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**

I. Introduction

In the early days of artificial intelligence, the first challenge a computer vision researcher would encounter would be the difficult task of digitizing a photograph [27]. An excerpt from [40] illustrates this difficulty: "*This figure (-figure not shown here-) provides a high quality reproduction of the six images discussed in the text. a and b were taken with a considerably modified Information International Incorporated Vidissector, and the rest were taken with a Telenmation TMC-2100 vidicon camera attached to a Spatial Data Systems digitizer (Camera Eye 108).*" Even once a picture was in digital form, storing a large number of pictures (say six) consumed most of the available computational resources. In addition to the algorithmic advances required to solve object recognition, a key component to progress is access to data in order to train computational models for the different object classes. This situation has dramatically changed 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 practical solutions exist for a few classes (*e.g.* human faces and cars [78], [49], [56], [77]). Notable recent exceptions are the Caltech 101 dataset [15], with 101 object classes (later extended to 256 object classes [20]), the PASCAL collection [12] containing 20 object classes, the CBCL-street scenes database [8], comprising 8 object categories in street scenes, and the database of scenes from the Lotus Hill Research Institute [85].

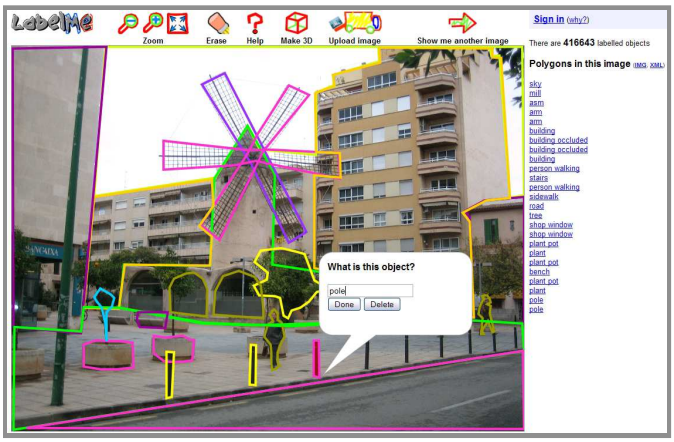

Fig. 1. Snapshot of the online application for image annotation.

Creating a large number of annotations for thousands of different object classes can become a time-consuming and challenging process. Because of this, 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 [79]. An implementation of this idea is the Seville project [4], where an 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 [80], 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 [14], [68], [84], [83], [16], [62], [51], [71], [18]. These works are characterized

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.

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 [81], [58], [53], [65], [67]. Recently, such efforts have shown to be successful. The Open Mind Initiative [67] 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 [58] 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 [81] 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 [82], has been created to provide location information of objects.

In 2005 we created LabelMe [53], 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. 1).

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 present an ordering and visualization of scenes in the real world. 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. Online annotation

Fig. 1 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 popup 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.

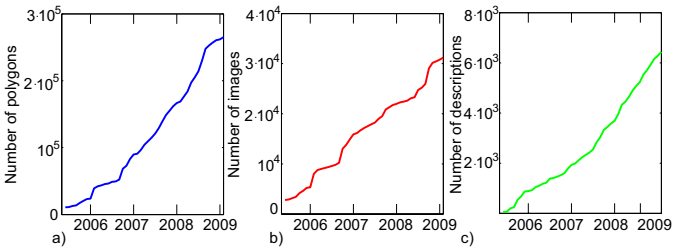

Fig. 2. Evolution of the dataset since the annotation tool came online in August 2005 through 2009. The horizontal axis denotes time (each mark is the beginning of the year), and the vertical axis represents: a) Number of annotated objects, b) Number of images with at least one annotated object, c) Number of unique object descriptions.

A. Dataset evolution and distribution of objects

Fig. 2 plots the evolution of the dataset since it went online in 2005. Fig. 2.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. 2.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. 2.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 [66] and seems to indicate that the number of visual object categories is large.

Fig. 3.b shows examples of the most frequently annotated object classes in our database, along with their segmentation masks. Fig. 3.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

