Generating Holistic 3D Scene Abstractions for Text-based Image Retrieval

Ang Li Jin Sun Joe Yue-Hei Ng Ruichi Yu Vlad I. Morariu Larry S. Davis
Institution for Advanced Computer Studies
Univesrity of Maryland, College Park, MD 20742
{angli,jinsun,yhng,richyu,morariu,lsd}@umiacs.umd.edu

Abstract

Spatial relationships between objects provide important information for text-based image retrieval. As users are more likely to describe a scene from a real world perspective, using 3D spatial relationships rather than 2D relationships that assume a particular viewing direction, one of the main challenges is to infer the 3D structure that bridges images with users' text descriptions. However, direct inference of 3D structure from images requires learning from large scale annotated data. Since interactions between objects can be reduced to a limited set of atomic spatial relations in 3D, we study the possibility of inferring 3D structure from a text description rather than an image, applying physical relation models to synthesize holistic 3D abstract object layouts satisfying the spatial constraints present in a textual description. We present a generic framework for retrieving images from a textual description of a scene by matching images with these generated abstract object layouts. Images are ranked by matching object detection outputs (bounding boxes) to 2D layout candidates (also represented by bounding boxes) which are obtained by projecting the 3D scenes with sampled camera directions. We validate our approach using public indoor scene datasets and show that our method outperforms baselines built upon object occurrence histograms and learned 2D pairwise relations.

1. Introduction

Text-based image retrieval, dating back to the late 1970s, has evolved from a keyword-based task to a more challenging task based on natural language descriptions (e.g., sentences and paragraphs) [10, 11, 22]. Queries in the form of sentences rather than keywords refer to not only object categorical information but also interations, such as spatial relationships, between objects. Those relationships are usually described in the real (3D) world due to the nature of human language. Intuitively, they can be the core feature for ranking images in many application scenarios, e.g., a user searching for images that are relevant to a particular mental image of a room layout. Not surprisingly, researchers have recently increased their focus on understanding spatial relationships from text input and retrieving semantically consistent visual information [10, 16, 23, 32].

Matching images with user provided spatial relations is challenging because humans naturally describe scenes in 3D while images are 2D projections of the world. Inferring 3D information from a single image is difficult. Most existing approaches learn from annotated data to map language directly to a probability distribution of pairwise relationships between object locations [10, 16]. However, such a distribution is non-convex and highly non-linear in the 2D image space because the (unknown) camera view affects the bounding box configurations. Consequently, the success of 2D learning based approaches naturally depends on the size of annotated training data. Also, the learner overfits easily since annotated spatial relations have a long-tailed distribution; many valid configurations happen rarely in the real world (e.g., a desk on another desk). With pairwise relations, it is also hard to enforce the fact that all objects are viewed from the same direction in an image. This argues for a holistic model for object relationships that jointly optimizes object configurations. Motivated by this, we explore an alternative model of spatial relations that generates 3D configurations explicitly based on physics.

We explore an approach that uses physical models and complex spatial relation semantics as part of an image retrieval system that generates 3D object layouts from text (rather than from images) and performs image retrieval by matching 2D projections of these layouts against objects detected in each database image. Our framework requires the a priori definition of a fixed set of object and spatial relation categories. Spatial relation terms are extracted from the dependency tree of the text. Objects are modeled using cuboids and spatial relations are modeled as inequality constraints on object locations and orientations. These inequality constraints can become very complex, containing nonlinear transformations represented using first order logic. Consequently, an interval arithmetic based 3D scene solver is introduced to search for feasible 3D spatial layout

solutions. Camera orientations are constrained and sampled for obtaining 2D projections of candidate scenes. Finally, images are scored and ranked by comparing object detection outputs to a sampled set of 2D reference layouts.

Compared to 2D learning based approaches, our approach has the following advantages: (1) the mapping from language to 3D is simple since the text-based spatial constraints have a very concrete and simple meaning in 3D, simple enough to define with a few rules by hand; (2) no training data is needed to learn complex distributions over the spatial arrangement of 2D boxes given linguistic constraints (the non-linear mapping from language to 2D is handled by projective geometry) and (3) adding common sense constraints is easy when referring to physical relationships in 3D (Sec. 4.2.2), while it is hard if these constraints are specified and learned in 2D (due to the non-linearity of projective geometry). We evaluate our approach using two public scene understanding datasets [3, 27]. The results suggest that our approach outperforms baselines built upon object occurrence histograms and learned 2D relations.

2. Related Work

Text-based image retrieval has been studied for decades [22]. As both computer vision and natural language processing have advanced, recent efforts have emerged that build connections between linguistic and visual information [12, 19]. Srivastava and Salakhutdinov [28] extend Deep Boltzman Machines (DBMs) to multimodal data for learning joint representations of images and text. They apply such representations to retrieving images from text descriptions. Their model learns mappings between objects with attributes and their corresponding visual appearances; however spatial relations are not modeled.

Spatial relationships play an important role in visual understanding. Previous works make use of text-extracted spatial relations in image retrieval. Zitnick *et al*. [32] generate and retrieve abstract cartoon images from text. Cartoon object models are pre-defined and 2D clipart images are composed according to the text. Siddiquie *et al*. [24] devise a multi-modal framework for retrieving images from sources including images, sketches and text by jointly considering objects, attributes and spatial relationships, and reducing all sources into 2D sketches. However, their framework handles text with only two or three objects and very limited 2D spatial relationships. Lin *et al*. [16] retrieve videos from textual queries. A set of motion text is defined with visual trajectory properties and parsed into a semantic graph to to match video segments via a generalized bipartite graph matching. All these works rely on 2D spatial relations while our work is based on real world physical models of 3D scenes to retrieve semantically consistent images.

