Building a database of 3D scenes from user annotations

Building a database of 3D scenes from user annotations

Bryan C. Russell
INRIA*
russell@di.ens.fr

Antonio Torralba
CSAIL MIT
torralba@csail.mit.edu

Abstract

In this paper, we wish to build a high quality database of images depicting scenes, along with their real-world three-dimensional (3D) coordinates. Such a database is useful for a variety of applications, including training systems for object detection and validation of 3D output. We build such a database from images that have been annotated with only the identity of objects and their spatial extent in images. Important for this task is the recovery of geometric information that is implicit in the object labels, such as qualtitative relationships between objects (attachment, support, occlusion) and quantitative ones (inferring camera parameters). We describe a model that integrates cues extracted from the object labels to infer the implicit geometric information. We show that we are able to obtain high quality 3D information by evaluating the proposed approach on a database obtained with a laser range scanner. Finally, given the database of 3D scenes, we show how it can find better scene matches for an unlabeled image by expanding the database through viewpoint interpolation to unseen views.

1. Introduction

A database of images and their three-dimensional (3D) description would be useful for a number of tasks in computer vision. For example, such a database could be used to learn about how objects live in the world and train systems to detect them in images. Techniques for aligning images [10, 25, 20] may also benefit from such data. The database can be used to validate algorithms that output 3D. Furthermore, image content can be queried based on absolute attributes (e.g. tall, wide, narrow). Our goal is to create a large database of images depicting many different scene types and object classes, along with their underlying real-world 3D coordinates.

Of course, there are a variety of ways to gather such a dataset. For instance, datasets captured by range scanners or stereo cameras have been built [27, 28]. However, these datasets are relatively small or constrained to specific locations due to the lack of widespread use of such apparatuses. More importantly, by hand-collecting the data, it is difficult to obtain the same variety of images that can be found on the internet. One could undertake a massive data collection campaign (e.g. Google Street View [1]). While this can be a valuable source of data, it is at the same time quite expensive, with data gathering limited to one party.

Instead of manually gathering data, one could harness the vast amount of images available on the internet. For this to reasonably scale, reliable techniques for recovering absolute geometry must be employed. One approach is to learn directly the dependency of image brightness on depth from photographs registered with range data [27] or the orientation of major scene components, such as walls or ground surfaces, from a variety of image features [12, 13, 14]. While these techniques work well for a number of scenes, they are not accurate enough in practice since only low and mid level visual cues are used. An alternative approach is to use large collections of images available on the internet to produce 3D reconstructions [30]. While this line of research is promising, it is currently limited to specific locations having many image examples. There has recently been interesting work that produces some geometric information and requires fewer images of the same scene [11, 29, 7].

We would like to explore an alternate method for producing a 3D database by exploiting humans labeling on the internet. Recent examples of such collaborative labeling for related tasks include ESPgame [35], LabelMe [26], and Mechanical Turk [31]. In a similar manner, we could ask a human to provide explicit information about the absolute 3D coordinates of objects in a scene, such as labeling horizon lines, junctions, and edge types. However, it is often not intuitive as to which properties to label and how to label them. Furthermore, annotating is expensive and great care must be taken to scale to all of the images on the internet. The challenge is to develop an intuitive system for humans to label 3D that scales well to internet images.

We propose a system that produces high quality absolute 3D information from only labels about object class identity and their spatial extent in an image. In this way, we only require that humans provide labels of object names and their

*WILLOW project-team, Laboratoire d'Informatique de l'École Normale Supérieure ENS/INRIA/CNRS UMR 8548

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

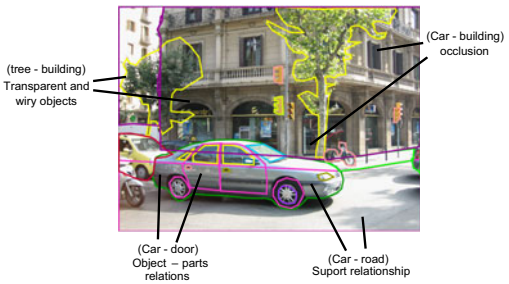


Figure 1. An image with object relations labeled. Support and part relations provide much information about object relations.

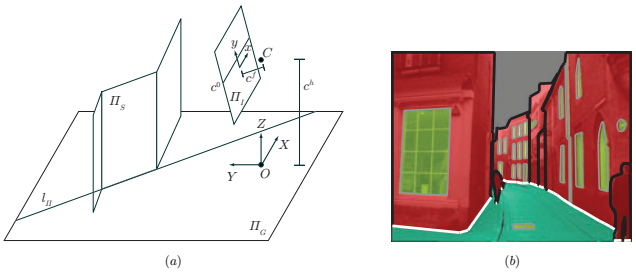


Figure 2. (a) Camera and scene layout. A scene is composed of objects, represented as piecewise-connected planes, standing on a ground plane. (b) Polygon and edge types. Polygons are either *ground* (green), *standing* (red), or *attached* (yellow). Edges are *contact* (white), *occluded* (black), or *attached* (gray).

location in an image. The advantages of this approach are twofold: (i) one needs to simply label the objects in a single image to get 3D, which is relatively cheap (especially with semi-automatic labeling methods [26, 19]), (ii) labeling static objects is in general fairly intuitive.