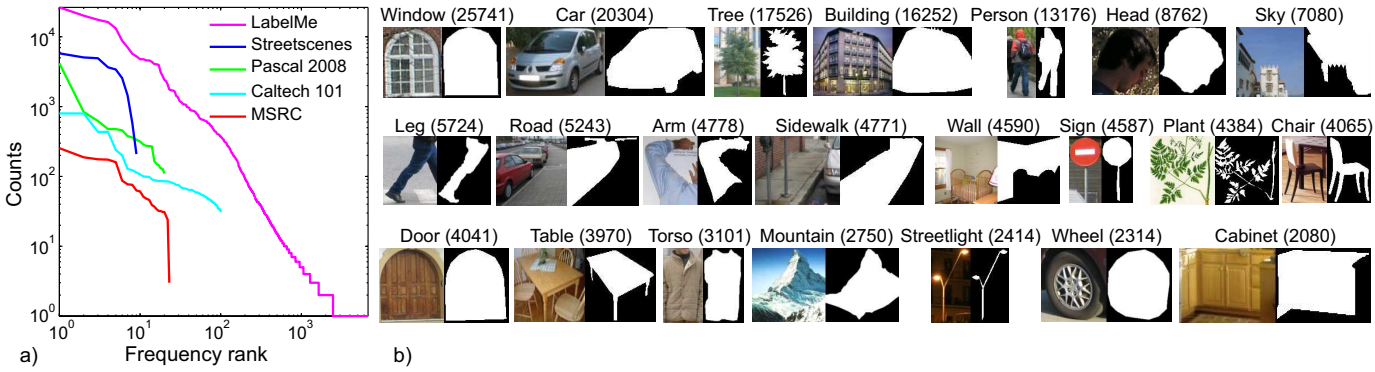

Fig. 3. a) Distribution of annotated objects in the LabelMe collection and comparison with other datasets. b) Examples of the most frequent objects in LabelMe. The number in parenthesis denotes the number of annotated instances. Those numbers continue to evolve as more objects are annotated every day.

follows Zipf's law [87], 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 [66], [73].

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 [15], [76], [7], [68], [47], [46], [13], [31].
- 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 IV, we discuss recent work in computer vision to exploit millions of image examples.

Note, however, as illustrated in Fig. 3.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.

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

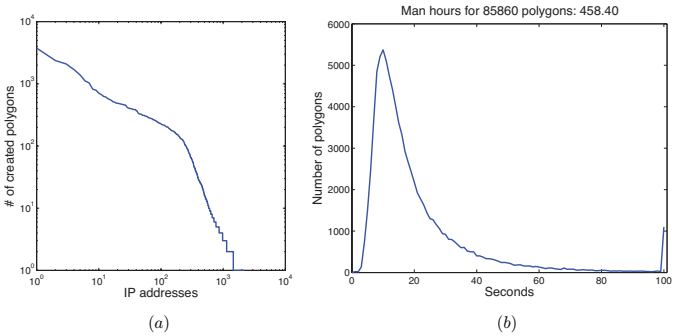

Fig. 4. (a) Number of new annotations provided by individual users of the online annotation tool from July 7th, 2008 through March 19th, 2009 (sorted in descending order, plotted on log-log axes). In total, 11382 unique IP addresses interacted with the labeling tool, with over 200 different IP addresses 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. 4(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.

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

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.

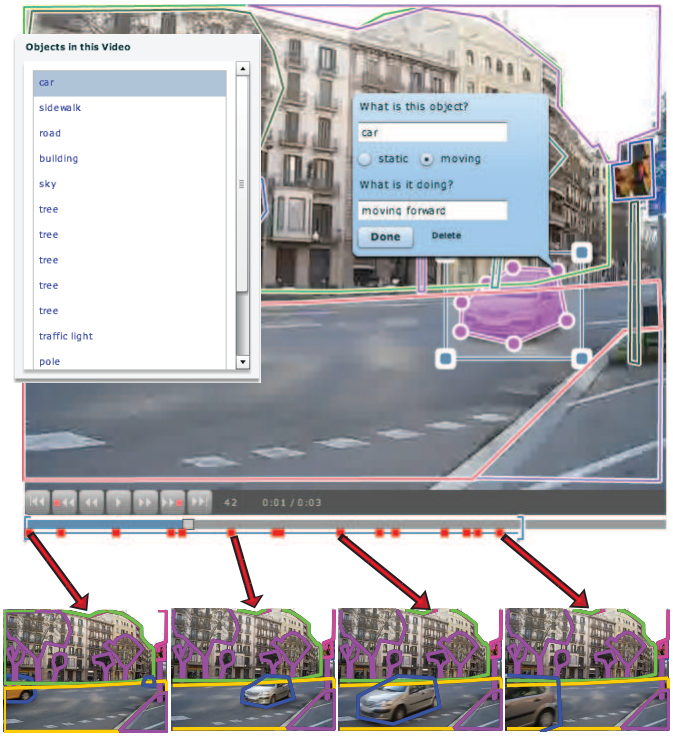

Fig. 5. A snapshot of our video annotation tool exemplifying a fully labeled example and some select key frames. Static objects are annotated in the same way as in LabelMe and moving objects require some minimal user intervention (manually edited frames are denoted by the red squares in the video track).

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. 4(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, 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.

C. Video annotation

The introduction of annotated image databases like LabelMe has contributed to the advancement of various areas in computer vision, such as object, scene, and category recognition. In the video domain, there have been efforts to collect datasets for benchmark and training purposes. Most of the currently available video datasets fall into one of two categories: (i) moderately annotated small datasets containing a rich, yet small set of actions [33], [32], [36], [57], and (ii) very specialized or minimally annotated, large databases mostly containing many hours of television or surveillance data [63], [2], [3], [17], [59].

