Analysis by Synthesis: 3D Object Recognition by Object Reconstruction

Mohsen Hejrati
University of California, Irvine
shejrati@ics.uci.edu

Deva Ramanan
University of California, Irvine
dramanan@ics.uci.edu

Abstract

We introduce a new approach for recognizing and reconstructing 3D objects in images. Our approach is based on an **analysis by synthesis** *strategy. A forward synthesis model constructs possible geometric interpretations of the world, and then selects the interpretation that best agrees with the measured visual evidence. The forward model synthesizes visual templates defined on invariant (HOG) features. These visual templates are discriminatively trained to be accurate for inverse estimation. We introduce an efficient "brute-force" approach to inference that searches through a large number of candidate reconstructions, returning the optimal one. One benefit of such an approach is that recognition is inherently (re)constructive. We show state of the art performance for detection and reconstruction on two challenging 3D object recognition datasets of cars and cuboids.*

1. Introduction

We focus on the task of recognizing and reconstructing objects in images. Specifically, we describe a single model that simultaneously detects instances of general object categories, and reports a detailed 3D reconstruction of each instance. Our approach is based on an *analysis by synthesis* strategy. A forward synthesis model constructs possible geometric interpretations of the world, and then selects the interpretation that best agrees with the measured visual evidence. One benefit of such an approach is that recognition is inherently (re)constructive.

Challenges: Though attractive, an "inverse rendering" approach to computer vision is wildly challenging for two primary reasons. (1) It is difficult to build accurate generative models that capture the full complexity of the visual world. (2) Even given such a model, inverting it is difficult because the problem is fundamentally ill-posed (different reconstructions may generate similar images) and full of local minima (implying local search will fail).

Our approach: Our approach addresses both difficulties. (1) Instead of generating pixel values, we use forward models to synthesize visual templates defined on invariant (HOG) features. These visual templates are discriminatively trained to be accurate for inverse estimation. (2) We describe a "brute-force" approach to inference that efficiently searches through a large number of candidate reconstructions, returning the optimal one (or multiple likely candidates, if desired).

Latent-variable object models: Our model is related to approaches that recognize objects with latent variable models, such as the state-of-the-art deformable part model (DPM) [5]. Zhu et al. [23] point out that DPMs implicitly synthesize a set of deformed templates by searching over possible latent values. The deformation set is limited to obey sparse 2D spring constraints, making the search amenable to dynamic programming. In contrast, we *explicitly* synthesize a massive set of templates by enumerating over latent parameters in an arbitrarily-complex forward model (that explicitly constructs 3D objects and cameras). We perform a brute-force search through these (re)constructions. Surprisingly, by making use of part indexing, our search can be even faster than a DPM.

Overview. We introduce our shape synthesis model in Sec. 3 and specify the forward rendering process for generating 2D templates in Sec. 4. We then describe our algorithms for inference in Sec. 5 and learning in Sec. 6. We conclude with experimental results in Sec. 7. Our model produces state-of-the-art benchmark performance for detection and reconstruction of cuboids in indoor images [21] and cars from the PASCAL dataset [9]. We also present a diagnostic analysis that shows that, in some cases, our synthesis model is close to optimal (given our feature space).

2. Related Work

3D categorical models: Many approaches represent object categories using local features and their geometric arrangement in a 3D coordinate system [16, 17, 24]. Most related to us are approaches based on view-based part models [13, 8, 15, 5]. In particular, [15] learn view-based car models making use of a geometric CAD model to generate synthetic training images. Instead of synthesizing images, we synthesize feature templates (an easier task). We syn-

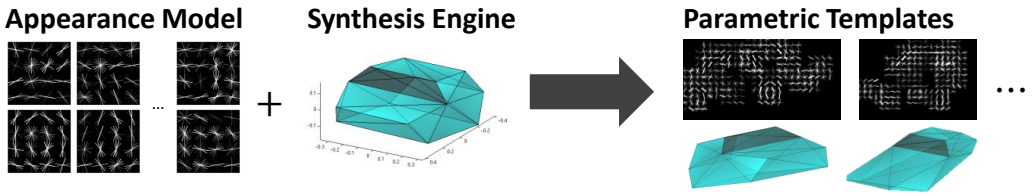

Figure 1. We describe a method for synthesizing a large set of discriminative templates, each associated with a candidate 3D reconstruction of an object (in this case, cars). Our model makes use of a generative 3D shape model to synthesize a large collection of 2D landmarks, which in turn specify rules for composing 2D templates out of a common pool of parts.

thesize templates that are detailed enough to perform 3D reconstruction, while this may be difficult for view-based approaches (since many views may be needed).

