Hierarchical Space Tiling for Scene Modeling

Shuo Wang[1,2], Yizhou Wang[1], and Song-Chun Zhu[2]

[1] Nat'l Engineering Lab for Video Technology Key Lab. of Machine Perception, Department of EECS, Peking University, Beijing, China
{shuowang, Yizhou.Wang}@pku.edu.cn
[2] Center for Vision, Cognition, Learning and Arts (VCLA), Department of Statistics, University of California, Los Angeles, U.S.
sczhu@ucla.edu

Abstract. A typical scene category, *e.g.,* street and beach, contains an enormous number (*e.g.,* in the order of 10^4 to 10^5) of distinct scene configurations that are composed of objects and regions of varying shapes in different layouts. A well-known representation that can effectively address such complexity is the family of compositional models; however, learning the structures of the hierarchical compositional models remains a challenging task in vision. The objective of this paper is to present an efficient method for learning such models from a set of scene configurations. We start with an over-complete representation called *Hierarchical Space Tiling (HST)*, which quantizes the huge and continuous scene configuration space in an And-Or tree (AOT). This hierarchical AOT can generate a combinatorial number of configurations (in the order of 10^{31}) through a small dictionary of elements. Then we estimate the HST/AOT model through a learning-by-parsing strategy, which iteratively updates the HST/AOT parameters while constructing the optimal parse trees for each training configuration. Finally we prune out the branches with zero or low probability to obtain a much smaller HST/AOT. The HST quantization allows us to transfer the challenging *structure-learning* problem to a tractable *parameter-learning* problem. We evaluate the representation in three aspects. (i) Coding efficiency. We show the learned representation can approximate valid configurations with less errors using smaller number of primitives than other popular representations. (ii) Semantic power of learning. The learned representation is less ambiguous in parsing configuration and has semantically meaningful inner concepts. It captures both the diversity and the frequency (prior) of the scene configurations. (iii) Scene classification. The model is not only fully generative but also yields discriminative scene classification performance which outperforms the state-of-the-art methods.

1 Introduction

Motivation A typical outdoor scene category, *e.g.,* street, beach, and country road, contains an enormous number of distinct scene configurations, *e.g.,* in the order of 10^4 to 10^5 (*i.e.,* $O(10^4) \sim O(10^5)$, let $O()$ denote the order) depending on the image resolution, which are composed of objects (*e.g.,* buildings,

vehicles) and regions (*e.g.,* sky, water) of varying shapes in different layouts. A well-known, or in a stronger word the only-known, representation that can explicitly address such complexity effectively is the family of hierarchical compositional models, which are reconfigurable and can generate combinatorial number of configurations through a small dictionary of shape elements. However, learning the structures of such models remains a challenge in vision, either for scene or object categories, and recently has become a hot topic in two communities: learning stochastic image grammar [13] and deep learning [3]. Two main factors contribute to the difficulties of structure-learning: (i) The space of the internal nodes is huge or essentially continuous (considering geometric deformations); (ii) The representations are often ambiguous (*i.e.,* not identifiable) and thus the learned model partially loses its power in parsing or classification as it diffuses probability (*i.e.,* prior for structures) over multiple possible interpretations.

The objective of this paper is to present an effective method for structure learning. We take outdoor scenes as an example and as a test bed for performance evaluations, and argue that the proposed method can be used in learning object categories as well.

We define *a scene configuration* C as a label map with objects and regions, and define *a scene category* as an unknown set Ω^* of all valid configurations. Given a set of training examples $D = \{C_m : m = 1...M\} \subset \Omega^*$, we learn a hierarchical and reconfigurable model which can equivalently be formulated as a stochastic grammar G [13]. G is fully generative and its language is the set of all valid configuration:

$$\Omega(G) = \{C : C = g(pt; \Delta)\} \tag{1}$$

Here pt is the parse tree for C, Δ is the dictionary or vocabulary of the model, and $g()$ is the generation function. $\Omega(G)$ should cover all variations in D with high probability and thus be diverse, and also generalize well to the underlying set Ω^* to achieve good performance in classification or parsing on the test dataset.

Related work Most existing work on scene category have been posed as a classification problem whose objective is to fuel the SVM training. (i) *Bag-of-Words (BoW) representation* [2] treats a scene as a collection of visual words and ignores the spatial layout information. (ii) *Grid structure representations*, such as, gist-based representation [6], spatial pyramid matching (SPM) [4], and a "reconfigurable" model [7]. These models implicitly adopt squares as elements in different sizes and locations and divide the image into grids. (iii) *Region based representations* [9] which segment an image into semantic regions rather than a grid, then model the contextual relations between the (adjacent) regions. (iv) *Non-parametric representation*, such as label transfer [5] that remembers all the observed configurations and interprets new configurations through nearest neighbor search and then deforming its label map through SIFT points. All these representations miss the hierarchical reconfigurable structures. The most related work is the Tangram model [12] which introduces the scene hierarchy by a predefined dictionary and infers an optimal configuration for each scene category. We extend [12] by proposing an effective method to learn the tiling dictionary for a compact and less ambiguous scene representation and addressing many issues about structure-learning: such as coding efficiency, ambiguity *etc.*

