

Scene Collaging: Analysis and Synthesis of Natural Images with Semantic Layers

Phillip Isola
MIT

phillipi@mit.edu

Ce Liu

Microsoft Research

celiu@microsoft.com

Abstract

To quickly synthesize complex scenes, digital artists often collage together visual elements from multiple sources: for example, mountains from New Zealand behind a Scottish castle with wisps of Saharan sand in front. In this paper, we propose to use a similar process in order to parse a scene. We model a scene as a collage of warped, layered objects sampled from labeled, reference images. Each object is related to the rest by a set of support constraints. Scene parsing is achieved through analysis-by-synthesis. Starting with a dataset of labeled exemplar scenes, we retrieve a dictionary of candidate object segments that match a query image. We then combine elements of this set into a “scene collage” that explains the query image. Beyond just assigning object labels to pixels, scene collaging produces a lot more information such as the number of each type of object in the scene, how they support one another, the ordinal depth of each object, and, to some degree, occluded content. We exploit this representation for several applications: image editing, random scene synthesis, and image-to-anaglyph.

1. Introduction

Parsing an image into a set of objects and interactions remains a grand challenge in computer vision. When humans look at a scene, e.g. a cityscape, a forest or a cafeteria, we see an organized interaction of objects, functions and spaces. However, many existing scene parsing approaches represent an image simply as a 2D array of pixel labels (e.g. [9], [18], [13], [22]), and these representations fail to account for occlusion. In typical images, huge swaths of scene structure are occluded from view. Further, occlusion even makes visible content difficult to parse: when projected into a 2D image, background objects are often fragmented by occluders.

To solve these problems, we propose a novel scene model, in which we represent discrete semantic objects on separate layers. We take our lead from human artists. While there are many ways by which an artist can synthesize a scene, one of the quickest and easiest is to collage together the image out of found pieces. Following this collaging approach, we model a scene as a collage of object segments

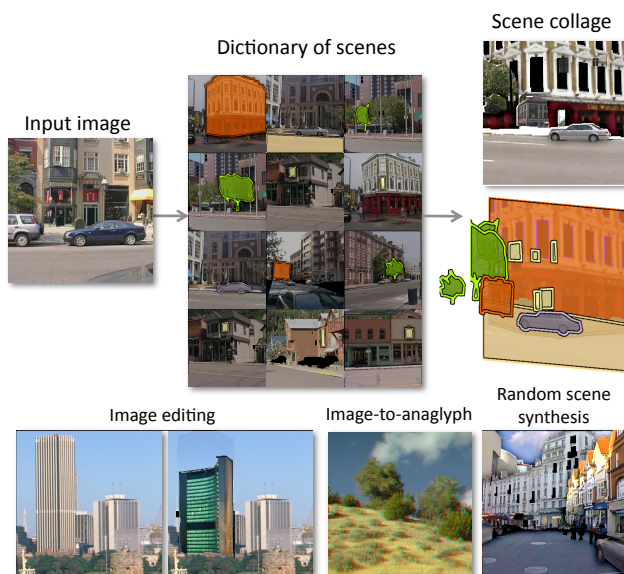



Figure 1: Top: We parse an input image (left) by recombining elements of a labeled dictionary of scenes (middle) to form a collage (right). Bottom: Our representation enables many applications. Representing discrete objects leads to easy scene editing (left; one building may be swapped for another). Representing layers gives us a rough estimate of depth (center; please view with anaglyph glasses ). And because our model is generative, we can synthesize random scenes (right).

sampled from a large database of example images for which humans have provided object label annotations (Figure 1; also see Figure 3). The statistics of features in the segments from the database are used to match regions in the query image. Given a query image, we infer a collage through analysis-by-synthesis, building up an explanation that both generates the query’s appearance and preserves structural properties of the examples from which the collage is pieced together. Many applications – including image editing, random scene synthesis, and rough image-to-anaglyph – become available once we have inferred a scene collage (Figure 1, bottom row).

1.1. Related work

Layer models have been widely used in both computer vision and graphics. In video segmentation, layers are often

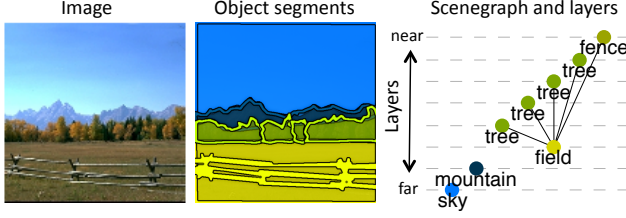


Figure 2: We model a scene as a layer-world collage of objects. Object support relationships are represented in a “scenograph”. For example, the scenograph above indicates that the trees are supported by the field.

used to group pixels that undergo the same motion and to handle occlusion [23], [10]. Unlike these past solutions, our present system operates on a single image rather than a video.

In computer graphics, several approaches synthesize images by compositing layers (e.g. [5], [1]). However, these applications rely on human input and their goal is image synthesis whereas our system aims at both scene analysis and scene synthesis.

In the realm of scene parsing, Guo and Hoiem have recently attempted to infer occluded scene content [7]. However, their system does not explicitly represent image layers, nor does it resolve depth ordering.

