying the 3D positions of part centroids, in a fixed, normalized reference frame; and
- Nk^2 *view transformation matrices*, $T^j_{\alpha,\beta} \in R^{3\times 3}$, one for each part j and pair of view bins α, β, which map points of an image of part j from view α to view β.

This model is appropriate for object classes in which the skeleton is relatively similar for all of the instances of the class; if there is more variability, and especially multi-modality, it will become necessary to extend the model to probability distributions over the skeleton and matrices, and to integrate or sample over model uncertainty when using it for prediction.

Of course, a single linear projection cannot correctly model projections from 3D coordinates to all views in a bin, or from all views in one bin to all views in another; we assume that the bins are small enough that the error in such a model is tolerable. The choice of bin size represents a model-selection problem. The smaller the bins, the more accurate the model, but the more data needed to train it reliably.

Label images indicate which pixels in a training image correspond to each part. A Potemkin model, together with a label image whose view bin is known, can be used

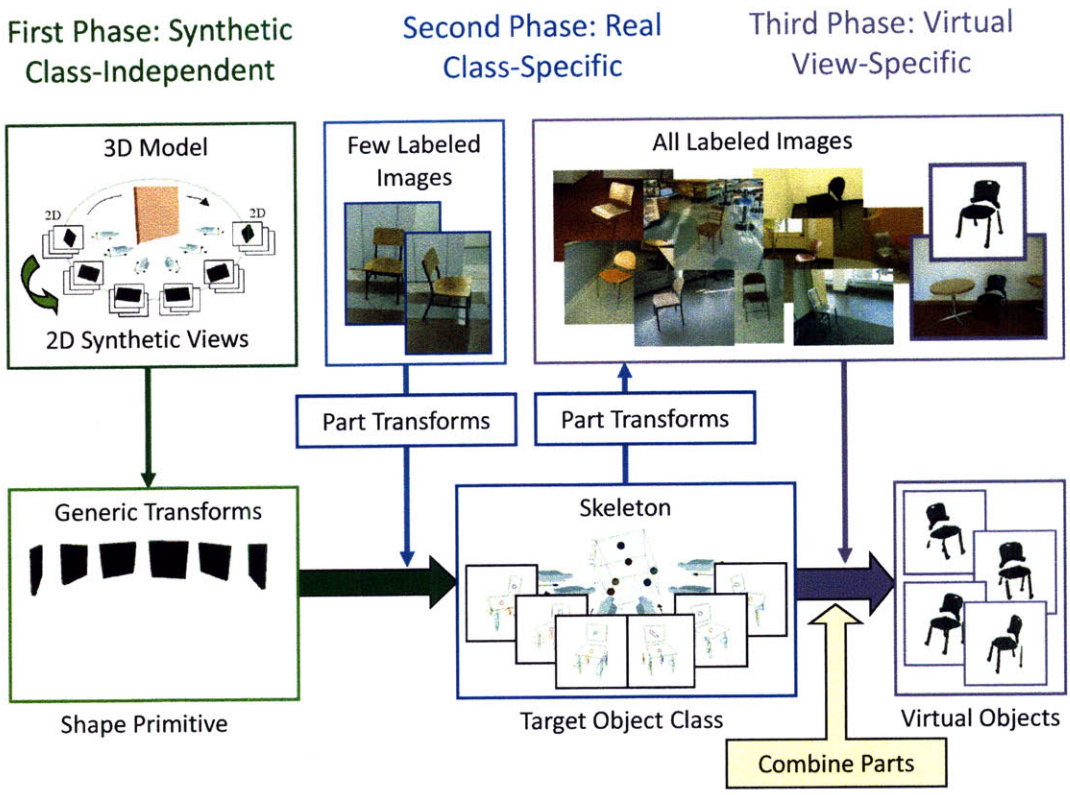

Figure 3-1: The architecture of the basic Potemkin model.

to produce additional images containing predicted views of the parts of the instance from other view bins from which this same set of parts are visible.

3.2 Estimating the Model

Our overall goal is to use the Potemkin model to enable learning from fewer images than would otherwise be necessary. It is crucial, then, that we be able to estimate the model itself from few labeled training images. To enable this, we initialize the view transforms using cheaply available synthetic data, then refine the transforms using the available training images. Similarly, we solve for the skeleton points by pooling the data from all the instances in a view bin, exploiting the assumption that the skeleton is relatively stable across instances of the class.

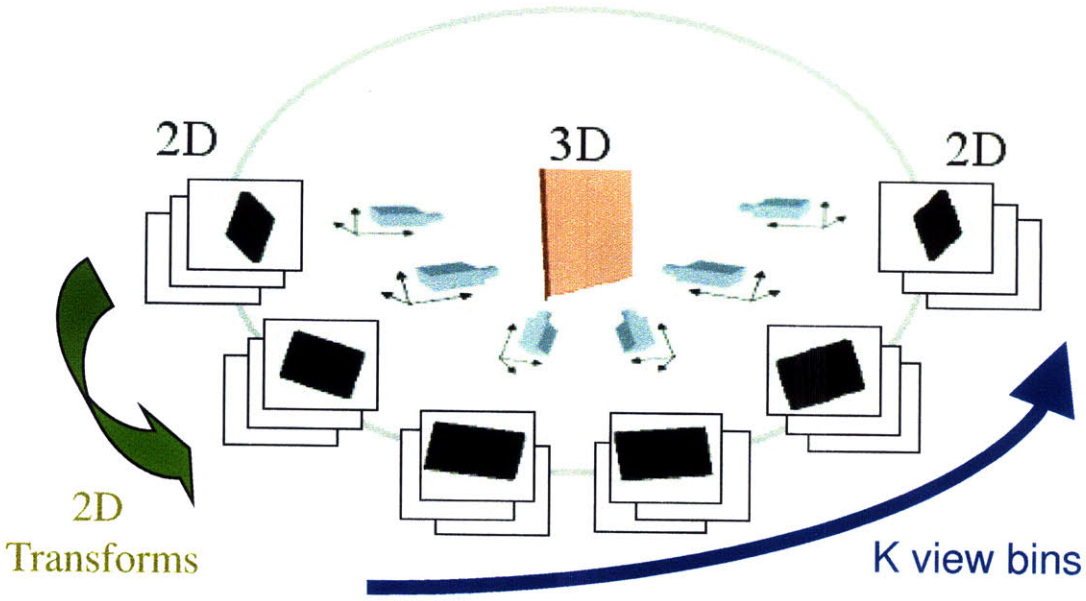

Figure 3-2: A synthetic nearly-planar box object used for learning the generic view transformations.

The view bins are currently selected by hand, but it might be desirable, in future, to use an adaptive quantization method to find a variable-resolution partition of the view space that optimizes model complexity and effectiveness.

The projection matrices P_α, from 3D to view, are dependent only on the choice of view bins, and can be computed analytically for the centroid of the view bin or estimated from data sampled over the whole bin.

The basic Potemkin model is trained and used in three phases (see Figure 3-1). The first two phases are used to estimate the model: one generic, and one object-class specific.

3.2.1 Generic Phase: Learning Generic View Transformations

The generic phase starts by learning a set of generic view transformations, based on a single oriented 3D primitive shape.

The set of generic transforms $T_{\alpha,\beta}$ map points of an image in view bin α to points of an image in view bin β. These transforms are learned from synthetic binary images (one drawn randomly from each of the k view bins) of approximately 30 relatively "flat" vertical blocks of varying dimensions, as depicted in figure 3-2. Note that since we are using synthetic images, we can generate enough data to get a good initial estimate for all the pairwise view transforms.

To learn the generic transforms $T_{\alpha,\beta}$, we begin by finding the boundaries of the object in each pair of images, from views α and β. Then we use the shape context algorithm [48] to obtain a dense matching between the 2D boundaries across the images. Finally, we use linear regression to solve for a 2D projective transform that best explains the observed matches across the set of training image pairs. Note that we are learning from images that vary substantially in pose and viewpoint, so an analytic solution for the best 2D projective transform is not straightforward. We will ultimately use this technique on real images as well, which makes analytic approaches infeasible.

General 2D projective transforms have 8 degrees of freedom and they can be decomposed [31] as

$$T_{\alpha,\beta} = \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{v}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} s\mathbf{R}(\theta) & \mathbf{t} \\ 0^T & 1 \end{bmatrix}, \tag{3.1}$$

with $R(\theta)$ the 2D rotation matrix representing angle θ, $\mathbf{v}^T = [v_1, v_2]$, $\mathbf{t} = [t_x, t_y]^T$, and $\mathbf{K}$ an upper-triangular matrix of the form

$$\mathbf{K} = \begin{bmatrix} K_a & K_b \\ 0 & 1/K_a \end{bmatrix}.$$

In our case, the translation $\mathbf{t} = [t_x, t_y]^T$ is a zero vector because $T_{\alpha,\beta}$ models the shape transformation around the centroid of the 2D projected object from view α to view β. For all synthetic images, the centroid of the object is the origin in the 2D

coordinate system.

The decomposition in (1) is unique if $s > 0$. We can therefore represent each transform as a vector of 6 parameters: $[\log s, \theta, K_a, K_b, v_1, v_2]$. Ideally, the 2D rotation angle θ should be handled specifically to avoid the continuous problem. For example, the mean of two values of θ, such as -179 and 180, may be too different from either of the two values. However, in our case, θ won't be bigger than 90 degree across the view bins we modeled. Thus, we use the mean of the set of parameter vectors estimated from each image pair in the training data as our generic view transform.

3.2.2 Class-Specific Phase: Refining the View Transformations

In the class-specific phase, a collection of real training images of different instances of the class from arbitrary views can be used to estimate the 3D class skeleton of the object class, and to tune the view transforms $T^j_{\alpha,\beta}$ for each of the parts, j, from view α to view β. The outlines of the parts must be labeled in all images used in this phase, but reasonable results can be obtained using only two such images.

If the real training data set happens to contain more than one view of a particular instance, in different view bins, then that data can be used to refine the generic view transforms, making them specific to the particular parts of the class. If it does not, the model can still be used with the original generic transforms.

Assume we are given a set of label-image pairs $\langle x^i_\alpha, x^i_\beta \rangle$ with views of the same instance in bins α and β. For each part j in each image pair, i, we use the shape-context algorithm to match points on the boundaries of the part and then construct the transform $\hat{T}^{ij}_{\alpha,\beta}$, represented as a vector of 6 parameters, as above.

We then combine the generic view transform learned in phase 1 with the part transforms estimated from each image pair to obtain a part-specific view transform $T^j_{\alpha,\beta}$:

$$T^j_{\alpha,\beta} = \frac{1}{\kappa + m} \sum_{i=1}^{m} \hat{T}^{ij}_{\alpha,\beta} + \frac{\kappa}{\kappa + m} T_{\alpha,\beta} \, . \tag{3.2}$$

The relative weighting of the generic and specific transforms is determined by the "pseudo-count" parameter κ, which is currently chosen empirically as 9 and seems to be well behaved in practice.

3.2.3 Class-Specific Phase: Learning the Class Skeleton

The class skeleton can be estimated from any collection of real training images of instances in any view, and does not require multiple views of any single instance. We directly compute the centroids of the individual parts from the training label images. We then use these to estimate the skeleton, using the projection matrices P_α, which specify the projection from 3D points into 2D view bin α.

We begin by aligning and scaling the training images in each view bin so that the object bounding boxes are all aligned. Next, for each view bin α and part j, we compute the mean μ^j_α and covariance Σ^j_α of the coordinates of the centroid of part j in the normalized images in bin α.

For each part, we find the 3D position whose 2D projections minimize a sum of weighted distances to the observed 2D mean centroids in each view. The weighted distances are the Mahalanobis distances with respect to the observed covariances in each view. The minimizing 3D location S_j for part j in the skeleton is given by [30]:

$$S_j = \left(\sum_\alpha P_\alpha^T (\Sigma^j_\alpha)^{-1} P_\alpha \right)^{-1} \left(\sum_\alpha P_\alpha^T (\Sigma^j_\alpha)^{-1} \mu^j_\alpha \right) \, , \tag{3.3}$$

The skeleton locations for each part are estimated independently, because we have no prior on the structure of a new object class.

Figure 3-3 shows a schematic version of this process. In each view bin, the distri-

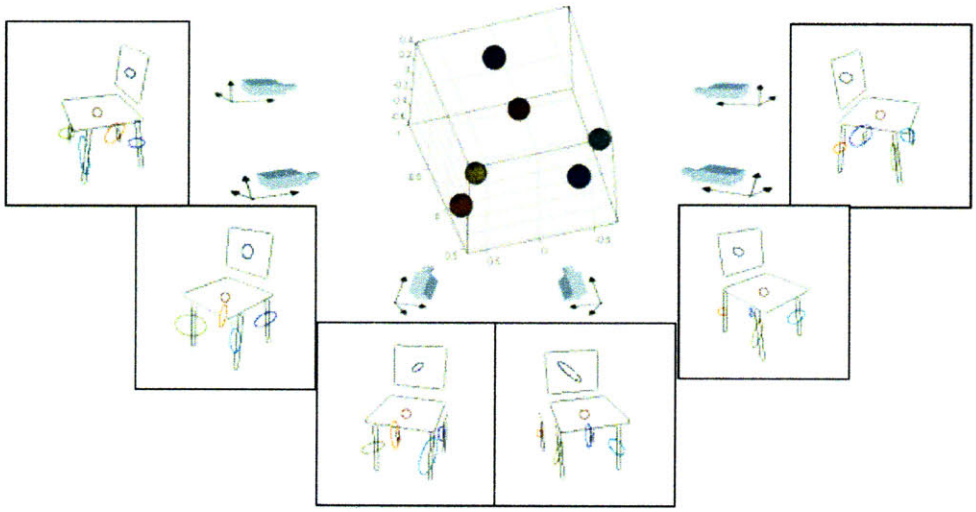

Figure 3-3: The ellipses indicate the distribution of 2D positions of the centroids of each part in the training set of chair images for each viewpoint. The 3D spheres show the estimate of the skeleton positions S_j for each part. The size of the spheres is not significant.

bution of the 2D centroid of each part is estimated, shown by the ellipses. Then, the 2D centroid distributions are used to estimate the 3D skeleton locations, shown in the center. The centroids of parts that are sometimes occluded by other object parts have higher variance. As we consider more complex objects from a larger variety of viewpoints, we may have to introduce explicit reasoning about occlusion into this modeling step.

3.3 View-Specific Phase: Using the Model to Generate Virtual Images

Finally, we can use the Potemkin model to generate "virtual" training data for a set of k view-specific 2D detectors. Any training instance in one view bin can be transformed to many other view bins, using the skeleton and the view transforms, and used as training data for that detector. This strategy effectively multiplies the value of each training image, and allows us to train detectors for view bins that have

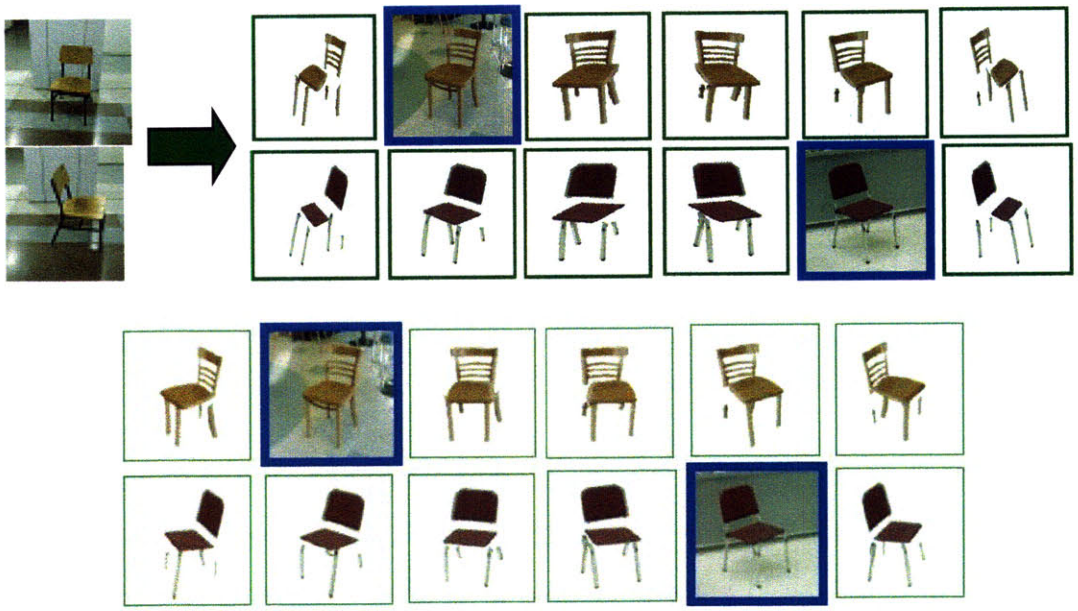

Figure 3-4: The basic Potemkin model in the top two rows is constructed from only two real labeled images (left). Given the label images for two object instances, each in only one viewpoint bin (highlighted), the other virtual views are generated from the basic Potemkin model. The model in bottom two rows is constructed from a total of 10 real images, from 5 object instances over 6 views.

never actually been seen, based only on virtual data.