Geometric indexing: Historically, many model-based recognition systems proceeded by aligning 3D models to image data. Typically, a sparse set of local features are initially detected, after which alignment is treated as a feature correspondence problem. Efficient correspondence search is implemented through affine/projective invariant indexing [6], geometric hashing [12], or simple enumeration [7]. Hough transforms use sparse feature detections to vote in a shape parameter space [2], resulting in 2D implicit shape models [1, 18]. Our part indexing scheme is similar in spirit, except that we use a dense set of part responses to cast votes for a discrete set of candidate 3D reconstructions.

Model synthesis: Synthetic parametric models of object-categories is an active area of research in the graphics community. Approaches include procedural grammars [14], morphable basis shapes [4], and component/part-based models [11]. We make use of morphable models to represent categorical shape variation, as in [9]. Following past work, we show that such basis models can be learned from 2D annotations using techniques adapted from non-rigid structure from motion (SFM) [20, 19]. Our models differ from past work in that we use a geometric model to synthesize a large set of exemplars, which are then used for "brute-force" matching. From this perspective, our approach is similar to work that relies on a non-parametric model of object shape [3].

3. Synthesis model

We begin by defining a parametric model for constructing 3D shapes from a particular object category (such as cars). We will then sample from this parametric family to define a large set of candidate 3D reconstructions. Our 3D shape model should capture nonrigid shape variation within an object category (sedans and SUVs look different) and viewpoint variation. To do so, we make use of morphable basis models [4, 9] that model any 3D shape instance as a linear combination of 3D basis shapes. Our shape model should also encode changes in appearance due to geometric variation (wheels look different when foreshortened). To do so, we learn separate local templates for landmarks conditioned on their 3D geometry (learning separate templates for frontal vs. foreshortened wheels).

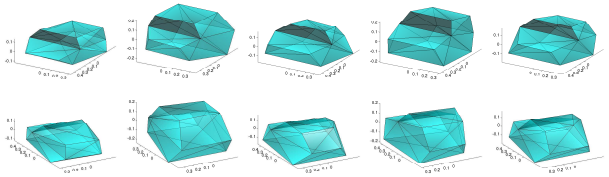

Figure 2. Following [9], we use basis shapes to model different types of cars, like sedans in the first column and SUVs in the fourth.

Shape parameters: We represent the 3D shape of an object with a set of N 3D keypoints, represented as $B \in \mathbb{R}^{3\times N}$. Given a set of n_B basis shapes $\{B_j\}$ and coefficients α, we synthesize a 3D shape B as follows:

$$B = B_0 + \sum_{j>0}^{n_B} \alpha_j B_j, \qquad \text{where} \qquad B, B_j \in \mathbb{R}^{3\times N} \quad (1)$$

where B_0 is the mean 3D shape. We visualize our shape basis in Fig. 2.

Camera parameters: We transform B into camera coordinates by rotating by R and translating by t

$$P = t + RB, \quad t \in \mathbb{R}^3, R \in \mathbb{R}^{3\times 3} \quad (2)$$

When augmented with camera intrinsic parameters (the focal length), the set of camera parameters are (t, R, f). We now summarize our parameters with a vector θ:

$$\begin{aligned} \theta &= \begin{bmatrix} Shape & Camera \end{bmatrix} \quad (3) \\ Shape &= \begin{bmatrix} \alpha_1 & \alpha_2 \dots \alpha_k \end{bmatrix} \\ Camera &= \begin{bmatrix} t & R & f \end{bmatrix} \end{aligned}$$

Forward projection: Given a parameter vector, we generate a set of 2D keypoints, scales, and local mixtures with the following:

$$Render(\theta) = \{(z_i, m_i) : i = 1 \dots N\} \quad (4)$$

$$z_i = (x_i, y_i, \sigma_i) = (f\frac{p_x^i}{p_z^i}, f\frac{p_y^i}{p_z^i}, \frac{f}{p_z^i}) \quad (5)$$

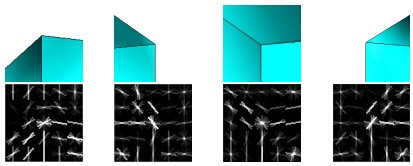

Figure 3. We learn local part mixtures by clustering the relative 3D position of keypoint i and its connected neighbors in the underlying 3D mesh. We show keypoint cluster means μ_k^i (**above**), along with their associated part templates β_i^k (**below**). Each synthesized 3D pose (and associated template) is constructed by adding together shifted copies of local part templates, which in turn allows for efficient run-time search.

where (5) is a standard perspective projection model, and p^i is the i^{th} column of matrix P. We have assumed unity-scaled pixels factors for simplicity (though they can easily be added).