Interesting recent work on retrieving images from text is based on the scene graph representation [10, 23]. A scene graph is a graph-based representation which encodes objects, attributes and object relations. In Johnson *et al*. [10], text input is converted to a scene graph by a human and a CRF model is used to match scene graphs to images by encoding global spatial relations of objects rather than only pairwise relations. Their approach requires learning spatial relations from annotated image data. Our work differs in that we take a generative perspective and inject physical relation models and human knowledge into the retrieval system without the requirement of large-scale data annotation.

Many existing works utilize 3D geometry in vision tasks such as object recognition [8], image matching [15], object detection [30, 31], *etc*. However, to the best of our knowledge, the use of 3D geometry in relating images with language has not been exploited. While inferring the 3D structure from a single image is challenging and complicated in vision [3, 5, 9, 20, 21], the problem of rendering scenes from text is of interest in the graphics community. The wordseye system [4] renders scenes from text with given 3D object models. Chang *et al*. [2] generates 3D scenes from text by incorporating the spatial knowledge learned from data. In addition, some recent works cast computer vision as inverse graphics and try to incorporate computer graphics elements into visual understanding systems [13, 14, 29]. Our work also involves scene generation. However, our purpose is to retrieve similar images based on bounding boxes , which can be efficiently computed using off-the-shelf software during a database indexing step, so real object models are not required, although better scene generation could potentially improve image retrieval accuracy.

3. Preliminary: Interval Analysis

Our approach involves finding feasible solutions to a mathematical program where the variables are object coordinates and orientations, and the constraints are inequalities translated from user descriptions. Since small placement pertubations usually do not affect the fullfilment of constraints, feasible variables can naturally be represented by a set of intervals (any value within the interval is feasible).

Interval analysis represents each variable by its feasible interval, e.g., $[l, u]$ (with lower bound l and upper bound u) and the goal is to find the bound for each dimension that satisfies all constraints [26]. When an interval does not satisfy all the constraints, it is split into smaller intervals and evaluated recursively. Arithmetic operators are defined in terms of intervals, e.g.,

- *addition*: $[l_1, u_1] + [l_2, u_2] = [l_1 + l_2, u_1 + u_2]$;
- *subtraction*: $[l_1, u_1] - [l_2, u_2] = [l_1 - u_2, u_1 - l_2]$;
- *comparison*: $[l_1, u_1] < [l_2, u_2]$ equals $[0, 0]$ if $u_2 \leq l_1$ (definitely false); equals $[1, 1]$ if $u_1 < l_2$ (definitely true); equals $[0, 1]$ otherwise (maybe true).

The fulfillment of a constraint can be represented by any of the three logical intervals, *i.e*., $[0, 0], [1, 1], [0, 1]$.

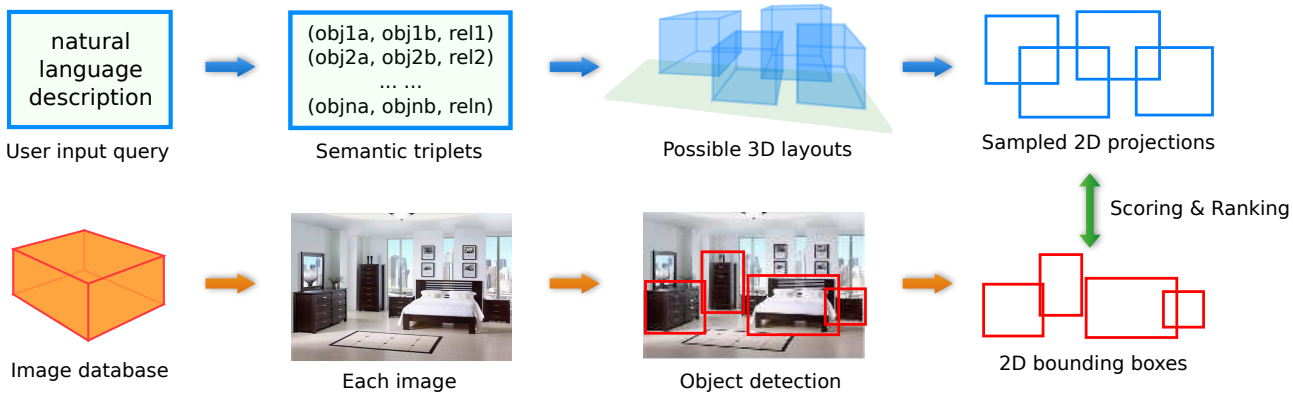


Figure 1. Framework overview: a textual description of the visual scene is parsed into semantic triplets which are used for solving feasible 3D layouts and their 2D projections as *reference configurations*. An object detector runs over each database image and generates a 2D bounding box layout, to be matched to reference configurations. All database images are ranked according to their configuration scores.

4. Our Approach

The proposed framework, as illustrated in Fig. 1, consists of several modules. First, the input text is parsed into a set of semantic triplets of object names and their spatial relationships. Second, the semantic triplets are used to solve possible 3D layouts of objects along with sampled camera locations and orientations. The 2D projections of the 3D scenes are used for generating 2D bounding boxes of objects, which we call *reference configurations*. Finally, the reference configurations are matched to the detected bounding boxes in each database image to score and rank according to their configuration similarity.

#	Sentence → (object-1, object-2, relation)
1	A picture is above a bed. `(picture-0, bed-0, above)`
2	A night stand is on the right side of the head of the bed. `(night-stand-0, bed-0:head, right)`
3	A lamp is on the night stand. `(lamp-0, night-stand-0, on)`
4	Another picture is above the lamp. `(picture-1, lamp-0, above)`
5	A dresser is on the left side of the head of the bed. `(dresser-0, bed-0:head, left)`

Table 1. Semantic triplet parsing from an example query

4.1. Text Parsing

The text parsing module translates text into a set of semantic triplets which encode the information about two object instances and their spatial interactions. How to robustly extract relations from text is still an open research problem in natural language processing [11], which is beyond the scope of this paper. For our application, a simple rule-based pattern matching works sufficiently well, requiring a predefined dictionary of object and spatial relation categories. A text example and its parsing output is shown in Table 1.

