A Scalable Tree-based Approach for Joint Object and Pose Recognition

Kevin Lai **Liefeng Bo**
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA

Xiaofeng Ren
Intel Labs
Seattle, WA 98195, USA

Dieter Fox
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA

Abstract

Recognizing possibly thousands of objects is a crucial capability for an autonomous agent to understand and interact with everyday environments. Practical object recognition comes in multiple forms: *Is this a coffee mug?* (category recognition). *Is this Alice's coffee mug?* (instance recognition). *Is the mug with the handle facing left or right?* (pose recognition). We present a scalable framework, *Object-Pose Tree*, which efficiently organizes data into a semantically structured tree. The tree structure enables both scalable training and testing, allowing us to solve recognition over thousands of object poses in near real-time. Moreover, by simultaneously optimizing all three tasks, our approach outperforms standard nearest neighbor and 1-vs-all classifications, with large improvements on pose recognition. We evaluate the proposed technique on a dataset of 300 household objects collected using a Kinect-style 3D camera. Experiments demonstrate that our system achieves robust and efficient object category, instance, and pose recognition on challenging everyday objects.

1 Introduction

Recognizing objects in its surroundings is a crucial capability for an autonomous robot to understand and interact with the world and be of use in everyday life scenarios. Additionally, more and more interactive systems require object recognition capabilities to interact with a user in an intelligent way (Kane et al. 2009). Object recognition has been a central and heavily studied research topic in computer vision, with a lot of progress made in recent years. Modern object recognition systems can distinguish between hundreds of categories and detect a variety of objects in complex real-world images (Felzenszwalb et al. 2009).

In practice, object recognition has multiple levels of semantics and multiple usage scenarios. When an autonomous robot encounters an object, we would like it to answer any or all of the following questions: *Is this a coffee mug or a plate?* (**category** recognition); *Is this Alice's coffee mug or Bob's coffee mug?* (**instance** recognition); *Am I looking at the mug with the handle facing left or right?* (**pose** recognition or approximate pose estimation). Although it is clear that category, instance, and pose recognition are closely connected and multiple facets of a single object perception problem, they have traditionally been studied in different contexts and solved using different techniques. The computer vision community has focused on category-level recognition and developed sophisticated features (Lowe 2004; Bo, Ren, and Fox 2010) and matching techniques (Lazebnik, Schmid, and Ponce 2006; Felzenszwalb et al. 2009). There have been efforts to study the problem of discrete (or rough) pose estimation (Savarese and Fei-Fei 2007), and it has been recently shown that category recognition and discrete pose recognition can be solved together (Gu and Ren 2010). The robotics community has typically focused on instance recognition and pose estimation, with techniques using sparse feature matching for textured object labeling (Martinez, Collet, and Srinivasa 2010), or ICP alignment for generic 3D pose estimation (Besl and McKay 1992; Klank, Zia, and Beetz 2009).

One other prevailing issue and urgent need of practical ob-

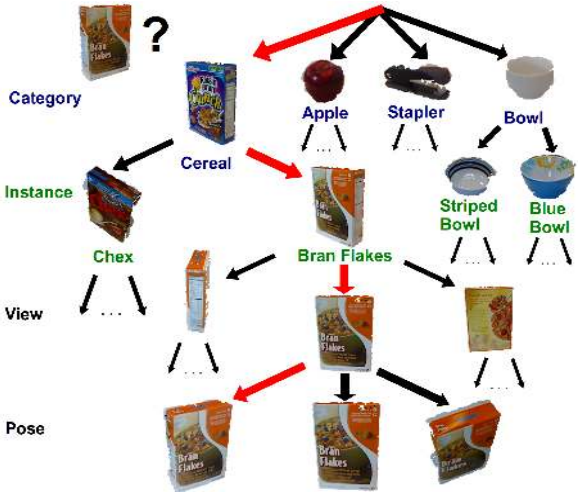

Figure 1: Category, instance, and pose recognition of a box of Bran Flakes cereal using the Object-Pose Tree. The system labels the test image by evaluating a set of classifiers arranged in a semantically structured tree, starting with the category-level at the top and traversing down the tree to the instance, view, and finally the pose level at the bottom. The system finds the most similar (but not identical) pose in the training set.

ject recognition research is that of *scalability*. For practical systems, partly because of the need for real-time processing, recognition is traditionally demonstrated on only a handful of objects. Recent studies have started to push the limit to dozens of object instances (Martinez, Collet, and Srinivasa 2010), but it is still in sharp contrast with the possibly thousands of objects one would encounter in everyday life. In computer vision, there have been ambitious efforts to move up to $100,000$ object classes (Deng et al. 2009), and the machine learning community has developed scalable tree-based approaches to solve such large-scale problems (Bengio, Weston, and Grangier 2010). A two-stage approach has also been proposed to handle large-scale recognition by filtering the database into small sets (He et al. 2010). There have been few such studies on scalable object recognition from a practical perspective. For example, in practice there is a strong need for online incremental learning, as an autonomous agent constantly explores and learns about novel objects.