Given an input image and associated label image, indicating which pixels correspond to which parts, in view bin α, for each viewpoint $\beta \neq \alpha$ for which the same set of parts are visible, we

- transform the pixels belonging to part j using $T^j_{\alpha,\beta}$, then
- center the resulting shape at $P_\beta S_j$, the view projection of the 3D skeleton location of part j into view bin β.

This process generates a complete virtual label for view β; in addition, it generates a virtual image with pixels corresponding to the object, which can be easily combined with the background of the original image to generate a complete image in the new view.

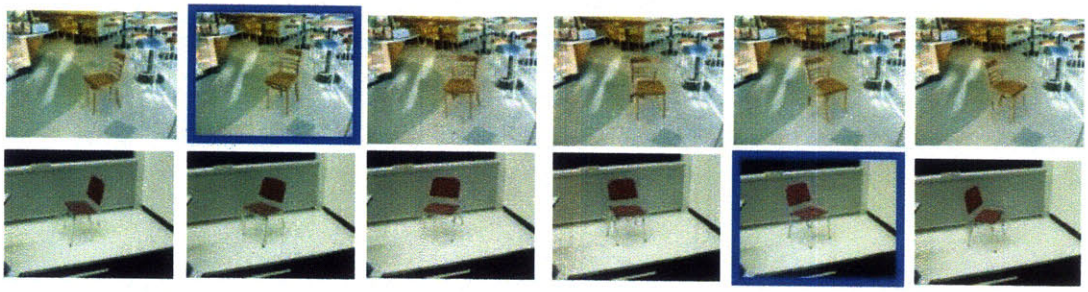

Figure 3-5: Full virtual images and the actual input images (bold border) of two object instances in the chair data set.

An example of the results of this process can be seen in figure 3-4 (top two rows), which shows the virtual images constructed from a basic Potemkin model, trained with only two real labeled images. Note that at least two images are necessary to solve for the skeleton positions in 3D. The results are already useful for detection, as shown in Chapter 4. As the amount of training data is increased, the quality of the transformed images improves. Figure 3-4 (bottom two rows) shows the results after a total of 10 training images; the transforms and skeletons are learned well enough to make credible virtual training examples for all view bins given a single label image. We constructed a background image by filling the labeled area of the given single image with generated texture and color drawn from the surrounding image data. Then the virtual examples can be placed into the background image to construct a realistic image suitable for using in any existing detection system as shown in Figure 3-5.

3.4 View-Specific 2D detection

The Potemkin model can be used to generate transformed versions of training data, both images and labels, so we can take any multi-view image corpus and increase its coverage by transforming every image into multiple viewpoints. The resulting corpus can be used to train view-specific 2D object-class detectors [22, 24, 10, 11]. We have tested the effectiveness of the basic Potemkin images in a detection task by using them as input to the method developed by Crandall et al. [10] (see section 3-5). We

have also tested localization performance using a closely related method we developed but which uses a somewhat richer model of the appearances of individual parts.

3.4.1 EigenEdgeStar (EES) Model

A 2D pictorial-structure model consists of a *shape* model that describes the spatial relations between the parts and an *appearance* model for each part, that describes its appearance in the image. We have developed a specific instance of this class of models [10], called the EigenEdgeStar model. In it, the shape model is a star, or Gaussian 1-fan model, meaning that a particular part is chosen as the reference part, which establishes a 2D reference frame, and then the location of each other part j is specified with a Gaussian distribution over its position in the frame, using parameters μ_j^s and Σ_j^s.

We assume that appearances of the individual parts are independent of one another and of their respective positions; the model for each part p_j is represented as a probability distribution over a rectangular window of pixels, $\Pr(I_w|p_j)$, with the dimensions of the window potentially different for each part model.

Let $e_{x,y}$ be a vector representing the edge response (strength and orientation) at pixel x, y in in the window. Edge strength is represented as a positive value between 0 and 1. We represent edge orientation θ (which ranges from 0 to 180°) using $\sin 2\theta$ and $\cos 2\theta$. So, $e_{x,y}$ is represented as a vector of three values. The contents of the window, e_w, can then be represented as a concatenation of the $e_{x,y}$ vectors for all pixels in the window.

Even for a relatively small window, representing a full joint Gaussian distribution over the space of e_w would require an enormous number of parameters (quadratic in the number of pixels), which would then require an unreasonably large amount of training data to estimate. In pictorial structures, it has been traditional to represent the distribution as a fully factored distribution over the edge orientations and

strengths of each pixel within the window. We have found that it is more robust to model some dependence among pixels within the window.

To reduce the size of the model and the data requirements, we project the N-dimensional e_w vectors down into an n-dimensional space, for n much less than N, and represent a joint Gaussian over vectors in that space. So,

$$\Pr(I_w|p_j) = \Pr(e_w|p_j) = \Pr(V_j e_w|p_j) \sim \mathcal{N}(\mu_j^a, \Sigma_j^a)$$

where V_j is an $n \times N$ matrix, μ_j^a is the n-dimensional mean vector, and Σ_j^a is the $n \times n$ covariance matrix.

3.4.2 Training

The EES model is trained on images that have been labeled with the outlines of each part. Bounding boxes for the root part are calculated and the images are scaled so that the root parts are aligned as well as possible. Now, the shape model is trained simply by maximum-likelihood estimation of each μ_j^s and Σ_j^s from the positions of the centroids of part j in each scaled training image, relative to the centroid of the root part.

To train the appearance models for the parts, we use the edge images that constitute the labeled boundary for each example part. We begin by estimating a bounding box for all the training examples, which defines the dimensionality of the pixel window for the part. Then, we compute the vector e_w for each example, estimating local orientations of the edges in the outlines given in the label. The e_w vectors are the input to a PCA algorithm, resulting in V_j. Finally, the set of transformed vectors, $e'_w = V_j e_w$, is used to generate the maximum-likelihood estimates of μ_j^a and Σ_j^a.

3.4.3 Detection

Now, given a trained EES model and an image, we would like to *localize* an instance of the object class in the image. That is, to find the collection of part locations, $l_0 \ldots l_k$, that maximize $\Pr(l_0 \ldots l_k | I)$, the probability that there is an object of the class of interest at those locations in the image I. This is proportional to $\Pr(I|l_0 \ldots l_k) \Pr(l_o \ldots l_k)$, which is proportional to $\Pr(I|l_0 \ldots l_k) \Pr(l'_1 \ldots l'_k|l_0)$, where l'_j is the relative location of part j with respect to l_0, because we have no bias on the location of the root part l_0 in the image. The first term can be factored into a product of the probability of the images pixels in the window for each part, times a background model for the rest of the pixels, which we will neglect because it is the same for all location hypotheses. The second term can be factored because of the 1-fan assumption. So, we seek the $l_0 \ldots l_k$ that maximize

$$\sum_{j=0}^{k} \log \Pr(e_w(l_j); \mu_j^a, \Sigma_j^a) + \sum_{j=1}^{k} \log \Pr(l'_j | l_0; \mu_j^s, \Sigma_j^s) \quad ,$$

where $e_w(l_j)$ is the edge information for the window centered on location l_j.

We begin by running the Canny edge detector [6] on the image, and computing, for each part j, $\Pr(e_w(l_j); \mu_j^a, \Sigma_j^a)$ for each candidate location of the part in the image. Then, we maximize the overall criterion using the efficient dynamic-programming method of the distance transform [22]. This gives us the best set of part centroids, which allows us to hypothesize the location of the object in the image as a set of bounding boxes of its parts. Note that in next section the scale of the object instance is assumed known and therefore detection is done at a single scale; the search could readily be expanded across scale.

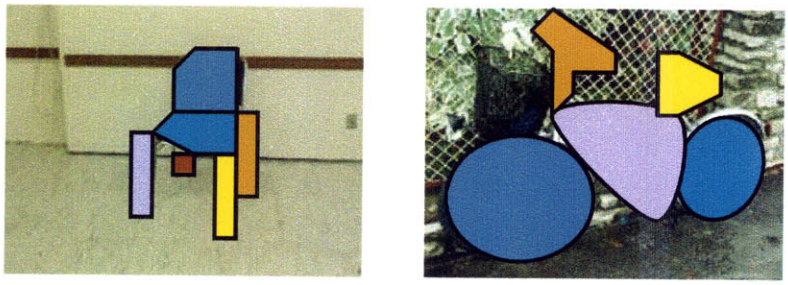

Figure 3-6: The 6 parts of a chair and the 5 parts of a bicycle.

3.5 Experimental Results

We tested the impact of virtual training images on localization accuracy of objects within images using the EES approach and in detection tasks using the system by Crandall et al [10]. We found that the virtual training images improved the performance of both systems.

We used two different object classes for testing: four-legged chairs and bicycles (figure 3-6). The four-legged chair dataset was collected and labeled by our research group. It contains 140 images of 80 different object instances, at a variety of viewpoints ranging over about 30° in elevation and 160° in azimuth. There is considerable variation in lighting, location of objects within the image, and clutter. We discretized the range of viewpoints into six bins, as seen in figure 3-3. The bicycle data set is the TU-Graz-02 database, which is part of the PASCAL Visual Object Classes (VOC) Challenge [20]. It contains 365 images of bicycles, again with considerable variation in pose, view, and clutter. Most of the images were taken from four general poses, and so we use those as our view bins for this data set.

3.5.1 Localization Performance

All the localization experiments reported here are on test sets of instances from a single class but with views drawn from the full range of viewpoint bins for the class, six for chairs and four for bicycles. The difference among the experiments is in the

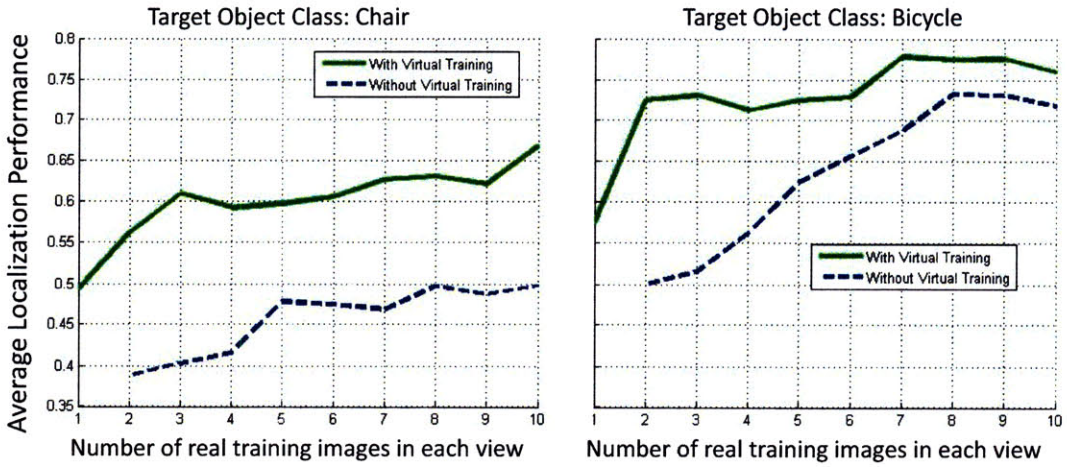

Figure 3-7: Localization performance of EES for chairs and bicycles, using all the training data for all views.

training sets. The detection system computes a score for each of the 2D models corresponding to each view bin and picks the one with the best score. Localization performance is measured by the percentage of the bounding box of the localized object that overlaps the ground-truth bounding box in the image.

To test the benefit of using the basic Potemkin model to generate virtual training data, we first compare localization performance when using only the training images for each view to the performance using the training images augmented by the virtual training data from the basic Potemkin model.

Figure 3-7 shows the change in performance as the number of training images in each view bin goes from 1 to 10. Note that we typically use three principal components to characterize the appearance of the parts, but if there are fewer than 4 training image for view bin, we reduce the number of components to be the number of images minus 1. This is only an issue when working without virtual data; the approach that uses virtual data has a training instance in each view bin for each image in all of the view bins. In fact, with virtual training data, we can usually make predictions for viewpoints for which we have no real training data at all; we explore that case below.

The performance when using virtual training data on chairs shows substantial improvement over the no virtual data case. Since the shape of chairs is so variable,

Figure 3-8: Successful localizations of chairs and bicycles using EES.

this type of detection system requires a substantial number of training images. When using virtual training data, each viewpoint has roughly six times the amount of training data than the case without virtual data. The increase in performance also attests to the quality of the virtual data. In the case of bicycles, the use of virtual examples produces a significant, but somewhat less dramatic improvement. However, the overall localization accuracy for bicycles is higher than for chairs, given the same amount of training data. This is not surprising since the variability in the shapes of bicycles is small and good performance is reached with little training data.

Figure 3-8 shows some correct localization results on chairs and bicycles.

3.5.2 Localization Results on Previously Unseen Viewpoints

As we pointed out above, using virtual data, it is possible to localize objects seen from a previously unseen viewpoint. We measured the system's performance by varying the number of view bins represented in the training set, while always having all views represented in the test set.

Figure 3-9 shows localization performance for the chairs dataset when the number of view bins with real images varies, each with 10 training images. The lower and upper lines are the performance for the case we saw earlier where every view bin has

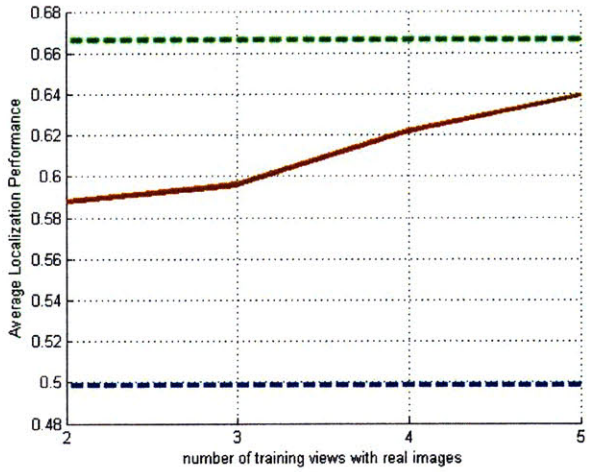

Figure 3-9: Localization performance of EES for chairs as a function of the number of views in the training data.

real data, with the top line using virtual training data and the bottom line not. These results illustrate that the quality of the virtual data is comparable to that of the real data.

3.5.3 Detection Results

We have also carried out detection experiments using the system developed by Crandall et al [10]. We used their implementation and did training using a combination of real and virtual images for the chair and bicycle data sets. Figure 3-10 (top-left) shows ROC curves for the chair data set. The task was to detect the presence of a chair in a known viewpoint versus background images of office environments.

Good performance on this task requires a substantial number of training images; the performance with 50 real training images is much worse than that with 100 real training images. The performance with 30 real training images and an additional 100 virtual images obtained from other viewpoints is intermediate between the performance with 50 and 100 real training images. In all cases, we used 55 chair images and 55 background images for testing.

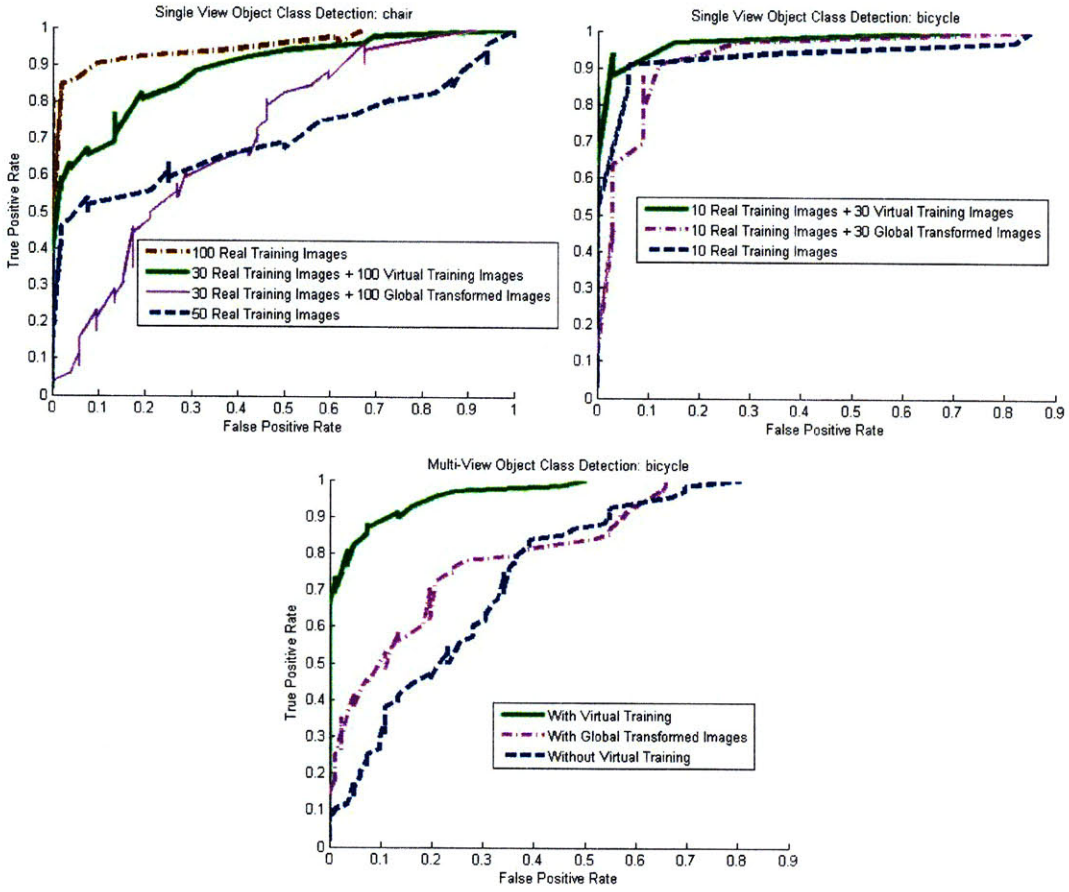

Figure 3-10: ROC curves for single-view detection (top). ROC curves for multi-view bicycle detection (bottom).