The LabelMe3D algorithm of Russell and Torralba infers the depth ordering of segments in a scene, but only with the aid of extensive human annotation at test time [16]. Our present aim is a system that is fully automatic at test time. Russell et al. [15] also developed an image segmentation algorithm that involves matching a query image to an “image composite”, that is, pieces of similar images stitched together. Our collage-based approach is similar in spirit, but our present goal extends beyond just segmentation.

Another related line of work is image parsing using 3D models [8], [26]. These methods differ from our own in that they represent objects with 3D models whereas we use a simpler representation: flat layers.

In addition to being layer-based, our approach is example-based and object-based. Recently, several example-based methods have been applied to scene parsing with promising results [13], [22]. However, our representation differs from these alternatives in that our examples are human labeled object segments, rather than full images [13] or super-pixels [22]. Other object-based methods have also been proposed: [25] and [21] used object detectors, and [14] showed promising qualitative results at scene parsing with object segments.

The primary contribution of our paper is the application of a novel layer-world representation to understanding and manipulating natural images. Our **scene collage** representation naturally consists of geometric relationships such as occlusion and depth ordering, and could be further extended

to a full 3D model. Furthermore, our polygon-based layer representation is user-friendly; namely, users can easily intervene with the layer world by translating and scaling the polygons or dragging each individual control point.

In addition, we introduce a nonparametric scene grammar for searching a space of reasonable scenes. In particular, we model each scene with a **scenograph** of object interrelations. These scenographs provide scene level context, which guides how we fit exemplar segments into a scene.

2. Scene model

We model a scene as a collage of transformed exemplar object segments. Each exemplar object is a labeled object segment from a dictionary, \mathcal{L} , of annotated images. These segments consist of the object’s class \tilde{c}_ℓ , a geometric mask, \tilde{Q}_ℓ , which specifies the object’s silhouette, image pixels \tilde{I}_ℓ , and an appearance model \tilde{g}_ℓ (throughout this paper we use $\tilde{\cdot}$ to mark variables that refer to information in our dictionary).

A **scene collage** consists of transformed versions of the dictionary segments. The indices into the dictionary for the segments used in a collage are denoted as $\ell \in \mathbf{L}$. With an object’s transformation parameters denoted as θ_ℓ , we transform exemplar segments masks as:

$$Q_\ell = T(\tilde{Q}_\ell; \theta_\ell). \quad (1)$$

For transformations T , we consider the following set of manipulations, motivated by the tools a digital compositor might use: **{translation, scaling, trimming, in-painting}**. In Section 4, we discuss how these transformations are applied. In addition, each object is assigned to a unique discrete layer, z_ℓ .

A scene collage also consists of a set of semantic object relationships represented as a **scenograph**, \mathcal{S} (Figure 2, right). The scenograph provides context for each object in the scene. The structure of the scenographs is described in Section 2.1. Treating the dictionary as deterministic, we model the probability of a scene collage just in terms of random variables for the dictionary indices used, the transformation and layering parameters, and the scenograph relationships: $\mathbf{X} = \{\mathbf{L}, \theta, \mathbf{z}, \mathcal{S}\}$. The posterior we seek to optimize is

$$P(\mathbf{L}, \theta, \mathbf{z}, \mathcal{S} \mid I) \propto P(I \mid \mathbf{L}, \theta, \mathbf{z}, \mathcal{S}) P(\mathbf{L}, \theta, \mathbf{z}, \mathcal{S}). \quad (2)$$

2.1. Scenographs

We use object interrelationships to add context to our scene representation. We represent these relationships as a graph *support* constraints: if object A physically supports object B , then B is a child of A . We call the resulting graph a scenograph, borrowing the term from the graphics community [20]. At most one parent is assigned to each child