Appearance synthesis: To capture changes in appearance caused by geometry (frontal and foreshortened wheels look different), we associate each keypoint with a discrete mixture m_i. We will use mixture-dependent local templates $\beta_i^{m_i}$ to capture such appearance variability. We now describe a simple approach for synthesizing m_i conditioned on P (the view-dependent 3D geometry). Let us define $rel_i(P)$ to be a vector of relative 3D landmark locations:

$$rel_i(P) = \{p_j - p_i : j \in N(i)\}. \tag{6}$$

where $N(i)$ is the set of keypoints connected to i under the 3D mesh model. We use the 3 other keypoints with highest spatial correlation to i. Offline, we extract the set of $\{rel_i\}$ from the set of synthesized shapes, and cluster them using k-means. We write the k^{th} mean as μ_k^i. Given this clustering, we now can synthesize mixture labels m_i by finding the closest geometric mean:

$$m_i = k^* \quad \text{where} \quad k^* = \underset{k \in M}{\operatorname{argmin}} ||rel_i(P) - \mu_k^i||^2. \tag{7}$$

We show 3D geometric means μ_k^i and their associated appearance-specific visual templates β_i^k in Fig. 3.

Parameter quantization: We explore various strategies for producing a set of parameters θ. One option is to use the set of parameters encountered in a set of training images. Alternatively, we can synthesize a set of parameters θ with a grid search over a bounded range of parameters (where bounds on the camera rotation matrix is defined in terms of elevation and azimuth Euler angles). In either case, we clamp camera translations to be 0 ($t = 0$) to ensure translation-invariance. We do search over focal lengths f to model perspective effects during synthesis. This produces a massive set of thousands or millions of parameters vectors, produced by enumerating over a training set or a grid search. In our results, we experiment with various quantized subsets. We wish to quantize together parameters that yield similar 2D projections. Specifically, we construct a vector of 2D (x_i, y_i) keypoint positions for each discrete θ, and cluster this set with K-means. We denote the final set of K-quantized parameter vectors as

$$\Omega_K = \{\theta_1 \dots \theta_K\} \tag{8}$$

4. Template model

Given a parameter vector θ and image I, we describe a method for scoring a visual template w_θ:

$$S(I, \theta) = \sum_{u \in U} w_\theta[u] \cdot I[u] \tag{9}$$

where $I[u]$ is an image feature extracted from a pixel location and scale $u = (x, y, \sigma)$ in image I. We write U for the set of all possible discrete pixel locations and scales enumerated in a feature pyramid. In practice, w_θ is a single-scale template with local spatial support. For notational simplicity, we assume that templates are zero-padded (across space and scales).

To efficiently represent our family of templates, we construct each template w_θ by adding together local keypoint templates shifted to lie at locations given by $Render(\theta)$ (5). We write $\beta_i^{m_i}$ for the (zero-padded) local visual template, or "part", associated with keypoint i, when tuned for mixture m_i:

$$S(I, \theta) = \sum_{i=1}^{N} \sum_{u \in U} \beta_i^{m_i}[u] \cdot I[u + z_i] \tag{10}$$

where we drop the dependance of the rendered keypoint location $z_i = z_i(\theta)$ and mixture $m_i = m_i(\theta)$ on parameter θ. If keypoint i is occluded given the viewpoint specified by θ, then the associated m_i acts as an occlusion-specific mixture. In such cases, the learned template $\beta_i^{m_i}$ may be set to all zeros, or it may capture image features characteristic of occlusions (such as t-junctions).

Let us define a dummy indexing variable $u' = u + z_i$ and switch the order of summations in the above equation. This allows us to write the global template w_θ from (9) as a superposition of shifted keypoint templates:

$$w_\theta[u] = \sum_{i=1}^{N} \beta_i^{m_i}[u - z_i] \tag{11}$$

where we have assumed keypoint templates β are zero-padded outside of their default spatial extent.

5. Inference

Inference corresponds to computing

$$\max_{\theta \in \Omega_K} S(I, \theta) \tag{12}$$

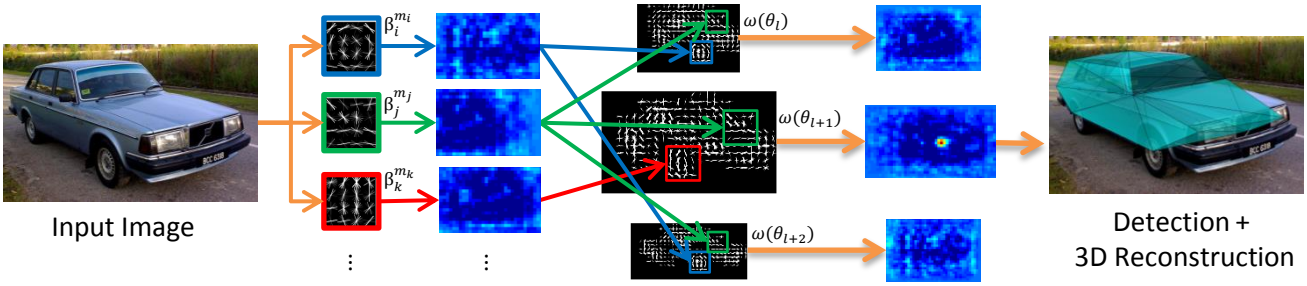

Figure 4. We search through a large collection of templates (with shared parts) by first caching part responses, and then looking up response values to score each template.