As a control, we also tested the effectiveness of using a single global transform for the whole object instead of the basic Potemkin model with multiple parts. The images using the basic Potemkin model were more useful for training.

By way of contrast, for single-view detection of the bicycle class (which is much simpler than the chair class and nearly planar) there is very little improvement in detection performance; see Figure 3-10 (top-right). In all cases, we used 35 bicycle images and 35 background images of street scenes for testing.

We also compared the effect of virtual images in a multi-view detection task: deciding whether any of 4 viewpoints of a bicycle is present in an image (Figure 3-10:bottom). This is done by training 4 independent classifiers and comparing the

strongest response to the learned threshold value. There are 10 real training images for each viewpoint, leading to 30 additional virtual training images per viewpoint. There are 85 test images distributed among the 4 viewpoints and an additional 83 background images.

In this more difficult task, the basic Potemkin virtual images are quite helpful. Surprisingly, the images derived from a single global transformation are much less helpful, even for the nearly planar bicycle. Although the globally transformed images look realistic, the individual parts are not consistent with the appearances of the parts in the real images. The basic Potemkin model refines the transforms for individual parts and does a better job of predicting the appearance of parts across distant viewpoints.

Throughout these tasks, we have demonstrated the effectiveness of the basic Potemkin model in reducing training data requirements in object class detection. The basic Potemkin model can be efficiently learned from few images and can be used to generate virtual training data for a wide range of viewpoints. It circumvents one of the key roadblocks to effective multi-view detection, namely the need for extensive training data in all the viewpoints of interest.

Chapter 4

The Generalized Potemkin Model

In this chapter we describe the generalized Potemkin model, and how it can be trained and used. We extend the basic Potemkin model to use a basis set of oriented primitive shapes, and select among them to represent each of the parts of the target class based on a pair of initial training images of the class (Section 4.1). We also provide a method for self-supervised labeling of parts in the training data (Section 4.2). This results in substantially improved quality of virtual training data over the basic model. We conduct experiments showing that these virtual examples are comparable to real examples for training view-dependent detectors, and demonstrating these extensions significantly outperform the basic model (Section 4.3).

4.1 The Use of Multiple Primitive Shapes

When a basic Potemkin model for an object class is built from only one real training image-pair of the same instance, the part-specific transforms default to the generic view transforms for an upright block.

An example of results under these minimal training conditions is shown in figure 4-1. This process can in some cases be very effective, but in other cases the transformed

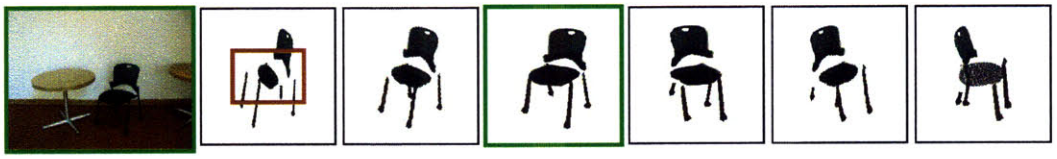

Figure 4-1: Virtual views generated from the highlighted image using a basic Potemkin model that is minimally trained.

parts are distorted in the new view. The problem highlighted by the red rectangle in figure 4-1 arises because the model is relying entirely on the generic view transform, which was learned for a vertical flat block. But, because the seat of the chair is actually horizontal, the generic transform does a very poor job of predicting pixels in the new view.

To address this problem, we expand the set of possible primitive 3D part shapes, with associated generic view transforms, available to the Potemkin model. Given a new object class, using just two views of one instance to select an appropriate shape primitive for each part allows the model to make much better predictions of appearance in unseen views.

With this extension, the Potemkin model can be generalized to rigid objects which are constituted with these 3D primitive shapes. However, if the shapes of objects are different from the primitives we used, this generalized Potemkin model can not predict accurate appearances of the transformed objects.

4.1.1 Oriented Shape Primitives

We will augment the basic Potemkin model with an *oriented primitive* model for each part. Each shape primitive is a simple 3D box at one of a small number of possible 3D orientations with respect to the frame of the skeleton. As shown in figure 4-2, we consider two rotational degrees of freedom: *azimuth* and *elevation* angles of rotation about the shape's centroid. This allows us to ignore irrelevant in-plane rotations of nearly planar shapes and rotations about the long axis of nearly linear shapes. It

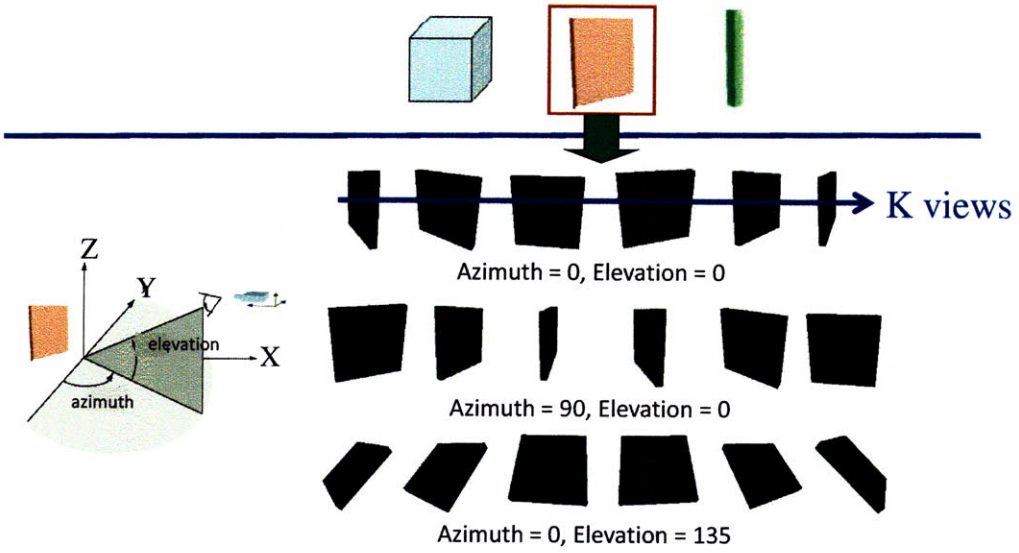

Figure 4-2: The top row of the figure shows the three 3D primitive shapes: cube, flat block, and stick. Each of these shapes is considered at several orientations with respect to the skeleton. The lower part of the figure shows, for the flat shape, several views of the shape at three example (azimuth, elevation) orientations.

does, however, limit the possible set of orientations of more general shapes.

We select the primitive shape and its orientation with respect to the skeleton frame jointly from M possibilities. For each oriented primitive m and each view bin, we generate a set of synthetic images. First, we generate a 3D shape that is a random minor variation on the 3D shape primitive in size and aspect ratio and place it at the specified orientation. Then, from each view bin, we select a view uniformly at random, and create a 2D image by projecting the 3D instance according to the chosen view. This gives us a set of images, one from each view bin, of each of a set of instances of each oriented primitive.

Now, we apply methods from the basic Potemkin model to use pairs of these images to estimate Mk^2 *primitive view transforms*, $U^m_{\alpha,\beta}$, one for each pair of view bins α, β and oriented primitive m. In addition to computing the mean transform, as was done in the basic Potemkin model, we also estimate the standard deviation, $\sigma^{U^m}_{\alpha,\beta}$. Note that the variance in the transform is modeling uncertainty due to variation in

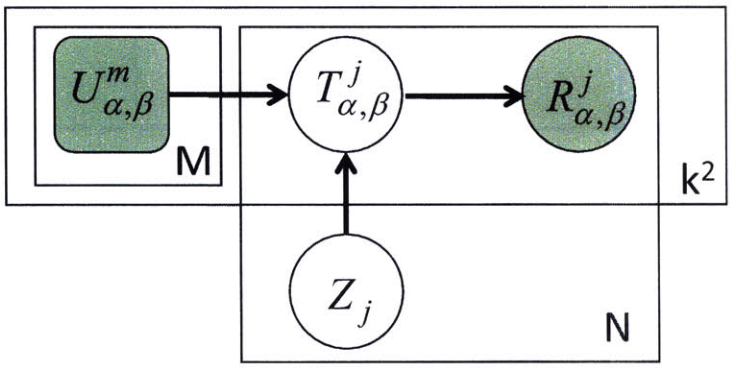

Figure 4-3: Generative model for 2D part-specific transforms. There are plates for the M possible oriented primitives, for the N parts of the object model, and for the k^2 view bin pairs. $U^m_{\alpha,\beta}$ and $T^j_{\alpha,\beta}$ represent the view transform from view bin α to β for oriented primitive m and part j respectively. $R^j_{\alpha,\beta}$ represents an actual observed transform of part j from view bin α to β in real data. Z_j is an indicator variable that selects which primitive m is the appropriate shape for part j of the object.

the shape primitive as well as the particular views chosen from the view bins.

4.1.2 Selecting Primitives

To learn a new object-class model, we need to select an oriented primitive for each part. Further, we might want to find a "basis set" of primitives that can be used, collectively, to represent the parts of a large number of object classes. This basis set could then be used as a restricted domain of primitives for constructing new class models, thereby making the process more efficient, and preventing the tendency to overfit when there are a huge number of primitives available.

We will approach the problem of selecting primitives by developing a latent-variable model for 2D part-specific transforms; the graphical model is shown in figure 4-3. In this model, there are plates for the M possible oriented primitives, for the N parts of the object model, and for the k^2 view bin pairs. We can think of the variable Z_j as an unobserved indicator variable that selects which oriented primitive m is the appropriate shape for part j of the object. Thus $Z_j \sim Multi(1/M, \ldots, 1/M)$

has a uniform distribution over the discrete set of possible oriented primitives.

The observed variables $U^m_{\alpha,\beta}$ represent the mean and standard deviation ($\bar{U}$ and σ^U) of the generic transform for oriented primitive m from view bin α to view bin β.

The variable $T^j_{\alpha,\beta}$ represents the the actual view transform of part j from view bin α to β. We treat it as normally distributed, according to the generic transform distribution associated with its primitive part. Thus, for a choice of $Z_j = m$,

$$T^j_{\alpha,\beta} \sim \mathcal{N}(\bar{U}^m_{\alpha,\beta}, \sigma^{U^m}_{\alpha,\beta}) \; . \tag{4.1}$$

Finally, the variable $R^j_{\alpha,\beta}$ represents an actual observed transform of part j from view bin α to β in real data. In general, we might have many such observations, in which case we would have a plate for R; but one will illustrate the inference process. The observed transforms are modeled as normally distributed about the true transform, with fixed variance set by hand:

$$R^j_{\alpha,\beta} \sim \mathcal{N}(T^j_{\alpha,\beta}, \sigma_R) \; . \tag{4.2}$$

So, given this model, our goal is to infer, for each part j, the maximum *a posteriori* probability (MAP) set of view transforms T^j; that is, $\arg\max_{T^j} \Pr(T^j|U, R^j)$. This model is completely independent for each part, so they can be maximized separately. Because we are ultimately interested in the MAP assignment of primitives to parts, as well as the transforms, we will seek

$$\arg\max_{T^j, Z_j} \Pr(T^j, Z_j|U, R^j) \; .$$

For any particular choice of $Z_j = m$, the posterior mean on $T^j_{\alpha,\beta}$ for any pair of

view bins α and β that have observed samples $R^j_{\alpha,\beta}$, can be computed [27] as:

$$\bar{T}^{j,m}_{\alpha,\beta} = \frac{\kappa_0}{\kappa_0+1}\bar{U}^m_{\alpha,\beta} + \frac{1}{\kappa_0+1}R^j_{\alpha,\beta} = \frac{\frac{{\sigma_R}^2}{\sigma^{U^m 2}_{\alpha,\beta}}}{\frac{{\sigma_R}^2}{\sigma^{U^m 2}_{\alpha,\beta}}+1}\bar{U}^m_{\alpha,\beta} + \frac{1}{\frac{{\sigma_R}^2}{\sigma^{U^m 2}_{\alpha,\beta}}+1}R^j_{\alpha,\beta}\ . \tag{4.3}$$

The weight parameter κ_0 could be determined as a function of σ_R and $\sigma^{U^m}_{\alpha,\beta}$, but, since σ_R was already being selected arbitrarily, we experimented informally with values of κ_0, and set it to 9 throughout results reported in this paper.

To find the MAP value of Z_j, we enumerate possible values m, and select the one for which

$$\Pr(\bar{T}^{j,m}, Z_j = m|U, R^j) \propto Pr(R^j|\bar{T}^{j,m})Pr(\bar{T}^{j,m}|U^m, Z_j = m)Pr(Z_j = m) \tag{4.4}$$

is maximal; because the distribution on Z_j is uniform, this means that we can select the m that maximizes $\prod_{\alpha,\beta}\Pr(R^j_{\alpha,\beta}|\bar{T}^{j,m}_{\alpha,\beta})\Pr(\bar{T}^{j,m}_{\alpha,\beta}|U^m_{\alpha,\beta})$, which is straightforward to compute.

4.1.3 Learning and Using an Object Class Model

Training the generalized Potemkin model is similar to training the basic Potemkin model, but it requires some new steps. In the first phase, constructing the generic transforms relating 2D shapes across each pair of views is essentially unchanged, except that it is done for each of the oriented primitives.

We begin the second phase by using any available paired instances to select the most appropriate oriented primitive for each part of the class, and to estimate the associated view transforms, as just described.

We can take the additional useful step of estimating a rough extent and planar model of the part in 3D. We find a similarity transform between the actual part outlines and the projections of the primitives into views α and β; then, already having

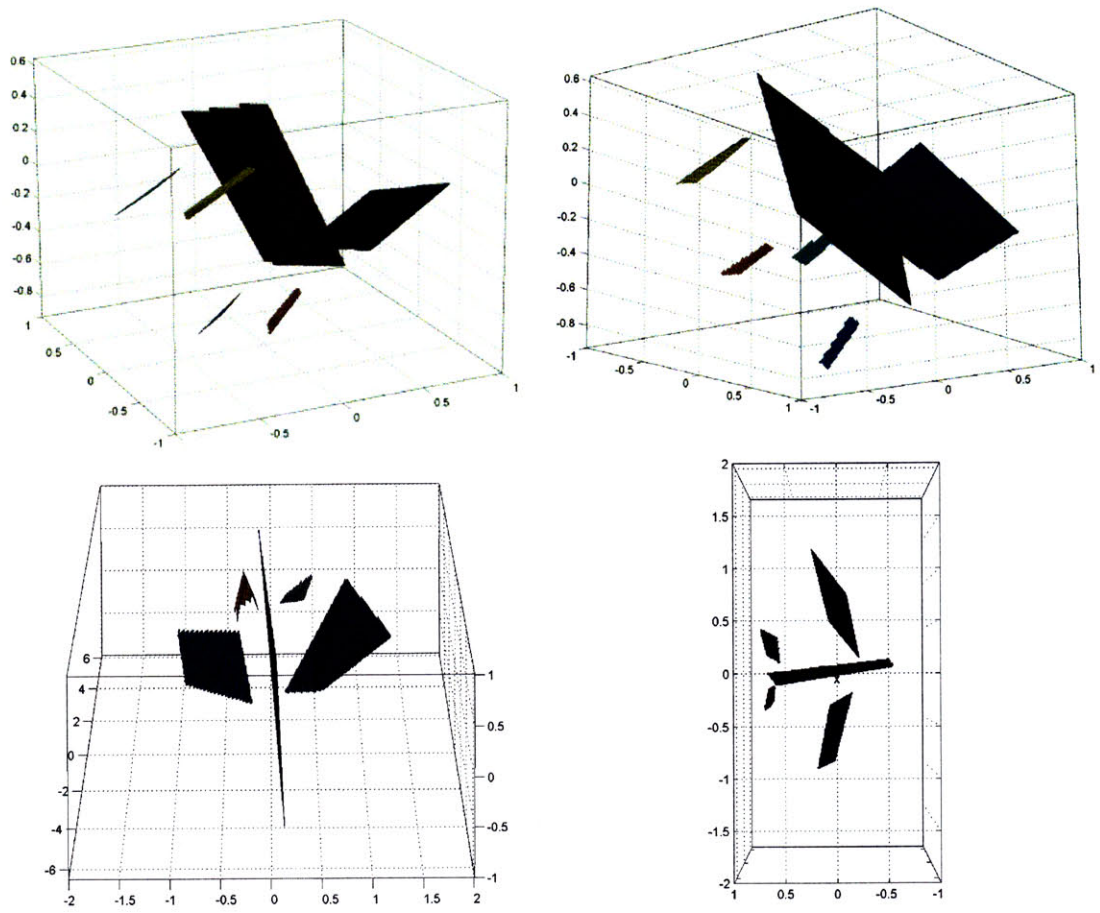

Figure 4-4: Visualization of 3D skeleton and shape primitives. Top: two views of the four-legged chair model; bottom: two views of the airplane model. Note that these models were each constructed from two part-labeled images of the same object, knowing the view bins but with no further camera calibration.

computed correspondences between the outlines of the projections of the primitives in phase 1, we can solve for 3D positions of points on the outline of the shape.