Inspired by the concept of an online annotation tool, we created an openly accessible annotation tool for video, which creates a medium for researchers and volunteers to easily upload and/or annotate moving objects and events, with potential applications in research areas like motion estimation, object, event, and action recognition, amongst others. 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. 5 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.

III. 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. 6. Our system can

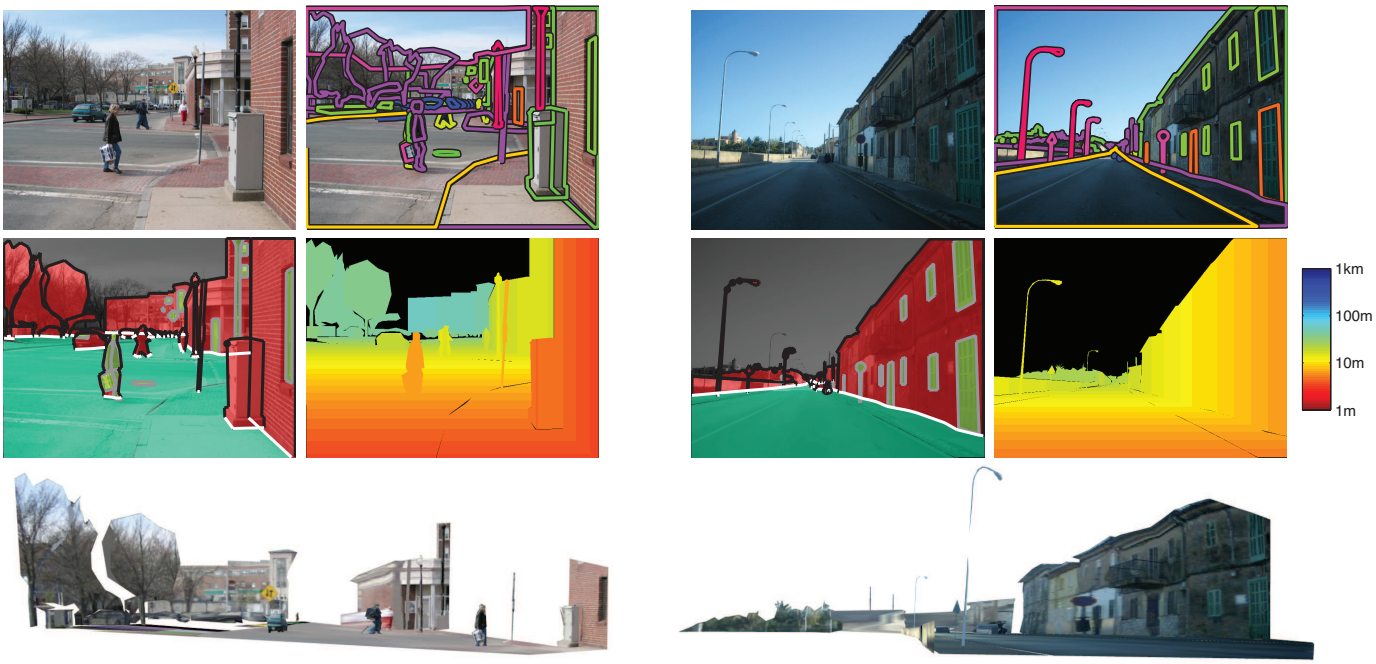

Fig. 6. We can recover 3D information from the user annotations. We show outputs for two input images. Top-left: Input image. Top-right: User annotations provided for the image. Middle-left: Recovered polygon and edge types. Polygons are either *ground* (green), *standing* (red), or *attached* (yellow). 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.

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.

Previous work has explored ways of associating 3D information to images. For example, there are existing databases captured with range scanners or stereo cameras [55], [54]. However, these databases tend to be small and constrained to specific locations due to the lack of widespread use of such apparatuses. Recent efforts have attempted to overcome this by manually collecting data from around the globe [1].

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 [55] or the orientation of major scene components, such as walls or ground surfaces, from a variety of image features [24], [25], [26]. 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 [64]. 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 [23], [61], [11], [48], [70], [35], [21], [41], [86].

An alternative approach is to ask humans to explicitly label 3D information [28], [10], [42]. 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 [5], [9], [6], [29], [69].

- **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

Fig. 7. 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.

