Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Learning to generate novel views of objects for class recognition

Han-Pang Chiu *, Leslie Pack Kaelbling, Tomás Lozano-Pérez

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

A R T I C L E   I N F O

Article history:
Received 14 July 2008
Accepted 10 June 2009
Available online 12 August 2009

Keywords:
Object recognition
Categorization
Shape representation

A B S T R A C T

Multi-view object class recognition can be achieved using existing approaches for single-view object class recognition, by treating different views as entirely independent classes. This strategy requires a large amount of training data for many viewpoints, which can be costly to obtain. We describe a method for constructing a weak three-dimensional model from as few as two views of an object of the target class, and using that model to transform images of objects from one view to several other views, effectively multiplying their value for class recognition. Our approach can be coupled with any 2D image-based recognition system. We show that automatically transformed images dramatically decrease the data requirements for multi-view object class recognition.

1. Introduction

After seeing a number of instances of objects from the same class from different views, a vision system should be able to recognize other instances of that class from arbitrary views, including views that have not been seen before. In most current approaches to object class recognition, the problem of recognizing multiple views of the same object class is treated as one of recognizing multiple independent object classes, with a separate model learned for each. This independent-view approach can be effective when there are many instances at different views available for training, but can typically only handle new viewpoints that are a small distance from some view on which it has been trained.

An alternate strategy is to use multiple views of multiple instances to construct a model of the distribution of three-dimensional shapes of the object class. Such a model would allow recognition of many entirely novel views. This is a very difficult problem, which is as yet unsolved.

In our work, we take an intermediate approach, which exploits the fact that there is a fundamental relationship between multiple views of the same class of objects, and allows subsequent unlabeled views of new objects from the same class to be transformed into novel views. We define the *Potemkin*[1] *model* of a three-dimensional object as 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 object. 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 of each part's image between views as a 2D perspective transformation.

The Potemkin model is trained and used in three phases. 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 relatively 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 using that knowledge to accelerate learning about new object types which are constituted of these primitives.

The second phase is class-specific: a few images from a target class (typically of 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 image transforms between pairs of views, for each part.

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 recognition system, which can then be used

* Corresponding author.
E-mail address: chiu@csail.mit.edu (H.-P. Chiu).

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

to detect instances of the object class in the novel viewpoint. In our experiments, we have trained a recognizer for a novel view of an object with 100 virtual images transformed from 100 training examples in other views; that recognizer, even though it had never seen an actual image of an object of this class from this view, performs as well as the same recognition algorithm trained on 50 actual images from this view.

In the following section we provide an overview of related work. Then in Section 3 we sketch a basic version of the Potemkin model, which uses only a single oriented primitive for learning the transforms from view to view in the first phase. These transforms are then used as the basis for transforms of all parts in the model. This basic model is ultimately too weak to describe many object classes well. Section 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 transforms between pairs of views. In Section 5 we provide a self-supervised labeling method that obviates the need for most of part labeling in the real training data. Section 6 demonstrates that these extensions significantly outperform the basic model and enable existing view-dependent detection systems to achieve the same level of performance with many fewer real training images required.

2. Related work

There has been a great deal of work on modeling and recognition for single 3D objects (e.g., [1–4]), but these methods are not designed to cover the variability in shape and appearance among instances of a class and cannot be readily generalized for object class recognition. There is also a long tradition of representing objects as arrangements of 3D building blocks (e.g., [5–11]). However, it has been difficult to achieve robust recognition performance in real images using these approaches.

Most recent advances in object class recognition attempt to detect examples of a target class from only a single viewpoint [12–22]. Our work leverages such advances, and enables them to be applied to multi-view recognition without excessive training-data requirements.

There is an existing body of work on multi-view object class recognition for specific classes such as cars and faces [23–28]. For example, Everingham and Zisserman [29] generate virtual training data for face recognition from multiple viewpoints using a reconstructed 3D model of the head.

For object recognition, more generally, there are a number of effective multi-view object class recognition systems that are constructed from several independent single-view detectors. Crandall and Huttenlocher [30] treat ranges of poses as separate classes, with a separate model learned for each. Torralba et al. [31] show that sharing features among models for multiple views of the same object class improves both running time and recognition performance. Leibe et al.'s detection system [32] for cars combines a set of seven view-dependent ISM detectors [33], each of which requires hundreds of training examples to be effective.

There is increased recent interest in methods that make deeper use of the relationship between views of the same class, via a range of approaches.

One approach is to link regions of the same or similar training instances across different viewpoints, forming a multi-view class model for recognition. Thomas et al. [34] construct separate view-dependent detectors for each of the viewing directions, and track regions of the same instance across different viewpoints. They use these integrated view-dependent detectors to improve detection, by transferring information from one viewpoint to another. Savarese and Fei-Fei [35] also relate multiple views by finding correspondences among 2D regions, formed by feature grouping, on the same instance across different views. Compared to the work of Thomas et al, these models are more compact and can be learned with very weak supervision. Rather than constructing multiple view-dependent detectors, Kushal et al. [36] 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 instances across different views. 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.

An alternative approach is to construct explicit 3D shape models of object classes. Hoeim et al. [37] create a coarse 3D model from mean segmentations of two views of training instances, and avoid the need for multiple views of the same training object instance during learning. Yan et al. [38] use training images, taken from a dense sampling of viewpoints around a specific object, to reconstruct a 3D model for the target class. Both methods form correspondences between 2D features across training instances at arbitrary viewpoints by projecting the surfaces of the 3D model onto each instance. Thus, the features can be shared across viewpoints through 3D correspondence. However, these systems still require many training instances at many viewpoints to achieve satisfactory recognition results. To avoid collecting real training images, Liebelt et al. [39] use a database of 3D CAD models of objects of the target class, and construct feature descriptors on synthetic images generated from these 3D models. However, 3D CAD models with realistic textures are not easily obtained for some object classes.

The Potemkin model [40] is intermediate between the approaches based on cross-view constraints and those based on detailed 3D models. 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 as few as two labeled images, but is powerful enough to generate virtual examples in novel viewpoints, amplifying the effectiveness of any method that needs images from multiple views.

3. Basic shape model

The Potemkin model is used to represent the approximate 3D shape and relationship among views of a class of objects. In this section we describe the basic version of the Potemkin model in detail, including how to estimate the parameters from a small number of training images, and use it to generate virtual training images in novel views. The basic model is ultimately too weak to describe many classes of objects well, because it has only a single primitive part shape. We will extend it to include multiple primitive shapes and self-occlusion in Section 4. In addition, the training algorithm for the basic model requires the outlines of the parts in all real training images to be labeled. We will describe a simple approach that obviates the need for most of this labeling but still requires a decomposition into parts in Section 5.

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. This paper makes assumption that all view bins can be transformed to all others. If the portion of the view sphere covered by these bins is large, it will necessary to limit transformations of images in one bin to nearby bins in which nearby the same set of parts is visible.

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*, $P_\alpha \in R^{2\times3}$, 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*, $T^j_{\alpha,\beta} \in R^{3\times3}$, 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 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.

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 model is estimated in two phases: 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 and aligned orientation, as depicted in Fig. 1. 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 [41] 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 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 [42] as

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

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: $[\log s, \theta, K_a, K_b, v_1, v_2]$. We use the mean of these parameters over 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.

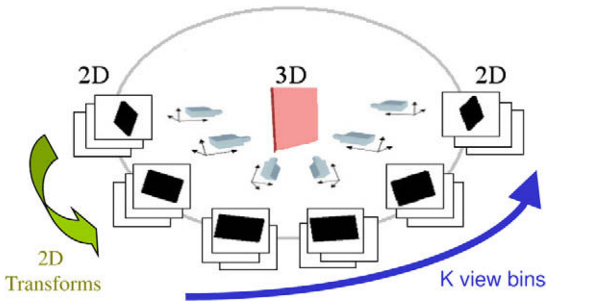

Fig. 1. A synthetic nearly-planar box object used for learning the generic view transformations.

