A coarse-to-fine taxonomy of constellations for fast multi-class object detection*

Sanja Fidler[1,2], Marko Boben[1], Aleš Leonardis[1]

[1]University of Ljubljana, [2]UC Berkeley EECS and ICSI
fidler@eecs.berkeley.edu,{marko.boben,alesl}@fri.uni-lj.si

Abstract. In order for recognition systems to scale to a larger number of object categories building visual class taxonomies is important to achieve running times logarithmic in the number of classes [1, 2]. In this paper we propose a novel approach for speeding up recognition times of multi-class part-based object representations. The main idea is to construct a taxonomy of constellation models cascaded from coarse-to-fine resolution and use it in recognition with an efficient search strategy. The taxonomy is built *automatically* in a way to minimize the number of expected computations during recognition by optimizing the cost-to-power ratio [3]. The structure and the depth of the taxonomy is not pre-determined but is inferred from the data. The approach is utilized on the hierarchy-of-parts model [4] achieving efficiency in both, the representation of the structure of objects as well as in the number of modeled object classes. We achieve speed-up even for a small number of object classes on the ETHZ and TUD dataset. On a larger scale, our approach achieves detection time that is logarithmic in the number of classes.

1 Introduction

Representing objects as spatial layouts of simpler parts has been shown as an effective way of modeling generic object classes [5–7]. In order to recognize and detect a larger number of object categories in images, several works have proposed feature sharing among the objects to achieve better generalization as well as to cut down computation time [5, 7]. However, these approaches still run with time linear in the number of classes [8], since they need to scan the (shared) feature space with a detector for each class separately. In contrast, visual class taxonomies [2, 9] induce a hierarchy over the class labels: usually a hierarchical tree of classifiers is used to achieve recognition complexity logarithmic in the number of classes [1].

In this paper we propose a novel approach for speeding up multi-class object detection based on *part-based object representations* [10, 7] by constructing a visual taxonomy of constellation models cascaded from coarse-to-fine resolution

* This research has been supported in part by the following funds: EU FP7-215843 project POETICON, and Slovenian Research Agency (ARRS) research program Computer Vision P2-0214, and ARRS research projects J2-3607 and J2-2221.

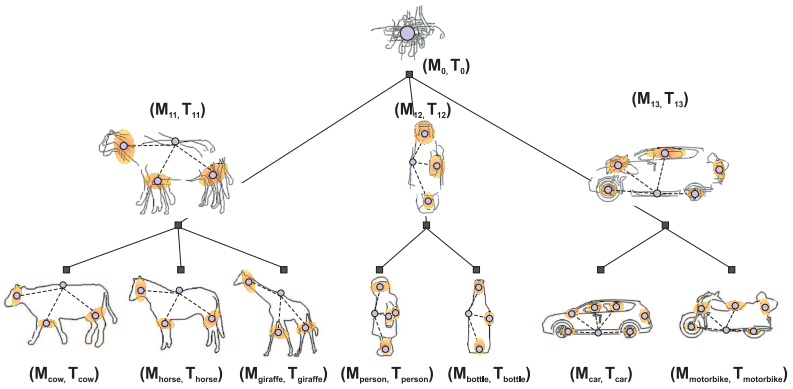

Fig. 1: A hierarchical tree of coarse-to-fine constellation models. M's denote the models, T's denote the tests (detectors for the models), explained in Sec. 4.1 and 5. The leaf nodes (object class models) are assumed known (we use [4] to learn them), whereas all the other models in the taxonomy are obtained automatically, by clustering the object models in the leaves. The depth and structure of the taxonomy are learned.

(Fig. 1). The taxonomy is generative, where each object class can be generated by following a particular path in the hierarchical taxonomy tree. The taxonomy is constructed by clustering a set of object class models into a hierarchical tree of increasingly coarser constellation models both in terms of structure as well as in the appearance of parts. The tree is built *automatically* in a way to minimize the number of expected computations during recognition by optimizing the cost-to-power ratio [3]. The structure and the depth of the taxonomic tree are not pre-determined but are inferred from the data. During recognition, our approach uses the learned taxonomy to *prune* the search space in a coarse-to-fine fashion, starting from the root using the depth-first search algorithm.

The approach is utilized on the hierarchy-of-parts model [4] achieving efficiency in both, the representation of the shape of the objects (by using a hierarchy of shareable shape features that gradually progress in complexity) as well as in the number of modeled object classes (by inducing a taxonomy over the class labels). Compared to the baseline [4], we demonstrate good speed-up even for a small number of object classes on the ETHZ [11] and TUD dataset [12]. On a larger scale (Caltech 101 [13] and LabelMe [5]), our approach achieves detection time that is logarithmic in the number of classes.

2 Related work

Prior work on multi-class object recognition is mainly concerned with speeding-up *classification* approaches [1, 8, 14], while our goal here is to speed-up *generative*, part-based object class models. Nevertheless, the ideas behind these approaches are related to ours. Zhender et al. [14] employ a hierarchical cascade of classifiers, while [8] translates the classification stage into matching in

a high-dimensional vector space, where fast, scalable solutions exist. Similarly, Stewenius and Nister [15] build a hierarchical cluster tree in the vector space (representing descriptor appearance) in an image retrieval scenario. Our problem is substantially different since we are dealing also with object geometry.