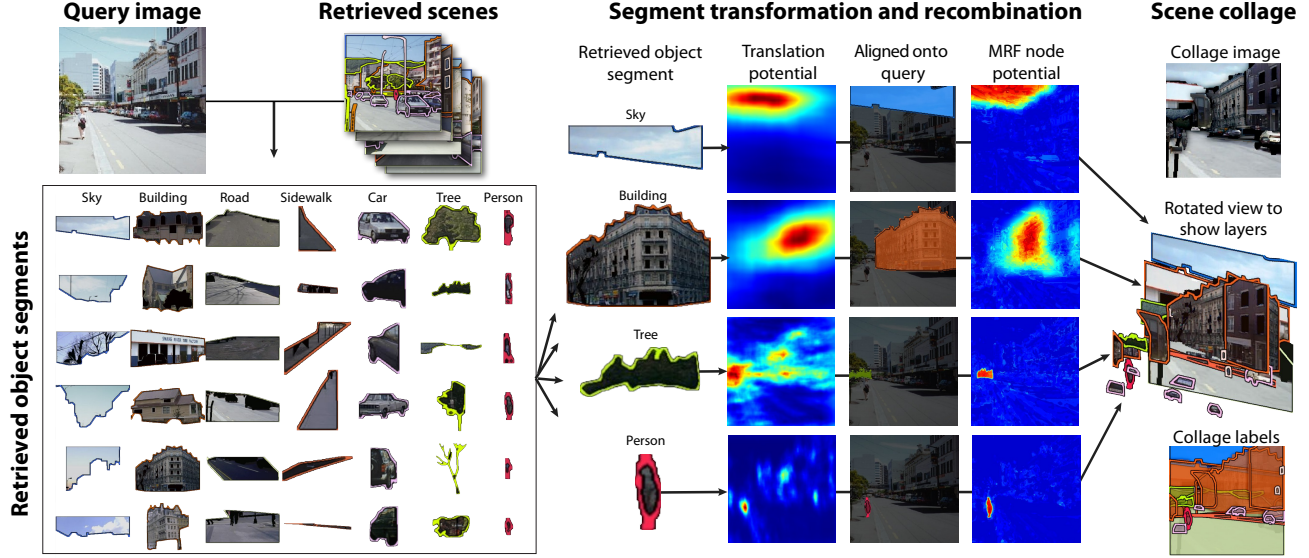


Figure 3: Overview of our approach. Starting with a query image (top left), we retrieve similar images from our dictionary. From these images, we select a set of candidate object segments for use in a collage (bottom left). We align each candidate segment onto the query image (column “Translation potential” depicts the cost of centering the segment at each point in the image, with red being low cost; column “Aligned onto query” shows the aligned mask). Segment boundaries are trimmed and refined with MRF-based segmentation (column “MRF node potential” depicts the probability of each pixel being assigned to a particular segment, with red being high probability). Finally, we choose a set of segments that semantically fit together and well explain the appearance of the query. These segments are layered together to form a “scene collage” (far right).

node, making the resulting graph a forest. An example of a scenegraph is shown in Figure 2.

Image parsing graphs have been previously proposed by a number of authors (e.g. [27], [26]), and physical support inference has also recently been explored [19]. Our representation of object relationships is especially similar to Russell et al.’s LabelMe 3D representation [16]. However, whereas Russell et al. demonstrated that this representation is useful for inferring scene structure from human annotations, here we attempt to show its utility toward inferring structure in unlabeled images. Gupta et al. also explored a similar representation, in which objects in a scene are related by a graph of support constraints [8]. However, they used a 3D block representation and automatic segmentation, while we use a simpler, layer-world representation built from human-labeled exemplar segments.

Scenegraph estimation: To estimate the scenegraph for a scene, we apply the following rules: Object A is a parent of B if A is on a higher layer than B and either 1) B is the predominant object in a small region directly below A in the y -coordinate, or 2) B covers $\geq 99\%$ of the region directly underneath A in the z -coordinate.

2.2. Likelihood model

An image, I , is generated by a scene collage, \mathbf{X} , as follows. Each object in a scene is associated with a function \tilde{g}_ℓ , which is a generative model of appearance. Intuitively, the object is filled with visual stuff and we model that stuff

as a spatially varying distribution over features we expect to see in different subregions of the object’s mask (Figure 4).

Rather than modeling the appearance of raw pixels, we model the generation of image features. Indexing coordinates within an image with q , $f_q^{(I)}$ is the feature vector at q in image I .

We use a spatial grid partitioning scheme. $\tilde{g}_\ell(\cdot, \cdot)$ is a 2D distribution with the first dimension over feature and the second over sub-bin of the object’s bounding box, BB_ℓ (BB_ℓ is the bounding box of Q_ℓ). Using \tilde{I}_ℓ to denote the image pixels of the exemplar indexed by ℓ in our dictionary, we first measure a histogram, \tilde{h}_ℓ , of visual words in each sub-bin B :

$$\tilde{h}_\ell(f, B) = \frac{1}{|B \cap \mathcal{V}_\ell|} \sum_{q \in B \cap \mathcal{V}_\ell} \mathbb{1}(f = f_q^{(\tilde{I}_\ell)}), \quad (3)$$

where \mathcal{V}_ℓ is an object’s visibility mask:

$$\mathcal{V}_\ell = Q_\ell \setminus \cup_{\{\ell' \in \mathcal{L} \text{ s.t. } z_{\ell'} < z_\ell\}} Q_{\ell'}. \quad (4)$$

We arrive at the appearance model for an object by averaging its histogram with the histograms of similar segments in our dictionary:

$$\tilde{g}_\ell(f, B) = (1 - \lambda) \tilde{h}_\ell(f, B) + \lambda \sum_{i \in \mathcal{K}} \tilde{h}_{\ell_i}(f, B), \quad (5)$$

where \mathcal{K} is a set of $K = |\mathcal{K}|$ nearest-neighbor dictionary segments of the same class \tilde{c}_ℓ . Averaging with the histograms of these similar segments acts regularize the appearance model. We learn the parameters λ and K from training data, as described in Section 3.

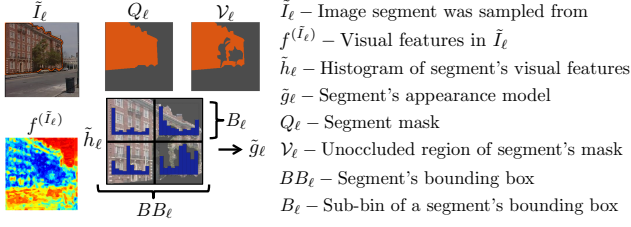


Figure 4: Notation for our likelihood model. Here we depict our model for the visual appearance of a building segment.

