LabelMe: online image annotation and applications

Antonio Torralba, Bryan C. Russell, and Jenny Yuen

***Abstract*—Central to the development of computer vision systems is the collection and use of annotated images spanning our visual world. Annotations may include information about the identity, spatial extent, and viewpoint of the objects present in a depicted scene. Such a database is useful for the training and evaluation of computer vision systems. Motivated by the availability of images on the Internet, we introduced a web-based annotation tool that allows online users to label objects and their spatial extent in images. To date, we have collected over 400K annotations that span a variety of different scene and object classes. In this paper, we show the contents of the database, its growth over time, and statistics of its usage. In addition, we explore and survey applications of the database in the areas of computer vision and computer graphics. Particularly, we show how to extract the real-world 3D coordinates of images in a variety of scenes using only the user-provided object annotations. The output 3D information is comparable to the quality produced by a laser range scanner. We also characterize the space of the images in the database by analyzing (i) statistics of the co-occurrence of large objects in the images and (ii) the spatial layout of the labeled images.**

***Index Terms*—online annotation tool, image database, object recognition, object detection, 3D, video annotation, image statistics**

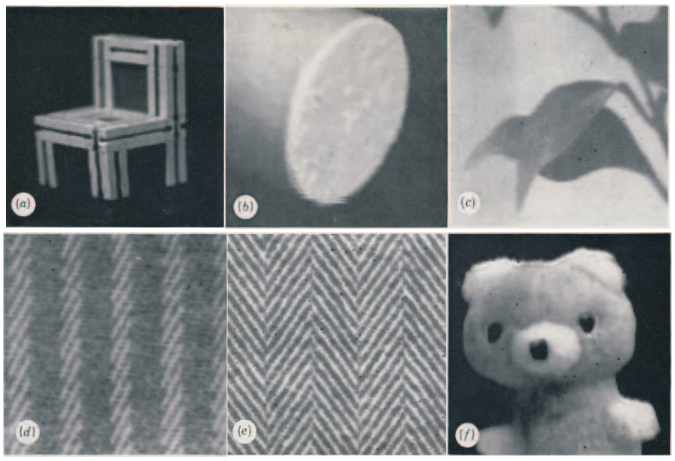

Fig. 1. The entire dataset from an early vision paper [36]. The original caption illustrates the technical difficulties of image digitization in the 1970s: "(a) and (b) were taken with a considerably modified Information International Incorporated Vidissector, and the rest were taken with a Telemation TMC-2100 vidicon camera attached to a Spatial Data Systems digitizer (Camera Eye 108)." quote from [37]

I. Introduction

In the early days of computer vision research, the first challenge a computer vision researcher would encounter would be the difficult task of digitizing a photograph [25]. Figure 1 shows the complete image collection used for the study presented in [37] and illustrates the technical difficulties existing in the 70's to capture digital images. Even once with a picture in digital form, storing a large number of pictures (say six) consumed most of the available computational resources.

Today, having access to large datasets is fundamental for computer vision. Small datasets have the risk of over-fitting by encouraging approaches that work only on a few selected cases. Moreover, it is hard to evaluate progress in the field with small datasets. This is specially relevant in the case of object recognition, where it is important to test the ability of a model to detect and recognize objects under a variety of conditions (different backgrounds, illuminations, occlusions, and poses). For this, we require access to large collections of annotated images covering the variability of the visual world.

The availability of visual data has experienced a dramatic change in the last decade, especially via the Internet, which has given researchers access to billions of images and videos. While large volumes of pictures are available, building a large dataset of annotated images with many objects still constitutes a costly and lengthy endeavor. Traditionally, datasets are built by individual research groups and are tailored to solve specific problems. Therefore, many currently available datasets used in computer vision only contain a small number of object classes, and reliable detectors exist for a few of them (*e.g.* human faces and cars [75], [47], [54], [74]). Notable recent exceptions are the Caltech 101 dataset [14], with 101 object classes (later extended to 256 object classes [18]), ImageNet [8], the PASCAL collection [11] containing 20 object classes, the CBCL-street scenes database [5], comprising 8 object categories in street scenes, and the database of scenes from the Lotus Hill Research Institute [82]. For a review and discussion of current data sets we refer to [41]. The goal of LabelMe is to provide a large variety of images with many annotated objects. Available datasets in computer vision have focused on getting annotated data for a set of predefined object categories [11], [5] or in collecting images of one or few prominent objects [14], [18].

Creating a large number of annotations for thousands of different object classes can become a time-consuming and challenging process. To cope with the difficulty of creating a large, annotated dataset, there have been several works that study methods for optimizing labeling tasks. For example, given enough annotations for a particular object class, one can train an algorithm to assist the labeling process. The algorithm would detect and segment additional instances in new images and be followed by a user-assisted validation stage [76]. An implementation of this idea is the Seville project [1], where an

A. Torralba and J. Yuen are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA e-mail: torralba@csail.mit.edu, jenny@csail.mit.edu.

B.C. Russell is with INRIA, WILLOW project-team, Laboratoire d'Informatique de l'École Normale Supérieure ENS/INRIA/CNRS UMR 8548, Paris, France e-mail: russell@di.ens.fr.

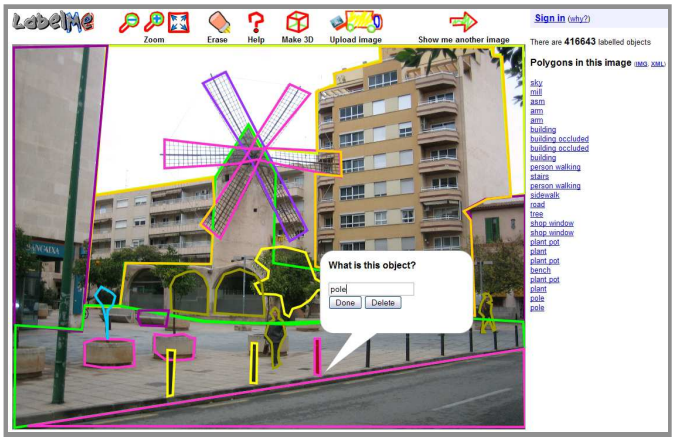

Fig. 2. Snapshot of the online application for image annotation.

incremental, boosting-based detector was trained. The pipeline begins by training a coarse object detector that is good enough to simplify the collection of additional examples. Furthermore, the user provides feedback to the system by indicating when an output bounding box is a correct detection or a false alarm. Finally, the detector is retrained with the enlarged dataset. This process is repeated until reaching the desired number of labeled images. Another work for optimizing label propagation is [77], where a learner is trained to balance the relative costs for obtaining different levels of annotation detail, along with the reduction of uncertainty the annotation provides to the system. A complementary line of research tries to avoid the need to annotate images by developing unsupervised learning algorithms [13], [65], [81], [80], [15], [59], [49], [68], [16]. These works are characterized by creating learners to recognize and distinguish object classes that can be trained with unlabeled and unsegmented scenes. However, independent of the methods for creating classifiers, ground truth data is always implicitly necessary to validate inferred annotations and to assign names to discovered object categories.

Web-based annotation tools provide a means of building large annotated datasets by relying on the collaborative effort of a large population of users [78], [56], [51], [62], [64]. Recently, such efforts have shown to be successful. The Open Mind Initiative [64] aims to collect large datasets from web users to develop intelligent algorithms. More specifically, common sense facts are recorded (e.g. red is a primary color), with over 700K facts recorded to date. This project seeks to extend their dataset with speech and handwriting data. Flickr [56] is a commercial effort to provide an online image storage and organization service. Users often provide textual tags as captions for depicted objects in an image. Another way lots of data has been collected is through an online game that is played by many users. The ESPgame [78] pairs two random online users who view the same target image. The goal is for them to try to "read each other's mind" and agree on an appropriate name for the target image as quickly as possible. This effort has collected over 10 million image captions since 2003 for images randomly drawn from the web. While the amount of collected data is impressive, only caption data is acquired. Another game, Peekaboom [79], has been created to provide location information of objects. Recently, a number of collection efforts have used online services such as Mechanical Turk [62] to distribute the annotation task to a large population of Internet users.

In 2005 we created LabelMe [51], an online annotation tool that allows sharing and labeling of images for computer vision research. The application exploits the capacity of the web to concentrate the efforts of a large population of users. The tool has been online since August 2005 and has accumulated over 400,000 annotated objects. The online tool provides functionalities for drawing polygons to outline the spatial extent of object in images, querying for object annotations, and browsing the database (see Fig. 2).

In this paper we describe the evolution of both LabelMe and its annotation corpus. We demonstrate statistics validating the ease of use and impact our system has had over the course of time. With the aid of collaborative collection and labeling of scenes at a large scale, we visualize a two-dimensional semantic layout of the labeled real-world scenes. Finally, we demonstrate applications of our rich database. For example, we developed a method to learn concepts not explicitly annotated in scenes, such as support and part-of relationships, which allows us to infer 3D information of scenes.

II. Web Annotation and Data Statistics

The following is a brief summary of the overall project goals and main features that distinguishes the LabelMe database from other databases.

- Designed for object class recognition as opposed to instance recognition. To recognize an object class, one needs multiple images of different instances of the same class, as well as different viewing conditions. Many databases, however, only contain different instances in a canonical pose.
- Designed for learning about objects embedded in a scene. Many databases consist of small cropped images of object instances. These are suitable for training patch-based object detectors (such as sliding window classifiers), but cannot be used for training detectors that exploit contextual cues.
- High quality labeling. Many databases just provide captions, which specify that the object is present somewhere in the image. However, more detailed information, such as bounding boxes, polygons or segmentation masks, is tremendously helpful.
- Many diverse object classes. Many databases only contain a small number of classes, such as faces, pedestrians and cars (notable exceptions are the Caltech 101, Caltech 256, and ImageNet databases).
- Many diverse images. For many applications, it is useful to vary the scene type (e.g. nature, street, and office scenes), distances (e.g. landscape and close-up shots), degree of clutter, etc.
- Many non-copyrighted images. For the LabelMe database most of the images were taken by the authors of this paper using a variety of hand-held digital cameras. Also, many images were contributed by various researchers seeking to label their images.

