Multi-Camera Object Detection for Robotics

Adam Coates Andrew Y. Ng

***Abstract*— Robust object detection is a critical skill for robotic applications in complex environments like homes and offices. In this paper we propose a method for using multiple cameras to simultaneously view an object from multiple angles and at high resolutions. We show that our probabilistic method for combining the camera views, which can be used with many choices of single-image object detector, can significantly improve accuracy for detecting objects from many viewpoints. We also present our own single-image object detection method that uses large synthetic datasets for training. Using a distributed, parallel learning algorithm, we train from very large datasets (up to 100 million image patches). The resulting object detector achieves high performance on its own, but also benefits substantially from using multiple camera views. Our experimental results validate our system in realistic conditions and demonstrates significant performance gains over using standard single-image classifiers, raising accuracy from 0.86 area-under-curve to 0.97.**

I. INTRODUCTION

Detecting classes of objects reliably remains a highly challenging problem in robotics. Robotic systems require extremely high accuracy for a large variety of object classes in order to perform even simple tasks (such as taking inventory of a few objects in a lab environment [1]) but off-the-shelf vision algorithms that achieve the necessary level of performance do not yet exist. Robots, however, have a number of advantages that are ignored by most detection systems. For instance, robots are able to view a real scene from multiple angles and can view interesting objects up close when necessary [2], [3], yet most existing algorithms cannot directly leverage these tools. In this work, we will present a method for improving detection accuracy by using multiple viewing angles and high resolution imagery that may be obtained either from multiple cameras or by moving the robot. We will show that this method significantly boosts detection accuracy and is a valuable tool for detecting objects whose appearance differs significantly with viewpoint.

Even considering a single object class from a single viewpoint (i.e., the same side of the object is always facing the camera), detecting classes of objects remains a daunting unsolved problem. Off-the-shelf vision algorithms can achieve good results for a handful of well studied classes, such as faces, cars, and pedestrians [4], [5], [6], [7], but even these detectors can have high false positive rates when high recall is required. A key problem with single-image object detection is the number of genuinely novel combinations of shape and texture that can appear in cluttered environments. Classifiers that are hand-engineered or trained on only a small number of examples are ill-equipped to deal with all of these variations: sooner or later, one of the countless variations in shape or texture will be mistaken for a positive instance. Meanwhile, lighting, contrast, and poor viewing angle can often make it difficult to distinguish legitimate object instances from other random clutter. These factors combined make single-viewpoint object detection quite difficult on its own, and multiple-viewpoint detection (where the object may be seen from many different angles) is regarded as even more difficult.

Fig. 1. A challenging detection task: Even in the absence of occlusion, objects like hammers can be difficult to identify when viewing conditions obscure their key features.

A major complication in object detection with multiple potential views is the variation of an object's appearance with changes in the viewpoint. Even detectors achieving state-of-the-art performance for a well-defined object class can turn out to be poor performers when applied to objects with many differing views. Good performance has been demonstrated for a handful of existing object classes, such as cars and pedestrians, whose geometric structure appears fairly consistent as the object rotates. For instance, pedestrians can often be identified by clusters of vertical edges [7], [8] that are consistent regardless of whether the camera is in front or behind them. The side view of a car, which is essentially planar (and the same on both sides), tends to undergo simple perspective distortions as the car rotates relative to the viewer. Thus, classifiers for these objects can often perform quite well over a broad range of views. In contrast, a more "exotic" object such as a claw hammer, like the one shown in Figure 1, can be surprisingly difficult. As the hammer rotates, not only do the various parts move relative to the image center but the key visual features of the object can appear dramatically different. The claw can be hidden when viewed edge-on, or may be difficult to recognize due to poor contrast with the background behind it (which is often shadowed by the hammer itself). These

Adam Coates and Andrew Y. Ng are with the Computer Science Department at Stanford University, Stanford, CA, 94309 USA. acoates@cs.stanford.edu, ang@cs.stanford.edu

types of objects may need to have separate classifiers for many different viewpoints to achieve good performance. Unfortunately, even if we succeed in building reasonably strong classifiers for each of N different views of the object, it can turn out that combining them is non-trivial. In particular, for N classifiers, it is common practice to simply label an object using the output of the classifier whose response is maximal. This implies, however, that the set of false positives for the combined classifier includes all of the false positives of the individual classifiers—potentially increasing the number of false positives by a factor of N. For difficult objects with many differing views this can seriously degrade performance.

Thus, to identify objects from many different views, we would like to solve two main difficulties: (i) the high error rates of single-image, single-viewpoint detectors, and (ii) the factor of N blowup in false positives if we naively combine these detectors into a multiple-viewpoint detector. In this paper, we will explore several complementary solutions to these problems.

