A Fast Data Collection and Augmentation Procedure for Object Recognition

Benjamin Sapp and **Ashutosh Saxena** and **Andrew Y. Ng**
Computer Science Department,
Stanford University, Stanford, CA 94305
{bensapp,asaxena,ang}@cs.stanford.edu

Abstract

When building an application that requires object class recognition, having enough data to learn from is critical for good performance, and can easily determine the success or failure of the system. However, it is typically extremely labor-intensive to collect data, as the process usually involves acquiring the image, then manual cropping and hand-labeling. Preparing large training sets for object recognition has already become one of the main bottlenecks for such emerging applications as mobile robotics and object recognition on the web. This paper focuses on a novel and practical solution to the dataset collection problem. Our method is based on using a green screen to rapidly collect example images; we then use a probabilistic model to rapidly synthesize a much larger training set that attempts to capture desired invariants in the object's foreground and background. We demonstrate this procedure on our own mobile robotics platform, where we achieve $135\mathbf{x}$ savings in the time/effort needed to obtain a training set. Our data collection method is agnostic to the learning algorithm being used, and applies to any of a large class of standard object recognition methods. Given these results, we suggest that this method become a standard protocol for developing scalable object recognition systems.

Further, we used our data to build reliable classifiers that enabled our robot to visually recognize an object in an office environment, and thereby fetch an object from an office in response to a verbal request.

Keywords: Data-driven artificial intelligence, Computer vision, Robotics: Application.

Introduction

Many succesful real-world object recognition systems require hundreds or thousands of training examples, e.g., (Viola and Jones 2004; Wu, Rehg, and Mullin 2004; Fergus, Perona, and Zisserman 2003; LeCun, Huang, and Bottou 2004). The expense of acquiring such large training sets has been a bottleneck for many emerging applications. Indeed, the issue of training data quality and quantity is at the heart of all learning algorithms—often even an inferior learning algorithm will outperform a superior one, if it is given more data to learn from. Either developing better learning algorithms or increasing training set size has significant potential for improving object classification performance, and they may have equal impact on practical applications. This paper describes a method for rapidly synthesizing large training sets. We also apply these ideas to an office assistant robot application, which needs to identify common office objects so as to perform tasks such as fetching/delivering items around the office.

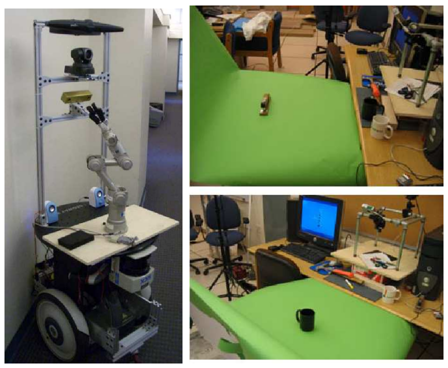


Figure 1: Left: Our mobile office assistant robot. Right: The green screen setup.

When building object recognition systems, data collection usually proceeds by first taking pictures of object instances in their natural environment, e.g., (Viola and Jones 2004; Dalal and Triggs 2005; Agarwal and Awan 2004). This involves either searching for the object in its environment or gathering the object instances beforehand and placing them, in a natural way, in the environment. The pictures are then hand-labeled either by marking bounding boxes or tight outlines in each picture. This is a tedious and time consuming process, and prohibitively expensive if our goal is a system which can reliably detect many object classes. The quality of the dataset is also heavily dependent on human collection and annotation performance.

In our approach, we start by rapidly capturing green screen images of a few example objects, and then synthesize a new, larger dataset by perturbing the foreground, background and shadow of the images using a probabilistic model. The key insight of our procedure is that we can model the true distribution of an object class roughly as well as real data, by using synthetic data derived from real images. Our method also automatically provides highly accurate labels in terms of the bounding box, perspective, object

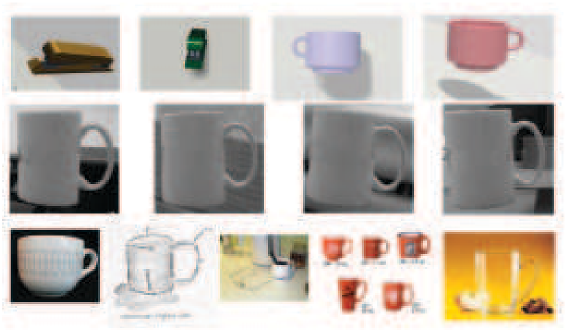


Figure 2: Other data collection techniques considered. Top row: Our synthetically generated 3D models. Middle row: Examples of warping a mug using perspective transformations. Bottom row: First page results of a Google image search for "coffee mug."

shape, etc. This is done at a tiny fraction of the human effort needed compared to traditional data collection protocols.