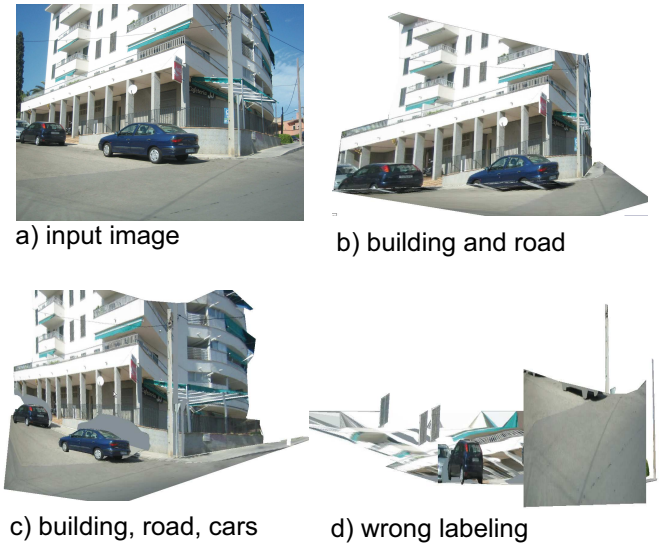

Fig. 8. As we the user adds more annotations, the quality of the reconstruction improves. a) input image, b) 3D model after the user annotated the *road* and the *building*, c) model obtained after adding the *car* to the list of annotated objects. d) Reconstructed model when the labels are incorrectly introduced so that the building is labeled as a road and vice versa.

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. To decide when an object category is part-of another, we evaluate the frequency of 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 overlap with 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 short.

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. 6, 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 label 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. Cues that are important for finding contact edges are edge length, orientation, and distance to a support object. Fig. 6 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. 6.

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 [30], 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 people. 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 [30], [50] for more details.

We show output depth maps of our system in Fig. 6. 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. 7, 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 [55]. 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. The baseline has relative error of 0.33 ± 0.04. Overall, we obtained less noisy outputs than the laser range scanner and 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. 8, 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

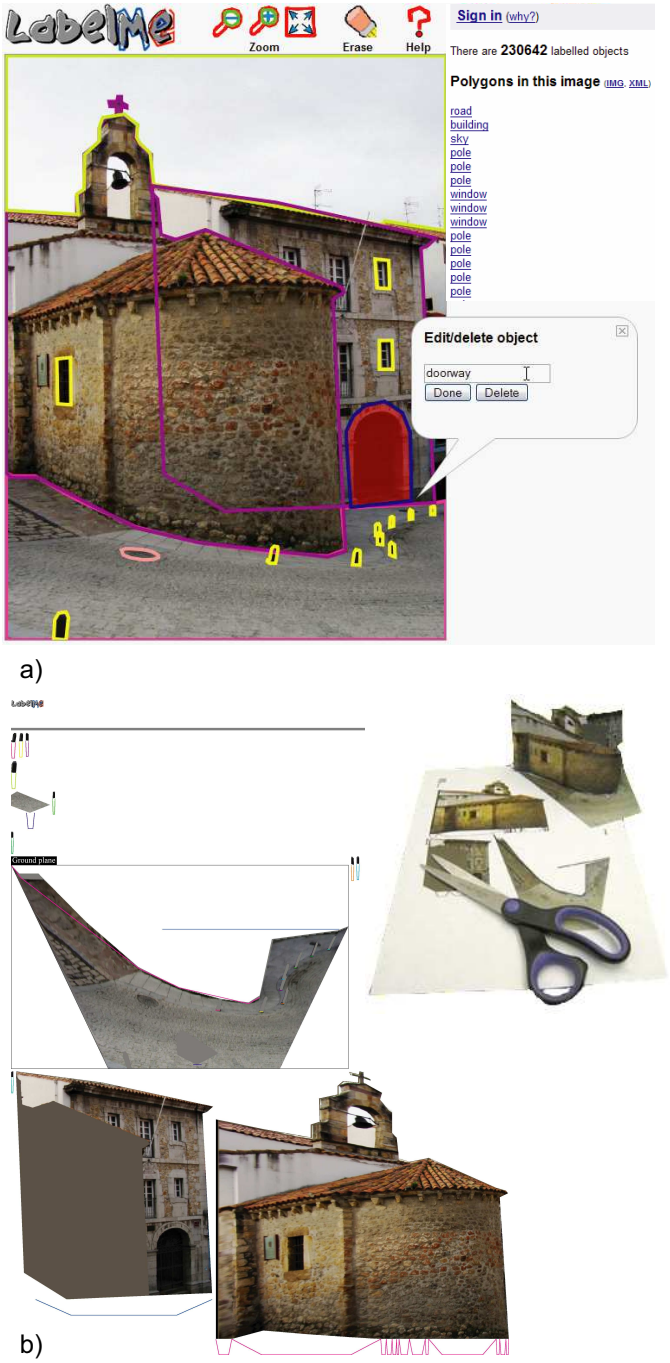

Fig. 9. Automatically generated instructions for a "do-it-yourself pop-up book" that can be constructed with paper, glue, and scissors.