In this work we develop a tree-based approach that simultaneously solves the three object recognition problems: category, instance, and pose. These three levels of object recognition form a tree as naturally defined by the semantic structures: a category covers multiple instances, an instance covers multiple (discrete) "views", and each view is a collection of (continuous) object poses (Fig. 1). We show that all three recognition tasks can be performed robustly and efficiently by traversing the tree. The tree, or hierarchical, view of the three tasks enables us to jointly learn recognition models at all three levels.

Our joint recognition approach directly tackles the problem of scalability. We show that (1) our tree-based approach leads to efficient recognition without losing accuracy; (2) the tree structure allows efficient and joint training of recognition models at all levels, much faster than standard 1-vs-All training; and (3) online learning, based on stochastic gradient descent, allows us to efficiently update an existing tree model and incorporate novel objects. By solving three recognition tasks together, we are able to perform near real-time recognition over thousands of object poses much more efficiently and accurately than nearest neighbor retrieval or 1-vs-All classifiers.

This paper is organized as follows. We first introduce our tree representation for joint object and pose recognition, followed by experimental results. We apply Object-Pose Trees to an interactive system and finally conclude.

2 Object-Pose Tree

In the joint category, instance, and pose recognition task, we are given a set of training samples $\{x_i, y_i^C, y_i^I, y_i^V, y_i^P\}_{i=1}^N$ where inputs x_i are cropped and segmented images of objects, the outputs y_i^C,y_i^I, y_i^V, and y_i^P are respectively the categories, instances, views, and poses of the objects, and N is the number of training samples. The goal is to predict the category name, instance name and pose of unseen images. An intermediate view level between the instance and pose levels is added where each view contains a set of nearby poses (see Section 3.1 for more details). To estimate the category, instance, and pose of a given object jointly, we build an Object-Pose Tree (OP-Tree). The OP-Tree is a semantically structured tree of classifiers that consists of four levels: category, instance, view, and pose as illustrated in Fig. 1, where one classifier (semantic label) is associated with each tree node.

2.1 Joint Category, Instance & Pose Recognition

To learn an OP-Tree, we will minimize the empirical loss on a set of labeled training data. Before defining this loss, we need to specify the classification performed by the OP-Tree. At test-time, the OP-Tree recognizes an object by evaluating the tree of classifiers one level at a time. The category-level classifiers are first evaluated and the object is assigned the category label of the highest scoring classifier

$$F^C(x) = \operatorname*{argmax}_i f_i^C(x) \ , \tag{1}$$

where $f_i^C(x)$ is the output of the classifier for category i.

The system then evaluates the set of instance classifiers belonging to the assigned category label and selects the instance label of the highest scoring classifier at the instance level. The system continues down through the view and pose levels to finally assign a pose label to the image. Instance, view, and pose classification are summarized by the following rule:

$$F^s(x) = \operatorname*{argmax}_{i:(F^r(x),i)\in E} f_i^s(x), \tag{2}$$
$$(s,r) \in \{(I,C),(V,I),(P,V)\}$$

Here, the set of edges $E = (p_1, c_1), \ldots, (p_{|E|}, c_{|E|})$ contains ordered pairs of parent and child node indices, and $f_i^s(x)$ are the classifiers associated with the nodes i at level s. For example, the *pose* of an input x is predicted as

$$F^P(x) = \operatorname*{argmax}_{i:(F^V(x),i)\in E} f_i^P(x), \tag{3}$$

where $F^V(x)$ is the predicted *view*, and the maximization is performed over all pose classifiers $f_i^P(x)$ associated with that specific view.