In this paper, we also explore the space of data collection techniques, and elucidate which design choices (such as manipulations of the foreground, background and shadow components of an image) result in the best performance. Although our method described is simple to implement, it pragmatically enables scaling object recognition to real world tasks and thus is timely (since deploying such systems is now within reach) and important. We propose that by adopting this data collection procedure, most application developers will be able to significantly reduce (by over $100\mathbf{x}$) the human effort needed to build object recognition systems.

Related Work

The problem of acquiring data is quickly becoming a critical issue in object recognition, as evidenced by several large-scale online manual annotation efforts created recently, e.g., LabelMe (Russell et al. 2005), Peekaboom (von Ahn, Liu, and Blum 2006) and Google Image Labeler (Google 2005). While promising, these systems still require manual labor, which is both time-consuming and error-prone.

A few researchers have explored training vision algorithms using synthetic computer graphics data. (Michels, Saxena, and Ng 2005) used synthetic images generated from 3D models of outdoor scenes for autonomous driving. (Heisele et al. 2001) and (Everingham and Zisserman 2005) used 3D morphable models for training face classifiers. (Agarwal and Triggs 2006) used human models on real backgrounds for pose estimation. (Saxena et al. 2006; Saxena, Driemeyer, and Ng 2008) used synthetic images to learn to grasp objects. (Black et al. 1997) used synthetic optical flow to learn parametrized motion models. Synthetic images can also be generated by morphing real images. For example, (Rowley, Baluja, and Kanade 1998), (Roth, Yang, and Ahuja 2000) and (Pomerleau 1991) perturbed training images (rotations, sheering, etc.) to enlarge the training set.

ETH-80 (Leibe and Schiele 2003) and COIL-100 (Nayar, Watanabe, and Noguchi 1996) were also collected using green screen capture methods. However, these methods resulted in non-cluttered, monochrome backgrounds, and therefore systems trained on these datasets fare poorly in a real-world environment (see Experiments and Results section). (LeCun, Huang, and Bottou 2004) also used an automated image capture system to create a large collection of images containing objects uniformly painted green. They focused on recognition of shape, testing their methods on their synthetic test set containing objects in the same setting—uniform color. Although this work evaluates performance of shape-based methods, it does not address object recognition in real world environments—when the objects are not "uniform green"—against real, cluttered backgrounds.

Although the use of green screens to segment foreground from background is a fairly common technique used in TV weather broadcasts and movie studios (Smith and Blinn 1996), our research differs from others in that we are leveraging synthetic data specifically to improve performance on real world data. To our knowledge, our paper is the first to empirically show that green screen data collection is an effective technique for creating training sets for real-world object recognition. Our data manipulation and augmentation techniques used to achieve this are similarly novel.

Other Collection Approaches

In our application work, we also considered three other standard collection procedures: Internet images, synthetic 3D models, and using affine transformations to augment the data. (See Fig. 2.)

(Ponce et al. 2006) give an excellent overview of large, publicly available data sets and annotation efforts. Several of these datasets (e.g., (Torralba, Murphy, and Freeman 2004; Everingham 2006; Fei-Fei, Fergus, and Perona 2006)) were created by web image search. (Griffin, Holub, and Perona 2007) reported that Google yields, on average, 25.6% *good* images (for Caltech-256; (Fergus et al. 2005) gives similar statistics). We found that this rate was much too optimistic for collecting a real world dataset, if we define a *good* example to be a qualitatively good representative of the object class, without occlusion, *in a real-world environment*. Most images returned by an Google Image search for office objects are from online product listings, and show the image against a monochrome or near monochrome background; this is not suitable for training a real-world recognition system.[1]

We also considered using 3D models of objects to synthesize images. We generated images as in the top row of Figure 2 using a computer graphics ray tracer, which models real world texture, shadows, reflections and caustics. Unfortunately, these images were insufficiently similar to real images, and in our experiments classifiers trained on them performed poorly compared to our procedure. Generating 3D models is also time consuming (about 1 hour for simple models such as a coffee mug; about 2.5 hours for a stapler). This is not scalable for building vision systems that recognize hundreds of different object classes. More impor-

[1]For example, in the category "watch" in Caltech-256, only 21 out of the 201 watch images are in a natural/realistic setting. In fact, for our 10 categories, only 2 (coffee mugs and keyboards) returned more than 100 *good* results. A Google Image search for "hammer," for example, yielded only 7 results found to be *good* out of all 948 images returned!

tantly, even having generated one coffee mug model (say), we found it extremely difficult to to perturb the model so as to obtain a training set comprising a diverse set of examples of mugs.

Finally, we ran experiments comparing real data vs. augmented synthetic and real data sets that were generated by perturbing the training images via different affine transformations. Applying these transformations to examples generated with our procedure did not improve test set performance, probably because our data already contained a large range of natural projective distortions (such as those obtained simply by our moving the object around on the green screen when collecting data).