First, recent work [9], [10], [11] has demonstrated that improved accuracy can be obtained by using relatively simple learning algorithms trained from very large datasets. In our previous work, we have demonstrated that using large numbers of negative examples (up to 1 million) can improve results for object detection [12]. Intuitively, by training our classifiers on huge numbers of examples, we can reduce the likelihood that a classifier will encounter a genuinely unique example about which it is confused. This will allow us to reduce the error rates of our single-image, single-viewpoint detectors to a level that is tolerable. In this work, we will show the effect of training on up to 100 million examples (10 million positives) on single-viewpoint detection performance. Our final multi-view detector will use 18 classifiers trained on 6 million examples each.

While our large dataset approach will give us classifiers that perform reasonably well, these classifiers will not be perfect. For instance, it is common for our detectors to miss some objects that are viewed from difficult angles (for instance, when the claw of the hammer blends in with desk clutter). More frustrating still are the numerous false positives where an "unlucky" combination of shapes or viewing conditions have conspired to inexplicably fool the detector. Each of these types of errors might disappear if the image had been taken from a different angle. The errors made by the detector in two different images of the same scene are often different, suggesting that their results can be combined to yield higher accuracy. Based on this observation, we present a method for probabilistically combining the outputs of multiple object detectors, allowing us to use multiple cameras in conjunction with almost any choice of base detection algorithm. We show that this can significantly improve the accuracy of the base detector.

In addition to the above, we also explore the use of high resolution imagery in our object detection system as another avenue for reducing the number of mistakes made by single-image classifiers. Though it has been shown that humans are capable of identifying objects in image fragments as small as 32 by 32 pixels [9], this makes the task far more difficult than perhaps necessary. Our system uses pan-tilt-zoom cameras that allow us to zoom in and acquire "close up" views of objects that reveal much more detail, often revealing features that were difficult to see in the original image. We will use this capability in Section V to verify object detections with higher accuracy than would otherwise be possible.

We will begin by surveying work related to both our single-image object detection algorithm and our work with multiple cameras in Section II. We will then present the three major components of our detection system: our detection algorithm based on large synthetic training sets in Section III, our method of combining detections from multiple cameras in Section IV, and our use of high resolution images in Section V. We will present our experimental results in Section VI, demonstrating the performance of the combined system.

II. RELATED WORK

Our single-image object detector builds on the system presented in [12]. Our system uses the patch-based features described by Torralba et al. [13] and the Histogram of Oriented Gradient (HoG) features of Dalal and Triggs [7]. These features have been used for object detection (using boosting in [13]) and pedestrian detection (using a template-matching method in [8]). In our work, both are treated as "black box" feature generators and tossed into our large-scale learning algorithm. In many ways, our use of the patch-based features from [13] combined with machine learning algorithms resembles the work of Schneiderman and Kanade [14]. They have used wavelet-based features and boosted decision trees to perform detection of faces and cars from multiple viewpoints. Their system also uses separate classifiers for each viewpoint, combined subsequently to form a single output. They also use synthetic variations of hand-labeled training examples (a common practice in vision applications) to obtain better generalization. Synthetic data, as we will use for our classifier, has also been used successfully in prior work to generate large datasets useful for computer vision work. LeCun et al. [15], for instance, used synthetic data to train algorithms that perform well on difficult recognition tasks.

Object detection from multiple cameras, in contrast to the closely related problem of detecting objects with multiple potential views, has been studied less well in robotics literature. Multiple camera detection algorithms have been used in surveillance applications to keep track of moving objects or people as they pass between camera views [16], and to locate faces of users in smart rooms [17]. More similar to our work, multiple cameras have been used by Alahi et al. [18] to locate a known object (observed by a fixed camera) in the views of other cameras, including mobile ones. Multiple cameras are used in [19] to locate soccer players in video captured from several cameras during a game, allowing their system to deal with occlusions and overlap that would otherwise require an advanced person detector. The application of this technique to general object detection when a high-accuracy single-image detector is available has not been explored, perhaps due to the inherent difficulty in building even a modestly accurate single-image object detector. Our own system is motivated by the same apparent advantages as prior work on multi-camera detection and tracking, but is more closely integrated with a high-performance object detection method that allows us to apply this approach to the hard task of object detection in robotics.

Use of high resolution imagery for object detection has also been considered previously in combination with zooming cameras. In particular, "peripheral-foveal" systems have been developed that mimic human uses of peripheral vision for coarse scanning of a scene, combined with high resolution "foveal" vision for detection of objects [20], [21], [22]. Our system similarly leverages the use of high resolution images for object detection, but does not use the "attention" models used in previous work. Instead, we use our object detection system (boosted by the use of multiple cameras) to select the regions where we will perform high resolution detection.

III. SINGLE-IMAGE, SINGLE-VIEW DETECTION

Our detection system uses a base classifier that can be executed independently for each camera view to find an object in a specific pose. While using multiple cameras will ultimately improve our accuracy, this base detector must perform relatively well on its own. A poor detection algorithm that tends to miss objects frequently or make large numbers of mistakes (especially mistakes that are correlated across object views) will not benefit much from the use of multiple cameras. We achieve high performance by learning our classifier from an extremely large training set. Our classifier is based on boosted decision trees and uses a distributed training system capable of handling up to 100 million training examples [12], far more than most off-the-shelf systems.