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 $T^j_{\alpha,\beta}$:

$$T^j_{\alpha,\beta} = \frac{1}{\kappa + m} \sum_{i=1}^{m} \overline{T}^{ij}_{\alpha,\beta} + \frac{\kappa}{\kappa + m} T_{\alpha,\beta}. \tag{2}$$

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 [43]:

$$S_j = \left(\sum_\alpha P^T_\alpha (\Sigma^j_\alpha)^{-1} P_\alpha \right)^{-1} \left(\sum_\alpha P^T_\alpha (\Sigma^j_\alpha)^{-1} \mu^j_\alpha \right), \tag{3}$$

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

Fig. 2 shows a schematic version of this process. In each view bin, the distribution on 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 recognizers. 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 recognizer. This strategy effectively multiplies the value of each training image, and allows us to train recognizers for view bins that have 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 determined, which can be easily combined with the

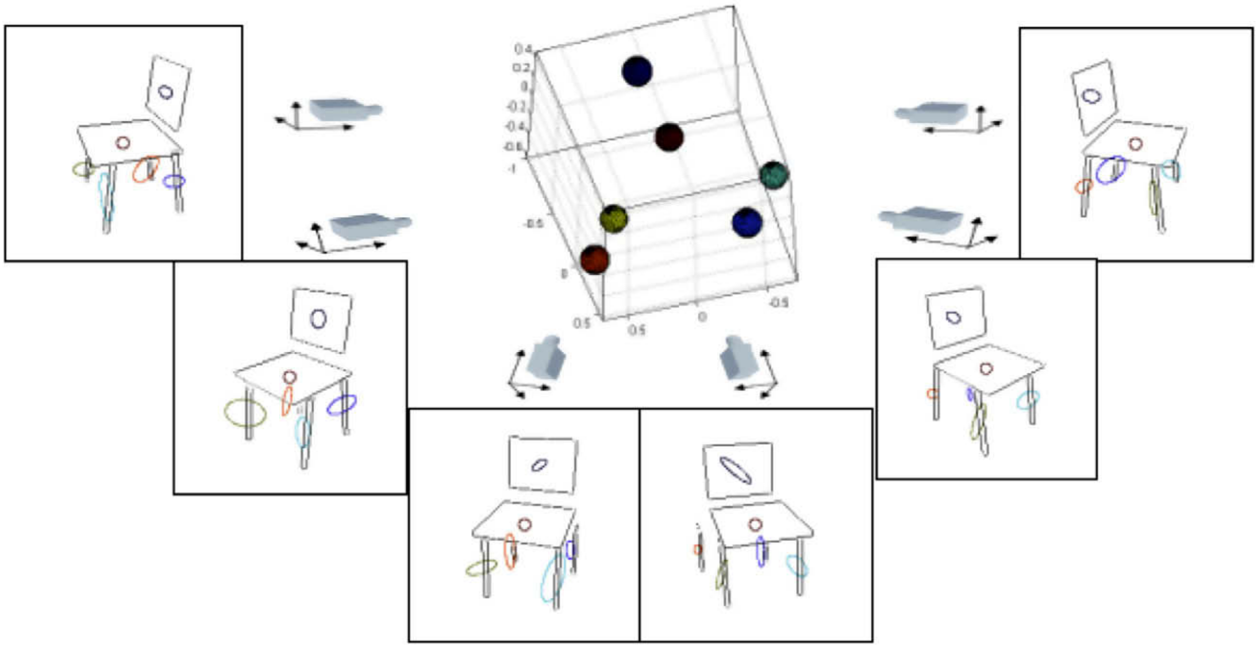

Fig. 2. The ellipses indicate the distribution of 2D positions of the centroids of each part in the training set of chair images for each viewpoint. The 3D spheres show the estimate of the skeleton positions S_j for each part.

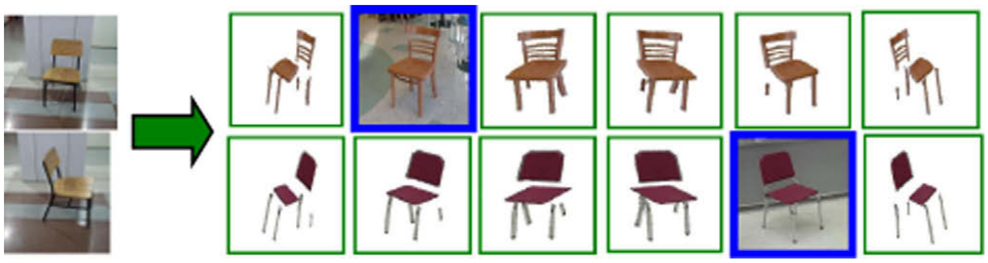

Fig. 3. A basic Potemkin model is constructed from the two real labeled images on the left. Given the label images for two object instances, each in only one viewpoint bin (highlighted), the other virtual views are generated from the model.

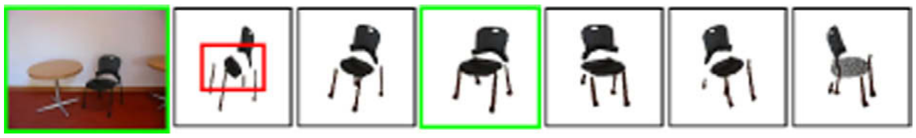

Fig. 4. Virtual views generated from the highlighted image using a basic Potemkin model that is minimally trained.

background of the original image to generate a complete image in the new view.

An example of this process can be seen in Fig. 3, which shows virtual images constructed from a basic Potemkin model, trained with only two real labeled images. Even with just two training images, the results are useful for recognition, as shown in Section 6. As the amount of training data is increased, the quality of the transformed images improves [40].

4. 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.

Another example of results under these minimal training conditions is shown in Fig. 4. This process can in some cases be very effective, but in other cases the transformed parts are distorted in the new view. The problem highlighted by the red rectangle in Fig. 4 arises because we are 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 us to make much better predictions of appearance in unseen views.

4.1. 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 Fig. 5, we consider two rotational degrees of freedom: *azimuth* and *elevation* angles of rotation about the shape's centroid. This allows us to ignore irrelevant rotations of in-plane rotation of nearly planar shapes and rotations about the long axis of nearly linear shapes. It 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 the shape primitive as well as the particular views chosen from the view bins.

4.2. 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

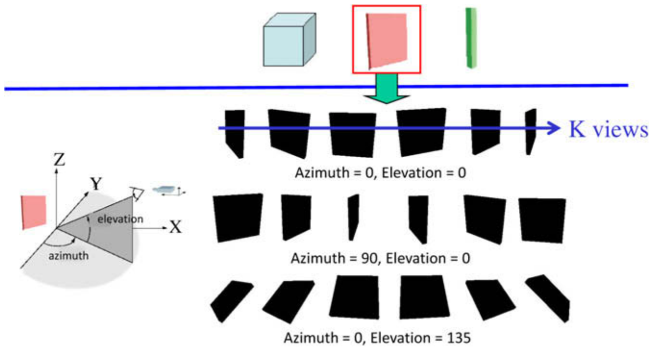
K views
Azimuth = 0, Elevation = 0
Z
Y
X
elevation
azimuth
Azimuth = 90, Elevation = 0
Azimuth = 0, Elevation = 135

Fig. 5. 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.