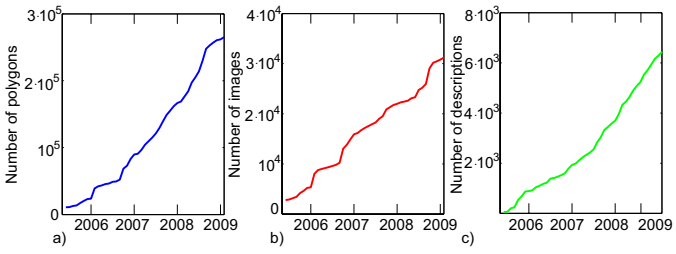

Fig. 3. Evolution of the dataset since public launch of the annotation tool in August 2005 through 2009. The horizontal axis denotes time (each mark is the beginning of the year), and the vertical axis represents: a) number of annotated objects, b) number of images with at least one annotated object, c) number of unique object descriptions.

- Open and dynamic. The LabelMe database is designed to allow collected labels to be instantly shared via the web and to grow over time.

In order to achieve these goals, we designed an online Javascript drawing interface that works on many platforms, is easy to use, and allows instant sharing of the collected data. Fig. 2 shows a snapshot of the LabelMe online annotation tool. The tool provides a simple drawing interface that allows users to outline the silhouettes of the objects present in each image. When the user opens the application, a new image is displayed. The image is randomly selected from a large collection of images available in LabelMe. The user provides an annotation by clicking along the boundary of an object to form a polygon. The user closes the polygon by clicking on the initial point or with a right click. After the polygon is closed, a pop-up dialog box appears querying for the object name. Once the name is introduced, the annotation is added to the database and becomes available for immediate download for research.

A. Dataset evolution and distribution of objects

Fig. 3 plots the evolution of the dataset since it went online in 2005. Fig. 3.a shows how the number of annotated objects (one annotated object is composed of the polygon outlining the object boundary and the object name) has been growing. Notice the constant database growth over time. Fig. 3.b shows the number of images with at least one object annotated. As users are not required to fully annotate an image, different images have varying numbers of annotated objects. As we try to build a large dataset, it will be common to have many images that are only partially annotated. Therefore, developing algorithms and training strategies that can cope with this issue will allow the use of large datasets without having to make the labor-intensive effort of careful image annotation.

Fig. 3.c shows the evolution of the number of different object descriptions present in the database. As users are not restricted to only annotate a pre-defined set of classes, the dataset contains a rich set of object classes that constantly grows as new objects are annotated every day. This is an important difference between the LabelMe dataset and other databases used as benchmarks for computer vision algorithms. Interestingly, the number does not seem to be saturating with time. This observation was made in [63] and seems to indicate that the number of visual object categories is large.

Fig. 4.b shows examples of the most frequently annotated object classes in our database, along with their segmentation masks. Fig. 4.a shows the distribution of annotated object classes. The vertical axis denotes the number of polygons assigned to a particular object class and the horizontal axis corresponds to its rank in the list of sorted objects according to the number of annotated instances. For instance, the most frequent object class in our dataset is *window*, with 25741 annotated instances, followed by *car*, with 20304 instances. The distribution of object counts is heavy-tailed. There are a few dozen object classes with thousands of training samples and thousands of object classes with just a handful of training samples (i.e. rare objects are frequent). The distribution follows Zipf's law [84], which is a common distribution for ranked data found also in the distribution of word counts in language. The same distribution has also been found in other image databases [63], [70].

The above observations suggest two interesting learning problems that depend on the number of available training samples N:

- Learning from few training samples (N $\rightarrow$ 1): this is the limit when the number of training examples is small. In this case, it is important to transfer knowledge from other, more frequent, object categories. This is a fundamental problem in learning theory and artificial intelligence, with recent progress given by [14], [73], [4], [65], [45], [44], [12], [29].
- Learning with millions of samples (N $\rightarrow \infty$): this is the extreme where the number of training samples is large. An example of the power of a brute force method is the text-based Google search tool. The user can formulate questions to the query engine and get reasonable answers. The engine, instead of understanding the question, is simply memorizing billions of web pages and indexing those pages using the keywords from the query. In Section III, we discuss recent work in computer vision to exploit millions of image examples.

Note, however, as illustrated in Fig. 4.a, that collected benchmark datasets do not necessarily follow Zipf's law. When building a benchmark, it is common to have similar amounts of training data for all object classes. This produces somewhat artificial distributions that might not reflect the frequency in which objects are encountered in the real world. The presence of the heavy tailed distribution of object counts in the LabelMe dataset is important to encourage the development of algorithms that can learn from few training samples by transferring knowledge from other, more frequent, object categories [14], [73], [4], [65], [45], [44].

B. Study of online labelers

An important consideration is the source of the annotations. For example, are few or many online users providing annotations? Ideally, we would collect high quality contributions from many different users since this would make the database more robust to labeling bias. In this section, we study the contributions made through the online annotation tool by

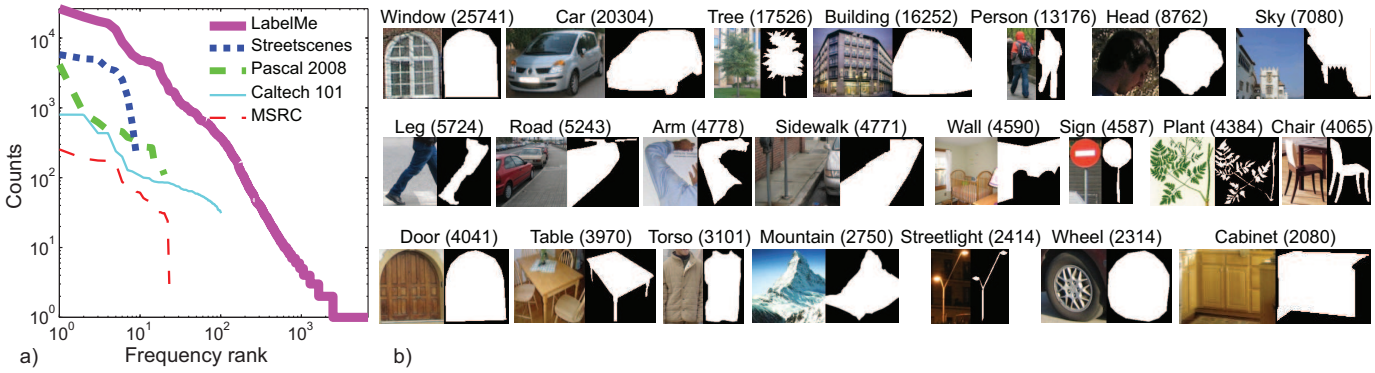

Fig. 4. a) Distribution of annotated objects in the LabelMe collection and comparison with other datasets (plotted on log-log axes). b) Examples of the most frequent objects in LabelMe. The number in parenthesis denotes the number of annotated instances. These numbers continue to evolve as more objects are annotated every day.

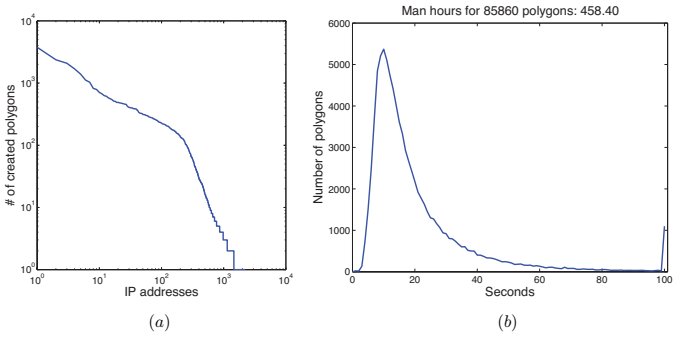

Fig. 5. (a) Number of new annotations provided by individual users of the online annotation tool from July 7th, 2008 through March 19th, 2009 (sorted in descending order, plotted on log-log axes). In total, 11382 unique IP addresses interacted with the labeling tool, with over 200 different IP addresses providing over 100 object labels. Notice that we get a diverse set of users who make significant contributions through the annotation tool. (b) Distribution of the length of time it takes to label an object (in seconds). Notice that most objects are labeled in 30 seconds or less, with the mode being 10 seconds. Excluding those annotations taking more than 100 seconds, a total of 458.4 hours have been spent creating new annotations.

analyzing the online user activity from July 7th, 2008 through March 19th, 2009.

Since the database grows when users provide new annotations, one way of characterizing the online contributions is by looking at the number of newly created polygons that each user makes. To analyze the number of new polygons that users created, we stored the actions of an online user at a particular IP address. In Fig. 5(a), we plot the total number of objects created by each IP address, sorted in descending order (plotted on log-log axes). We removed from consideration polygons that were deleted during the labeling session, which often corresponded to mistakes or from testing of the annotation tool. There were in total 11382 unique IP addresses that interacted with the labeling tool. During this time, 86828 new objects were added to the database. Notice that over 200 different IP addresses provided over 100 object labels. This suggests that a diverse set of users are making significant contributions through the annotation tool. Primarily, contributers tended to come from research universities, with occasional contributions from random visitors to the annotation tool.

Another interesting question is the amount of effort online labelers spend annotating objects. To answer this, we analyze the length of time it takes a user to label an object. We count the time starting from when the user clicks the first control point until the user closes the polygon and finishes entering the object name. Fig. 5(b) shows the distribution of the amount of time (in seconds) to create an object. Notice that most objects are labeled in under 30 seconds, with a mode of 10 seconds. Considering only annotations taking less than 100 seconds to produce (to avoid outlier annotations), the database contains 458.4 hours (19.1 days) of annotation time across all users during this time period. We wish to note that this analysis does not include the amount of time spent looking at the image or editing other annotations.

We further look at the difficulty of labeling particular object classes. In Table I, we show the average time (in seconds) to label an object for a particular class, along with the total man hours devoted to labeling that object. We exclude annotation times exceeding 100 seconds from our analysis. Windows, which often require only four control points, are easiest to label. Region-based objects, such as sky and ground, are more difficult.