To simplify notation, we assume that the translation-invariant set of parameters $\theta \in \Omega_K$ are augmented with camera translations at run-time. This allows the above maximization to perform a scanning-window search over image translations and scales. To efficiently search over scores for all $\theta \in \Omega_K$ given an image I, we first pre-compute a response map of keypoint template responses for each location u:

$$R_i^{m_i}[u] = \sum_{u' \in U} \beta_i^{m_i}[u'] \cdot I[u' + u] \tag{13}$$

We pre-compute the above response map for each keypoint i and mixture m_i by convolving the feature pyramid I with the part template $\beta_i^{m_i}$. We now can define the score associated with a particular object parameter by looking up values in the cached response maps:

$$S(I, \theta) = \sum_{i=1}^{N} R_i^{m_i}[z_i(\theta)] \tag{14}$$

The final inference algorithm, visualized in Fig. 4, is as follows.

1. Offline, enumerate parameters $\theta \in \Omega_K$ and cache the associated set of rendered keypoints $Render(\theta)$.
2. Online, given an image, compute the response map for all N parts and M mixtures (13).
3. Evaluate $S(I, \theta)$ for each $\theta \in \Omega_K$ with N (≈ 10) table lookups (14).
4. Return parameters θ above a detection threshold, along with their associated reconstructions B (1).

6. Learning

Our models require two sets of parameters; those associated with shape synthesis θ, and those associated with local keypoint templates $\beta_i^{m_i}$. We learn both using training images annotated with 2D keypoint locations.

Synthesis parameters: In some cases, one can use a graphics engine or CAD models to directly synthesize a set of 3D parameter vectors θ. We can also infer such 3D parameters from 2D keypoint annotations so as to minimize 2D reprojection error. As in [9], we employ nonrigid structure from motion (SFM) [19] to learn a 3D basis. Stack all 2D keypoints from N training images into a $2N \times K$ matrix. In the noise-free case, this matrix is rank $3n_B$ (where n_B is the number of basis shapes), since each row can be written as a linear combination of the 3D coordinates of n_B basis shapes. This means that one can use rank constraints to learn a 3D morphable basis. We use the publicly-available nonrigid SFM code from [9] to estimate both the shape basis β and parameters θ for each training example. We learn local appearance clusters μ by clustering view-dependent 3D shapes obtained from the set of $\{\theta_i\}$.

Template parameters: We will learn templates that are discriminatively tuned for accurate detection and reconstruction on single-view training images. Assume we are given supervised training data including positives $\{I_i, \theta_i\}$ and negatives $\{I_i\}$. Oftentimes supervision is more naturally specified in terms of 2D keypoint annotations rather than 3D shapes. In such a scenario, we use the nonrigid SFM procedure from the previous paragraph to estimate shape parameters θ given 2D keypoint annotations. Combining (9) with (11), we can explicitly denote the score as linear in keypoint templates $\beta = \{\beta_i^{m_i}\}$:

$$S(I, \theta) = \beta \cdot \Phi(I, \theta) \tag{15}$$

We will learn templates that minimize the following training objective function:

$$\min_{\beta} \frac{1}{2}||\beta||^2 + C \sum_i \xi_i \tag{16}$$

$$\text{s.t.} \quad \forall i \in \text{pos}, \quad \beta \cdot \Phi(I_i, \theta_i) \geq 1 - \xi_i \tag{17}$$

$$\forall i \in \text{neg}, \forall \theta \in \Omega_K, \quad \beta \cdot \Phi(I_i, \theta) \leq -1 + \xi_i \tag{18}$$

The above constraint states that positive examples should score better than 1 (the margin), while negative examples, for all configurations of keypoints positions and mixtures defined by Ω_K, should score less than -1. Violations of these constraints are penalized through slack terms. We

find margin violations on negative images (not containing the object) by running the efficient inference algorithm from Sec. 5 to find detections that score above -1. This form of learning is known as a structural SVM, and there exist many well-tuned solvers such as SVMStruct [10] and stochastic gradient descent [5]. We use a stochastic dual coordinate-descent implementation based on [22].

Scalability: Training time scales roughly linearly with K (the number of synthesized shapes). This holds true because β is independent of K, while hard-negative mining scales linearly with K since each shape must be enumerated. For large K, we found that one could speed up training times by stochastically subsampling shapes during hard-negative mining without sacrificing accuracy. We subsampled a fixed number (50) regardless of K, making training time practically independent of K.