Bart et al. [9] build a taxonomy of images represented as bags of visual words in an unsupervised manner by learning a hierarchical tree of topic models. Their taxonomy is generative like ours, however, they are dealing with taxonomy of images and not objects and the representation does not take into account the geometry between features. Closer to our approach is the work by Sivic et al. [2], where the authors build a taxonomy of increasingly coarser object representations both in terms of the spatial layout (fixed grid with varying degree of resolution) and appearance (by varying the degree of clustering of SIFT features) using a Hierarchical LDA model. In our approach we use the constellation model [16, 10, 4] as the means to represent the objects and show how to build a generative taxonomy of increasingly coarser constellations.

Our work is also related to coarse-to-fine model matching which has appeared in many forms in the literature. Gavrila [17] proposed a method for hierarchical clustering of object templates and applying the cascade to speed-up template matching in the domain of pedestrian detection.

In [18], the authors propose a cascade of detectors based on the constellation model to detect *specific instances* of objects. Similarly as in our approach, the detectors are cascaded in a coarse-to-fine resolution, however, the coarsening of the spatial grid is pre-defined, while in our approach the coarsening of both the location and appearance is *learned* by optimizing the cost-to-power ratio on a set of training images. We also deal with generic object classes.

Amit and Geman [19] learn "spread" tests to check multiple object hypotheses simultaneously. Spreading corresponds to OR-ing (disjunctions) the locations of simple oriented edges so that the conjunction of these coarsely positioned features is common (shared) among several object classes. In our approach the spreading (OR-ing) is done in both location as well as in appearance making the method generally applicable to part-based models.

Our work builds on some of the theoretical ideas on coarse-to-fine search strategies [3, 20]. We apply them to our particular problem, which is building visual taxonomies of object classes for part-based object representations.

Note that our constellation-based object taxonomy differs from hierarchical representations of object structure [4, 21]. Here we induce a hierarchy on the *object class labels*, while [4, 21] deal with a hierarchy of *shape features*.

3 Overview and contributions

The problem we are tackling is multiple object class detection using part-based object representations, in particular the constellation-type models [16, 10]. We assume that we have available constellation models for a set of object classes and we want to perform recognition with them in arbitrary images (we do not know which objects are present in an image). In the original approach [16, 10],

a detector for each class separately needs to be run on a query image resulting in recognition times linear in the number of modeled objects.

The novel idea of this paper is to speed-up recognition by using a generative taxonomy of constellation object detectors organized hierarchically in a coarse-to-fine manner. Recognition will proceed from the root down by first employing a small set of coarse detectors and pruning improbable search paths. Performed in this way, the detectors in the leaves, which are the given object class models, will rarely be implemented, thus speeding up the overall recognition procedure.

This paper makes three novel contributions:

1. **Representing a visual taxonomy of object classes** with a coarse-to-fine taxonomy of constellation models.
2. **Automatic construction of the taxonomy** by minimizing the expected number of computations during recognition. We will build on the coarse-to-fine search strategies proposed by Blanchard and Geman [3].
3. **Combining the class taxonomy with a structure hierarchy** for fast multi-class object recognition.

The approach is organized as follows. In Sec. 4 we present the representation of the constellation taxonomy which will be referred to as the *taxonomic constellation tree (TCT)*. Sec. 5 explains the recognition procedure using TCT. In Sec. 6 we propose an approach to automatic construction of the TCT model.

4 Representation: coarse-to-fine constellation taxonomy

Let $\mathcal{C}$ be a set of classes. We assume we have available a *constellation-type model* [10] M_c for each class $c \in \mathcal{C}$, the collection of which forms our *database* $\mathcal{M} = \{M_c\}_{c \in \mathcal{C}}$ of object models. Note that each object class can be represented with (a mixture of) multiple models (e.g. a model per view or object articulation), however for the ease of exposition, we assume the existence of one model per class. We first define the basic constellation model in Subsec. 4.1. In Subsec. 4.2 we propose the representation using a taxonomic constellation tree.

4.1 The probabilistic constellation model of objects

From a query image I a set of features $\mathbf{F}$ along with their locations $\mathbf{X}$ is first extracted. We assume that features are discrete, i.e., each feature in $\mathbf{F}$ is characterized by *type* (e.g. a particular shape or visual word). The set of all feature types forms a *vocabulary*. We assume the object models to have a star topology, although the approach can be easily extended to more complex topologies.

Each model represents an object class as a collection of parts and spatial relations among them. We follow [10] to define the model. Each class $c \in \mathcal{C}$ is represented with a model $M_c = (P_c, \theta_c^{app}, \theta_c^g)$, $M_c \in \mathcal{M}$, which has P_c parts, appearance parameters θ_c^{app} for the parts, and geometry parameters θ_c^g, where the positions of all parts are conditioned on the position of a so-called *reference*

part, but are mutually independent conditionally on the reference part. We additionally define the assignment variable $\mathbf{h}$ of size P_c, which assigns some features in $\mathbf{F}$ to the parts in the model M_c.

