Virtual Training for Multi-View Object Class Recognition

Han-Pang Chiu Leslie Pack Kaelbling Tomás Lozano-Pérez
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
{chiu,lpk,tlp}@csail.mit.edu

Abstract

Our goal is to circumvent one of the roadblocks to using existing approaches for single-view recognition for achieving multi-view recognition, namely, the need for sufficient training data for many viewpoints. We show how to construct virtual training examples for multi-view recognition using a simple model of objects (nearly planar facades centered at fixed 3D positions). We also show how the models can be learned from a few labeled images for each class.

1. Introduction

Seeing a number of instances of an object class from different views ought to enable a vision system to recognize other instances of that class from 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 recognizing multiple independent object classes, with a separate model learned for each. This independent-view approach can be made to work well 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 alternative strategy 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 recognition of many entirely novel views. This is a very difficult problem, which is as yet unsolved.

In this paper, we take an intermediate approach. We define the *Potemkin model* of 3-dimensional objects as a collection of *parts*. 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 model that would be necessary for predicting the transformation of each part shape from one view to another, we model the change in views of each shape as a 2D perspective transformation.

The Potemkin model is trained and used in two phases. In the first phase, multiple views of some simple object instances, such as boxes, are used to learn fundamental transforms from view to view. The necessary data can be relatively simply acquired from real or synthetic image sequences of a few objects rotating through the desired space of views. In the second phase, a collection of views of different object instances of the target class, from arbitrary views, is used to estimate the 3D skeleton of the object class and tune the view transforms for each of the parts. Given a new view of interest, the skeleton can easily be projected into that view, specifying the 2D centroids of the parts. Then, the 2D part images of the training examples in other views can be transformed into this view using the transforms learned in the initial phase. This gives us a method for generating "virtual training examples" of what it would be like to see this object from a novel view. These virtual training examples, along with any real training examples that are available, can then be fed into any view-dependent 2D part-based recognition system, which can be used to look for instances of the object class in the novel viewpoint.

The first phase of the model need only be carried out once, and it can be done entirely off-line. After that, the model can be used for new object classes with very little overhead. In this paper, we demonstrate that the model is useful both for recognizing objects at previously unseen views and for leveraging sparse training data by transforming every training instance into a view that is suitable for training every view-dependent recognizer. We show results using both Crandall et al's [2] recognition system in a detection task and a recognition system we will describe below in a localization task.

Related Work. This work leverages recent advances in single-view object recognition for objects modeled as a collection of parts and spatial relationships [8, 9, 12, 2, 3] and enables them to be applied to multi-view recognition without excessive training-data requirements. There is an exist-

1-4244-1180-7/07/$25.00 ©2007 IEEE

ing body of work on multi-view object class recognition for specific classes such as cars and faces [1, 6, 13, 15, 16, 19]. In particular, Everingham and Zisserman [4] generated virtual training data for face recognition from multiple viewpoints using a reconstructed 3D model of the head. Two important contributions to general multi-view object class recognition are the work of Torralba et al. [18], who show that sharing features among multi-view models improves both running time and recognition performance, and Thomas et al. [17], who link similar appearance features of the same training object to enable better multi-view recognition. In both of these systems, the training images need to be taken from many views for many instances to achieve satisfactory recognition results. In contrast, we introduce a method for leveraging fewer training images from fewer views to obtain multi-view recognition. In our method every object instance contributes to the recognition performance in many nearby views.

2. Modeling 3D shape and appearance

The Potemkin model is used to represent an approximate 3D shape and appearance model. In this section we describe it in detail, including how to estimate the parameters from a small amount of training data, and use it to generate virtual training data.

2.1. The Potemkin model

A Potemkin model for an object class with m 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*, $S_1, \ldots, S_m$, specifying the 3D positions of part centroids, in a fixed, normalized reference frame; and
- mk^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, represented in homogenous coordinates, 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 multimodality, it will become necessary to extend the model to probabability 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.

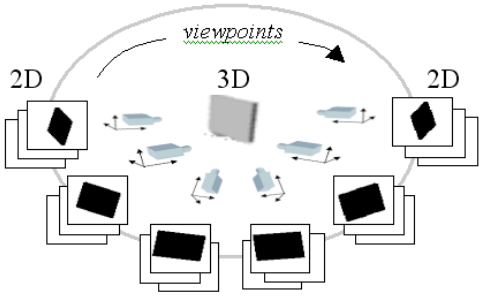

Figure 1. A synthetic nearly planar box object used for learning the generic view transformations.

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 $k-1$ additional images containing predicted views of the parts of the instance from the other view bins. Below we review how we estimate a Potemkin model from training data and then use it to predict new label images.

2.2. 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 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.

2.2.1 Learning generic view transformations

We start by learning a set of generic view transformations that are not part-specific. These will be used to initialize learning of object-part-specific transformations, based on very few real training images.