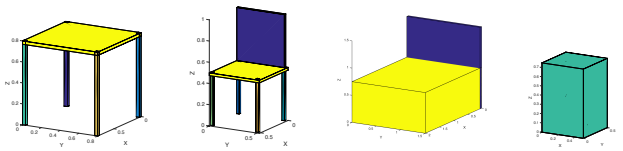
The input text is processed by the Stanford CoreNLP library [18] with part-of-speech tagging and dependency tree. We implement a rule-based approach to extract spatial relations (such as *on*, *under*, *in front of*, *behind*, *above*, etc.) from the dependency tree and compose its corresponding semantic triplet representation *(target object, reference object, relation)*. The co-reference module in the CoreNLP library is used to aggregate multiple noun occurrences that correspond to the same object instance. Each object reference is represented by its category name and a unique ID within the category, e.g. `sofa-0` and `dining-table-2`.

Natural objects are usually composed of multiple sub-objects and there are often cases when a sub-object is referenced instead of the whole object. A bed, for instance, has its head and rear. And a chair has its back and seat. We take sub-objects into consideration and represent any sub-object reference by its object category name, unique in-category ID and sub-object name, e.g. "the rear of the bed" is represented as `bed-0:rear` if the ID is 0.

Besides object categories and spatial relationships, we also consider the count of each object, e.g. three chairs, two monitors, etc. The parser maintains a list of object ID and their counts. If the count of `chair-0` is 3, then the parser will expand `chair-0` to a set of three instances {`chair-0-0`, `chair-0-1`, `chair-0-2`} in the outputs.

4.2. 3D Abstract Scene Generation

The 3D abstract scene generation module is the central component in our image retrieval framework; it takes as input semantic triplets and generates a set of sampled possible 3D object layouts. We describe below the three core components of the scene generator: the cuboid based object model, the spatial relation model and the 3D scene solver.



(a) (b) (c) (d)

Figure 2. Sample cuboid based object representations: (a) table (b) chair (c) bed (d) night-stand. Different colors represent different sub-objects. The night stand (d) is represented by a single cuboid.

4.2.1 Cuboid based object model

The basic *cuboid representation* of an object is $\mathbf{C} = (l_x, l_y, l_z, z_s)$ where (l_x, l_y, l_z) is the size of the cuboid that bounds the object in x, y, z directions respectively and z_s is the z-coordinate of the supporting surface of the object. We mostly use regular sizes but also set different sizes for objects with attributes such as `long-desk`, `triple-sofa`, etc. The supporting surface is usually the top face of the object cuboid, but it can sometimes be located elsewhere with respect to the cuboid, e.g., for a chair it is in the middle of the cuboid. Spatial relations such as `on` and `under` are modeled with respect to the surface of the object. Most of the objects can be modeled using this cuboid representation such as `garbage-bin`, `picture`, `night-stand`, etc.

However, the single cuboid representation is not sufficient for some object categories such as `chair` and `desk` since the under-surface area is empty. Considering the fact that most objects can be easily decomposed into smaller sub-objects, we represent these object categories as the union of a set of cuboids, which we call a *cuboid set representation*. Each sub-cuboid corresponds to a sub-object and is considered a simple object, whose top face is the supporting surface. The k-th sub-cuboid is represented by $\mathbf{S}^k = (d_x^k, d_y^k, d_z^k, l_x^k, l_y^k, l_z^k)$ where (d_x^k, d_y^k, d_z^k) is the offset from the lowest point of the sub-cuboid to the lowest point of the original object, and (l_x^k, l_y^k, l_z^k) is the size of the sub-cuboid. The sub-cuboid parameters $\mathbf{S}^k$ are computed as functions of the original object parameters $\mathbf{C}$. Four sampled cuboid based object models are visualized in Fig. 2.

4.2.2 Spatial relation model

The spatial location and orientation of each object is represented as $\mathbf{X} = (x, y, z, d)$ where (x, y, z) is the lowest point of the object cuboid and d is its orientation. The object rotation is around the z-axis.

Atomic relations. We model 8 basic spatial relations using the following mathematical expressions. Given the object pose and its size, the lowest point $\mathbf{p} = (x_p, y_p, z_p)^\intercal$ and highest point $\mathbf{q} = (x_q, y_q, z_q)^\intercal$ of the object cuboid can be computed by rotating the object models w.r.t. the object orientation such that

$$\mathbf{p} = \mathbf{R}_d \left[-\frac{l_x}{2}, -\frac{l_y}{2}, -\frac{l_z}{2}\right]^\intercal + \left[x + \frac{l_x}{2}, y + \frac{l_y}{2}, z + \frac{l_z}{2}\right]^\intercal,$$
$$\mathbf{q} = \mathbf{R}_d \left[\frac{l_x}{2}, \frac{l_y}{2}, \frac{l_z}{2}\right]^\intercal + \left[x + \frac{l_x}{2}, y + \frac{l_y}{2}, z + \frac{l_z}{2}\right]^\intercal \quad (1)$$

where $\mathbf{R}_d$ is the z-axis rotation matrix w.r.t. to orientation d. So an object can be represented using tuple $(\mathbf{p}, \mathbf{q}, d)$. Letting the cuboid of object-1 be $\mathbf{O}_1(\mathbf{p}_1, \mathbf{q}_1, d_1)$ with support surface z_{s1} and the cuboid of object-2 be $\mathbf{O}_2(\mathbf{p}_2, \mathbf{q}_2, d_2)$ with support surface z_{s2}, we define 8 atomic relations as