Properties of Real Images and Data Augmentation

We consider a real image of an object as comprising object and background components. The object component itself comprises a shape component, together with an interior texture/color. For many object classes, the interior texture/color varies widely among object instances. For example, coffee mugs have a large variety of colors, pictures, patterns, and textures printed on them. Hammer handles may be made of wood, metal, plastic, or arbitrarily textured rubber, and also come in many colors. (Object classes that do contain useful interior features, such as faces and keyboards, also exist.)

Because the space of possible interiors is so large for most object classes, it is difficult for most learning algorithms to achieve invariance towards it. Having more training data often increases classifier performance by making it robust to different backgrounds or towards the texture on the specific instance of the object (such as the design on a coffee mug). In our method, we will compose synthetic examples by automatically combining different foregrounds, textures, backgrounds and shadows. One foreground can be paired with m backgrounds to create m training examples. To make our synthetic images as close as possible to real images, we will also develop a probabilistic method that learns how best to synthesize the artificial examples.

Experimental Setup

We collected data for 10 common office objects: coffee mugs, staplers, scissors, pliers, hammers, forks, keyboards, watches, cell phones and telephones. We used 10-12 objects per class, from which we collected roughly 200 images using the standard data collection procedure. For this, we took pictures of the objects against a variety of backgrounds in a real office environment, placing the objects in a natural way on desks and bookshelves. Based on logs kept (calculated by the timestamps on the files to calculate the time for each data collection session), this took on average 106 minutes per object class. Hand labeling took an additional 69 minutes per class on average.

We also collected roughly 200 images of each object class using our collection procedure. (Figure 3a,b). Here, we simply placed each object on the green screen, pressed a key to capture an image, moved the object to a different orientation and location on the green screen, and repeated 20 times for each object. Having all the objects nearby allows the human

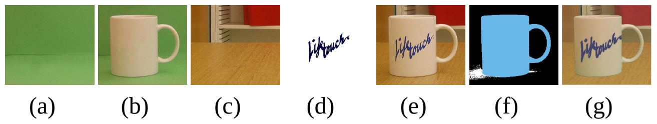


Figure 3: (a) I_{GRN}, (b) I_{OBJ}, (c) I_{BG}, (d) I_{FG}, (e) S, a real (non-synthetic) image obtained by placing mug with logo I_{FG} in background I_{BG}, (f) the foreground (blue) and shadow mask (white) obtained, (g) $\hat{S}$, the image synthetic image inferred using our probabilistic model.

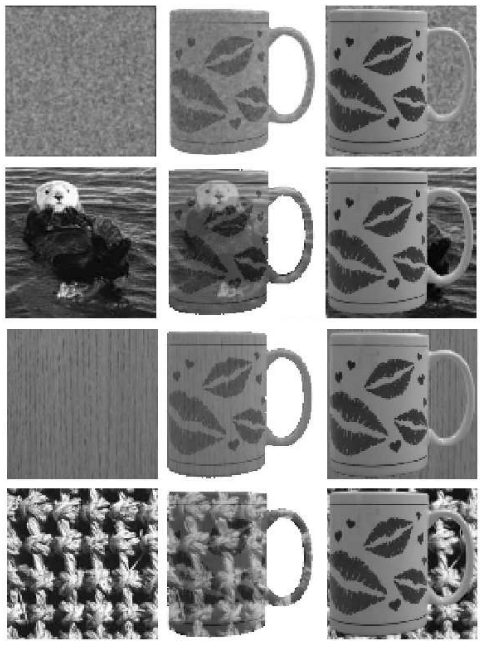


Figure 4: Different background and foreground manipulation techniques. Each row corresponds to a different I_{FG}/I_{BG} technique: *noise*, *corel*, *office* and *NRT*. The left column has instances of each technique. The middle column contains examples of each technique used for I_{FG}, fixing I_{BG} to be *white*. The right column uses each technique for I_{BG}, fixing I_{FG} to be *unaltered*.

operator to move no more than an arm's length for the entire process. Data collection took 19 minutes per class on average, resulting in a speedup of $\sim 9.2\mathbf{x}$.[2] We have made the real and green-screen data available at:
http://stair.stanford.edu/data.php

Probabilistic Model for Data Synthesis

We now describe our probabilistic model for learning how to accurately synthesize new images. We model the distribution of real images of objects as a generative model over the different components of the image—foreground, background, and texture. We define the following random variables:

I_{GRN}	The blank green screen image (e.g., Fig. 3a).

[2]The bottleneck for the process was the time it took our off-the-shelf digital camera to take a high resolution picture, store the image to flash memory, and communicate with the software; this took a few seconds per image. With better hardware, we could reduce synthetic data collection to 5 minutes for 200 images.