The 3D position of the centroid of each part is estimated as for the 3D skeleton of the basic Potemkin model.

Figure 4-4 shows the skeleton and shape primitives estimated from two part-labeled images. These easily-obtained 3D class models are not sufficiently detailed to be used directly for detection, but they provide important information about self-occlusion. When transforming an input image to a novel view, it often happens that pixels belonging to different parts in the original image are transformed to the

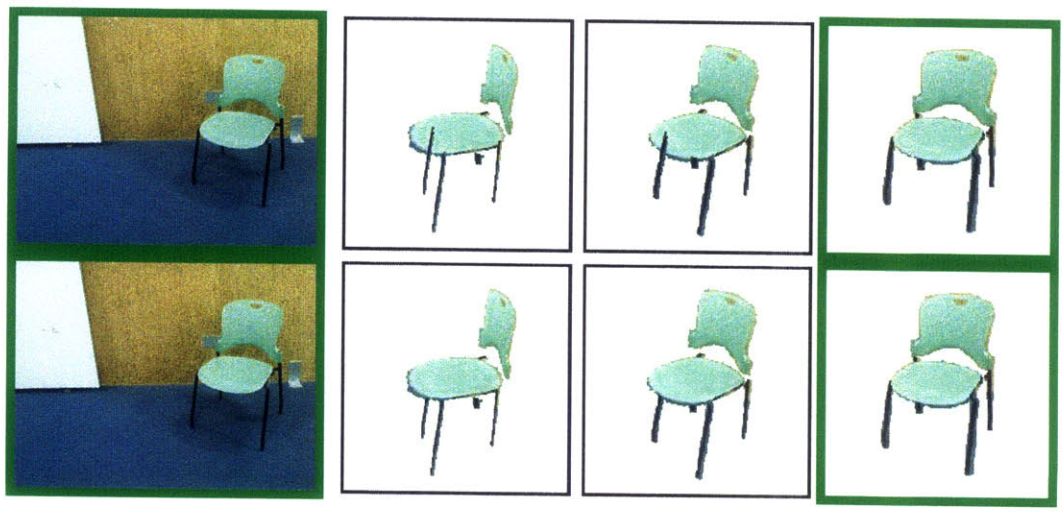

Figure 4-5: First row: virtual images generated from highlighted instance without occlusion handling; second row: with occlusion handling.

same point in the new view. In that case, we use the 3D class model to determine which part is in front, and select those pixels to be painted. This strategy of self-occlusion handling works when there is a clear order in depth for the parts in the target viewpoint. Figure 4-5 shows how understanding self-occlusion can improve the transformed images.

By increasing the set of primitives and supporting reasoning about occlusion, the generalized Potemkin model generates significantly more realistic virtual images compared to those generated by the basic Potemkin model, as shown in figure 4-6. Note that in both cases, the models are trained with only two real labeled images.

4.2 Self-Supervised Part Labeling

The training algorithm for the Potemkin model requires the viewpoints of the objects and the outlines of the parts in all real training images to be labeled. While viewpoint labels and object outlines are readily available, from sources like LabelMe [59], part-labeled images are not.

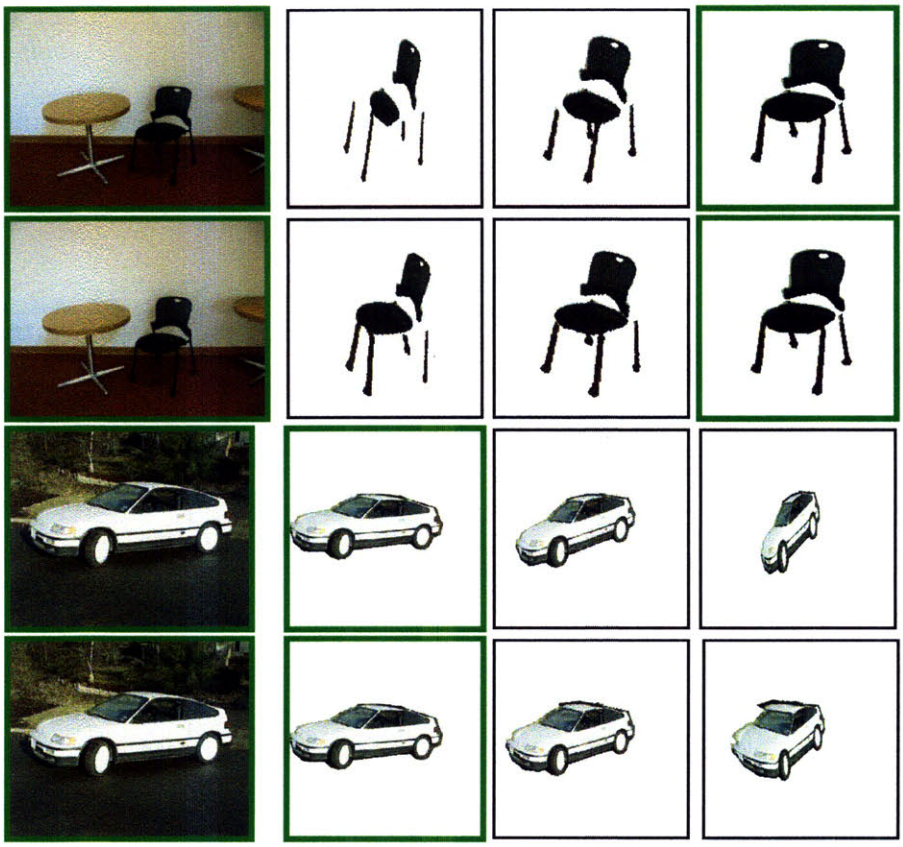

Figure 4-6: The generalized Potemkin model (second and fourth rows) generates more realistic virtual images than the basic Potemkin model (first and third rows).

We have developed a simple approach that obviates the need for most of this labeling; we require hand part-labeling on only a few images of the same object to build the 3D class model and on one image of any instance for each additional view bin *from* which images will be transformed.

Our approach is based on the fact that objects in the same class, seen from the same view, have similar arrangements of parts. For each view of the class, we select one instance and hand-label its parts. Then we use the shape context algorithm [48] to match the boundary shapes and deform the boundaries of the labeled instance into the unlabeled instances, as shown in figure 4-7. The deformed part boundary encloses a region in the new image, which is labeled as the corresponding part.

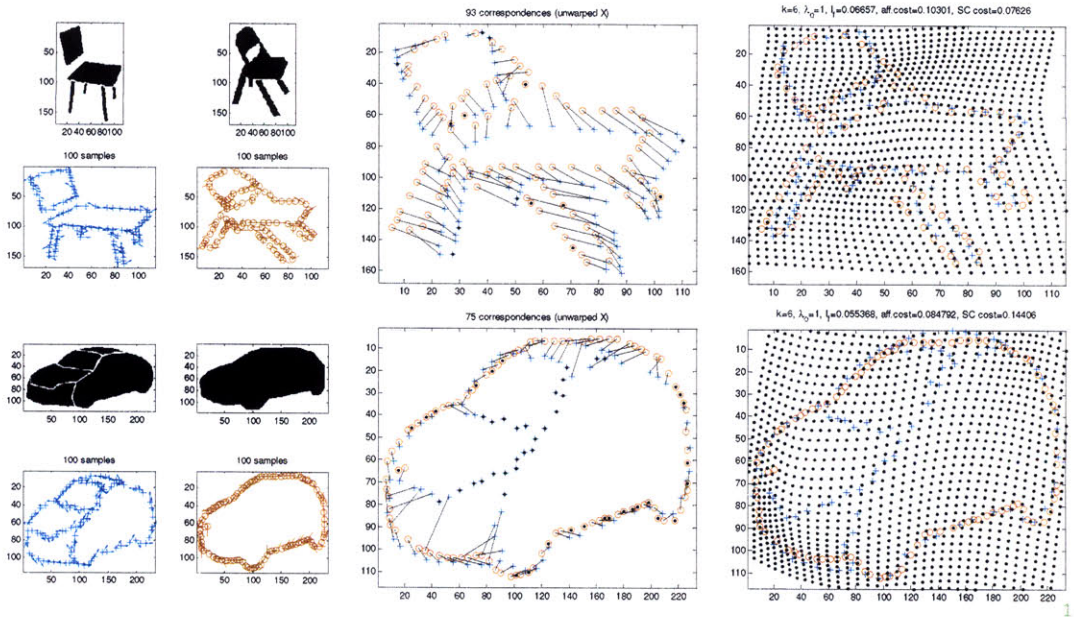

Figure 4-7: Given a model instance with labeled parts (blue), the parts of another instance (red) in the same view can be found by matching points along the boundaries of the instances (middle) and by deforming the model instance into the target instance (right).

4.3 Experimental Results

In this section, we report a number of experiments aimed at evaluating different aspects of the performance of the generalized Potemkin model. We also demonstrate that the virtual training examples generated by the generalized Potemkin model can be used effectively to train an existing view-dependent detection system [10], with the result that many fewer real training images are required to reach the same level of performance.

4.3.1 Dataset

We used four different object classes, shown in figure 4-8, for our experiments. We described chairs using six parts, bicycles using five parts, airplanes using five parts, and cars using five parts. The four-legged chair data set and the airplane data set were collected by our research group. These two data sets contain 432 images and

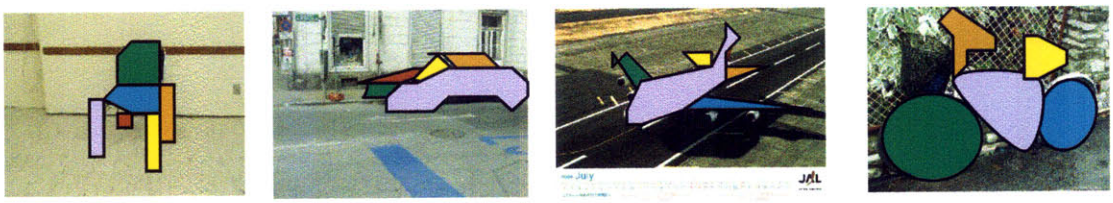

Figure 4-8: Decomposition of object classes into parts.

Figure 4-9: Four selected view bins for bicycles, cars, and airplanes.

262 images respectively. The chair data are discretized into six view bins (figure 4-1) and the other data sets into four (figure 4-9). The bicycle data set is from the TU-Graz-02 database, which is part of the PASCAL Visual Object Classes (VOC) Challenge [20]. It contains 365 images of bicycles. The car data set contains 243 images, some collected by our research group and some taken from the car database of the VOC Challenge. Images in all four data sets have considerable variation in pose, view, and clutter. Note that in this paper the scale of the object instance is assumed to be known. Outlines of objects in all images were labeled by users of LabelMe [59] and normalized based on object scale We also collected four different background data sets—indoor scenes, street scenes, sky scenes, and road scenes—which were used to evaluate detection performance for the corresponding object classes.

4.3.2 Basis Set of Primitives

We began by conducting an experiment to see whether there was a small "basis set" of primitives that would effectively model all four of our object classes, which together have 21 separate parts. For this experiment only, we provided one part-labeled real image of a single instance of each object class in each of its canonical views, which we used to generate one observed transform $R^j_{\alpha,\beta}$ for each part and view pair.

We used a set of oriented primitives generated from three different types of 3D shapes: flat blocks, sticks, and cubes. We considered flat blocks and sticks at each of eight possible orientations (four elevation angles and two azimuth angles) and cubes at two different orientations (two azimuth angles), yielding a set of 18 possible oriented primitives. For each primitive, we generated 30 slightly varying instances, and then for each view bin we selected a view and rendered a 2D image. Now, for each pair of view bins, we computed the transforms between the image-pair for each instance, and used that data to estimate the mean and standard deviation of the underlying transform distribution.

To select a set of primitives sufficient to represent all four models, we ran a greedy forward-selection process, starting with the single primitive that maximizes

$$score(m) = \prod_c \prod_j \prod_{\alpha,\beta} \Pr(R^j_{\alpha,\beta} \mid \bar{T}^{j,m}_{\alpha,\beta}) \,, \tag{4.5}$$

the data likelihood over all classes c given a single primitive. Then, we added the next single primitive which, together with the first, increased the aggregate score the most, etc. We stopped when the score failed to improve substantially with further additional primitives.

We measure the quality of these transforms on real images with an aggregate overlap score:

$$\frac{1}{NK^2} \sum_j \sum_{\alpha,\beta} overlap(\bar{T}^{j,Z_j}_{\alpha,\beta} A_j, B_j) \,, \tag{4.6}$$

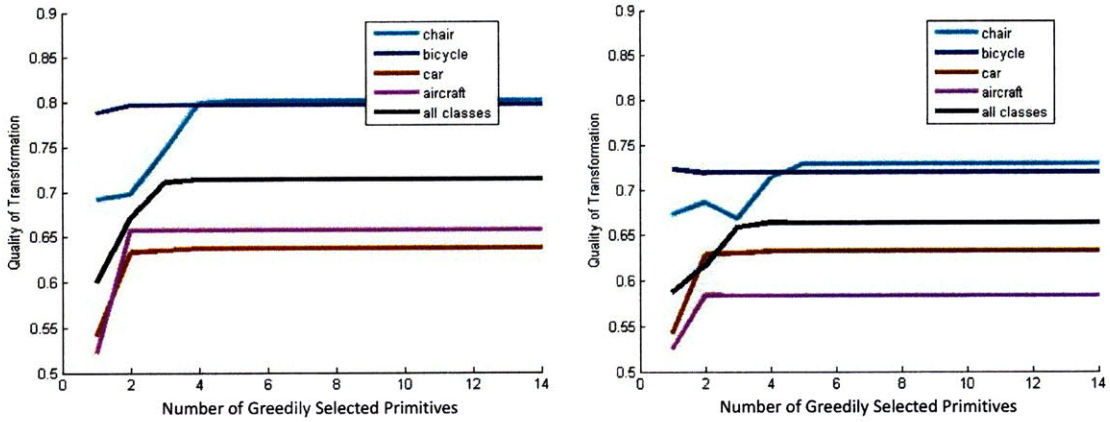

Figure 4-10: The quality of transforms as the number of available primitives (M) increases. Left: evaluated on training data; right: evaluated on held-out test data.

where A_j is the region associated with part j in view α, $\bar{T}_{\alpha,\beta}^{j,Z_j} A_j$ is the transform of that region into view β, B_j is the true region for that same part in view β, and the *overlap* of two regions is defined to be the ratio between their intersection and their union.

We ran this greedy selection algorithm for each of the four object classes independently, and for all four classes jointly. Figure 4-10 (left) shows the quality of the transforms on the training set as the number of primitives increases. We can see that four primitives suffice to model all of the classes effectively. Testing on held-out data (figure 4-10, right) shows that we are not overfitting the data, in general, but does suggest that a single primitive may be better than two for the bicycle class.

In all future experiments, we restricted our set of primitives to the four chosen in this process. All four were flat blocks (the same 3D shape), but at different 3D orientations. The primitives chosen for each part of each class are shown in Figure 4-11. The most popular primitive, which represents 10 of 21 parts, is actually the one used in the basic Potemkin model; but considerable representational accuracy is gained by adding other primitives to the set. We anticipate that for a more varied collection of object classes, it would be useful to increase the set of primitives, but are encouraged to find that individual primitives seem to be applicable to a large number of different object parts.