We are now ready to specify the empirical loss associated with a set of labeled inputs x_i. Note that since the view level is only added for efficiency reasons, the loss is specified only at the category, instance, and pose level:

$$\begin{aligned} R_{emp} &= \frac{1}{N}\sum_{i=1}^{N} \delta(F^C(x_i) \neq y_i^C) \\ &+ \frac{1}{N}\sum_{i=1}^{N} \max_{s\in\{C,I\}} \delta(F^s(x_i) \neq y_i^s) \\ &+ \frac{1}{N}\sum_{i=1}^{N} \max\{\max_{s\in\{C,I\}} \delta(F^s(x_i) \neq y_i^s), \max_{s\in\{V,P\}} \Delta(F^s(x_i), y_i^s)\} \ . \end{aligned} \tag{4}$$

Here, $\delta(\cdot)$ is the indicator function, and $\Delta(\cdot)$ is a continuous loss function normalized into $[0,1]$. The first term measures the error for assigning an incorrect category label. The second and third terms, which account for the errors of instance and pose recognition, respectively, take the maximum over

multiple levels of the tree because a mistake at any of the higher levels will also lead to an incorrect prediction. For example, if the tree makes a mistake at the category level for a particular object, it will also fail on both the instance and pose recognition task.

Unlike category and instance recognition, which involve a discrete set of labels, for pose recognition it is natural to consider a continuous loss function. In our application, we represent views and poses as angles in $[0, 2\pi]$ and use the following continuous loss to measure the difference between two views or pose labels:

$$\Delta(y_i, y_j) = \frac{\min\{|y_i - y_j|, 2\pi - |y_i - y_j|\}}{\pi} \quad (5)$$

This function converts angle differences into a value in $[0, 1]$ to make the loss comparable with the 0-1 loss metric used at the category and instance levels.

Although the empirical loss described above is ideal, using it would yield a combinatorial optimization problem. We approximate this non-differentiable empirical loss with the hinge loss (used in support vector machines) to get a convex optimization problem. For efficiency we use linear classifiers $f_i^s(x) = (w_i^s)^\top x + b_i^s$ in this paper. At the category level, this hinge loss exactly corresponds to the loss function for a joint multi-class support vector machine (Crammer and Singer 2001):

$$\begin{aligned} R_{svm}^C &= \frac{1}{N}\sum_{i=1}^{N} \xi_i^C \quad (6) \\ s.t. \quad & f_{y_i^C}^C(x_i) - f_y^C(x_i) \geq 1 - \xi_i^C, \forall y \in L^C \\ & \xi_i^C \geq 0, i = 1, \ldots, N \end{aligned}$$

where L^C is the category label set.

We approximate the empirical loss over multiple levels of the tree using the approach proposed in (Bengio, Weston, and Grangier 2010), which takes the largest loss from the current level and the levels above (the ancestral levels). The hinge loss over the instance and pose levels are given by

$$\begin{aligned} R_{svm}^I &= \frac{1}{N}\sum_{i=1}^{N} \max\{\xi_i^C, \xi_i^I\} \quad (7) \\ s.t. \quad & f_{y_i^C}^C(x_i) - f_y^C(x_i) \geq 1 - \xi_i^C, \forall y \in L^C \\ & f_{y_i^I}^I(x_i) - f_y^I(x_i) \geq 1 - \xi_i^I, \forall y : (y_i^C, y) \in E \\ & \xi_i^C, \xi_i^I \geq 0, i = 1, \ldots, N \end{aligned}$$

and

$$\begin{aligned} R_{svm}^P &= \frac{1}{N}\sum_{i=1}^{N} \max\{\xi_i^C, \xi_i^I, \xi_i^V, \xi_i^P\} \quad (8) \\ s.t. \quad & f_{y_i^C}^C(x_i) - f_y^C(x_i) \geq 1 - \xi_i^C, \forall y \in L^C \\ & f_{y_i^I}^I(x_i) - f_y^I(x_i) \geq 1 - \xi_i^I, \forall y : (y_i^C, y) \in E \\ & f_{y_i^V}^V(x_i) - f_y^V(x_i) \geq \Delta(y_i^V, y) - \xi_i^V, \forall y : (y_i^I, y) \in E \\ & f_{y_i^P}^P(x_i) - f_y^P(x_i) \geq \Delta(y_i^P, y) - \xi_i^P, \forall y : (y_i^V, y) \in E \\ & \xi_i^C, \xi_i^I, \xi_i^V, \xi_i^P \geq 0, i = 1, \ldots, N \end{aligned}$$

where $y : (y_i^C, y) \in E$ is the set of child nodes of the node y_i^C and similarly for y_i^I, y_i^V, and y_i^P.

We learn the weights of classifiers by optimizing the overall convex loss function, the summation of three above convex loss functions (Eq. 6), (Eq. 7), and (Eq. 8) and a convex term:

$$W^* = \underset{W}{\operatorname{argmin}}\{R_{svm} = \frac{\lambda}{2}W^\top W + R_{svm}^C + R_{svm}^I + R_{svm}^P\} \quad (9)$$

where the first term is the l_2-norm regularizer, λ is the trade-off parameter, and W is the concatenation of all weight vectors in the tree.