I_{OBJ} An object image captured against the green screen (e.g. the mug image in Fig. 3b).
I_{BG} A background image (e.g., an office environment, Fig. 3c).
I_{FG} A foreground texture image (e.g., a logo on the mug in Fig. 3d).
S A real (non-synthetic) image obtained by placing an object (from I_{OBJ}) with texture (I_{FG}) against a background (I_{BG}).
These random variables have a joint distribution. We model pixel i in the real image S, given the other images, as

$$\begin{aligned} & P(S_i|I_{OBJ_i}, I_{FG_i}, I_{BG_i}, I_{GRN_i}) \\ &= \sum_{m_i \in \{fg,bg,sh\}} P(S_i|m_i, I_{OBJ_i}, I_{FG_i}, I_{BG_i}, I_{GRN_i}) \\ &\qquad \cdot P(m_i|I_{OBJ_i}, I_{FG_i}, I_{BG_i}, I_{GRN_i}) \end{aligned} \tag{1}$$

Here, m_i is a random variable taking on the values $\{fg, bg, sh\}$, and indicates whether the i-th pixel is part of the foreground, background or shadow component ($\sum_{i=fg,bg,sh} P(m_i|I_{OBJ_i}, I_{FG_i}, I_{BG_i}, I_{GRN_i}) = 1$). Whether the pixel is foreground (part of object), shadow or background, depends only on I_{GRN} and I_{OBJ}, therefore

$$\begin{aligned} P(m_i|I_{OBJ_i}, I_{GRN_i}, I_{FG_i}, I_{BG_i}) &= P(m_i|I_{OBJ_i}, I_{GRN_i}) \\ &\propto P(I_{OBJ_i}, I_{GRN_i}|m_i)P(m_i) \\ &\propto P(I_{GRN_i}|m_i, I_{OBJ_i})P(I_{OBJ_i}|m_i)P(m_i) \end{aligned} \tag{2}$$

We model $P(I_{OBJ_i}|m_i)$, for the background and shadow components ($m_i = fg$ and $m_i = sh$), each as a mixture of n Gaussians, with the same covariance (Σ_{bg} for background and Σ_{sh} for shadow respectively), and with different means for each mixture component (μ_{bg_j} for background and μ_{sh_j} for shadow respectively, for $j = 1, ..., n$). The foreground is modeled with a large-variance Gaussian: $P(I_{OBJ_i}|m_i = fg) = \mathcal{N}(I_{OBJ_i}; \mu_{fg}, \Sigma_{fg})$. Here, $\mathcal{N}(x; \mu, \Sigma)$ denotes the density for a Gaussian with mean μ and covariance Σ.

We also model I_{GRN_i} given m_i and I_{OBJ_i} as Gaussian, with mean depending on whether pixel i is background or shadow: $P(I_{GRN_i}|m_i = bg, I_{OBJ_i}) = \mathcal{N}(I_{GRN_i}; I_{OBJ_i}, \Sigma_1)$, and $P(I_{GRN_i}|m_i = sh, I_{OBJ_i}) = \mathcal{N}(I_{GRN_i}; I_{OBJ_i} + \mu_s, \Sigma_2)$. Our model for I_{GRN_i} conditioned on pixel i being shadow ($m_i = sh$) takes into account the darkening effect that the shadow has on the green screen background pixels; thus, the distribution has mean around a brighter value $I_{OBJ_i} + \mu_s$, rather than mean I_{OBJ_i}. (I.e., with the object removed, the shadow pixels become some amount μ_s brighter on average.) Because the color of the green screen does not depend on I_{OBJ_i}, the distribution $P(I_{GRN_i}|m_i = fg, I_{OBJ_i}) = P(I_{GRN_i}|m_i = fg)$ can also be modeled as a mixture of Gaussians with the same variance Σ_{bg}, and with separate μ_{bg_j}. Finally, we used a uniform prior $P(m_i)$, which in practice was sufficient for our green screen setting.

To obtain realistic images, we blend the existing component with a new component. For example, we blend the object image with the new object texture, or the existing background with a new background. We represent these combinations using a multivariate Gaussian and a linear blending of I_{OBJ_i} with I_{FG_i} or I_{BG_i}:

$$\begin{aligned} & P(S_i|m_i = fg, I_{OBJ_i}, I_{FG_i}; w_{fg}) = \\ &\qquad (1/Z) \exp\left(-\|S_i - {w_{fg}}^T [I_{OBJ_i}, I_{FG_i}]^T\|^2/2\right) \\ & P(S_i|m_i = bg, I_{OBJ_i}, I_{BG_i}; w_{bg}) = \\ &\qquad (1/Z) \exp\left(-\|S_i - {w_{bg}}^T [I_{OBJ_i}, I_{BG_i}]^T\|^2/2\right) \\ & P(S_i|m_i = sh, I_{OBJ_i}, I_{BG_i}; w_{sh}) = \\ &\qquad (1/Z) \exp\left(-\|S_i - {w_{sh}}^T [I_{OBJ_i}, I_{BG_i}]^T\|^2/2\right) \end{aligned} \tag{3}$$