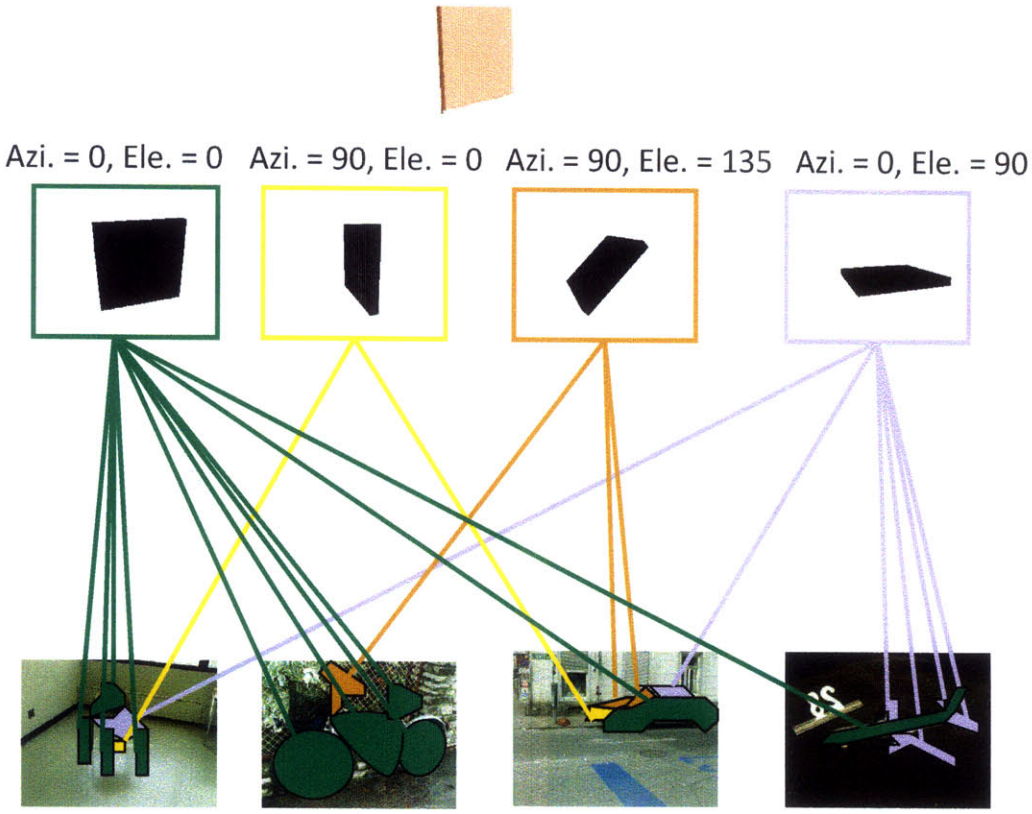

Figure 4-11: 3D shape primitives selected for each part of each class.

4.3.3 Part Labeling

Since we have introduced a self-supervised part labeling strategy, we would like to independently evaluate the quality of the labelings. We do this by comparing the automatic part labelings to hand-labeled images, measuring the percentage of pixels that have the same part labels in both the automatically labeled and hand labeled images.

Averaged over all real images from all views, the aggregate labeling overlap was: 96.72% for chairs, 98.23% for bicycles, 92.59% for airplanes, and 85.84% for cars.

The automatically generated labels for chairs and bicycles are extremely accurate, probably because the parts in these classes are clearly separable, as shown in figures 4-12 and 4-13.

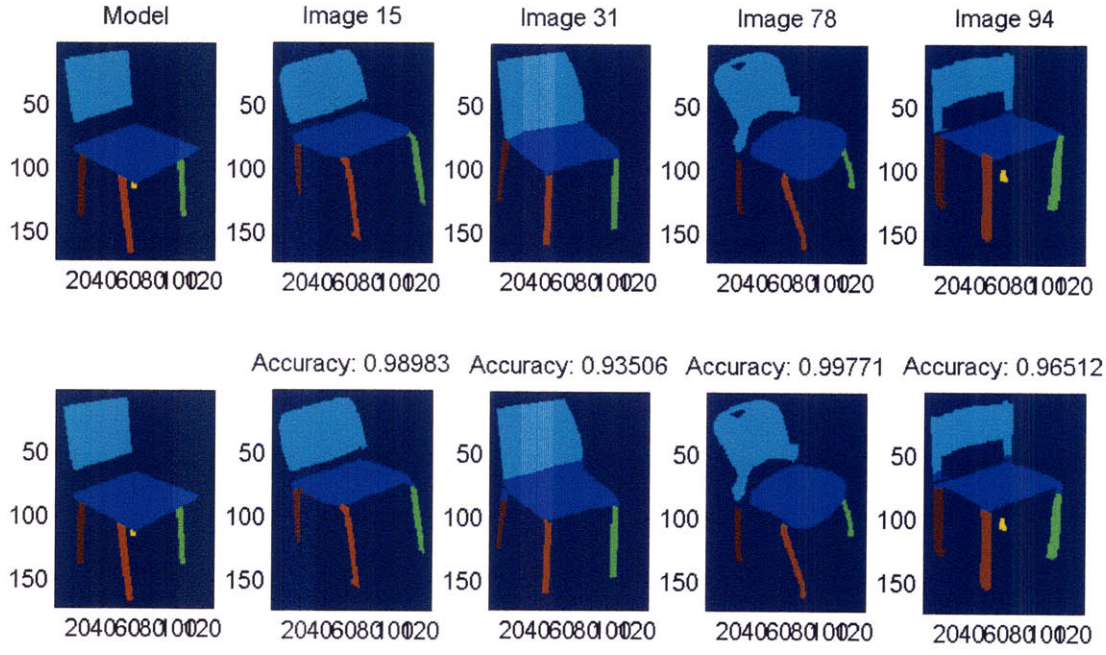

Figure 4-12: Some part labeling results for four-legged chairs. First row: hand labeling; second row: self-supervised labeling.

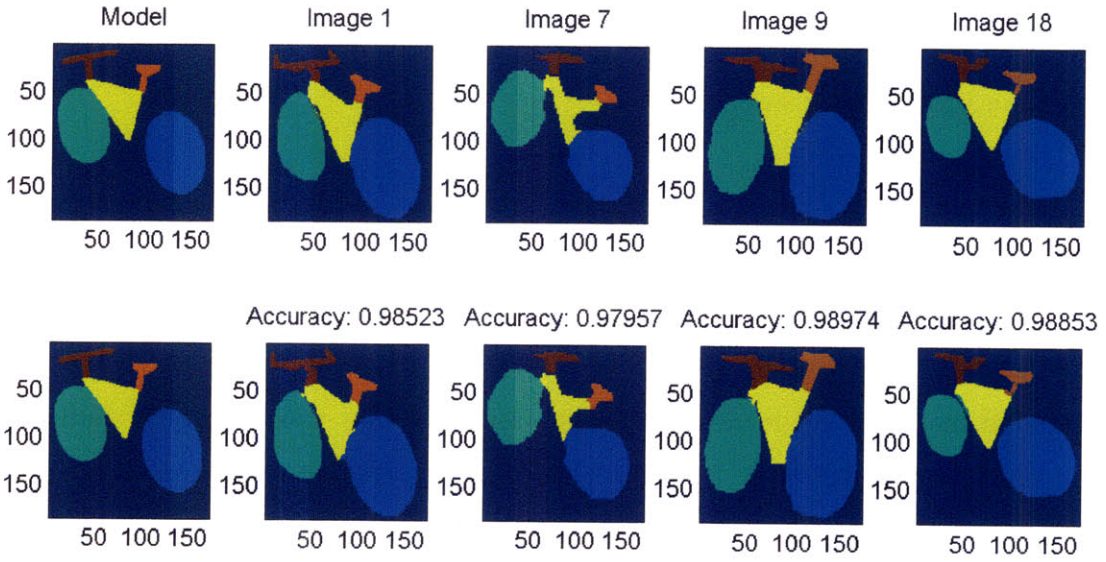

Figure 4-13: Some part labeling results of bicycles. First row: hand labeling; second row: self-supervised labeling.

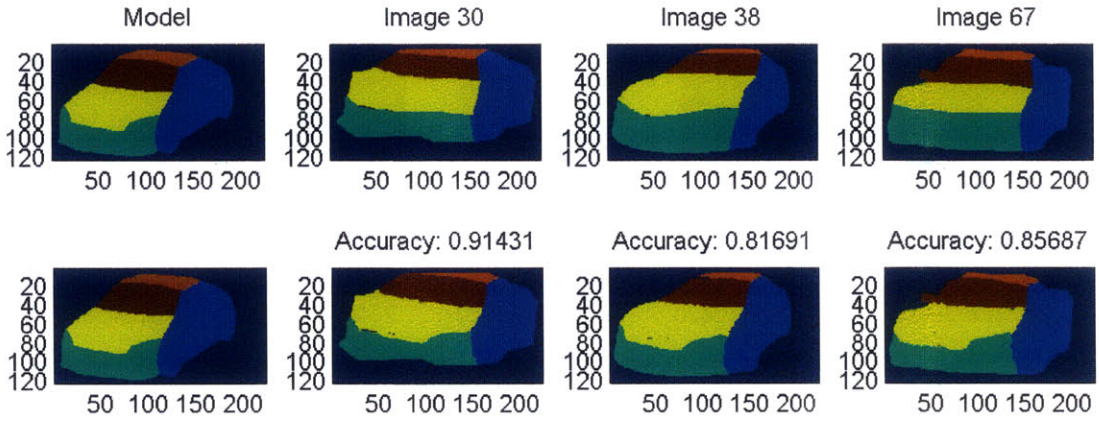

Figure 4-14: Some part labeling results of cars. First row: hand labeling; second row: self-supervised labeling.

On the other hand, the label results for cars are less accurate, as shown in figure 4-14. We found that different users labeled parts of cars in different ways, since these parts are not clearly defined, but the automatically generated part labels are more consistent. For example, some users thought the headlights of cars should belong to the "grille" part, while other users included headlights in the "hood" part. Ultimately, the real test is whether the labels improve detection performance. We will also compare the performance of these two labeling strategies in the end-to-end system in the following sections.

4.3.4 Single-View Detection

As in chapter 3, we test the quality of virtual examples generated by the generalized Potemkin model in single-view and multi-view detection tasks using the detector developed by Crandall et al. [10]. The goal of our experiments is to show that these virtual training images are almost as effective as input for a standard 2D detection method as novel real training images.

In order to illustrate the impact of using multiple oriented primitives on the quality of the virtual training images generated by the generalized Potemkin model, as well

as to highlight the model's effectiveness with very little training data, all 3D models and transforms were constructed using only two real images of the target object class. In this situation, the quality of virtual training images is most influenced by the prior transforms for each of the parts, which are determined by the chosen oriented primitive. This does mean, however, that the performance shown here could be improved by using more image pairs to refine the cross-view transforms, as shown in chapter 3. For efficiency, all detection was done at a single scale.

We trained the single-view detector using a combination of real and virtual images, for each of the four data sets. The real images are all from the same view bin, called "the target view bin". The virtual images were transformed from all "source" view bins different from the target view bin. In all cases, for testing we used 30 object images from the target view bin and 30 background images from the corresponding background data set (indoor scenes for chairs, street scenes for bicycles, sky scenes for airplanes, and road scenes for cars), and computed an ROC curve for discrimination between the object and background.

We did four repetitions of the detection experiments for each class, varying the target view bin. In each case, we held the number of virtual images constant (at 100 for chairs, 10 for bicycles, 75 for airplanes, and 50 for cars), and varied the number of real images in the target view bin. One data point on a curve in the first two columns of figure 4-15 is the percentage area under the ROC curve, averaged over all four repetitions. Note that an object class with higher variability among instances requires more virtual training images to achieve satisfactory detection performance. Thus we provided different amounts of virtual training images for different classes.

For each object class, we tested five methods of generating virtual images:

- No virtual images;
- Virtual images generated using a single global transform for the entire object, ignoring part structure;

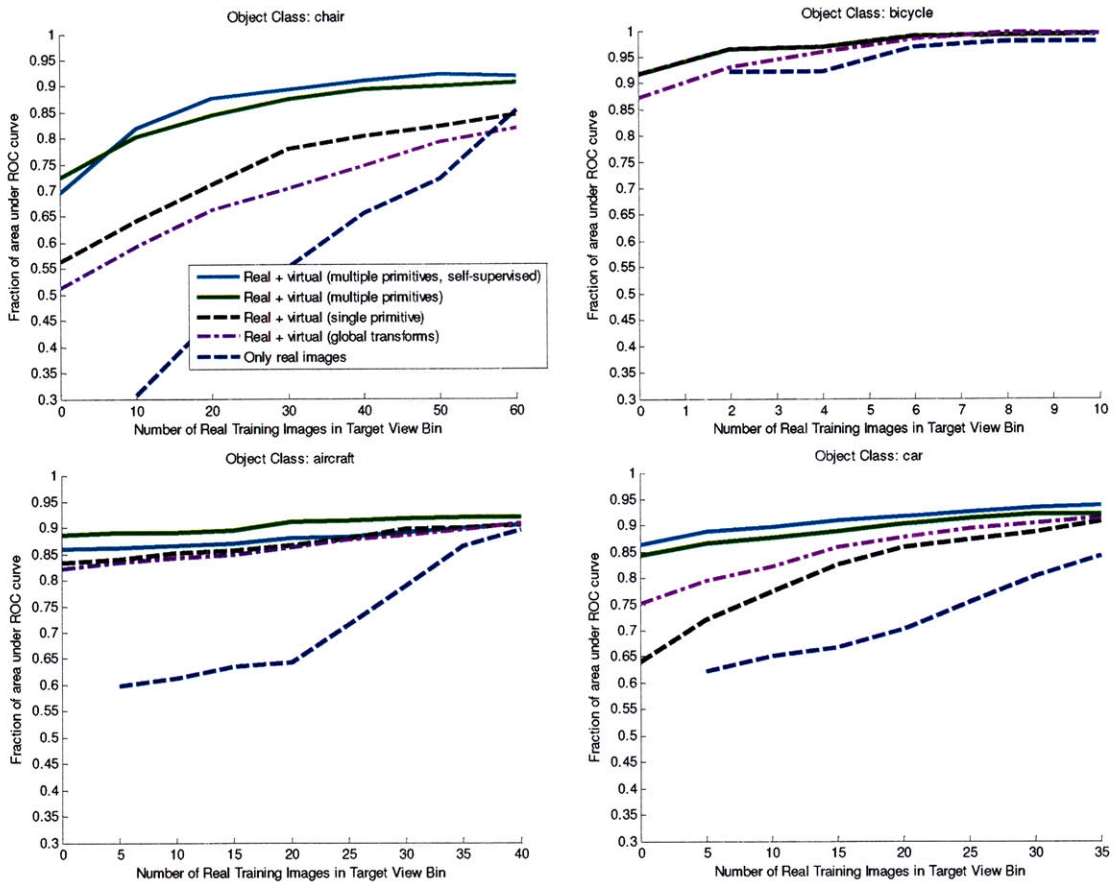

Figure 4-15: Average single novel view detection results over four repetitions of each of four object classes: chairs (left-top), bicycles (right-top), airplanes (left-bottom), and cars (right-bottom).

- Virtual images generated using a separate transform for each part, but based on a single primitive (the basic Potemkin model), using hand-labeled parts;
- Virtual images generated with multiple primitives (the generalized Potemkin model), using hand-labeled parts;
- Virtual images generated with multiple primitives (the generalized Potemkin model), using self-supervised part labeling.

In every case but the first, the same number of virtual training images, along with real training images, were used for training the view-based detector for the target class. The basic Potemkin model and the generalized Potemkin model were

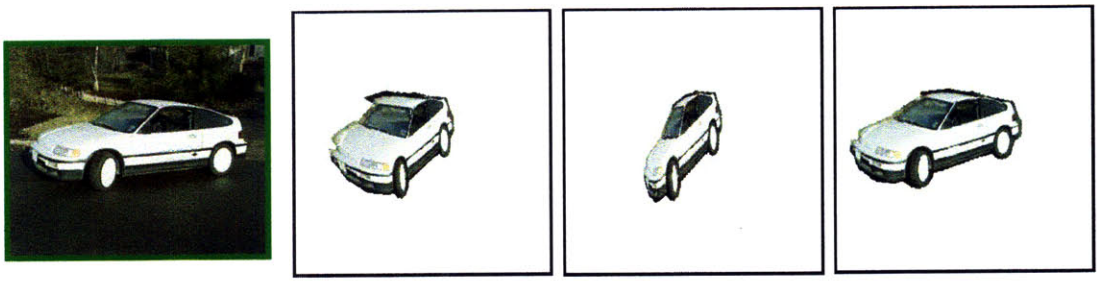

Figure 4-16: The virtual examples (from left to right) are constructed by using the generalized Potemkin model, the basic Potemkin model, and a global transformation respectively. These virtual examples are generated by transforming the car in the highlighted image from side viewpoint to frontal viewpoint.

constructed from only two real images, which were used to construct the skeleton and select the primitives for each part.

In the case of bicycles, all five parts selected only one primitive, which is the single primitive of the basic Potemkin model, so the results of the generalized Potemkin model are the same as for the original. In the other three classes, the use of additional primitives improves performance considerably. Note that for cars, the basic Potemkin model performs worse than a single global transform because of unrealistic virtual images, as shown in figure 4-16. However, the generalized Potemkin model generates reasonable virtual images which improve the detection performance. Some virtual training images (virtual examples placed into the background from the original image) generated from the generalized Potemkin model are shown in figure 4-17.

Self-supervised part labeling decreases detection performance of airplanes, compared to the model trained with hand-labeled parts. We observed that incorrect labels in self-supervised part labeling are mostly due to instances without tails or with one wing occluded, which are significantly different from the model instance we used to deform the shapes. In the future, we could use several model instances in the same view bin for matching shapes and defining labels. The labels of other instances could then be determined using the best matching model instance.