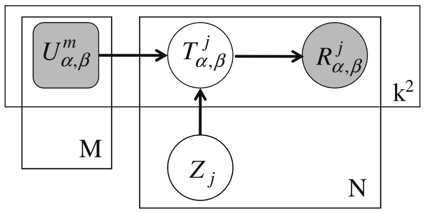

Fig. 6. Generative model for 2D part-specific transforms. There are plates for the M possible oriented primitives, for the N parts of the object model, and for the k^2 view bin pairs. $U^m_{\alpha,\beta}$ and $T^j_{\alpha,\beta}$ represent the view transform from view bin α to β for oriented primitive 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 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 Fig. 6. 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 ($\overline{U}$ 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 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 $Z_j = m$,

$$T^j_{\alpha,\beta} \sim \mathcal{N}(\overline{U}^m_{\alpha,\beta}, \sigma^{U^m}_{\alpha,\beta}). \tag{4}$$

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:

$$R^j_{\alpha,\beta} \sim \mathcal{N}(T^j_{\alpha,\beta}, \sigma_R). \tag{5}$$

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 [44] as:

$$\overline{T}^{j,m}_{\alpha,\beta} = \frac{\kappa_0}{\kappa_0+1}\overline{U}^m_{\alpha,\beta} + \frac{1}{\kappa_0+1}R^j_{\alpha,\beta} = \frac{\frac{\sigma_R^2}{\sigma^{U^{m2}}_{\alpha,\beta}}}{\frac{\sigma_R^2}{\sigma^{U^{m2}}_{\alpha,\beta}}+1}\overline{U}^m_{\alpha,\beta} + \frac{1}{\frac{\sigma_R^2}{\sigma^{U^{m2}}_{\alpha,\beta}}+1}R^j_{\alpha,\beta}. \tag{6}$$

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

$$\Pr(\overline{T}^{j,m}, Z_j = m|U, R^j) \propto \Pr(R^j|\overline{T}^{j,m})\Pr(\overline{T}^{j,m}|U^m, Z_j = m)\Pr(Z_j = m) \tag{7}$$

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

4.3. Learning and using an object class model

Training the Potemkin model with multiple primitives 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 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.

Fig. 7 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 recognition, 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 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. Fig. 8 shows how understanding self-occlusion can improve the transformed images.

By increasing the set of primitives and supporting reasoning about occlusion, the Potemkin model with multiple primitives generates significantly more realistic virtual images compared to those generated by the basic Potemkin model, as shown in Fig. 9. Note that in both cases, the models are trained with only two real labeled images.

5. 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 [45], part-labeled images are not.

We have developed a simple approach that obviates the need for most of this labeling; we require hand part-labeling on only a single pair of 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 [41] to match

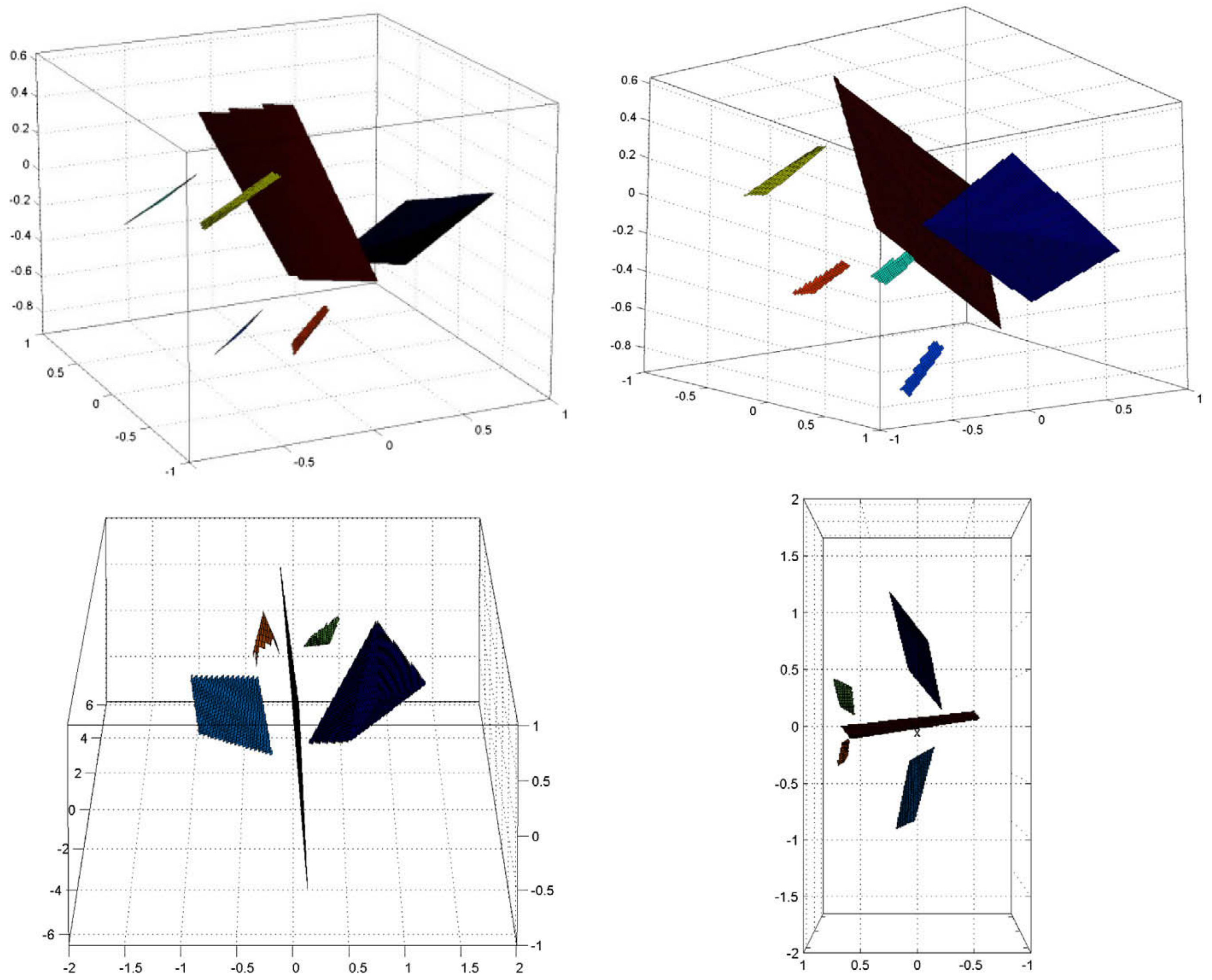

Fig. 7. Visualization of 3D skeleton and shape primitives. Top: two views of the four-legged chair model; bottom: two views of the airplane model. Note that these models were each constructed from two part-labeled images of the same object, knowing the view bins but with no further camera calibration.

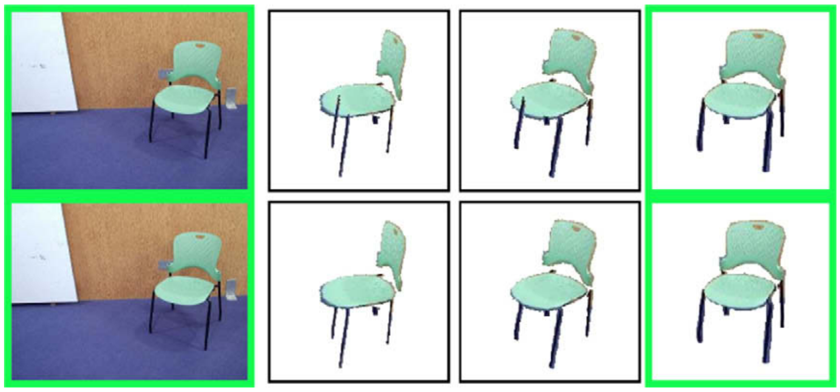

Fig. 8. First row: virtual images generated from highlighted instance without occlusion handling; second row: with occlusion handling.

the boundary shapes and deform the boundaries of the labeled instance into the unlabeled instances, as shown in Fig. 10. The deformed part boundary encloses a region in the new image, which is labeled as the corresponding part.

