Image and Vision Computing 30 (2012) 923–933

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

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

Viewpoint-aware object detection and continuous pose estimation ☆

Daniel Glasner [a,*], Meirav Galun [a], Sharon Alpert [a], Ronen Basri [a], Gregory Shakhnarovich [b]

[a] Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel
[b] Toyota Technological Institute at Chicago, United States

ARTICLE INFO

Article history:
Received 31 May 2012
Received in revised form 7 September 2012
Accepted 30 September 2012

Keywords:
Viewpoint-aware
Object detection
Pose estimation
3D model
Viewpoint estimation
Structure from motion

ABSTRACT

We describe an approach to category-level detection and viewpoint estimation for rigid 3D objects from single 2D images. In contrast to many existing methods, we directly integrate 3D reasoning with an appearance-based voting architecture. Our method relies on a nonparametric representation of a joint distribution of shape and appearance of the object class. Our voting method employs a novel parameterization of joint detection and viewpoint hypothesis space, allowing efficient accumulation of evidence. We combine this with a re-scoring and refinement mechanism, using an ensemble of view-specific support vector machines. We evaluate the performance of our approach in detection and pose estimation of cars on a number of benchmark datasets. Finally we introduce the "Weizmann Cars ViewPoint" (WCVP) dataset, a benchmark for evaluating continuous pose estimation.

1. Introduction

The problem of category-level object detection has been at the forefront of computer vision research in recent years. One of the main difficulties in the detection task stems from variability in the objects' appearance due to viewpoint variation (or equivalently pose variation). While most existing methods treat the detection task as that of 2D pattern recognition, there is an increasing interest in methods that explicitly account for view variation and that combine detection with pose estimation. This paper presents an approach that integrates detection and pose estimation using 3D class models of rigid objects and demonstrates this approach on the problem of car detection.

Building a viewpoint-aware detector presents a number of challenges. The first question is how to acquire a 3D representation of a class. The recent availability of CAD models, 3D cameras, and robust structure-from-motion (SfM) software has simplified this problem. Using SfM methods or 3D cameras makes it straightforward to relate the available 3D representations to the appearance of objects in training images. Secondly, finding the pose of an object at test time requires search in the 6D space of possible Euclidean transformations. This can be accomplished by searching exhaustively through a discrete binning of this 6D space. An alternative is to use a combinatorial search e.g., RANSAC [1] procedure. Both options, however, can be computationally expensive. Finally, how should detection and pose estimation be integrated? Pose estimation can deteriorate significantly when detection is inaccurate. Can detection be improved if pose estimation is integrated into the process?

We suggest an approach that combines a nonparametric voting procedure with discriminative re-scoring for detection and pose estimation of rigid objects.

We address the challenge of building a 3D class model by applying state-of-the-art SfM reconstruction software [2,3] to a set of training images that we have collected. A 3D point cloud is reconstructed for each scene and the car of interest is manually segmented. The class model is composed by registering and merging these point clouds in 3D. Further details regarding the construction of the model can be found in Section 5. A notable advantage of using SfM is that it provides us with correspondences between an accurate 3D shape model (see Fig. 5) and 2D appearance in real images. We model the within class variability by using multiple exemplars in a non-parametric model. By performing a simple registration of the point clouds in 3D we are able to circumvent the difficult problem of finding correspondences between parts across different class instances. The registered point clouds can be seen in Fig. 4(d).

At test time we wish to infer the location and pose of a class instance or equivalently the most likely transformation which relates the 3D model to the 2D projection in the test image. We use a nonparametric voting procedure, by searching the database for patches similar to ones seen in the input image. Each match generates a vote, in the spirit of [4] and other methods inspired by Hough transform. However, here the vote implies not only a bounding box, as is done, e.g., in [4], but a full 6 DOF weak perspective transformation of the object. The 6 DOF voting mechanism relates detection and pose estimation starting at the very early stages of the visual processing. In addition to providing

☆ This paper has been recommended for acceptance by Thomas Brox.

* Corresponding author. Tel.: +972 89344443.
E-mail addresses: daniel.glasner@weizmann.ac.il (D. Glasner), meirav.galun@weizmann.ac.il (M. Galun), sharon.alpert@weizmann.ac.il (S. Alpert), ronen.basri@weizmann.ac.il (R. Basri), gregory@ttic.edu (G. Shakhnarovich).

0262-8856/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.imavis.2012.09.006

a natural framework to estimate viewpoint, this serves to improve the localization performance of the detector, by constraining the detections to be consistent with a specific viewpoint estimate. Specifically, if the detector proposes a hypothesized detection of a car seen from a particular viewpoint, we should expect that all the features that support it are consistent with that viewpoint. Intuitively, this is a built-in verification mechanism.

The voting mechanism outlined above, and described in detail in Section 3, serves as the first stage of detection. It can be thought of as an attention mechanism, generating a relatively small set of hypothesized object detections, each with a specific hypothesized 3D pose. In line with the previous work [5] we follow the voting phase by a discriminative stage. The second stage, refines the hypothesized detections, and ranks them by applying a set of support vector machines, each trained to score detections in a sector of viewpoints. At test time we use the pose estimation generated by the voting procedure to index the correct SVM. Thus, this final stage of verification is also viewpoint-aware. Overall, this process allows us to improve and refine our candidate detections.

We focus our experiments on cars and apply our algorithm to four datasets: Pascal 2007, Stanford 3Dpose dataset [6], EPFL car data set [7] and a new benchmark introduced here the "Weizmann Cars ViewPoint" dataset.

2. Background

A common approach for coping with viewpoint variability is to use multiple, independent 2D models. In this multiview approach one describes the appearance of an object class at a discrete set of representative viewpoints. These algorithms (e.g., [8,9,7,10]) implicitly assume that the 2D appearance of an object near the representative views varies smoothly and that local descriptors are robust enough to handle these appearance variations.

Extensions to the popular Deformable Part Model (DPM) [9] which allow for viewpoint estimation have been suggested in [11] and in the recent work of Pepik et al. [12]. Both works consider a model in which the mixture components are identified with a discrete set of viewpoints. [11] further suggest continuous viewpoint estimation by learning a linear model around each of the discrete representative views. [12] extend the DPM by modeling continuous part locations in 3D. Their method produces state-of-the-art detection and pose estimation results on the Stanford 3Dpose dataset in which the task is to classify test images into one of a discrete set of viewpoints. They do not report results on a continuous viewpoint estimation task.

Another line of studies [6,13,14] approaches the problem of view variation by building 2D multi-part representations and establishing correspondences among the parts across different class views. The resulting model accounts for a dense, multiview representation and is capable of recognizing unseen views.

Many of the algorithms which explicitly model 3D shape utilize 3D CAD models [15–18,12,19]. These works take different approaches to generate correspondences between the 3D CAD models and the 2D appearances which can be matched to the test images.

The approach in [15,16,18] is to generate non-photorealistic renderings of the CAD models from which they extract features (e.g., edges) or learn 2D detectors which can then be used to find matches in real images. The rendering approach maintains an exact correspondence between 3D shape and 2D appearance, but the resulting 2D appearance models are not as powerful as those learned from real 2D images.

The authors of [17,12,19] also take advantage of real images as part of their training data. The approach described in [17] is to learn 3D shape models and 2D appearance models separately. The 3D and 2D models are then linked by a rough correspondence which is established between bounding boxes extracted from real images and from ones which are rendered using the 3D CAD. The bounding boxes are partitioned into blocks centered on a regular grid and correspondence is determined by the grid coordinates.

In their 3D parts model the authors of [12] suggest a Deformable Part Model with 3D volumetric parts. These are parameterized as axis aligned fixed size 3D bounding cubes. The 2D appearance templates are learned from non-photorealistic renderings of CAD models. Their framework also allows for the inclusion of real training images as part of training.