III. The Space of LabelMe Images

A number of recent papers have used large datasets of images in conjunction with non-parametric methods for computer vision [72], [35], [34], [36], [9] and graphics applications [61], [20], [58]. The main observation is that when large amounts of images are available, image indexing techniques can be used to retrieve images with similar object arrangements as the query image. This observation suggests a non-parametric approach for scene understanding. With a large enough database, we can find some images in the database that are close to a query image, such as similar scenes with similar objects arranged in similar spatial configurations. If the images in the retrieval set are partially labeled, then we can transfer the knowledge of the labeling to the query image.

In section II we studied the number of different object categories available in the LabelMe dataset and the distribution of annotated examples for each category. In this section we are interested in using the database to study how many different

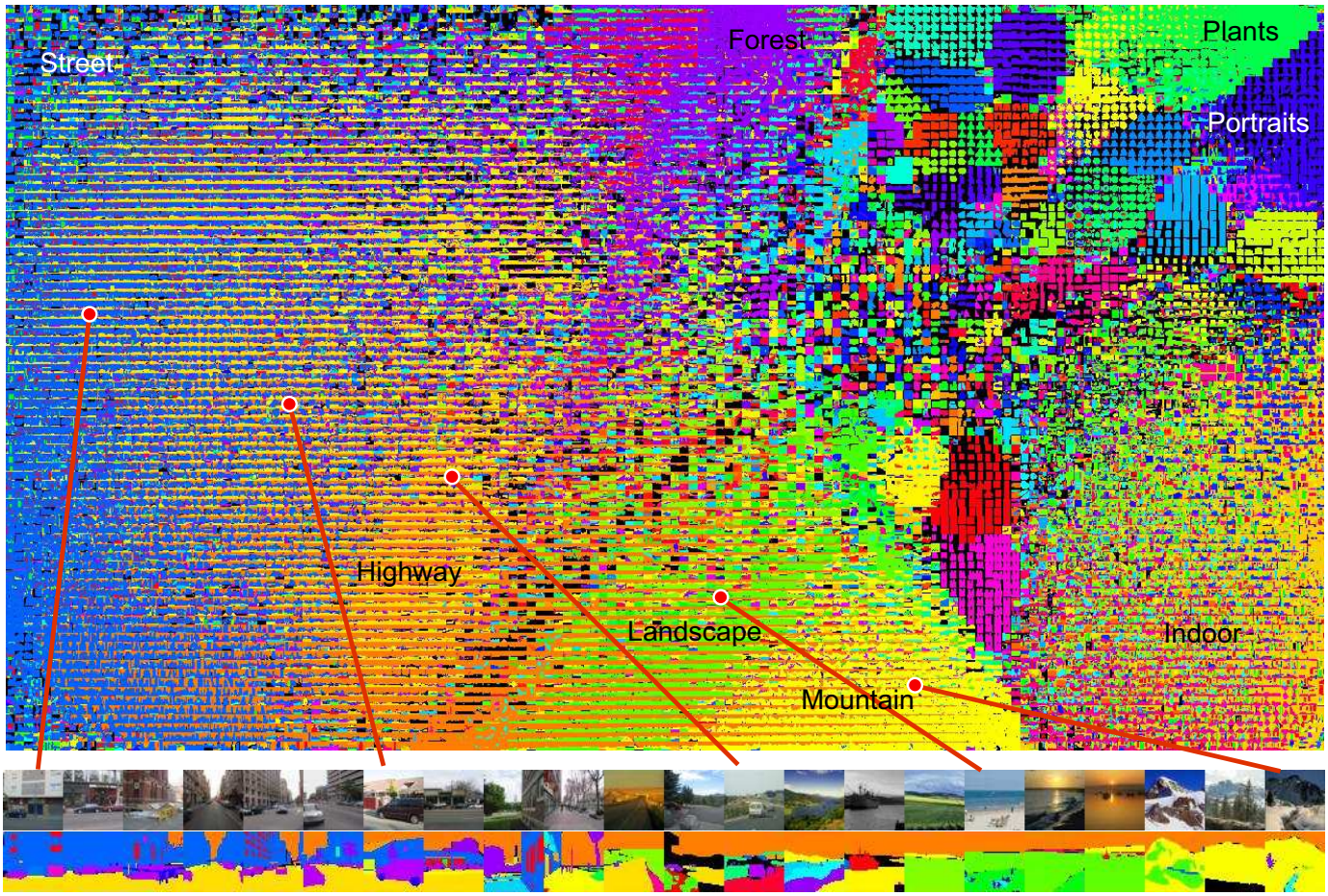

Fig. 6. The images are arranged according to semantic similarity between images (nearby images will contain similar objects in similar spatial configurations). Each thumbnail shows the object segments of each image, with the objects consistently colored across the database. Although there are some easily identifiable clusters in the space, most of the images are organized across a continuous space in which transitions across images are smooth.

scenes there are in LabelMe, and how are they organized. We will also look into how many images need to be annotated.

A. Distribution of scene types

In cognitive psychology, studies on scene perception suggests that the gist of a scene might be composed of the scene category and a list of 4 or 5 objects. In [69], it was shown that observers can recognize images at low resolution. In the extreme case where images have just 32×32 pixels, observers are able to recognize the scene category, together with 4-5 objects, with an accuracy of 80%. Our goal now is to study how many configurations of 4 objects are present in the LabelMe database. This is similar to studies in language that build probabilistic models of groups of n words.

Fig. 7 shows the distribution of n-grams obtained as the n words that describe the n largest objects in each image. These statistics are derived from the analysis of 12201 scenes containing a total of 180391 annotated objects. For each image, we sort all the objects according to the percentage of the image covered by each polygon. We only consider the n largest objects. The figure shows the distribution of scenes (n-grams) for $n = 1, 2, 4, 8$. For all the tested values of n, the distribution appears to follow a power law [57]. As n increases, the number of different scene configurations increases and only a small percentage of scenes seem to co-occur often. In the case of $n = 4$, Fig. 7.b shows some of the most frequent 4-grams, along with an example image for each 4-gram. There are more than 100 4-grams that appear 10 times or more in the database. Therefore, one can expect that, as the database increases in size, the most common scenes will have many instances. The heavy tail of the distribution also points to the fact that, independent of how large the database is, there will always be a large number of scene configurations for which we will have only a handful of training examples.

B. The space of images

In the previous section we discretized the space of scenes by defining a scene as being a collection of $n = 4$ large objects and ignoring their spatial organization. Here, we will consider a description of all the objects in the image that will also incorporate spatial information.

We first define a distance between annotations that captures our notion of semantic distance between two images. Ideally, two images are semantically similar if their segmentations and object labels are interchangeable across the two images. Our definition of semantic distance between two images is based

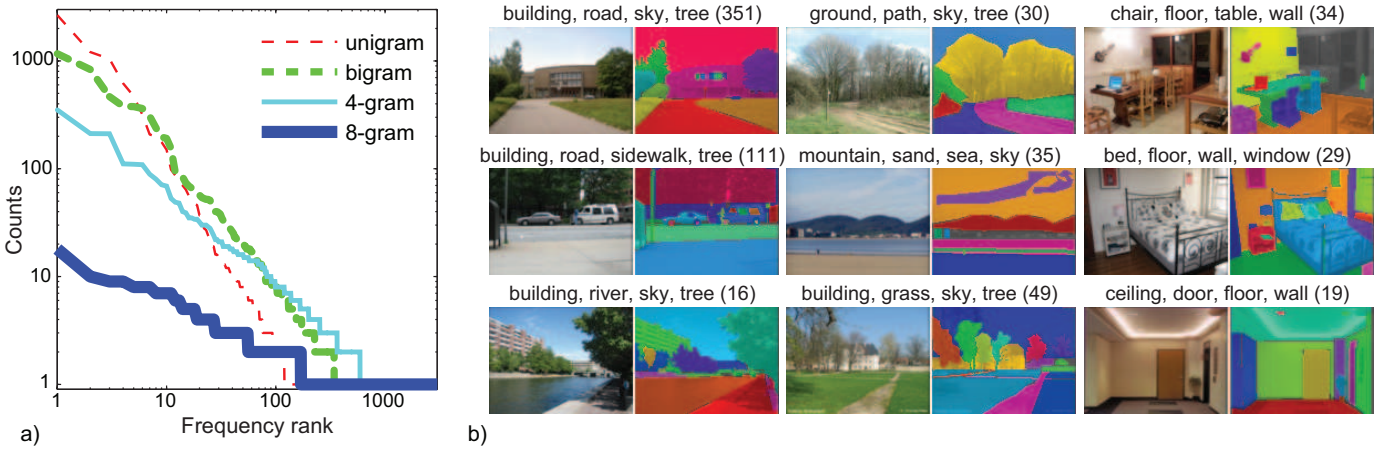

Fig. 7. a) Distribution of n-grams in LabelMe. Each n-gram corresponds to the list of n largest objects in each scene. b) Examples of scenes and 4-grams.

Object	Average labeling time	Total labeling time (hours)
window	9.52	11.08
door	9.98	2.23
sign	10.35	2.16
lamp	11.47	6.93
bottle	14.42	2.02
head	14.79	8.40
plant	16.12	2.22
arm	17.04	14.92
car	17.99	5.49
wall	18.54	19.65
grass	18.54	2.99
floor	19.27	7.95
ceiling	20.57	6.43
table	20.88	3.14
sidewalk	21.09	4.26
shelves	22.57	2.41
leg	22.77	24.04
building	23.16	14.83
person	23.40	2.94
road	23.44	4.17
torso	23.80	14.14
chair	24.16	4.18
tree	25.94	11.85
sky	29.37	10.76
plate	34.42	3.69
fork	34.60	2.75
wineglass	41.52	2.00

TABLE I
AVERAGE TIME TO LABEL A GIVEN OBJECT CLASS, ALONG WITH THE TOTAL NUMBER OF HOURS SPENT LABELING THE CLASS. NOTICE THAT CERTAIN OBJECT CLASSES ARE EASIER TO LABEL (E.G. WINDOWS), WHICH REQUIRE FEWER CONTROL POINTS. OTHERS ARE HARDER (E.G. ROAD, SKY), WHICH ARE REGIONS AND REQUIRE MORE CONTROL POINTS.