The joint density is factored as follows [10]:

$$p(\mathbf{F}, \mathbf{X}, \mathbf{h}|c, M_c) = \underbrace{p(\mathbf{F}|\mathbf{h}, c, M_c)}_{appearance} \underbrace{p(\mathbf{X}|\mathbf{h}, c, M_c)}_{geometry} \underbrace{p(\mathbf{h}|c, M_c)}_{occlusion}$$

For the appearance term we have the following factorization:

$$p(\mathbf{F}|\mathbf{h}, c, M_c) = \prod_{i=1}^{P_c} p(\mathbf{F}(h_i) \mid c, \theta_{ci}^{app}) \tag{1}$$

We will assume that each part corresponds to one feature type or is modeled as a distribution over a small number of feature types, i.e. $p(f \mid c, \theta_{ci}^{app}) > 0$ for a small number of all types f in the vocabulary.

The geometry term can be factorized as follows:

$$p(\mathbf{X}|\mathbf{h}, c, M_c) = p(\mathbf{x}_R|h_R) \prod_{j \neq R} p(\mathbf{x}_j|\mathbf{x}_R, h_j, c, \theta_{cj}^{g}),$$

where R denotes the reference part. The distribution $p_{jR} := p(\mathbf{x}_j|\mathbf{x}_R, h_j, c, \theta_{cj})$ is taken to be Gaussian [10, 4], i.e., $p_{jR} = \mathcal{N}(x_j - x_R \mid \theta_{cj})$, where $\theta_{cj} = (\mu_{cj}, \Sigma_{cj})$.

In recognition, a decision whether an object of class c is present in an image (at a particular location) or not is made on the likelihood-ratio [16]:

$$\frac{p(c|\mathbf{F}, \mathbf{X})}{p(B|\mathbf{F}, \mathbf{X})} \propto \frac{\sum_{\mathbf{h}} p(\mathbf{F}, \mathbf{X}, \mathbf{h}|c, M_c)}{p(\mathbf{F}, \mathbf{X}, \mathbf{h}_0|B)}, \tag{2}$$

where B denotes the background ("no object of class c present") and $\mathbf{h}_0$ is the null hypothesis explaining all features as background. The occlusion term can be defined as in [10], however, we will not explicitly deal with it in this paper.

4.2 A coarse-to-fine taxonomic constellation tree (TCT)

Our approach exploits the fact that constellation models for similar object classes have (or share) similar spatial arrangements of parts (of possibly similar appearance) and can thus be grouped in a natural way: we can design "coarse" constellations that capture the distribution over *multiple* object classes. In particular, we will cluster the models in the database $\mathcal{M}$ to obtain a hierarchical tree $\mathcal{H}$ of constellations, where the constellations close to the root are coarse in both geometry and appearance and span the distribution over a larger number of object classes, while the constellations closer to the leaves are more specific. The leaves of the tree contain the given object constellation models from $\mathcal{M}$. This induces a taxonomic organization on object classes which will be used during detection to speed-up the computations. Fig. 1 illustrates $\mathcal{H}$.

A *taxonomic constellation tree* (TCT) $\mathcal{H}$ is represented with a tree graph, $\mathcal{H} = (V, E)$, where each node $\xi \in V$ in the tree contains a model $M_\xi = (P_\xi, \theta_\xi^{app}, \theta_\xi^g)$ and a corresponding test T_ξ (which will be explained in more detail in Sec. 5). The leaves of the tree correspond to the object models $M_c \in \mathcal{M}$ and are assumed to be known in advance. All other models in the tree are obtained by hierarchical clustering of the models in $\mathcal{M}$. By definition, the root node will be set at level 0 of the tree.

Define the *domain* $D(M_\xi)$ of a model M_ξ to be the set of object classes $\{c\}$, such that the likelihood $p(I(c)|M_\xi)$ of observing an image $I(c)$ of any of these classes under the model M_ξ is non-zero, or rather, higher than a threshold τ_ξ (discussed in Sec. 5): $D(M_\xi) = \{c \in \mathcal{C} : p(I(c)|M_\xi) > \tau_\xi\}$. A node ξ from level ℓ is a parent of node η from level $\ell + 1$ below, if the domain of the model M_η is entirely contained in the domain of M_ξ: $D(M_\eta) \subset D(M_\xi)$. This means that if the likelihood of class c is high under M_η it must also be sufficiently high under its parent M_ξ. Thus a model that has multiple children spans a distribution over *multiple* object classes, and the variances of the distribution in both appearance of parts as well as their geometry have to be large enough to accommodate for this. In the root node, we put a "trivial" model M_0 that has only one part, where θ_0^{app} is uniform over the feature space and θ_0^g is trivial.

5 Efficient inference

The main purpose of TCT is to reduce the number of full object class models that need to be evaluated at a particular region in an image during recognition. To achieve this we will evaluate TCT from the root down in a depth-first search manner and *prune* the improbable search paths (the unlikely object hypotheses).