At first glance it seems impossible to infer 3D information from object labels alone. However, there is rich geometric information implicitly captured in the collection of object labels. For example, consider the labeled image depicting a scene shown in Figure 1. Notice that cues important for the recovery of geometry, such as attachment, support, and occlusion relationships holding between objects, are manifested in the spatial configuration of the polygons. For instance, overlapping polygons correspond to attachment or occlusion and adjacent edges from abutting polygons correspond to contact, occlusion, or attachment. We wish to develop cues to detect these relationships from the annotations provided for an image.

An important issue is that cues extracted from the polygons are ambiguous and may correspond to multiple geometric relationships. To disambiguate these relationships, we can leverage the fact that many of them are labeled consistently across the database for object class instances. This holds because of the regularity in the spatial configuration of objects in a scene. For example, when we see a person against the side of a building, we know that the person is not physically attached to the building since people also appear outside the spatial extent of buildings. In contrast, we know that windows do not rest against buildings, but are attached to them, since windows tend to lie inside the spatial extent of buildings. This reasoning also applies to edges, as in the case of chimneys, where only the lower part of the boundary is attached to buildings. We wish to learn this information by analyzing the statistics of detected relationships across a labeled database and use it to infer 3D information.

There is a rich body of prior work that has looked into recovering these relationships in images, with early accounts described in the textbook of Ballard and Brown [2]. We also draw inspiration from early work in line-drawing analysis [4, 3, 16, 32], which can be seen as a precursor to work in this area. In our case, we have the advantage of a large database of labeled images to learn from. More recently, there has been work to develop systems for humans to explicitly label 3D information [15, 6, 22]. Also relevant are methods to recover geometric information from image information [27, 28, 14, 24, 34, 18, 9, 21, 37].

The main contribution of this paper is a high quality database of labeled images spanning many different scenes and objects, and their corresponding absolute 3D coordinates. To obtain such a database, we develop a model that integrates cues from the object labels across the database to infer geometric information, such as qualitative object relationships (attachment, support, occlusion) and quantitative ones (camera parameters). In addition, we show some applications where such a database is useful.

A number of databases have labeled objects and may be useful for our task [26, 36, 8]. For our work, we use the LabelMe database since it has labeled examples for many different scene types and object classes. Furthermore, the database grows over time since the labels are provided by humans on the internet.

2. Recovering 3D from user annotations

We wish to develop an integrated model to recover absolute 3D coordinates from user annotations. For this, we will integrate various cues that can be extracted from the annotations to infer the 3D coordinates of the labeled objects in the scene. In addition, we will leverage information about cues that are consistently observed across the dataset. Furthermore, the model will recover geometric information important for recovering the 3D coordinates, such as attachment and support relationships, where objects contact the ground, and distributions of absolute object size.

We make several assumptions about the camera and the geometric layout of a scene. For the camera we assume perspective projection, which is expressed as a 3×4 matrix $\mathbf{P} = (\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4})$ that relates world coordinates

$\mathbf{X} = (X, Y, Z, W)$ to image coordinates $\mathbf{x} = (x, y, w)$ (written as homogeneous vectors) via the transformation $\mathbf{x} = \mathbf{PX}$. As in [15, 6], we assume that a scene is composed of a number of objects standing on a ground plane (dubbed *standing objects*), with the objects represented as piecewise-connected planes oriented orthogonally to the ground plane. In addition, there may exist attached objects, whose 3D coordinates are derived from the objects that they are attached to. Figure 2(a) depicts our camera and scene layout assumptions.

To recover $\mathbf{P}$, we assume that the origin of the world coordinates $\mathbf{O}$ lies at the intersection of the ground plane and the line perpendicular to the ground plane that passes through the camera center $\mathbf{C} = (0, 0, c^h)$. The X and Y axes lie in the ground plane, with the Z axis oriented towards the direction of the camera center. Furthermore, we assume that there is no yaw or roll in the rotation of the camera relative to the world coordinates (there is only pitch). The pitch can be recovered from the camera focal length c^f, principal point (which we take to be the center of the image), and the image location of the horizon line of the ground plane (since there is no roll, the line can be parameterized by its y location in the image c^0). Together, $c = \{c^f, c^h, c^0\}$ are the intrinsic and extrinsic camera parameters for $\mathbf{P}$.