2.2 Object-Pose Tree Learning

We optimize (Eq. 9) using stochastic gradient descent (SGD), which is suitable to problems where the full gradients decompose as a sum of individual gradients of the training samples. Unlike batch methods that estimate gradients using the full set of training samples, SGD approximates gradients using only a subset of training samples of fixed size and thus makes the cost of each iteration constant. This makes SGD a practical choice for large datasets. We use the SGD algorithm proposed by (Shalev-Shwartz, Singer, and Srebro 2007) which iteratively updates model parameters in two steps. The first step is

$$W_{t+\frac{1}{2}} = W_t + \frac{1}{\lambda t}\frac{\partial R_{svm}^t}{\partial W_t} \quad (10)$$

where R_{svm}^t is the hinge loss over a randomly chosen subset of training samples at step t, and the learning rate is set to be $\frac{1}{\lambda t}$. In the second step, we set W_t to be the projection of $W_{t+\frac{1}{2}}$ onto the set $\{B = W : \|W\| \leq \frac{1}{\sqrt{\lambda}}\}$

$$W_{t+1} = \min\{1, \frac{1}{\sqrt{\lambda}\|W_{t+\frac{1}{2}}\|}\}W_{t+\frac{1}{2}} \quad (11)$$

(Shalev-Shwartz, Singer, and Srebro 2007) present theoretical guarantees and empirical results to demonstrate that this stochastic gradient descent algorithm converges in reasonable time. As our experiments will demonstrate, our experience also confirms the usefulness of this optimization technique.

The ability to update the classification model in an incremental fashion is important for certain applications. Consider the scenario where there is an ever-growing database of objects shared by a number of robots for doing object recognition, where objects may be added to the database at any time. In such a scenario we want the model to be updated quickly so that robots can begin recognizing the newly added objects.

An important advantage of the proposed OP-Tree learning technique is that the tree can be quickly updated in an incremental fashion. Using stochastic gradient descent to optimize the tree parameters allows incremental updating of the tree as the system encounters new objects. This is accomplished simply by continuously running SGD as objects are added to the tree. The parameters of each new classifier is initialized using parameters from a randomly chosen

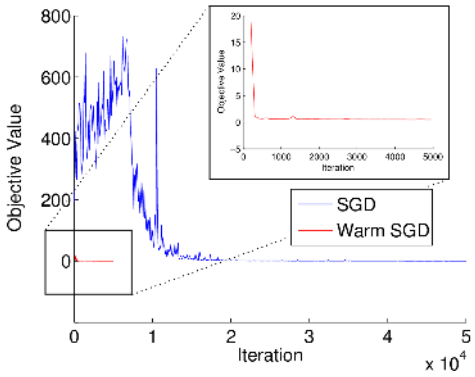

Figure 2: Optimization convergence comparison of stochastic gradient descent from scratch (SGD) and stochastic gradient descent starting with parameters learned for 290 objects and adding 10 more (Warm SGD).

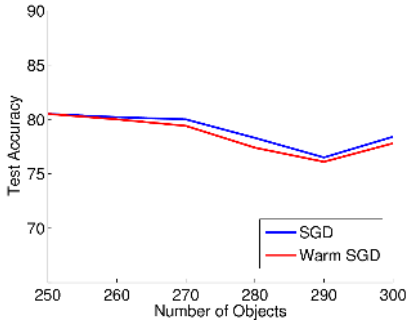

Figure 3: Instance recognition test accuracy of stochastic gradient descent from scratch (SGD) and stochastic gradient descent with pre-initialized parameters (Warm SGD), starting from 250 objects and adding 10 each round for 5 rounds.

sibling, while existing classifiers are initialized with their current parameters.

To verify this claim, we conducted an experiment to compare the efficiency of running SGD from scratch versus initializing parameters with previously learned weights. We first trained a tree from scratch using 290 objects from the RGB-D Object Dataset (see Section 3.1). We then consider two methods for updating the tree after adding ten objects to the classifier: 1) run SGD from scratch on the complete set of 300 objects (SGD), 2) run SGD with the parameters initialized as described in the previous paragraph (Warm SGD).

Fig. 2 shows how the objective function value changes as the two optimization algorithms proceed. Although the objective value for SGD appears to have stabilized after 30,000 iterations, we found that test accuracy continued to improve until 50,000 iterations, using 200 random samples per iteration. In contrast, Warm SGD only took 5000 iterations, also 200 samples per iteration, to converge and achieve the same test accuracy, yielding a 10x speed-up.