Recognition vs reconstruction: The above constraints naturally corresponds to detection accuracy. One could augment them to ensure that, for a positive example I_i, the true shape θ_i outscores incorrect shapes $\theta \neq \theta_i$ by some amount. This corresponds to a structured prediction task that explicitly trains parameters so as to generate accurate shapes. We found that these additional constraints did not improve performance given a large enough negative set (we use a generic set of 1000 outdoor images).

Data scarcity: Interestingly, the above formulation learns accurate models even with a small number of positives that are dwarfed by the the number of templates $|pos| \ll |\Omega_K|$. It may seem strange that *we are learning templates for shapes that have never been seen* - but this is precisely the benefit of synthesis! We learn good templates so long as there exist enough positives to train the local parts $\beta_i^{m_i}$. Given this fact, one might be tempted to simply train the local parts independently, but the above structured formulation takes advantage of contextual interactions between all parts, defined by the *entire* set of parameters Ω_K. Because "hard negative" margin violations are produced by searching across all templates in Ω_K, the learning algorithm above will tend to produce a strong set of models $\{w_\theta : \theta \in \Omega_K\}$.

7. Results

Datasets: We evaluate our model using two object detection/ datasets. The SUN primitive dataset [21] contains 785 images with 1269 annotated cuboids. The UCI-Car dataset [9] contains 500 images from PASCAL VOC2011 containing 723 cars with detailed landmark annotation. For both datasets, we use the same train-test split provided by the curators for training and evaluation.

Evaluation: Our models report back object detections with associated 3D reconstructions. Because annotating images with 3D shapes is cumbersome, we evaluate our reconstructions by evaluating 2D landmark re-projection error. This allows us to use standard benchmarks and compare to past work. For object detection we use now-standard average precision measure introduced in PASCAL. For landmark localization, we follow [21] and plot landmark accuracy for various levels of object-detection recall. A predicted landmark is defined to correct if it lies within t pixels of the ground-truth location, where t = 15% of the square root of the area of the ground-truth box.

Baselines: We compare to previously published results for both datasets. In particular, we use state-of-the DPMs as baseline for object detection [5], and supervised tree-based part models as baselines for landmark prediction [21, 9]. Notably, both [21, 9] use a two-stage inference procedure for reconstruction, where detections from a 2D tree-based part model are refined to produce 3D reconstructions. Our model performs both detection and reconstruction in a *single* stage.

Implementation: For our car models, we set the number of basis shapes $n_b = 5$. We learn a model with $N = 20$ 3D landmarks, each modeled with $|M| = 9$ local mixtures. For our cuboid models, we manually define a 3D parametric cuboid model of varying aspect ratios. Our model consists of $N = 17$ 3D landmarks (consisting of cube corners and midpoints), with $|M| = 12$ local mixtures.

Synthesis strategies: We explored numerous strategies for constructing a set of 3D shape parameters $\{\theta_i\}$. First, our *Exemplar* model uses the shapes encountered in the training set of annotated images, augmented with synthetic camera translations. *Exemplar Synthesis* augments this set with additional exemplar shapes. We implement this strategy by learning a model with a subset of training images, but using the larger (full) set of keypoint annotations. This mimics scenarios where we have access to a limited amount of image data, but a larger set of keypoint annotations. *Parametric Synthesis* constructs a shape set by discretely enumerating θ_i over bounded parameter ranges. Finally, *Oracle Synthesis* uses shapes extracted from annotated test-data. We use this upper bound on performance (given the the "perfect" synthesis strategy) for additional analysis.

Interactive synthesis: We have implemented an interface for interactive synthesis (Fig. 5). A common tool for visualizing morphable models is an interface where a user can dynamically toggle/slide shape coefficients, and view the resulting model. We have constructed such an interface, and can use it to visualize our family of 3D shapes, camera viewpoints, and associated HOG templates. We find it to be an intuitive user experience for "understanding" the modeling capacity of our representation.

Anytime recognition/reconstruction: Our models have a free parameter K, the number of enumerated shapes. Both performance and run-time computation increase with K. When comparing to baselines with fixed run-time costs, we plot performance as a function of run-time, measured in