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

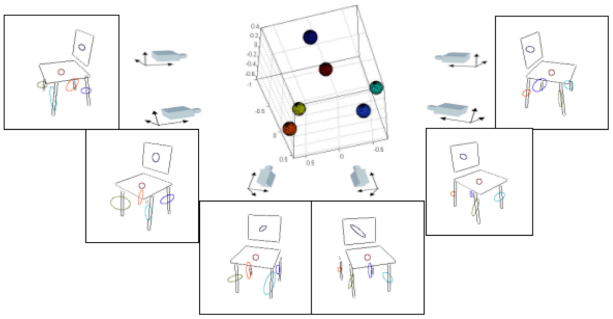

Figure 2. The ellipses indicate the distribution of 2D positions of the centroids of each part in the training set of chair images for each viewpoint. The 3D spheres show the estimate of the skeleton positions S_j for each part.

transforms are currently learned from synthetic binary images (one drawn randomly from each of the k view bins) of approximately 30 relatively "flat" blocks of varying dimensions, as depicted in figure 1. Note that since we are using synthetic data, we have enough data to get a good initial estimate for all the pairwise view transforms. A smaller set of real images could be used in addition to or instead of the synthetic images.

In this paper, the blocks used for learning the transforms are all upright, similar to those shown in the figure; this has been sufficient to get reasonable initial estimates of 2D shape transformation in our experiments. In future, we could learn the transforms for several simple shapes and several 3D orientations of the shapes and use the first few instances of a new class to select the most appropriate transform for each part.

To learn the generic transforms $T_{\alpha,\beta}$, we first establish correspondences between points on the (synthetic or real) training images, each of which is the outline of a part in the image. We use the shape context algorithm [14] to obtain a dense matching between the 2D boundaries across the images. Then, we use linear regression to solve for a 2D projective transform that best explains the observed matches across a set of image pairs representing a sample of object dimensions and viewpoints within the view bins.

General 2D projective transforms have 8 parameters and (assuming $v = 1$ below) they can be decomposed [11] as:

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

In our case, the translation $\mathbf{t}$ is known to be zero. Choosing $\mathbf{K} = \begin{bmatrix} K_a & K_b \\ 0 & 1/K_a \end{bmatrix}$, $\mathbf{v}^T = [v_1, v_2]$ and $s > 0$, 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 6 parameters over the training data as our generic view transform.

2.2.2 Refining the view transformations

The actual training data given to the recognition system are images of different instances of the class at different views. If the 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 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 label-image pair $\langle x_\alpha, x_\beta \rangle$ with views in bins α and β. For each part j in each image, we use the shape-context algorithm to match points on the boundary of the part and then construct the transform $\hat{T}^j_{\alpha,\beta}$, represented as a vector of 6 parameters, as above. Assuming n image pairs are available, we can find the mean of the resulting parameter vectors.

We then combine the generic view transform (the mean of the transforms derived from the synthetic images) and the mean of the part transforms to obtain a part-specific view transform $T^j_{\alpha,\beta}$.

$$T^j_{\alpha,\beta} = \frac{n}{\kappa + n}\hat{T}^j_{\alpha,\beta} + \frac{\kappa}{\kappa + n}T_{\alpha,\beta}$$

The relative weighting of the generic and specific transforms is determined by the parameter κ, which is currently chosen empirically; in principle it could be treated as a Bayesian hyperparameter and learnt based on training data from many object classes.

2.2.3 Learning the class skeleton

The class skeleton can be estimated from any collection of images of instances in any views, and does not require there to be multiple views of any instance. The label images that are used for training allow us to directly compute the centroids of the individual parts. We will 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 translating and normalizing the training images relative to a bounding box that contains them all. 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. Finally, we solve for the set of 3D positions that best project to the mean centroids of the corresponding labeled parts, using the covariance in the estimates to weight their contribution to the solution.

We do this using the PowerFactorization method [10] to solve the weighted least squares problem:

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

where S_j is the 3D position for part j in the skeleton. The skeleton location for each part is estimated independently,

Figure 3. The 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 Potemkin model. The model in bottom two rows is constructed from a total of 10 real images, from 5 object instances over 6 views.

because we have no prior on the structure of a new object class.

Figure 2 shows a schematic version of this process. In each view bin, the distribution on the centroid of each part is estimated, shown by the ellipses. Then, the centroids 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; ultimately it may be important to explicitly model self-occlusion.

2.3. Using the model

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 by k, and allows us to train recognizers for view bins that have never actually been seen, based only on virtual data.

Given an input label image in view bin α, we can break the prediction problem into two stages. For each part j and each viewpoint $\beta \neq \alpha$, we

1. transform the label image of each part j using $T^j_{\alpha,\beta}$
2. 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 image for view β and can be readily extended to generate complete images.