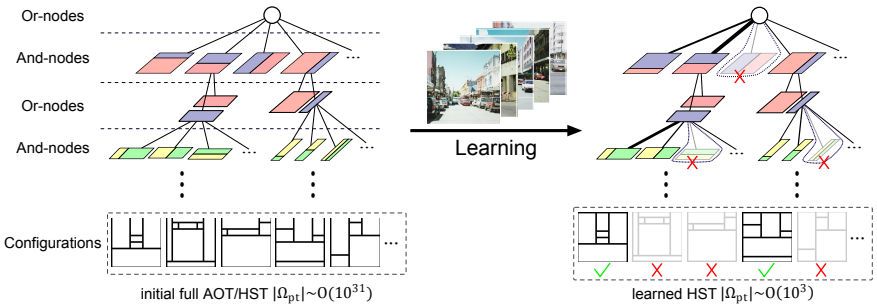


Fig. 1. HST transfers a structure-learning problem to a parameter-learning problem by pruning the unlikely branches from an excessive HST/AOT quantization (left) with $O(10^{31})$ parse trees to a much smaller HST/AOT (right) with $O(10^3)$ parse trees.

Overview Our method is inspired by the practice of ceramic tile industry where people manufacture tiles of a few shapes (triangles, squares, rectangles) and in a few sizes (from 2×2 inches to 20×20 inches), and compose any patterns to customer's needs in an economic budget. We start with an over-complete representation called *Hierarchical Space Tiling (HST)*, which quantizes the huge and continuous scene configuration space in an And-Or tree (AOT). The HST/AOT is shown in Fig. 1. An And-node represents a way of decomposing a region; an Or-node represents alternative decompositions with branching probabilities; and a terminal node is an element, such as squares and rectangles. This hierarchical HST/AOT can generate a combinatorial number of configurations in just a few layers (bottom left panel in Fig.1). Then we estimate the HST/AOT model through a learning-by-parsing (EM-like) strategy, which iteratively updates the HST/AOT parameters (branching probabilities at Or-nodes) while constructing the optimal parse trees for each training configuration. The learning process maximizes the likelihood subject to a model complexity. Finally we prune out the branches with zero or low probability to obtain a much smaller HST/AOT, shown in Fig.1(right). This final HST/AOT is a more compact model G whose language $\Omega(G)$ has a much smaller number of valid configurations, as most unlikely configurations are removed (bottom right panel in Fig 1). The probability of the model is focused on $\Omega(G)$ and is used as the prior for parsing.

In summary, the main contribution of our method is that it transfers the challenging *structure-learning* problem to a tractable *parameter-learning* problem, and it achieves this by hierarchically tiling the configuration space.

Evaluation We evaluate the representation in three aspects. (i) Coding efficiency: given any configuration $C \in \Omega^*$, we generate a configuration $\hat{C} \in \Omega(G)$ by a parse tree pt so that $\hat{C}$ approximates C with less than ϵ error and pt is small. More specifically, we compute the rate-distortion curves in coding theory, and show that our representation is clearly more effective than other popular representations (*e.g.,* the Quadtree and spatial pyramids). (ii) Semantic power of learning. The learned representation is less ambiguity in parsing configuration and has semantically meaningful inner concepts discovered. (iii) Scene classification. We apply the HST model to provide descriptions and show discriminative scene classification performance outperforms the state-of-the-art methods.

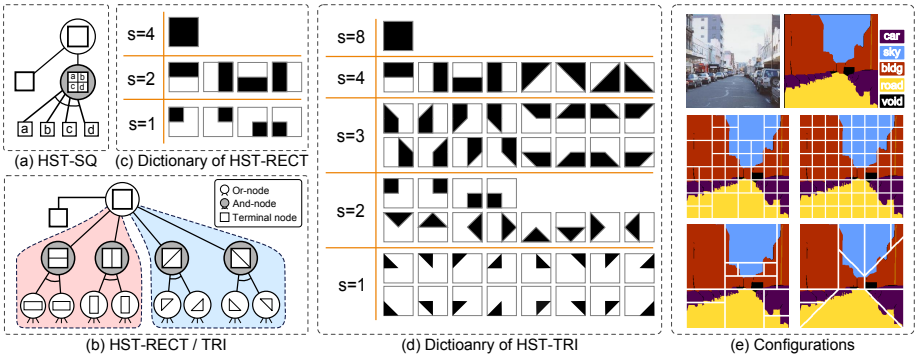


Fig. 2. HST examples. (a) The And-Or structure of HST-SQ. (b) The And-Or structure of HST-RECT (red) and HST-TRI (blue). (c)&(d) The tiling dictionary for HST-RECT/TRI on 2×2 grid (empty levels, *e.g.,* s=3 in (c), are not shown). (e) The "street" scene configurations generated from HST variants (Qt, SP, HST-RECT, HST-TRI respectively).

2 Hierarchical Space Tiling (HST) Representation

2.1 Definition of the HST

We define HST on an And-Or tree (AOT). There are three types of nodes in the HST/AOT: *And-nodes* (V^{AND}), *Or-nodes* (V^{OR}) and *terminal nodes* (V^{T}).

The terminal nodes V^{T}, shown as the square nodes in Fig.2(a)&(b), form a hierarchical tiling dictionary of the scene parts. They directly generate image regions. At the bottom of the hierarchy ($s = 1$), an image lattice (*i.e.*, scene label map) is divided into a $n \times n$ grid, and each cell is seen as an atomic shape element of the dictionary. A number of the atomic elements compose the higher-level terminal nodes at different scales, locations and shapes (to avoid the combination explosion, only regular shapes *e.g.*, squares, rectangles, triangles, trapezoid, are allowed) as shown in Fig.2(c)&(d). The "level" here means the number of atomic shape elements being used.