We can relate points on the image plane $\mathbf{\Pi_I}$ to the ground plane $\mathbf{\Pi_G}$ by the homography matrix $\mathbf{H} = (\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_4})$. For a plane $\mathbf{\Pi_S}$ orthogonal to $\mathbf{\Pi_G}$ (dubbed *standing plane*), let $\mathbf{l_\Pi}$ be the line intersecting the two planes, which projects to $\mathbf{l_\Pi}'$ in $\mathbf{\Pi_I}$ (the latter dubbed *contact line*). The two lines are related by $\mathbf{l_\Pi} = \mathbf{H}^{-T}\mathbf{l_\Pi}'$. Points on $\mathbf{\Pi_I}$ are related to $\mathbf{\Pi_S}$ by a homography $\tilde{\mathbf{H}}$, which can be computed by choosing a new coordinate system with origin $\tilde{\mathbf{O}}$ that lies on $\mathbf{l_\Pi}$. Since the relationship between $\mathbf{O}$ and $\tilde{\mathbf{O}}$ is known, we can relate points in the new coordinate system to the world coordinates.

Since standing objects are represented by a set of piecewise-connected standing planes, let $\{\mathbf{l_1}, \ldots, \mathbf{l_D}\}$ be the set of contact lines corresponding to the standing planes and $\mathbf{s} = \{\mathbf{s_1}, \ldots, \mathbf{s_D}\}$ the corresponding set of piecewise-connected line segments and rays that mark the valid extent of the contact lines for an object. We restrict $\mathbf{s}$ so that any line passing through the vanishing point $\mathbf{v}$ corresponding to the Z axis intersects with at most one line segment or ray (i.e. we do not model object self-occlusion). As an approximation, we assume that $\mathbf{v}$ lies at infinity, which is reasonable for standard cameras with the horizon line lying within the image. For image point $\mathbf{x}$, we use the contact line corresponding to the line segment or ray intersecting the x coordinate of the image point.

For an annotation, a user provides an object class label o and a polygon that marks the spatial extent of the object in the image. For the polygon, let $p = (\mathbf{x}_1, \cdots, \mathbf{x}_K)$ be the set of image points, $e_k = (\mathbf{x}_k, \mathbf{x}_{k+1})$ the edge connecting adjacent points, and $e = (e_1, \cdots, e_{K-1})$ the set of edges. Given the scene layout assumptions, a polygon must be either on the ground plane, a standing object, or attached to another object. Let $q \in \{ground, standing, 1, \cdots, M\}$ be the polygon type, where the index of another polygon is provided if the polygon is attached. We assume that edges must be one of the following types: $r_k \in \{contact, occlusion, attached\}$. Finally, we assume that ground and attached objects have attached edges only. Figure 2(b) illustrates the polygon and edge types.

We define the function $f(\mathbf{x}, j, e, q, r, c)$ that takes as input the edges, polygon types, and edge types for all polygons in the image, along with the camera parameters, and computes the 3D coordinates for an image point $\mathbf{x}$ on polygon j. For attached objects, we compute the 3D coordinates from the object it is attached to. For standing objects, the set of piecewise-connected line segments and rays $\mathbf{s}$ are derived from the contact edges.

For images $i = 1, \ldots, N$ and model parameters $\theta, \phi, \alpha, \beta, \eta, \vartheta$, we compute the MAP estimate of the camera parameters and the polygon and edge types:

$$\arg\max_{c_i, q_i, r_i} \prod_{j=1}^{M_i} P(q_{i,j}|o_i, s_{i,j}, \theta, \alpha, \eta) \prod_{k=1}^{L_{i,j}} P(r_{i,j,k}|t_{i,j,k}, q_{i,j}, \beta)$$
$$P(c_i|o_i, u_{i,j}, r_{i,j,k}, \phi, \vartheta) \prod_{(k,l,m)} \psi(r_{i,j,k}, r_{i,j,l}, r_{i,j,m})$$

where $s_{i,j}$, $t_{i,j,k}$, and $u_{i,j}$ are cues derived from the polygons in the image. The rest of this section will describe how the cues are extracted, the precise form of the probability distributions, and how the model parameters are learned.

2.1. Recovering polygon types

Critical to the recovery of a polygon's type is its relationship to other polygons in the same image. Two relationships we consider are (i) when an object is attached to another object and (ii) when a ground object supports a standing object. We describe a model for these two relationships.

Discovering attachment relationships: Given the database of N labeled images, each having M_i polygons, we wish to recover information about which objects are *attached* to (i.e. part of) other objects in an image. To recover attachment relationships, we assume the following: (i) the spatial extent of an attached object mostly lies inside the object it is attached to; (ii) the relationship is consistently observed across the entire database when instances of the two object classes appear in the same image. When an attachment relationship holds in an image and users label one polygon inside the other, they are implicitly labeling the relationship. Therefore, critical to finding attachment

relationships is reasoning about how polygons overlap across the database.

However, we must take care when analyzing overlapping polygons in the LabelMe database. In particular, two polygons may also overlap when there is an occlusion and the user labels behind the occlusion. We need to detect when these cases arise. For this, we assume that polygon overlap due to occlusion is more likely to occur for objects that occupy a large area of the image.