The work described in [19] suggests modeling 3D objects using "aspect parts". These are defined as a portion of the object whose entire surface is either visible or occluded from any viewpoint. Correspondence between 3D configurations of aspect parts and their appearance in the image is modeled as a conditional random field. Rather than using multiple appearance templates the authors suggest applying homographies to rectify parts to a canonical viewpoint. To generate correspondences between the 3D aspect-parts and their appearances in 2D images the authors rely on extensive hand labeled part annotations of the training data.

In other work, Arie-Nachimson and Basri [20] construct a 3D model by employing an SfM process on the entire training set of class images. An advantage of this approach is that the SfM provides correspondences between 3D shape and 2D appearance. However, their method requires finding correspondences between parts as they appear in different class instances.

Villamizar et al. [21] present a two-step approach. In the first step a Hough-based pose estimator identifies candidate detections along with their pose estimates. In the second step these are validated using pose-specific classifiers. In both steps their method uses a common set of Random Fern features which are also shared between the classifiers.

Sun et al. [22] suggest the use of depth information, and train models using depth maps acquired with a range camera. Detections are generated by depth-encoded voting. Pose estimation is then achieved by registering the inferred point cloud and a 3D CAD model.

Payet and Todorovic [23] learned a collection of 2D shape templates describing the appearance of the object contours at a discrete set of viewpoints. At test time, the learned templates are matched to the image contours while allowing for an arbitrary affine projection. The matching problem is approximated using a convex relaxation. The parameters of the chosen affine projection serve as a continuous estimate of the pose in 3D while the best matching template identifies a discrete estimate.

Finally, a hybrid 2D–3D model is suggested in [24]. The model consists of stick-like 2D and 3D primitives. The learning selects 3D primitives to describe viewpoint varying parts and 2D primitives where the appearance is viewpoint invariant.

In contrast to related work, we propose a simple and flexible method to construct rich, nonparametric 3D class models. State-of-the-art SfM software allows us to model 3D shape without requiring a library of CAD models. It also provides us with accurate correspondence between 3D shape and real-world 2D appearances. Our method does not need costly manual part annotation, in-fact it manages to bypass the difficult problem of finding correspondences between parts in 2D, by solving an easy global registration problem in 3D. Finally, unlike other works which are restricted to discrete pose classification, our method optimizes over a continuous 6D transformation space and is able to generate accurate continuous pose-estimates.

3. Nonparametric detection and pose estimation

We approach the problem of object detection and pose estimation in two stages. First we apply nonparametric voting to produce a bank of candidate detections along with their estimated poses. Then we apply a discriminative re-scoring procedure designed to improve the detection and pose estimation results. In this section we describe the

construction of a 3D model and the voting in the 6D space of possible pose variables. The re-scoring step is described in Section 4.

Hough-like voting procedures have been proven effective in object detection both in 2D [4] and in 3D methods [20,22]. Their success is due to the frequent resemblance of corresponding visual elements across instances of a class. Thus, for example, an image region similar to a stored patch depicting the appearance of a bottom left windshield corner in a previously seen car may indicate the presence of a windshield corner of a (possibly different) car in the test image. Moreover, since appearance can change significantly with viewpoint, such a match may also indicate the viewpoint from which the car is observed. Naturally, such evidence would not be very reliable, as we confine ourselves to small regions. Voting allows us to overcome this by accumulating information from a large number of regions and identifying constellations of patches that cooperatively look like parts of previously seen class instances. These patches are seen from similar viewpoints, and arranged in positions consistent with each other under that viewpoint.

3.1. Model representation

The object category of interest is represented by a set of 3D models of object instances. Each single model consists of a cloud of 3D points in a class-centered coordinate frame, i.e., the 3D models of the different class instances are aligned to produce consistent poses. Along with these 3D models we store a collection of regions obtained from the set of training images at different scales. Each image region (patch) is associated with a particular 3D position and a particular viewpoint and scale. These patches are the basic elements of our nonparametric model. Each patch is described by a 3D feature represented as a tuple $f^3=\langle \mathbf{e},\mathbf{l},t,\mathbf{x}\rangle$, where $\mathbf{e}$ is a descriptor of the patch appearance, $\mathbf{l}$ is the 3D location of the associated keypoint, and t and $\mathbf{x}$ are related to the training image from which the patch associated with f^3 was extracted. t indexes the image and scale, and $\mathbf{x}$ is the location of the patch in that image. Note that multiple features will share the same 3D location $\mathbf{l}$, but will have different t values (indicating that they come from different images at different scales) and possibly different appearance $\mathbf{e}$ (see Fig. 1). We will refer to the entire collection of 3D features $f_1^3,\ldots,f_D^3$ pooled from all the models as the 3D database. Preservation of multiple instances in the model allows us to mix and match parts from different instances, without potential loss of discriminative information associated with quantization, which has been pointed out in [25].

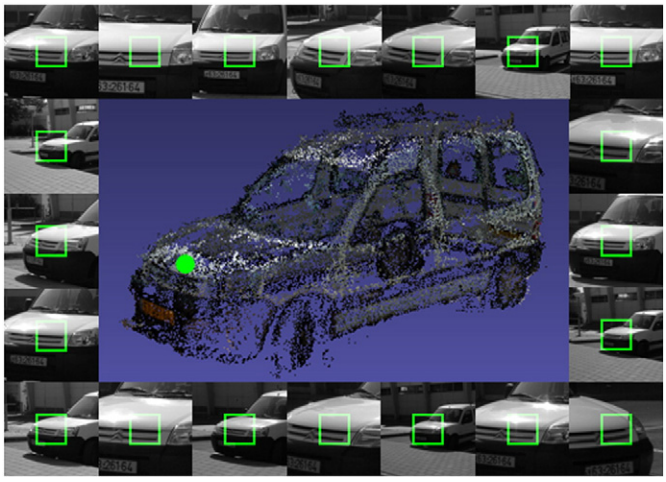

Fig. 1. 2D appearances: An example of the different 2D appearances $\{\mathbf{e}\}$ of a 3D point $\mathbf{l}$ (denoted as a green dot) as seen from different view-points and scales (corresponding to different training image indices t).

3.2. Pose-sensitive voting

The voting step lies at the core of our method: it allows us to detect and determine the pose of a class instance by identifying constellations of local features which suggest a consistent location and pose. In this work we use the full 6 DOF continuous voting space. This choice allows us to produce continuous pose estimates and to identify novel poses which were not part of our training data. On the other hand, working in a high dimensional space, introduces some new challenges which we must address. Two immediate issues are representation and efficiency.

The approach suggested by Arie–Nachimson and Basri [20] is to evaluate a finite set of samples from the 6D pose space. These samples are determined by enumeration of triplets of matches between features from the test image and the database. We take a less restrictive approach and allow individual features to cast votes independently. We extract a dense sample of patches from the test image and match these patches to the database of model patches. Each match between a patch from the test image and a patch from the database suggests a hypothesis for the 6 DOF transformation relating the model to an instance in the test image. The detection problem now reduces to identifying consistent subsets among this set of votes in a high dimensional space. The large number of votes and the high dimension of the voting space make this a challenging problem.

To identify the consistent subsets we suggest using mean-shift clustering; however the application is not straightforward. First we must deal with the issue of representation: the 6 DOF space of Euclidean transformations is not uniform. The units of the rotation parameters have nothing to do with the units of the scale or translation parameters. To overcome this issue we suggest a novel representation. A transformation is parameterized by its effect on a fixed set of 4 designated points. This produces an embedding which makes the votes amenable to clustering using a Euclidean norm. We provide further details of the suggested representation below. Second, as is typical of nonparametric models, most of the computations occur at test time. This emphasizes the need for an efficient vote clustering method, towards this end we suggest a number of heuristics to speed up mean-shift clustering, these are described in Section 5.2.

Our voting procedure proceeds as follows. The image is covered by an overlapping grid of patches; each patch corresponds to a 2D feature represented by $f^2=\langle \mathbf{e},\mathbf{x}\rangle$ where $\mathbf{e}$ is the descriptor extracted from the patch, and $\mathbf{x}$ is the 2D coordinates of the patch.