We model the likelihood of an entire image I as the probability of each pixel in the image under the appearance model of the object whose visibility mask covers that pixel:

$$p(I|\mathbf{X}) = \prod_{\ell \in \mathbf{L}} \prod_{q \in \mathcal{V}_\ell} \tilde{g}_\ell(f_q^{(I)}, B_\ell(q)), \quad (6)$$

where $B_\ell(q)$ is the bounding box sub-bin of BB_ℓ in which q falls.

For features f , we use a variant of HOG visual words [2]: we augment HOG feature vectors with the a and b components of the L^*a^*b color space, and we vector quantize these into 1000 visual words using k-means, giving ‘‘HOG-color’’ visual words.

2.3. Priors – scene grammar

We represent scene priors with a nonparametric scene grammar. Our grammar defines a set of valid scenegraphs. The intuition behind our grammar is that we interpolate between the scenegraphs of exemplar scenes in our dictionary. We consider the following set of moves: **{birth, death, swap}**. Each move takes a given valid scenegraph and restructures some part of it to be more like the scenegraph of an exemplar scene. Intuitively, a birth move adds an object to our scenegraph, a death move removes an object, and a swap exchanges one object for another. In Table 1, we define the set of possible moves that can be applied to make a scenegraph \mathcal{S} more like an exemplar scenegraph \mathcal{E} . As a concrete example, if one of our exemplar scenegraphs has a ‘‘rock’’ that is the child of a ‘‘sea’’, then our grammar will include the production rule: a ‘‘rock’’ may be born on a ‘‘sea’’. Further examples of valid moves in this scenario are shown in Figure 5.

Any scene whose scenegraph can be generated in this way is assigned prior probability $p(\mathbf{X}) = c$ (c is a constant such that $\int p(\mathbf{X}) = 1$). All other possible scenes are given probability 0. In addition, we require that the transformed, layered objects in the scene are consistent with the scenegraph, \mathcal{S} , in the sense that if we were to apply the scenegraph estimation rules (Section 2.1) to a scene with variables $\{\mathbf{L}, \theta, \mathbf{z}\}$, it would result in a scenegraph identical to \mathcal{S} . Scenes that do not satisfy this requirement are assigned prior probability $p(\mathbf{X}) = 0$.

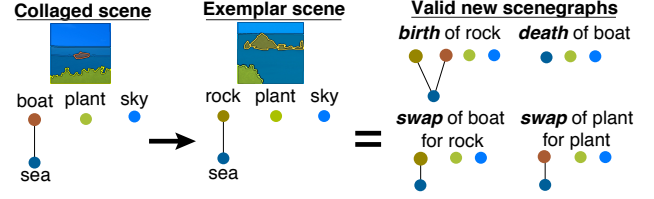


Figure 5: Valid scenegraphs (right) that are produced by taking the collaged scene (left) and moving it toward the scenegraph of a dictionary exemplar (middle).

3. Learning

We learn the parameters λ and K of our likelihood model (Equation 5) from our dictionary of training segments. We cluster dictionary segments by object class and mask shape, and a cluster-specific λ and K is learned for each cluster. We split each cluster into training and test sets $(\mathcal{T}_1, \mathcal{T}_2)$, and then choose the parameters that maximize the likelihood of test segments under the maximizing assignments of training segment models \tilde{g} to test segment appearances f , i.e.

$$\{\lambda^*, K^*\} = \arg \max_{\{\lambda, K\}} \sum_{i \in \mathcal{T}_2} \arg \max_{\ell \in \mathcal{T}_1} \sum_{q \in \mathcal{Q}_i} \tilde{g}_\ell(f_q^{(\tilde{I}_i)}, q; \lambda, K). \quad (7)$$

We perform this optimization using hard-EM [3].

4. Inference

Through analysis-by-synthesis, we infer a scene collage that maximizes the model’s posterior:

$$\mathbf{X}^* = \arg \max_{\mathbf{X}} p(I | \mathbf{X}) p(\mathbf{X}). \quad (8)$$

Our system pipeline (in Algorithm 1) consists of three stages: segment retrieval, segment transformation, and segment recombination (Figure 3). We first retrieve a set of candidate segments to be used in collaging. We then transform each candidate to align with the query image. Finally, we select a subset of the transformed candidates and combine them into an collage that explains the query.

4.1. Segment retrieval

We start by retrieving the N -nearest-neighbors images in our dictionary, based on spatial pyramid matching (SPM) [11] with our HOG-color visual words. For each of these N exemplar scenes, we retrieve its k -nearest semantic neighbors using SPM over pixel labels. This gives us N candidate sets of retrieved scenes, $\{\mathcal{C}\}_{n=1}^N$, which we call **context sets** ($N = 10$ in our experiments). Each context set contains k_1 semantically coherent scenes ($k_1 = 20$ in our experiments). For example, in matching a picture of sand dunes, we may retrieve one context set of beaches and one context set of deserts.