An example of the results can be seen in figure 3 (top two rows), which shows the virtual images constructed from a minimal 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 recognition; see Section 4. If more training data is available, the results can be improved substantially. Figure 3 (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. The virtual examples can be placed into the background from the original image to construct a realistic image suitable for using in any existing recognition system as shown in Figure 4.

3. View-specific 2D recognition

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 recognizers [7, 9, 2, 3]. We have tested the effectiveness of the Potemkin images in a detection task by using them as input to the method developed by Crandall et al. [2] (see the Experimental Results section). 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.

3.1. 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, 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^s_j and Σ^s_j.

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

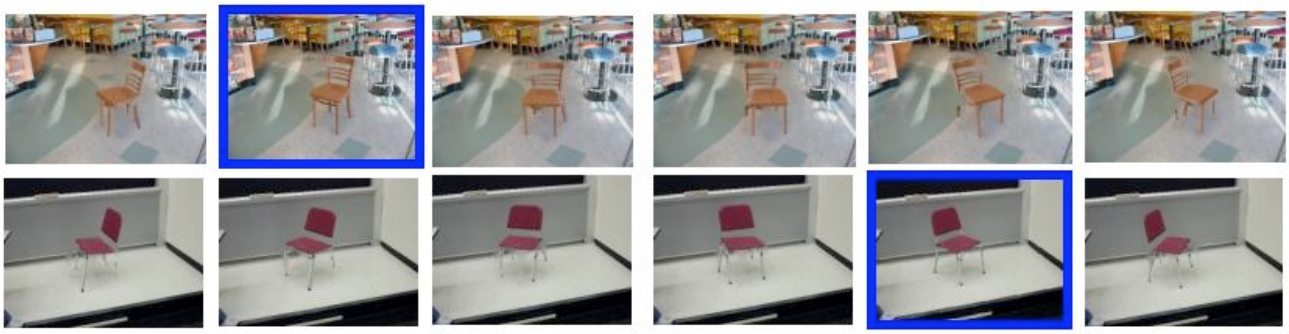

Figure 4. Full virtual images and the actual input images (bold border) of two object instances in the chair data set.

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.

3.2. 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.

3.3. Recognition

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 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 [7]. 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 this paper the scale of the object instance is assumed known and therefore recognition is done at a single scale; the search could readily be expanded across scale.

4. 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

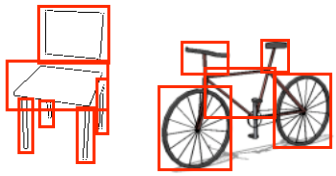

Figure 5. The 6 parts of a chair and the 5 parts of a bicycle.

et al [2]. We found that the virtual training images improved the performance of both systems.

We used two different object classes for our testing: four-legged chairs and bicycles (figure 5). The four-legged chair dataset was collected by 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 2. The bicycle data set is the TU-Graz-02 database, which is part of the PASCAL Visual Object Classes (VOC) Challenge [5]. 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.

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 training sets. The recognition 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 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 Potemkin model.

Figure 6 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 data, we can usually make predictions for viewpoints for which we have no real training data at all; we explore that case below.

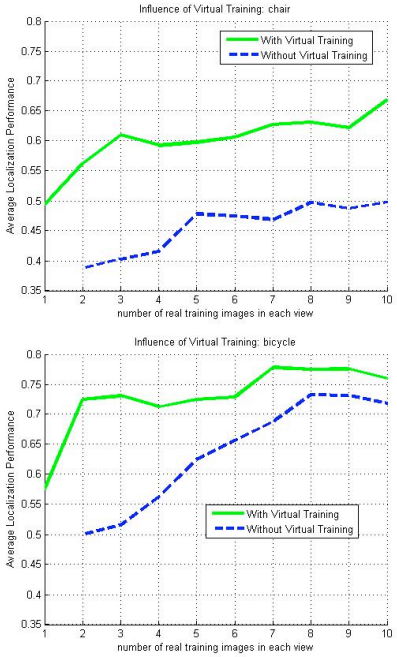

Figure 6. Localization performance of EES for chairs (top) and bicycles, using all the training data for all views.

The performance when using virtual data on chairs shows substantial improvement over the no virtual data case. Since the shape of chairs is so variable, this type of recognition system requires a substantial number of training images. When using virtual 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 7 shows some correct localization results on chairs and bicycles.

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 8 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 real data, with the top line using virtual data and the bottom line not. These results illustrate

Figure 7. Successful localizations of chairs and bicycles using EES.