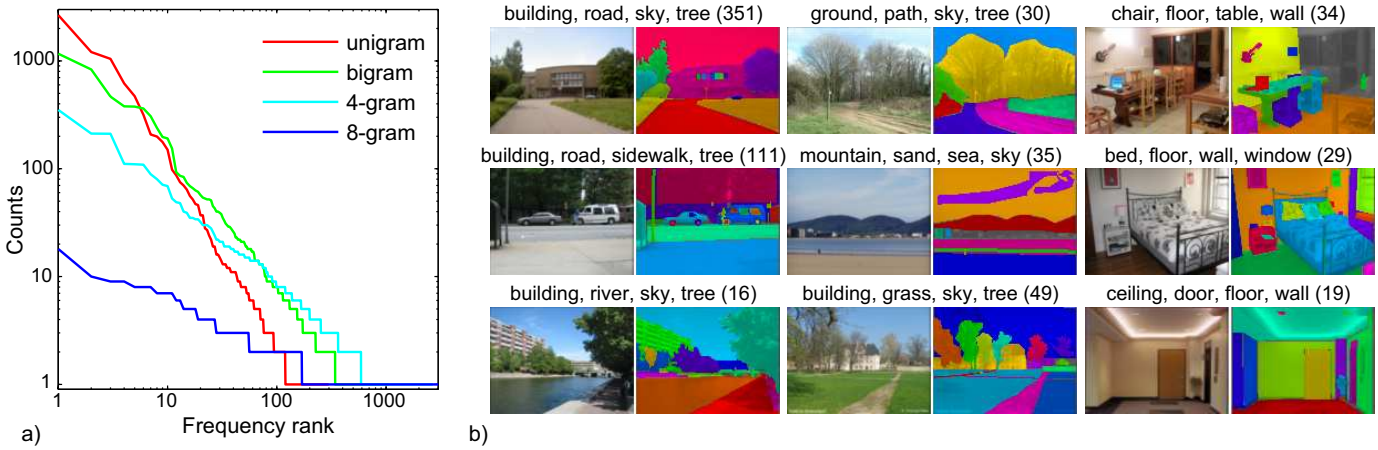

Fig. 10. a) Distribution of n-grams in LabelMe. Each n-gram corresponds to the list of n largest objects on each scene. b) Examples of scenes and 4-grams.

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. 9. 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.

IV. 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 [75], [38], [37], [39], [11] and graphics applications [64], [22], [61]. 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. Here, we are interested in how many different scenes there are. In this section, we study the space of different scenes in the database. Through our studies, an important question arises: does the dataset span a large number of different scene configurations?

A. Distribution of scenes

Here, we require a definition of what a scene is and when two scenes are considered similar. Without any constraints, there are as many scenes as pictures we can take. In cognitive psychology, studies on scene perception suggests that a representation of the scene might be composed of the scene category and 4 or 5 objects. In [72], it was shown that observers can recognize images at low resolution. In the extreme case where images have just 32x32 pixels, observers are able to recognize the scene category, together with 4-5 objects, with an accuracy of 80%. Therefore, we will define two images as being the same scene if the 4 largest objects depicted in an image belong to the same object categories (we will provide a more precise definition of image similarity later). 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. 10 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 [60]. 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. 10.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 4 large objects and ignoring their spatial organization. However, the space