To measure whether polygon A lies inside polygon B, we compute the relative overlap between the two polygons $R_{A,B} = \frac{\text{area}(A \cap B)}{\text{area}(A)}$, which is the intersection area normalized by the area of A. If A lies completely inside B, then $R_{A,B} = 1$ (conversely $R_{A,B} = 0$ if A lies outside of B). We also measure the relative area of a polygon as $a_A = \frac{\text{area}(A)}{\text{area}(Image)}$, which is the area of the polygon normalized by the image area.

Suppose for polygon j in image i we observe $s_{i,j} = (R_{i,j}, a_i)$, where $R_{i,j} = (R_{i,j,1}, \cdots, R_{i,j,M_i})$ is a vector containing the relative overlap between polygon j and the other polygons in the image and $a_i = (a_{i,1}, \cdots, a_{i,M_i})$ is a vector of the relative polygon areas. We assume the following generative model: $P(s_{i,j}|o_i, \alpha, \theta) = \sum_{q_{i,j} \in \xi \setminus \xi_j} P(s_{i,j}|q_{i,j}, \alpha) P(q_{i,j}|o_i, \theta)$. The first term is the likelihood of the polygon overlap and area measurements given $q_{i,j}$, which indicates the presence or absence of an attachment relationship between polygon j and the other polygons in the image. The second term is the likelihood of an attachment relationship holding between instances of two object classes (e.g. *windows* and *cars*), and is learned across the database.

More specifically, $q_{i,j} = (q_{i,j,0}, \cdots, q_{i,j,M_i})$ is a latent binary vector indicating which polygon in the image, if any, that j is attached to. For example, if $p \in \{1, \ldots, M_i\}$, then $q_{i,j,p} = 1$ indicates that j is a part of polygon p. Conversely, if $p = 0$, then polygon j is not attached to any other polygon. We assume that j can be attached to at most one polygon and not to itself (i.e. $q_{i,j} \in \xi \setminus \xi_j$, where ξ is the standard basis).

We assume that the data likelihood factorizes into the product of two terms $P(s_{i,j}|q_{i,j}, \alpha) = P(R_{i,j}|q_{i,j}, \alpha_R) P(a_i|q_{i,j}, \alpha_a)$, with the first term modeling the likelihood of relative overlap given attachment and the second as modeling the likelihood of occlusion given no attachment. These terms further decompose into the product of terms for each component. We assume that $R_{i,j,p} \sim Beta(\alpha_R, 1)$ and $a_{i,p} \sim Beta(\alpha_a, 1)$. The parameters α_R and α_a are set so that (i) relative overlap must be close to one for attachment relationships and (ii) overlap due to occlusion is biased toward zero.

To determine the likelihood of which polygon j is attached to (if any), we rely on the frequency of attachment across the database of other polygons belonging to the same object class. For j belonging to object class $o_{i,j} = l$, we use a decision tree to decide which polygon it is attached to. First, decide if object class l is attached to anything, parameterized by $\theta_{l,0}$. If so, then decide which object class m it is most likely attached to, parameterized by $\theta_{l,m}$. If there are multiple examples of object class m in the image, then choose among them with equal probability.

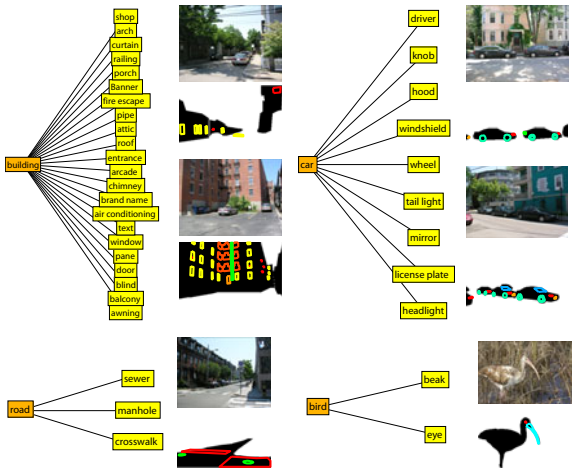


Figure 3. Recovered attachment relationships. We show graphs of several object classes (building, car, road and bird) from the LabelMe dataset, along with objects that are likely to be attached to them. The images to the right of each graph show examples of images and the object polygons that are found to be parts of the parent object class.

Concretely, we assume that $q_{i,j} \sim Multinomial(\bar{\theta})$, where $\bar{\theta}_0 = \theta_{l,0}$ and $\bar{\theta}_p = \frac{(1-\theta_{l,0})\theta_{l,m}}{N_{m \setminus j}}$ for $p \neq 0$ and $o_p = m$. Here, $N_{m \setminus j}$ is the number of instances of object class m in the image, excluding polygon j. Notice that with this formulation, we handle attachment detection and assignment. We also handle multiple instances of the same object and polysemy (e.g. a "window" can be attached to a "car" or a "building" in an image).