For the purpose of pruning, we will construct a binary computationally inexpensive *test* T_ξ for each model M_ξ in $\mathcal{H}$. The test will be, during recognition, used to form a decision whether the children of the model M_ξ should be further explored or not. A test will yield a value 1 if the finer hypotheses (the children of ξ) are "worth" evaluating further, and 0 if the entire subbranch of hypotheses can be reliably eliminated from the search process. During training, each test will be designed to have no missed detections, at the expense of having some false positives. This idea is adopted from [19]. The false positives during recognition mean that some search paths will get explored even though the corresponding object classes might not be present. If some search path ends in a leaf (meaning that no parents yielded a negative answer), the likelihood-ratio as defined in (2) will be evaluated for the corresponding object class model and a full probabilistic decision will be made on the presence or absence of an object.

A test $T_\xi(I)$ will take the following form:

$$\begin{aligned} T_\xi(I) &= \mathbf{1}\left(J_\xi(I) > \tau_\xi\right), \quad \text{where} \\ J_\xi(I) &= \max_{\mathbf{h}} p(\mathbf{F}, \mathbf{X}, \mathbf{h} \mid \xi, \theta_\xi) \end{aligned} \tag{3}$$

where $\mathbf{1}(\cdot)$ is the indicator function. It checks whether the most probable hypothesis under the model M_ξ is higher than a particular threshold τ_ξ. We use

the *max* instead of a *sum* (which would correspond to the image likelihood under the model M_ξ) since it allows for further speed-ups as discussed below. If the probability of the most likely hypothesis is too small the test will be 0 and none of the children of M_ξ in the TCT will be evaluated further. If $T_\xi(I)$ is 1 the children of M_ξ get tested in a similar way. We adopt a depth-first search as in [3, 20], meaning that the child μ with the highest value of J_μ is explored first.

By using the tests as defined in (3), we assume the existence of thresholds τ_ξ which effectively separate the "foreground" (classes in the domain $D(M_\xi)$) from the "background", similarly as in [19]. As in [19], the thresholds τ_ξ will be learned such that the probability of a missed detection will be 0 (or close to 0): $p\big(T_\xi = 0|D(M_\xi)\big) = 0$, while the number of false positives will be sufficiently small: $p(T_\xi = 1|\mathit{background}, M_\xi) \ll 1$.

For further speed-up, we additionally define the individual likelihoods of the appearances and geometry under M_ξ: $J^{app}_{\xi,j}(I) = \max_{h_j} p(\mathbf{F}(h_j)|M_\xi)$ and $J^{g}_{\xi,j}(I) = \max_{h_j} p(\mathbf{x}_j|\mathbf{x}_R, h_j, M_\xi)$. While evaluating the tests in an image we can exploit the factorization of $p(\mathbf{F}, \mathbf{X}, \mathbf{h} \mid \xi, \theta_\xi)$: since the overall probability $J_\xi(I)$ should exceed the threshold τ_ξ, each of the individual probabilities $J^{app}_{\xi,j}(I)$ and $J^{g}_{\xi,j}(I)$ in the product should also satisfy this condition. This limits the allowed geometry to a small region making computations fast, while at the same time quickly prunes off assignments where the individual likelihoods are small.

6 Learning the taxonomic constellation tree

Given the database of object class models $\mathcal{M}$, our goal is to construct a taxonomic constellation tree $\mathcal{H}$ by optimizing its expected run time on a set of training images. We adopt some of the conceptual ideas from [20, 3], i.e., optimizing the power-to-cost ratio to learn the representation.

We first introduce the notation. Let $\mathcal{H}^\ell = (V^\ell, E^\ell)$ denote the subgraph of $\mathcal{H}$ at level ℓ with the set of models $\mathcal{M}^\ell$ and tests T^ℓ defined in the nodes V^ℓ. Denote with $\theta^{g,\ell}$ and $\theta^{app,\ell}$ the parameters of the models at level ℓ and with $\mathcal{V}^\ell$ the vocabulary of feature types at this level. Further, let $B(M_\xi)$ denote the background interpretation under M_ξ, which is any interpretation outside of the model's domain $D(M_\xi)$. Define the missed detection rate of a test with $\alpha(T_\xi) = p\big(T_\xi = 0|D(M_\xi)\big)$. Each test will be assigned a *cost* denoted with $t(T_\xi)$, which can be measured with dedicated CPU time, and *power* denoted with $\beta(T_\xi) = p\big(T_\xi = 0|B(M_\xi)\big)$. The power is thus the selectivity of the test. It is shown in [3], that under certain conditions, a sufficient condition for the optimality of the coarse-to-fine search is:

$$\forall \xi \in V: \quad \frac{t(T_\xi)}{\beta(T_\xi)} \leq \sum_{\eta=\mathrm{child}(\xi)} \frac{t(T_\eta)}{\beta(T_\eta)}, \tag{4}$$

which will motivate our learning objective function. The tree will be built from the leaves up. The models at a particular level ℓ will be obtained by clustering the models from the previous level, $\ell + 1$ (the root node will be at level 0). Each

(clustered) model $M_\xi \in \mathcal{H}^\ell$ will be evaluated with respect to the effectiveness of its corresponding test (detector) T_ξ.