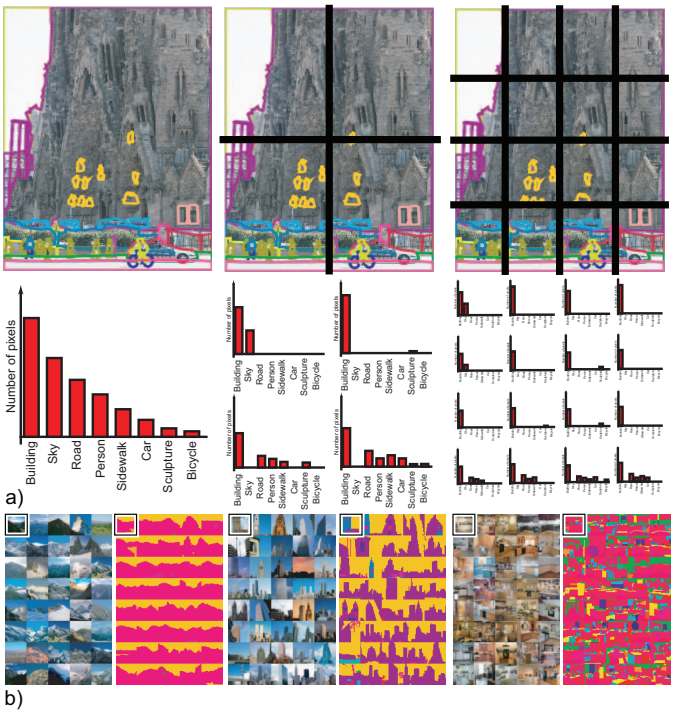

Fig. 8. a) Procedure to estimate the semantic distance between pairs of images. For each image we compute the histogram of object labels in different non-overlapping image windows. Then, we define the similarity between two images as the intersection between the two object histograms. b) Examples of similar images with this metric.

on the histogram of object labels in the two images [71]. First, we assign each pixel to a single object category. Then, we create the histogram of the number of pixels that belong to each category. In order to account for spatial information we divide the image into $N \times N$ non-overlapping windows and we build the object histogram for each window. Then, to measure the distance between two images we use spatial pyramid matching [32], [17] over object labels. This process is illustrated in figure 8. Matching of object label histograms results in a simple similarity measure that takes into account all the objects present in the image (and not just the 4 largest ones), in addition to their spatial organization. The spatial pyramid matching allows for the comparison of images such that two images that have the same object labels in similar spatial locations are rated as closer than two images with the same objects but in different spatial locations. Furthermore, this is rated closer than two images with different object classes.

Fig. 6 shows a visualization of 12201 images that are fully annotated from the LabelMe dataset. The images are organized according to the similarity defined above. As a consequence, two nearby images in this mosaic are likely to contain the

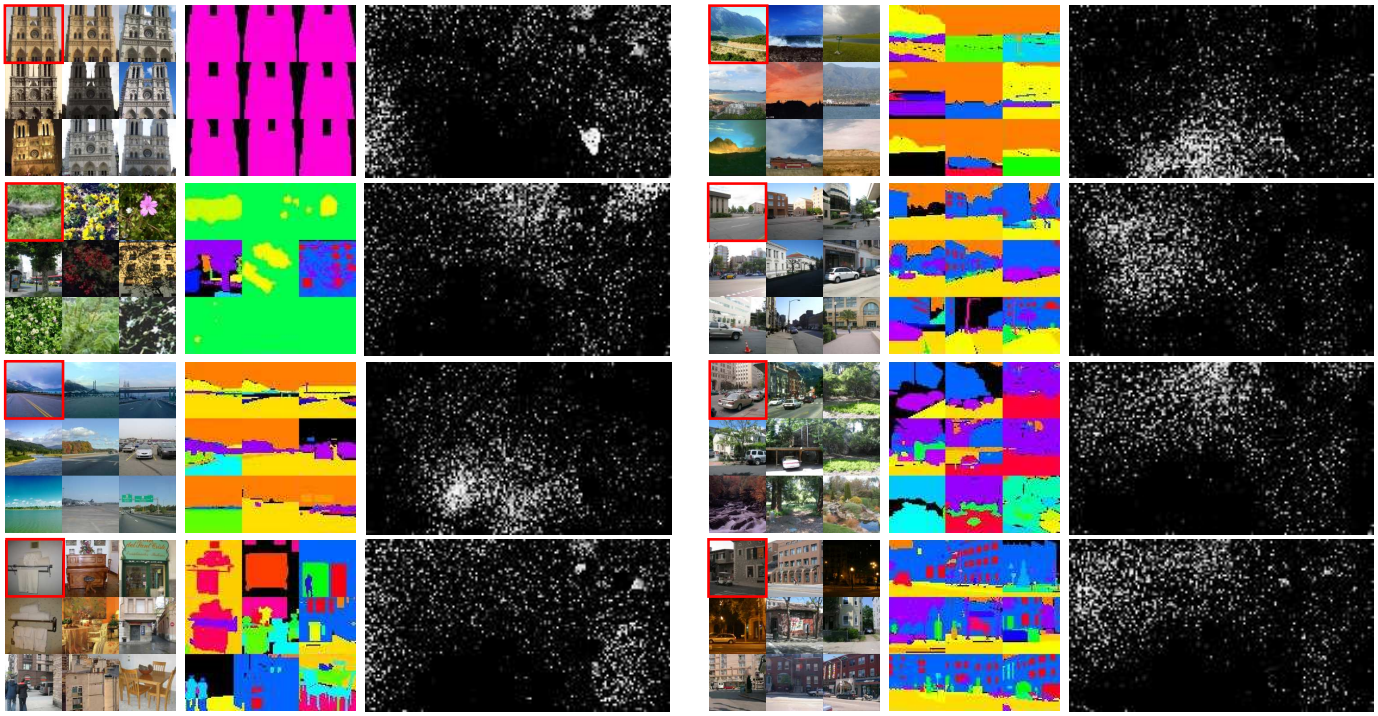

Fig. 9. Examples of input images and their nearest neighbors in the dataset using the GIST descriptor [40]. For each panel: (left) mosaic showing the query image (red box) and its 8 nearest neighbors. (middle) labeled objects within each image and (right) the LabelMe map showing the location of the 1,000 closest images among the 12,201 images that compose this test set.

same object categories in similar spatial configurations. Each tile shows the segmentation of an image, with each object class having a unique color[1].

There are a number of methods that can be used to obtain a 2D visualization of the space of images from the matrix of semantic similarities defined above. For the visualization of Fig. 6 we used kernelized sorting [42]. The advantage of this technique is that it allows specifying the form of the output space (in this case a rectangular grid). Kernelized sorting will try to find the best correspondence between the images and the locations in the rectangular grid, while trying to preserve the same neighborhood structure.

Although there are some easily identifiable clusters in the space, most of the images in the LabelMe dataset are organized across a continuous space in which transitions across images appear to be smooth. Most of the images in the LabelMe dataset are taken by photographers standing on the ground, which helps to reduce the variability across images. However, it might also introduce biases not found in other photo collections, such as Flickr. The clusters that are visible in Fig. 6 correspond to regions of the image space that are not appropriately sampled in the LabelMe dataset (e.g. a collection of flower photographs, pictures of specific monuments, or a collection of pictures of silverware). However, there is a large portion of the space that has no clearly defined boundaries. For instance, we can start on a picture of a busy downtown center and continue moving in the space by reducing the size of the buildings and adding more sky until we get a highway scene. Furthermore, we can reduce the size of the road until the picture becomes a field. Finally, we can add mountains in the background until the scene becomes a mountainous landscape. This transformation can take place by traversing the space of images, as shown in the bottom of Fig. 6.

[1]An interactive version of the tool is available at: http://people.csail.mit.edu/torralba/research/LabelMe/labelmeMap/

C. Recognition by scene alignment

As illustrated in Fig. 6, some regions of the scene space seem to be covered by a large number of examples. The goal now is, given a new image, to extract a set of image features to locate the region of the space that is the closest, at the semantic level, to the input image [20], [71], [70].

In the examples used here, we use the GIST descriptor [40] to estimate the similarity between two images. To compute the GIST descriptor, the image is first decomposed by a bank of multiscale-oriented filters (tuned to six orientations and four scales). Then, the output magnitude of each filter is averaged over 16 non-overlapping windows arranged on a 4×4 spatial grid. The resulting image representation is a 512 dimensional feature vector. The distance between two images is computed as the Euclidean distance between GIST descriptors.

Fig. 9 shows examples of 8 input images and their nearest neighbors in the dataset using the GIST descriptor. For each panel, we show the query image (red box), the 8 nearest neighbors, the annotations of the neighbors and the location of the 1,000 closest images among the 12,201 images that compose this test set, as shown in Fig. 6. When searching for pictures of specific places, such as a picture of Notre Dame, if the database contains many exemplars of that place, it is possible to get very tight matches. However, in general,

we will work at the category level. We want to find images corresponding to visually similar places (i.e. containing similar objects roughly with the same spatial configuration) but that do not necessarily correspond to the same world location or even the same city. As shown in Fig. 9, for several of the input images, the images in the database that have close visual similarity (as captured by the GIST descriptor) also fall within a localized region of the map organized by semantic distance (Fig. 6).

This property provides the basis for several approaches for recognition that use the retrieved images to make proposals about possible object categories that can be present in the input image [20], [50], [71], [70], [35], [34]. To illustrate the power of large scale databases, we evaluate the following simple algorithm: given an image and an annotated database, search for the image in the database that is closest to the input image (using GIST to measure image similarity). Then, output the annotation of the nearest neighbor as a labeling of the input image. As a performance metric, we use the percentage of pixels that are correctly labeled. To test the algorithm, we will use as input the set of 12,201 images used in Fig. 9. For this algorithm, we can also provide an upper bound for the recognition rate. Since the input image is also annotated, we can search for the image in the database that has the largest number of pixels with the same label as the input. As our goal is to predict the labels of all the pixels of the input image using a single nearest neighbor, this measure will give us an upper bound to the performance. Notice how the bound increases proportionally to the size of the database.