We learn the parameters for attachment likelihood via Gibbs sampling. We sample over the posterior distribution of $q_{i,j}$ given attachment assignments for all the other polygons in the database. We assume a $Beta/Dirichlet$ prior over θ, and set the parameters as $\eta_0 = 5$, $\eta_l = 0.5$, $\alpha_a = 2$, and $\alpha_R = 20$.

Figure 3 shows attachment relationships that our model recovers by thresholding θ. The left side shows a graph of objects that our model deems as possible parts of an object. The right side shows objects in an image their automatically recovered parts. Notice that we recover attachment relationships for a variety of object classes.

Discovering support relationships: We assume that objects in a scene are supported by other objects (e.g. road,

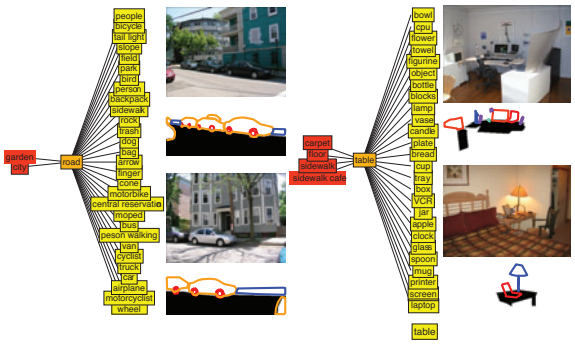


Figure 4. Recovered support relationships. We show graphs for example object classes (road and table) that support other objects. The images to the right of each graph show the polygonal boundaries of the objects being supported.

sidewalk, floor, table). We would like to automatically recover these support relationships. This is important since we will need to reason about the geometry of objects through the objects that support them. As for attachment, we would like to analyze the frequency of support relationships across the database.

We assume that a support relationship occurs when the bottom of an object touches (makes contact with) its object of support. Users implicitly label this when they draw a polygon where the bottom touches a labeled support object. However, attached objects also satisfy this condition. We must first infer attachment and remove those objects from consideration.

To recover support relationships, first remove attached objects. Then, for pairs of instances from two object classes in an image, count the number of pairs where the bottom of one instance lies inside the other. Finally, normalize this count by the total number of observed pairs.

In Figure 4, we show object classes likely to be in support relationships. We show relationships with normalized score exceeding 0.5. Notice that we recover many different relations spanning indoor and outdoor scenes.

2.2. Recovering edge types

We employ a variety of cues for recovering the different edge types for a polygon. If a polygon is attached to another polygon, then we assume that all of its edges are attached. Edges close to ground objects are likely to be contact edges. Also, edges that are long, relative to the width of the object, and horizontally oriented are more likely to be contact edges. We also assume that contact edges cannot have another edge below it from the same polygon. All other edges are assumed to be occlusion edges.

We describe how to extract edge cues for edge k belonging to polygon j in image i. To measure if an edge is close to a support object, we compute the mean distance $d_{i,j,k}$ from evenly sampled points along the edge to the support object and normalize the distance with respect to the image width. The edge length $l_{i,j,k}$ is computed relative to the object width and is clipped to lie in $[0, 1]$. The edge orientation $v_{i,j,k}$ is measured with respect to a horizontal line and is normalized to lie in $[0, 1]$. Finally, we set $b_{i,j,k} = 1$ if there is no other edge below k. We collect all of these observations into an observation vector $t_{i,j,k}$.

We wish to infer the latent edge label $r_{i,j,k}$ as attached, contact, or occlusion for edge k. We assume that the different edge cues are distributed independently given the edge label. For attached and occlusion edge labels, we assume that the cues are distributed uniformly. For contact edge labels, we assume that $d_{i,j,k} \sim Beta(\beta_d, 1)$, $l_{i,j,k} \sim Beta(\beta_l, 1)$, $v_{i,j,k} \sim Beta(\beta_v, 1)$, and $b_{i,j,k} \sim \delta(1)$. We assume the following prior probabilities for the different edge types given the attachment relationship $q_{i,j}$: $P(r_{i,j,k} = \text{att} | q_{i,j} = \xi_0) = 1$, $P(r_{i,j,k} = \text{cont} | q_{i,j} \neq \xi_0) = \rho$, $P(r_{i,j,k} = \text{occ} | q_{i,j} \neq \xi_0) = 1 - \rho$.

2.3. Incorporating edge constraints

We wish to exploit additional information about groups of edges. For example, this may come in the form of Gestalt cues (c.f. a model for exploiting colinear edges to detect crosswalks in images [5]). In this work, we explore incorporating edge constraints to more robustly handle objects that do not make flush contact with the ground, but at isolated points (e.g. car wheels, chair and table legs). Often, non-contact edges along the bottom of these objects lie in close proximity to a support polygon because they live well below the camera height. For these edges, local cues alone are not sufficient to disambiguate their type.

We use information about neighboring edges to resolve the edge type ambiguity. For edge l, if its two neighboring edges are below it and occlusion edges, then most likely l is not a contact edge. This is a powerful constraint since vertical edges (e.g. along the leg of a table) are likely to be occlusion edges, thus causing the horizontal edge along the bottom of the table to be labeled as an occlusion edge.