Assume we already have a model M_ξ. Its test T_ξ (which has one parameter: a threshold τ_ξ) is learned as follows. Since we want the missed detection rate to be close to zero, $\alpha(T_\xi) \approx 0$, we set τ_ξ as the minimum of $\{J_\xi(I)\}_{I \in \text{train}}$, or rather, some percentage of it to allow for generalization outside the training data [19, 22]. When the test is learned, we can calculate its cost t and the power β from the training data. For better generalization, we use different training data to learn the tests than the data used to train the original set of models $\mathcal{M}$.

Our learning objective for each level ℓ is to optimize the cost-to-power ratio:

$$\mathcal{H}^\ell_* = \arg\min_{\mathcal{H}^\ell} \sum_{\xi \in V^\ell} \frac{t(T_\xi)}{\beta(T_\xi)}, \tag{5}$$

where each T_ξ also satisfies the condition in (4). Finding the global maximum of (5) is obviously intractable, thus our solution will be to generate good guesses and use stochastic optimization to improve the result. In particular, our learning approach uses the following steps: 1.) **parameter clustering**: cluster the parameters θ^g and the vocabulary of feature types $\mathcal{V}$ from the models at the level below, 2.) **model clustering**: cluster the models from the level below with respect to the appearance and geometry clusters, 3.) **local optimization**: optimize $\mathcal{H}^\ell$ on the training data. Steps 1 to 3 are repeated several times (by varying the clustering parameters) and the $\mathcal{H}^\ell$ with the lowest cost in (5) is selected.

Parameter clustering. To cluster the models, geometry will be the primary cue: models with similar spatial arrangements of parts will have a higher chance of being merged. We start by clustering the geometry parameters and group the feature types afterwards. *Geometry*: We use the k-means algorithm to cluster the means $\mu^{\ell+1}$ of the Gaussian parameters $\theta^{g,\ell+1}$ (defined in Sec. 4.1) and assign each mean $\mu^{\ell+1}$ to its closest cluster. *Appearance*: With respect to the centers, we can calculate the similarity between different types of features from vocabulary $\mathcal{V}^{\ell+1}$ and use a clustering algorithm to group them, which yields the vocabulary $\mathcal{V}^\ell$. Clustering in the case of discrete feature types means OR-ing (disjunction). To calculate the similarity we can do the following: for every pair of models in which a part of one model coincides with a part of the other (the means of the geometric distributions fall in the same k-means cluster) we simply increase the similarity score of the feature types corresponding to these two parts. With respect to $\mathcal{V}^\ell$ we can re-define the appearance parameters $\theta^{app,\ell+1}$.

Model clustering. If we do not consider the variances of the geometric distributions, we can now group the models $M^{\ell+1}$ which have the same geometric means (k-means clusters) as well as the appearance distributions $\theta^{app,\ell+1}$ into one model M^ℓ at level ℓ and make corresponding edges in the graph $\mathcal{H}^\ell$ (to models that were merged into M^ℓ). For each M^ℓ we can estimate the variances of the geometric distributions from the variances of its children at level $\ell + 1$.

Local optimization. Since the model clustering stage disregards the information on the variances of the geometry, the resulting models can have too large variances and the corresponding tests will not be very selective. This means that

the condition in (4) will not be fulfilled. It is thus necessary to optimize the clusters: we can remove one edge from $\mathcal{H}^\ell$, re-estimate the variances of the parent model M_ξ^ℓ and re-calculate the cost-to-power ratio $t(T_\xi^\ell)/\beta(T_\xi^\ell)$. At each step the edge with the highest cost-to-power ratio is removed from $\mathcal{H}^\ell$.

The levels of TCT are built until there is still a computational saving with respect to the cost-to-power ratio (5).

7 Combining TCT with the hierarchy-of-parts model [4]

To represent, learn and detect object classes we will use the hierarchy-of-parts model [4], which will serve as the *baseline* in the experiments. We give a brief

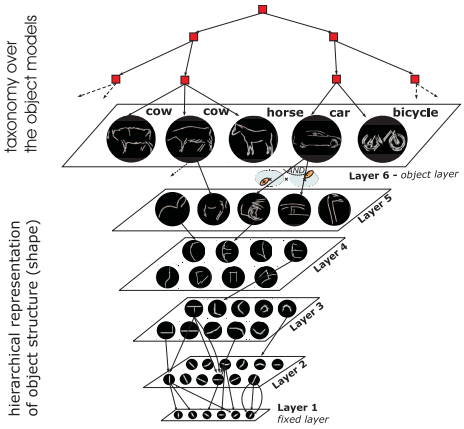

Fig. 2: Hierarchy-of-parts model [4] (hierarchy of shape features of increasing complexity) combined with the proposed taxonomic constellation tree TCT (taxonomy of object class models). The structure hierarchy has six layers. The taxonomy is built over the last layer's models which are the compositions (constellations) modeling the whole shape of the objects. The depth of the taxonomy is not predetermined.

summary of the model. Objects are represented with a recursive compositional shape vocabulary, the structure of which is learned from images without supervision. The vocabulary contains a set of shape *compositions* at each layer. Each composition is defined recursively: it is a hierarchical generative probabilistic model that represents a geometric configuration of a small number of parts which are themselves hierarchical probabilistic models, i.e., compositions from a previous layer of the vocabulary. Different compositions can share models for the parts, which makes the vocabulary efficient in size and results in faster inference.