6. Experimental results

In this section, we report a number of experiments aimed at evaluating different aspects of the performance of the Potemkin

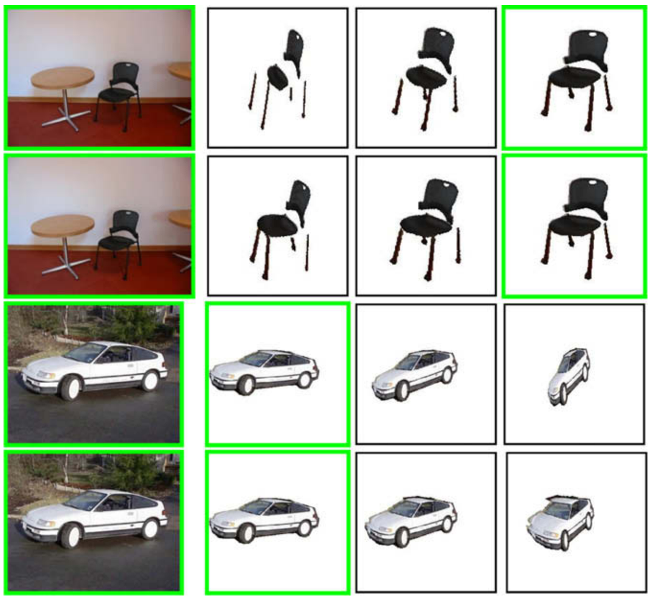

Fig. 9. The Potemkin model with multiple primitives (second and fourth rows) generates more realistic virtual images than the basic Potemkin model (first and third rows).

model. We also demonstrate that the virtual training examples generated by the Potemkin model can be used effectively to train an existing view-dependent detection system [15], with the result that many fewer real training images are required to reach the same level of performance.

6.1. Dataset

We used four different object classes, shown in Fig. 11, 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 262 images, respectively. The chair data are discretized into six view bins (Fig. 4) and the other data sets into four (Fig. 12). The bicycle data set is from the TU-Graz-02 database, which is part of the PASCAL Visual Object Classes (VOC) Challenge [46]. It contains 365 images of bicycles. The car data set contains 243 images, which were collected by our research group and 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 [45] 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.

6.2. 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.

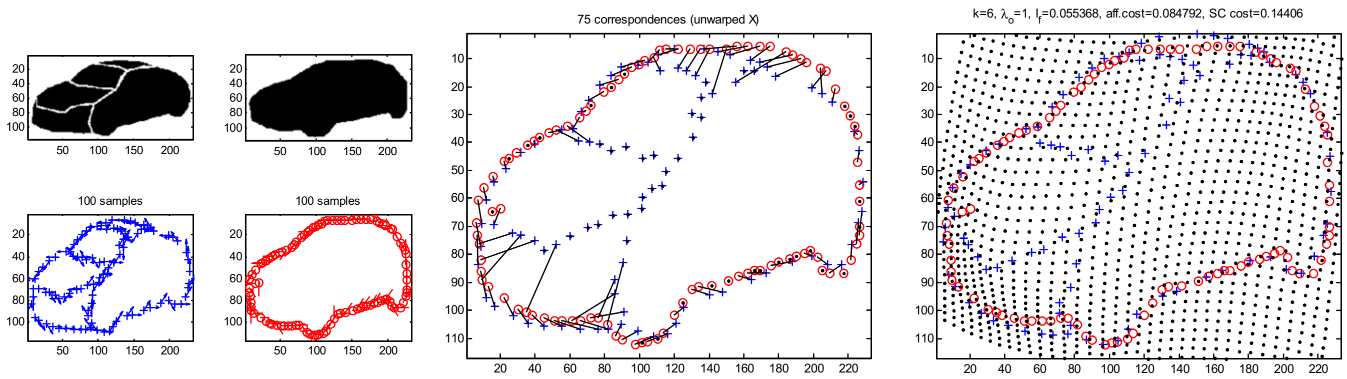

Fig. 10. Given a model instance with labeled parts (blue), the parts of another instance (red) in the same view can be found by matching points along the boundaries of the instances (middle) and by deforming the model instance into the target instance (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

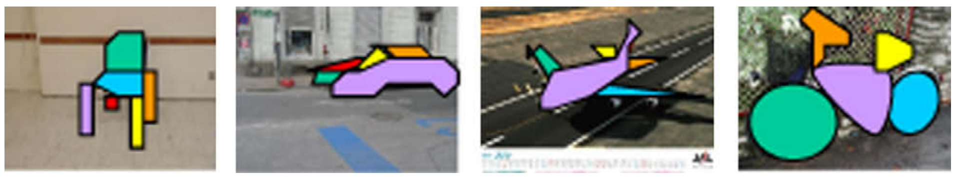

Fig. 11. Decomposition of object classes into parts.

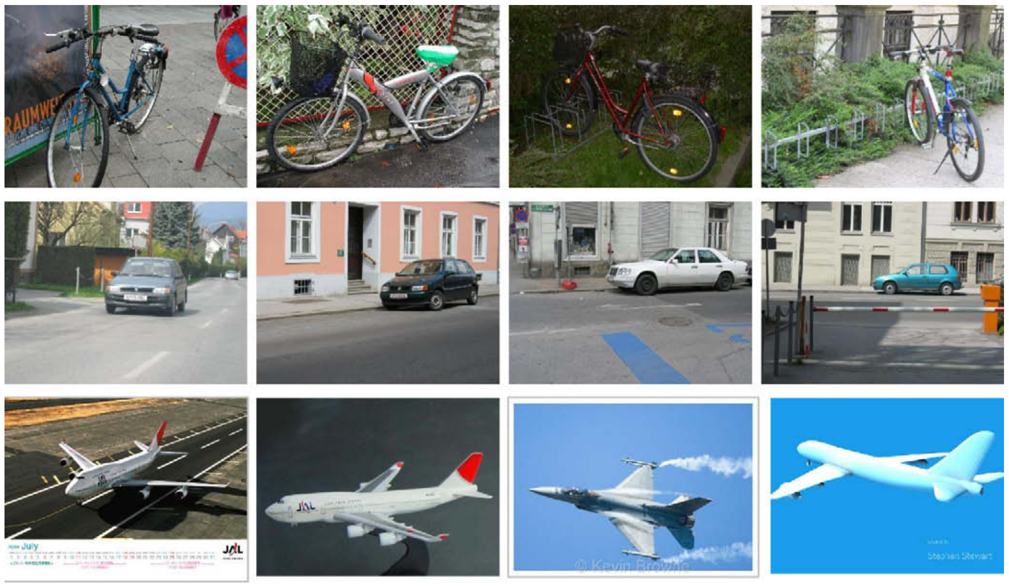
Fig. 12. Four selected view bins for bicycles, cars, and airplanes.

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

$$score(m) = \prod_c \prod_j \prod_{\alpha,\beta} \Pr\left(R^j_{\alpha,\beta} \middle| \overline{T}^{j,m}_{\alpha,\beta}\right), \quad (8)$$

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:

$$\frac{1}{NK^2} \sum_j \sum_{\alpha,\beta} overlap\left(\overline{T}^{j,Z_j}_{\alpha,\beta} A_j, B_j\right), \quad (9)$$

where A_j is the region associated with part j in view α, $\overline{T}^{j,Z_j}_{\alpha,\beta} A_j$ 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. Fig. 13 (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 (Fig. 13, 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. This restricted domain of four primitives makes our modeling process more efficient, and prevents the tendency to overfit when there are a huge number of primitives available. 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 Fig. 14. 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.