We model the above constraint by first detecting edges $r_{i,j,l}$ with neighbors $r_{i,j,k}$ and $r_{i,j,m}$ below it. We define a ternary potential function $\psi(r_{i,j,k}, r_{i,j,l}, r_{i,j,m})$ for the edge triple. ψ has unit energy for all edge configurations except when edge k and m are occlusion edges. If l is a contact edge, the energy is ν_c (ν_o if l is an occlusion edge). We set $\nu_c = 0.01$ and $\nu_o = 10$.

2.4. Estimating camera parameters

To estimate the camera parameters, we follow the procedure outlined in [17], which uses information about the contact location and polygon height $u_{i,j} = (v_{i,j}, h_{i,j})$ in the image. The method estimates the horizon line c^0 and the camera height c^h by assuming that the polygon

height is Gaussian distributed $h_{i,j} \sim \mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$ where $\bar{\mu} = \frac{\mu_l(c^0 - v_{i,j})}{c^h}$ and $\bar{\sigma} = \frac{\sigma_l(c^0 - v_{i,j})}{c^h}$. Here, $o_{i,j} = l$ is the object class of the polygon, and $\phi_l = (\mu_l, \sigma_l)$ is the mean and standard deviation of the 3D height for the object class. We only use polygons for which we recovered contact edges and assume a fixed focal length of $c^f = 800$ pixels.

To the learn the parameters for object heights, we use the procedure outlined in [17], where the mean and standard deviation of the height of a person is used to find initial estimates of the camera parameters. These parameters are then, in turn, used to estimate the object heights. Objects with reliable height estimates are then used to refine the camera parameters. This process is repeated until a stable solution is found. Finally, we use a prior on the camera height, with mean 1.7 meters and standard deviation 0.5 meters.

2.5. Inference

We perform inference to recover the edge labels and camera parameters by marginalizing over the attachment variables $q_{i,j}$. Since there are loops in the model, we use sum-product loopy belief propagation, which converges within 10 iterations (the entire system run within a few seconds per image). We restrict our 3D scene models to one level of the recovered support/part graph (i.e. we do not render items on a table that is supported by the floor). Furthermore, we ignore objects for which we do not infer any contact edges, along with portions of ground objects that reside above the recovered horizon line. We assume that objects with a single short contact edge (edge length less than 5% of the image width) are frontal-parallel, with the contact line parameterized by the lowest polygon point.

3. Evaluation

In this section, we evaluate various components of our system. We first evaluate inference for labeling polygon edges. We then show qualitative and quantitative results of the overall system.

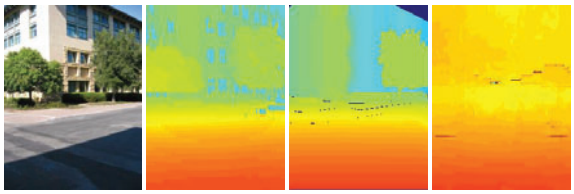
To evaluate edge classification, we manually labeled edges corresponding to the polygonal annotations of 13 images depicting different views of street scenes. This corresponded to 8621 edges over 854 polygons, spanning 61 object classes. As a result, there were 313 contact, 3198 attached, and 5110 occluded edges.

Figure 2(b) shows the inferred edge types, as determined by our system. Notice that our output agrees well with the image data. Also, we handle different object classes having different number of planar facets and being in different occlusion and attachment relationships.

Table 1 shows a confusion matrix for the edge classification. The columns show for each edge type the fraction of ground truth edges that were classified into each type. Perfect performance corresponds to the identity matrix. Notice that we achieve high accuracy.

	Contact	Attached	Occlusion
Contact	0.58	0.00	0.03
Attached	0.24	0.97	0.22
Occlusion	0.19	0.03	0.76

Table 1. Confusion matrix for edge classification. Each column shows for an edge type the fraction of ground truth edges that were classified into each edge type. The average confusion is 0.77.



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

Figure 6. Depth estimates on a dataset gathered with a laser range scanner [27]. (a) Input image. (b) Range scanner depth map. (c) System output depth maps. (d) Baseline depth map, which is the harmonic mean of the depth maps in the training set.

We demonstrate our full system on a variety of images depicting many different scenes and object categories and spanning different depths and scene viewpoints. Figure 5 shows pairs of rows depicting input images, along with their object labels, and output depth maps from our system. A legend for the absolute depths (in meters, plotted in log-scale) is shown at the top-right.

To quantitatively assess the accuracy of our depth maps, we used a ground-truth dataset gathered with a laser range scanner [27]. This dataset consists of 400 images taken on the Stanford campus, with a subset having associated depth-maps. We provided dense object annotations for 62 of these images. The images are 256x192 resolution and depict outdoor scenes. We measured depth accuracy using the mean per-pixel relative error across all pixels in the test set. To handle systematic bias in our depth outputs, we found a linear regressor that minimized the relative error of our system depth outputs to the ground truth depth values. We performed cross validation, using 20 images for training and the rest for testing. Due to noise in the range data, we only considered depths in the range of 5-70 meters in the ground truth and system output depths.