An input feature $f_i^2=\langle \mathbf{e}_i,\mathbf{x}_i\rangle$ "solicits" votes from the instance models, as follows. We find the K nearest neighbors of $\mathbf{e}_i$ among the descriptors of model patches, and consider the corresponding 3D features to be the matches for f_i^2. Without loss of generality, let these be $f_1^3,\ldots,f_K^3$. A match $f_i^2=\langle \mathbf{e}_i,\mathbf{x}_i\rangle \rightarrow f_k^3=\langle \mathbf{e}_k,\mathbf{l}_k,t_k,\mathbf{x}_k\rangle$ implies a hypothesized 6 DOF transformation $T_{i,k}$, as explained below.

3.2.1. Projection model

In this work we consider the weak perspective projection model: an object undergoes isotropic scaling, 3D rotation and 2D translation parallel to the image plane, followed by orthographic projection onto the image plane. The scaling is equivalent to translation along the axis orthogonal to image plane prior to projection. Projective transformations in this family have six degrees of freedom (DOF): two for in-plane translation, one for scaling, and three for rotation.

We assume that an object point's appearance is invariant under translation but varies under rotation and scale. We can thus hypothesize that since the patch of f_i^2 is similar to that of f_k^3, the corresponding object point is viewed from the same viewpoint and at the same scale (equivalently translation along the optical axis z). Thus, four out of six DOF of $T_{i,k}$ can be inferred directly from f_k^3 (by looking up

the scale and viewpoint of the training image indexed by t_k). The remaining two parameters of translation parallel to the image plane are recovered from the equation

$$T_{i,k}(\mathbf{l}_k) = \mathbf{x}_i. \tag{1}$$

3.2.2. Vote representation

We now need to turn the estimated $T_{i,k}$ into a vote in a space representing the possible transformations. Keeping in mind the eventual need to identify peaks and evaluate similarity between votes we avoid the natural representation as a 6-vector with 3 entries parameterizing a rotation, 2 for translation and one for scale. The problem with such a representation is that the space $SO(3) \times \mathbb{R}^3$ is not isotropic. For example the units of the rotation parameters are not comparable to the translation or the scale. Using such a representation we cannot cluster points using a Euclidean norm. Instead we suggest a novel parameterization which amounts to an embedding in an 8 dimensional Euclidean space which is amenable to mean-shift clustering with a Euclidean norm. The parameterization is described below and also summarized in Fig. 2.

Let $\{\mathbf{p}_j^{\text{des}}\}_{j=1}^4$ denote a set of 4 designated points in $\mathbb{R}^3$. We use 4 vertices of a cube of side length one half, centered at the origin. We represent $T_{i,k}$ as a point in $\mathbb{R}^8$:

$$V(i,k) = \left[T_{i,k}\left(\mathbf{p}_1^{\text{des}}\right)^T, \ldots, T_{i,k}\left(\mathbf{p}_m^{\text{des}}\right)^T\right]^T, \tag{2}$$

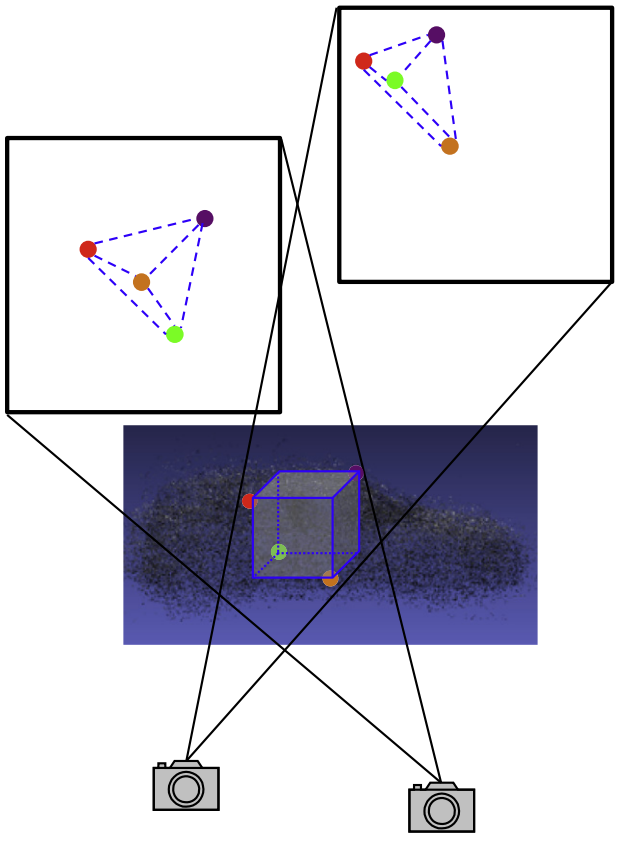

Fig. 2. Parameterization. We suggest to parameterize the 6 DOF transformation which relates the 3D model to an instance in a 2D image by the action of the transformation on a set of 4 fixed designated points. The 4 designated points are shown, superimposed over the 3D model. Different cameras project the points to different locations in the image plane. The concatenation of the coordinates of these projections defines our 8 dimensional representation.

where

$$T_{i,k}\left(\mathbf{p}_j^{\text{des}}\right) = \mathbf{x}_i + \left(\mathbf{x}_j^{t_k} - \mathbf{x}_k\right). \tag{3}$$

Here $\mathbf{x}_j^{t_k}$ is the projection of the j'th designated point onto the training image indexed by t_k. This is illustrated in Fig. 3.

Since our embedding represents a family of transformations with 6 DOF in an 8 dimensional space not all points in the new representation correspond to a valid transformation. That is, not every point in $\mathbb{R}^8$ is obtainable as a projection of the designated points in the weak perspective model. To address this issue we map any point $V \in \mathbb{R}^8$ to a "corrected" point $\tilde{V} \in \mathbb{R}^8$ that corresponds to a valid transformation. The new point $\tilde{V}$ is obtained as the projection of the 4 designated points using a weak perspective transformation which minimizes the reprojection error. Finding this transformation is equivalent to the problem of camera calibration from correspondences between image points and their known 3D counterparts. In our case 4 image points are given by V and their counterparts are the 4 designated points in $\mathbb{R}^3$. We compute the transformation using the method of [26] as implemented in the "camera calibration toolbox", and use it to project the designated points onto the test image. The concatenation of the projections gives us the "corrected" point $\tilde{V}$.

The number of designated points we use determines the dimension of the voting space. Therefore we would like to keep it as close to the minimal dimension which is 6. When using only 3 points the transformation is determined up to a reflection ambiguity. We avoid this ambiguity by adding a fourth point.

We position the points so as to maximize numerical stability when solving the camera calibration problem. The points are evenly spread out at a fixed distance from the origin to avoid bias. The distance is chosen so that the points are on the same scale as the 3D model.

Note that the weak perspective assumption allows us to store the locations of the projections of designated points in each training image and simply apply the translation part of $T_{i,k}$ at test time to generate a vote. We denote by V the set of all the votes cast by features of the input image; if the number of 2D features extracted from the input image is N_0 then $|V| = K \cdot N_0$.

3.3. Vote consolidation

Once all votes have been cast, we seek peaks as modes of the estimated density of votes, subject to pose consistency. These can be found by the mean-shift algorithm which climbs the estimated density surface from each vote. We found this to be somewhat slow in practice, and therefore resorted to a multi-stage approximation, described in some detail in Section 5.