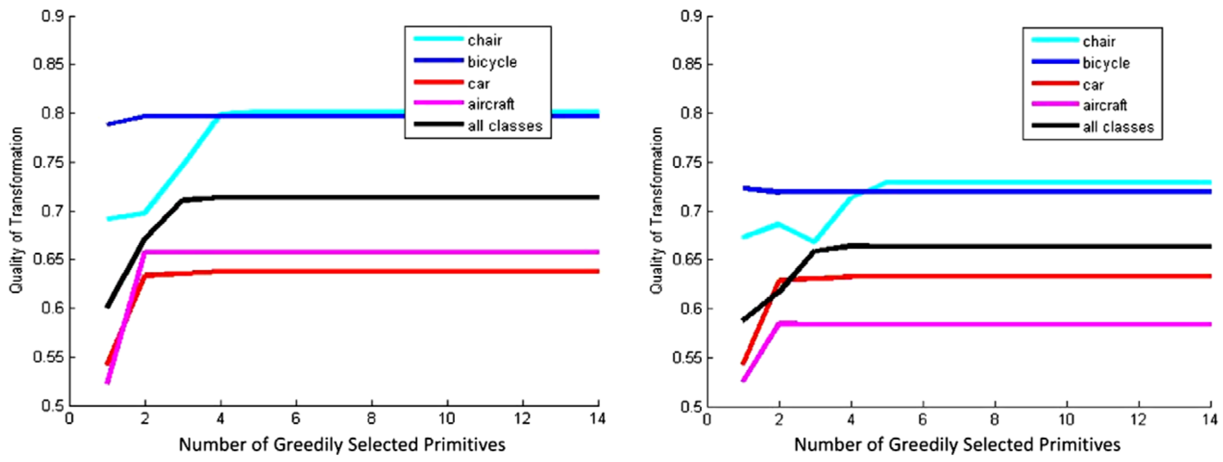

Fig. 13. The quality of transforms as the number of available primitives (M) increases. Left: evaluated on training data; right: evaluated on held-out test data.

6.3. 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.

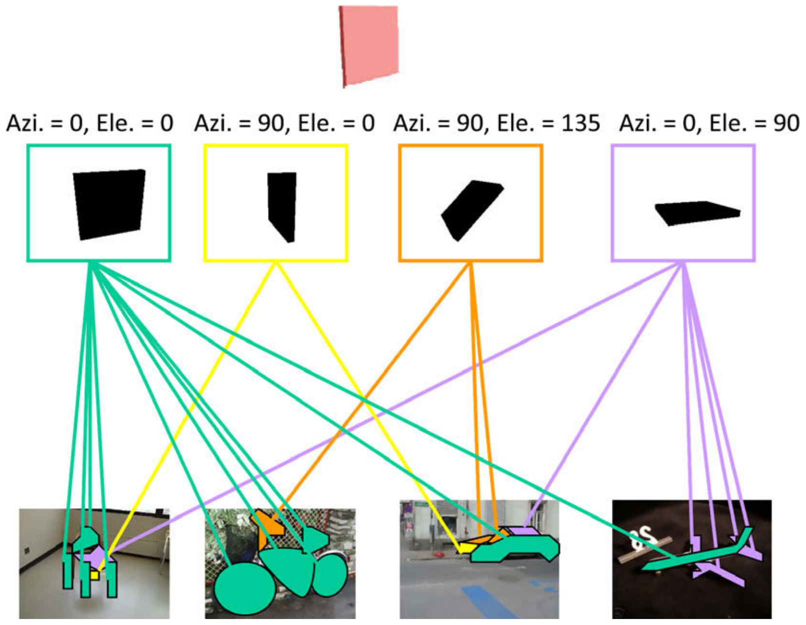

Fig. 14. 3D shape primitives selected for each part of each class.

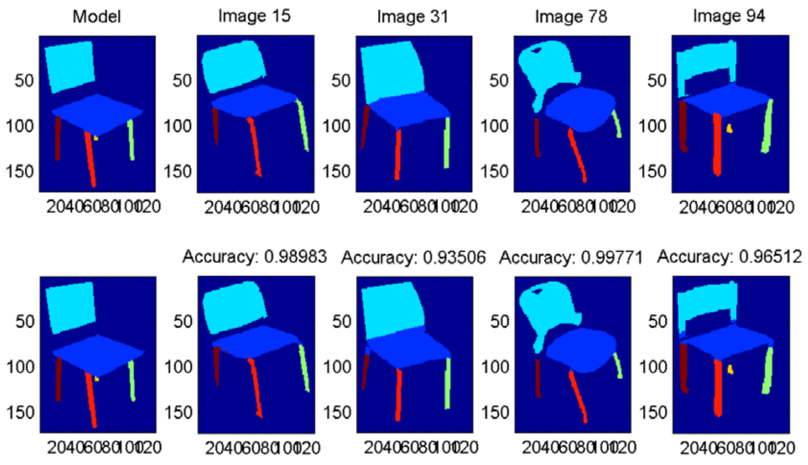

Fig. 15. Some part labeling results for four-legged chairs. First row: hand labeling; second row: self-supervised labeling.

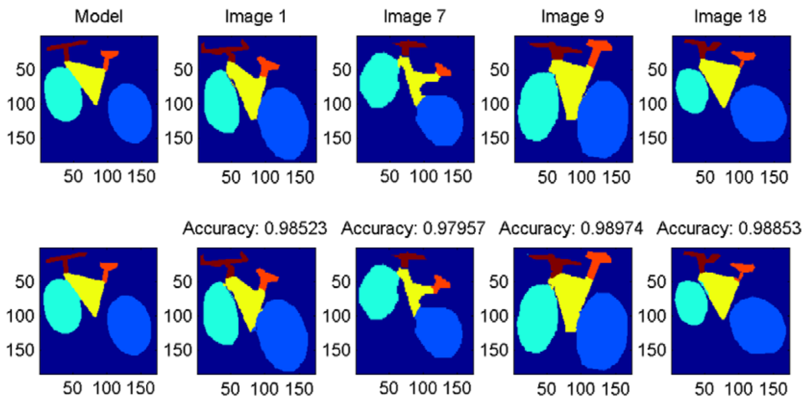

Fig. 16. Some part labeling results of bicycles. First row: hand labeling; second row: self-supervised labeling.

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 Figs. 15 and 16.

On the other hand, the label results for cars are less accurate, as shown in Fig. 17. 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 recognition performance. We will also compare the performance of these two labeling strategies in the end-to-end system in the following sections.

6.4. Single-view detection

The Potemkin model is useful both for recognizing objects at previously unseen views and for leveraging sparse training data by transforming any training image into a view that is suitable for training any view-dependent recognizer [40]. In this set of experiments we evaluate the quality of virtual training examples

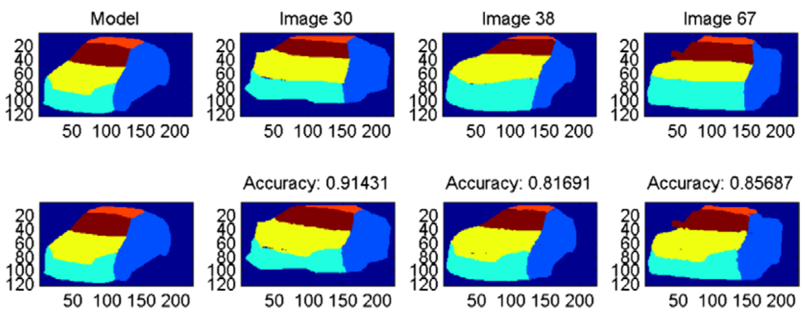

Fig. 17. Some part labeling results of cars. First row: hand labeling; second row: self-supervised labeling.