- `near`: $\mathbf{O}_1 \cap (\mathbf{p}_2 - d_{\text{near}}\mathbf{e}_{d_2}, \mathbf{q}_2 + d_{\text{near}}\mathbf{e}_{d_2}, d_2) \neq \emptyset$;
- `on`: $z_{p1} = z_{s2} \wedge \frac{\mathbf{p}_1 + \mathbf{q}_1}{2} \in_{xy} \mathbf{O}_2$;
- `above`: $z_{q2} + d_{\text{min-above}} \leq z_{p1} \leq z_{q2} + d_{\text{max-above}} \wedge \frac{\mathbf{p}_1 + \mathbf{q}_1}{2} \in_{xy} \mathbf{O}_2$;
- `under`: $z_{s1} < z_{s2} \wedge \mathbf{O}_1 \cap_{xy} \mathbf{O}_2 \neq \emptyset$;
- `behind`: $\max(\mathbf{u}_{d_2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2}^\intercal \mathbf{q}_1) \leq \min(\mathbf{u}_{d_2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2}^\intercal \mathbf{q}_2)$;
- `front`: $\min(\mathbf{u}_{d_2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2}^\intercal \mathbf{q}_1) \geq \max(\mathbf{u}_{d_2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2}^\intercal \mathbf{q}_2)$;
- `on-left`: $\min(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_1) \geq \max(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_2)$;
- `on-right`: $\max(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_1) \leq \min(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_2)$;

where $d_{\text{near}}, d_{\text{min-above}}, d_{\text{max-above}}$ are distance thresholds, $\mathbf{p} \in_{xy} \mathbf{C}$ means point $\mathbf{p}$ is inside the cuboid $\mathbf{C}$ on the x-y plane, $\cap$ represents the intersection of two cuboids and $\cap_{xy}$ the intersection of two cuboids on the x-y plane, and $\mathbf{u}_\theta = (\cos\theta, \sin\theta, 0)^\intercal$ is a unit direction vector and $\mathbf{e}_\theta = (\cos\theta - \sin\theta, \sin\theta + \cos\theta, 1)^\intercal$ is a vector that enlarges the effective object cuboid.

Composite relations. In natural language, there are far more spatial relation descriptions than the above mentioned 8 relations. However, most of the spatial relations can be defined based on the 8 atomic relations. Two examples are

- `next-to`: `on-left`$(\mathbf{O}_1, \mathbf{O}_2) \vee$ `on-right`$(\mathbf{O}_1, \mathbf{O}_2)$;
- `side-by-side`: $d_1 = d_2 \wedge$ `near`$(\mathbf{O}_1, \mathbf{O}_2)$;

In addition, another relation is modeled which is usually used for a set of multiple instances $\{\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_k\}$ of the same object category, i.e.,

- `in-a-row`: $d_i = d_{i+1} \wedge$ `on-right`$(\mathbf{O}_i, \mathbf{O}_{i+1}), \forall i$;

Group relations. If an object reference has a count more than 1, then all of its instances form a group, which often interacts with other objects as an entirety. If a group of k instances occurs in the triplet as the target, we create k new triplets with the same reference and relation. If the group occurs as the reference, then we create a new virtual object whose cuboid is bounded by all of its instances.

Prior constraints. An effective way to reduce the search space is to incorporate common sense and reasonable assumptions into the constraints. First, we make the following

assumptions: (a) the room has two walls ($x = 0$ and $y = 0$); (b) the text description is coherent, i.e., the objects in each semantic triplet are close to each other; (c) objects are usually oriented along x-axis or y-axis directions. Second, no pair of objects overlap with each other, i.e.,

- `exclusive`: $\mathbf{S}_i^v \cap \mathbf{S}_j^w = \emptyset \forall i, j, v, w$

where $\mathbf{S}_i^v$ is the v-th component (sub-cuboid) of the i-th object. Many other constraints are related with object properties: (a) picture, door, mirror are on the wall, i.e. $x = 0 \vee y = 0$; (b) for relation next-to, in-a-row, side-by-side, if either reference or target is against the wall, the other ones are also against the wall and they should also have the same orientation; (c) bed, night-stand, sink are against the wall; (d) bed, night-stand, sofa are on the ground.

4.2.3 3D scene solver

Let $\mathbf{X} = \{x_1, y_1, z_1, d_1, \ldots, x_n, y_n, z_n, d_n\} \in \mathbb{R}^{4n}$ be a *layout state* representing the locations and orientations of all objects. We construct *constraint function* $F : \mathbb{R}^{4n} \rightarrow \{0, 1\}$ which evaluates all prior constraints and relational constraints. The goal is to find the feasible solution set $\mathbf{S}$ such that $F(\mathbf{X}) = 1$ for all $\mathbf{X} \in \mathbf{S}$.

Our solver is based on interval analysis [26] where any variable is represented by an interval (an uncertain value) instead of a certain value. We use a vector of size 2 to represent an interval, i.e., a lower bound and an upper bound. Under interval analysis, the domain of layout states becomes $\mathbb{R}^{4n\times 2}$ and the constraint function becomes $F : \mathbb{R}^{4n\times 2} \rightarrow \{[0, 0], [0, 1], [1, 1]\}$. Starting with a candidate queue containing an initial interval layout state $\{\mathbf{X}_0\}$, our solver examines the candidate states one at a time. For each state $\mathbf{X}_i \in \mathbb{R}^{4n\times 2}$, if $F(\mathbf{X}_i) = [1, 1]$, then $\mathbf{X}_i$ is feasible and appended to the solution set. If the constraint fullfillment is undecidable, i.e., $F(\mathbf{X}_i) = [0, 1]$, then $\mathbf{X}_i$ is divided into two equally sized intervals by splitting the variable with the largest uncertainty. The two new states are appended to the candidate queue. Otherwise, $F(\mathbf{X}_i) = [0, 0]$ and no feasible solution is within the space bounded by $\mathbf{X}_i$. In the end, any layout in the solution set is guaranteed to meet all constraints. An advantage of the method is that it does not require computing the gradient of constraint F. The pseudo-code is shown in Algorithm 1.