We also conducted an experiment where we start with an OP-Tree for 250 objects and then incrementally add 10 objects at a time for 5 rounds. Fig. 3 shows that Warm SGD achieves comparable instance recognition test accuracies with SGD from scratch, even though only 5000 iterations were run on each round of Warm SGD while SGD from scratch had to be run for 50,000 iterations each round to reach comparable accuracy.

Figure 4: Objects from the RGBD Object Dataset. (Left) 16 different objects from the dataset. (Right) 8 views of a pitcher (top) and 8 views of a bok choy (bottom).

Finally, we verified that the Warm SGD optimization algorithm can learn an entire OP-Tree from scratch. We started an OP-Tree with no objects and added the entire dataset of 300 objects, 10 at a time, for 30 rounds. Due to time constraints this experiment was conducted using only the category and instance levels of the tree. We found that an OP-Tree learned this way, with 5000 iterations per round, achieved category and instance recognition accuracy within 1% of running SGD on the full dataset for 50,000 iterations.

3 Experiments

3.1 RGB-D Object Dataset

We compared OP-Trees to other approaches using the RGB-D Object Dataset, which contains images of 300 objects taken from multiple views with a Kinect-style 3D camera (Lai et al. 2011). The dataset is available at http://www.cs.washington.edu/rgbd-dataset. The 300 objects are organized into 51 categories. 640×480 RGB and depth image pairs of each object are taken with the camera mounted at three different heights corresponding to angles of $30°$, $45°$ and $60°$ with the horizon. Each image pair is annotated with ground truth category and instance labels, as well as pose angles in $[0, 2\pi]$. We divide the range of pose angles into 8 slices each covering $45°$ to form nodes in the view layer of the OP-Tree. Pose angles are aligned across all video sequences of instances from the same category. Following the *Leave-Sequence-Out* evaluation procedure in (Lai et al. 2011), we uniformly sample 48 images per video and use the $30°$ and $60°$ sequences as training data and the $45°$ sequence for evaluation. This yields around 28,000 RGB + Depth image pairs for training and 14,000 for testing. Fig. 4 shows some example objects from the dataset.

3.2 Experimental Setup

To represent each RGB+Depth image pair, we first extract gradient and shape kernel descriptors (Bo, Ren, and Fox 2010) over a dense 8×8 grid from the RGB and depth images separately. We use Efficient Match Kernels (EMK) described in (Bo and Sminchisescu 2009) to generate image-level features separately for each set of local kernel descriptors. This combination of kernel descriptors and EMK was demonstrated in (Bo, Ren, and Fox 2010) to outperform alternative state-of-the-art techniques on publicly available

Technique	Category	Instance	Avg Pose	Med Pose	Avg Pose (C)	Med Pose (C)	Avg Pose (I)	Med Pose (I)	Test Time (s)
NN	86.8	60.3	39.1	20.0	45.1	41.6	65.2	81.4	54.76
FLANN	84.6	55.9	38.2	10.9	42.5	33.4	64.3	78.4	0.21
1vsA	93.5	75.7	n/a	n/a	n/a	n/a	n/a	n/a	n/a
1vsA+NN			48.4	51.3	53.2	61.3	63.9	77.7	1.99
1vsA+RR			44.0	47.7	48.3	52.9	58.0	61.2	1.65
Indep Tree	92.0	77.4	50.4	59.3	54.8	65.6	65.0	75.2	0.33
OPTree	**94.3**	**78.4**	**53.5**	**65.2**	**56.8**	**71.4**	**68.3**	**83.2**	**0.33**
OPTree+NN			50.3	55.4	53.3	61.7	64.2	78.4	0.53
OPTree+RR			45.5	49.6	48.2	52.9	58.0	61.2	0.30

Figure 5: Category, instance, and pose recognition accuracy and running time comparison of several techniques. *NN* is exact nearest neighbor classification, *FLANN* is an approximate nearest neighbor classification, *1vsA* is one-vs-all linear SVM, *1vsA+NN* is one-vs-all linear SVM for category and instance recognition, followed by nearest neighbor for pose estimation, *1vsA+RR* is one-vs-all linear SVM for category and instance recognition, followed by ridge regression for pose estimation, *Indep Tree* is a tree of classifiers where each level is trained as an independent linear SVM. *OPTree* is the Object-Pose Tree technique described in this paper, *OPTree+NN* is an Object Tree for category and instance recognition, followed by nearest neighbor for pose estimation, *OPTree+RR* is an Object Tree for category and instance recognition, followed by ridge regression for pose estimation. *1vsA* for pose recognition cannot be trained in a reasonable amount of time and is omitted.

computer vision datasets like Caltech-101 and CIFAR-10. Our previous work (Bo et al. 2011) demonstrated that these features outperform the set of shape and visual features used in (Lai et al. 2011).