In previous work, we have achieved good performance using a large corpus of negative examples and a small set of hand-labeled positive examples for each object class but the lack of positive examples made it difficult to handle objects whose appearance can vary significantly due to lighting (especially specular reflections) or intra-class variation. In this paper, we use synthetic positive examples instead of hand-labeled ones allowing us to artificially generate many variations of each object. By training on a large number of variations we can achieve greater robustness of the final classifier.

We have experimented with training sets as large as 100 million examples.[1] Figure 2 shows the effect of training a single-viewpoint detector on larger and larger datasets. Performance increases dramatically up to several millions of examples, and continues to increase all the way up to 100 million examples. We note that our final experiments will use 6 million examples for each single-view classifier (roughly the limit of performance illustrated in Figure 2), and that this is *two orders of magnitude more data* than typical detection and recognition datasets (Table I).

[1]Each example is a 320 by 240 image patch containing either the target object, centered and cropped to the same scale, or random background imagery.

TABLE I
DATASETS SIZES

Dataset	Positive examples per class
Coates et. al, 2009 [12]	≤ 730
Caltech 101 [23]	≤ 800
Caltech 256 [24]	≤ 827
LabelMe (Pedestrian) [25]	25330
NORB [15]	38880
Our single-view detector	1×10^6
Our multi-view detector	18×10^6

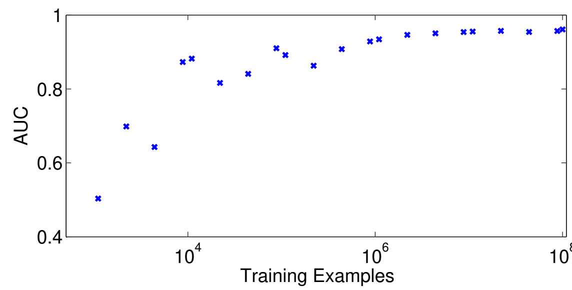


Fig. 2. The performance (area under precision-recall curve) of a single-viewpoint classifier for claw hammers trained on up to 100 million examples. (10 to 1 negatives-to-positives ratio.)

A. Preliminaries

Formally, we will assume that we are given a training set composed of labels $y_i \in \{0, 1\}$ and example images $\mathcal{I}_i$. (We will describe the generation of the positive example images briefly in Section III-B.) In addition, we will assume for every positive example that we know the rotation α_i of the object relative to some canonical pose. For example, an object facing directly away from the camera might have $\alpha_i = 0°$ while another example of the object with its side facing the camera would have $\alpha_i = 90°$. We then split the positive examples into 18 overlapping bins based on their pose. So, for instance, all positive examples with $\alpha_i \in [0°, 24°)$ would be grouped together and those with $\alpha_i \in [20°, 44°)$ would form another group.[2] A separate feature bank is constructed for each bin using the method described in Section III-C, and the feature values are computed for every positive example image in the bin, as well as for all negative example images.[3] For each bin we now have a training set of labels y_i, feature vectors x_i and, for positive examples, poses α_i. We then use the training algorithm described in Section III-D to learn a separate classifier for each bin. This classifier estimates $P(y = 1|\mathcal{I}; \alpha)$ for a new example image $\mathcal{I}$ (this is the probability that $\mathcal{I}$ is an image of the target object in pose α). As we will show in our experiments (Section VI), these pose-specific classifiers achieve high performance.

Given an input image, we apply our classifiers using the standard "sliding window" approach [7], [4], [26]. We

[2]The extra $4°$ overlap ensures that there are enough positive examples with poses near the bin limits that we do not end up with "gaps" in poses covered by all of the classifiers.

[3]Our negatives also include small, incomplete snippets cut from positive example images. This has the effect of forcing the classifier to learn to label full objects and avoid relying on just a few parts.

evaluate all of the classifiers independently on a series of windows of varying sizes spaced at uniform intervals over the image to detect objects at all locations, scales, and poses. Thus, for each sub-region $\mathcal{I}_i$ of an image $\mathcal{I}$, our classifiers output the probability that the target object is contained in the sub-region at the pose for which the classifier was trained ($P(y_i = 1|\mathcal{I}_i; \alpha)$). Normally, these results would be collected and the classifier with maximal response at each location would be selected as the final output of the system. However, we will use these probability estimates later to improve our detection accuracy using multiple cameras. We now briefly describe the sub-components of our classification system in more detail.

B. Training Data Synthesis

Ideally we always want to train on the same kinds of data on which we will test our classifier. However, if large amounts of data are needed to learn good classifiers, collecting real data for training becomes arduous if not infeasible. For example, in order to train detectors that are invariant to lighting, exposure, and distortions we must either hand-engineer features and algorithms that take these into account or otherwise find examples of our object that exhibit all of these variations in different combinations. Synthetic data offers us the ability to train our classifier to be robust to different types of variations by simply generating new examples that demonstrate the kinds of variations that might be seen in reality.

For our system, we acquire large numbers of positive examples by synthesizing them using a "green screen" method similar to those used in [15] and [27]. A typical image captured from this system is shown in Figure 3a.