Our self-supervised approach for labeling parts works well if most instances of the

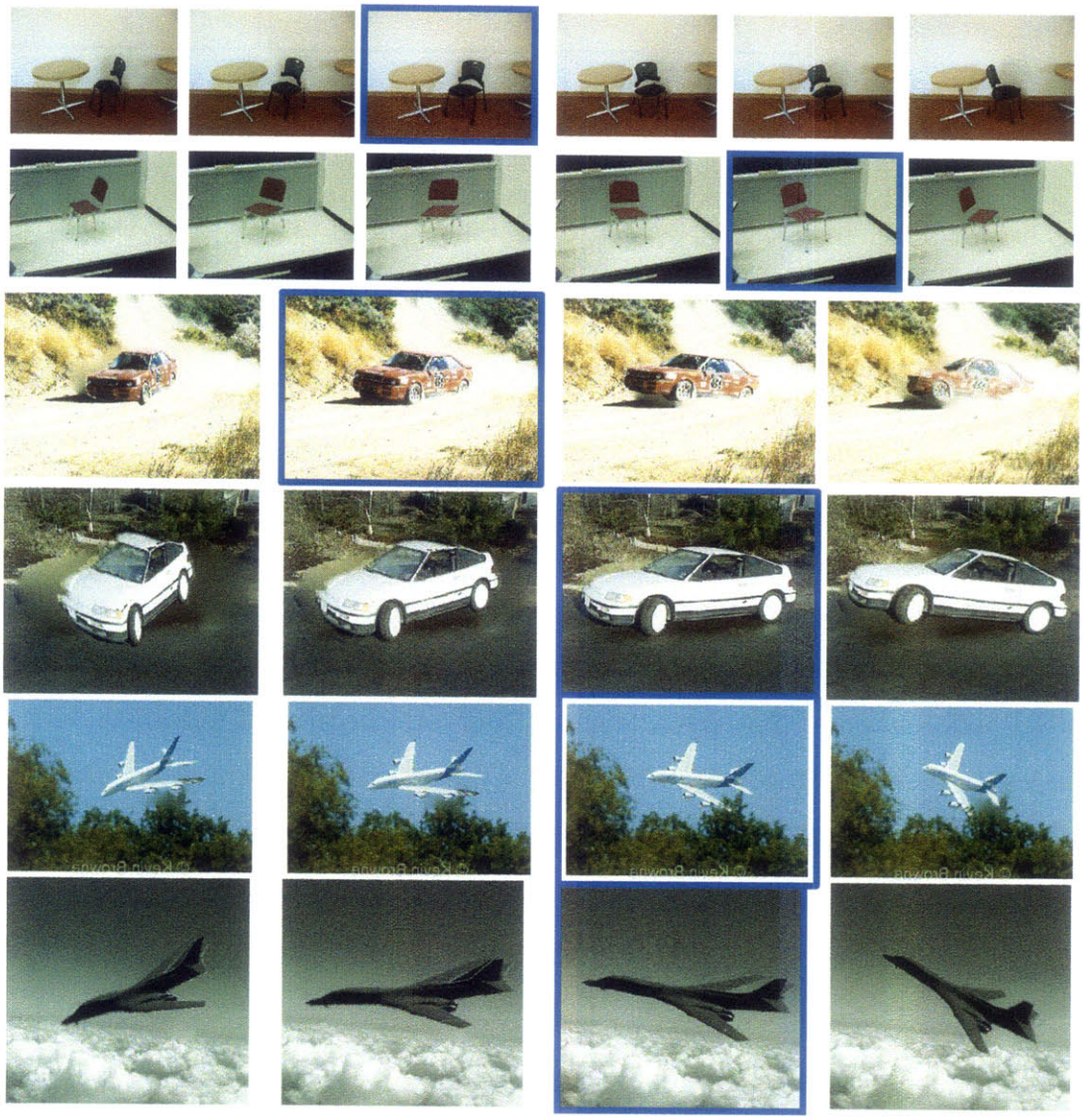

Figure 4-17: Six examples of real images (highlighted) and virtual images constructed by the generalized Potemkin model.

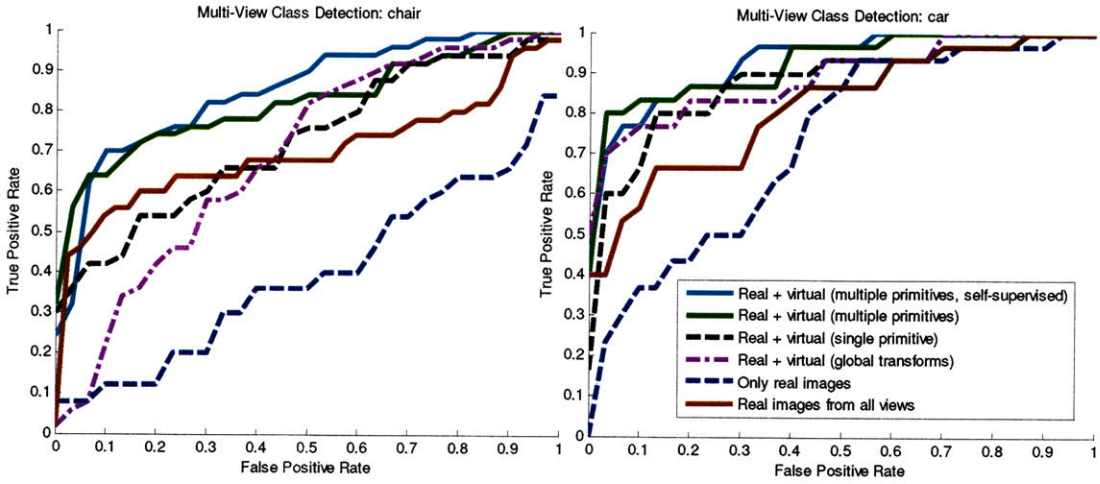

Figure 4-18: Multi-view detection performance (ROC curve) for chairs and cars.

target class have the same number of parts and big parts are not occluded, such as for four-legged chairs. It is particularly interesting that for cars, self-labeling works better than hand-labeling, probably because self-labeling provides a more consistent "definition" of parts, as shown in figure 4-14.

4.3.5 Multi-View Detection

We have also carried out experiments in multi-view detection, in which the goal is to label the class of an object, independent of its view point. We train single-view detectors for each view of each object, using virtual training data generated by the generalized Potemkin model, and output the class associated with the detector generating the largest response on each test image. We compared the same five training conditions as in the single-view detection experiment, as well as one in which all of the views are pooled and used to train a single detector. The results are shown in figure 4-18. The combined detectors trained with the data generated from the generalized Potemkin model outperform the others by a substantial margin.

4.3.6 Dalal-Triggs Detector

To support our claim that our virtual training images can improve the performance of any 2D image-based detector, we also conducted experiments using the detector from Dalal and Triggs [14], which is currently among the most accurate for cars. In this section, we performed experiments on the PASCAL VOC Challenge 2005 Cars Test 2 data set [20], which is a standard data set for many state-of-the-art approaches. The Dalal-Triggs detector requires a pool of positive/negative (car/non-car) examples for training. Because the data set in the PASCAL Challenge doesn't provide the outline segmentations for all objects, we use the car data set collected by our research group for training. However, we still used the "background" data set provided by the PASCAL challenge in the car detection task as the negative examples for training the Dalal-Triggs detector.

For each of the four training viewpoints of cars as shown in Figure 4-9, we collected 20 real images and generated 60 virtual images, by transforming real images from the other three viewpoints. We also made use of symmetries and generated mirrored versions of these images. Thus we had a total of 160 real training images and 480 virtual training images.

We set the parameters of the Dalal-Triggs detector and evaluated the detection results as in [20]. Note the best three average precision (AP) scores (see [20] for the definition of AP) on this standard test set in PASCAL VOC Challenge 2005 [20] are 0.304 (the Dalal-Triggs detector), 0.181, and 0.106. Our real training data set of cars is smaller than the training data set provided by the PASCAL VOC Challenge. In addition, our training data set doesn't cover all viewing directions of cars as those in [20]. Thus, for example, our trained detector cannot detect rear views of cars. Our goal in these experiments is only to give an estimate for the improvement expected from virtual images using a state-of-the-art detector on a standard data set.

Figure 4-19 shows the precision-recall curves of the Dalal-Triggs detector using our training data set on the VOC Challenge 2005 Car Test 2 data set. The incorporation

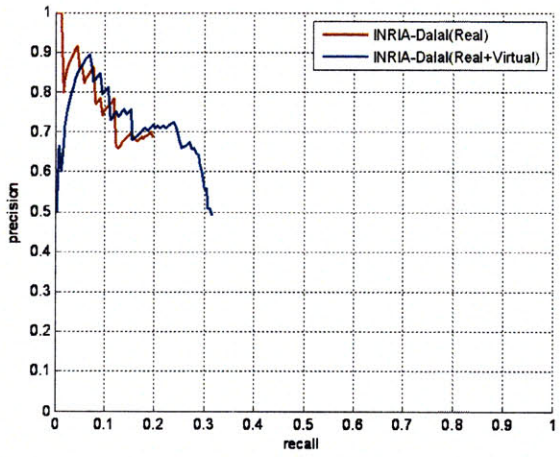

Figure 4-19: The precision-recall curves of the Dalal-Triggs detector using our training images.

of our virtual training images increased the AP score of the same detector from 0.162 to 0.272. Because we use the same number of the negative training examples (the background dataset), the accuracy of the trained detector doesn't improve (even lower when the recall is low). However, more cars in the test set were detected (highest recall: 0.32) using both real and virtual images of cars for training.

Chapter 5

The 3D Potemkin Model

In this Chapter, we augment the Potemkin model to support automatic reconstruction of the 3D shapes of object instances. We describe an extension of the Potemkin model, and show how it can be learned from a few labeled images for each class (Section 5.1). The learned 3D class model then can be used to estimate the 3D shape of an object instance, including occluded parts, from a single image (Section 5.2). We provide a quantitative evaluation of the shape estimation process on real objects, and demonstrate the 3D estimation is sufficiently accurate for a robot to successfully grasp the object (Section 5.3). We also demonstrate that the 3D reconstructions allow 2.5D data, such as depth maps from stereo processing, to improve the detection performance of existing detection systems [49, 14] (Section 5.4). In addition, we can use the learned 3D class model to construct 3D 'pop-up' models from photos (Section 5.5).

5.1 3D Class Models

Informally, the 3D Potemkin (3DP) *class model* can be viewed as a collection of 3D planar shapes, one for each part, which are arranged in three dimensions. The model can also be viewed as an approximation of a detailed 3D model using a small set of

3D planar polygons. The 3DP model specifies the locations and orientations of these parts in an object-centered 3D reference frame. In addition, it contains canonical images with labeled parts, which allow detection results to be decomposed into parts. The view space is divided into a discrete set of *view bins*, and an explicit 3D rotation from the object-centered 3D reference frame to the view reference frame is represented for each view bin.

The detection process produces a 3DP *instance model*, which is also a collection of 3D planar shapes arranged in three dimensions, corresponding to the parts of the particular 2D instance from which it was constructed.

More formally, a 3DP object class model with N parts is defined by:

- k *view bins*, which are contiguous regions of the view sphere. Each view bin is characterized by a *rotation matrix*, $T_\alpha \in R^{3\times 3}$, which maps object-centered 3D coordinates to 3D coordinates in each view reference frame α;

- k *part-labeled images*, specifying the image regions of parts of an instance in each view bin α;

- a *class skeleton*, $S_1, \ldots, S_N$, specifying the 3D positions of part centroids, in the object-centered reference frame; and

- N *3D planes*, $Q_i, i \in 1, \ldots, N$, specifying the 3D plane parameters for each planar part, in the object-centered reference frame;

$$Q_i : a_i X + b_i Y + c_i Z + d_i = 0. \tag{5.1}$$

In addition, the 3DP class model contains an estimated bounding polygon to represent the extent of the 3D part graphically, but this polygon plays no role in reconstruction. Instead, the part shapes in the part-labeled images for each viewpoint are used for reconstruction.

5.1.1 Estimating a 3DP Model from Data

In broad outline, the part centroids are obtained by solving for 3D positions that best project into the observed part centroids in the part-labeled images in at least two views. The 3D planes are chosen so as to optimize the match between the 2D transformations between the boundaries of corresponding parts in the part-labeled images. Below, we give a brief overview of this estimation process; further details can be found in previous chapters.

- The view bins are selected. The choice of view bins is arbitrary and guided by the demands of the application. In this chapter, we have used 12 views bins equally spaced around a circle at a fixed elevation. The view bins determine the associated rotation matrices.

- The part-labeled images in each viewpoint should be for similarly-shaped instances of the class (though they can be significantly deformed during the detection process) and two of them must be for the same actual instance.

- The skeleton locations S_j are estimated, as shown in Chapter 3, from the mean and covariance of the coordinates of the centroids of labeled part j in the set of part-labeled images.

- Learning the 3D planes is more involved. The process is trained in two phases: one generic, and one object-class specific.

  In Chapter 4, we ran a greedy selection algorithm to select a small set of primitives that would effectively model four test object classes (chair, bicycle, airplane, car), which together have 21 separate parts. Four primitive orientations suffice to model all of the parts of these classes effectively.

  Once the primitives are selected, a small set of images, which are a subset of the k part-labeled images in the model, of the same object instance, from any set of views, as long as each part is visible in at least two views, are used to estimate the positions and orientations of the parts for this class. By finding

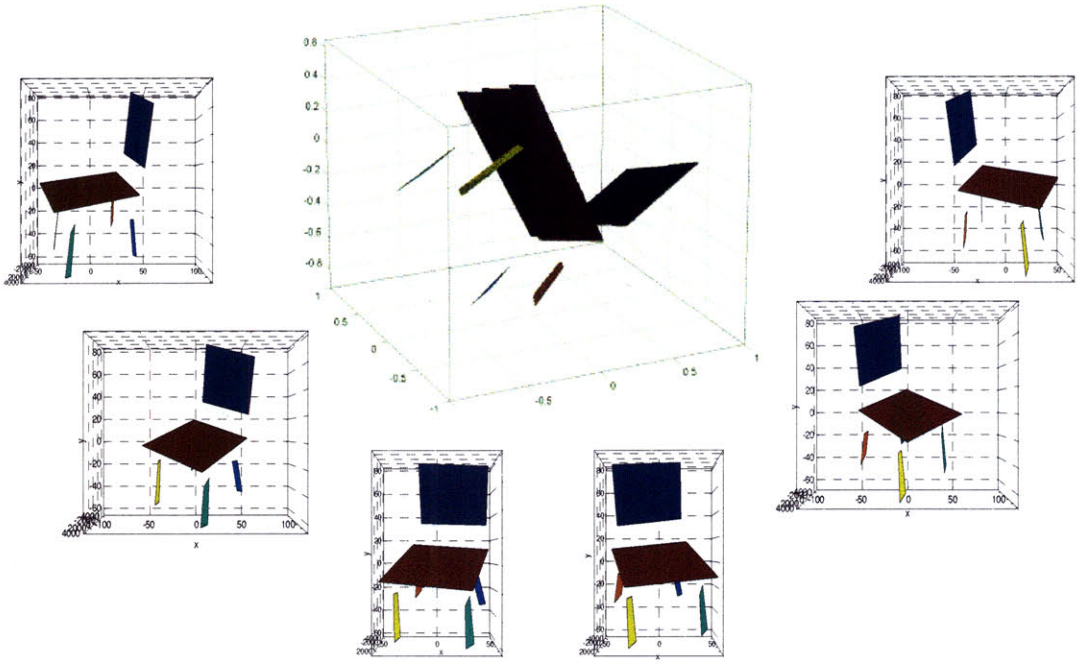

Figure 5-1: Learned 3DP class model for four-legged chairs in the object-centered reference frame, and in each view reference frame.

a similarity transform between the actual part outlines and the projections of the primitives in two different views, and having computed correspondences between the outlines of the projections of the primitives in phase 1, we can solve for 3D positions of points on the outline of the shape. This allows us to estimate a rough extent and planar model of the part in 3D, even when there is very little data available. We compute $Q_1, \ldots, Q_N$ based on these planar parts.

Figure 5-1 shows an estimated 3DP class model for chairs. It was constructed from two part-labeled images of the same object instance, knowing the view bins but with no further camera calibration.

These easily-obtained 3DP class models may not be able to capture highly detailed shape information or all of the variability within a class, but each provides adequate information to represent the basic 3D structure shared by instances of a class. For example, the 3D class model of airplanes is not unlike the paper cutout airplane in Figure 5-2.

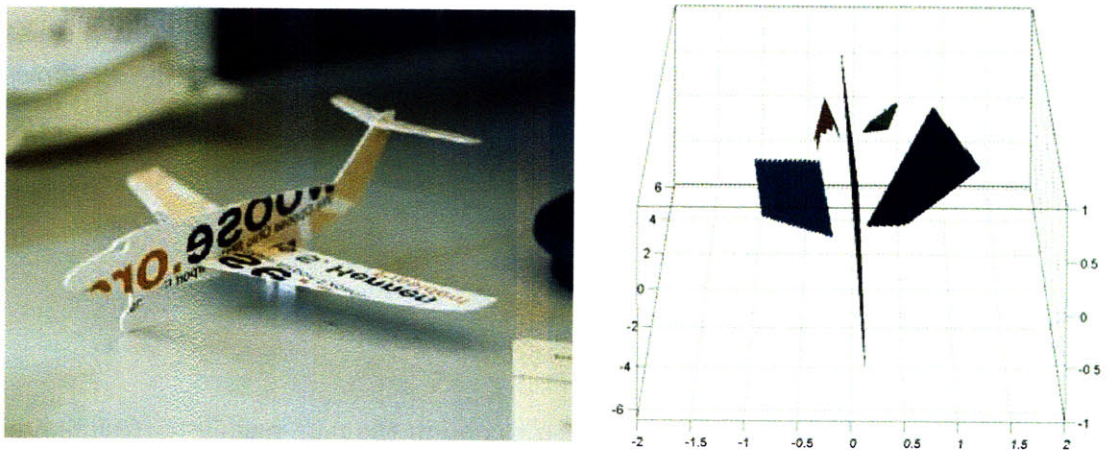

Figure 5-2: The 3DP class model of airplanes (right), constructed from two images of a real airplane, looks like the paper cutout airplane (left).