Our next step is to retrieve good candidate object segments from $\{\mathcal{C}\}_{n=1}^N$. For each context set \mathcal{C}_n , we select a

Table 1: Table of valid scenegraph changes. Notation: $\text{par}(A, \mathcal{S})$ is the parent of object A in scenegraph \mathcal{S} , $\text{class}(A)$ is the object class of A . \mathcal{S} is the scenegraph being changed and \mathcal{E} is the exemplar scenegraph the change is based on.

Move type	Effect	Rule for validity
<i>Birth</i>	Add an object A	Parent requirement: $\text{class}(\text{par}(A, \mathcal{S})) = \text{class}(\text{par}(A, \mathcal{E}))$
<i>Death</i>	Remove an object B	Always valid
<i>Swap</i>	Exchange an object B for A	<i>Birth</i> (A) is valid and $\frac{Q_{A \cap Q_B}}{Q_{A \cup Q_B}} > 0.25$

Algorithm 1 Greedy optimization

```

for all  $\mathcal{C}_n \in \{\mathcal{C}\}_{n=1}^N$  do
   $\mathcal{R}_n \leftarrow \text{RETRIEVESEGMENTS}(\mathcal{C}_n)$ 
   $\mathbf{X}_n \leftarrow \text{EMPTYSCENE}$ 
  while not converged do
     $\mathcal{M} \leftarrow \text{VALIDMOVES}(\mathbf{X}_n, \mathcal{C}_n, \mathcal{R}_n)$ 
    for all  $m \in \mathcal{M}$  do
       $\mathbf{X}_n^{(m)} \leftarrow \text{APPLYMOVE}(\mathbf{X}_n, m)$ 
    end for
     $\mathbf{X}_n \leftarrow \arg \max_m p(\mathbf{X}_n^{(m)} | I)$ 
     $\mathbf{X}_n \leftarrow \text{TRIMANDGROW}(\mathbf{X}_n, I)$ 
  end while
end for
 $\mathbf{X} \leftarrow \arg \max_n p(\mathbf{X}_n | I)$ 
return  $\mathbf{X}$ 

```

retrieval set, \mathcal{R}_n , of object segments, chosen from amongst the segments that make up the scenes in \mathcal{C}_n . We choose segments based on the similarity of their visual features to the features of the query image. In order to sample segments of a variety of sizes, we consider big object segments separately from small object segments. In particular, we first group objects by the area of their mask ($|Q_\ell|$) (in our experiments, we use 10 linearly spaced size categories). Then, for objects in each given size category, we select the k_2 segments that maximize the histogram intersection between the spatial pyramid distribution of HOG-color visual words in the segment and in the query image ($k_2 = 40$ in our experiments). The union of all these sets of k_2 segments gives us our full retrieval set \mathcal{R}_n .

4.2. Segment transformation

Once we have retrieved a set of candidate segments, we warp each to align with the query image using transformation function T (Equation 1), which consists of three separate stages: “translation and scaling”, “layering”, and “trimming and growing” (Figure 3). Objects are translated and scaled during an initial pass over all retrieved segments, whereas layering, trimming and in-painting occurs at each iteration of the segment recombination algorithm (Section 4.3). A brief overview of each step is provided below. For details, please refer to the supplemental materials.

Translation and scaling: Each object from our dictionary is transformed independently. Using a sliding windows approach, we place the object at a location and scale

where it maximizes the image likelihood under our generative model from Section 2.2.

Layering: When an object ℓ is added to a collage, we insert it at a layer z_ℓ . All other object segments in the collage at or below this layer are pushed back: $z_{\ell'} \leftarrow z_{\ell'} - 1 \quad \forall z_{\ell'} \geq z_\ell$. We choose z_ℓ as the assignment that maximizes image likelihood under the new collage. However, we only consider layer assignments that are valid in the sense that the collage’s scenegraph remains consistent with the collage’s layering order (consistent according to the scenegraph estimation rules of Section 2.1). Also, as a special case, “sky” segments are always placed on the bottom-most layer.

Trimming and growing: We edit object masks (Q_ℓ) after each iteration of segment recombination. We formulate this part of the problem as 2D MRF-based segmentation, in which segments in a collage compete to explain pixels near their mask boundaries. The energy of the MRF is based on the likelihood each segment assigns to the pixels under our likelihood model from Section 2.2.

4.3. Segment recombination

For each context/retrieval set, we greedily recombine our retrieved, transformed segments in order to explain the query image. We start with a set of N empty collages, one for each of our N context sets. Each context set, \mathcal{C}_n , defines a context-dependent scene grammar. This context-dependent grammar is the same as our scene grammar from Section 5, but rather than considering moves toward any exemplar scene in our dictionary, we only consider moves toward exemplar scenes in the context set. Intuitively, if our context set is a bunch of beach scenes, we will only consider beach-like scenegraphs as valid.

The context-dependent scene grammar defines the set of valid semantic changes we can make to a collage. In order to instantiate these semantic changes with actual object segments changes, we consider using each of the object segments in the retrieval set \mathcal{R}_n . At each iteration of greedy optimization, we simply choose the change that maximizes our posterior. This generates N collages, from which we choose the max a posteriori collage as a final explanation of the query (Algorithm 1).