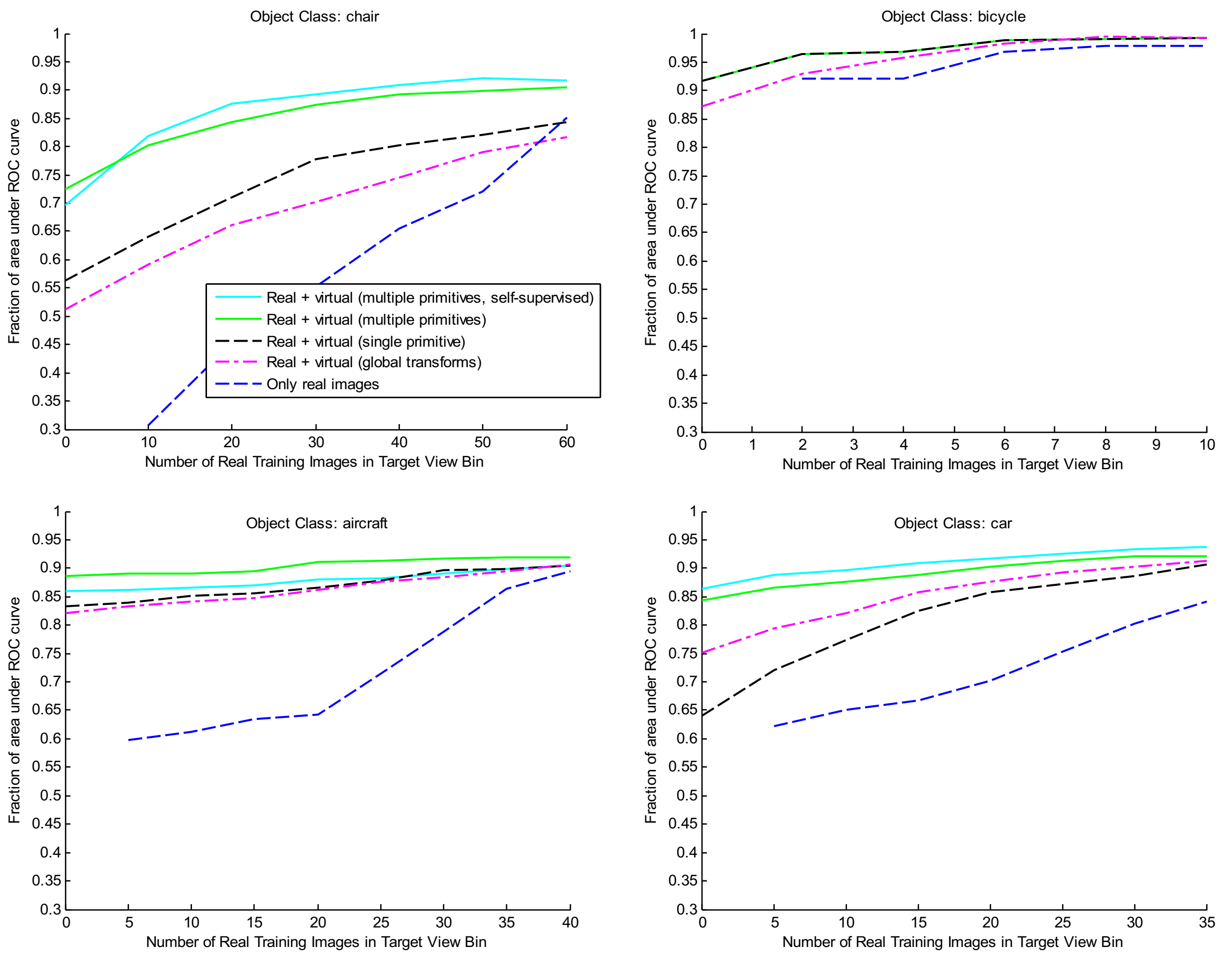

Fig. 18. Average single novel view detection results over four repetitions of each of four object classes: chairs (left-top), bicycles (right-top), airplanes (left-bottom), and cars (right-bottom).

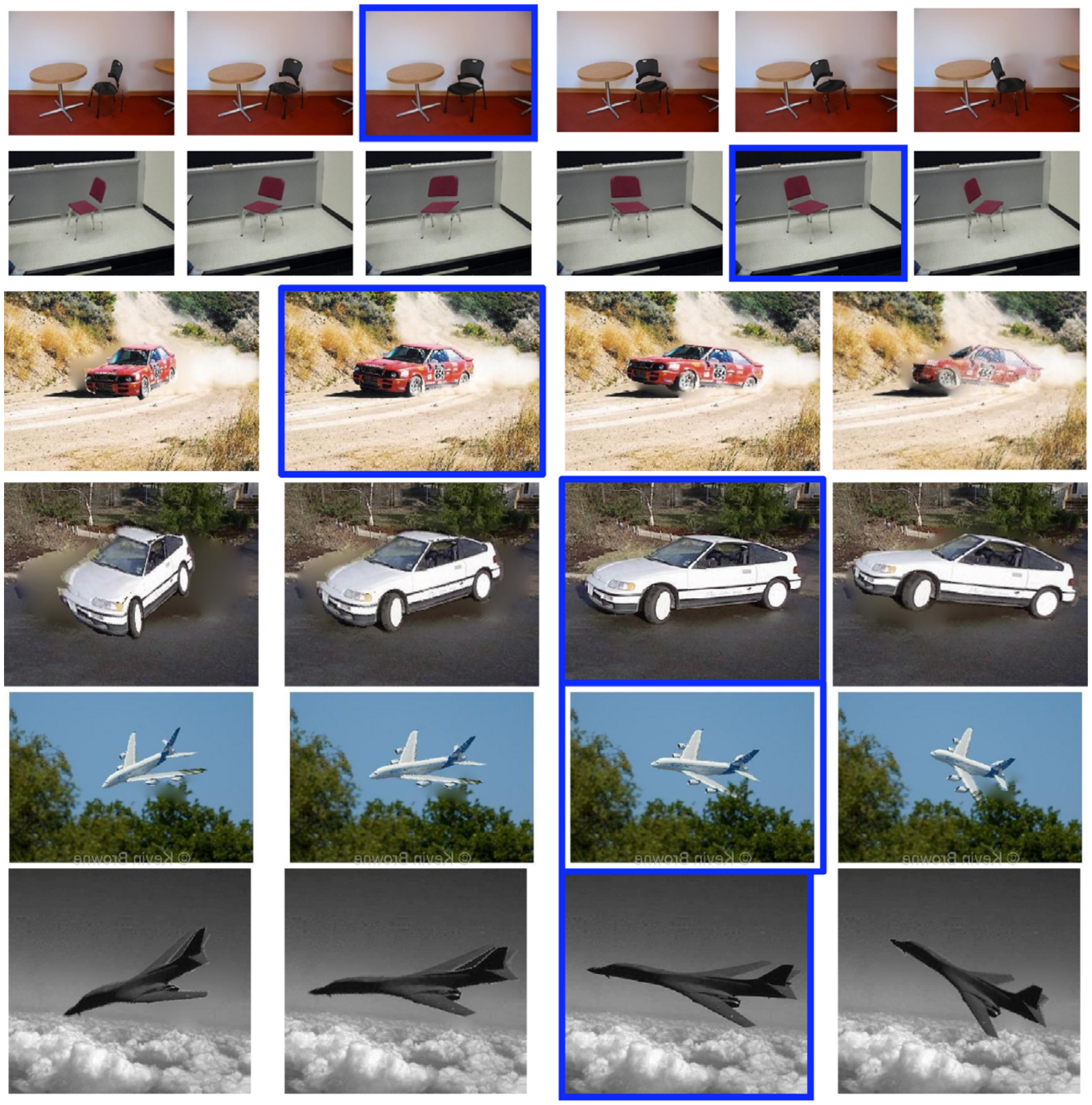

Fig. 19. Six examples of real images (highlighted) and virtual images constructed by the Potemkin model with multiple primitives.

generated using Potemkin models. We test them in single-view and multi-view detection tasks using a publicly available view-dependent recognizer developed by Crandall et al. [15]. The goal of our experiments is to show that these virtual training images are almost as effective as input for a standard 2D recognition 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 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 previously [40]. For efficiency, all recognition was done at a single scale.