5.2 Automatic Single-View Reconstruction

In this section we will describe how to use 3DP object class models to reconstruct 3D objects from a single image. To achieve complete automation of the reconstruction process, we developed an approach involving several steps: detection, segmentation, part registration, and model creation. We will address the details of each step below.

5.2.1 Detection and Segmentation

Given the input image, we need to detect the object, identify the viewpoint, and obtain the contour of the object. This step can be carried out by using any existing multi-view object-class detection system. For example, Leibe et al.'s car detection system [39], composed of a set of seven view-dependent detectors [41], provides robust results on localizing cars (a bounding box and a coarse object segmentation for each detected car) and identifying their viewpoints on test images.

We assume that the detection system is able to, at least, determine a bounding box for the detected object and to identify the viewpoint bin. Within the bounding box, the outline of the detected object can be obtained by existing model-based

segmentation techniques [42, 36]. We use the part-labeled outline for the identified view bin in our model to initialize the segmentation process. All of the examples in this chapter were obtained by using the publically available implementation of level-set evolution by Li et al. [42].

5.2.2 Part Registration

Once an object outline is available, we need to obtain the part regions corresponding to the individual parts in the model. The approach we use is described in Section 4.2. This approach is based on the fact that objects in the same class, seen from the same view, have similar 2D arrangements of parts. That is, the centroids of the projected parts have characteristic arrangements.

Our approach uses the shape context algorithm [48] to match and deform the boundaries of the stored part-labeled image for the detected view bin into the corresponding boundary of the detected instance. This match induces a deformation of the part-labeled image that is used to predict internal part boundaries for the detected instance. Further details can be found in Section 4.2.

5.2.3 Creating the 3D Model

Now we are able to generate a 3D instance model from the segmented parts of the detected object in the input image using our 3D model of the class. We will assume a known camera matrix $M \in R^{3\times4}$ and a known 3D ground plane $Q_g(a_gX + b_gY + c_gZ + d_g = 0)$. Later we explore various ways of obtaining these.

We proceed in the following stages:

- Use the method developed by Hoiem et al. [32] to classify the ground region in the input image, and recover 3D coordinates of each image point (x_{im}, y_{im}) on the ground region by solving for X, Y, and Z in the following projection

equations.

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}. \tag{5.2}$$

$$x_{im} = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}. \tag{5.3}$$

$$y_{im} = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}. \tag{5.4}$$

$$a_g X + b_g Y + c_g Z + d_g = 0. \tag{5.5}$$

- For each planar part i of the 3DP class model, compute the parameters $(a_{i\alpha}, b_{i\alpha}, c_{i\alpha})$ of the 3D plane $Q_{i\alpha}$ in the 3D reference frame of view bin α (identified by the detector) by applying the 3D rotation matrix T_α to Q_i. Note that the scale of parameter $d_{i\alpha}$ is unknown.

- Fit a line l_g through image points where the detected object touches the ground region in the image, and get the 3D coordinates of those ground points.

- For each object part j that includes points along the line l_g, estimate $d_{j\alpha}$ based on the recovered 3D coordinates of points on that ground line. Then, solve for the 3D coordinates of all image points of part j using equations (5.2)–(5.4) and $Q_{j\alpha}$ (the plane supporting part j).

- For each part k connected via adjoining pixels in the image to some previously recovered part j, estimate $d_{k\alpha}$ based on the recovered 3D coordinates of those points on the intersection of part j and part k. Then solve for the 3D coordinates of all the image points of part k using equations (5.2)–(5.4) and $Q_{k\alpha}$ (the plane supporting part k). Repeat this process until all parts are reconstructed.

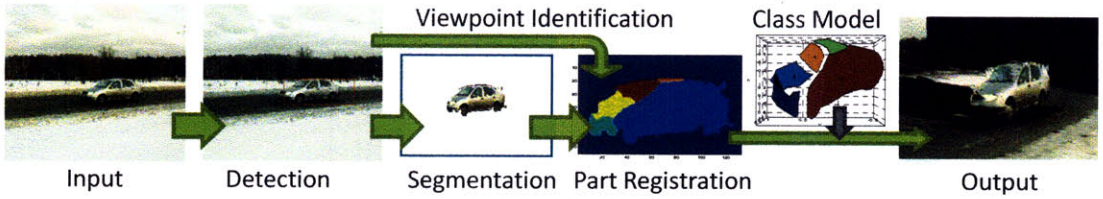

Figure 5-3: The processing pipeline for automatic single-view 3D reconstruction.

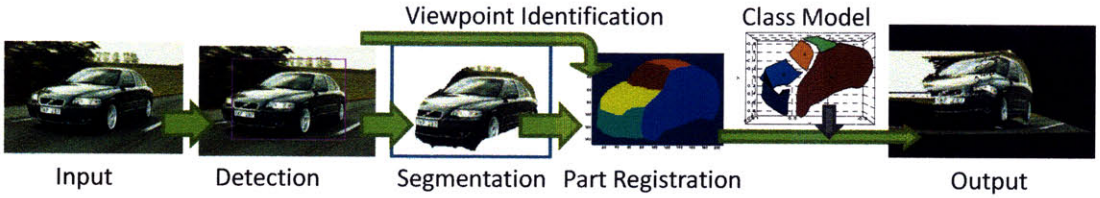

Figure 5-4: One failed example of our automatic single-view 3D reconstruction.

Figure 5-3 shows one example of a completely automated reconstruction. It involves detection [39], segmentation [42], part registration, and finally the projection of the 3D instance model into a new viewpoint. Figure 5-4 shows one failed example of our automated reconstruction due to inaccurate detection and segmentation.

5.2.4 Estimating Locations of Occluded Parts

After we reconstruct a 3D model for the visible parts of the detected instance in the source image, we are able to further predict approximate 3D coordinates of the occluded parts. We compute a 3D transformation from the 3D class model to the reconstructed 3D instance, by mapping 3D coordinates between the recovered 3D parts of the instance and corresponding 3D primitive parts in the class model. Then for each occluded part i of the instance in the source image, we apply this 3D transformation to part i in the class model.

The ability to estimate the 3D shape and extent of the entire instance, including parts that are not visible in the source image, is very useful in robotics applications, as demonstrated in section 5.3.2.

Figure 5-5: A 3D reconstruction (left) from the green-circled car, which is in the image (right) of 20 model cars.

5.3 3D Object Localization

One important use of 3D localization and shape estimation is in robotic manipulation, where it can be crucial for the robot to have good 3D estimate of the object's pose and shape. Although in some special cases, when grasping is done perpendicular to the image plane, simple grasping can be done without a 3D understanding of the scene, robust manipulation of complex objects requires reliable 3D information.

In this section, we present several evaluations of the reliability of the 3D instance models constructed by our system.

5.3.1 Localization Performance

We evaluated the quality of the 3D reconstruction results on a domain consisting of 20 diecast model cars (Figure 5-5), varying in shape and appearance. We calibrated a fixed camera (both M and Q_g are estimated) in advance, using the Matlab camera calibration toolbox. We then randomly placed each of the cars on the known 3D ground plane, a black table, within a 1m by 1.2m area, visible from the camera.

We took images of each of the cars, and constructed 3D instance models of them. In this setting, detection was done simply by color segmentation. Figure 5-5 shows a typical reconstruction (for the green-circled car in the figure) and its bounding box in the 3D coordinate system.

We measured the accuracy of the single-view reconstructions by comparing the ground truth with the recovered 3D model according to three criteria:

- The overlap of two volumes (the volume of the intersection divided by the volume of the union): the estimated 3D bounding box of the car and the ground truth bounding box;
- The distance from the estimated 3D centroid of the car to the ground truth centroid; and
- The absolute value of the difference between the estimated orientation of the car and the ground truth.

For comparison, we also tested the quality of reconstruction using a single 3D surface perpendicular to the ground plane for the whole object (as in [32]) instead of our 3D class model. The 3D objects reconstructed using our 3D class model were much more accurate than those modeled by only a vertical 3D plane, as shown by the average measurements in the table below. (The single-plane method, which use a single 3D surface perpendicular to the ground plane for the whole object, cannot compute bounding boxes. Thus, overlap scores are not available).

	overlap	centroid error	orientation error
3DPotemkin	77.5%	8.75mm	2.34°
Single plane		73.95mm	16.26°

All our estimates were reasonably accurate, except for the red-circled Ferrari F1, whose shape is the most different from our 3D class model of cars, for which it had 26.56% overlap, 24.89 *mm* centroid error, and 3.37° angle error.

5.3.2 Robot Manipulation

We can use estimated 3D poses of the parts of objects as input to a robot motion planner, which calculates a motion trajectory for a robot arm and hand, which should

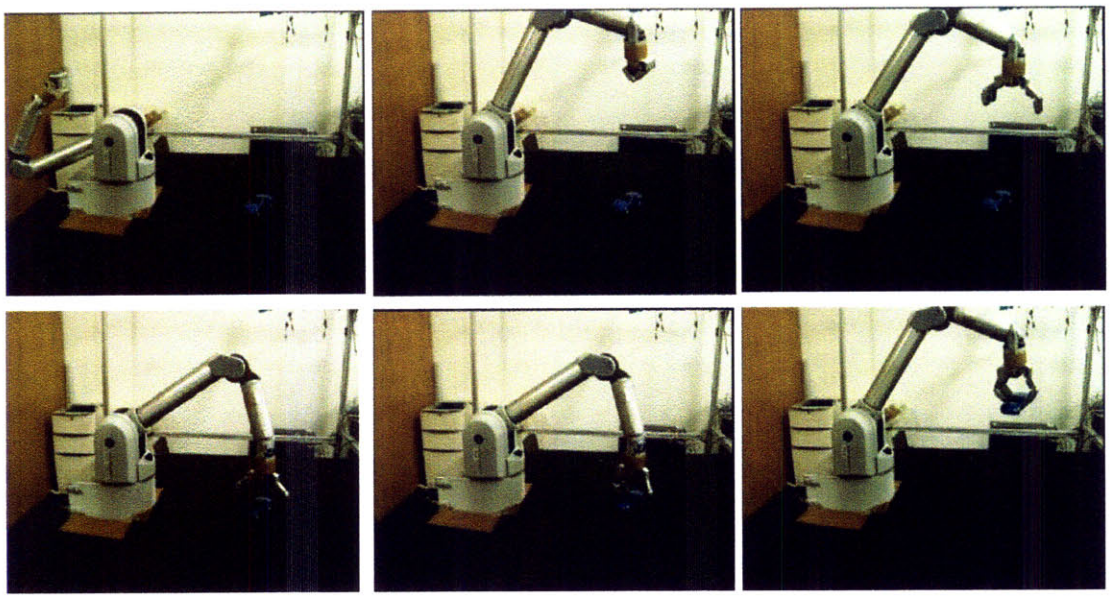

Figure 5-6: Some snapshots taken in the process (from left to right, from the first row to the second row) of the robot grasping a car.

result in the robot grasping the object. We have used a Barrett robot arm and hand, together with the OpenRave robot motion planning system [17] together with the 3DP model estimation outputs, to build a system that demonstrates hand-eye coordination in picking objects. Figure 5-6 shows some snapshots of the Barrett arm picking up a model car.

To test the utility of the 3D localization results, we placed each of the 20 model cars in 3 different positions and orientations on the table, and reconstructed the 3D car from each input image. The robot successfully grasped the car in all these 60 trials, except for the trials involving the red Ferrari F1 (circled in red in Figure 5-5). Although the reconstruction for the Ferrari was accurate enough for a grasp, the model car was too fragile and it shattered.

We repeated the same 60 experiments using the single-plane reconstruction method. The robot was only able to grasp the car in 6 of 60 trials. The successful grasps happened only when thin cars were placed with their sides nearly face-on to the camera. For these cases, the estimated orientation and 3D centroids are accurate enough for a grasp.

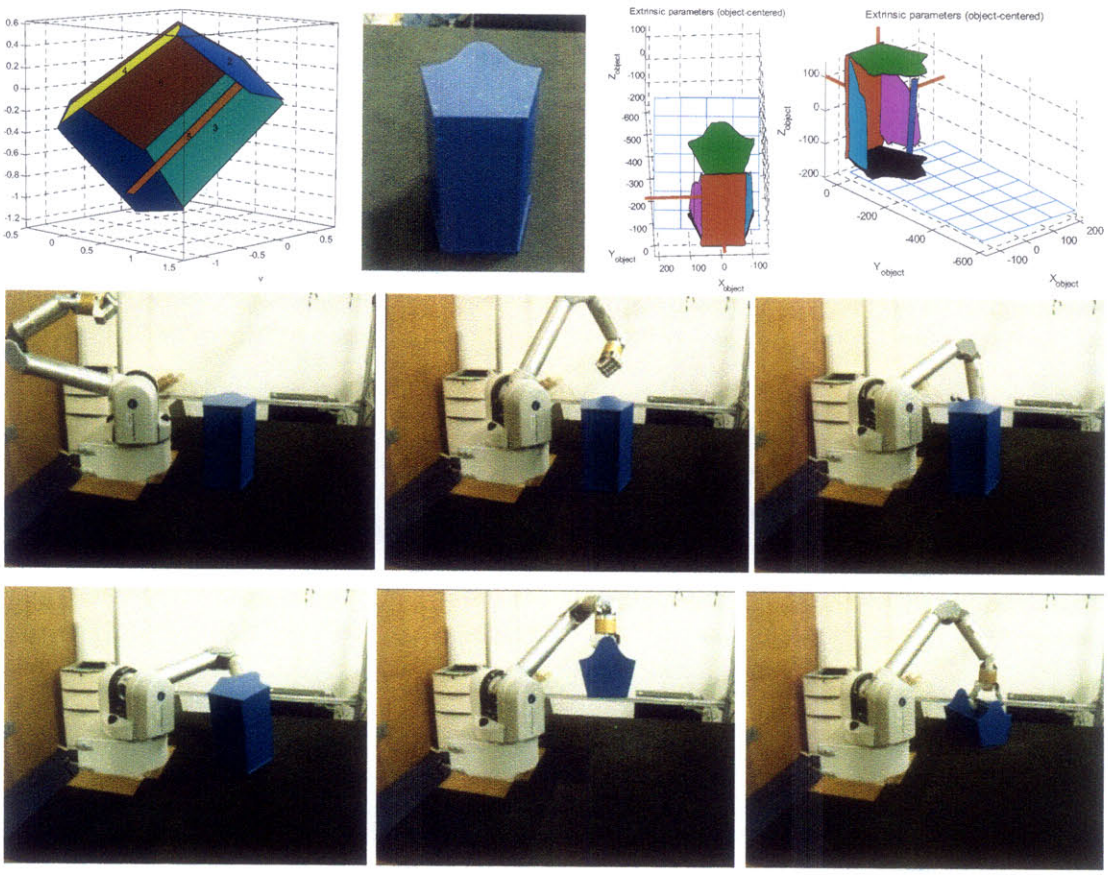

Figure 5-7: Top (From left to right): Our 3D class model of baskets, the input image, and two views of the reconstructed 3D basket. The handle is recovered even it is occluded in the input image. Bottom: Some snapshots taken in the process (from left to right, from the first row to the second row) of the robot's grasp of a basket

To demonstrate the ability to predict the position of, and then grasp, occluded parts, we built a 3DP model from two views of a wooden 'basket', and then showed the system a completely novel view in which the handle is occluded (shown in Figure 5-7). The system recovered approximate 3D coordinates for the handle of the basket and the robot grasped the basket successfully via the occluded handle, as shown in figure 5-7. In this particular case, the class model was built for the particular instance, so there was no generalization exhibited, but we expect the results to continue to be good when there are more instances in the class.