Parameter Learning and MAP Inference

To learn the parameters $w = [w_{fg}, w_{bg}, w_{sh}]$ of our model, we began by collecting a ground-truth dataset with component images $I_{GRN}, I_{OBJ}, I_{BG}, I_{FG}$, and real (non-synthetic) images S of the object placed in the background, together with hand-labeled masks m for the shadow, foreground and background. We learn the parameters w of our model by maximizing the conditional log-likelihood $\log P(S_i|I_{OBJ_i}, I_{FG_i}, I_{BG_i}, I_{GRN_i}, m_i; w)$. Since Eq. 3 is Gaussian, w can be estimated in closed form. The parameters of $P(I_{OBJ_i}|m_i)$ and $P(GRN_i|m_i, I_{OBJ_i})$ were estimated from the ground-truth images—estimating the empirical mean and covariance for the Gaussian distribution, with user-initialized points for estimating the parameters of the mixture of Gaussians, including n, the number of Gaussians in the mixture.[3]

Given individual image components and the learned parameters, MAP inference of image $\hat{S}$ is straightforward, and can be derived in closed form by solving $\hat{S}_i = \arg\max_{S_i} P(S_i|I_{OBJ_i}, I_{FG_i}, I_{BG_i}, I_{GRN})$. This therefore gives a framework for creating large amounts of synthetic data from an image I_{OBJ_i} by sampling from a large set of individual components I_{FG} and I_{BG}.

To explore the effectiveness of different interior and background manipulation methods, we created datasets with a variety of w, described in Table 2. We found that modeling shadows had no significant impact on results, and thus are not included in our experiments.

Experiments and Results

We now evaluate our synthetic generation techniques. Although this work concerns synthetic generation of datasets, all test sets are *real data* comprising actual (non-synthetic) images either from standard datasets, or from our data collection of objects in their natural environments.

Office Object Classification

For classifying the office object categories, we used a supervised learning algorithm that uses the first band of C1 features (Serre, Wolf, and Poggio 2005), and a Gentleboost ensemble of 200 weak classifiers.[4] This algorithm was fast enough to permit repeated experiments, while still achieving results comparable to state-of-the-art recognition. Although our experiments used a standard combination of features and

[3]In practice, n was typically set to 4 or 5.

[4]Details: Each weak classifier is a decision stump formed by thresholding a single C1 feature. Changing the size of the ensemble or using decision trees instead of stumps gave similar results to those reported here.

Background Variations										Foreground Variations								
Object	unaltered	black	white	uniform	noise	corel	office	NRT		Object	unaltered	black	white	uniform	noise	corel	office	NRT
mug	.727	.566	.530	.521	.882	.874	**.904**	.827		mug	.727	.496	.497	.608	**.959**	.956	.954	.937
scissor	.822	.529	.516	.505	.937	.951	**.952**	.899		scissor	.822	.939	.885	.691	.954	**.967**	.941	.941
stapler	.760	.550	.568	.504	.846	.862	**.913**	.860		stapler	.760	.780	.749	.507	.942	**.957**	.953	.911
keyboard	.954	**.961**	.948	.577	.944	.955	.950	.939		keyboard	.954	.500	.851	.850	**.980**	.978	.977	.972
hammer	.886	.511	.515	.565	.941	.949	**.977**	.921		hammer	.886	.962	.932	.742	**.987**	.972	.977	.973
plier	.709	.509	.552	.520	.883	.888	**.913**	.795		plier	.709	.819	.757	.652	.872	**.935**	.921	.930
fork	.612	.515	.517	.493	**.670**	.623	.668	.605		fork	.612	.618	.613	.537	**.806**	.758	.804	.720
watch	.900	.541	.519	.494	.967	**.970**	.965	.962		watch	.900	.496	.498	.746	**.974**	.971	.967	.969
flipphone	.729	.545	.574	.496	**.765**	.696	.757	.674		flipphone	.729	.497	.501	.554	**.868**	.830	.873	.803
telephone	.940	.500	.502	.694	.954	.942	.958	**.963**		telephone	.940	.491	.490	.615	**.989**	.982	.980	.973

Table 1: Performance comparison of different background and foreground techniques on the 10 office object dataset. The foreground is fixed with technique *unaltered* in the background variation experiments; the background is fixed to *office* in the foreground experiments. Each entry is the result of 10-fold cross validation, with 50 examples in each of the positive and negative training and test sets.

Technique	$\mathbf{w}^T$	$I^T_{FG_i,BG_i}$	Description
unaltered	[1, 0]	–	Leave bg/fg from I_{OBJ} unaltered
white/black	[0, 1]	$[255, 255, 255]/[0, 0, 0]$	Replace bg/fg with white/black
uniform	[0, 1]	(u_1, u_2, u_3), $u_i \sim$ U(0,255)	Randomly sample each pixel uniformly
noise	learnt	I_{noise}	$I_{\text{noise}} = I_{\text{uniform}} * G$, G is a 3×3 Gaussian filter with $\sigma = 0.95$
corel	learnt	I_{corel}	Generic image database from the web, see Li03
office	learnt	I_{office}	Images collected from office environments.
NRT	learnt	I_{tex}	Near-Regular Texture database with 188 textures, see CMU-NRT

Table 2: A description of the various techniques used for I_{BG} and I_{FG}.

learning algorithm, we believe our data synthesis method applies equally well to other methods.[5]

Background and Foreground Techniques: Table 1 shows the effect of using different background techniques.