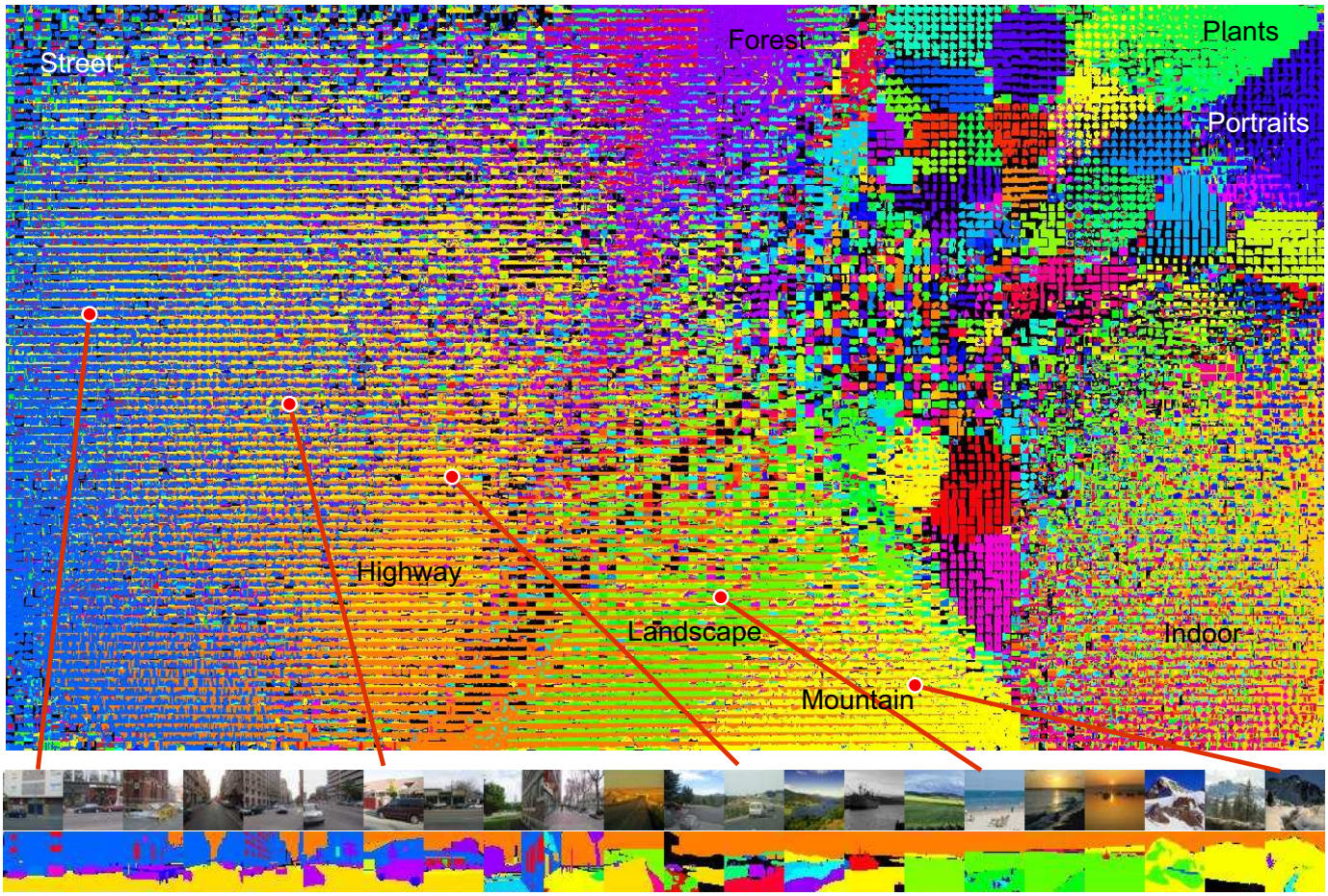

Fig. 11. The images are arranged according to semantic similarity between images (nearby images will contain similar objects in similar spatial configurations). Each thumbnail shows the object segments of each image color coded. Although there are some easily identifiable clusters in the space, most of the images are organized across a continuous space in which transitions across images are smooth.

of images is a continuous surface. Here, we will use a representation that will incorporate spatial information, along with all the objects present in the scene, in order to get a continuous organization of scenes. What we need first is to define the semantic distance between two images using the annotations. 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 on the histogram of object labels in the two images [74]. For this, we use spatial pyramid matching [34], [19] over object labels. This results in a simple similarity measure that takes into account the objects present in the image, in addition to their spatial organization. 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. 11 shows a visualization of 12201 images that are fully annotated from the LabelMe dataset. The images are organized according to semantic similarity: two nearby images are likely to contain the same object categories in similar spatial configurations. Each tile shows the segmentation of an image. Each object class has 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. 11 we used kernelized sorting [44]. 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 are organized across a continuous space in which transitions across images are smooth. The clusters that are visible in the figure 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