Interval shrinkage. The original interval analysis does not make full use of equality constraints, e.g., when a variable is constrained to equal another variable, it becomes redundant to divide both of their intervals since one can be directly computed based on the other. In addition, many spatial relations are transitive, e.g., if object A is in front of object B and B is in front of C, then A is likely to be in front of C but with a larger distance. Such inferred constraint can benefit the solver with a better pruning power. Based on these observations, we develop the interval shrinkage operation which pre-computes lower bound matrices $\mathbf{L}^x, \mathbf{L}^y, \mathbf{L}^z \in \mathbb{R}^{n\times n}$ and upper bound matrices $\mathbf{U}^x, \mathbf{U}^y, \mathbf{U}^z \in \mathbb{R}^{n\times n}$ for pairwise coordinate differences, i.e., $L^x_{i,j} \leq x_i - x_j \leq U^x_{i,j} \wedge L^y_{i,j} \leq y_i - y_j \leq U^y_{i,j} \wedge L^z_{i,j} \leq z_i - z_j \leq U^z_{i,j}$. The bound matrices are initialized using the original constraints and updated once we find $L^*_{i,j} < L^*_{i,k} + L^*_{k,j}$ or $U^*_{i,j} < U^*_{i,k} + U^*_{k,j}$ ($* \in \{x, y, z\}$). Before evaluating each candidate interval layout state, we shrink its variables according to the bound matrices, e.g., $\mathbf{x}_i^{\text{shrink}} = \cap_j [x_j + L^x_{i,j}, x_j + U^x_{i,j}] \cap \mathbf{x}_i$ where $\mathbf{x}_i$ is the interval of variable x_i and $\mathbf{x}_i^{\text{shrink}}$ is the interval after shrinkage.

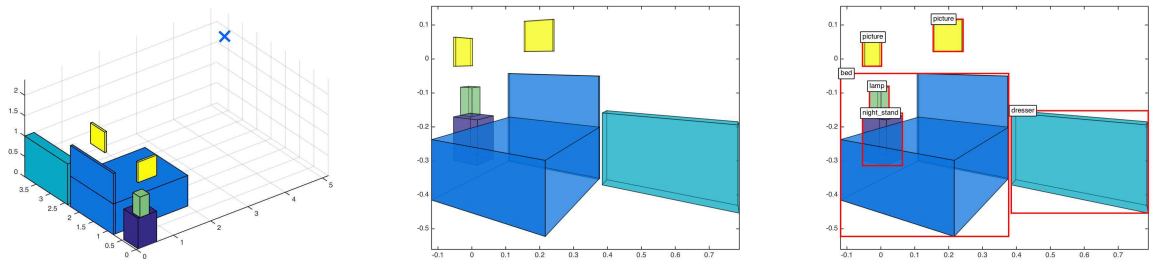
Algorithm 1: 3D scene solver

Data: Initial bounds $\mathbf{X}_0 = [\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1, \mathbf{d}_1, \ldots, \mathbf{x}_n, \mathbf{y}_n, \mathbf{z}_n, \mathbf{d}_n] \in \mathbb{R}^{4n\times 2}$
Data: Constraint $F : \mathbb{R}^{4n\times 2} \rightarrow \{[0, 0], [0, 1], [1, 1]\}$
Result: Feasible regions (or solution set) $\mathbf{S}$

```
initialization: S = ∅, Q = {X_0};
while Q ≠ ∅ do
    read the first interval: X_i = Q.front();
    remove the first interval: Q.pop();
    interval shrinkage: X_i = shrinkage(X_i);
    if F(X_i)=[0, 0] then
        X_i is not feasible;
    else if F(X_i)=[1, 1] then
        X_i is feasible: S.append(X_i);
    else if max_k |X_ik.max − X_ik.min| > tol then
        k = arg max_k |X_ik.max − X_ik.min|;
        half split k-th dimension of X_i into X_i^(1) and X_i^(2);
        Q.append(X_i^(1));
        Q.append(X_i^(2));
end
return S;
```

Early stopping. The feasible solution space can be large if the input constraints are weak. Since we sample K layouts in our framework for subsequent image matching, the 3D scene solver stops when at least K layouts are found. The sampling behaviour is achieved by implementing the candidate queue with Knuth shuffling, i.e., each time after appending a new element, the queue randomly pick an element and swaps it with the new element.

The problem is a combinatorial optimization which is NP-hard and interval analysis is essentially a breadth first search with pruning. As a result, the algorithm has no time limit guarantee. However, with interval shrinkage and early stopping, our algorithm is able to solve most queries in a reasonable amount of time. Without interval shrinkage, our MATLAB implementation can not find a solution for the query in Table 1 within 10 minutes, while it returns 5 solutions with only 6 seconds using the shrinkage operation.



(a) (b) (c)

Figure 3. The generated scene geometry for the query in Table 1: (a) a sampled 3D layout with the sampled camera location (a blue cross in the figure), (b) 2D projections of the object cuboids and (c) 2D bounding boxes of the objects.

4.3. Image Retrieval

To compare a query with image bounding boxes, we first sample feasible 3D layouts and potential camera locations and orientations to produce reasonable 2D projections of objects and then compute their bounding boxes. The whole image database is scored and ranked according to the similarity between bounding boxes detected by object detectors and those from sampled 2D layouts.