Using white/black backgrounds performed poorly; note that most standard image-collection methods such as Internet have white backgrounds. The "keyboard" class performs well regardless of background, because in most examples, the rectangular keyboard fills up the whole image, making background irrelevant. The office environment consistently performs the best or within 1% of the best technique across all 10 object classes. This indicates that background plays an important role in classifier performance.

Table 1 also shows the effect of using different foreground techniques. Note that adding random texture (*noise*, *corel*, *office*, *NRT*) to the object increases performance over using an unaltered object foreground, with *noise* and *corel* slightly better than the others. especially telephones, keyboards, and watch faces). This reflects different foreground textures increasing the diversity in the data.

Evaluation of Data Synthesis Techniques: Even though a synthetic example (e.g. in Fig. 3g) appears visually similar to a real one (Fig. 3e), a classifier trained on synthetic examples could still perform poorly on real test examples. Therefore, in this experiment, we compare a classifier trained on real examples to one trained on the same number of synthetic examples. The synthetic examples were created using the best technique from the previous experiments. Fig. 5 shows that our synthetic examples (red curve) are competitive with the performance on the real data (blue curve).

[5]Other details: Our algorithm was trained on balanced training sets. The real test set used 50 positive and 50 negative examples, and no test set object appears in any training image. Negative examples are sampled from the office environment. Unless otherwise specified, performance is measured as total accuracy: $\frac{Number\ correct}{Total}$. Each data point is an average of 10 trials of randomly sampled training and test sets.

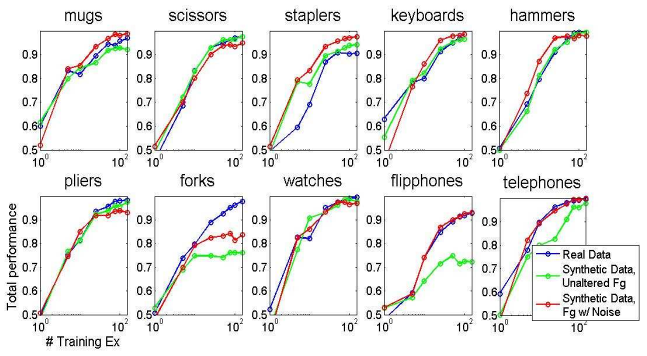


Figure 5: Evaluation of our synthesis techniques versus real data (blue curve). The red curve is foreground technique *noise*, the green curve, *unaltered*; both with background *office*.The synthetic curve matches the performance of the real data.

Data Set Augmentation: Figure 6 compares the performance of classifiers trained on real data (blue curve) versus classifiers trained on synthetic data created using the same number of real examples. In the synthetic curves, the dataset is always augmented to create 150 examples. E.g., at the point where the number of training examples is 50, each example is paired with three different backgrounds (and the foreground perturbed uniquely three different ways), to create 150 examples. If we do not change the foreground while augmenting the synthetic data (green curve), then the performance only marginally improves; this shows the utility of our method in achieving diversity in the data.

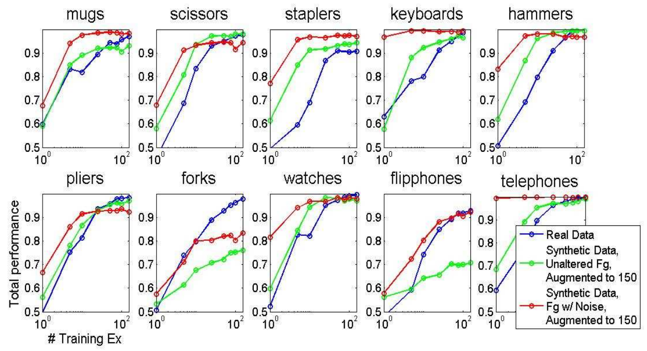


Figure 6: Improvement of synthetic dataset augmentation over real data. The synthetic red and green curves use the same techniques as in Fig. 5, however this time each number of original training examples is mapped to 150 synthetic examples, showing significant improvement over the real (blue) curve in most cases.