We trained the single-view recognizer 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 Fig. 18 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.
- 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 Potemkin model), using hand-labeled parts.
- Virtual images generated with multiple primitives (the 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 recognizer for the target class. The basic Potemkin model and the Potemkin model with multiple primitives were 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 Potemkin model with multiple primitives 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. However, the Potemkin model with multiple primitives 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 Potemkin model with multiple primitives are shown in Fig. 19.

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 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.

6.5. 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 recognizers for each view of each object, using virtual training data generated by the Potemkin model, and output the class associated with the recognizer 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 Fig. 20. The combined recognizers trained with the data generated from the Potemkin model with multiple primitives outperform the others by a substantial margin.

6.6. 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 [47], 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 [46] and PASCAL VOC Challenge 2007 Car Test data set, which are two standard data sets 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 does not 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 Fig. 12, 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.

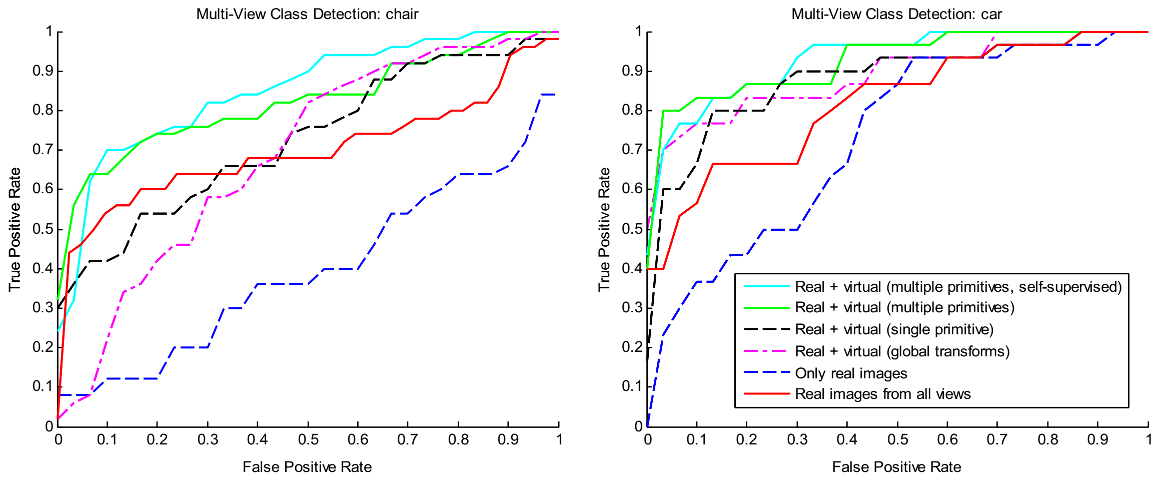

Fig. 20. Multi-view recognition performance (ROC curve) for chairs and cars.

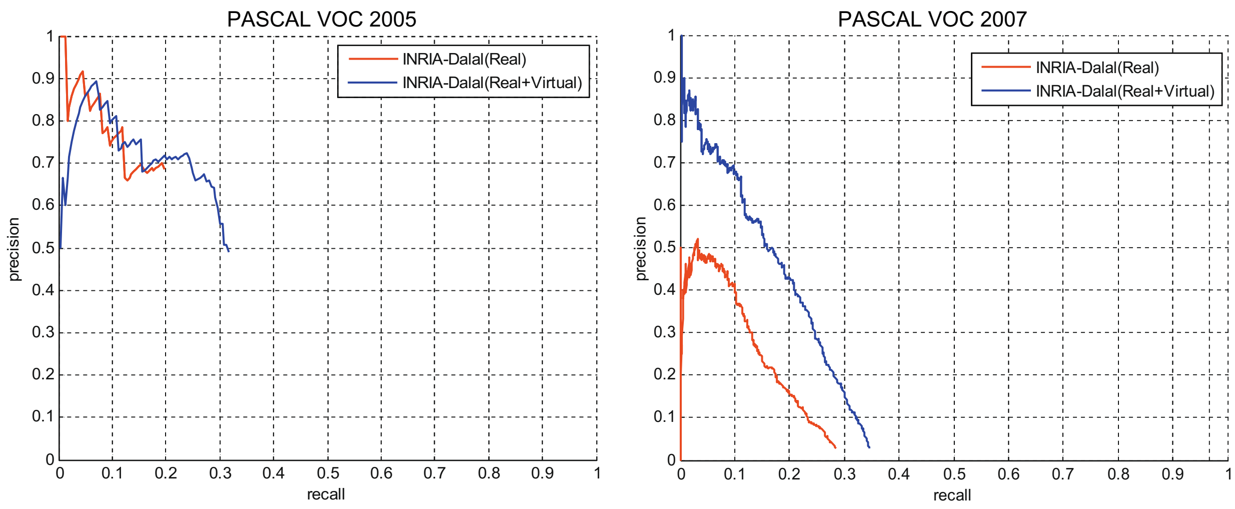

Fig. 21. The precision–recall curves of the Dalal–Triggs detector using our training images on PASCAL Challenge 2005 Car Test 2 (left) and PASCAL Challenge 2007 Car Test (right).

We set the parameters of the Dalal–Triggs detector and evaluated the detection results as in Ref. [46]. Note the best three average precision (AP) scores on this standard test set in PASCAL VOC Challenge 2005 [46] 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 does not cover all the viewing directions of cars used in Ref. [46]. Thus, for example, our trained detector cannot recognize 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.

Fig. 21 shows the precision–recall curves of the Dalal–Triggs detector using our training data set on both the VOC Challenge 2005 Car Test 2 data set and VOC Challenge 2007 Car Test data set. The incorporation of our virtual training images increased the AP score of the same detector from 0.162 to 0.272 and from 0.098 to 0.206, respectively. Using both real and virtual images for training, more cars in the test set were detected (highest recall: 0.32 for PASCAL 2005 and 0.35 for PASCAL 2007).

7. Conclusions

In this paper, we have proposed an approach to multi-view object class recognition based on independent view-dependent recognizers. Our approach makes efficient use of training data in each view by transforming it into virtual training data for all other views. The virtual training examples we generate can be used as training input for any view-dependent 2D recognizer. We have demonstrated the effectiveness of this approach in learning single-view recognizers in novel views and in multi-view recognition.

This approach 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 that as leverage for efficient and effective learning of view-based recognizers.

The demonstrations in the paper have been for a set of view bins that are significantly different, but that do not exhaust the entire view sphere. In order to handle such a wide variety of views, we would need to add at least two extensions. First, explicit modeling of occlusion during the skeleton-learning phase so that largely occluded views of parts do not contribute to the estimation of the 3D structure. Second, reasoning about how the viewpoint change influences the quality of the transformed virtual images. For some view pairs, the transformed images will be excellent; but as the views become very different, their quality will degrade. When asked to generate virtual images for a new view, we should only use a subset of related views as "source" images for the transformation. With these extensions, we believe the Potemkin model can serve as the basis for a robust, efficient strategy for multi-view object class recognition.