3.3.1. Vote clustering

Finally, we form vote clusters by a greedy procedure. The top ranked mode V_1' is associated with the first cluster. By solving a camera calibration problem as described in Section 3.2 we compute a point $\tilde{V}' \in \mathbb{R}^8$ which corresponds to the closest valid transformation (i.e. it is obtainable as a projection of the designated points in the weak perspective model). We associate with the first cluster all votes that are sufficiently similar to V_1' in the location of the detected object and the estimated viewpoint (see Section 5 for more details about this similarity). All points which have been associated with the first cluster are culled from the vote set V. The second cluster is generated in a similar manner. We identify the highest ranking mode which has not yet been culled: V_2'. We compute the corrected mode $\tilde{V}_2'$ and assign all votes which are still in V and sufficiently similar to $\tilde{V}_2'$ to the cluster. The process continues until we have the reached some pre-determined number of clusters or until V is empty.

Fig. 3. Voting process. Four patches from the test image (top left) are matched to database patches. The matching patches are shown with the corresponding color on the right column. Each match generates a vote in 6D pose space. We parameterize a point in pose space as a projection of designated points in 3D onto the image plane. These projections are shown here as dotted triangles. The red, green and blue votes correspond to a true detection, the cast pose votes are well clustered in pose space (bottom left) while the yellow match casts a false vote.

4. Verification, refinement, and rescoring

So far we have described the voting procedure which generates a set of candidate detections. In the second stage of our algorithm each one of these candidates is further processed. The location of the bounding box is refined. Each detection is assigned a score that reflects our confidence in it. And finally, viewpoint estimates are possibly corrected.

The overall objective of the second stage is to improve the precision-recall performance. We cast this as a scoring problem: given a region $\mathbf{b}$ in the image, we assign a score value $S(\mathbf{b})$ which is higher the more likely we deem $\mathbf{b}$ to be the bounding box of an object instance. This score can be used to classify the region, by thresholding S, and to rank detections, ordering by decreasing value of S.

4.1. SVM scoring

We use support vector machine (SVM) to estimate the score $S(\mathbf{b})$. A region $\mathbf{b}$ is represented as a feature vector $h(\mathbf{b})$ which is a concatenation of histograms of oriented gradients computed over a pyramid of spatial bins. We train the SVM on a set of feature vectors $\{\mathbf{b}_n\}$ computed from labeled example regions. Details are given in Section 5. Once trained, the SVM score is computed as

$$S(\mathbf{b}) = \sum_{n \in \mathrm{SV}} \alpha_n K(h(\mathbf{b}), h(\mathbf{b}_n)) \tag{4}$$

where α_n are positive coefficients, SV is a subset of indices of training examples and K is an RBF kernel function

$$K\left(x, x'\right) = exp\left\{-\frac{1}{\sigma^2}\|\left(x - x'\right)\|_2^2\right\}. \tag{5}$$

4.2. Viewpoint specific training

We can either train a single SVM, or a set of SVMs, designating a separate machine per sector in the viewpoint sphere. Our motivation for choosing the latter is related to the observation, shared by [27], that pose changes may be better fit by a mixture of appearance-based models. In this case, we provide a different set of positive examples to each SVM — namely those in which the correct viewpoint falls within the associated viewpoint region. The set of negative examples is shared across SVMs.

At test time we use the viewpoint estimated by the voting procedure to determine which SVM to apply. Given a candidate detection with an estimated viewpoint, we compute the score of the SVM "responsible" for that viewpoint.

4.3. Refinement

Inspired by [9] we also use the SVM scoring to refine the detection bounding box via local search. Given the initial bounding box $\mathbf{b}$ generated by the voting, we consider a set of perturbed versions of $\mathbf{b}$,

obtained by a fixed set of shifts and scale changes relative to **b**. Each of these is scored, and the version with the highest score is used.

4.4. Symmetry

For symmetric objects we also evaluate the SVM responsible for the symmetric viewpoint. For example in the case of cars we evaluate the SVM corresponding to the 180° reversal of viewpoint. This is due to the empirical observation that the errors "flipping" the object along the symmetry axis seem to be far more frequent than other errors in viewpoint estimation. The higher SVM score is used as the detection score and the pose is flipped if this score was produced by the SVM responsible to the flipped direction.

5. Experiments

We evaluate our approach on the problem of car detection. Below we describe the training data and the model obtained, and report the results on a number of benchmark data sets.

5.1. Training data and the model

5.1.1. Data collection and initial model building

We collected and processed 22 sets of images of different car models. A set consists of approximately 70 images on average, of one car taken from different viewpoints which cover a full rotation around the car. The pictures were taken in an unconstrained outdoor setting using a hand-held camera. There are significant illumination changes, many images include cars in the background, and in some images the car is cropped or occluded. See Fig. 4(a).

5.1.2. Model construction and alignment

We use Bundler [2] and PMVS2 software [3] to turn a collection of images of an instance from the class into a model. PMVS2 generates a dense point cloud representing the complete scene. We manually segment this point cloud to extract the car. The annotation was generated using Meshlab [28] and took about 2 minutes per model. Note that the accuracy of this segmentation is not very important and in fact inclusion of some background points in the model can actually improve detection by modeling the context. For example cars tend to appear on roads so a road patch could contribute to identifying a car. However, in this work we do not explore the use of context. This procedure yields a set of models with coordinate frames that are somewhat arbitrary. We transform the coordinates so that the object centroid is at the origin, and the coordinate frame is aligned with the three principal components of the 3D point cloud (enforcing a left-handed system to avoid ambiguities) for each instance. We then manually identify an image of each instance that is closest to an (arbitrarily defined) canonical viewpoint, and refine the alignment. Finally, each point cloud is scaled so that the extent along a selected dimension is 1 (for cars we use the width). Note that this process modifies the values of $\mathbf{l}$, but not $\mathbf{e}$ or $\mathbf{x}$, of the 3D features; it also modifies the viewpoint assigned to a specific image and scale indexed by t. See Fig. 4.