3D layout sampling. The 3D solver finds (continuous) interval solutions for 3D object coordinates; any solution within such intervals is feasible. However, the solutions within an interval are redundant; those object locations shift in tiny distances. So we sample only one layout within each interval, which results in a set of representative feasible 3D layouts. We further sample a few 3D layouts from this feasible set in order to generate their 2D projections.

2D layout projections. For each layout, we sample camera locations and orientations to obtain 2D projections which allows matching images under multiple views. Object bounding boxes are computed according to the 2D projections. Since we solve for scale and translation for each image individually during matching, in this step we only consider a canonical camera. Some heuristics are used for sampling camera locations and orientations. First, the camera always faces the objects and should be neither too close nor too far, so we sample its location from 5-10 meters from the origin. Second, the camera should not be located behind the wall, so the coordinates are positive. Third, when an object is on the wall, the camera direction should be within 60 degree offset from the object orientation. We assume the camera is 1.7 meters above the ground and situated horizontally. Fig. 3 shows an example of 3D layout, 2D projections and 2D bounding boxes for the query in Table 1.

2D layout similarity. Both detection outputs and 2D reference layouts can be represented by $\{\mathbf{b}_i, c_i\}$ where $\mathbf{b}_i$ is the 2D box of the i-th object and c_i is its category. Let $\{\mathbf{b}_i, c_i\}$ be a 2D reference layout and $\{\mathbf{b}'_i, c'_i\}$ be the detected boxes. Since scaling and translation are left as free variables, the bounding box matching involves optimizing

$$\max_{s,t,\mathbf{a}} \sum_i p(\mathbf{b}'_{a_i}) \cdot \text{IOU}(s\mathbf{b}_i + t, \mathbf{b}'_{a_i}), \quad s.t.\ c_i = c'_{a_i}, \quad (2)$$

where $p(\mathbf{b}'_k)$ is the detection confidence, IOU is intersection-over-union and assignment vector $\mathbf{a}$ indicates the correspondence between two sets of bounding boxes. In our experiment, we evaluate two versions: (a) the *hard* version uses a threshold on detection outputs and uniform $p(\mathbf{b}'_k)$ and (b) the *soft* version makes $p(\mathbf{b}'_k)$ equal to the detection score. We use a sliding window to find the best matched transformation and assignment. Specifically, we uniformly sample 5 scale factors from 0.5 to 1 w.r.t. the image space and search with a 10-pixel stride. We use a greedy strategy to compute assignments and scores (Eq. 2). The score for a query is computed as the highest score among the scores of all its sampled 2D layouts.

5. Experiments

We validate our approach using two indoor scene datasets (SUN RGB-D [27] and 3DGP [3]). Although the original goal of the two datasets is not text-based image retrieval, both contain groundtruth object bounding boxes which enables evaluation in our image retrieval setting. We compare 3 baselines built upon object occurrence histogram and 2D spatial relation based scene graph matching.

5.1. Setup

Baseline (H). The first baseline is based on the histogram of object occurrences. Specifically, both the image and text are converted to a histogram representation, i.e., a vector $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, where x_i is the number of occurrences of the i-th object category. The similarity between occurrence histograms is measured by ℓ^1 distance.

Baseline (2D). The second baseline is based on learned object relations in 2D image space. Specifically, the baseline learns a bounding box distribution of the first object w.r.t. the second object box (normalized in both x and y

coordinates). We have all eight atomic relations annotated in 1,000 images in the training set of SUN RGB-D dataset and use IOU-based nearest neighbor (IOU-NN) classifier to score for each test image the spatial relationships between object pairs. Following [1], we convert the text to a simplified scene graph that maps all instances of an object category into a single node, and assign the count of each relation as an attribute of the corresponding edge. An image scene graph with relation probabilities on edges can be constructed for each test image by using the IOU-NN relation classifier upon each pair of detected object instances. To measure the similarity between text scene graph and image scene graph, we sum for each edge (u, v, r) in the text scene graph the top $k_{u,v,r}$ corresponding relation scores in the image scene graph, where $k_{u,v,r}$ is the count of the relation r between object categories u and v in text scene graph.

Baseline (CNN). The third baseline replaces the IOU-NN relation classifier in Baseline 2D with a Convolutional Neural Network (CNN). Following [17], we finetune the pretrained VGG-19 [25] to predict predicates from cropped union image regions of the two objects. The word2vec vectors of the two objects are concatenated with the response of layer *fc7*. We backpropagate through the whole network with initial learning rate 0.001 for 90 epochs.

Evaluation metric. We evaluate different approaches to retrieving indoor images from text descriptions by measuring the percentage of queries (recall) at least one of whose ground truth images are retrieved within top k ranked images (*R@k*). The median rank (median of the ranks of all ground truths) is used as a global measurement.

Parameter selection. We set the room size to be $5m \times 5m \times 5m$. $d_{\text{near}} = 0.5m, d_{\text{min-above}} = 0.25m, d_{\text{max-above}} = 0.5m$. The tolerance in 3D scene solver is $0.2m$ because $20cm$ replacement of objects is unlikely to change the constraint fulfillment. We sample 5 reference layouts per query and 1 camera view per layout unless otherwise specified.

5.2. SUN RGB-D dataset with R-CNN detectors

SUN RGB-D Dataset [27] is a recent dataset for scene understanding which contains $10,335$ RGBD images. We use only the RGB images without depth information. We follow the same protocol as [27] by using $5,285$ images for training the detectors and the remaining $5,050$ images as the evaluation set. We annotated text queries for 150 sampled test images. SUN RGB-D contains various objects and complex spatial relations. We choose 19 object categories in our evaluation: {*bed, chair, cabinet, sofa, table, door, picture, desk, dresser, pillow, mirror, tv, box, whiteboard, night_stand, sink, lamp, garbage_bin, monitor*}, which contains not only objects on the floor but also those off the ground or on the wall such as *picture* and *mirror*.