We speed up this optimization with three further tricks. First, we use the lazy greedy evaluation method described in [12]. Second, we consider segments in \mathcal{R}_n in a coarse-to-fine manner. On the first iteration of greedy optimization, we only consider changes involving

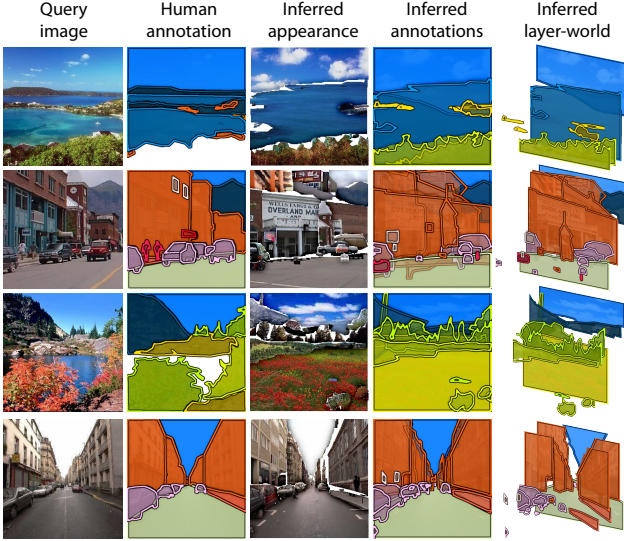


Figure 6: LMO results. Our algorithm is relatively successful at parsing outdoor scenes. Notice that each region of a query image is matched to a similar but not identical object segment from our dictionary.

large objects. Once no further large objects improve the posterior (or after a fixed max number of iterations), we move on to objects at a smaller scale. We repeat this process, iterating over the following set of scales, where the numbers represent the range of object mask sizes ($|Q_\ell|/|I|$) under consideration at each iteration: $\{[0.24, 1], [0.12, 0.5], [0.06, 0.25], [0.03, 0.12], [0, 0.06]\}$. As a final trick, between each regular iteration, we apply all valid *death* moves that improve the posterior, which serves to quickly cull away any unhelpful segments.

5. Results

We test our algorithm on two datasets: the SUN Database [24] (including a benchmark subset called the LabelMe Outdoor Database (LMO) [13]), and the NYU RGBD dataset version 2 [19]. Each of these datasets has been annotated with object masks by humans. Example parses using our system are shown in Figures 6, 7, and 8. An example inferred scenegraph is shown in Figure 9. More examples can be found in the supplemental materials.

Our implementation takes about 30 minutes at test time to parse a single scene on a single core. Training – computing features and learning the likelihood model parameters – takes a few hours on a single core for a dictionary of several thousand scenes, and scales linearly with dictionary size.

5.1. Evaluating segmentation and labeling

To assess our performance at the traditional problem of pixel-labeling, we employ two standard measures from the literature: mean per-pixel label accuracy and mean per-class label accuracy [22]. We compare results to state-of-the-art

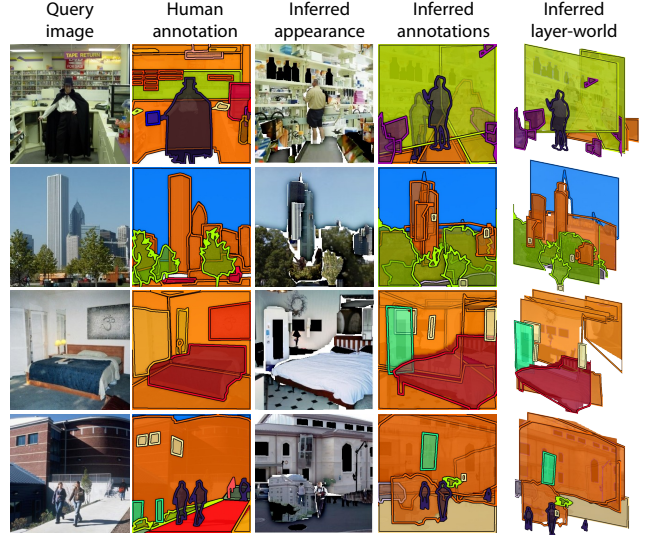


Figure 7: SUN results. Because our algorithm uses a layer world, it represents to some degree occluded portions of the scenes. Notice in row 3 that occluded regions of the wall and floor are inferred, which would not be possible in a purely 2D representation.

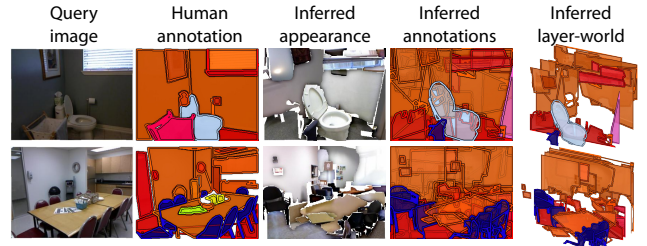


Figure 8: Indoor scenes pose greater difficulty.

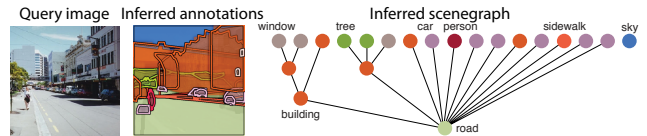


Figure 9: In addition to inferring layered object segments, we also infer how they support one another, as captured by a scenegraph.

methods in Table 2. While we do worse at pixel-labeling than these algorithms, we infer a much richer representation than a grid of pixel labels.