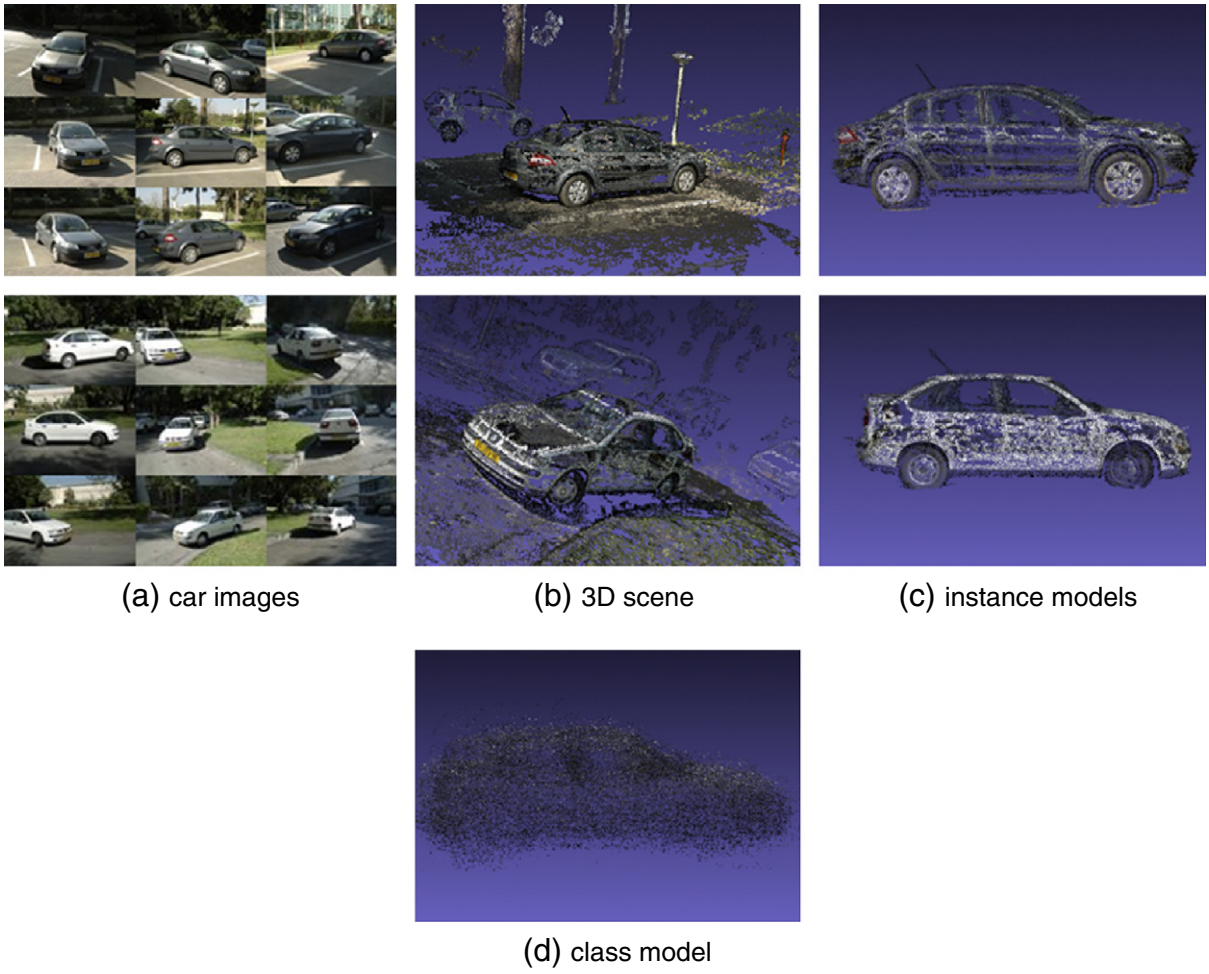

Fig. 4. Model construction. We construct the model from multiple sets of car images, some example frames from two different sequences can be seen in (a). Using Bundler we reconstruct a 3D scene (b). The car of interest is manually segmented and aligned (c). Finally a view of the class model is shown in (d).

5.1.3. Reconstruction accuracy

To verify the accuracy of our reconstructions we compared one of the reconstructed models to a publicly available CAD model of the same car.[1] We used the Iterative Closest Point algorithm [29] as implemented in MeshLab (with manual initialization) to align the two models and then measured the errors. The median error was 1.6 *cm* which is less than a percent of the car's width. Fig. 5 shows the two point clouds.

5.1.4. Pruning

Combined with high image resolution and the relatively dense sampling of the viewpoints in our data, the initial output from PMVS2 contains an extremely large number of 3D keypoints sampled very densely in some regions, and, consequently, of patches. We concluded that this density increases computational burden on a non-parametric detector without significant benefit. Thus we chose to prune the database. For each model, we divided the 3D bounding box of the cloud of 3D keypoints constructed by PMVS2 into equal sized cells. In each cell, we used the estimation of the normal direction as produced by PMVS2 to select a small number of representative keypoints. We binned the points according to the octant in which their normal resides and selected one representative from each bin as the one closest to the cell center. The pruned database consists, for each model, of the 3D features corresponding to these representatives.

5.1.5. Efficient similarity search

Even after the pruning described above, the 3D database remained prohibitively large for a brute force similarity search. Instead, we used the ANN library by Mount and Arya [30], and built a data structure allowing sublinear time approximate nearest neighbor search. The metric used on descriptors was ℓ_2.

5.2. Implementation details

5.2.1. Patch descriptors

We use a descriptor [31] which is similar to the HoG descriptors used extensively in the literature. Given a reference point $\mathbf{x}$, we take the square region with side $B*C$, with $\mathbf{x}$ at its left corner. This region is partitioned into a grid of $B\times B$ square blocks of size $C\times C$ pixels. Within each block, we compute intensity gradient at each pixel, bin the gradient orientations into P orientation bins, and compute the histogram of total gradient magnitudes within each bin. Finally, all B^2 histograms are divided by the total gradient energy averaged over the B^2 blocks, truncated at 1, and concatenated. Parameters B, P and C are set to 3, 5 and 8 respectively, producing 45-dimensional descriptors.

5.2.2. Finding modes of vote density

Along with the feature matching, the mean-shift procedure is the bottleneck in terms of computation at test time. To minimize this computation we suggest a number of heuristics to speed up mean-shift clustering. First, for each recorded vote V_i we compute a kernel density estimate (KDE) $\hat{p}(V_i)$ using RBF kernels in the $\mathbb{R}^8$ vote representation space. We select the n votes with the highest value of $\hat{p}$, these points are likely to be close to the modes of the probability $\hat{p}$. To avoid a bias which might be caused by selecting only these points we add an additional n' randomly selected votes. For example in a distribution with one very strong mode and another weak mode, it is possible that all of the n highest KDE value points will lead to the strong mode. The addition of n' random points allows for finding the weaker mode. Next, by applying mean-shift (computed over the density implied by all of the votes) we find the mode associated with each of the selected $n+n'$ votes. These $n+n'$ modes are then ordered according to their densities (again estimated by KDE, using all the votes).

5.2.3. Vote clustering

We used two criteria, the conjunction of which implies sufficient similarity between a vote and a cluster prototype. First, let the bounding box implied by $\tilde{V}'$ be $\mathbf{b}'$, and the viewpoint be represented by a unit norm vector $\mathbf{r}'$. The bounding box implied by a transformation is the bounding box of the projection of the model onto the test image.

A vote V_i with bounding box $\mathbf{b}_i$ and viewpoint vector $\mathbf{r}_i$ is similar to $\tilde{V}'$ if

$$o\left(\mathbf{b}_i,\mathbf{b}'\right)\geq 0.5 \text{ and} \left|\angle\left(\mathbf{r}_i,\mathbf{r}'\right)\right|\leq \pi/8, \tag{6}$$

where the bounding box overlap is defined by

$$o\left(\mathbf{b}_i,\mathbf{b}'\right)=\frac{\mathbf{b}_i\cap\mathbf{b}'}{\left|\mathbf{b}_i\cup\mathbf{b}'\right|}, \tag{7}$$

and $\angle\left(\mathbf{r}_i,\mathbf{r}'\right)$ is the angle between two 3D vectors.

5.2.4. SVM training and application

A region $\mathbf{b}$ is represented by a histogram of oriented gradients, computed over a pyramid of spatial partitions similar to [32]. At the first level, we simply compute the histogram of gradient orientations over the entire region, binned into P orientations. At the second level, we partition the region into 2×2 subregions, and compute the four histograms, one per subregion, and similarly for the third level producing 16 histograms. The histograms for all levels are concatenated to form a single descriptor for the region. A region is considered positive if its overlap with the bounding box of a known object detection, as defined in Eq. (7), is above 0.5, and negative if the overlap is below 0.2.

To refine detections, we consider vertical shifts by $\{0,\pm 0.2\cdot H\}$ pixels, and horizontal shifts by $\{0,\pm 0.2\cdot W\}$ where H and W are the height and width of $\mathbf{b}$. For each combination of shifts, we scale the bounding box around its center by {80%, 90%, 100%, 110, 120}. This results in 45 bounding boxes (one of which is the original $\mathbf{b}$), among which we choose the one with the highest value of SVM score S.

5.3. Results

We present detection (localization) and pose estimation results on three publicly available datasets:

The car category of the Pascal VOC 2007 challenge [33], the car category of the Stanford 3Dpose dataset of [6] and the EPFL multi-view cars dataset [7]. We also introduce the "Weizmann Cars View-Point" dataset for continuous viewpoint estimation and report our pose estimation results as baseline results for the benchmark.

5.3.1. Pascal VOC detection results

We evaluate the detection performance of our detector on the car category of the Pascal VOC 2007 data-set. The reported average precision (AP) scores were computed using the Pascal VOC 2007 evaluation protocol. The average precision is the area under the precision-recall curve.

As a baseline for detection evaluation we use our voting mechanism in a redundant "2D mode". In the "2D mode" each match generates a vote for the location of a 2 dimensional bounding box. 3D voting slightly outperforms the 2D voting with an AP of 16.29% compared to 15.86%.