The definition of a composition with respect to just one layer below is akin to that of the constellation model [10]. Each part is spatially constrained on the parent composition via a spatial relation which is modeled with a two-dimensional Gaussian distribution. Each part in a composition has also an "appearance" which is defined as a (discrete) distribution over the set of compositions from the previous layer of the vocabulary. At the lowest layer, the vocabulary consists of a small number of short oriented contour fragments. The vocabulary at the top-most layer contains compositions that represent the shapes of the whole objects. Altogether six layers are learned.

We will combine our taxonomy model with the hierarchical vocabulary in the following way. The top layer in the vocabulary is an object layer and grows linearly with the number of modeled classes. Thus the models at the final layer form our database of models for which the taxonomy is built over. The features **F** which are extracted from an image prior to recognition will be the detections corresponding to the compositions in the fifth layer of the vocabulary (one layer below the object layer). The illustration of the combined model is given in Fig. 2.

8 Experimental results

The aim of the experiments is to compare the speed of the proposed taxonomy model vs the baseline as the number of classes increases, and show that its detection rate does not degrade significantly with respect to the baseline model.

8.1 ETHZ shape dataset: 5 object classes

We use the ETHZ shape dataset of 5 object classes: apple logo, bottle, giraffe, mug and swan. For training and testing we follow the same protocol as in [11]. For this and all experiments in the paper, the following was used to train the

Table 1: Average detection-rate (in %) at 0.4 FPPI for the ETHZ dataset [11].

	applelogo	bottle	giraffe	mug	swan	average
[11]	83.2(1.7)	83.2(7.5)	58.6(14.6)	83.6(8.6)	75.4(13.4)	76.8
[23]	95.0	89.3	75.4	90.3	94.1	88.8
baseline	88.2(3.4)	87.6(1.5)	83.5(1.1)	86.1(2.)	80(3.5)	85.1
TCT	87.3(2.6)	87.3(2.2)	83.1(1.3)	85.8(3.1)	79.4(5.4)	84.6

model. Bounding boxes (scaled to size approx. 200 pixels in diagonal) were used in training. Two image scales, spaced apart by $\sqrt{2}$, were used to train object models, while 4 to 6 scales were used in testing. For constructing the taxonomy the positive training images as well as a set of 100 natural images not containing the objects were used to construct TCT. The positive training images available in each dataset were split into 5 images used as positive examples for test learning and the rest for learning the object representation, i.e., training the hierarchy-of-parts model [4].

For the five classes, the learned TCT consisted of 2 layers. The detection times are reported in Fig. 3 (the first five classes in the Shape-15 plot). The times reported do not include the process of feature extraction. Since both the baseline and our approach involve the same feature extraction process, such a comparison gives a clearer picture of the achieved speed-up.

The detection performance is reported in Table 1. The TCT approach performs slightly worse than the baseline which is due to the depth-first search strategy during recognition: while it boosts recognitions times, the performance might slightly degrade.