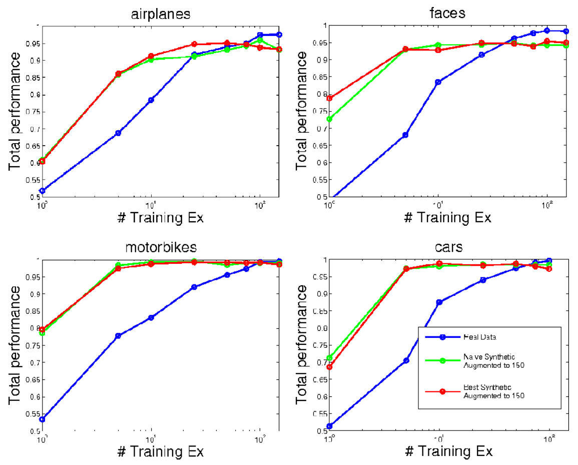


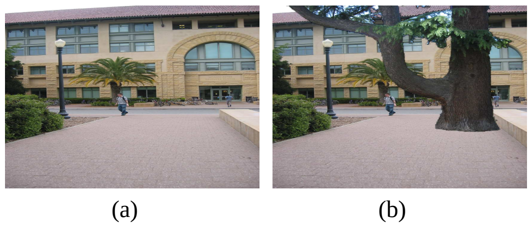
Figure 7: Improvement of synthetic dataset augmentation over real data on 4 categories of the Caltech-101. Synthetic methods (red and green curves) are the same techniques as in Fig. 6. Real data is the blue curve.

In half of the object classes tested, our data collection procedure applied to 10 examples achieves the same performance as 150 real examples. This reduces our work to collect data per object to $\sim$ 1.3 minutes, yielding a total speedup over the standard data collection procedure of roughly $135\mathbf{x}$.

We also evaluated our dataset augmentation technique on four categories from Caltech-101 (Fei-Fei, Fergus, and Perona 2006). (See Fig. 7.) Because our goal is to evaluate dataset augmentation, rather than do well on Caltech-101 per se, we focus on the relative performance of real versus augmented data. We used the object masks provided with the Caltech set. Since this set of objects has no consistent background, we used our office background technique. Despite this poor fit to these objects, a small number of training examples created using our method still beats using real data.

3-D Data Synthesis / Tree Classification

We now consider a second, outdoor-scene understanding problem, in which our goal is to classify each pixel in a test set as either *tree* or *non-tree*. Here, the training data consists of images along with aligned depth maps, while the test data consists of only images. We first apply our synthetic data augmentation techniques to place the object (tree) correctly in new backgrounds using 3D depth information, (Saxena, Sun, and Ng 2007) producing a much larger augmented training set. Specifically, our training data is a set of real examples with trees present $|T| = 24$, and a set of "non-tree" images $|N| = 103$, where the only trees present are small and in the background, e.g., Fig. 8(a). Using this real data, we can only achieve a training set size of $|N| + |T|$. However, our method generates a dataset of size $|N| \times |T|$. Fig. 8(c) shows that using our method, we obtain higher classification performance, measured as area under the ROC curve (AUC).



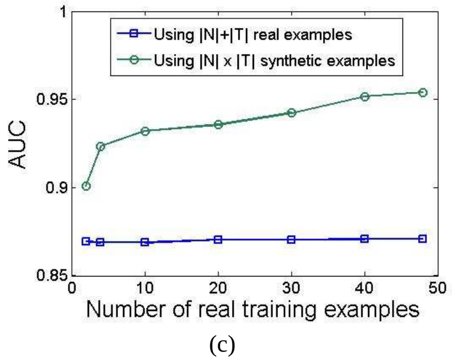


Figure 8: (a) An image with few tree pixels in set N. (b) A member of $N \times T$. (c) Learning curve improvement for synthetic augmented data.

In all test cases, the real training set consists of an equal number of tree and non-tree images. The real training set at most contains only 24 trees, fixed at their true positions and scales, and the information gain from going from 2 to 24 examples is marginal. The synthetic training set, however, has trees in a much more diverse set of realistic scales and positions, allowing the classifier to generalize better to unseen examples. This is true even for the first data point, $|N| = |T| = 2$, in which the real training set has 2 tree and 2 non-tree images, while the synthetic set has 4 tree images.

100,000 Examples

With the ease of our collection and augmentation techniques, we can push the upper limits of training set size. We collected images of 100 coffee mugs at different orientations (24 azimuths and 4 elevations) covering the entire upper hemisphere. We used these to synthesize training sets of up to 10^5 examples using different backgrounds and textures. We evaluated our data using an efficient implementation of k-NN using cover trees (Beygelzimer, Kakade, and Langford 2006) against the extremely challenging 'coffee-mug'

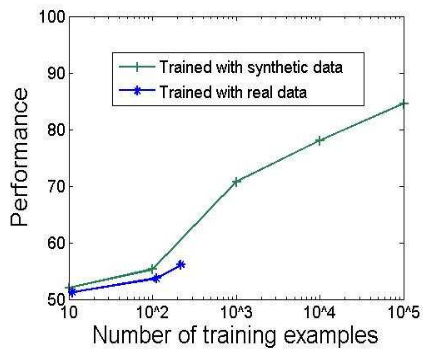


Figure 9: Synthetic generation techniques boost performance on the Caltech-256 category 'coffee-mug' using 10^5 examples.

category of the Caltech-256. We achieve a best performance of 84.58% accuracy with 10^5 examples. Unfortunately, our dataset is 450 times larger than the largest available real data set that we can use to compare to our method's performance. As a baseline, we include results trained on the 220 good samples of coffee mugs from the LabelMe database. (Both the LabelMe and Caltech-256 data sets were manually pre-filtered to remove noisy examples.) To actually collect 10^5 examples using standard techniques would have taken 1458 hours, based on our collection time analysis (see Experimental Setup section). Our method, on the other hand, took a total of 16 hours.