Using the (known) background color, a mask is computed that covers the object in the image frame (Figure 3b). Using the captured image and mask, we then apply a series of photometric and geometric distortions to generate new positive examples. These distorted examples attempt to capture variations that frequently appear in real data. These include: (i) random backgrounds (placed behind the object using the object's mask for blending), (ii) random perspective distortions, (iii) non-linear lighting changes (simulating under- and over-exposed images), and (iv) blurring.

Figure 3c shows an example of a claw hammer synthesized using this approach. In addition to the synthesized image, our system also determines the pose of the object from the location of a fiducial marking located on the green screen surface, and thus we obtain the necessary α_i value needed to group the examples for training. In our experiments, each classifier is trained on 1 million positive examples generated in this way.

C. Features

Once we have a set of positive and negative example images (from a single orientation bin), we compute feature vectors for each. We use two types of features: (i) the patch-based features first described in [13], and (ii) Histogram of Oriented Gradient descriptors [7]. We review them here only briefly.

We begin by constructing a dictionary from small image fragments. Each fragment g is randomly extracted from the image channels (intensity and edges) of our synthetic positive examples. Each patch is annotated with a rectangle $\mathcal{R}$ specifying its approximate location relative to the object center, and the index c of the image channel from which it was extracted. Specifically, a patch is defined as a triple $\langle g, \mathcal{R}, c\rangle$. Given an input example image, a patch feature value is computed by first computing the (normalized) cross-correlation of the dictionary patch with the corresponding image channel, and then taking the maximum response over the patch rectangle.

We also use HoG features in addition to patch-based features, applied in a similar fashion to the patches described above. In particular, we build a dictionary of descriptors extracted from positive examples. A feature value is then defined by taking the maximum dot-product between the stored descriptor and a descriptor computed at each point in $\mathcal{R}$ of the example image. This computation can be performed quickly by first computing $\mathrm{HoG}(\mathcal{I}, i, j)$ for all i, j using integral images [28].

The patch and HoG features computed as above are concatenated into a single feature vector that is given to our classification algorithm. In our experiments we have about 400 patch features and 200 HoG features per object view.

D. Training

Once we have generated positive and negative example images and computed the feature vector for each, we train a boosted decision tree classifier using the Gentle Boost algorithm [29]. Since our training sets are extremely large, it is impractical to train these classifiers on single machines. Instead, we use the distributed training approach described in [12]. Except for quantizing the feature values to 8 bits each, the distributed training system yields the same result as training on a single machine.

IV. MULTI-CAMERA DETECTION

Once we have a single-image object detector, we can combine their outputs to obtain improved detection performance. Roughly speaking, if we receive multiple observations of the same scene but from different angles, then even if these observations are correlated it is quite possible that the combined beliefs derived from all of the observations will be better than any single observation. In our setting, we assume that we are given multiple different images of the same scene where the viewpoints from which the images were captured are separated by a modest baseline (similar to what could be achieved with multiple cameras on a robot, or by a robot that can move itself or the cameras a short distance). We'll assume from now on that we have just two cameras, but the same technique can straight-forwardly be applied to any number of observations.

Recall from Section III that our classifiers are trained to identify a single object over a small range (20°) of poses. Each classifier is run in a sliding-window fashion, outputting a posterior probability $P(y_i = 1|\mathcal{I}_i; \alpha)$, where α is the parameter for the pose of the object being detected, and y_i is

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

Fig. 3. Our synthetic data system: (a) Object view as seen by the capturing camera, with fiducial marking. (b) Mask extracted from the green-screen image. (c),(d),(e) Synthesized positive examples.

the label for sub-region $\mathcal{I}_i$. The label is $y_i = 1$ if the object is detected in pose α, or 0 otherwise.

We assume that, after running our single-view classifiers on the image, we are left with a set of candidate bounding boxes. Each box is annotated by the posterior probability computed by the classifier and the approximate pose α. Our basic approach to combining these detections will be broken into several steps:

1) Determine the correspondence between pairs of detections. I.e., which detections in both cameras actually correspond to the same object in the scene.
2) For each pair of corresponding detections, compute the posterior probability for the class label.
3) Perform non-maximum suppression on the detections *jointly* between the cameras to find the most probable locations of the objects considering all observations.

We'll now describe each of these steps in turn.

A. Determining Correspondences

Determining the correspondence between detections in each image is easy if our robot is equipped with a depth sensor or some other knowledge of the distances to each detection. Using the extrinsic and intrinsic parameters of the cameras we could easily determine whether the centers of the bounding boxes corresponded to the same point in the scene, and similarly could verify that the boxes were approximately of the same size (accounting for the object's distance from each camera). Since we do not assume that such a sensor is available we will need to solve the correspondence problem. This turns out to be somewhat non-trivial when accurate parameters for the camera extrinsics are not known. This is not only the case for our pan-tilt-zoom cameras[4] but is also true, for example, when images are captured by moving the robot base. In these cases, we may have only coarse estimates of the camera extrinsics from the robot's localization system.