In fig. 10 we demonstrate how the performance of nearest neighbors improves as we enlarge the dataset. We also show how errors are distributed in the map of Fig. 6. In order to test the dependency of the database size, we randomly sampled our database of 12,201 images to create 4 image databases of different sizes: 12, 122, 1220, and 12201. For testing, we exclude the query image from the database to avoid over-fitting. Despite the simplicity of the nearest neighbor algorithm, we observe performance increases proportional to the database size, as shown in Fig 10.a.

The performance of this algorithm depends on the sampling density of the image space. Therefore, one can expect that the performance will vary depending on the regions of the space. In this study we can use the organization of scenes from Fig. 6 to visualize the distribution of errors. Fig 10.b shows how the performance is distributed in the map of scenes as we change the size of the database. As we can see, the performance appears to smoothly vary across different regions of the image space. This suggests that different regions of the space are harder to recognize and require higher density of image samples. Moreover, the distribution of performance is very similar between the algorithm using GIST descriptors and the upper bound for each image.

The region with highest performance corresponds to a region of the space that contains many pictures of specific monuments under similar viewpoints. In such a case, it is possible to find very close matches, with the annotations between the input and retrieved images being almost identical. The worst performance is found in the indoor scenes region, in part because there are fewer than 2000 indoor scenes out of the 12,201 images, but also due to the large variability of visual appearances for indoor scenes. Lower performances on indoor scenes are also found with other datasets [43].

Fig. 10.a also gives a hint to an important question: How many more images do we need to label? The figure shows the upper bound of the extrapolated performance as we increase the database size (here we assume that, by increasing the database size we do not introduce new kinds of scenes). As shown in the graph, performance reaches 90% for a database of 8×10^6 images. If we had 8×10^6 images, then, on average, for an image we can find another image that has 90% of the pixels labeled with the same object category. Although increasing LabelMe will require a significant labeling effort, this target database size is feasible.

IV. Beyond 2D images

In this section we present two extensions of LabelMe. The first one shows how it is possible to infer knowledge not explicitly present in LabelMe annotations. It uses object annotations across the LabelMe database to build a model of 3D scenes. With this extra information, we are able to recover the 3D layout of the scene and learn about other spatial properties of scenes in 3D from 2D images. A second extension of LabelMe explores video annotation. We describe challenges encountered in video annotation and propose an annotation tool to aid the creation of ground truth video data.

A. From annotations to 3D

In the previous section we described the annotation tool and analyzed the content of the database. In the online annotation tool we ask users to only provide outlines and names for the objects present in each picture. However, there are many other different types of information that could be requested. In this section we will show that object outlines and names from a large number of images are sufficient to infer many other types of information, such as object-part hierarchies or reasoning about occlusions, despite not being explicitly provided by the user. Furthermore, we will discuss how to recover a full 3D description of the scene, as shown in Fig. 11. Our system can reconstruct the 3D structure of the scene, as well as estimate the real-world distances between the different depicted objects. As an added benefit, the quality of the reconstruction tends to improve as the user improves the annotation of the image.

A database of images and their 3D description would be useful for various tasks in computer vision. For example, the information can be used to learn about how objects live in the world and to train systems to detect images. Techniques for aligning images [20], [26], [50] 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). We demonstrate how using the user annotations we have so far collected from the LabelMe system, we can complement our database depicting different scene and object classes with information about their underlying real world 3D coordinates. Previous

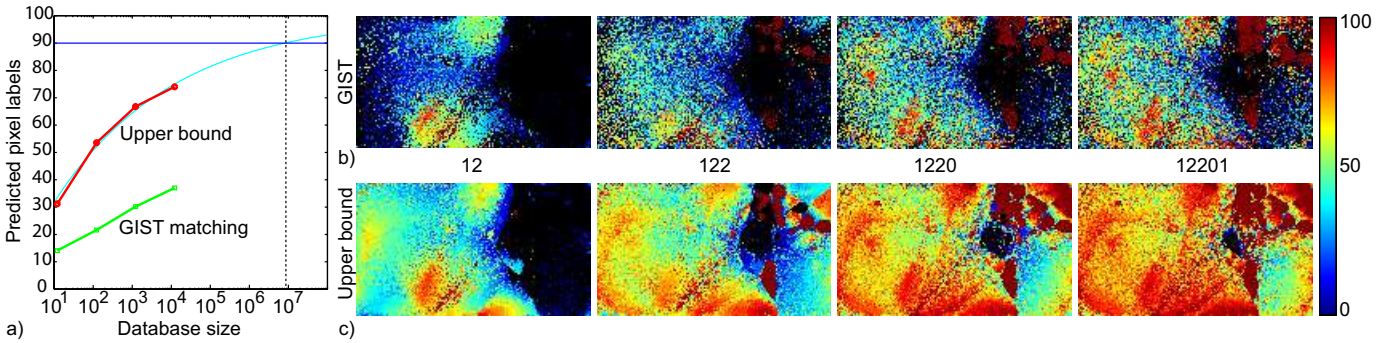

Fig. 10. a) Recognition performance as a function of dataset size. b) Distribution of the recognition performance in the different regions of the image space defined in Fig. 6.

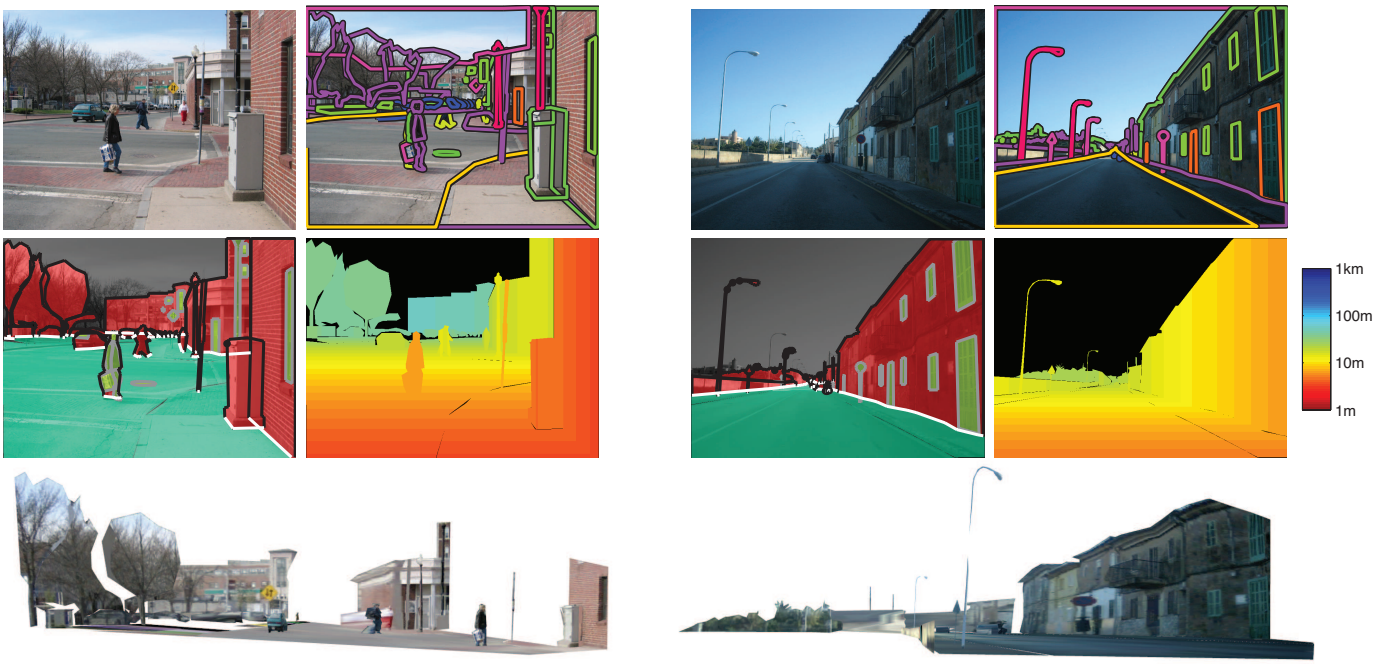

Fig. 11. We can recover 3D information from the user annotations. We show outputs for two input images. Top-left: Input image. Top-right: User annotations provided for the image. Middle-left: Recovered polygon and edge types. Polygons are either *ground* (green), *standing* (red), or *attached* (yellow). Edges are *contact* (white), occluded *black*, or *attached* (gray). Middle-right: Recovered depth map in real-world coordinates (a color key, in log scale, appears on the right). Bottom: A visualization of the scene from a different viewpoint.

work has explored ways of associating 3D information to images. For example, there are existing databases captured with range scanners or stereo cameras [53], [52]. However, these databases tend to be small and constrained to specific locations due to the lack of widespread use of such apparatuses.

Instead of manually gathering data with specialized equipment, other approaches have looked at harnessing the vast amount of images available on the Internet. For example, recent work has looked at learning directly the dependency of image brightness on depth from photographs registered with range data [53] or the orientation of major scene components, such as walls or ground surfaces, from a variety of image features [22], [23], [24]. Since only low and mid level visual cues are used, these techniques tend to have limited accuracy across a large number of scenes. Other work has looked at using large collections of images from the same location to produce 3D reconstructions [61]. While this line of research is promising, at present, producing 3D reconstructions is limited to a small number of sites in the world. Finally, there are other recent relevant methods to recover geometric information for images [21], [58], [9], [46], [67], [33], [19], [38], [83].

An alternative approach is to ask humans to explicitly label 3D information [26], [7], [39]. However, this information can be difficult and unintuitive to provide. Instead, we develop a method that does not require from the user any knowledge about geometry, as all of the 3D information is automatically inferred from the annotations. For instance, the method will know that a *road* is a horizontal surface and that a *car* is supported by the *road*. All of this information is learned by analyzing all the other labels already present in the database.