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

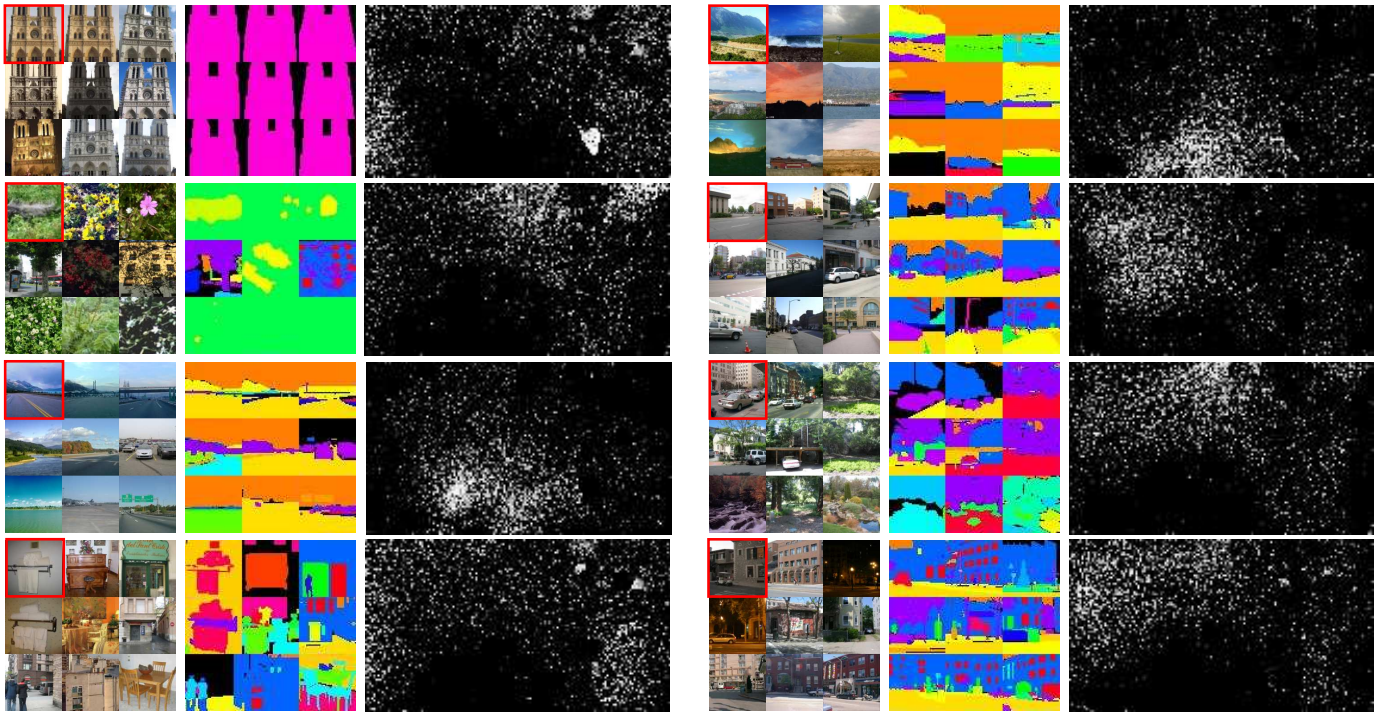

Fig. 12. Examples of input images and their nearest neighbors in the dataset using the GIST descriptor. For each panel: mosaic showing the query image (red box) and its 8 nearest neighbors. The objects within each image and the LabelMe map showing the location of the 1,000 closest images among the 12,201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

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. 11.

C. Recognition by scene alignment

As illustrated in Fig. 11, 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 [22], [74], [73].

In the examples used here, we use the GIST descriptor [43] 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 nonoverlapping 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 euclidian distance between GIST descriptors.

Fig. 12 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. 11. 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. 12, 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. 11).

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 [22], [74], [73], [38], [37]. 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. 12. 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.

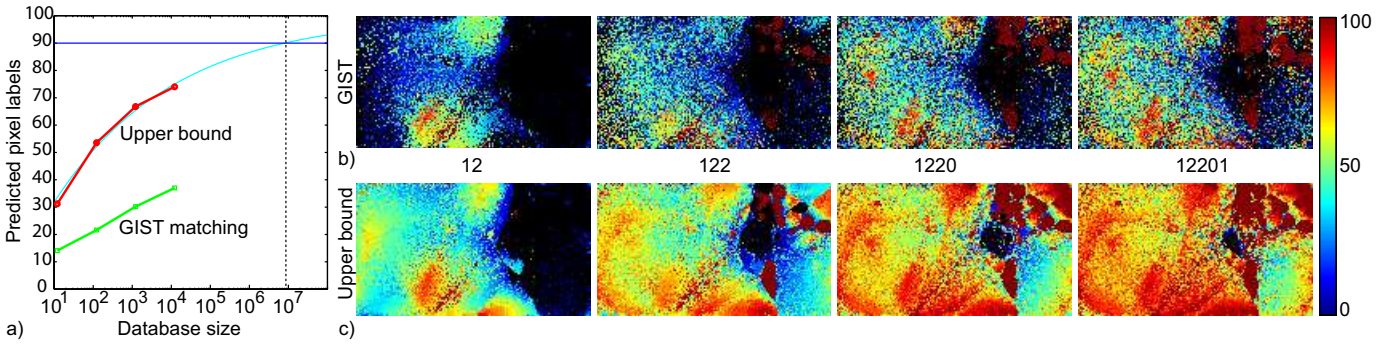

Fig. 13. a) Recognition performance as a function of dataset size. b) Distribution of the recognition performance in the different regions of the image space defined in Fig. 11.

In fig. 13 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. 11.

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 overfitting. Despite the simplicity of the nearest neighbor algorithm, we observe performance increases proportional to the database size, as shown in Fig 13.a.

The perfomance of this algorithm depends on the sampling density of the space images. Therefore, one can expect that the perfomance will vary depending on the regions of the space. In this study we can use the organization of scenes from Fig. 11 to visualize the distribution of errors. Fig 13.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 perfomances are found in the indoor scenes region. Indoor scenes remain challenging for many algorithms, with performance being low in general [45].

Fig. 13.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 significative labeling effort, this target database size is feasible.

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 scenes 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 go well beyond simply as a benchmark for computer vision algorithms. In this work, we presented recent results on directions that move toward 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 texture in-painting [22], intelligent insertion of objects into a scene [30] or object recognition in scenes [52], [73], [37].

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).

