A Supervised Learning Approach for Fast Object Recognition from RGB-D Data

David Paulk[§]*
dpaulk@princeton.edu

Vangelis Metsis[†]
vmetsis@uta.edu

Christopher McMurrough[†]
mcmurrough@uta.edu

Fillia Makedon[†]
makedon@uta.edu

[§]Department of Computer Science
Princeton University
Princeton, NJ 08544
United States

[†]Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019
United States

ABSTRACT

Object recognition serves obvious purposes in assisted living environments, where robotic devices can be used as companions to assist humans in need. The recent introduction of vision based sensors, which are able to extract depth sensing information about the environment, in addition to the traditional RGB video, presents new opportunities and challenges for more accurate object recognition.

The current work, presents an object recognition approach that uses RGB-D point cloud data and a novel feature extraction methodology, in combination with well-known supervised learning algorithms, to achieve accurate, real-time recognition of a large number of objects. In our experiments, we use a dataset of household objects organized into 51 categories, and evaluate the recognition accuracy and time efficiency of a set of different supervised learning methods.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Learning—*parameter learning*; I.4.8 [**Computing Methodologies**]: Scene Analysis—*object recognition, shape, color*; I.4.8 [**Computing Methodologies**]: Design Methodology—*classifier design and evaluation, feature evaluation and selection*

General Terms

Algorithms, Experimentation, Performance

Keywords

Object recognition, supervised learning, adaboost, artificial neural network, support vector machine, classification, RGB-D, point cloud.

1. INTRODUCTION

The task of machine-based object recognition is of major importance in assisted living environments, because it constitutes a necessary input module of bigger systems, intended to provide intelligent services to people in need. For example, to assist severely disabled users, recently McMurrough et al. developed a point-of-gaze detection device capable of detecting the intersection of the gaze vector of the user, with an object in the environment [11, 12, 13]. The next step of such a system is to recognize the type of object of interest, in order to understand the intention of the user and the way the object should be handled, e.g. by a robotic arm.

Throughout the most recent decade, scene-sensing technologies have undergone major improvements. By measuring red, green, and blue (RGB) color, modern vision sensors can provide color information from objects in a scene. By adding depth-sensing information, these sensors can also provide a three dimensional layout of objects in a scene. Sensors that collect both color and spatial information are known as RGB-D sensors. When an RGB-D sensor perceives a scene, it collects thousands of data points, each containing a unique set of RGB and depth-sesing values, storing them in a point cloud data (PCD) file for future access. Given the wealth of data in a PCD file, it is possible to compute many useful features that can be used to analyze the content of a scene and distinguish the objects within the scene. The type of PCD-based scene analysis that this research focuses on is categorical object classification.

RGB-D point cloud-based object data can be retrieved from a previously published dataset or created using a 3D sensor such as the Kinect. In this work we experiment with the publicly available dataset provided by [8]. Once object data has been collected, feature information can be extracted from the data and then used by a learning algorithm to build a classification model. Such a model can be used to predict the label of an unobserved object, given the prior knowledge of other objects, some of which may be associated with the same label.

Extracting informative features from the objects, is of paramount importance to the learning process. Previous works have proposed feature extraction methods for RGB images [19, 18]. In this work, we adopt a combination of color-based and 3D geometric and volumetric features extracted from the captured RGB and point cloud data images. This set of features, has been chosen because it forms

[*] Majority of work carried out during summer REU internship at the University of Texas at Arlington.

PETRA '14, May 27 - 30 2014, Island of Rhodes, Greece.

ACM 978-1-4503-2746-6/14/05$15.00.
http://dx.doi.org/10.1145/2504335.2504405

a good trade-off between recognition accuracy and speed. Since object recognition methods are often used in real-time applications, sub-second response times are necessary in order for any such method to be of practical use.

The utility of geometric features is shown in [15], where Rusu et al. classify household objects such as cabinets and other common kitchen appliances. These objects are represented by planes and cuboids. Their specific labels are determined by computing the number of knobs and handles associated with the inspected plane or cuboid and using these two feature attachments to characterize the associated geometric structure.

After extracting the set of features, we employ well-known supervised learning methods to test the recognition accuracy and running time of our recognition approach on a dataset of 51 common household objects. Specifically we experiment with Support Vector Machines, Adaboost and Artificial Neural Networks using the same set of features. Accuracy levels and running times for each method are reported.

The remaining of the paper is organized as follows. Section 2 provides an overview of the related work in object recognition and the methods used in our experiments. Section 3 introduces our object recognition approach, describing the classification model construction at a high level and the feature extraction and machine learning processes at a low level. Section 4 provides details of how the performance of our recognition approach is assessed and reports the performance results, identifying trends in classification. Finally, section 5 concludes our paper.