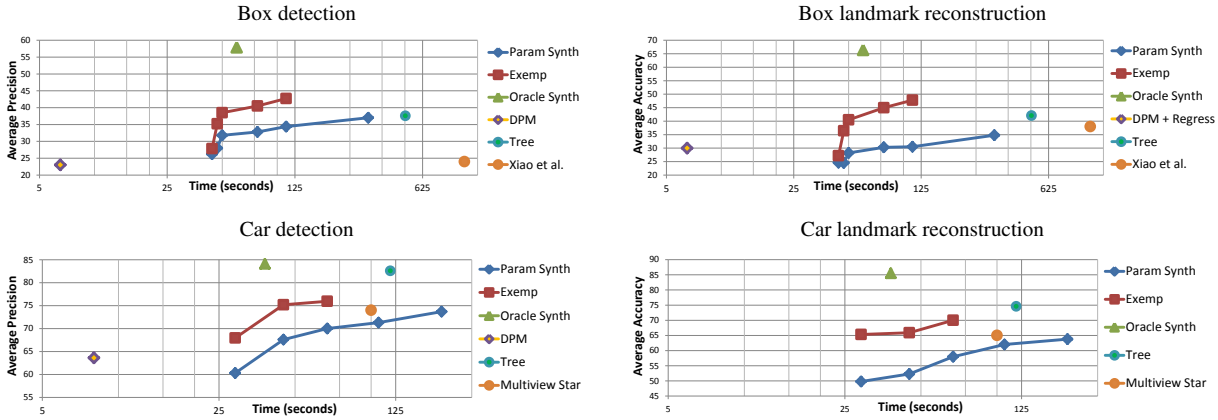

Figure 6. Detection (**left**) and reconstruction accuracy (**right**) versus running time of our method and other baselines, including DPMs [5], supervised-tree models [9], and multi-view star models [9]. Points correspond to different (constant-time) baselines, while curves correspond to our models. Because our models can process a variable number of synthesized templates, we sweep over $K \in \{20, 50, 100, 500, 1000, 4000\}$ templates to generate the curves. Note that Exemplars are limited by the number of training images. Exemplars always dominate Parametric Synthesis (for a given K), suggesting our parametric model is failing to capture important shape statistics. We examine this further in Fig. 7. Our box (**top**) detection and reconstruction results (43% and 48%) nearly double the best previously-reported performance from Xiao et al.[21] (24% and 38%), while being 10X faster. Our car (**bottom**) results approach the state-of-the-art tree models of [9], but directly report 3D shape while being 5X faster.

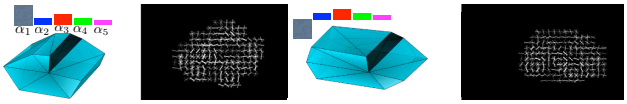

Figure 5. A visualization of our interactive, morphable interface for exploring 3D shapes and their associated templates. We display the corresponding shape coefficients α as colored bars.

terms of seconds per image. All methods are run on the same physical system (a 12-core Intel 3.5 Ghz processor). Recall that we obtain shape sets for smaller K by clustering a larger set of shapes. Our plots reveal that a simple *re-ordering* of shapes in a coarse-to-fine fashion (with hierarchical clustering) can be used for *any-time* analysis. For example, after enumerating the first $K = 20$ coarse shapes, one can still obtain 65% car landmark reconstruction accuracy (which in turns improves as more shapes are enumerated).

Box benchmark results: Fig. 6 plots performance for box detection and localization. Exemplars almost *double* the best previously-reported numbers in [21], in terms of detection (43% vs 24%) and landmark reconstruction (48% vs 38%). Interestingly, the tree model of [9] outperforms [21], perhaps due to its modeling of local part mixtures. Our models even surpass [9], while directly reporting 3D reconstructions and while being 10X faster. Exemplar and Parametric Synthesis perform similarly for low numbers of templates, but Exemplars do better with more templates, particularly with respect to reconstruction accuracy. Moreover, both methods still fall short of the upper-bound given by Oracle Synthesis. These results suggests that our parametric model is not capturing true shape statistics. For example, people may take pictures of certain objects from iconic viewpoints. Such dependancies are not modeled by Parametric Synthesis, but are captured by Exemplars. We later demonstrate (Fig. 7) that Exemplar Synthesis also captures such dependancies.

Car benchmark results: We find similar trends when evaluating detection and landmark accuracy for cars. Our models fall just shy of the tree model of [9], but directly report 3D reconstructions while being 5X faster. As before, Exemplars dominate Parametric Synthesis for any fixed number of templates. But Parametric Synthesis can potentially outperform Exemplars with additional shapes (because Exemplar is limited to observed training data). Moreover, our upper-bound analysis reveals that both models are close to the upper bound provided by Oracle Synthesis. This suggests that our morphable 3D model is a rather accurate description of car shapes.

Diagnostic analysis: We present qualitative results for both boxes and cars in Fig. 8. In Fig. 7, we present further diagnostic analysis of our box model with respect to training data size. When training on small amounts of training data, Exemplar Synthesis noticeably improves performance by 5%. To realize this improvement, it is important to discriminatively-train the full synthetic set of templates. These results suggest that accurate shape statistics are crucial to realize the benefit of synthesis. Indeed, we show that one can produce a state-of-the-art model with as little as 20 training images.