At first glance, it may seem impossible to recover the absolute 3D coordinates of an imaged scene simply from object

labels alone. However, the object tags and polygons provided by online labelers contain much implicit information about the 3D layout of the scene. For example, information about which objects tend to be attached to each other or support one another can be extracted by analyzing the overlap between object boundaries across the entire database of annotations. These object relationships are important for recovering 3D information and, more generally, may be useful for a generic scene understanding system.

Our reconstructions are approximations to the real 3D structure as we make a number of strong simplifying assumptions about the object geometries. Here we summarize all the information that is needed by our system in order to provide a 3D reconstruction of the scene. Our reconstructions are based on the following components, which are inspired from early work in line-drawing analysis [2], [6], [3], [27], [66].

- **Object types**. We simplify the 3D recovery problem by considering three simple geometric models for the objects that compose each scene:
  - Ground objects: we assume that ground objects are horizontal surfaces (e.g. road, sidewalk, grass, sea).
  - Standing objects: we assume that standing objects are modeled as a set of piecewise-connected planes oriented orthogonally to the ground plane (e.g. person, car, boat, table).
  - Attached objects: we assume that attached objects are part-of other objects (e.g. hand, window, road marking), with their 3D position completely determined by their parent object.
- **Relations between objects**: in addition to the object types described above, we also consider two types of relationships between pairs of objects:
  - Supported-by relationship: we assume that standing objects in the scene are supported by a ground object, with the relationship extracted at the category level. For instance, we expect that sidewalks support people, fire hydrants, and parking meters.
  - Part-of relationship: attached objects are part-of other objects, with the relationship extracted at the category level. For instance, heads are attached to people, windows are attached to buildings, and manhole covers are attached to roads.

In our model, we assume that a scene consists of a number of objects that stand on the ground. This assumption holds true for many different imaged scenes (e.g. streets, natural landscapes, lakes, indoors). In addition, we assume that the horizon line is parallel to the horizontal axis of the camera (this is true for most normal pictures).

There are two steps for obtaining the 3D information: (i) the learning stage, where the system learns from all the annotated objects in the database the relationships that hold between all the object classes (part-of and supported-by) and (ii) the reconstruction stage, where, given an annotated image and all the learned relationships, the system builds a 3D model for the input image.

We start by describing the learning stage to recover the part-of and supported-by relationships that hold between object classes. Fig. 12(a) shows automatically recovered part-of relationships across the database. To decide when an object category is part-of another, we evaluate the frequency of high relative overlap between polygons of the two categories. For instance, as windows are part of buildings, whenever windows and buildings co-occur in a scene, it is quite likely that the polygon defining a window will completely lie inside the polygon defining the building. On the other hand, street lamps are not part of buildings, so one would expect that the polygons do not systematically overlap.

In a similar manner, we can reason about the supported-by relationships. Objects that are supported by another tend to have the bottom part of its polygon live inside the supporting object. For instance, we can make a list of all the object categories that overlap with the bottom part of the polygon defined by all the *street lamps* in the database. If the object is a supported object, we will see that this list is relatively

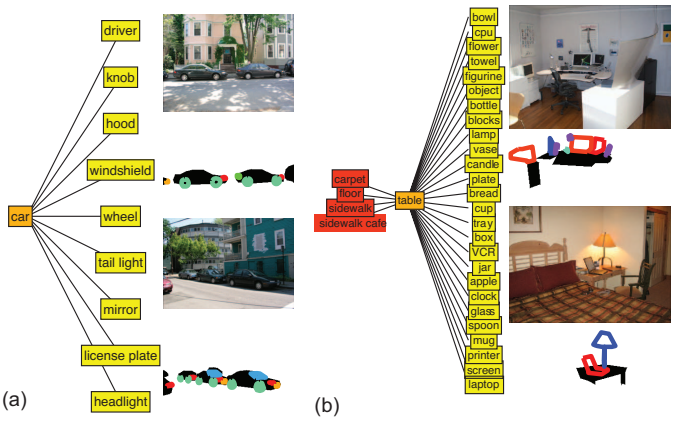

Fig. 12. Automatically recovered spatial relationships between objects from the LabelMe dataset. The left-hand side of each pair depicts a graph of the spatial relationship holding between object classes. The right-hand side shows examples of the spatial relationship. (a) Attachment relationships for car. (b) Support relationships for table. Notice that the spatial relationship may form a hierarchy.

Fig. 13. Snapshot of the 3D measuring tool. Once we compute the 3D coordinates for a depicted scene, we can make measurements of scene components. Here, we show the height and width of the car, which is 13.68 meters away from the camera center. We can also compute the distance between any two points in the scene, such as the selected points on the building and the truck.

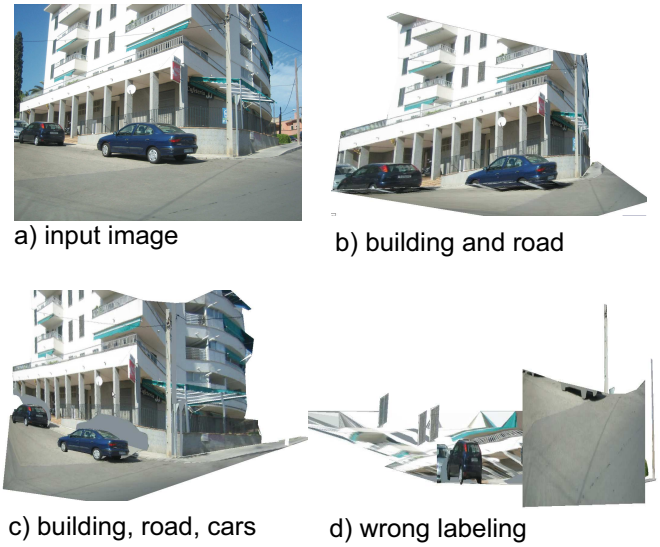

Fig. 14. As the user adds more annotations, the quality of the reconstruction improves. a) input image, b) 3D model after the user annotated the *road* and the *building*, c) model obtained after adding the *car* to the list of annotated objects. d) Reconstructed model when the labels are incorrectly introduced so that the building is labeled as a road and vice versa.

short. Fig. 12(b) shows automatically recovered supported-by relationships across the database.

Once the learning is done and we have collected all the co-occurrence statistics between object category pairs, we can use the discovered relationships to recover 3D models of new images. Given an annotated image, we will use the polygons and object names, along with the discovered relationships, to decide the object types (standing, ground, attached) for all of the annotations in the image. For this, we extract the cues for the supported-by and part-of relationships (polygon overlap and distance to ground objects) and use the recovered co-occurrence statistics to infer the object types. We show the inferred polygon types in Fig. 11, where standing objects are colored red, ground objects are green, and attached objects are yellow. Notice that the recovered object types agree well with the objects present in the scene.

In addition to knowing the support relationship between different object categories, it is also important to know which part of the object makes contact with the ground. For example, the contact points with the ground plane for standing objects will provide information about the relative distance of the object to the camera. For this, we automatically label polygon edges into three types: *contact*, *attached*, *occlusion*. We assume that attached and ground objects have all of their edges labeled as attached. Standing objects can have contact or occlusion edges. We extract the following cues for each polygon edge: length, orientation, and distance to a support object. These cues are used to infer whether the edge is a contact or an occlusion edge. A generative model is used to classify the edge type, with the model parameters trained on a held-out training set. Fig. 11 show the edges labeled into the different types, with white lines corresponding to contact edges, black lines corresponding to occlusion edges, and gray lines corresponding to attached edges. By recovering the polygon and edge types, we can already pop-up the scene by placing standing objects on the ground objects and letting attached objects remain on the objects they are attached to, as illustrated in Fig. 11.

We wish to also extract absolute 3D coordinates. Important for this is to (i) produce 3D coordinates such that objects keep consistent heights across the database and (ii) enforce constraints on the image imposed by the camera through perspective projection. More specifically, as in [28], we learn the distribution of object heights across the database and the camera parameters corresponding to each image in the database. This is achieved in an iterative fashion by first estimating the camera parameters given the current guesses of the heights of the objects in the image. Then, the object heights are updated using the estimated camera parameters. The entire process is seeded by providing the mean and variance of the height of the "person" object class. For the camera, we assume that it is held level with the ground, with the parameters being the horizon line (the image location of where the ground plane vanishes to infinity), camera height from the ground, and focal length. Once we recover the camera parameters for an image, it is straightforward to obtain the 3D information of the scene. Please see [28], [48] for more details.

We show output depth maps of our system in Fig. 11. The distance (in meters) is given by the color key, which is plotted in log scale. In addition, we can interact with the scene by taking measurements of the scene components. In Fig. 13, we show the height and width of a depicted car. We also show the distance between two points in the scene. Notice that the measured points appear to be consistent with the perceived distances.

We measured the accuracy of our system output depth maps on a dataset that simultaneously utilized both camera and laser range scanner apparatuses [53]. The dataset was gathered on the Stanford University campus and primarily depicts outdoor scenes. We provided dense object labels for 62 images in the dataset, with each image having 256x192 pixel resolution. The system output was then compared with the output of the laser range scanner using mean per-pixel relative error (i.e. absolute difference between the two depth maps normalized by the output of the laser range scanner). Due to noise in the range data, we only considered ground truth and system output depths in the 5-70 meter range. To overcome bias in the data, we performed cross-validation, with training sets consisting of 20 images and validation sets consisting of 42 images, and found linear regressors that minimized the mean per-pixel relative error over the training sets.