The And-nodes V^{AND}, shown as the solid circles in Fig.2(a)&(b), correspond to grammar rules like $r^{AND} : A \rightarrow a \cdot b$, which represent a fixed decomposition from a node A into lower-level parts a and b with branching probability $p(a, b|A) = 1$.

The Or-nodes V^{OR}, shown as the hollow circles in Fig.2(a)&(b), correspond to grammar rules like $r^{OR} : A \rightarrow a|b$, which act as "switches" between possible compositions. The branching probabilities $p(a|A), p(b|A)$ indicate the preference for each composition and can be learned from scene label maps (in Section.3).

The HST/AOT is naturally recursive, starting from a root which is an Or-node, generating the alternating levels of Or-nodes and And-nodes, and stopping at the terminal nodes. The And-Or structure defines a full space of possible parsing with probabilistic context free grammar (PCFG). By selecting the branches at Or-nodes, a parse tree pt can be derived from this grammar. Intuitively, when a parse tree collapses, it produces a planar configuration (Fig.2(e)), which is a subset of terminal nodes, and we utilize this configuration to represent the structure/configuration of a scene.

Table 1. Summarization of HST variants

		Rules	Terminal shapes	Δ	Or-nodes	$\lvert\Omega_{pt}\rvert$		
						2×2	4×4	8×8
HST-SQ	Qt	quartering	square	×	✓	2	16	65,536
	SP			×	×	1		
HST-RECT		horizontal-cut vertical-cut	square rectangle	✓	✓	9	2.59×10^6	4.48×10^{31}
HST-TRI		horizontal-cut vertical-cut oblique-cut	square rectangle triangle trapezoid	✓	✓	2,891	2.81×10^{17}	6.30×10^{80}

2.2 Three variants of HST

With different atomic elements and compositional rules, the derived HST representations are also different. In this section, we explore three variants of HST: square tiling, rectangular tiling and triangular tiling.

Square Tiling (HST-SQ) As shown in Fig.2(a), Quadtree (Qt)[1] is a type of HST, in which an "And-node" always decomposes its region into quadrants of equal size and an "Or-node" always switches either a terminal node (no split) or a 4-way split. This tiling method recursively divides a scene map until reaching a threshold of homogeneity or size (middle left in Fig.2(e)). The spatial pyramid (SP) model[4] generates an S-level representation for a scene, of which each level divides the scene map into $2^s \times 2^s$ grid ($s = 0...S-1$). The middle right in Fig.2(e) shows one layer of SP. Since both the Qt and SP model has only one splitting rule – the quartering rule, no dictionary is considered.

Rectangular Tiling (HST-RECT) Since the quartering rule is rigid, we extend the HST-SQ with more flexible rules: horizontal-cut and vertical-cut (red part in Fig.2(b)), to adapt the variance of scene configurations. Using the square as atomic elements and only allowing regular shape elements (*i.e.,* rectangles and squares), a hierarchical tiling dictionary is shown in Fig.2(c). The configuration inferred from this variant is displayed in the bottom left of Fig.2(e).

Triangular Tiling (HST-TRI) Because of the object shape variations, viewpoints and perspective, scenes do not only have block structures, *e.g.*, the "street" scene. Therefore, we adopt triangles as the atomic shapes and further relax the compositional shapes to four types: triangle, rectangle, square and trapezoid (shown in Fig.2(d)). This improvement adds more shape elements to the tiling dictionary. The HST-TRI configuration is shown in the bottom right of Fig.2(e).

We summarize the HSTs in Table.1 and evaluate their complexity by the number of possible parse trees can be generated ($\lvert\Omega_{pt}\rvert$). As shown in Table.1, the space expands exponentially as the granularity of atomic shape elements decreases. Compared to HST-SQ, HST-RECT and HST-TRI have much larger space volume and the potential to account for the variety of scene configurations. However, the richer representation increases the model complexity (size of dictionary). To trade off the model expressive power against the model complexity,

we finally adopt the HST-RECT for scene representation in this paper. The detailed comparison and analysis are shown in Section. 4.1.

2.3 Formulation

We define the HST as a 5-tupe

$$HST = (S, V^N, V^T; \Theta, \Delta) \tag{2}$$

where S is a start symbol at root denoting a scene category. $V^N = V^{AND} \cup V^{OR}$ is a set of non-terminal nodes including the And-nodes V^{AND} and the Or-nodes V^{OR}. V^T is a set of terminal nodes. We use v to index the nodes, and $v_i \in Ch(v); i = 1...N^{(v)}$ is the i-th child node of v, where $Ch(v)$ is the child node set and $N^{(v)}$ is the number of v's children. We assign the first choice of an Or-node as a terminal node, *i.e.,* $v_1 \in V^T$, if $v \in V^{OR}$. That is, each non-terminal node is grounded to a terminal shape (see Fig.2(a)&(b)) which is the learned inner concept of this node. The parameters Θ are the branching probabilities of each branch at the Or-nodes $\Theta = \{\theta_{vi}; v = 1...|V^{OR}|, i = 1...N^{(v)}\}$ where $|V^{OR}|$ is the total number of Or-nodes. The tiling dictionary of the scenes is denoted by $\Delta = V^T$ which can be iteratively learned in Section.3.