8. Acknowledgements

This work was supported under DARPA IPTO Contract FA8750-05-2-0249, "Effective Bayesian Transfer Learning".

References

[1] M. Vrown, D. Lowe, Unsupervised 3d object recognition and reconstruction in unordered datasets, in: 3D Imaging and Modeling, 2005.
[2] V. Ferrari, T. Tuytelaars, L. Van Gool, Simultaneous object recognition and segmentation from single or multiple model views, International Journal of Computer Vision (2006).
[3] D. Lowe, Object recognition from local scale-invariant features, in: Proceedings of International Conference on Computer Vision, 1999, pp. 1150–1157.
[4] F. Rothganger, S. Lazabnik, C. Schmid, J. Ponce, 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints, Internationl Journal of Computer Vision 66 (2006) 231–259.
[5] L.G. Roberts, Machine perception of three-dimensional solids, Technical Report 315, Lincoln Laboratory, 1963.
[6] D. Marr, H.K. Nishihara, Representation and recognition of the spatial organization of three dimensional structure, Proceedings of the Royal Society of London, B 200 (1978) 269–294.
[7] D. Marr, Vision, W.H. Freeman, San Francisco, 1982.
[8] I. Biederman, Recognition-by-components: a theory of human image understanding, Psychological Review. 94 (1987) 115–147.
[9] I. Biederman, E.E. Cooper, Priming contour-deleted images: evidence for intermediate representations in visual object recognition, Cognitive Psychology 23 (1991) 393–419.
[10] A. Pentland, Recognition by parts, in: Proceedings of International Conference on Computer Vision, 1987, pp. 612–620.
[11] G. Prasanna, G. Mulgaonkar, L.G. Shapiro, R.M. Haralick, Matching sticks, plates and blocks objects using geometric and relational constraints, Image Vision Computation 2 (1984) 85–98.
[12] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, in: Proceedings of Computer Vision and Pattern Recognition, 2003, pp. 264–271.
[13] R. Fergus, P. Perona, A. Zisserman, A sparse object category model for efficient learning and exhaustive recognition, in: Proceedings of Computer Vision and Pattern Recognition, 2005.
[14] M.P. Kumar, P.H.S. Torr, A. Zisserman, Extending pictorial structures for object recognition, in: Proceedings of British Machine Vision Conference, 2004.
[15] D. Crandall, P.F. Felzenszwalb, D.P. Huttenlocher, Spatial priors for part-based recognition using statistical models, in: Proceedings of Computer Vision and Pattern Recognition, 2005.

[16] D. Crandall, D.P. Huttenlocher, Weakly supervised learning of part-based spatial models for visual object recognition, in: Proceedings of European Conference on Computer Vision, 2006.
[17] A. Berg, T. Berg, J. Malik, Shape matching and object recognition using low distortion correspondences, in: Proceedings of Computer Vision and Pattern Recognition, 2005.
[18] C. Dance, J. Willamowski, L. Fan, C. Bray, G. Csurka, Visual categorization with bags of keypoints, in: European Conference on Computer Vision Workshop on Statistical Learning in Computer Vision, 2004.
[19] K. Grauman, T. Darrell, The pyramid match kernel: discriminative classification with sets of image features, in: Proceedings of International Conference on Computer Vision, 2005.
[20] B. Leibe, B. Schiele, Combined object categorization and segmentation with an implicit shape model, in: Workshop on Statistical Learning in Computer Vision, 2004.
[21] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of Computer Vision and Pattern Recognition, 2001, pp. 511–518.
[22] M. Weber, M. Welling, P. Perona, Unsupervised learning of models for recognition, in: Proceedings of European Conference on Computer Vision, 2000, pp. 101–108.
[23] E. Bart, E. Byvatov, S. Ullman, View-invariant recognition using corresponding object fragments, in: Proceedings of European Conference on Computer Vision, 2004.
[24] Z.G. Fan, B.L. Lu, Fast recognition of multi-view faces with feature selection, in: Proceedings of International Conference on Computer Vision, 2005.
[25] S. Li, Z. Zhang, Floatboost: learning and statistical face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 1112–1123.
[26] J. Ng, S. Gong, Multi-view face detection and pose estimation support vector machine across the view sphere, in: Workshop on Recognition, Analysis and Tracking of Faces and Gestures, 1999.
[27] H. Schneiderman, T. Kanade, A statistical method for 3D object detection applied to faces and cars, in: Proceedings of Computer Vision and Pattern Recognition, 2000.
[28] M. Weber, W. Einhaeuser, M. Welling, P. Perona, Viewpoint-invariant learning and detection of human heads, in: Conference on Automatic Face and Gesture Recognition, 2000.
[29] M. Everingham, A. Zisserman, Identifying individuals in video by combining generative and discriminative head models, in: Proceedings of International Conference on Computer Vision, 2005, pp. 1103–1110.
[30] D. Crandall, D.P. Huttenlocher, Composite models of objects and scenes for category recognition, in: Proceedings of Computer Vision and Pattern Recognition, 2007.
[31] A. Torralba, K.P. Murphy, W. Freeman, Sharing visual features for multiclass and multiview object detection, in: Proceedings of Computer Vision and Pattern Recognition, 2004.
[32] B. Leibe, N. Cornelis, K. Cornelis, L. Van Gool, Dynamic 3D scene analysis from a moving vehicle, in: Proceedings of Computer Vision and Pattern Recognition, 2007.
[33] B. Leibe, E. Seemannand, B. Schiele, Pedestrian detection in crowded scenes, in: Proceedings of Computer Vision and Pattern Recognition, 2005.
[34] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, L. Van Gool, Towards multi-view object class detection, in: Proceedings of Computer Vision and Pattern Recognition, 2006.
[35] S. Savarese, L. Fei-Fei, 3D generic object categorization, localization and pose estimation, in: Proceedings of International Conference on Computer Vision, 2007.
[36] A. Kushal, C. Schmid, J. Ponce, Flexible object models for category-level 3D object class recognition, in: Proceedings of Computer Vision and Pattern Recognition, 2007.
[37] D. Hoeim, C. Rother, J. Winn, 3D LayoutCRF for multi-view object class recognition and segmentation, in: Proceedings of Computer Vision and Pattern Recognition, 2007.
[38] P. Yan, M. Khan, M. Shan, 3D model based object class detection in arbitrary view, in: Proceedings of International Conference on Computer Vision, 2007.
[39] J. Liebelt, C. Schmid, K. Schertler, Viewpoint-independent object class detection using 3d feature maps, in: Proceedings of Computer Vision and Pattern Recognition, 2008.
[40] H. Chiu, L.P. Kaelbling, T. Lozano-Perez, Virtual training for multi-view object class recognition, in: Proceedings of Computer Vision and Pattern Recognition, 2007.
[41] G. Mori, S. Belongie, J. Malik, Shape contexts enable efficient retrieval of similar shapes, in: Proceedings of Computer Vision and Pattern Recognition, 2001.
[42] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.
[43] R. Hartley, F. Schaffalitzky, PowerFactorization: 3D reconstruction with missing or uncertain data, in: AJAWCV, 2003.
[44] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Rubin, Bayesian Data Analysis, second ed. Chapman and Hall/CRC, 2004.
[45] B.C. Russell, A. Torralba, K.P. Murphy, W.T. Freeman, LabelMe: a database and web-based tool for image annotation, International Journal of Computer Vision (2007).
[46] M. Everingham, A. Zisserman, C. Williams, L. Van Gool, The PASCAL visual object classes challenge 2005 (VOC2005) results, in: 1st PASCAL Challenges Workshop, 2006.
[47] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proc. CVPR, 2005.