8.2 TUD shape dataset: 10 object classes

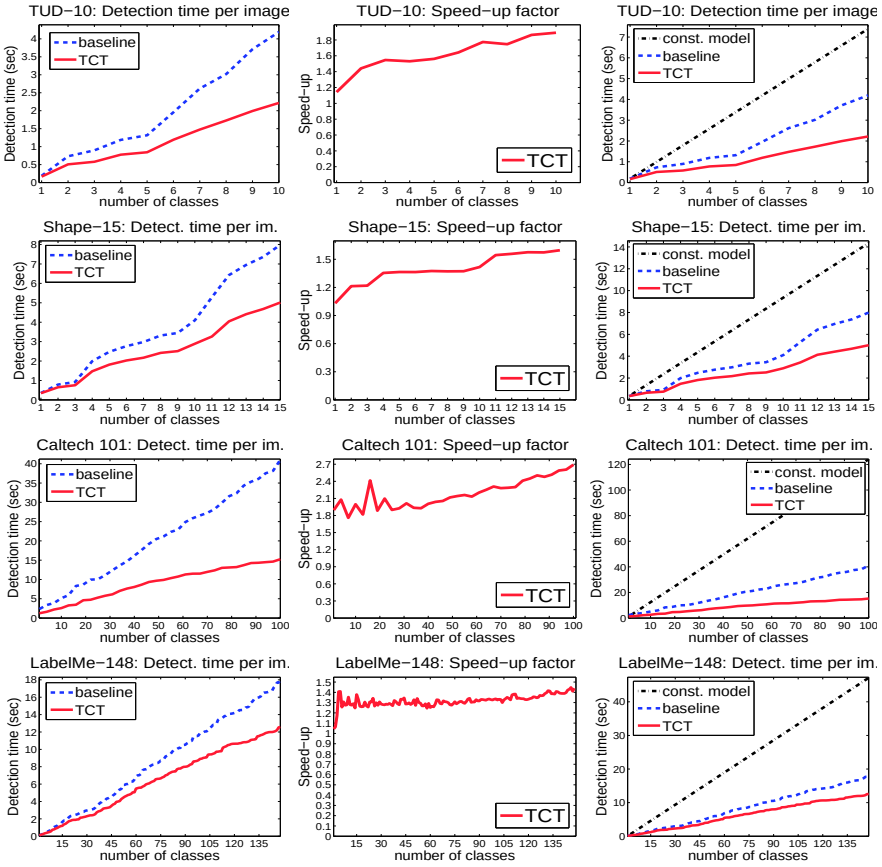

Fig. 3: Comparison between baseline and proposed TCT model for several datasets in: **a)** average detection times per image, **b)** speed-up of TCT over baseline, **c)** comparison of detection times of baseline and TCT against the recognition procedure of the constellation model [16] (see text for details), all as a function of the number of classes.

TUD *shape*2 [12] contains 10 classes of home appliances such as knife, fork, mug, saucepan, etc. The test set contains 10 images of objects for each class, and roughly 100 for training – from these we randomly choose 20 images for training the object class models with [4] and another 5 for training the TCT (with the addition of 100 negative natural images). The learned TCT consisted of 2 layers.

The recognition times of both the baseline and TCT are plotted in Fig. 3. The speed-up factor for 10 classes is almost 2 and grows as more classes are added. We also compare the baseline (hierarchy-of-parts [4]) and TCT against the recognition procedure used in the original constellation model [16, 10]. There,

a detector for each class is run separately over the image, thus resulting in complexity linear in the number of classes, i.e., $n \cdot O(\textit{detector for one class})$ (where n is the number of classes). Due to part sharing and the indexing and matching recognition scheme in [4], not all class models get evaluated in each image location already for our baseline, however, the running time of [4] is still linear, but with a smaller constant, i.e., $k \cdot O(\textit{detector for one class})$ (where $k << n$).

The classification accuracy of the baseline approach is 69%, while the TCT achieves a 66% classification rate. For comparison, Stark et al. [12] report a 44% accuracy using a discriminative approach (SVM over different types of features).

8.3 Shape 15

The approach was also tested on detection of 15 shape-based object classes. The first 5 are taken from ETHZ [11] and the 10 additional from the GRAZ dataset [7]. The comparison in detection performance is given in Fig. 4, while the comparison in detection times is plotted in Fig. 3. TCT is depicted in Fig. 6.

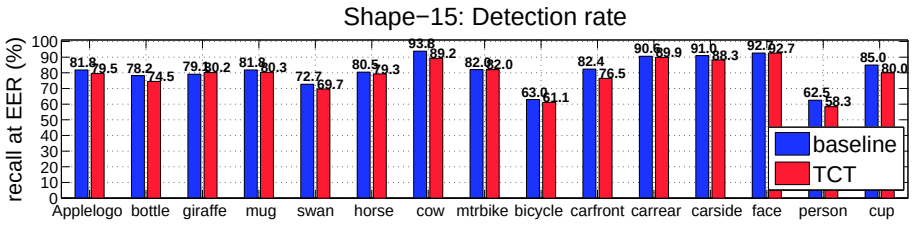

Fig. 4: Comparing detection rates (recall at EER) on Shape-15 – composed of the ETHZ [11] (first 5 classes in the plot) and 10 classes from GRAZ dataset [7].

Fig. 5: Detection examples: *horse*, *motorbike*, *giraffe*, *cow*, *bottle*.

8.4 Caltech 101 and LabelMe Dataset (148 object classes)

The approach has been also tested on a larger scale. We used 100 classes from the Caltech-101 dataset [13] and 148 classes from the LabelMe dataset [24]. For both datasets, 30 images were randomly chosen from the available annotations to train each class. For both datasets the taxonomy resulted in a 4 layer TCT. The approach was tested for detection time on 200 images randomly sampled from each dataset. Evaluation is done on full images, i.e., of approx. size 500×400. The comparison in detection times (averaged over 200 images) is plotted in Fig. 3.

The speed-up on the LabelMe dataset is not very high due to various possible reasons: 1.) too few classes are still being used to adequately showcase a taxonomy (as also reported in [8]), 2.) the LabelMe annotations are noisy and thus the baseline does not learn the classes very well, 3.) the images used for testing in LabelMe are images of complex scenes containing many objects and significant texture (with which already the baseline [4] does not deal well) resulting in many false-positives decisions, 4.) a number of classes have very simple shape (e.g. a *street lamp* which mostly gets represented as a long vertical line) and thus lead to a higher number of false positives per image already with the baseline method. A detection (even if it is a false positive) means that a whole branch in the TCT tree needs to be explored as well, making the speed-up smaller.