To determine the correspondences we apply a number of filtering steps to first reject pairs of detections that are not likely to correspond to the same object. For a given pair of detections, we have bounding boxes $\mathcal{R}_1$ and $\mathcal{R}_2$ in the first and second camera respectively. Since we have approximate intrinsic and extrinsic parameters for our cameras, we can use the center points of the bounding boxes to (roughly) triangulate the center point of the hypothetical object in 3D space. We discard points that violate the (approximate) epipolar constraints between the cameras. For instance, if the depth of the center point is very large (say, greater than 20 meters[5]), we can safely assume that the detections do not correspond to the same object. Using the approximate 3D position we can also compute the size of the bounding boxes in 3D space. We reject correspondences whose bounding boxes have a relative error in width of more than 10%.[6]

We also have geometric information about the object itself: for each candidate detection we know the pose parameter α_i from the classifier that generated the detection. Again, using the approximate 3D location of the object relative to the cameras, we can reject any correspondences whose poses are not consistent. For instance, if the detection in the first camera has $\alpha_i = 50^\circ$, then we can determine geometrically that the second camera should find the same object with an orientation of, say, $\alpha_i = 70^\circ$ if the cameras are separated by 20 degrees around the object location. Any correspondences whose poses do not (approximately) satisfy these constraints are rejected.

We now use a simple stereo keypoint-matching system to choose the best correspondences from amongst those satisfying the geometric constraints outlined above. As a preprocessing step, we compute point correspondences between the images captured by each camera using SURF descriptors [30]. For each detection in the first image we then search for the best match in the second image. In particular, for a detection with bounding box $\mathcal{R}_1$ in the first image and bounding box $\mathcal{R}_2$ in the second image, we compute the following score:

$$\min\left(\frac{N_{12}(\mathcal{R}_1, \mathcal{R}_2)}{N_1(\mathcal{R}_1)}, \frac{N_{12}(\mathcal{R}_1, \mathcal{R}_2)}{N_2(\mathcal{R}_2)}\right).$$

$N_{12}(\mathcal{R}_1, \mathcal{R}_2)$ is the number of SURF correspondences where the first point falls inside $\mathcal{R}_1$ and the second point falls inside $\mathcal{R}_2$. $N_1(\mathcal{R}_1)$ is the total number of SURF correspondences where the first point falls inside $\mathcal{R}_1$ (regardless of where its corresponding point is found in the second image), and similarly for N_2. We discard low-scoring candidate correspondences, then use a greedy algorithm to find the best pairing between detections in the two images.

[4] Our pan-tilt-zoom cameras do not have encoders to determine their pan and tilt. Instead, we can only dead-reckon their orientation. The pan and tilt angles are reasonably accurate since the cameras use stepper motors, but are not sufficiently reliable for off-the-shelf stereo packages.

[5] Large depth values are unlikely: objects beyond 20 meters are not likely to be legitimately detected.

[6] Note that if the size of the bounding box changes depending on the object view (e.g., if it bounds the object tightly) then this heuristic will not work properly. We ensure during training that our detector learns to place bounding boxes at the same size relative to the object regardless of viewpoint—allowing us to use the bounding box width as a proxy for object size.

B. Computing Posterior

The result of the above procedure is a reduced set of candidate detections for each image, and pair-wise correspondences between the detections indicating which pairs of detections potentially correspond to the same object in the scene. The next step is to combine the detections to make more confident predictions about the label of the object. Our classifiers output probabilities instead of discrete labels, thus giving us a measure of their confidence. By probabilistically combining the detections using the probabilities supplied by the classifiers we can obtain a more confident labeling than either classifier could achieve alone.

In the following we will consider one pair of the remaining candidate detections. We will refer to the sub-region $\mathcal{I}_1$ containing the object as viewed from the first camera and the sub-region $\mathcal{I}_2$ containing the object as viewed from the second camera. Each detection is annotated by probabilities $P(y = 1|\mathcal{I}_1; \alpha_1)$ and $P(y = 1|\mathcal{I}_2; \alpha_2)$ respectively from our classifiers, where α_1 and α_2 are the pose parameters of the classifiers that generated the detections. Note that we have dropped the subscript for the label y since both classifier outputs now refer to the same object (which must ultimately be labeled the same way in both camera views). In the following, we will omit the α_i as they are not needed for computing the posterior probability of the class label.