By selecting the branches at Or-nodes, a parse tree pt can be derived from HST. Given a configuration C (*i.e.,* the scene label map), one can infer the parse tree pt from a posterior distribution of the Gibbs form,

$$p(pt|C; \Theta, \Delta) = \frac{1}{Z} \exp\{-E(pt|C; \Theta, \Delta)\} \tag{3}$$

where $Z = \sum_{pt} \exp\{-E(pt|C; \Theta, \Delta)\}$ is a partition function summing over the full parsing space of HST and can be calculated analytically in this case.

In the formulation, the compatibility or contextual relations among the And-nodes is not considered and thus the HST is a probabilistic context free grammar (PCFG). The energy of a parse tree is therefore defined on two potential terms corresponding to the Or-nodes and terminal nodes of a parse tree:

$$E(pt|C; \Theta, \Delta) = \sum_{v \in V^{OR}_{pt}, v_i \in Ch(v)} E^{OR}(v_i|v) + \lambda \sum_{v \in V^T_{pt}} E^T(C_v|v) \tag{4}$$

where λ is the parameter balancing the two terms (λ=0.25 in this paper). C_v denotes the patch of a label map covered by the terminal node v.

The energy of a child of an Or-node is defined on its branching probability:

$$E^{OR}(v_i|v) = -\ln \theta_{vi} = -\ln \frac{\#(v \to v_i)}{\sum_{i=1}^{N^{(v)}} \#(v \to v_i)} \tag{5}$$

where $\#(v \to v_i)$ is the number of times v selects i^{th} branch v_i. Intuitively, this term favors the compositional ways that often make up a larger part. We learn the branching probability for each scene category in Section.3.

The energy for terminal nodes is defined as

$$E^T(C_v|v) = -\ln \frac{1}{|C_v|} \sum_{i \in C_v} \mathbb{1}[l_i = l_v] \tag{6}$$

where $\mathbb{1}[\cdot]$ is the indicator function, l_i is the semantic label of pixel i, l_v is the dominant label of the terminal node v. This term measures the homogeneity of the terminal nodes in terms of semantic labels.

3 Learning of HST

Given a set of scene configurations $D = \{C_m : m = 1...M\}$, we learn the HST representation. To avoid the false compositions, *e.g.,* sky and ocean may be grouped wrongly into one region due to their similar appearance, we use the scene label maps as the ground-truth scene configurations as training examples. The learning requires us to estimate the optimal branching probabilities Θ at the Or-nodes and tiling dictionary Δ subject to constraints:

$$\begin{aligned}(\Theta, \Delta)^* &= \arg\max_{\Theta,\Delta} \log p(D; \Theta, \Delta) = \arg\max_{\Theta,\Delta} \sum_{m=1}^{M} \log p(C_m; \Theta, \Delta) \\ &= \arg\max_{\Theta,\Delta} \sum_{m=1}^{M} \log \sum_{pt_m} p(C_m, pt_m; \Theta, \Delta) \quad s.t. \sum_{i=1}^{N^{(v)}} \theta_{vi} = 1; v = 1...|V^{OR}|\end{aligned} \tag{7}$$

The objective function can be rewritten as:

$$L(\Theta, \Delta) = \sum_{m=1}^{M} \log \sum_{pt_m} p(C_m, pt_m; \Theta, \Delta) + \sum_{v=1}^{|V^{OR}|} \alpha_v (1 - \sum_{i=1}^{N^{(v)}} \theta_{vi}) \tag{8}$$

where α_v is the Lagrange multiplier for the branching probabilities at each Or-node to be normalized. To maximize $L(\Theta)$, we adopt a learning-by-parsing (EM-like) method including: E-Step, inference of pt, and M-Step, parameter estimation of Θ. Finally, we summarize the dictionary Δ by removing the rare elements.

3.1 The E-step: Parse tree inference

In the E-step, we keep the current branching probability Θ on the Or-nodes fixed and infer the parse trees for each given configuration (*i.e.,* scene map). Because the full parsing space is huge, *e.g.*, $|\Omega_{pt}^{RECT}| = 4.48 \times 10^{31}$ for an 8×8 grid, it is intractable to solve Eq.8 by summing over all possible parse trees. Thus Viterbi algorithm is introduced as an approximated method by using the optimal one instead of all parse trees [10]. Since the HST is tree-structured, and the objective energy function (Eq.8) is defined in a linear form, a Dynamic Programming (DP) algorithm can be employed to infer a globally optimal parse tree. However, when the training set is small *w.r.t.* the large HST space, although the Viterbi algorithm infers the optimal parse tree for each data sample, the number of parse trees may be insufficient in order to robustly estimate the HST parameters. Therefore, we extend the standard DP to infer the top-$K(K \geq 1)$ parse trees to enlarge the number of parse trees used in the learning. In the following, we first introduce the standard DP followed by the multi-case strategy.