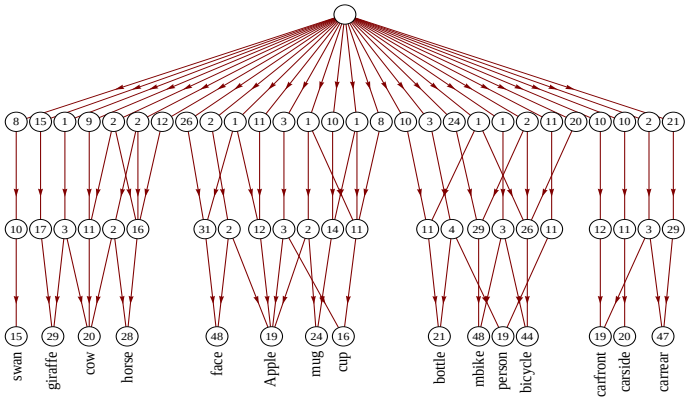

Fig. 6: TCT learned on Shape-15. Bottom level are object classes – the number denotes the number of models for each class.

9 Summary and conclusions

In this paper we proposed a novel approach for automatic construction of a generative visual class taxonomy with the aim to speed-up multi-class object detection with the constellation models. Specifically, we learned a taxonomic tree of constellations cascaded from coarse-to-fine resolution, and the corresponding detectors which take the form of inexpensive, binary tests. Both the taxonomic constellation tree and the corresponding tests are *learned* from the training data.

The approach has been utilized on the hierarchy-of-parts model [4] achieving efficiency in both, the representation of the object structure as well as in the number of modeled object classes. We demonstrated good speed-up on several recognition datasets with promising scaling tendency for recognition on a larger scale. As part of future work, we plan to add discriminative information to the taxonomy to boost also its recognition accuracy.

References

1. Marszałek, M., Schmid, C.: Constructing category hierarchies for visual recognition. In: ECCV. Volume IV of LNCS., Springer (2008) 479–491
2. Sivic, J., Russell, B.C., Zisserman, A., Freeman, W.T., Efros, A.A.: Unsupervised discovery of visual object class hierarchies. In: CVPR. (2008)
3. Blanchard, G., Geman, D.: Hierarchical testing designs for pattern recognition. Annals of Statistics **33** (2005) 1155–1202
4. Fidler, S., Leonardis, A.: Towards scalable representations of visual categories: Learning a hierarchy of parts. In: CVPR. (2007)
5. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE PAMI **29** (2007) 854–869
6. Huttenlocher, D., Felzenszwalb, P.: Pictorial structures for object recognition. IJCV **61** (2005) 55–79
7. Opelt, A., Pinz, A., Zisserman, A.: Learning an alphabet of shape and appearance for multi-class object detection. IJCV **80** (2008) 16–44
8. Stefan, A., Athitsos, V., Yuan, Q., Sclaroff, S.: Reducing jointboost-based multiclass classification to proximity search. In: CVPR. (2009)
9. Bart, I., Porteous, E., Perona, P., Wellings, M.: Unsupervised learning of visual taxonomies. In: CVPR. (2008)
10. Fergus, R., Perona, P., Zisserman, A.: Weakly supervised scale-invariant learning of models for visual recognition. IJCV **71** (2007) 273–303
11. Ferrari, V., Fevrier, L., Jurie, F., Schmid, C.: Accurate object detection with deformable shape models learnt from images. In: CVPR. (2007)
12. Stark, M., Schiele, B.: How good are local features for classes of geometric objects? In: ICCV. (2007)
13. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In: IEEE CVPR'04, Workshop on Generative-Model Based Vision. (2004)
14. Zehnder, P., Meier, E.K., Gool, L.V.: An efficient shared multi-class detection cascade. In: BMVC. (2008)
15. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR. Volume 2. (2006) 2161–2168
16. Weber, M., Welling, M., Perona, P.: Towards automatic discovery of object categories. In: CVPR. (2000) 2101–2108
17. Gavrila, D.M.: A bayesian, exemplar-based approach to hierarchical shape matching. IEEE PAMI **29** (2007) 1408–1421
18. Moreels, P., Perona, P.: A probabilistic cascade of detectors for individual object recognition. In: ECCV. Volume III. (2008) 426–439
19. Amit, Y., Geman, D., Fan, X.: A coarse-to-fine strategy for multiclass shape detection. IEEE PAMI **26** (2004) 1606–1621
20. Gangaputra, S., Geman, D.: A design principle for coarse-to-fine classification. In: CVPR. (2006) 1877–1884
21. Zhu, L., Lin, C., Huang, H., Chen, Y., Yuille, A.: Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In: ECCV. Volume 2. (2008) 759–773
22. Fleuret, F., Geman, D.: Coarse-to-fine face detection. IJCV **41** (2001) 85–107
23. Ommer, B., Malik, J.: Multi-scale object detection by clustering lines. In: ICCV. (2009)
24. Russell, B., Torralba, A., Murphy, K., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. IJCV **77** (2008) 157–173