Figure 6 shows example images from the test set along with the ground truth, baseline, and system output depth maps. Notice that the estimated depths from our system are close to those produced from the range scanner. Our system has relative error of 0.29 ± 0.02. As a baseline, we compared against the harmonic mean of the depth maps corresponding to the training images. The baseline has relative error of 0.33 ± 0.04. We considered $40\% \pm 2\%$ of the pixels for the evaluation. Notice that we achieve improved performance over the baseline.

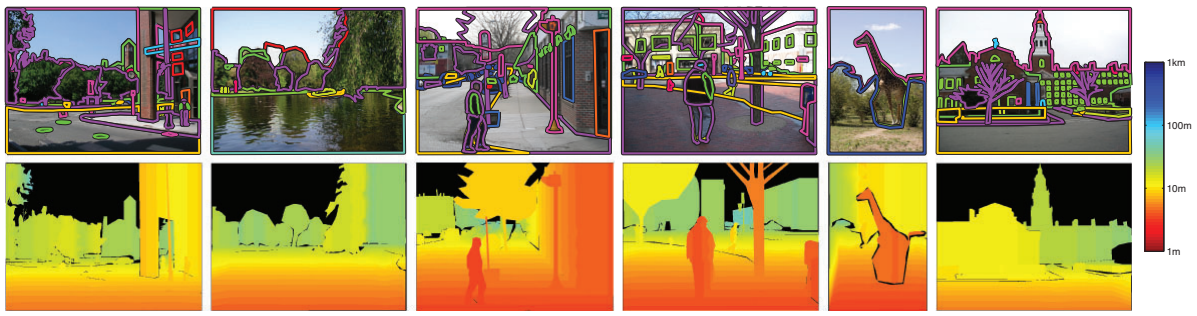


Figure 5. System outputs. Top row: input image and object labels. Bottom row: depth maps (in meters) produced by our system, with a legend for the depths (in log-scale) shown at right. Our system handles a variety of scenes and objects with different depths and viewpoints.

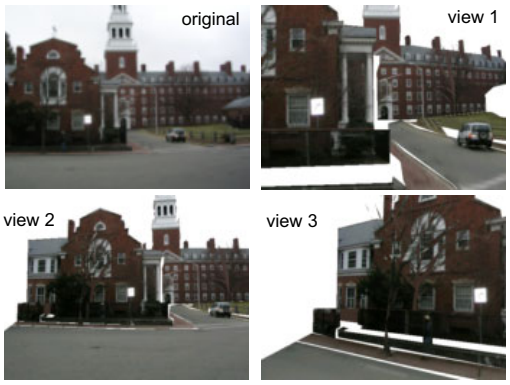


Figure 7. Examples of new images obtained by simulating a person walking in the scene and taking pictures.

4. Extending a database with virtual views

In object detection, a typical strategy to improve training is to generate new virtual examples obtained by randomly perturbing the training examples. This works especially well for faces when it is possible to build accurate generative models [33].

In the case of image retrieval, this task is challenging since the set of possible image transformations is large and difficult to model. One simple solution is to generate new images by randomly cropping each picture in the database. However, this strategy produces images with different statistics than normal pictures taken by a person standing on the ground.

If a database has 3D information, then a better way of extending it is by simulating a person walking on the ground and taking pictures in random orientations (but keeping the camera parallel to the ground plane). The right column of Figure 7 shows three images (matched in size with the left column) generated by a random walk on the ground plane.

We used 1600 fully annotated images depicting street scenes to generate a new set of 6000 annotated images by generating 20 new images for each image and removing those with less than 40% valid pixels. We used the new set

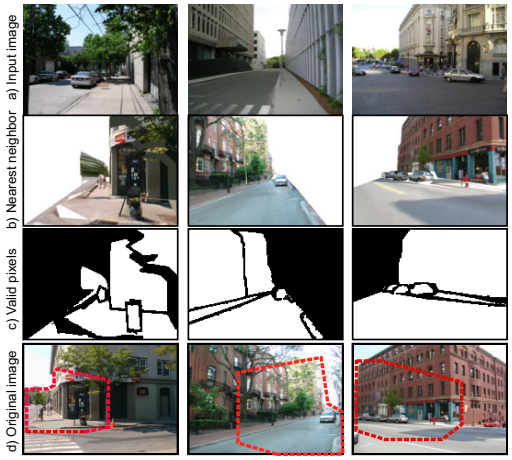


Figure 8. Matching unlabeled images to our expanded database with virtual views generated from the recovered 3D. (a) Input images. (b) First nearest neighbor match from the expanded database. (c) The set of valid pixels in the match. (d) The original location of the virtual image.

to perform queries for an input image. To compute image similarity, we used the gist descriptor [23]. For the synthetic images, we keep a list of the valid pixels and only use those to compute the gist descriptor.