REFERENCES

[1] http://www.maps.google.com.
[2] *PETS 2001 Benchmark Data*. Online, 2001.
[3] *PETS 2006 Benchmark Data*. Online, 2006.
[4] Y. Abramson and Y. Freund. Semi-automatic visual learning (seville): a tutorial on active learning for visual object recognition. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR05), San Diego*, 2005.
[5] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
[6] H. Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26. Academic Press, N.Y., 1978.
[7] E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *CVPR*, 2005.
[8] CBCL. Streetscenes. Technical report.
[9] M. Clowes. On seeing things. *Artificial Intelligence Journal*, 2(1):79–116, 1971.
[10] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *Intl. J. Computer Vision*, 40(2):123–148, 2000.
[11] 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.
[12] 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.
[13] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.
[14] 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.
[15] 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.
[16] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.
[17] R. Fisher. *CAVIAR Test Case Scenarios*. Online Book, October 2004.
[18] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *CVPR*, 2006.
[19] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *CVPR*, 2007.
[20] G. Griffin, A. Holub, and P. Perona. The Caltech-256. Technical report, California Institute of Technology, 2006.
[21] A. Gupta and L. S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*, 2008.
[22] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 26, 2007.
[23] J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008.
[24] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005.
[25] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *IEEE Intl. Conf. on Computer Vision*, 2005.
[26] D. Hoiem, A. Stein, A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *IEEE Intl. Conf. on Computer Vision*, 2007.
[27] B. Horn. The image dissector eyes. Technical report, Massachusetts Institute of Technology, 1971. Project MAC, Vision Flash 16, Cambridge.
[28] 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.
[29] D. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8:493–509, 1977.
[30] J. F. Lalonde, D. Hoiem, A. Efros, J. Winn, C. Rother, and A. Criminisi. Photo clip art. In *SIGGRAPH*, 2007.
[31] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.
[32] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
[33] I. Laptev and P. Perez. Retrieving actions in movies. In *IEEE Intl. Conf. on Computer Vision*, 2007.
[34] 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.
[35] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR*, 2007.
[36] C. Liu, W. Freeman, E. Adelson, and Y. Weiss. Human-assisted motion annotation. In *CVPR*, pages 1–8, 2008.
[37] C. Liu, J. Yuen, and A. Torralba. Dense scene alignment using sift flow for object recognition. In *CVPR*, 2009.
[38] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: dense correspondence across different scenes. In *ECCV*, 2008.
[39] T. Malisiewicz and A. A. Efros. Recognition by association via learning per-exemplar distances. In *CVPR*, 2008.
[40] D. Marr. Early processing of visual information. In *Philosophical Transactions of the Royal Society of London*, pages 483–519, 1976.
[41] V. Nedovic, A. Smeulders, A. Redert, and J.-M. Geusebroek. Depth information by stage classification. In *IEEE Intl. Conf. on Computer Vision*, 2007.
[42] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. *SIGGRAPH 01*, 2001.
[43] 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.
[44] N. Quadrianto, L. Song, and A. J. Smola. Kernelized sorting. In *NIPS*, 2008.
[45] A. Quattoni and A.Torralba. Recognizing indoor scenes. In *CVPR*, 2009.
[46] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. pages 1–8, 2008.
[47] 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.
[48] X. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, 2006.
[49] H. A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In *Advances in Neural Info. Proc. Systems*, volume 8, 1995.
[50] B. Russell and A. Torralba. Building a database of 3d scenes from user annotations. In *CVPR*, 2009.
[51] 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.
[52] 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.
[53] 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.
[54] 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.
[55] 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.
[56] H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *CVPR*, 2000.
[57] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, 2004.
[58] F. P. S. Service. http://www.flickr.com.
[59] L. Sigal and M. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. 2006.
[60] H. Simon. On a class of skew distribution functions. *Biometrika*, 42:425–440, 1955.
[61] 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.
[62] 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.
[63] 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.
[64] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):137–154, 2006.
[65] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *First IEEE Workshop on Internet Vision at CVPR 08*, 2008.
[66] M. Spain and P. Perona. Measuring and predicting importance of objects in our visual world. Technical report, California Institute of Technology, 2007.
[67] D. G. Stork. The open mind initiative. *IEEE Intelligent Systems and Their Applications*, 14(3):19–20, 1999.
[68] 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.
[69] K. Sugihara. An algebraic approach to the shape-from-image-problem. *Artificial Intelligence Journal*, 23:59–95, 1984.
[70] 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.
[71] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. In *CVPR*, 2006.
[72] A. Torralba. How many pixels make an image? *Visual Neuroscience*, 26:123–131, 2009.
[73] 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.
[74] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.
[75] A. Torralba and W. Fergus, R. Freeman. Tiny images, 2007.
[76] 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.
[77] A. Torralba and P. Sinha. Detecting faces in impoverished images. Technical Report 028, MIT AI Lab, 2001.
[78] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neuroscience*, 3(1):71–86, 1991.
[79] T. Vetter, M. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of object classes. In *CVPR*, 1997.
[80] S. Vijayanarasimhan and K. Grauman. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Info. Proc. Systems*, 2008.
[81] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proc. SIGCHI conference on Human factors in computing systems*, 2004.
[82] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects in images. In *In ACM CHI*, 2006.
[83] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *CVPR*, pages 101–109, 2000.
[84] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *IEEE Intl. Conf. on Computer Vision*, 2005.
[85] 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.
[86] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. Single view modeling of free-form scenes. In *CVPR*, 2001.
[87] G. K. Zipf. *The Psychobiology of Language*. Houghton Mifflin, 1935.