We rewrite Eq.4 in a recursive expression by defining an energy function over the sub-tree starting with an Or-node v as the root node, and $\{v_i \in Ch(v), i = 1...N^{(v)}\}$ is child-node set of v:

$$E_v(pt(v_i)|C; \Theta, \Delta) = \begin{cases} E^T(C_v|v) & \text{for } i = 1 \\ \sum_{j=1}^{N^{(v_i)}} E_{v_i}(pt(v_{ij})|C; \Theta, \Delta) & \text{for } i > 1 \end{cases} \tag{9}$$

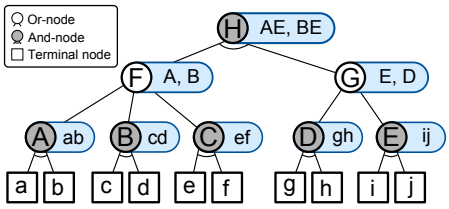


Fig. 3. Illustration of multi-case inference when $K = 2$. Multiple compositions kept for each non-terminal node are highlighted in the blue bar.

where $i = 1$ indexes the Or-node v terminates on itself; otherwise, a branch $v_i \in V^{AND}$ will be chosen. v_{ij} is a child node of v_i.

The DP proceeds by evaluating all possible branches for the Or-nodes v in the HST, and selects the best one such that $v_i^* = \arg\min_{v_i} E_v(pt(v_i)|C; \Theta, \Delta)$.

We propose a multi-case strategy to infer the top-K (50 in this paper) parse trees for each given configuration in an ascending order of the energy defined in Eq.9. It guarantees that the globally optimal solution is in the first place since the standard DP is a special case of the multi-case algorithm when $K = 1$. Fig.3 shows an example of $K = 2$ for illustration. Compared with the standard DP, here each non-terminal node v can keep $K = 2$ compositions (blue bar in Fig.3). During the bottom-up inference, (i) if reaching an Or-node, say "F", we sort all its child nodes in ascending order according to their energies (A, B, C : $A < B < C$) and keep the first $K = 2$ items $\{A, B\}$. (ii) If reaching an And-node, say "H", we first enumerate all possible combinations of its child nodes $\{AE, AD, BE, BD\}$, then select the first $K = 2$ items in the similar way. (iii) If reaching the root node, multiple parse trees can be obtained by tracing back all the kept compositions in a top-down manner.

3.2 The M-step: Branching probability update

In the M-step, we estimate the parameter Θ by Maximum Likelihood Estimation (MLE), which takes the derivative of $L(\Theta)$ (Eq.8) *w.r.t.* θ_{vi} and sets it to zero, then update the branching probability for each branch v_i according to

$$\theta_{vi}^{(t+1)} = \frac{1}{\alpha_v} \sum_{m=1}^{M} \sum_{pt_m} \mathbb{1}[v_i \in pt_m] \cdot p(pt_m|C_m; \Theta^{(t)}, \Delta^{(t)}) \tag{10}$$

where $\mathbb{1}[\cdot]$ is the indicator function that indicates whether the branch v_i is selected in the top-K parse tree pt_m. Θ is set to be uniform as initialization.

Finally, those branches whose probabilities are below a certain threshold (say 0.01) are pruned. We collect the terminal nodes from all the parse trees, and they compose the tiling dictionary Δ.

4 Experiments

In this section, we justify the proposed HST representation in three aspects. (i) We evaluate the coding efficiency of HST variants by the rate-distortion curve.

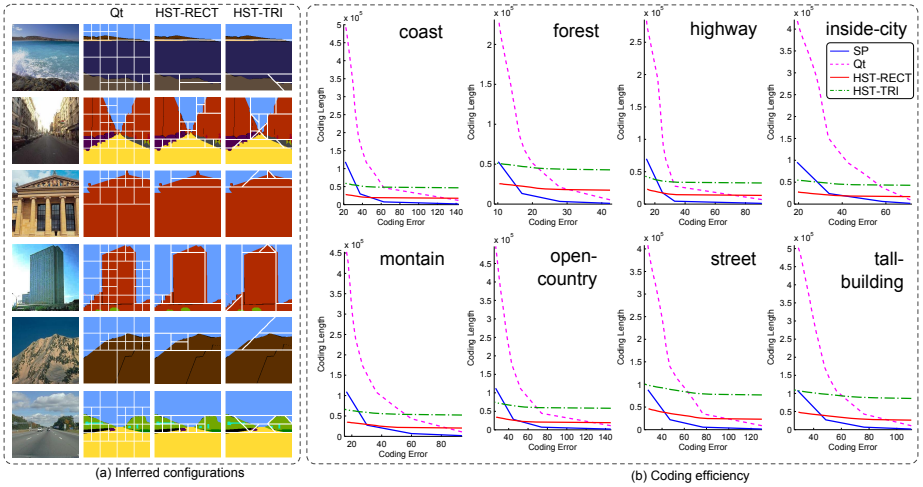


Fig. 4. Rate-distortion curve. (a) The inferred configurations by Qt, HST-RECT and HST-TRI. (b) The rate-distortion curve of HST variants where the horizontal axis denotes the coding error and the vertical axis denotes the coding length.

(ii) We show the changes of model compactness, ambiguity reduction and the learned meaningful dictionary through the iterative learning process. (iii) We apply the learned categorical scene spatial layout prior to scene classification and show the improved performance.

Dataset The dataset we use in this paper is a subset of the LabelMe dataset [8] as in the work on label transfer [5], which includes 2,688 images from 8 categories of outdoor scenes (*e.g.*, coast, highway) and their corresponding scene label maps with 33 semantic labels plus a "void" label, $\mathcal{L} = \{`sky', `road', ..., `void'\}$. The image size is 256×256. Note that some semantic regions of very small areas are merged into surrounding regions for simplification, *e.g.*, "moon" and "sun" are merged into "sky", "window" and "door" are merged into "building". After this pre-processing, those images annotated by single label (no configuration at all) or whose "void" area exceeds a threshold (*i.e.*, 30%) are discarded. Finally we get 1,968 valid "image-scene map" pairs with 27 semantic labels.