Our system has relative error of 0.29 ± 0.02, with $40\% \pm 2\%$ of the pixels used in the evaluation. As a baseline, we compared against the harmonic mean of the depth maps corresponding to the training images. For each pixel, the harmonic mean is computed as $d = \frac{N}{\sum_{i=1}^{N} \frac{1}{d_i}}$ where d_i is the depth value at the corresponding pixel in the ith image in the training set. The harmonic mean minimizes the squared relative error $\sum_{i=1}^{N} \frac{(d_i - d)^2}{d_i}$ across the training set. The baseline has relative error of 0.33 ± 0.04. Overall, we obtained less noisy outputs than the laser range scanner and

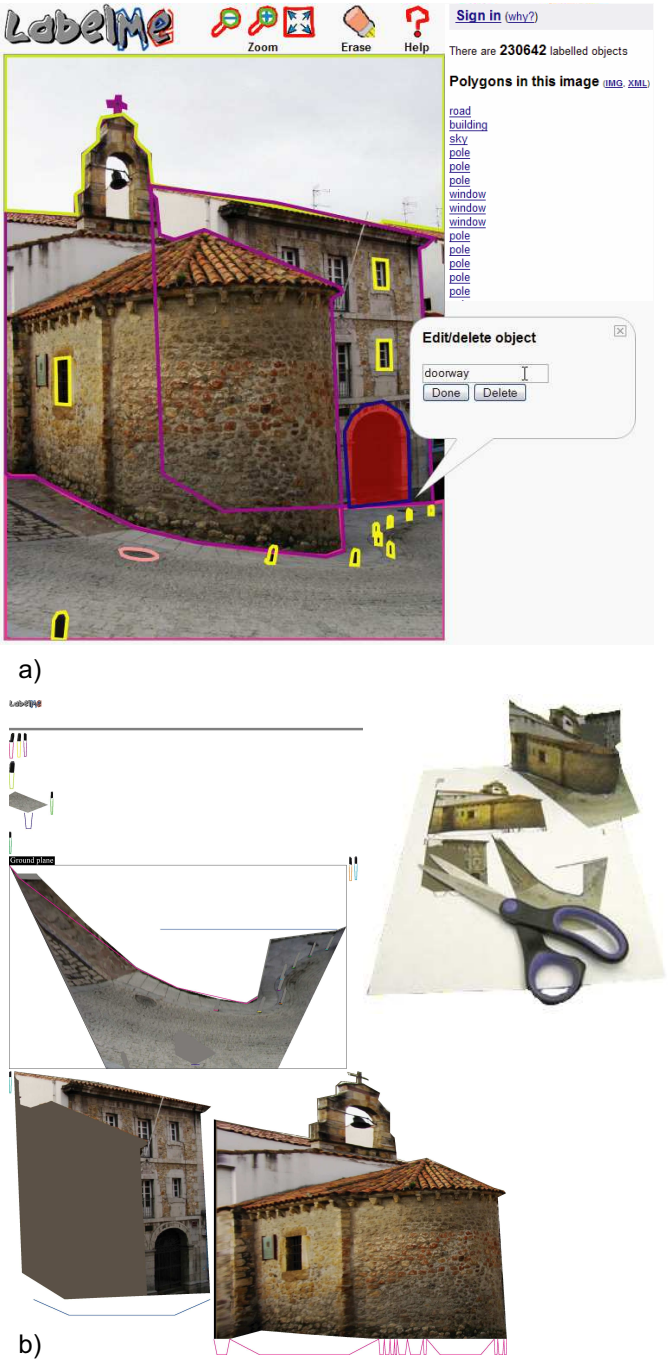

Fig. 15. Automatically generated instructions for a "do-it-yourself pop-up book" that can be constructed using paper, glue, and scissors.

were able to produce visually plausible output depths beyond the 5-70 meter range. Furthermore, we were able to overcome errors resulting from the range scanner that were caused by object reflection (e.g. mirrors, shiny surfaces) and transparency (windows, tree leaves).

Because our system uses only user annotations, the quality of the output is heavily dependant on the quality of the labels. For example, consider Fig. 14, which shows outputs for different labelings of the same scene. If few objects are labeled, the output is less reliable since there are few constraints for estimating the camera parameters. As more objects are labeled, the estimates improve. If a user enters incorrect object tags, then this may result in poor outputs. Moreover, the estimated 3D coordinates can be greatly affected by the placement of the control points. This can have a noticeable effect on distant objects since they occupy fewer pixels in the image and the change in depth increases as one moves closer to the horizon line in the image.

Another output of our system is a set of instructions for building a "do-it-yourself pop-up book", shown in Fig. 15. This is automatically generated and allows the user to cut and glue the picture (with all the objects' perspective corrected) in order to build a physical model of their picture.

B. Video annotation

In the last decade, annotated images have played an important role in the advancement of various areas in computer vision. Image datasets have evolved from containing few to thousands of categories thanks to large collaborative efforts. The concept of annotating an image to generate ground truth is not new. However, the computational power gained by outsourcing this task and sharing data freely has aided the development of algorithms that take advantage of large image datasets. Despite these advancements and the large volumes of video data generated everyday from video surveillance, consumer camcorders, and mobile devices, video datasets are not as advanced as static image datasets. Annotated video can be useful in the development of algorithms for motion estimation and object, event, and action recognition. Unlike image annotation, video annotation is not as simple. Challenges include the massive additional workload that video frames generate, the annotation ambiguity in situations like occlusion and out-of-frame objects, and multiple choices in annotation granularity, amongst others.

There has been prior work on collecting and annotating videos. The KTH database has been widely used as a video benchmark, and depicts close-up views of a number of human action classes performed at different viewpoints [55]. A similar database was collected containing various sports actions [10]. While these databases offer a rich vocabulary of actions, the number of object and action classes and examples is small compared to their static image counterparts.

There also has been recent work to scale up video databases to contain a larger number of examples. The *TRECVID* [60] project contains many hours of television programs and is a widely used benchmark in the information retrieval community. This database provides tags of scenes, objects, and actions, which are used for training and validation of retrieval tasks. Another example is the database in [31], and later extended in [30], which was collected from Hollywood movies. This database contains up to hundreds of examples per action class, with some actions being quite subtle (e.g. *drinking* and *smoking*). However, there is little annotation of objects and their spatial extent and the distribution of the data is troublesome due to copyright issues.

We created an open database of videos where users can upload, annotate, and download content efficiently. In creating this application, some desired features include speed,

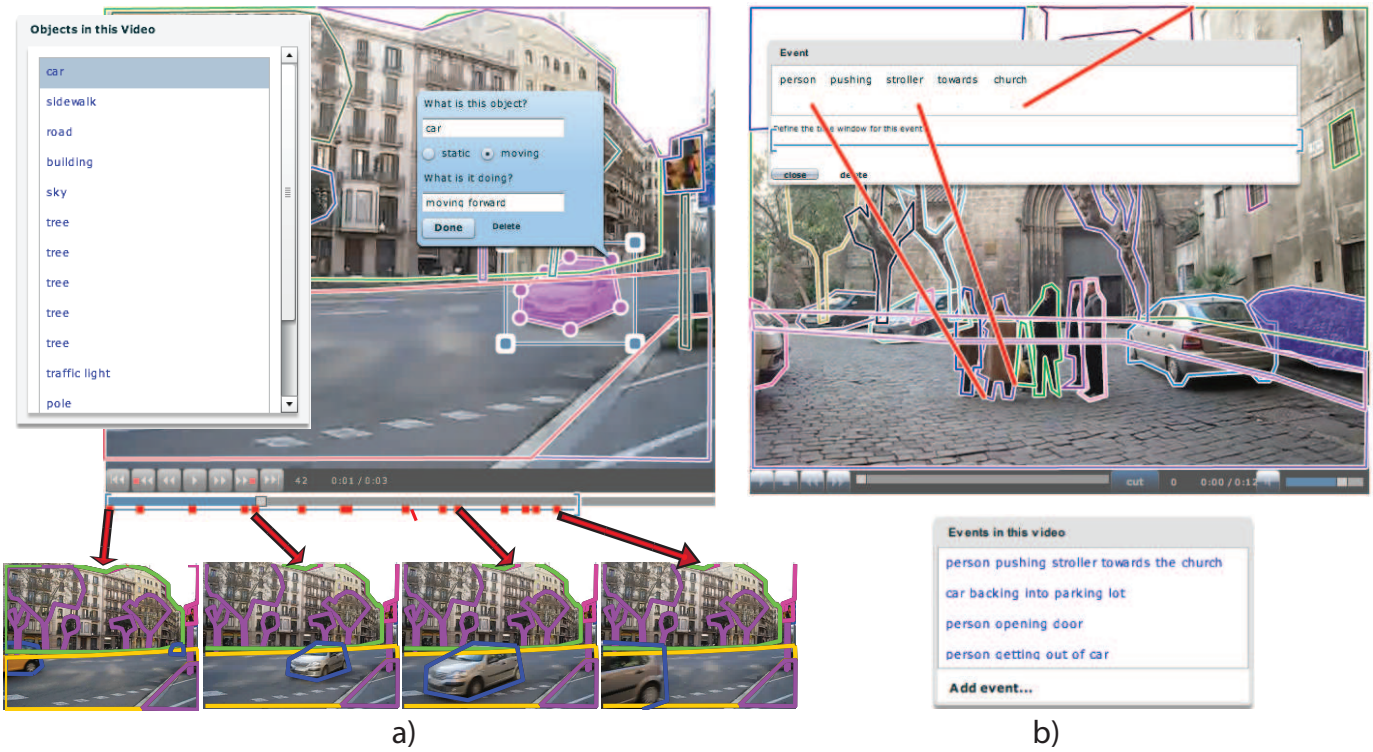

Fig. 16. (a) A snapshot of our video annotation tool exemplifying densely annotated objects and some selected key frames. Static objects are annotated in the same way as in LabelMe and moving objects require some minimal user intervention (manually edited frames are denoted by the red squares in the video track). (b) Events can also be annotated by entering free form sentences and associating text tokens to object annotations.

responsiveness, and intuitiveness. It is also important to handle system failures, such as those related to camera tracking and interpolation, so as not to dramatically hinder the user experience. The consideration of these features is vital to the development of our system as they constrain the computer vision techniques that can be feasibly used. With these ideas in consideration, we created LabelMe video, an extension of LabelMe, to annotate and share videos.