We train an SVM classifier as described in Section 4, for positive examples we use windows from the Stanford 3Dpose dataset, the

[1] We used the model "Volkswagen Golf 2009N280111" downloaded from www.archive3d.net where it was posted by the user Billy Main.

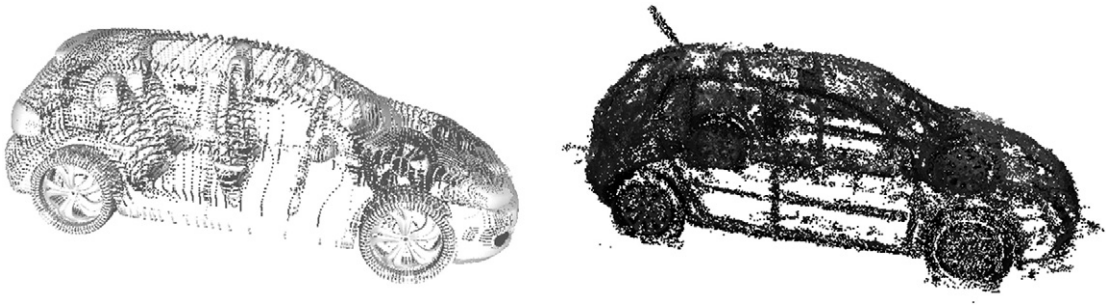

Fig. 5. Comparing CAD and SfM. We compare a CAD model (left) to the point cloud reconstructed using [2] and [3]. Using ICP we align the two models and measure the errors. The median error is 1.6 cm, less than one percent of the car's width.

EPFL dataset, our car dataset and the training images in the car category of Pascal VOC2007. Negative examples were taken from the training subset of Pascal VOC2007. The view-independent SVM classifier increases the AP for both 2D and 3D voting. The 3D retains a slight advantage with 27.97% compared to the 2D score of 24.34%.

In the final experiment we apply viewpoint specific SVM classifiers to the 3D votes. We train the classifiers as described in Section 4, using the same training data used in the view independent training but omitting the Pascal positive training examples, which are not labeled with (sufficiently fine) viewpoint information. The pose estimated by the 3D-voting is used to index the different classifiers. The combination of 3D voting and 8-viewpoint specific SVM classifiers produces the best result with an AP of 32.03%. Note that this score is achieved without any positive training examples from Pascal. Our AP score of 32.03% is an improvement compared to the AP score of 21.8% reported in [22]. That result was achieved using different training data. Namely, the authors collected images of 5 cars from 8 viewpoints and used these to transfer approximate depth information to Pascal training images which were then used to train their detector.

Our result is also better than 24.9% reported by [12] for their DPM-3D-Const model when trained using synthetic images only. However their DPM-3D-Const model achieves a much better result of 63.1% when real Pascal images are also used during training.

Our AP results are summarized in Table 1 and the recall precision curves are shown in Fig. 6(a). To reduce effects of additional training data we excluded all positive examples from the Stanford 3Dpose dataset, and the EPFL dataset and reran this last experiment using only positive examples from our own dataset without using any positive Pascal training images. The AP decreased from 32.03% to 29.43%.

5.3.2. Pose estimation on Pascal

The Pascal data includes only coarse pose labels (frontal/rear/right/left). Arie-Nachimson and Basri [20] augmented the pose labels on a subset of the 2007 test category. They labeled approximately 200 car instances with one of 40 labels which corresponds to an approximately uniform sample of azimuth angles. In their paper they report differences between the labels of their estimated pose and the ground truth labels for 188 objects. We detected 180 of these objects, and compared our pose estimation to theirs, see Fig. 6(b).

Table 1
Pascal VOC 2007 cars. Average precision achieved by our detectors compared to a 2D baseline.

	2D voting	2D voting + SVM	3D voting	3D voting + SVM	3D voting + 8view-SVM
AP	15.86%	24.34%	16.29%	27.97%	**32.03**%

Bold face denotes best result.

5.3.3. Stanford 3Dpose dataset

The Stanford 3Dpose dataset was introduced by [6] to evaluate detection and pose estimation. The car category includes 10 sets of car images, each set includes 48 images taken at 3 scales, 8 viewpoints and 2 different elevations. We follow [16], and use sets 1–5 for training and sets 6–10 for testing. We train an SVM using sets 1–5 along with positive examples from our own dataset, negative examples are taken from Pascal. AP scores were computed using the Pascal VOC2010 evaluation protocol and are summarized in Table 2. The combination of 3D voting and an 8-view SVM produces an AP result of 99.16%.

Our average classification accuracy results are given in Table 2. A confusion matrix and label differences are presented in Fig. 7.

When the proceedings version of this paper [34] was published our results of AP = 99.16% and AA = 85.28 were an improvement over the best previously published ones of [16] which were AP = 89.8% and AA = 81%. Since then further improvements have been achieved by [12] who report the best results to date, AP = 99.9% and AA = 97.9%.

5.3.4. EPFL car data set

The EPFL multiview car dataset was introduced in [7]. The dataset was acquired at a car show, 20 different models were imaged every 3 to 4° while the cars were rotating to produce a total of approximately 2000 images. We train 8-view SVM classifiers using the positive examples from the first 10 models along with images from our dataset and from the Stanford 3Dpose dataset. Negative examples were taken from Pascal training images.

We ran our 3D-voting followed by SVM on the last 10 models achieving an average precision of 89.54% (measured using Pascal VOC 2010 evaluation protocol). We then evaluate our pose estimates on the true detections. We achieve a median angular error of 24.83°. We show a histogram of the angular errors in our pose estimates in Fig. 8(a). A similar histogram is shown in [7], however our results are not directly comparable since they report pose estimates on all of the windows which were considered by their classifier and overlapped the ground truth by more than one half.

5.4. Weizmann Cars ViewPoint dataset

In this work we introduce a new benchmark for continuous viewpoint estimation which we call the "Weizmann Cars ViewPoint" dataset. The dataset is available for download at http://www.wisdom.weizmann.ac.il/ vision/WCVP.

We use the training data which we collected to build our 3D model to establish a benchmark which measures performance of continuous viewpoint estimates. Publicly available datasets which have been used to benchmark pose estimation algorithms such as Pascal, Stanford 3Dpose and EPFL are restricted to discrete pose classification tasks. The suggested dataset which was generated using SfM [2] associates a continuous viewpoint label with each image. The need for

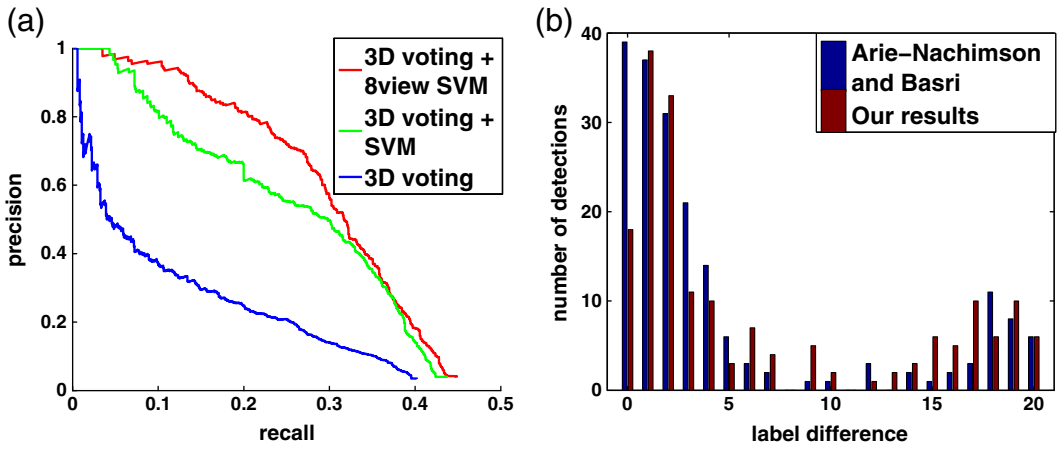

Fig. 6. Pascal VOC 2007 cars. (a) Recall-precision. 3D voting followed by 8-view SVM (red) outperforms 3D voting (blue) and 3D voting followed by SVM (green). We achieve an average precision of 32.03% without using positive training examples from Pascal. (b) Pose estimation. A subset of the cars was annotated with one of 40 different labels corresponding to approximately uniform samples of the azimuth range. We show our label differences alongside those reported in [20].

such a benchmark has already been identified in [18] where the authors manually aligned a CAD model to real images of a single car taken from various viewpoints.