4.1 Efficiency of representing scene configurations

In the first experiment, we compare four HST variants: Quadtree(Qt), spatial pyramid(SP), rectangular tilling(HST-RECT) and triangular tilling(HST-TRI), by the rate-distortion curves which defined as the coding error *w.r.t.* coding length.

The *coding error* counts the per-pixel error ratio in the generated configurations (reconstructed scene maps), using inferred parse trees, which is defined as

$$CE = \sum_{m=1}^{M} \frac{1}{|C_m|} \sum_{i \in C_{mv}, v \in V^T_{ptm}} \mathbb{1}[l_i \neq l_v]. \tag{11}$$

The *coding length* is defined as the total bits storing in a reconstructed scene map binary file which varies with different HSTs. (i) For Qt, since it does not have

a dictionary, the coding length is calculated as $CL_{Qt} = \sum_{m=1}^{M} n_m * B_{sq}$, where n_m is the number of terminal nodes for the given configuration C_m (scene label map) and $B_{sq} = B_l + B_p + B_s$ is the coding bits for each square terminal node, where $B_l = \lceil \log_2(|\mathcal{L}|) \rceil$ is the coding bits for the semantic label, $B_p = \lceil \log_2(w_{C_m}) + \log_2(h_{C_m}) \rceil$ is the coding bits for the node position, w_C and h_C are the width and height of C_m. $B_s = \log_2 S$ is the coding bits for scale, where S is the maximum level allowed in Qt. (ii) For S-level SP, $CL_{SP} = \sum_{m=1}^{M} \sum_{s=0}^{S-1} 2^s * B_l$. (iii) For HST-RECT and HST-TRI, considering the hierarchical dictionary, the coding length is defined as

$$CL_{HST-RECT/TRI} = CL(\Delta) + \sum_{m=1}^{M} \sum_{v \in V_{pt_m}^T} (B_l + B_p - \log p(v)) \tag{12}$$

where $p(v)$ is the frequency of the terminal node v appearing in the parse trees which is learned in Section.3.2, and $CL(\Delta)$ is the coding length of learned dictionary defined as:

$$CL(\Delta) = \sum_{s=1}^{S} |\Delta_s| \cdot (s-1) \cdot 2 \log_2 s \tag{13}$$

where $|\Delta_s|, s = 1...S$ denotes the number of terminal nodes in the s-level of the dictionary. An s-level terminal node consists s atomic shape elements (shown in Fig.2(c)&(d)). $s-1$ means we shall code the rest of atomic elements *w.r.t.* the first one. $2 \log_2 s$ is the sum of the coding bits to localize where is an atomic element *w.r.t.* the first one in horizontal and vertical by assuming that a terminal node is a connected components of the atomic elements.

For the images of size 256×256, we divide them to an 8×8 grid at the bottom level. Fig.4(a) shows the inferred configurations by Qt, HST-RECT and HST-TRI. As shown in Fig.4(b), by controlling the number of selected terminal nodes we can measure the changes of coding error (horizontal axis) *w.r.t* the coding length (vertical axis) for each scene category. We have the following observations: (i) when the coding error is high, less terminal nodes are selected. The coding lengths of HST-RECT and HST-TRI are always above SP and Qt due to the coding bits for the hierarchical dictionaries. (ii) When the coding error decreases, the coding lengths of SP and Qt increase exponentially because of the exponential growth of the number of additional terminal nodes. However, due to the over-completeness of the dictionary and adaptive variable-length coding strategy (Shannon entropy coding), the coding lengths of HST-RECT and HST-TRI are much more stable and finally go below SP and Qt when the coding error is small. (iii) Compared with HST-RECT, although HST-TRI contains richer shape dictionary (it has triangle, trapezoid), it costs more coding bits consistently for every category. Therefore, we adopt HST-RECT to model the scenes, which can always get reasonable coding accuracy with compact coding length.

4.2 Ambiguity reduction, learned dictionary and configuration priors

For each category, we randomly select 100 images as a training set to learn the HST-RECT. We divide the scene maps into an 8×8 grid at the bottom level and empirically set $K = 50$ and $\lambda = 0.25$.

Table 2. The shrinkage of HST-RECT space for "street" scene

Round	$\|V^{AND}\|$	$\|V^{OR}\|$	$\|V^{T}\|$	$\|\Omega_{pt}^{RECT}\|$
0	6,048	1,296	1,296	4.48×10^{31}
1	570	519	366	8.01×10^{7}
2	351	386	256	2.23×10^{5}
3	238	302	184	1.14×10^{3}
4	221	290	173	9,140

In initialization, we set the branching probability of HST-REST follow an uniform distribution. During the iterative learning, the probabilities of the often selected branches are increasing while the rarely selected ones are decreasing. Thus the distribution over branches becomes concentrated. Moreover, the branches whose probabilities are lower than a certain threshold (say 0.01) are pruned. The learning leads to the following results:

(i) Taking "street" scene as an example, we count the total number of possible parse trees as the volume of HST space Ω_{pt}^{RECT}.The full parsing space of HST shrinks quickly as shown in Table.2 and converges after 4 rounds.