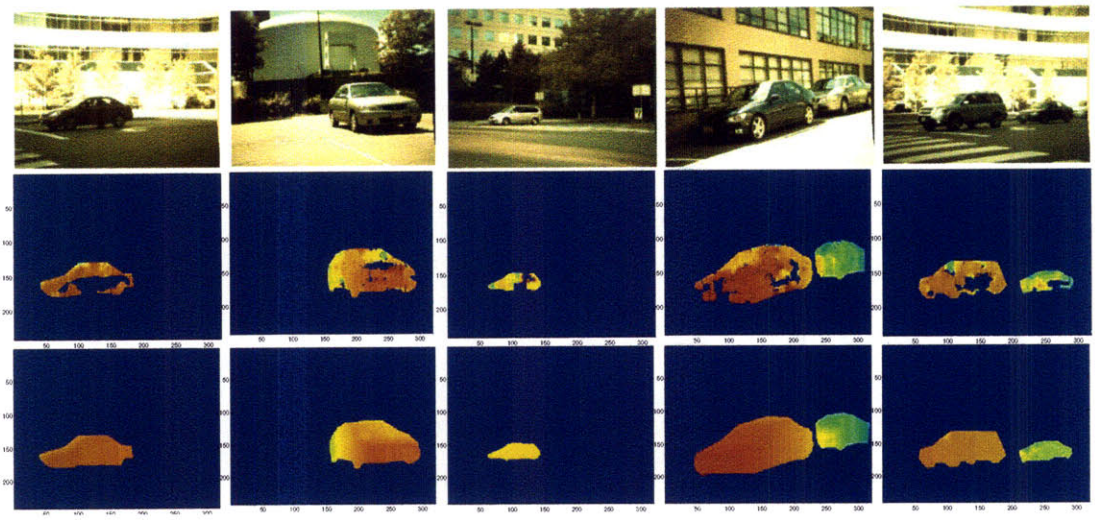

Figure 5-8: Imaged objects with corresponding stereo depth maps (the second row) and depth maps generated by reconstruction from the 3DP model (the third row).

5.4 Object Detection Using 2.5D Data

We can use 3DP class models to predict reasonably accurate depths of objects in 2D intensity images (Figure 5-8 for labeled cars) using projections from 3D reconstructions. Given a predicted depth map, we can match it against other available 2.5D data (such as from stereo images), to improve detection performance. In this section, we apply this strategy to enhance the performance of two existing 2D object detection systems, using intensity images coupled with 2.5D range images.

We performed experiments using two trained single-view detectors for cars: one from Murphy, Torralba, and Freeman [49] and one from Dalal and Triggs [14]. The Dalal-Triggs detector is currently among the most accurate for cars.

In this experiment, our test set consists of 109 outdoor intensity images and corresponding depth maps obtained from a Videre Designs stereo device [9]. These images include 127 annotated cars, all seen from the same viewpoint.

We ran the 2D detector on each image to get the 15 highest-scoring candidate detections. Each candidate c_i consists of a bounding box b_i and a likelihood l_i. We converted the SVM outputs of the Dalal and Triggs system to probabilities using the

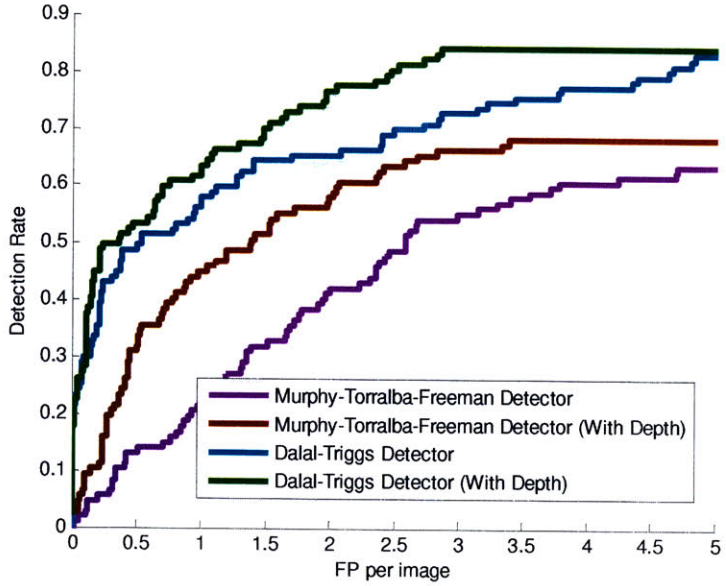

Figure 5-9: ROC curves for car detection.

method of Platt [52]. Inside each candidate box, we used the 3DP reconstruction method to estimate a depth map Z_i. Note that the 3D ground planes and the camera parameters of these images are unknown, so we used a default camera matrix and 3D ground plane (as in [33]).

Then, we compared the predicted depths with the measured depths, in each of the detection bounding boxes. Because the camera was not calibrated, we had to first register our predicted depth map Z_i to the stereo depth map Z_s. We computed the difference D_i between Z_i and the normalized stereo depth map Z_s over the depth region inside b_i as follows (a_1 is a scale parameter and a_2 is an offset):

$$D_i = \min_{a_1,a_2} \sqrt{(Z_s - (a_1 Z_i + a_2))^2} \tag{5.6}$$

Then, we computed the posterior likelihood for each of the 15 candidates in the test image using a very simple log-linear model:

$$\exp(\log(l_i) \times w + \log(1 - D_i) \times (1 - w)) \tag{5.7}$$

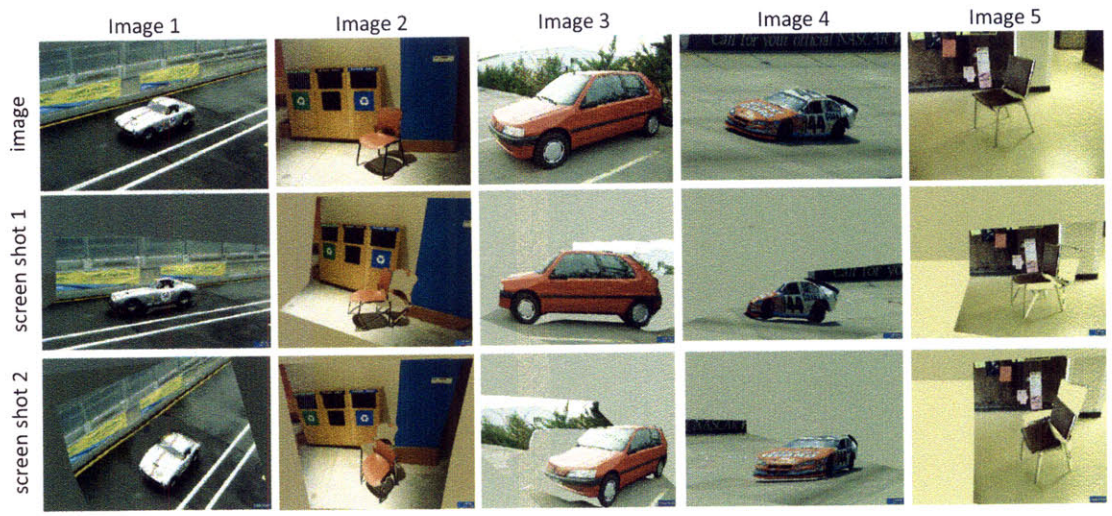

Figure 5-10: Row 1: Original image. Row 2 and 3: 3D instance model generated from the 3DP class model.

The weighting parameter w was determined empirically by optimizing over a separate validation set of 20 images and corresponding depth maps. For the Dalal-Triggs detector, we found that $w = 0.6$ and for the Murphy-Torralba-Freeman detector, $w = 0.5$.

Figure 5-9 shows that the performance of both detectors can be substantially improved by generating predicted range information for each detection and filtering detections based on the match between the predicted and actual ranges.

5.5 3D Pop-Up Scenes from a Single Image

Once we reconstruct a 3DP instance model from an image, we can automatically construct a 3D 'pop-up' visualization of that instance. We use the same method as [32] to set the camera matrix and a 3D ground plane. Figure 5-10 shows some 3D popup models of chairs and cars. In contrast to previous work on photo pop-up [32, 62, 59], our results focus on creating realistic 3D shapes of objects.

Chapter 6

Conclusions

In this dissertation we have proposed a new learning approach that efficiently learns object classes in a wide variety of viewpoints. We have demonstrated that our learning approach can be coupled with any 2D image-based detection system, and that it successfully improves detection performance on a variety of object classes.

The contribution of this work is the Potemkin model, which exploits the fundamental relationship between multiple views of an object class, and facilitates information transfer across viewpoints. We have shown how the Potemkin model can be efficiently constructed from a few images, and can be used for effective multi-view object class detection.

The Potemkin model provides a middle ground between methods that are based only on 2D images and those based only on a 3D model; it seeks to gain the best of both worlds by taking advantage of our general knowledge of the relationship between 3D objects and their 2D projections and using this knowledge as leverage for efficient and effective learning for multi-view object class detection.

The most important use of the Potemkin model is to circumvent one of the key roadblocks to effective multi-view recognition, namely the need for collecting large amounts of training data in all the viewpoints of interest. The Potemkin model makes

efficient use of training data in each view by transforming it into virtual training data for all other views. The virtual training examples can be used as training input for any existing 2D image-based detection system. We have demonstrated the effectiveness of these virtual examples in reducing training data requirements by training single-view detectors in novel views and in multi-view detection. We have also shown these virtual examples improve detection performances in object localization and detection tasks.

We have also introduced an extension of the Potemkin model which can be used to reconstruct 3D shapes of detected objects automatically from a single image. This 3D Potemkin model is based on a set of 3D oriented primitives, and can be learned from a small set of labeled views of an object in the class. Once the 3D Potemkin model is learned, the reconstruction mechanism can be built on top of any multi-view object class detection system. We have shown this reconstruction mechanism generates realistic views of 3D models, and provides accurate 3D information for entire objects. We have also demonstrated the usefulness of this reconstruction in three applications: robot manipulation, object detection, and object pop-up.

Bibliography

[1] E. Bart, E. Byvatov, and S. Ullman. View-invariant recognition using corresponding object fragments. In *Proceedings of European Conference on Computer Vision*, 2004.

[2] T. Beier and S. Neely. Feature-based image metamorphosis. In *SIGGRAPH*, 1992.

[3] A. Berg, T. Berg, and J.Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. CVPR*, 2005.

[4] I. Biederman. Recognition-by-components: A theory of human image understanding. In *Psychological Review*, pages 94:115–147, 1987.

[5] I. Biederman and E. E. Cooper. Priming contour-deleted images: Evidence for intermediate representations in visual object recognition. In *Cognitive Psychology*, pages 23:393–419, 1991.

[6] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[7] S. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics*, 27:279–288, 1993.

[8] H. Chiu, L. P. Kaelbling, and T. Lozano-Perez. Virtual training for multi-view object class recognition. In *Proc. CVPR*, 2007.

[9] Company. Videre design. mega-d megapixel digital stereo head. In *http://users.rcn.com/mclaughl.dnai/products.htm*, 2000.

[10] D. Crandall, P. F. Felzenszwalb, and D. P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Proceedings of Computer Vision and Pattern Recognition*, 2005.

[11] D. Crandall and D. P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *Proceedings of European Conference on Computer Vision*, 2006.

[12] D. Crandall and D. P. Huttenlocher. Composite models of objects and scenes for category recognition. In *Proc. CVPR*, 2007.

[13] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *IJCV*, 40(2):123–148, 2000.

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

[15] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *European Conference on Computer Vision Workshop on Statistical Learning in Computer Vision*, 2004.

[16] X. Decoret, F. Durand, F. Sillion, and J. Dorsey. Billboard clouds for extreme model simplification. In *ACM SIGGRAPH*, 2003.

[17] R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, CMU, 2008.

[18] G. Elidan, G. Heitz, and D. Koller. Learning object shape: from drawings to images. In *Proceedings of Computer Vision and Pattern Recognition*, 2006.

[19] M. Everingham and A. Zisserman. Identifying individuals in video by combining generative and discriminative head models. In *Proceedings of International Conference on Computer Vision*, pages 1103–1110, 2005.

[20] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The PASCAL visual object classes challenge 2005 (VOC2005) results. In *1st PASCAL Challenges Workshop*, 2006.

[21] Z. G. Fan and B. L. Lu. Fast recognition of multi-view faces with feature selection. In *Proceedings of International Conference on Computer Vision*, 2005.

[22] P. Feltzenswalb and D. Hutenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 2005.

[23] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of Computer Vision and Pattern Recognition*, pages 264–271, 2003.

[24] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of Computer Vision and Pattern Recognition*, 2005.

[25] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 2006.

[26] D. Forsyth and J. Ponce. *Computer vision: a modern approach.* Pearson Education, Inc., 2003.

[27] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis: Second Edition.* Chapman and Hall/CRC, 2004.

[28] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of International Conference on Computer Vision*, 2005.

[29] K. Grauman, G. Shakhnarovich, and T. Darrell. Virtual visual hulls: example-based 3d shape inference from sihouettes. In *Proc. of ECCV workshop on statistical methods in video processing*, 2004.

[30] R. Hartley and F. Schaffalitzky. PowerFactorization: 3D reconstruction with missing or uncertain data. In *AJAWCV*, 2003.

[31] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004.

[32] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, 2005.

[33] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. CVPR*, 2006.

[34] D. Hoiem, C. Rother, and J. Winn. 3D LayoutCRF for multi-view object class recognition and segmentation. In *Proc. CVPR*, 2007.

[35] Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *ACM SIGGRAPH*, 1997.

[36] M. Kumar, P. Torr, and A. Zisserman. Obj cut. In *Proc. CVPR*, 2005.

[37] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proceedings of British Machine Vision Conference*, 2004.

[38] A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3D object class recognition. In *Proceedings of Computer Vision and Pattern Recognition*, 2007.

[39] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3D scene analysis from a moving vehicle. In *Proc. CVPR*, 2007.

[40] B. Leibe and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision*, 2004.

[41] B. Leibe, E. Seemannand, and B. Schiele. Pedestrian detection in crowded scenes. In *Proc. CVPR*, 2005.

[42] C. Li, C. Xu, C. Gui, and M. Fox. Level set evolution without re-initialization: a new variational formulation. In *Proc. CVPR*, 2005.

[43] S. Li and Z. Zhang. Floatboost: Learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112–1123, 2004.

[44] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-indpendent object class detection using 3d feature maps. In *Proceedings of Computer Vision and Pattern Recognition*, 2008.

[45] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of International Conference on Computer Vision*, pages 1150–1157, 1999.

[46] D. Marr. *Vision.* San Francisco: W. H. Freeman, 1982.

[47] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three dimensional structure. In *Proceedings of the Royal Society of London, B*, pages 200:269–294, 1978.

[48] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *Proc. CVPR*, 2001.

[49] K. Murphy, A. Torralba, and W. T. Freeman. Graphical model for recognizing scenes and objects. In *Proc. NIPS*, 2003.

[50] J. Ng and S. Gong. Multi-view face detection and pose estimation support vector machine across the view sphere. In *Workshp on Recognition, Analysis and Tracking of Faces and Gestures*, 1999.

[51] A. Pentland. Recognition by parts. In *Proceedings of International Conference on Computer Vision*, pages 612–620, 1987.

[52] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 2000.

[53] M. Prasad, A. W. Fitzgibbon, and A. Zisserman. Fast and controllable 3d modelling from silhouette. In *Eurographics*, 2005.

[54] M. Prasad, A. Zisserman, and A. Fitzgibbon. Single view reconstruction of curved surfaces. In *Proc. CVPR*, 2006.

[55] G. Prasanna, G. Mulgaonkar, L. G. Shapiro, and R. M. Haralick. Matching sticks, plates and blocks objects using geometric and relational constraints. *Image Vision Computation*, 2(2):85–98, 1984.

[56] L. G. Roberts. Machine perception of three-dimensional solids. Technical Report 315, Lincoln Laboratory, 1963.

[57] S. Romdhani and T. Vetter. Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In *Proc. CVPR*, 2005.

[58] F. Rothganger, S. Lazabnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Internationl Journal of Computer Vision*, 66(3):231–259, 2006.

[59] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 2007.

[60] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *Proceedings of International Conference on Computer Vision*, 2007.

[61] S. Savarese and L. Fei-Fei. View synthesis for recognizing unseen poses of object classes. In *Proceedings of European Conference on Computer Vision*, 2008.

[62] A. Saxena, M. Sun, and A. Ng. Learning 3d scene structure from s single still image. In *Proc. of ICCV workshop on 3D representation for recognition*, 2007.

[63] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *Proceedings of Computer Vision and Pattern Recognition*, 2000.

[64] S. Seitz and C. Dyer. View morphing. In *SIGGRAPH*, 1996.

[65] P. Sturm and S. J. Maybank. A method for interactive 3d reconstruction of piecewise planar objects form single images. In *BMVC*, 1999.

[66] R. Szeliski. Video mosaics for virtual environments. *Computer Graphics and Applications*, 16:22–30, 1996.

[67] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *Proceedings of Computer Vision and Pattern Recognition*, 2006.

[68] A. Torralba, K. P. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. In *Proceedings of Computer Vision and Pattern Recognition*, 2004.

[69] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[70] M. Vrown and D. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *3D Imaging and Modeling*, 2005.

[71] M. Weber, W. Einhaeuser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *Conference on automatic face and gesture recognition*, 2000.

[72] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proceedings of European Conference on Computer Vision*, pages 101–108, 2000.

[73] J. Xiao and M. Shah. Tri-view morphing. *CVIU*, 96, 2004.

[74] P. Yan, M. Khan, and M. Shan. 3D model based object class detection in arbitrary view. In *Proceedings of International Conference on Computer Vision*, 2007.

[75] L. Zhang, G. Dugas-Phocion, J. Samson, and S. Seitz. Single view modeling of free-form scenes. In *Proc. CVPR*, 2001.