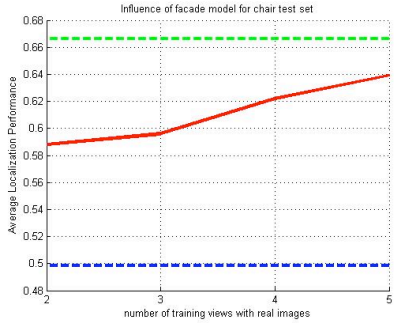

Figure 8. Localization performance of EES for chairs as a function of the number of views in the training data.

that the quality of the virtual data is comparable to that of the real data.

Detection results with Crandall et al's system. We have also carried out detection experiments using the system developed by Crandall et al [2]. We used their implementation and did training using a combination of real and virtual images for the chair and bicycle data sets. Figure 9 (top) 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.

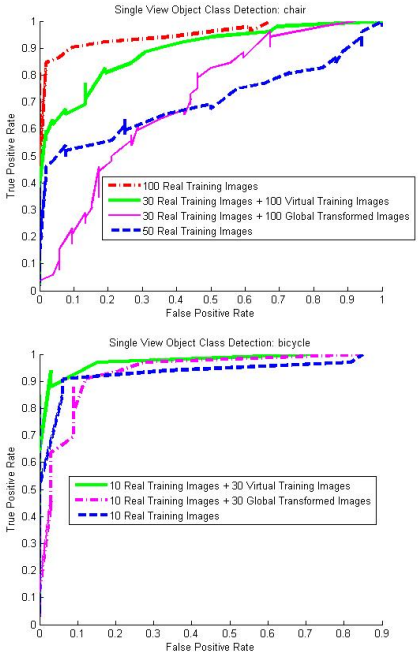

Figure 9. ROC curves for detecting chairs vs background images in a single view using [2] (see text).

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.

As a control, we also tested the effectiveness of using a single global transform for the whole object instead of the full Potemkin model with multiple parts. The images using the Potemkin model were much 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 recognition performance; see Figure 9 (bottom). 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 10). 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 Potemkin virtual images

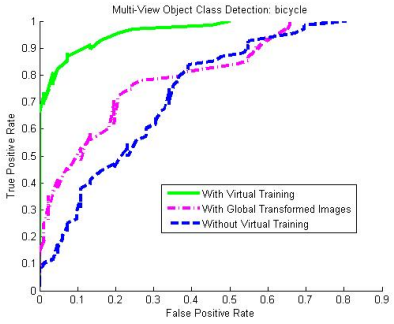

Figure 10. ROC curves for multi-view bicycle detection using [2] (see text).

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 Potemkin model refines the transforms for individual parts and does a better job of predicting the appearance of parts across distant viewpoints. It is possible that Crandall et al's unsupervised training method [3] could do better with the globally transformed images than the supervised system does.

5. Conclusion

We have proposed an approach to circumventing one of the key roadblocks to effective multi-view recognition based on single-view recognition approaches, namely the need for extensive training data in all the viewpoints of interest. We propose a simple class of object models (the Potemkin model) that can be efficiently learned from few images and that can be used to generate virtual training data for a wide range of viewpoints. We have demonstrated the effectiveness of this model in reducing training data requirements in object localization and detection tasks. Future work will extend the model to handle a greater variety of initial part shapes and a more sophisticated combination of prior expectations and training data.

Acknowledgments

This research was supported in part by DARPA IPTO Contract FA8750-05-2-0249, "Effective Bayesian Transfer Learning." We thank M. Aycinena, S. Davies, S. Finney and K. Hsiao for their help in gathering and labeling the images.

References

[1] E. Bart, E. Byvatov, and S. Ullman. View-invariant recognition using corresponding object fragments. In *ECCV*, 2004.

[2] D. Crandall, P. F. Felzenszwalb, and D. P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, 2005.

[3] D. Crandall and D. P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *ECCV*, 2006.

[4] M. Everingham and A. Zisserman. Identifying individuals in video by combining generative and discriminative head models. In *ICCV*, pages 1103–1110, 2005.

[5] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual object classes challenge 2006 (voc2006) results. In *1st PASCAL Challenges Workshop*, to appear.

[6] Z. G. Fan and B. L. Lu. Fast recognition of multi-view faces with feature selection. In *ICCV*, 2005.

[7] P. Feltzenswalb and D. Hutenlocher. Pictorial structures for object recognition. In *IJCV*, 2005.

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, pages 264–271, 2003.

[9] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005.

[10] R. Hartley and F. Schaffalitzky. Powerfactorization: 3d reconstruction with missing or uncertain data. In *AJAWCV*, 2003.

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

[12] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *BMVC*, 2004.

[13] S. Li and Z. Zhang. Floatboost learning and statistical face detection. *PAMI*, 26(9):1112–1123, 2004.

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

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

[16] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR*, 2000.

[17] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *CVPR*, 2006.

[18] A. Torralba, K. P. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. In *CVPR*, 2004.

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