Figure 8 shows (a) three different input images, (b) the first nearest neighbor for each query found using Euclidean distance, (c) the set of valid pixels for each neighbor, and (d) the original image used to synthesize the virtual image and the region seen by the virtual camera that best matches the input image. Notice that we obtain database matches that are similar to the query image.

5. Conclusion

We have shown how to build a high quality database of images depicting scenes with absolute 3D information from object labels alone. We formulated an integrated model to infer the geometric information implicit in the object labels.

We showed that we can achieve a low relative error rate on the benchmark database obtained with a laser range scanner [27]. Finally, we showed one application of the database to improve scene matches for an unlabeled image.

Even with object detection and segmentation information, we had to make a number of assumptions about the scene geometry. While our results hold for a wide variety of images, there are still a number of labeled images for which we cannot extract reliable 3D information. We believe that this, and other applications of our database, yields promising directions for future research.

6. Acknowledgements

We thank Jean Ponce and Josef Sivic for helpful feedback. Funding for this research was provided by National Science Foundation Career award (IIS 0747120).

References

[1] http://www.maps.google.com. 1

[2] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982. 2

[3] H. Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26. Academic Press, N.Y., 1978. 2

[4] M. Clowes. On seeing things. *Artificial Intelligence Journal*, 2(1):79–116, 1971. 2

[5] J. Coughlan and H. Shen. A fast algorithm for finding crosswalks using figure-ground segmentation. In *2nd Workshop on Applications of Computer Vision, in conjunction with ECCV*, 2006. 5

[6] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *IJCV*, 40(2):123–148, 2000. 2, 3

[7] S. K. Divvala, A. A. Efros, and M. Hebert. Can similar scenes help surface layout estimation? In *IEEE Workshop on Internet Vision, associated with CVPR*, 2008. 1

[8] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual object classes challenge 2006 (VOC 2006) results. Technical report, September 2006. 2

[9] A. Gupta and L. S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*, 2008. 2

[10] J. Hays and A. Efros. Scene completion using millions of photographs. In *"SIGGRAPH"*, 2007. 1

[11] J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008. 1

[12] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005. 1

[13] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005. 1

[14] D. Hoiem, A. Stein, A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007. 1, 2

[15] Y. Horry, K.-I. Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. *SIGGRAPH*, pages 225–232, 1997. 2, 3

[16] D. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8:493–509, 1977. 2

[17] J. F. Lalonde, D. Hoiem, A. Efros, J. Winn, C. Rother, and A. Criminisi. Photo clip art. In *SIGGRAPH*, 2007. 5, 6

[18] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR*, 2007. 2

[19] L.-J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: automatic object picture collection via incremental model learning. In *CVPR*, 2007. 2

[20] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT flow: dense correspondence across different scenes. In *ECCV*, 2008. 1

[21] V. Nedovic, A. Smeulders, A. Redert, and J.-M. Geusebroek. Depth information by stage classification. In *ICCV*, 2007. 2

[22] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. *SIGGRAPH 01*, 2001. 2

[23] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 7

[24] X. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, 2006. 2

[25] B. C. Russell, A. Torralba, C. Liu, R. Fergus, and W. T. Freeman. Object recognition by scene alignment. In *Advances in Neural Info. Proc. Systems*, 2007. 1

[26] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008. 1, 2

[27] A. Saxena, S. Chung, and A. Ng. Learning depth from single monocular images. In *NIPS*, 2005. 1, 2, 6, 8

[28] A. Saxena, M. Sun, and A. Ng. Learning 3-d scene structure from a single still image. In *ICCV workshop on 3D Representation for Recognition*, 2007. 1, 2

[29] J. Sivic, B. Kaneva, A. Torralba, S. Avidan, and W. T. Freeman. Creating and exploring a large photorealistic virtual space. In *First IEEE Workshop on Internet Vision, associated with CVPR*, 2008. 1

[30] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *SIGGRAPH*, 2006. 1

[31] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *IEEE Workshop on Internet Vision, associated with CVPR*, 2008. 1

[32] K. Sugihara. An algebraic approach to the shape-from-image-problem. *Artificial Intelligence Journal*, 23:59–95, 1984. 2

[33] N. P. H. Thian, S. M., and S. Bengio. Improving face authentication using virtual samples. In *ICASSP*, 2003. 7

[34] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, and L. V. Gool. Depth-from-recognition: Inferring meta-data by cognitive feedback. In *ICCV Workshop on 3d Representation for Recognition*, 2007. 2

[35] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI*, 2004. 1

[36] B. Yao, X. Yang, and S.-C. Zhu. Introduction to a large scale general purpose ground truth dataset: methodology, annotation tool, and benchmarks. In *EMMCVPR*, 2007. 2

[37] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. Single view modeling of free-form scenes. In *CVPR*, 2001. 2