In our experiments we consider three broad approaches: 1) Nearest Neighbor classification, 2) One-versus-all linear classifier, and 3) the proposed Object-Pose Tree.

A nearest neighbor classifier can solve these 3 tasks jointly by labeling each test image using the category, instance, and pose labels of the k nearest training images in feature space. We tried different values of k and found that $k = 1$ gives the best results on the RGB-D Object Dataset.

Another approach is to train one-versus-all linear support vector machines separately for each recognition task. However, on our dataset a one-vs-all SVM for pose recognition involves training 28,000 binary classifiers, which did not finish in a reasonable amount of time. Hence, one must combine the use of one-vs-all classifiers with an alternative approach for pose estimation. One way is to use the nearest neighbor training image within the instance predicted using a one-vs-all classifier. Another way is to fit a ridge regression model for each instance and use the regression model for the predicted instance to predict the pose angle.

The OP-Tree technique described in this paper presents a unified framework for jointly addressing category, instance, and pose recognition. The regularization trade-off parameter was chosen using cross validation on the training set.To demonstrate the merits of this approach on all three recognition tasks, we also consider alternative methods of doing pose recognition given the category and instance output of the OP-Tree. As in one-versus-all, we consider using a nearest neighbor classifier or using ridge regression within the predicted instance for doing pose recognition.

3.3 Evaluation Criteria

All of the described approaches are evaluated on three recognition tasks: category, instance, and pose. We report standard category and instance recognition accuracies. As mentioned in Section 2, pose estimation is naturally a continuous problem, so in our evaluation we use one minus the continuous loss function (Eq. 5) as the pose accuracy.

3.4 Experimental Results

Fig. 5 presents a comparison of the various techniques described above. The results are grouped into three overall approaches: nearest neighbor (NN), one-versus-all (1vsA) and Object-Pose Tree (OPTree). Since the different approaches perform differently on instance recognition, the pose accuracies given correct instance predictions are only comparable within an approach but not across approaches. Test times are given for performing all three recognition tasks on a single test image. Feature extraction takes around one second regardless of the technique and is not included in the reported running times.

We report pose accuracies under three different scenarios. We chose to report both median and mean pose accuracies because the distribution across objects is skewed (see Fig. 7). For Avg & Med Pose, pose accuracies are computed on the entire test set, but images that were assigned an incorrect category or instance label have a pose accuracy of 0. Avg & Med Pose (C) are computed only on test images that were assigned the correct category by the system. Avg & Med Pose (I), are computed only on test images that were assigned the correct instance by the system.

Compared to alternative approaches, exact nearest neighbor classification (NN) is extremely slow and gives poor results. Methods for speeding up nearest neighbors include using a kd-tree and doing approximate nearest neighbors (FLANN) (Muja and Lowe 2009). Due to the high dimensionality of our features, using a kd-tree for computing exact nearest neighbors is actually slower than exhaustive search. FLANN with automatically tuned parameters for 95% NN search accuracy gives running time comparable to one-versus-all and OP-Tree approaches, but leads to even lower recognition performance.

One-versus-all approaches (1vsA) improve over the ac-

Figure 6: Recognition results from the Object-Pose Tree for two objects: *Red Mug* (top left), and *Ultrabrite Toothpaste* (bottom left). (Top) From left to right, the top five objects with the highest classifier response at the instance level and at the pose level for *Red Mug*. (Bottom) Instance and pose classifier responses for *Ultrabrite toothpaste*.

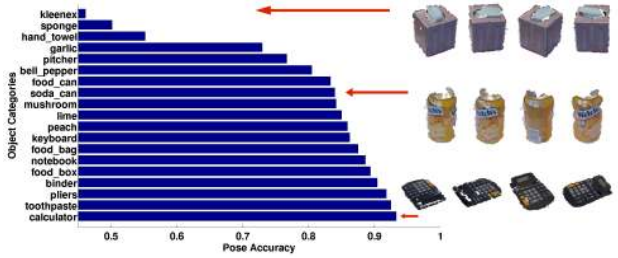

Figure 7: Median pose recognition accuracies from an Object-Pose Tree, given correct instance prediction, for a uniformly sampled subset of object categories. Object categories are sorted in increasing accuracies. Different views of a tissue box, a soda can, and a calculator illustrate the varying difficulty of pose estimation for objects in the RGB-D Object Dataset.

curacy and running time of nearest neighbors significantly. However, one-versus-all cannot be used for pose recognition because training a classifier for each image is not scalable. At test-time the system must also evaluate all classifiers, which scales linearly with the number of objects and poses. This means pose recognition must be addressed using a separate technique like nearest neighbors (1vsA+NN) or ridge regression (1vsA+RR).