(ii) The dictionary size ($|\Delta| = |V^T|$, *i.e.*, the number of terminal nodes) can be reduced greatly as shown in Table.2. Fig.5 shows some learned terminal node groups which are often observed in different scenes. They are composed of several terminal nodes and form meaningful sub-configurations of semantic regions in the scenes.

(iii) The compositional ambiguity of representation is greatly reduced. The ambiguity arises from the shape elements shared by more than one parent node, which will admit two or more reasonable parse trees for one configuration. *e.g.*, Fig.6(a) shows four different parse trees for the same street scene configuration. Reduced ambiguity leads to distinct representation and lower model complexity. The ambiguity of parse trees are measured by their posterior probability distributions (Eq.3). Fig.6(b)&(c) shows the similar probabilities in initialization (Round=0) and their posterior probabilities become increasingly polarized through each round of learning.

(iv) The number of typical configurations for a scene category is small. For each scene category, we infer 5000 (50×100 images) parse trees by multi-case inference (Section.3.1), then calculate their posterior probabilities and form a posterior probability distribution as shown in Fig.7(a), where the horizontal axis denotes the index of parse trees and vertical axis denotes their posterior probabilities. The sharp decay curve demonstrates that the categorical configurations concentrate on only a few typical ones. The typical configurations and their corresponding images are shown in Fig.7(b)-(i).

4.3 Scene classification

In application, we apply the learned scene spatial layout priors from HST-RECT as templates to improve the supervised scene classification. The data is split in

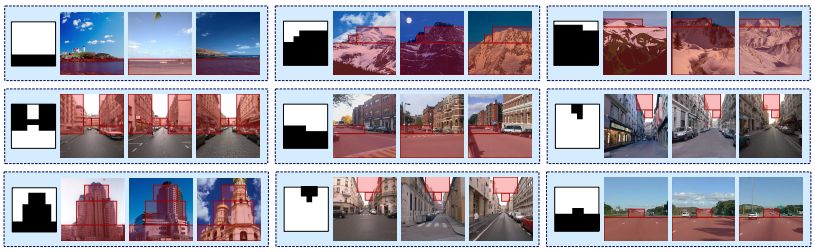
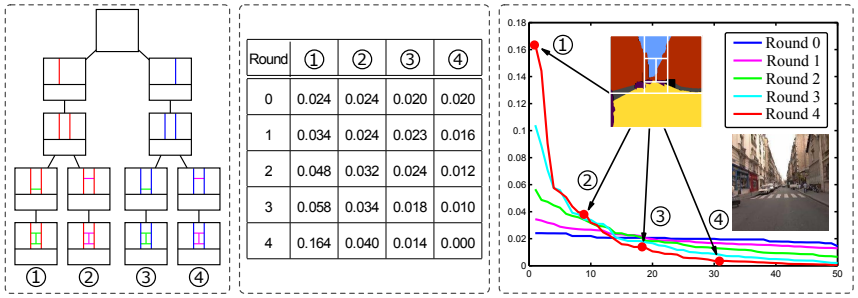


Fig. 5. Terminal node groups which are often observed in different scenes.



Round	①	②	③	④
0	0.024	0.024	0.020	0.020
1	0.034	0.024	0.023	0.016
2	0.048	0.032	0.024	0.012
3	0.058	0.034	0.018	0.010
4	0.164	0.040	0.014	0.000

(a) Ambiguous parse trees (b) Posterior probability (c) Posterior probability distribution of parse tree

Fig. 6. Ambiguity reduction through learning. (a) An illustration of ambiguous parse trees. (b) The posterior probability in each learning round of the ambiguous parse trees. (c) The posterior distribution of the top-K inferred parse trees.

the same way as introduced in Section. 4.2 – 100 images for training and the rest for testing in each category.

The categorical scene templates are selected according to the posterior probability. We set a threshold empirically (90%) and select the smallest parse tree set whose posterior summation is above the threshold, e.g., select the top-5 parse trees if the summation of their posterior probabilities covers more than 90% of the posterior probability distribution. We then remove the reduplicate configurations and merge the similar configurations generated by the selected parse trees, and finally get 2-5 typical configurations (Fig.7(b)-(i)) as templates for each scene category and use them for classification.

After that, the features are extracted as shown in the right of Fig.8. Firstly, the appearance descriptors such as "SIFT" and "color moment" are extracted from all training images. Secondly, the K-means clustering algorithm is applied to quantize the appearance descriptors into codewords with $W_{\text{txt}} = 200$ and $W_{\text{clr}} = 50$ for texture and color respectively. Thirdly, for each template, we align it to the training images by scaling, and then collect a histogram of detected appearance codewords inside each region of the template and concatenate them into a *template histogram vector*. The template histogram vectors of each template are then connected together into a long feature vector named *scene configuration appearance descriptor (SCAD)*. Finally, we train multi-class scene classifies using the SCADs as input data. The classifier is support vector machines (SVMs) using one-versus-all mode. In testing, we extract SCAD from an