5.2. Evaluating layers and support relationships

In addition to inferring a segmentation map and pixel-wise labels, our model explicitly represents discrete, layered objects and their support relationships. To compare these inferences to “ground truth”, we need a way of estimating ground truth. To estimate the layer order of a scene with only 2D object annotations (i.e. LMO and SUN scenes), we use the heuristics from [17]. For RGBD scenes (i.e. the NYU dataset), we sort segments by max depth. To estimate ground truth scenegraph support relationships, we

Table 2: Comparison with other methods on LMO.

Algorithm	Mean per-pixel accuracy (mean per-class accuracy)
Liu et al. [13]	76.7 (N/A)
Fabaret et al. [6]	78.5 (29.6) / 74.2 (46.0)
Eigen and Fergus [4]	77.1 (32.5)
Tighe and Lazebnik [21]	78.6 (39.2)
Our method	70.0 (26.2)

Table 3: Performance of our algorithm

Dataset	Mean per-pixel accuracy	Mean per-class accuracy	Layer Score	Support Score
LMO	0.70	0.26	0.24	0.23
SUN	0.40	0.03	0.12	0.11
NYU RGBD	0.30	0.01	0.11	0.12

apply the scenegraph construction rules (Section 2.1) to the ground truth annotations.

We then compare our inferred layer orders and scenegraphs to ground truth using the following metric. First, we match each object ℓ' in the ground truth to a best matching object ℓ in our explanation, which is defined to be the object in the explanation of the same object class as ℓ' (if one exists) for which $(Q_{\ell'} \cap Q_{\ell}) / (Q_{\ell'} \cup Q_{\ell})$ is maximized. Next, we list all pairwise relationships between objects in the ground truth scene, \tilde{r} , and all such relationships between objects in the inferred collage, \tilde{r} . We consider both *pairwise layer order* and *scenegraph support relationship* (*parent*, *child*, *no relation*). We measure performance as the number of relationships in \tilde{r} that also hold for r when the objects are matched as described above, divided by the total number of relationships in \tilde{r} and r . That is, $Score = (\tilde{r} \cap r) / (\tilde{r} \cup r)$. We separately report the score measured in this way for layer relationships (*LayerScore*) and for support relationships (*SupportScore*).

Table 3 enumerates our results on each dataset. While our method often produces reasonable parses of outdoor scenes, it has more difficulty with indoor environments, such as those in the NYU RGBD dataset.

5.2.1 System variation comparisons

In Table 4, we compare several variations of our system. “Full system” refers to the full system described above. “Nearest-neighbor” refers to just using the single nearest-neighbor retrieved scene as the raw explanation for a query image. “No recombination” does not allow objects from more than one dictionary scene to be used in a collage (i.e. the number of scenes in each context set is 1), but otherwise the algorithm proceeds as usually. As the results show, allowing segments to transform and recombine from multiple sources boosts pixel labeling accuracy compared to the “Nearest-neighbor” and “No recombination” baselines. However, interestingly, just using the raw nearest-neighbor does better on layer and support scores. Apparently, by

Table 4: System variations.

	Mean per-pixel accuracy	Mean per-class accuracy	Layer Score	Support Score
Nearest-neighbor	0.55	0.20	0.35	0.32
No recombination	0.62	0.21	0.18	0.20
Full system	0.70	0.26	0.24	0.23
Maxent prior	0.68	0.18	0.15	0.15

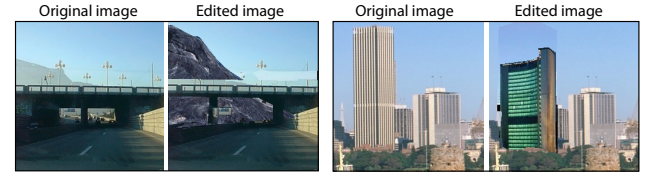


Figure 10: Using our layer world representation, we can easily swap one object for another. Notice that the mountain gets automatically placed behind the bridge since layers are explicitly represented.

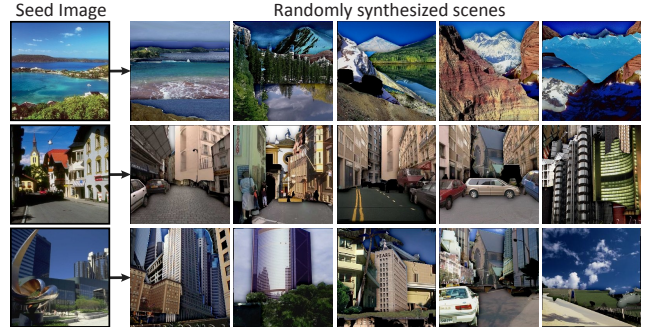



Figure 11: Random scenes, on right, synthesized from seed in column on left. These samples look like reasonable scenes, suggesting we are staying close to the set of natural scenes as we randomly walk over our scene grammar. The right-most column shows failure cases where the synthesized scenes do not look very natural.

better fitting the appearance of a query image through segment transformations and recombinations, we break some of the rich structure that comes naturally intact in nearest-neighbor images. Finally, we compare to adding a maximum entropy prior over object co-counts (“Maxent prior”; described in detail in the supplemental materials). This prior over-regularizes somewhat, resulting in a decrease in the per-class accuracy and the layer and support scores.