We will assume for the moment that we are given accurate posterior probabilities $P(y = 1|\mathcal{I}_1)$ and $P(y = 1|\mathcal{I}_2)$. Clearly the classifier outputs are highly correlated, since the images $\mathcal{I}_1$ and $\mathcal{I}_2$ are observations of the same scene. Nevertheless, we will assume that the observed sub-regions $\mathcal{I}_1$ and $\mathcal{I}_2$ are conditionally independent given the label y:

$$P(\mathcal{I}_1, \mathcal{I}_2|y) = P(\mathcal{I}_1|y)P(\mathcal{I}_2|y).$$

Under this assumption the combined posterior probability $P(y = 1|\mathcal{I}_1, \mathcal{I}_2)$ can be computed using several applications of Bayes's Rule yielding:

$$P(y|\mathcal{I}_1, \mathcal{I}_2) \propto \frac{P(y|\mathcal{I}_1)P(y|\mathcal{I}_2)}{P(y)}. \quad (1)$$

From this it is easily seen that the log odds of the posterior in Eq. (1) can be written:

$$\log \frac{P(y = 1|\mathcal{I}_1, \mathcal{I}_2)}{P(y = 0|\mathcal{I}_1, \mathcal{I}_2)} =$$

$$\log \frac{P(y = 1|\mathcal{I}_1)}{P(y = 0|\mathcal{I}_1)} + \log \frac{P(y = 1|\mathcal{I}_2)}{P(y = 0|\mathcal{I}_2)} - \log \frac{P(y = 1)}{P(y = 0)}. \quad (2)$$

Thus, we can combine the detection probabilities by simply adding the log odds together (plus a constant). The choice of constant $P(y = 1)$ is generally not important since we will ultimately threshold the combined detections at whatever value yields the best performance.

We have assumed that the posterior probabilities output by the detector are accurate. In reality, the probabilities are incorrect because they refer to the training distribution.[7] Thus, our classifier actually gives us $\log \frac{\hat{P}(y=1|\mathcal{I})}{\hat{P}(y=0|\mathcal{I})}$ for a distribution $\hat{P}$ that is not the same as the test distribution. We compensate for this difference by learning linear function parameters α and β that map the detector outputs to the correct values needed for Eq. (2) using logistic regression [31] on a hold-out set of synthetic data:

$$\log \frac{P(y = 1|\mathcal{I}_1)}{P(y = 0|\mathcal{I}_1)} = \alpha \log \frac{\hat{P}(y = 1|\mathcal{I}_1)}{\hat{P}(y = 0|\mathcal{I}_1)} + \beta.$$

C. Non-Maximal Suppression

After completing the above steps, we will have a set of corresponding detections with a combined posterior probability for each reflecting the likelihood that the target object will be found at a particular location. As is commonly the case with sliding window detection algorithms, we must suppress detections that are not local maxima.[8] Since we have multiple images, however, this suppression must be done jointly to maintain consistency between the detections in each camera view. A natural solution is to perform the non-maximal suppression in the 3D world space rather than the 2D image space. As we have done previously, we can triangulate the 3D position of the detections using our (rough) camera parameters. We then perform non-maximal suppression in the usual way over 3D position and object scale, removing any detections whose probability values are lower than another detection that is nearby in the 3D space.

V. FOVEATION

A major limitation in detecting objects in single 2D images is the quality and resolution of the image itself. Our cameras, for instance, output 640 by 480 pixel images. While the quality is sufficiently high for humans to pick out most objects, many key details of an object are often lost, making the detection process much more difficult. One solution is to use a camera with zoom capability to take high resolution pictures of "interesting" areas in the scene and then perform object detection in these regions.

In this work, we use foveation as a "verification" mechanism for candidate detections found by our multi-camera detector. For each candidate detection found using the methods described above, we will direct our cameras to acquire high resolution images of the detection's bounding box (one image for each camera, as before). We will then re-run our detectors on the higher resolution images, yielding a set of detections and a probability estimate for each. Since we are zooming in on a previously selected candidate region there is no stereo correspondence problem to solve; we can simply combine the detection probabilities as described previously for any detection pairs whose poses are consistent with the camera geometry. We can then use these combined detections for our results and discard the detections acquired from the low-resolution images.

VI. EXPERIMENTS

For our experiments, we used two AXIS 214 pan-tilt-zoom cameras and tasked them with identifying claw hammers

[7]The classifier has been optimized for the training distribution (and, indeed, usually separates the data perfectly) and thus will be over-confident. The training data also has a different distribution of positive and negative examples than real data.

[8]As a sliding window detector passes over an object in an image the probability values from the detector increase to a maximum (when the object is roughly centered) and then begin decreasing.

around our lab. The two cameras are mounted slightly less than 1 meter apart on a fixed horizontal bar attached to a tripod at a height of about 2.5 meters. A picture of our system is shown in Figure 4. We collected green-screen images of 7 different claw hammers for training. Using these images, we synthesized 1 million examples for each pose as described in Section III. Negative examples were extracted from a set of images that do not contain other hammers, yielding roughly 5 million negative examples for each classifier. We used decision trees of depth 3 and 400 rounds of boosting.

Fig. 4. Our multi-camera system.

Each of the single-view, single-image classifiers performs fairly well in isolation. We tested several of the classifiers individually by placing hammers in specific poses relative to one of the cameras and then running the classifier for that pose on the captured image. The precision-recall curves for these classifiers can be seen in Figure 7. They perform quite well considering the difficulty of some of the detections (the example shown in Figure 1, for instance, is classified correctly by the $\alpha = 50°$ classifier). Nevertheless, they are imperfect, and the large number of classifiers (18) amplifies the problem of false positives. As a result, when we naively combine the detections from all of the views we expect the precision of the combined detector to suffer.