In our dataset we have 1539 images of cars, composed of 22 sets. Each set consists of approximately 70 images and shows a different car model. The pictures were taken in an unconstrained outdoor setting using a hand-held camera. There are significant illumination changes, and in some images the car is cropped or occluded. Unlike other datasets such as EPFL and Stanford 3Dpose, many images in our dataset also include cars in the background, see Fig. 4(a). These less restricted conditions make for a more challenging detection task.

Denote the set of images by $\{x_i\}_{i\in I}$ where

$$I = \{(1,1),(1,2),\ldots,(1,65),\ldots,(22,56)\}$$

is a set of pairs of indices, the first is a car index and the second an image index. Each image is associated with an annotation $y_i = (BB_i, vp_i)$. Here $BB_i = (mincol_i, minrow_i, maxcol_i, maxrow_i)$ is a 4 vector indicating the coordinates of the bounding box of the car in the image and $vp_i = (\theta_i, \phi_i)$ is a two vector representing the azimuth and elevation of the camera respectively. Azimuth angles are in the range $[-180^\circ, 180^\circ]$ where 0° corresponds to a back view car and 90° to a right facing car. Elevation angles are in the range $[-90^\circ, 90^\circ]$. There is a single annotation y_i associated with an image x_i and this annotation always refers to the largest car in the image, more precisely the car whose bounding box in the image has the largest area.

5.4.1. Estimation task

The estimation task is to generate a set $\{vp_i\}_{i\in I} = \left\{\left(\hat{\theta}_i, \hat{\phi}_i\right)\right\}_{i\in I}$ of predictions for the viewpoints. We partition the 22 car models into three sets of sizes 7, 7 and 8 and generate a corresponding partition of the images $I = I_1 \cup I_2 \cup I_3$. We go over these three subsets using one as a test set and the other two as training. Thus, in order to generate a pose estimate vp_i for some $i \in I_p$ we allow the use of all pairs $\left(x_j, y_j\right)_{\{j\in I_q | q\neq p\}}$.

Table 2
Results on Stanford 3Dpose cars. Average precision (AP) and average accuracy (AA) for pose estimation. Average accuracy is computed by normalizing the columns of the confusion matrix to sum to 1 and taking the mean value on the diagonal.

	Voting	Voting + SVM	Voting + 8view-SVM
AP	90.17%	94.85%	**99.16**%
AA		83.88%	**85.28**%

Bold face denotes best result.

5.4.2. Evaluation

We score an estimate by generating a vector of azimuth errors and a vector of elevation errors

$$err_i^\theta = min\left\{\hat{\theta}_i - \theta_i \pmod{360}, \theta_i - \hat{\theta}_i \pmod{360}\right\} \quad (8)$$

$$err_i^\phi = \left|\hat{\phi}_i - \phi_i\right|. \quad (9)$$

We summarize each one of these error vectors with three statistics, the median, the mean and the standard deviation. The median provides a measure which damps the effect of flipped estimates which are quite common. As a more detailed summary of the errors we also generate 46 bin histograms of these vectors.

5.4.3. Pose estimation on our car dataset

We conclude with a pose estimation experiment intended to serve as a baseline result on the WCVP dataset. We ran our method on the 1539 images in the dataset and extracted viewpoint estimates from the single top scoring detection for each image. The results are summarized in Table 3 46 bin histograms of the azimuth and elevation errors are shown in Fig. 8(b) and c.

5.4.4. Effect of pose estimate on detection

Of course the pose estimates generated in the voting stage are not always accurate. When the pose estimate is incorrect we will apply the wrong viewpoint specific classifier. We ran a few experiments to study the effect of choosing the wrong classifier. Interestingly we observed that even when we use the wrong classifier, it can still improve detection. We chose 100 candidate detections generated by the voting procedure on images from the Stanford 3Dpose data. We chose detections for which the pose estimate was accurate but the bounding box was not (i.e. its overlap with the ground truth bounding box as defined in Eq. (7) was smaller than 0.5). When we applied the refinement process using 8 viewpoint specific classifiers, 56 of the bounding boxes were successfully corrected. We then applied the same refinement process but instead of using the classifier corresponding to the pose estimate we picked one at random, this procedure still corrected 46 of the bounding boxes.

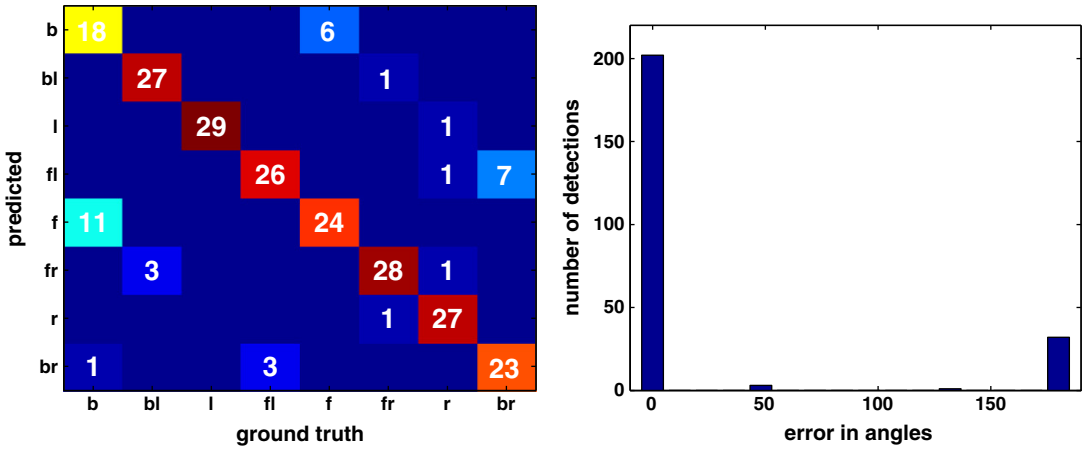

Fig. 7. Stanford 3Dpose cars — pose estimation. A confusion matrix and a histogram of label differences. Average accuracy is 85.28%.

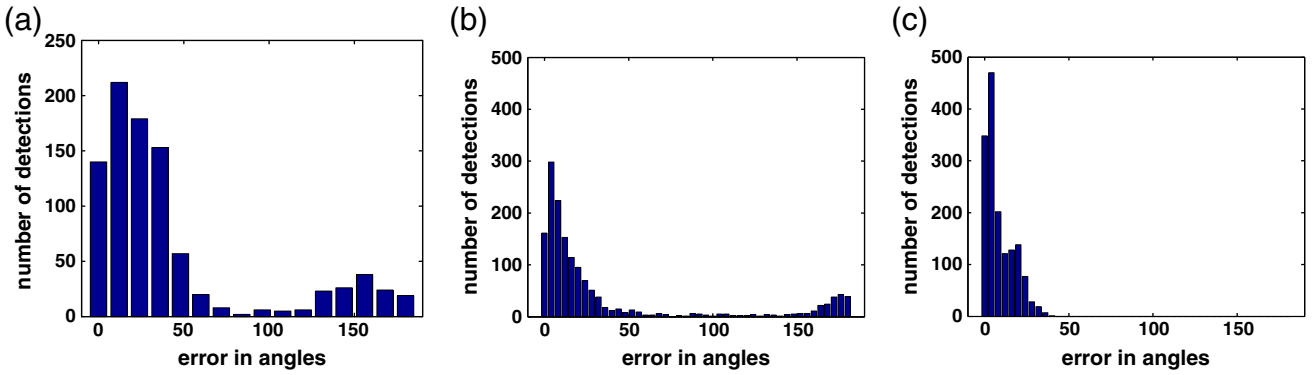

Fig. 8. Pose estimation results. (a) EPFL multiview cars — pose estimation. A histogram of the angular errors in pose estimates. Our median angular error is 24.83°. Computed on an 89.54% AP detection set. (b) Weizman Cars ViewPoint — azimuth estimation. A histogram of the azimuth errors. Our median azimuth error is 12.25°. Weizman Cars ViewPoint — elevation estimation. A histogram of the elevation errors. Our median elevation error is 5.41°.