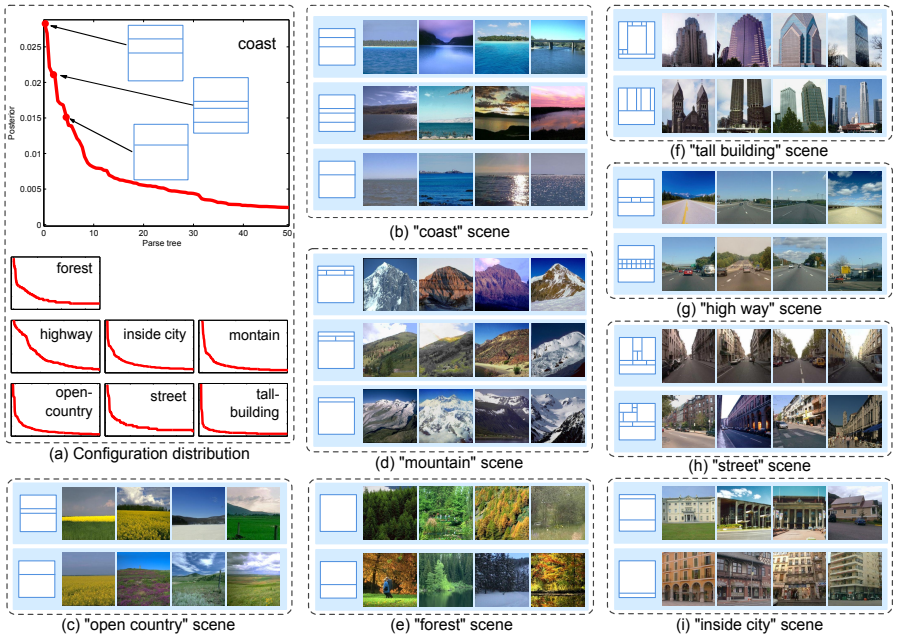


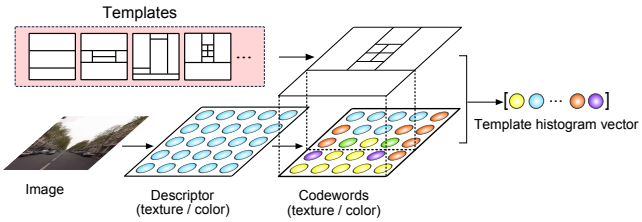
Fig. 7. Categorical typical configurations. (a) The posterior probability distribution for each scene category, where the horizontal axis is the index of parse tree and vertical axis is the posterior probability. (b)-(i) The categorical typical configurations for each scene category.

input image, and the scene category label is the one whose classifier has the highest response.

With the same data split, we compare our classification performance with other five methods: (i) a holistic "Gist" feature based method [6], (ii) a BoW based method [2], (iii) spatial pyramid matching (SPM) method [4], and (iv) an extension of SPM named locality-constrained linear coding (LLC) [11], (iv) the tangram model (Tangram) [12]. We ran their released source codes and reported the results in the right table of Fig.8. The average precision (AP) of the proposed model (HST-RECT) is 91.71% which outperforms the others. Compared with the previous methods, our proposed model allows more flexible configurations, and it also incorporates the scene spatial layout priors so as to improve the classification performance.

5 Discussion and future work

In this paper we propose an effective method to learn the structure of hierarchical and reconfigurable scene models by quantizing the space of configurations using Hierarchical Space Tiling (HST). We show that it can learn a parsimonious and less ambiguous representation, as well as meaningful tiling dictionary. We also show improved scene classification performance using the learned configurations as templates. For learning the meaningful categorical configurations



Methods	AP(%)
Gist [6]	72.15
BoW [2]	84.57
SPM [4]	84.92
LLC [11]	87.97
Tangram [12]	86.07
Ours	**91.71**

Fig. 8. The pipeline of feature extraction using typical configurations as templates (left) and the scene classification performance (right)

we focused only on the outdoor scenes which have strong geometric structures and roughly aligned. In the future, we can study the relationships between the learned structure (dictionary) and scene attributes to associate scene attributes to nodes in the AOT and thus local windows in images.

Acknowledgement The authors thank Jun Zhu and Tianfu Wu for discussions. This work is supported by: NSF-CNS-1028381, 973-2009CB320904, NSFC-61272027, NSFC-61231010, NSFC-91120004 and China Scholarship Council.

References

1. Berg, M., Cheong,O., Kreveld, M., Overmars, M.: *Computational Geometry: Algorithms and Applications (Third edition)*. Springer-Verlag (2008)
2. Fei-Fei, L., Perona, P.,: A Bayesian hierarchical model for learning natural scene categories. CVPR (2005) 524–531
3. Hinton, G. E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation (2006)
4. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. CVPR (2006) 2169–2178
5. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: label transfer via dense scene alignment. CVPR (2009)
6. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. IJCV (2001) 145–175
7. Parizi, S. N., Oberlin, J., Felzenszwalb, P.: Reconfigurable models for scene recognition. CVPR (2012)
8. Russell, B. C., Torralba, A., Murphy, K. P., Freeman, W. T.: LabelMe: a database and web-based tool for image annotation. IJCV (2008) 157–173
9. Socher, R., Lin, C., Ng, A., Manning, C.: Parsing natural scenes and natural language with recursive neural networks. ICML (2011)
10. Viterbi, A. J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory (1967)
11. Wang, J., Yang, J., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. CVPR (2010)
12. Zhu, J., Wu, T. F., Zhu, S. C., Yang, X. K., Zhang, W.J.: Learning reconfigurable scene representation by Tangram Model. Workshop on Application of Computer Vision (2012)
13. Zhu, S. C., Mumford, D.: A stochastic grammar of images. Foundations and Trends in Computer Graphics and Vision (2006) 259–362