We now test the multi-camera and foveation detection strategies described above and compare the results to the accuracies obtained using the standard method of naively combining the detections in a single image (which we expect to exhibit a high false positive rate.) We again place the (previously unseen) hammers around our lab, but this time allow them to be placed in any pose relative to the cameras. Thus, in order to detect all of these hammers, we must use all of the classifiers together. We collected images from 15 different scenes, each containing from 0 to 3 hammers.

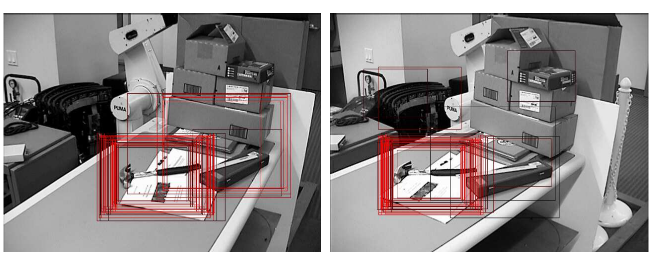


Fig. 5. Output of all 18 single-image classifiers for the left and right cameras for one of the test scenes. (Best viewed in color.)

To illustrate the operation of the system, a typical example of the outputs of each step of our detection system are shown in Figures 5 and 6. In each image, the brightness

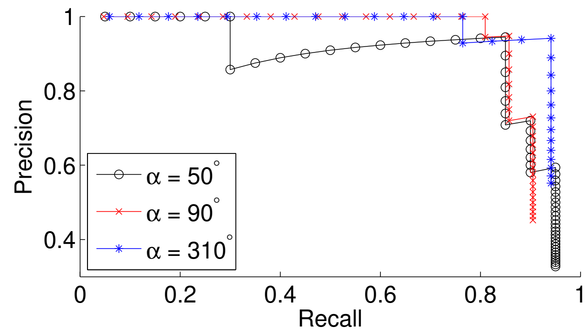


Fig. 7. Precision-Recall curves for several individual claw-hammer classifiers. The pose (α) for each classifier is given in the legend.

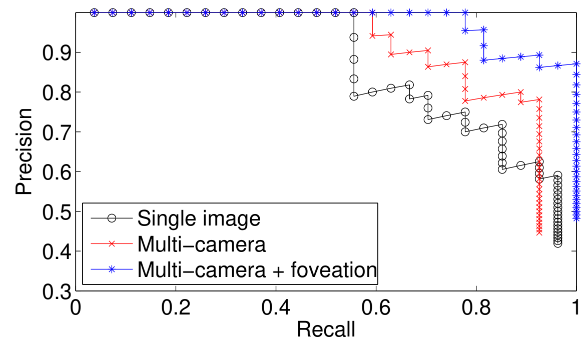
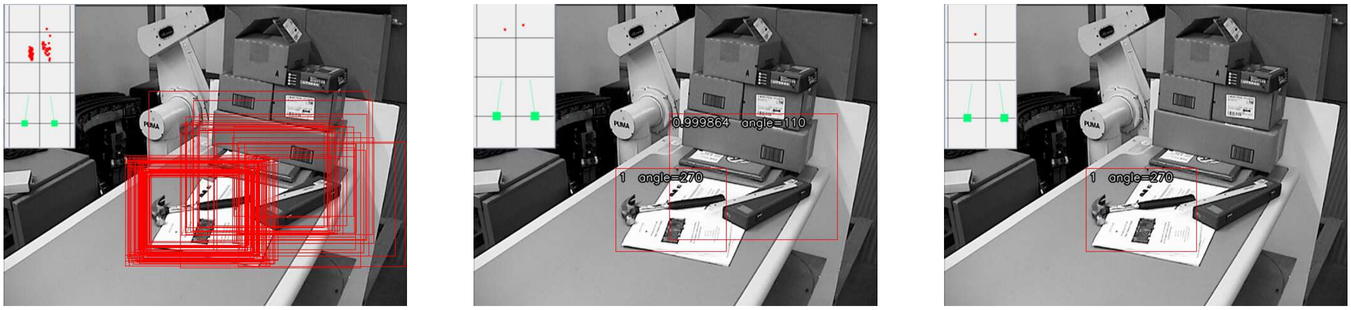


Fig. 8. Precision-Recall curves for our claw-hammer detector.

of the rectangles indicates the probability associated with the detection. Figure 5 shows a scene as viewed from each camera along with the detections output by the single-image classifier. These detections are then combined using the procedure described in Section IV to obtain a reduced set of corresponding detections with more accurate probability estimates. These detections, as viewed from the left camera, are shown in Figure 6a. Also shown are the 3D triangulated positions of the detections, where it is clear that there are two clusters of detections. Figure 6b shows the result of applying the 3D non-maximum suppression step of Section IV-C, where all of the detections have been collapsed to two locations. Finally, our system foveates on each of the two remaining candidate detections and executes the classifiers a second time. In this case, the previously incorrectly labeled region is correctly labeled as negative during the foveation, while the correctly labeled hammer is verified as shown in Figure 6c.