6. Conclusions

In this paper we have described an approach that handles detection and viewpoint estimation as a joint task, and integrates reasoning about appearance and shape of the objects in a "native" way. Along the way we have made a number of choices that stand in contrast to related work in the literature. One is the construction of a nonparametric model, which maintains multiple instances of objects and multiple features without quantization or clustering. Another is to reason about detection and viewpoint jointly in a 6D parameter space, and to parameterize hypotheses in this space by means of projecting a set of designated points. Finally, we use the viewpoint estimate provided by the voting method to apply viewpoint-aware verification and refinement mechanism. We believe that these choices all serve to improve performance of the detector, as demonstrated in our experiments. In addition we introduce the "Weizmann Cars ViewPoint" (WCVP) dataset a new benchmark for measuring the performance of algorithms for continuous viewpoint estimation.

Table 3
Results on Weizmann Cars ViewPoint. Statistics of azimuth and elevation errors achieved by our method.

	Azimuth	Elevation
Median	12.25°	5.41°
Mean	36.44°	8.66°
Std	55.32°	8.18°

In the future, we would like to extend our framework to other object categories, in particular non-rigid ones. We believe that the voting process can be significantly improved in terms of accuracy as well as efficiency. Better similarity measures such as those proposed in [35,36] should be explored. In the current system each vote counts equally. An interesting direction is to assign weights to the 3D database elements. Discriminative learning of these weights can lead to improved performance [5]. Finally, techniques such as Locality Sensitive Hashing [37], Vocabulary Trees [38] or Decision Forests [39] can be used to improve efficiency.

Acknowledgments

Research supported in part by the Israeli Ministry of Science and Technology, grant number 3-8700, by the Israel Science Foundation grant number 628/08 and by the Vulcan Consortium funded by the Magnet Program of the Israeli Ministry of Commerce, Trade and Labor, Chief Scientist Office. The vision group at the Weizmann Institute is supported in part by the Moross Laboratory for Vision Research and Robotics.

References

[1] M. Fischler, R. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (1981) 381–395.
[2] N. Snavely, S. Seitz, R. Szeliski, Photo tourism: exploring photo collections in 3d, in: ACM Transactions on Graphics, TOG 2006 ACM, vol.25, 2006, pp. 835–846.

[3] Y. Furukawa, J. Ponce, Accurate, dense, and robust multiview stereopsis, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 1362–1376.
[4] B. Leibe, A. Leonardis, B. Schiele, Robust object detection with interleaved categorization and segmentation, Int. J. Comput. Vis. 77 (2008) 259–289.
[5] S. Maji, J. Malik, Object detection using a max-margin Hough transform, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 1038–1045.
[6] S. Savarese, L. Fei-Fei, 3d generic object categorization, localization and pose estimation, in: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, IEEE, 2007, pp. 1–8.
[7] M. Ozuysal, V. Lepetit, P. Fua, Pose estimation for category specific multiview object localization, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 778–785.
[8] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, L. Van Gool, Towards multi-view object class detection, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on IEEE, vol. 2, 2006, pp. 1589–1596.
[9] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, Ieee, 2008, pp. 1–8.
[10] Y. Li, L. Gu, T. Kanade, A robust shape model for multi-view car alignment, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, Ieee, 2009, pp. 2466–2473.
[11] C. Gu, X. Ren, Discriminative mixture-of-templates for viewpoint classification, in: Computer Vision–ECCV 2010, Springer, 2010, pp. 408–421.
[12] B. Pepik, M. Stark, P. Gehler, B. Schiele, Teaching 3D geometry to deformable part models, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 3362–3369.
[13] M. Sun, H. Su, S. Savarese, L. Fei-Fei, A multi-view probabilistic model for 3D object classes, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 1247–1254.
[14] H. Su, M. Sun, L. Fei-Fei, S. Savarese, Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 213–220.
[15] J. Liebelt, C. Schmid, K. Schertler, Viewpoint-independent object class detection using 3D feature maps, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.
[16] M. Stark, M. Goesele, B. Schiele, Back to the future: learning shape models from 3D cad data, in: British Machine Vision Conference, BMVC, Aberystwyth, Wales, 2010.
[17] J. Liebelt, C. Schmid, Multi-view object class detection with a 3d geometric model, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1688–1695.
[18] M. Zia, M. Stark, B. Schiele, K. Schindler, Revisiting 3d geometric models for accurate object shape and pose, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 569–576.
[19] Y. Xiang, S. Savarese, Estimating the aspect layout of object categories, in: S. Belongie, A. Blake, J. Luo, A. Yuille (Eds.), Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, Providence, United States, 2012.
[20] M. Arie-Nachimson, R. Basri, Constructing implicit 3D shape models for pose estimation, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 1341–1348.
[21] M. Villamizar, H. Grabner, J. Andrade-Cetto, A. Sanfeliu, L. Van Gool, F. Moreno-Noguer, K. Leuven, Efficient 3D object detection using multiple pose-specific classifiers, in: British Machine Vision Conference, BMVC 2011, BMVA, 2011, pp. 1–20.
[22] M. Sun, G. Bradski, B. Xu, S. Savarese, Depth-encoded hough voting for joint object detection and shape recovery, in: Computer Vision–ECCV 2010, 2010, pp. 658–671.
[23] N. Payet, S. Todorovic, From contours to 3D object detection and pose estimation, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 983–990.
[24] W. Hu, S. Zhu, Learning a probabilistic model mixing 3D and 2D primitives for view invariant object recognition, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 2273–2280.
[25] O. Boiman, E. Shechtman, M. Irani, In defense of nearest-neighbor based image classification, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, Ieee, 2008, pp. 1–8.
[26] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 1330–1334.
[27] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 1627–1645.
[28] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, Meshlab: an open-source mesh processing tool, in: Sixth Eurographics Italian Chapter Conference, 2008, pp. 129–136.
[29] P. Besl, N. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 239–256.
[30] D. Mount, S. Arya, Ann: a library for approximate nearest neighbor searching, in: CGC 2nd Annual Fall Workshop on Computational Geometry, 1997.
[31] D. Glasner, G. Shakhnarovich, Nonparametric voting architecture for object detection, TTIC Technical Report, , 2011.
[32] A. Vedaldi, V. Gulshan, M. Varma, A. Zisserman, Multiple kernels for object detection, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 606–613.
[33] M. Van, A. Zisserman, The Pascal visual object classes (voc) challenge, J. Con. purVis. 88 (2010) 3–338.
[34] D. Glasner, M. Galun, S. Alpert, R. Basri, G. Shakhnarovich, Viewpoint-aware object detection and pose estimation, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1275–1282.
[35] G. Shakhnarovich, Learning task-specific similarity, Ph.D. thesis, Massachusetts Institute of Technology, 2005.
[36] C. Strecha, A.M. Bronstein, M.M. Bronstein, P. Fua, LDAHash: Improved matching with smaller descriptors, Pattern Analysis and Machine Intelligence, IEEE Transactions on, IEEE 34 (1) (2012) 66–78.
[37] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, ACM, 1998, pp. 604–613.
[38] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on Ieee, vol. 2, 2006, pp. 2161–2168.
[39] A. Criminisi, Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning, Found. Trends® Comput. Graph. Vis. 7 (2011) 81–227.