The object annotation feature in LabelMe video is similar to that of static LabelMe. An object annotation consists of a name, a Boolean field determining if the object is moving, and a text field answering the question *what is it doing?* (if anything). The spatial extent of an object is described by a polygon at each frame in the video. To annotate an object, the user is asked to outline the boundary of the object at the selected frame. Initially, the newly introduced polygon is propagated across all frames in the video. The user can inspect the annotation throughout the video and adjust the polygon at other frames. The application relies on interpolation algorithms to propagate the polygon edits in-between key frames. This process is repeated until the user is satisfied with the annotation throughout the video. Fig. 16.a shows an example video with annotated objects at multiple key frames.

Users can also annotate events where one or more nouns interact with each other. To enter an event, the user clicks on the *Add Event* button, which prompts a panel where the user is asked for a sentence description of the event (*e.g. the dog is chewing a bone*). The event annotation tool renders a button for each token in the sentence, which the user can click on and link with one or more polygons in the video. Finally, the user is asked to specify the time when the described event occurs using a time slider. Once the event is annotated, the user can browse through objects and events to visualize the annotation details. Fig. 16.b illustrates this feature.

Nowadays, as video cameras are ubiquitous, we expect a considerable portion of videos to be captured by handheld recorders. Even under shake-correction modes, static objects in the real world might appear as motion in video. Camera motion can make annotation tedious as simple cloning of polygon locations across time might produce misaligned polygons even for static objects depicted in the scene. Our system consists of estimating camera motion as a homographic transformation between each pair of consecutive frames during an offline pre-processing stage. The camera motion parameters are encoded, saved in our servers, and downloaded by the web client when the user loads a video to annotate. When the user finishes outlining an object, the web client software propagates the location of the polygon across the video by taking into account the camera parameters. Therefore, if the object is static, the annotation will move together with the camera and not require further correction from the user. In this setup, even with camera tracking failures, we observe that the user can correct the annotation of the polygon and continue annotating without generating uncorrectable artifacts in the video or in the final annotation.

We have begun by contributing an initial database of over 1500 videos and annotated over 1903 objects, spanning over 238 object and 70 action classes. Fig. 16 shows a screenshot

of our labeling tool and a sample annotation for a video. With an evolved dataset, we expect to help develop new algorithms for video understanding similar to the contribution of LabelMe in the static image domain.

V. Conclusion

In this work, we developed a web-based annotation tool that allows the labeling of objects and their location in images. Through the tool, we have collected a large annotated database of images spanning many different scene and object classes. We have observed constant growth of the database over time and, recently, significant contributions from a variety of online users. The database is intended as a resource for the computer vision and computer graphics communities, with the images and annotations immediately available for download. In addition, search tools have been developed to interact with the database online.

In creating this database, we also intended that its use goes well beyond simply as a benchmark for computer vision algorithms. In this work, we presented recent results on directions towards this goal. Namely, we investigated the nature of the space of the images in the database and looked at how to recover additional information not directly provided by the online users. We demonstrated how to recover the 3D description of an image depicting a variety of scenes. Moreover, we showed that the output quality is similar to the output produced by a laser range scanner. We also analyzed the space of the images and observed properties of the distribution of the objects (e.g. Zipf's and power laws for the distribution of object labels present and scene n-grams, respectively).

In addition, there has been other recent work in computer vision and computer graphics that have utilized the database in creative ways. A recent trend has been to find, given a query image, other images with objects in a similar spatial configuration and to transfer the information associated with the retrieved images onto the query image. This has been used for intelligent insertion of objects into a scene [28] or object recognition in scenes [50], [34].

We believe that further creative uses of this database, along with the extension into video, offer promising directions for computer vision and computer graphics.

Acknowledgment

Funding for this research was provided by National Science Foundation Career award (IIS 0747120) and a National Defense Science and Engineering Graduate Fellowship.

References

[1] Y. Abramson and Y. Freund. Semi-automatic visual learning (seville): a tutorial on active learning for visual object recognition. In *CVPR*, 2005.
[2] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
[3] H. Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26. Academic Press, N.Y., 1978.
[4] E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *CVPR*, 2005.
[5] CBCL. Streetscenes. Technical report, 2006.
[6] M. Clowes. On seeing things. *Artificial Intelligence Journal*, 2(1):79–116, 1971.
[7] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *Intl. J. Computer Vision*, 40(2):123–148, 2000.
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.
[9] S. K. Divvala, A. A. Efros, and M. Hebert. Can similar scenes help surface layout estimation? In *IEEE Workshop on Internet Vision, at CVPR'08*, 2008.
[10] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *IEEE Intl. Conf. on Computer Vision*, pages 726–733, Nice, France, 2003.
[11] M. Everingham, A. Zisserman, C. Williams, L. V. Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, J. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang. The 2005 pascal visual object classes challenge. In *First PASCAL Challenges Workshop*. Springer-Verlag, 2005.
[12] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.
[13] L. Fei-Fei, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *IEEE Intl. Conf. on Computer Vision*, 2003.
[14] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.
[15] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.
[16] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *CVPR*, 2006.
[17] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *CVPR*, 2007.
[18] G. Griffin, A. Holub, and P. Perona. The Caltech-256. Technical report, California Institute of Technology, 2006.
[19] A. Gupta and L. S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*, 2008.
[20] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 26, 2007.
[21] J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008.
[22] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005.
[23] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *IEEE Intl. Conf. on Computer Vision*, 2005.
[24] D. Hoiem, A. Stein, A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *IEEE Intl. Conf. on Computer Vision*, 2007.
[25] B. Horn. The image dissector eyes. Technical report, Massachusetts Institute of Technology, 1971. Project MAC, Vision Flash 16, Cambridge.
[26] 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.
[27] D. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8:493–509, 1977.
[28] J. F. Lalonde, D. Hoiem, A. Efros, J. Winn, C. Rother, and A. Criminisi. Photo clip art. In *SIGGRAPH*, 2007.
[29] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.
[30] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
[31] I. Laptev and P. Perez. Retrieving actions in movies. In *IEEE Intl. Conf. on Computer Vision*, 2007.
[32] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *cvpr*, pages 2169–2178, 2006.
[33] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR*, 2007.
[34] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: label transfer via dense scene alignment. In *CVPR*, 2009.
[35] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: dense correspondence across different scenes. In *ECCV*, 2008.
[36] T. Malisiewicz and A. A. Efros. Recognition by association via learning per-exemplar distances. In *CVPR*, 2008.
[37] D. Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London*, B-275:483–524, 1976.
[38] V. Nedovic, A. Smeulders, A. Redert, and J.-M. Geusebroek. Depth information by stage classification. In *IEEE Intl. Conf. on Computer Vision*, 2007.
[39] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling

and photo editing. *SIGGRAPH 01*, 2001.
[40] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Intl. J. Computer Vision*, 42(3):145–175, 2001.
[41] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman. *Dataset issues in object recognition*. Springer, 2006.
[42] N. Quadrianto, L. Song, and A. J. Smola. Kernelized sorting. In *NIPS*, 2008.
[43] A. Quattoni and A.Torralba. Recognizing indoor scenes. In *CVPR*, 2009.
[44] A. Quattoni, M. Collins, and T. J. Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008.
[45] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 759–766, New York, NY, USA, 2007. ACM.
[46] X. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, 2006.
[47] H. A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In *Advances in Neural Info. Proc. Systems*, volume 8, 1995.
[48] B. Russell and A. Torralba. Building a database of 3d scenes from user annotations. In *CVPR*, 2009.
[49] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.
[50] 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.
[51] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *Intl. J. Computer Vision*, 77(1-3):157–173, 2008.
[52] 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.
[53] A. Saxenaa, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in Neural Info. Proc. Systems*, volume 18, 2005.
[54] H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *CVPR*, 2000.
[55] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, 2004.
[56] F. P. S. Service. http://www.flickr.com.
[57] H. Simon. On a class of skew distribution functions. *Biometrika*, 42:425–440, 1955.
[58] 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.
[59] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *IEEE Intl. Conf. on Computer Vision*, 2005.
[60] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, 2006.
[61] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):137–154, 2006.
[62] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *First IEEE Workshop on Internet Vision at CVPR 08*, 2008.
[63] M. Spain and P. Perona. Measuring and predicting importance of objects in our visual world. Technical report, California Institute of Technology, 2007.
[64] D. G. Stork. The open mind initiative. *IEEE Intelligent Systems and Their Applications*, 14(3):19–20, 1999.
[65] E. Sudderth, A. Torralba, W. T. Freeman, and W. Willsky. Learning hierarchical models of scenes, objects, and parts. In *IEEE Intl. Conf. on Computer Vision*, 2005.
[66] K. Sugihara. An algebraic approach to the shape-from-image-problem. *Artificial Intelligence Journal*, 23:59–95, 1984.
[67] 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.
[68] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. In *CVPR*, 2006.
[69] A. Torralba. How many pixels make an image? *Visual Neuroscience*, 26:123–131, 2009.
[70] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *IEEE PAMI*, 30(11):1958–1970, November 2008.
[71] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.
[72] A. Torralba and W. Fergus, R. Freeman. Tiny images, 2007.
[73] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007.
[74] A. Torralba and P. Sinha. Detecting faces in impoverished images. Technical Report 028, MIT AI Lab, 2001.
[75] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neuroscience*, 3(1):71–86, 1991.
[76] T. Vetter, M. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of object classes. In *CVPR*, 1997.
[77] S. Vijayanarasimhan and K. Grauman. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Info. Proc. Systems*, 2008.
[78] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proc. SIGCHI conference on Human factors in computing systems*, 2004.
[79] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects in images. In *In ACM CHI*, 2006.
[80] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *CVPR*, pages 101–109, 2000.
[81] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *IEEE Intl. Conf. on Computer Vision*, 2005.
[82] Z. Yao, X. Yang, and S. Zhu. Introduction to a large scale general purpose groundtruth database: methodology, annotation tools, and benchmarks. In *6th Int'l Conf on EMMCVPR, Ezhou, China*, 2007.
[83] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. Single view modeling of free-form scenes. In *CVPR*, 2001.
[84] G. K. Zipf. *The Psychobiology of Language*. Houghton Mifflin, 1935.