6. Applications

Image editing: One benefit of our representation is that it is easy to manipulate by humans. Indeed, layer-worlds are often used in consumer image editing software such as Photoshop. In Figure 10, we show a demo of editing an image. Say a user does not like the tall building in the right image, and wants to replace it. We can use our system to parse the image and provide the user with a set of discrete, manipulable image layers. The user may then swap out the building for a more colorful one. Since our system knows layer



Figure 12: Image-to-anaglyph. Please view with red-blue anaglyph glasses . Our system is able to recover rough 3D from the layer order. However, not all scenes are well modeled by a layer-world. For example, the roads in the street scenes above should extend through multiple depth layers, but our representation cannot capture this.

order, and partially represents occluded pixels, it can intelligently in-paint the region behind a changed object. Here we have implemented just a demo in-painting algorithm that fills in these regions with nearest-neighbor segments.

Random scene synthesis: Since our method is generative, it can be used to synthesize random scenes. Starting with a seed image, we retrieve a random context set from similar looking scenes, which gives us a context-dependent scene grammar. We then take a random walk over productions from this grammar (biasing toward collages that cover as much of the image frame as possible). Example syntheses are given in Figure 11.

Image-to-anaglyph: The depth order of layers in a collage provides rough 3D information. We can use this information to visualize a scene in anaglyph stereo (Figure 12). First we transfer depth order from the collage to the query image. Shifting the camera reveals occluded pixels, which we in-paint with the pixels from both the collage and nearest-neighbor images.

7. Conclusion

We have presented a novel and intuitive framework for scene understanding in which we explain an image by piecing together a collection of exemplar scene elements. Our system moves beyond the standard problem of labeling a pixel grid and into an exciting realm of representing layers, discrete objects, and object interrelationships.

Acknowledgments: We would like to thank Ted Adelson, Bill Freeman, Yair Weiss, and Katie Bouman for helpful discussions. Most of this work was done while Phillip Isola was an intern at Microsoft Research New England. Phillip Isola is supported by an NSF graduate research fellowship.

References

- [1] T. Chen, M. M. Cheng, P. Tan, and A. Shamir. Sketch2Photo. *SIGGRAPH Asia*, 2009. 2
- [2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005. 4
- [3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 4
- [4] D. Eigen and R. Fergus. Nonparametric Image Parsing using Adaptive Neighbor Sets. *CVPR 2012*, 2012. 7
- [5] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. PhotoSketch: A Sketch Based Image Query and Compositing System. In *SIGGRAPH*, 2009. 2
- [6] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers. *Arxiv preprint arXiv:1202.2160*, 2012. 7
- [7] R. Guo and D. Hoiem. Beyond the Line of Sight: Labeling the Underlying Surfaces. *ECCV*, 2012. 2
- [8] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 2, 3
- [9] X. He, R. Zemel, and M. Carreira-Perpinán. Multiscale conditional random fields for image labeling. *CVPR*, 2004. 1
- [10] N. Jojic and B. Frey. Learning flexible sprites in video layers. *CVPR*, 2001. 2
- [11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2:2169–2178, 2006. 4
- [12] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007. 5
- [13] C. Liu, J. Yuen, and A. Torralba. Nonparametric Scene Parsing via Label Transfer. *PAMI*, 2011. 1, 2, 6, 7
- [14] T. Malisiewicz and A. A. Efros. Recognition by association via learning per-exemplar distances. In *CVPR*, 2008. 2
- [15] B. C. Russell, A. Efros, J. Sivic, W. Freeman, and A. Zisserman. Segmenting Scenes by Matching Image Composites. *NIPS*, 2009. 2
- [16] B. C. Russell and A. Torralba. Building a database of 3D scenes from user annotations. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pages 2711–2718. IEEE, 2009. 2, 3
- [17] B. C. Russell, A. Torralba, and K. Murphy. LabelMe: a database and web-based tool for image annotation. *IJCV*, 2008. 6
- [18] J. Shotton, J. Winn, and C. Rother. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009. 1
- [19] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. *ECCV*, 2012. 3, 6
- [20] P. S. Strauss and R. Carey. An object-oriented 3d graphics toolkit. *SIGGRAPH Comput. Graph.*, 26(2):341–349, July 1992. 2
- [21] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR*, 2013. 2, 7
- [22] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *IJCV*, 2013. 1, 2, 6
- [23] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 1994. 2
- [24] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 6
- [25] J. Yuen, C. Zitnick, C. Liu, and A. Torralba. A Framework for Encoding Object-level Image Priors. *Microsoft Research Technical Report*, 2011. 2
- [26] Y. Zhao and S. Zhu. Image Parsing via Stochastic Scene Grammar. *NIPS*, 2011. 2, 3
- [27] S. Zhu and D. Mumford. *A stochastic grammar of images*, volume 2. Now Publishers Inc., 2007. 3