We use the $5,285$ training images and their ground truth object bounding boxes to train Fast R-CNN [7] detectors for the 19 object categories. The R-CNN approach is built upon object proposals; non-maximum suppression is not used in postprocessing. For each test image, R-CNN detectors generate probability-like scores for all object categories on each object proposal bounding box. The category with the highest score is chosen as the bounding box category and its score is used as the bounding box confidence.

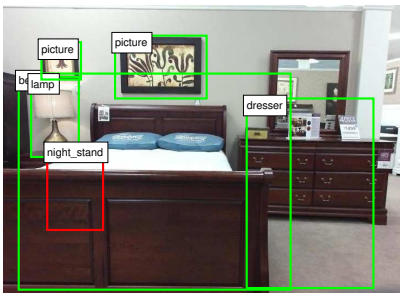
	R@1	R@10	R@50	R@100	R@500
Baseline H	1.3	4.0	14.0	20.0	43.3
Baseline 2D	2.7	15.3	35.3	44.0	64.0
Baseline CNN	2.7	16.7	30.7	36.0	63.3
Ours hard[5,1]	3.9	16.4	31.7	42.3	71.7
Ours soft[5,1]	4.5	16.7	34.0	46.4	76.0
Ours soft[5,5]	<u>4.9</u>	<u>18.7</u>	<u>37.9</u>	<u>48.1</u>	<u>76.9</u>
Ours soft[5,5] + 2D	**8.7**	**21.6**	**40.5**	**50.7**	**77.6**

Table 2. SUN RGB-D: Top-k retrieval accuracy for 150 queries. The retrieval candidate set contains 5,050 images. We evaluate the occurrence baseline (H), 2D relation baseline (2D), CNN baseline, the proposed hard version, proposed soft versions, and a combination between our soft version and the 2D baseline. The parameter of our model $[x, y]$ means sampling x 3D layouts and y camera views for each layout. All results of our model are averaged over 5 random trials. The threshold for detection outputs is 0.5. The best is shown in **bold** and the second best is shown with <u>underline</u>.

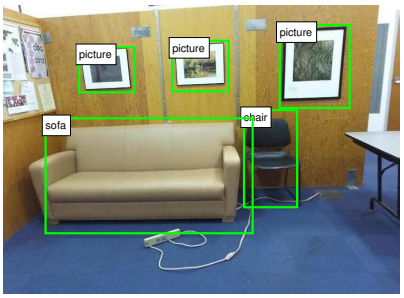
The top-k retrieval recalls are shown in Table 2. In addition with the baselines, two versions of our approach are evaluated. The baselines and our hard model use bounding boxes with over 0.5 confidence and weigh them equally, while our soft models use all bounding boxes and assign their confidences as weights in Eq. 2. The results suggest that the hard model with 5 layout samples outperforms the occurrence baseline and is on par with the 2D baseline. Our soft models perform even better than the hard one. With increased layout samples, our approach outperforms the baselines significantly. We also evaluate a combination between our soft model and the 2D baseline by adding their normalized scores. The result suggests that such combination further boost the accuracy and that our physical model based solution is complementary to learning based approaches.

Fig. 4 shows 3 examples whose ground truths are ranked top 5. The object bounding boxes that best match the generated 2D layouts are shown on the images. Green boxes are matched objects and red boxes are missing ones, expected in the generated 2D layout but unseen in the object detection output. The figure shows that our model has some level of tolerance on missing detections. A more interesting finding is that our model suggests potential locations for missing objects even though they could be heavily occluded.

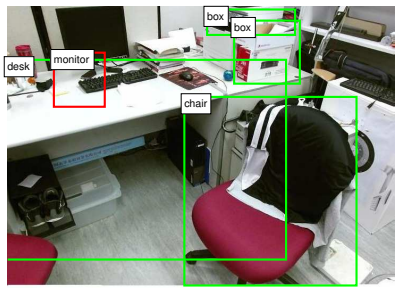
To obtain 2D layouts, we sample 3D layouts and camera views. Fig. 5 shows how the sample size of both affects the the median rank of ground truths (keeping one and varying the other). Fig. 5 suggests that more samples generally yield better performance and the improvement saturates as the



(a) A picture is above a bed. A night stand is on the right side of the head of the bed. A lamp is on the night stand. Another picture is above the lamp. A dresser is on the left side of the head of the bed.

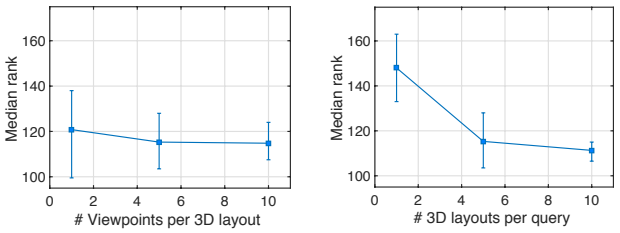


(b) There is a triple sofa. The sofa is against the wall. A chair is next to the sofa. And the chair is also against the wall. Two pictures are above the sofa. And another picture is above the chair.



(c) A chair is in front of the desk. Some boxes are on the desk. A monitor is on the desk. The desk is against the wall.

Figure 4. Matched object layouts based on our greedy 2D layout matching for three ground truth images that are ranked top 5 among all candidate images. Green bounding boxes are object detection outputs that match the 2D layouts generated from the text queries. Red bounding boxes represent a missing object (not detected by the object detector) within the expected region proposed by 2D layouts.