Conclusions: We have introduced a new approach for recognizing and reconstructing 3D objects in images based on an analysis by synthesis strategy. We make use of forward synthesis models to synthesize a large number of possible geometric interpretations, and efficiently search

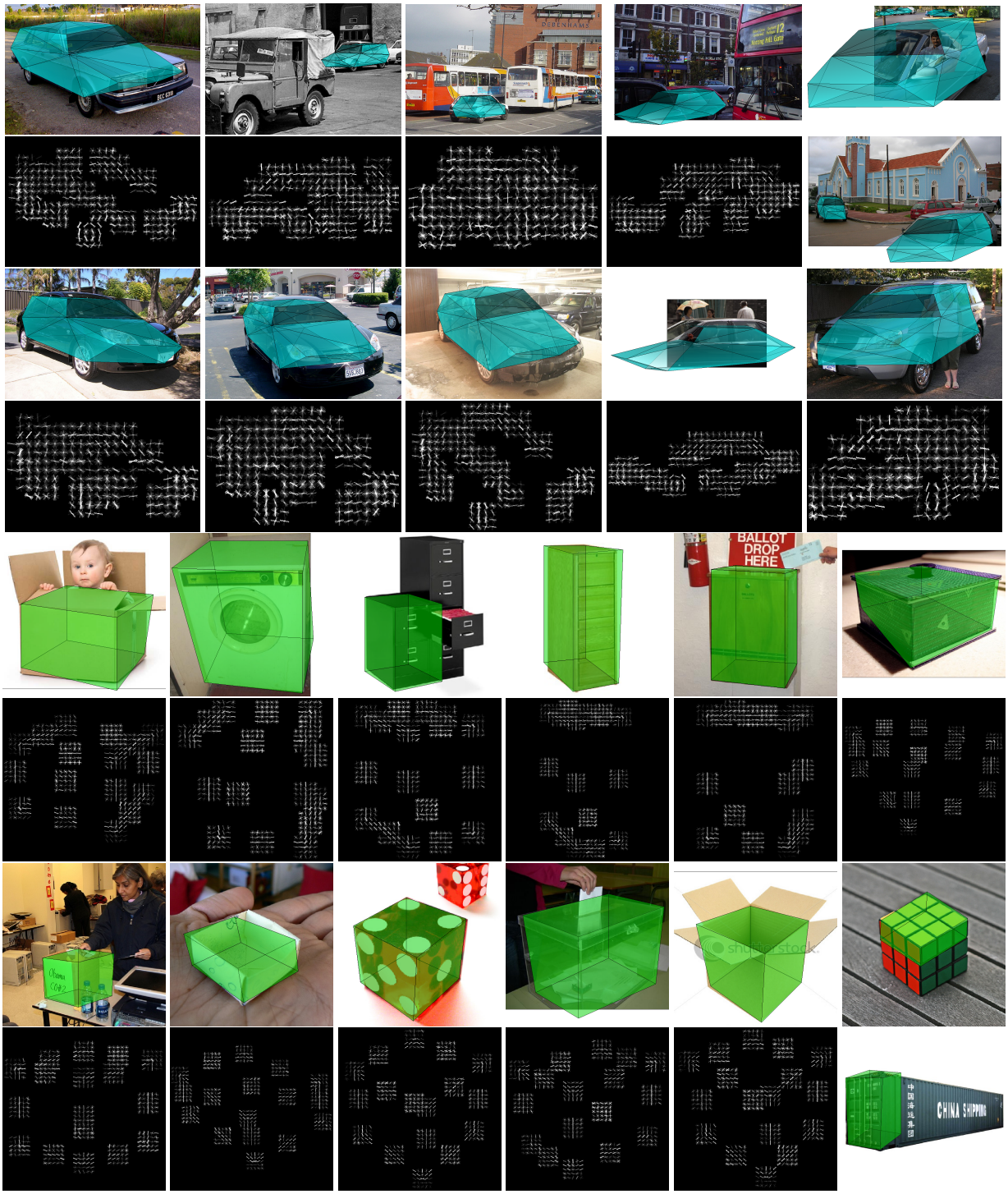

Figure 8. Recognition + reconstructions from our method. Odd rows show the test image and recognized + reconstructed object overlaid on it. Even rows illustrate the associated template that triggered the detection. Our method can recognize objects from various viewpoints, shapes and is robust to heavy occlusion. Because every synthesized template has a 3D shape, recognition is inherently reconstructive. On the top right, we show results for images with multiple cars. Our box results show accurate reconstructions across various viewpoints, aspect ratios, and even perspective effects. However, some images are genuinely ambiguous, like the Rubik's Cube (bottom-right).