Our Object-Pose Tree (OPTree) exceeds the accuracy and running time performance of nearest neighbor classification, one-versus-all classifiers, as well as an identically structured tree of classifiers where each set of nodes sharing a common parent is level of the tree is independently trained as a multi-class linear SVM (Indep Tree). The fact that OPTrees jointly address all three tasks enables the formulation of an objective function that takes advantage of the structure of the problem by using a continuous loss function at the view and pose levels.

Fig. 6 shows recognition results on two images using the OPTree. The image shown for the top 5 matching instances of each test image is the best matching pose from that instance according to the OPTree, by traversing the tree down to the pose layer regardless of whether the instance is the right one. These images, as well as the pose accuracies given correct category and correct instance reported in Fig. 7, show that the OPTree can give good pose estimates even when the instance classification is wrong.

Figure 8: An interactive LEGO playing scenario in which an Object-Pose Tree is used to recognize several LEGO objects and estimate their pose. Recognizing the front of the house enables projection of a street. Detecting which direction the dragon is facing enables projection of a fire breathing animation (left), while detecting the fire truck enables projection of a fire extinguishing animation (right).

The experiments also demonstrate that nearest neighbor and ridge regression within the predicted instance (OP-Tree+NN, OPTree+RR) are both inferior to the OP-Tree in accuracy and running time. Nearest neighbor pose estimation can attain good accuracy, but its running time scales linearly with the number of training images of each object, while the OP-Tree scales logarithmically. Linear ridge regression offers constant-time performance, but gives much worse results. While the use of kernel ridge regression may improve accuracy, like nearest neighbors its computational cost also scales linearly with the number of poses.

Fig. 7 shows the median pose recognition accuracies from an OP-Tree, given correct instance predictions, for a uniformly sampled subset of object categories. The diversity of shape and appearance of objects in the RGB-D Object Dataset means that the difficulty of pose recognition can vary significantly, ranging from textureless and symmetric objects like balls and bowls where it is extremely difficult to distinguish the different poses, to calculators and flashlights where it is easy to tell based on appearance and shape. The OP-Tree achieves $> 83\%$ accuracy on 37 of 51 categories and 185 of the 300 objects. This level of accuracy corresponds to a pose estimation error of $< 30°$, obtained by inverting the continuous loss function (Eq. 5). A small subset of categories with symmetry and/or little texture like kleenex, sponge and hand towel has large pose accuracies of $90°$ or $180°$.

3.5 Object-Aware Situated Interactive System

In collaboration with colleagues at the University of Washington and Intel Labs Seattle, we experimented with using our OP-Tree as the core object recognition algorithm in OASIS, an Object-Aware Situated Interactive System. (Ziola et al. 2011) introduced an interactive LEGO toy playing scenario (LEGO OASIS) where objects placed on a table can be augmented with projected animations and interact with other

physical and virtual objects. Objects are associated with different animations and interact with each other in different ways depending on both their identities and poses. The OP-Tree plays an important role in LEGO OASIS by providing real-time object recognition and pose estimation (100ms per image), which enables responsive object-centric animations and interactions.

In LEGO OASIS, an RGB-D camera and a projector are both mounted above a table and pointed downwards. Since the table is at a fixed position relative to the camera, an object placed on the table can be easily detected and segmented via depth thresholding. The cropped and segmented image is presented to the proposed OP-Tree for determining the object's identity and pose (one of 12 discrete clock directions). Using this result, the system projects the appropriate animations for the identified object. For example, by recognizing a LEGO dragon and the direction at which its head is pointed, the system can project a fire breathing animation. Multiple objects can be recognized in the same play area, allowing interactions such as a dragon setting fire to a house, and a fire truck subsequently putting out the fire (Fig. 8). Four different objects were included in LEGO OASIS: a dragon, a house, a fire truck, and a train. The OP-tree, trained using around 200 images per object taken from various poses, is able to determine the identity and pose of these four objects at $> 95\%$ accuracy. The system was deployed during the 2011 Consumer Electronics Show (CES), where it was seen by more than 10,000 attendees (see http://www.cs.washington.edu/rgbd-dataset/demos.html).