(a) (b)

Figure 5. Influence of # viewpoint samples and # layout samples: (a) 5 3D layouts sampled for each query, and (b) 5 viewpoint sampled for each 3D layout. The y-axis is median rank of ground truths. We random 5 times for each data point. Lower is better.

	w/ DPM bbox				w/ GT bbox			
	H	2D	CNN	Ours	H	2D	CNN	Ours
R@1	4.0	2.0	4.0	**4.4**	4.0	4.0	4.0	3.0
R@10	10.0	14.0	16.0	**16.8**	16.0	18.0	14.0	20.2
R@50	30.0	30.0	30.0	**31.2**	34.0	38.0	32.0	41.4
R@100	46.0	32.0	32.0	**52.0**	64.0	66.0	66.0	68.0

Table 3. 3DGP dataset: Top-K image retrieval accuracy. Left half is based on DPM (the best is with **bold**) and right half is based on ground truth bounding boxes (the best is in underline). The results of our approach (soft[5,5]) are averaged over 10 random trials.

sample size increases. The improvement brought by more 3D layouts is more significant than that brought by more camera views. In addition, the performance uncertainty due to randomness decreases as the sample size increases.

5.3. 3DGP dataset with DPM detectors

The 3DGP dataset [3] contains $1,045$ images with three scene types: living room, bedroom and dining room. Each image is annotated with bounding boxes for 6 object categories: *sofa*, *table*, *chair*, *bed*, *side table* and *dining table*. Following the same protocol as in [3], 622 training images are used to train the furniture detectors and the remaining 423 images are used as the retrieval image database. We use pre-trained Deformable Part Models (DPM) [6] of indoor furnitures provided by the 3DGP dataset and use the thresholds in the pre-trained models to cut off false alarms. Non-maximum suppression is used to remove duplicates.

3DGP dataset is less diverse than SUN RGB-D; many images have very similar layouts. We annotated 50 unique layout descriptions which cover 222 test images. The retrieval results are shown in Table 3. Because our method is agnostic about object detector algorithms, we split the results into two parts to separate the impact from using a specific detection algorithm: one using ground truth bounding boxes and the other using DPM detection outputs. The results suggest that our approach outperforms baseline algorithms under both bounding box settings and the improvement is independent from detector performances.

6. Conclusion

We presented a general framework for retrieving images from a natural language description of the spatial layout of an indoor scene. The core component of our framework is an algorithm that generates possible 3D object layouts from text-described spatial relations and matching these layout proposals to the 2D image database. We validated our approach via the image retrieval task on two public indoor scene datasets and the result shows the possibility of generating 3D layout proposals for rigid objects and the effectiveness of our approach to matching them with images.

Acknowledgement. This research was supported in part by funds provided from the Office of Naval Research under grant N000141612713 entitled Visual Common Sense Reasoning for Multi-agent Activity Prediction and Recognition.

References

[1] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision (ECCV)*, 2016.

[2] A. Chang, M. Savva, and C. D. Manning. Learning spatial knowledge for text to 3d scene generation. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2028–2038, Doha, Qatar, October 2014.

[3] W. Choi, Y. W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[4] B. Coyne and R. Sproat. Wordseye: An automatic text-to-scene conversion system. In *28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 487–496, New York, NY, USA, 2001.

[5] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International Conference on Computer Vision (ICCV)*, pages 2650–2658, Dec 2015.

[6] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, 32(9):1627–1645, Sept 2010.

[7] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.

[8] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 2008.

[9] D. Hoiem and S. Savarese. *Representations and Techniques for 3D Object Recognition and Scene Interpretation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[10] J. Johnson, R. Krishna, M. Stark, J. Li, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] P. Kordjamshidi, M. Van Otterlo, and M.-F. Moens. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Trans. Speech Lang. Process.*, 8(3):4:1–4:36, Dec. 2011.

[12] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, pages 1–42, 2017.

[13] T. Kulkarni, I. Yildirim, P. Kohli, W. Freiwald, and J. B. Tenenbaum. Deep generative vision as approximate bayesian computation. In *NIPS 2014 ABC Workshop*, 2014.

[14] T. D. Kulkarni, V. K. Mansinghka, P. Kohli, and J. B. Tenenbaum. Inverse graphics with probabilistic CAD models. *CoRR*, abs/1407.1339, 2014.

[15] A. Li, V. I. Morariu, and L. S. Davis. Planar structure matching under projective uncertainty for geolocation. In *European Conference on Computer Vision (ECCV)*, 2014.

[16] D. Lin, S. Fidler, C. Kong, and R. Urtasun. Visual semantic search: Retrieving videos via complex textual queries. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[17] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016.

[18] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 3111–3119. 2013.

[20] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2719–2726, 2012.

[21] L. D. Pero, J. Bowdish, B. Kermgard, E. Hartley, and K. Barnard. Understanding bayesian rooms using composite 3d object models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 153–160, June 2013.

[22] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39 – 62, 1999.

[23] S. Schuster, R. Krishna, A. Chang, L. Fei-Fei, and C. D. Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *4th Workshop on Vision and Language*, pages 70–80, 2015.

[24] B. Siddiquie, B. White, A. Sharma, and L. S. Davis. Multimodal image retrieval for complex queries using small codes. In *International Conference on Multimedia Retrieval (ICMR)*, 2014.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[26] J. M. Snyder. Interval analysis for computer graphics. *SIGGRAPH Comput. Graph.*, 26(2):121–130, July 1992.

[27] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research (JMLR)*, 15:2949–2980, 2014.

[29] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems 28 (NIPS)*. 2015.

[30] Y. Xiang and S. Savarese. Object detection by 3d aspectlets and occlusion reasoning. In *International Conference on Computer Vision Workshops (ICCVW)*, Dec 2013.

[31] M. Z. Zia, M. Stark, and K. Schindler. Towards scene understanding with detailed 3d object representations. *International Journal of Computer Vision*, 112(2):188–203, 2015.

[32] C. L. Zitnick, D. Parikh, and L. Vanderwende. Learning the visual interpretation of sentences. In *International Conference on Computer Vision (ICCV)*, pages 1681–1688, 2013.