Fig. 9 shows that the performance of even a simple classification algorithm improves significantly with the amount of data. Thus, we believe our procedure holds promise to improve performance of learning algorithms in general.

Robotic Application

Finally, this work was driven by the real world problem of developing an efficient method to train an office assistant robot. We take our classifiers trained on the rapidly collected, manipulated, and augmented data and integrate them into our robotics platform. Our method was able to build reliable classifiers that enabled our robot to visually recognize an object in an office environment, and thereby fetch an object from an office in response to a verbal request. Video demonstration of these results is provided at:
http://stair.stanford.edu

References

Agarwal, S., and Awan, A. 2004. Learning to detect objects in images via a sparse, part-based representation. *IEEE PAMI* 26(11):1475–1490.

Agarwal, A., and Triggs, B. 2006. A local basis representation for estimating human pose from cluttered images. In *ACCV*.

Beygelzimer, A.; Kakade, S.; and Langford, J. 2006. Cover trees for nearest neighbor. In *ICML*. New York, NY, USA: ACM Press.

Black, M. J.; Yacoob, Y.; Jepson, A. D.; and Fleet, D. J. 1997. Learning parameterized models of image motion. In *CVPR*.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR*.

Everingham, M., and Zisserman, A. 2005. Identifying individuals in video by combining generative and discriminative head models. In *ICCV*.

Everingham, M. e. a. 2006. The 2005 pascal visual object classes challenge. In *Machine Learning Challenges*. Springer-Verlag.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2006. One-shot learning of object categories. *IEEE PAMI* 28(4):594–611.

Fergus, R.; Fei-Fei, L.; Perona, P.; and Zisserman, A. 2005. Learning object categories from google's image search. In *ICCV*.

Fergus, R.; Perona, P.; and Zisserman, A. 2003. Object class recognition by unsupervised scale-invariant learning. In *CVPR*.

Google. 2005. Google image labeler. http://images.google.com/imagelabeler.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset. Technical Report 7694, Caltech.

Heisele, B.; Serre, T.; Pontil, M.; Vetter, T.; and Poggio, T. 2001. Categorization by learning and combining object parts. In *NIPS*.

LeCun, Y.; Huang, F.-J.; and Bottou, L. 2004. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*.

Leibe, B., and Schiele, B. 2003. Analyzing appearance and contour based methods for object categorization. In *CVPR*.

Michels, J.; Saxena, A.; and Ng, A. Y. 2005. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*.

Nayar, S. K.; Watanabe, M.; and Noguchi, M. 1996. Real-time focus range sensor. *IEEE PAMI* 18(12):1186–1198.

Pomerleau, D. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural Comp* 3(1):88–97.

Ponce, J.; Hebert, M.; Schmid, C.; and Zisserman, A., eds. 2006. *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*. Springer.

Roth, D.; Yang, M.; and Ahuja, N. 2000. A SNoW-based face detector. NIPS.

Rowley, H.; Baluja, S.; and Kanade, T. 1998. Neural network-based face detection. *IEEE PAMI* 20(1):23–38.

Russell, B. C.; Torralba, A.; Murphy, K. P.; and Freeman, W. T. 2005. LabelMe: a database and web-based tool for image annotation. Technical report, MIT.

Saxena, A.; Driemeyer, J.; Kearns, J.; and Ng, A. Y. 2006. Robotic grasping of novel objects. In *NIPS*.

Saxena, A.; Driemeyer, J.; and Ng, A. Y. 2008. Robotic grasping of novel objects using vision. *IJRR* 27(2).

Saxena, A.; Sun, M.; and Ng, A. Y. 2007. Learning 3-d scene structure from a single still image. In *ICCV workshop on 3D Representation for Recognition (3dRR-07)*.

Serre, T.; Wolf, L.; and Poggio, T. 2005. Object recognition with features inspired by visual cortex. In *CVPR*.

Smith, A. R., and Blinn, J. F. 1996. Blue screen matting. In *SIGGRAPH*, 259–268.

Torralba, A.; Murphy, K. P.; and Freeman, W. T. 2004. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*.

Viola, P., and Jones, M. 2004. Robust real-time object detection. *IJCV* 57(2).

von Ahn, L.; Liu, R.; and Blum, M. 2006. Peekaboom: a game for locating objects in images. In *SIGCHI*.

Wu, J.; Rehg, J.; and Mullin, M. 2004. Learning a rare event detection cascade by direct feature selection. In *NIPS*.