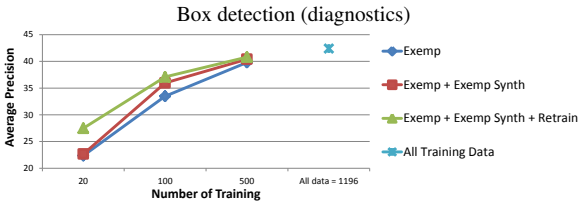

Figure 7. We plot the performance of various synthesis approaches as a function of the amount of training images. **Exemp** enumerates the set of shapes encountered in the training set of images. **+Exemp Synth** uses the learned local templates β from **Exemp** and instantiates new shapes obtained from keypoint annotations not in the training set. This improves performance by up to 2%. **+Retrain** discriminatively retrains β given this synthesized set of shapes, further improving performance by up to 5%. Hence it is crucial to discriminatively-tune the synthesized set. Our synthesis models outperform state-of-the-art methods [21] with orders-of-magnitude less training data.

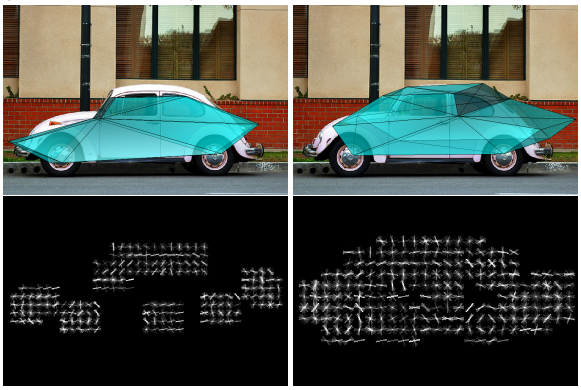

Figure 9. We show an example detection for which the reconstruction problem is fundamentally ill-posed (in our HOG feature space). Our brute-force strategy for enumerating all reconstructions can readily return multiple high-scoring interpretations, addressing a classic limitation of "inverse rendering" approaches.

through this set with indexing schemes. Our methods discriminatively train a large set of synthetic geometric models, such that they are accurate for both recognition and reconstruction. Constructing this set from an observed collections of exemplar shapes does remarkably well, but one can still improve on these results with accurate shape-driven synthesis.

Acknowledgements: Funding for this research was providedby NSF Grant 0954083 and ONR-MURI Grant N00014-10-1-0933.

References

[1] M. Arie-Nachimson and R. Basri. Constructing implicit 3d shape models for pose estimation. In *ICCV*, 2009. 2

[2] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2), 1981. 2

[3] P. N. Belhumeur, D. W. Jacobs, D. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *CVPR*, pages 545–552. IEEE, 2011. 2

[4] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999. 2

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE PAMI*, 99(1), 5555. 1, 5, 6

[6] D. Forsyth, J. L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3 d object recognition and pose. *IEEE TPAMI*, 13(10):971–991, 1991. 2

[7] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *ICCV*. IEEE, 2011. 2

[8] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. *ECCV*, pages 408–421, 2010. 1

[9] M. Hejrati and D. Ramanan. Analyzing 3d objects in cluttered images. In *NIPS*, 2012. 1, 2, 4, 5, 6

[10] T. Joachims, T. Finley, and C. Yu. Cutting plane training of structural SVMs. *Machine Learning*, 2009. 5

[11] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):55, 2012. 2

[12] Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *ICCV*, pages 238–249, 1988. 2

[13] R. Lopez-Sastre, T. Tuytelaars, and S. Savarese. Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV Workshops*, 2011. 1

[14] P. Merrell and D. Manocha. Model synthesis: A general procedural modeling algorithm. *Visualization and Computer Graphics, IEEE Transactions on*, 17(6):715–728, 2011. 2

[15] B. Pepik, M. Stark, P. Gehler, and B. Scheile. Teaching geometry to deformable part models. In *CVPR*, 2012. 1

[16] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *ICCV*, 2007. 1

[17] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3d object classes. In *CVPR*, pages 1247–1254. IEEE, 2009. 1

[18] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool. Towards multi-view object class detection. In *CVPR*, volume 2, pages 1589–1596. IEEE, 2006. 2

[19] L. Torresani, A. Hertzmann, and C. Bregler. Learning non-rigid 3d shape from 2d motion. *NIPS*, 16, 2003. 2, 4

[20] L. Torresani, D. Yang, E. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *CVPR*, volume 1, pages I–493. IEEE, 2001. 2

[21] J. Xiao, B. Russell, and A. Torralba. Localizing 3d cuboids in single-view images. In *NIPS*, 2012. 1, 5, 6, 8

[22] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011. 5

[23] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection?. In *BMVC*, pages 1–11, 2012. 1

[24] M. Zia, M. Stark, B. Schiele, and K. Schindler. Revisiting 3d geometric models for accurate object shape and pose. In *ICCV Workshops*, pages 569–576. IEEE, 2011. 1