4 Conclusion

We introduced Object-Pose Trees, a scalable approach for joint object category, instance, and pose recognition. We provide extensive experimental results on a recently published dataset (Lai et al. 2011), which covers 300 object instances captured in both color and depth using RGB-D cameras (Kinect 2010). Our tree-based system is able to search the entire object database within 0.33 seconds to identify an object's category, instance, and pose. The Object-Pose Tree trains and utilizes $28,000$ classifiers at the leaf level, which are too many to train via standard 1-vs-All classification. Online stochastic gradient descent for parameter optimization allows us to achieve a 10x speed-up when adding new objects to the database. We achieve $< 30^\circ$ pose estimation error on a wide range of household objects, often exhibiting symmetry and lacking distinctive shape or texture.

This paper opens several directions for future work. Object-Pose Trees can easily be extended to incorporate levels above categories, for example by using WordNet hypernym-hyponym relations as in ImageNet (Deng et al. 2009). Another direction is to use the pose estimation of the Object-Pose Tree to initialize a more accurate shape matching procedure such as ICP, which can refine the pose estimate and return a full 6-D pose of an object. Finally, the combination with interactive projector systems such as OASIS allows the investigation of more challenging object recognition scenarios such as interactive cooking systems.

Acknowledgements

We thank Ryder Ziola, Jinna Lei, Pauline Powledge, James Fogarty, and Beverly Harrison for developing the LEGO OASIS system. This work was funded in part by an Intel grant, by ONR MURI grant number N00014-07-1-0749, and by the NSF under contract number IIS-0812671.

References

Bengio, S.; Weston, J.; and Grangier, D. 2010. Label Embedding Trees for Large Multi-Class Tasks. In *Proc. of NIPS*.

Besl, P. J., and McKay, N. D. 1992. A method for registration of 3-d shapes. *IEEE PAMI* 14(2).

Bo, L., and Sminchisescu, C. 2009. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Proc. of NIPS*.

Bo, L.; Ren, X.; and Fox, D. 2010. Kernel Descriptors for Visual Recognition. In *Proc. of NIPS*.

Bo, L.; Lai, K.; Ren, X.; and Fox, D. 2011. Object Recognition with Hierarchical Kernel Descriptors. In *Proc. of CVPR*.

Crammer, K., and Singer, Y. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR* 2:265–292.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Fei-fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of CVPR*.

Felzenszwalb, P.; Girshick, R.; McAllester, D.; and Ramanan, D. 2009. Object detection with discriminatively trained part based models. *IEEE PAMI* 32(9):1627–1645.

Gu, C., and Ren, X. 2010. Discriminative Mixture-of-Templates for Viewpoint Classification. In *Proc. of ECCV*, 408–421.

He, R.; Hu, B.; Zheng, W.; and Guo, Y. 2010. Two-Stage Sparse Representation for Robust Recognition on Large-Scale Database. In *AAAI-10*.

Kane, S.; Avrahami, D.; Wobbrock, J.; Harrison, B.; Rea, A.; Philipose, M.; and LaMarca, A. 2009. Bonfire: A nomadic system for hybrid laptop-tabletop interaction. In *Proc. of UIST*.

Kinect. 2010. Microsoft kinect. `http://www.xbox.com/en-us/kinect`.

Klank, U.; Zia, M.; and Beetz, M. 2009. 3d model selection from an internet database for robotic vision. In *Proc. of ICRA*, 2406–2411.

Lai, K.; Bo, L.; Ren, X.; and Fox, D. 2011. A large-scale hierarchical multi-view RGB-D object dataset. In *Proc. of ICRA*.

Lazebnik, S.; Schmid, C.; and Ponce, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of CVPR*, volume 2, 2169–2178.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. *IJCV* 60(2):91–110.

Martinez, M.; Collet, A.; and Srinivasa, S. 2010. MOPED: A scalable and low latency object recognition and pose estimation system. In *Proc. of ICRA*, 2043–2049.

Muja, M., and Lowe, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP*, 331–340. INSTICC Press.

Savarese, S., and Fei-Fei, L. 2007. 3d generic object categorization, localization and pose estimation. In *Proc. of ICCV*.

Shalev-Shwartz, S.; Singer, Y.; and Srebro, N. 2007. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proc. of ICML*, 807–814.

Ziola, R.; Grampurohit, S.; Landes, N.; Fogarty, J.; and Harrison, B. 2011. Examining Interaction with General-Purpose Object Recognition in OASIS. In *University of Washington Technical Report UW-CSE-11-05-01*.