The final precision-recall curves for our multi-camera detection methods are shown in Figure 8.

As Figure 8 makes clear, the naive strategy for combining detections suffers from reduced precision compared to the accuracies of the individual classifiers (compare with Figure 7). This method achieves an area-under-curve of 0.86. The performance of the multi-camera detector is significantly better (AUC of 0.89), recovering some of the lost precision. Moreover, the addition of foveation to the system improves performance much further still (AUC of 0.97). These results suggest that the false positive errors created by using many classifiers can be successfully overcome using these types of multiple-camera, multiple-resolution techniques.



(a) (b) (c)

Fig. 6. (a) Detections in the left camera after combining with the detections from the right camera. (b) Detections in the left camera after applying non-maximum suppression in the 3D world space, labeled with the posterior probability and the approximate pose (angle in degrees) output by the classifier. (c) The final detection results after foveation. [The insets show the 3D triangulated positions of the detections in each view relative to the two cameras.]

VII. CONCLUSION

We have demonstrated a system for object detection that demonstrates a number of key elements that can be combined to achieve high performance: (i) learning from extremely large datasets including synthetic examples, (ii) leveraging multiple camera views, (iii) using high resolution imagery to verify detections. Our results show that the use of multiple cameras and resolutions allows us to take a strong single-image, single-view object detector and construct a multi-view object detector that achieves higher accuracy than is possible using the standard approach of combining the responses in a single image. Since these techniques are applicable to any type of object detector that outputs probability estimates, we believe these results demonstrate that multi-camera and multi-resolution methods are valuable tools for building robust multi-view object detection systems. Moreover, we expect that the methods presented here will continue to yield benefits as more sophisticated single-image detectors are developed.

VIII. ACKNOWLEDGMENTS

Adam Coates is supported by a Stanford Graduate Fellowship.

REFERENCES

[1] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. V. Le, A. Wellman, and A. Y. Ng, "High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening," in *ICRA*, 2009.
[2] D. H. Ballard and C. M. Brown, "Principles of animate vision," *CVGIP*, 1992.
[3] A. Arsenio, "Embodied vision - perceiving objects from actions," in *12th Workshop on Robot and Human Interactive Communication*, 2003.
[4] H. A. Rowley, S. Baluja, and T. Kanade, "Human face detection in visual scenes," in *NIPS*, 1995.
[5] P. Viola and M. Jones, "Robust real-time object detection," *IJCV*, 2001.
[6] H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars," in *CVPR*, 2000.
[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[8] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *CVPR*, 2008.
[9] A. Torralba, R. Fergus, and W. Freeman, "80 million tiny images: a large dataset for non-parametric object and scene recognition," in *PAMI*, 2007.
[10] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006, pp. 2161–2168.
[11] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," in *39th Annual Meeting on Association for Computational Linguistics*, 2001.
[12] A. Coates, P. Baumstarck, Q. Le, and A. Y. Ng, "Scalable learning for object detection with GPU hardware," in *IROS*, 2009.
[13] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *PAMI*, 2007.
[14] H. Schneiderman and T. Kanade, "Object detection using the statistics of parts," *IJCV*, 2004.
[15] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *CVPR*, 2004.
[16] V. Kettnaker and R. Zabih, "Bayesian multi-camera surveillance," in *CVPR*, 1999.
[17] Z. Zhang, G. Potamianos, M. Liu, and T. Huang, "Robust multi-view multi-camera face detection inside smart rooms using spatio-temporal dynamic programming," in *7th Conf. on Automatic Face and Gesture Recognition*, 2006.
[18] A. Alahi, P. Vandergheynst, M. Bierlaire, and M. Kunt, "Object detection and matching in a mixed network of fixed and mobile cameras," in *1st ACM workshop on Analysis and Retrieval of Events/Actions*, 2008.
[19] B. M. Junior and R. de Oliveira Anido, "Object detection with multiple cameras," in *Workshop on Motion and Video Computing*, 2002.
[20] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Messner, G. R. Bradski, P. Baumstarck, S. Chung, and A. Y. Ng, "Peripheral-foveal vision for real-time object recognition and tracking in video," in *IJCAI*, 2007.
[21] M. Björkman and D. Kragic, "Combination of foveal and peripheral vision for object recognition and pose estimation," in *ICRA*, 2004.
[22] A. Ude, C. G. Atkeson, and G. Cheng, "Combining peripheral and foveal humanoid vision to detect, pursue, recognize and act," in *IROS*, 2003.
[23] 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 *CVPR*, 2004.
[24] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep., 2007.
[25] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology, Tech. Rep., 2005.
[26] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2008.
[27] B. Sapp, A. Saxena, and A. Y. Ng, "A fast data collection and augmentation procedure for object recognition," in *AAAI*, 2008.
[28] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *CVPR*, 2006.
[29] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," Dept. of Statistics, Stanford University, Tech. Rep., 1998.
[30] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, 2008.
[31] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, 1999, pp. 61–74.