2. RELATED WORK

The problem of object recognition has been studied in the past, in which cases, the authors used different approaches to achieve high recognition accuracy, usually based on 2-dimensional (2D) features. For example, in [10], the authors try to extract local, scale-invariant features that share similar properties with neurons in inferior temporal cortex that are used for object recognition in primate vision. In [1], the shape of the object is used as the main characteristic to achieve recognition by matching it with an existing object of similar shape, whose label is known.

Supervised learning has also been used used in the past for the task of object recognition/classification. There are multiple approaches that differ by the learning methods they employ. A common learning algorithm to employ for the task of object recognition is Support Vector Machines (SVM), where the maximal margin of divided data in a high dimensional feature space is found in order to distinguish and classify examples [14].

In [17], the SVM learning algorithm is trained on four shape features: the volume of the point cloud's convex hull, the standard deviation of the distances between the Kinect sensor and the points, the standard deviation of the distances between the centroid of the point cloud and the points, and a nine-parameter descriptor for the best fit ellipsoid.

In 2011, Lai and colleagues published a benchmark PCD dataset known as the RGB-D Object Dataset [8]. A year later, Lai and colleagues tested the performance of the SVM learning algorithm on the RGB-D Object Dataset [9]. Since then, variations of the SVM classifier such as linear SVM and Gaussian kernel SVM, as well as Random Forest (RF) have had their performances tested using the RGB-D Object dataset by Lai and colleagues [8]. Lai et al. hypothesized that their SVM-based classifier would perform more accurately than other classifiers because their classifier combined color and depth information provided by the RGB-D camera, while other classifiers did not. Lai and colleagues used the standard sliding window approach where the classification system evaluates a score function for all positions and scales in an image, and thresholds the scores to obtain object-bounding boxes.

To evaluate the performance of their classification system, Lai and colleagues trained their classifier on the point cloud data from their dataset, and then tested both category and instance level recognition. At the category level, the classifier's task was to classify previously unseen objects as a member of one of the 51 categories containing objects that had previously been seen. At the instance level, the classifier's task was to identify whether an object was physically the same object that had previously been seen. According to Lai and colleagues, both the category and instance levels were important tests to be performed because object recognition systems such as those embedded in robots may need to identify a generic object or a specific object, depending on the context of the task.

Lai and colleagues also evaluate the performance of their classification system by testing the runtimes of the different computational parts involved in their object recognition process. They find that their feature extraction and sliding window classification computations each take approximately 1.8 seconds per object [9]. The high cost of these two computations in their object recognition process lead them to conclude that their process does not currently run in real-time. Bo et al. [3] experiment with the same dataset and adopt an unsupervised feature learning approach, to object recognition.

In our work, we use the same RGB-D object dataset, but we propose a different feature extraction methodology that allows us to achieve recognition times suitable for real-time applications, while at the same time achieving the same or higher levels of accuracy compared to previous work from other researchers on the same dataset.

3. OBJECT RECOGNITION APPROACH

3.1 Description of the Dataset

The RGB-D Object Dataset that we used in our experiments, contains 300 objects organized into 51 categories, and for each object there are multiple pictures taken from different viewpoints, forming a total of $250,000$ sample pictures. Each sample consists of a pair of a portable network graphics (PNG) file and the corresponding point cloud data (PCD) file. The objects in this dataset were collected using an RGB-D (Kinect-style) camera [7]. This device is unique from other vision sensory devices in that it collects and merges color (RGB) and 3D depth-sensing information from the environment that it senses.

The RGB-D Object Dataset objects are common household objects. Although these objects come from a common domain, they are physically distinct and have been recorded from multiple viewpoints, providing information that makes it possible to distinguish one object from another, as well as one class of objects from another. The objects provided in the dataset have been previously segmented from their background. Figure 1 shows a 5-by-9 matrix of a subset of

Figure 1: A visualization of 45 objects from the dataset. Image courtesy of Lai et al. [8].

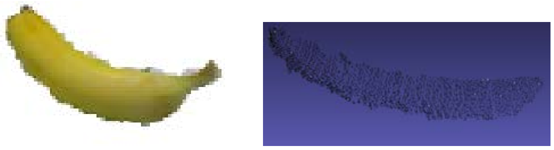


Figure 2: Banana Category Instance, RGB and PCD Representations

the objects used. For more information about the dataset, please refer to [8].

3.2 Extracted Features

Features are extracted from the PCD file representations of objects spanning over 51 categories from the RGB-D Object Dataset. Figure 2 below shows how a PCD instance from the "banana" category in the dataset would appear using either its color information, obtained from a PNG file and rendered using a basic preview tool, or its spatial information, obtained from a PCD file and rendered using Meshlab [5].

The goal of the feature extraction process is to extract valuable information from the PNG and PCD representations of an object in the RGB-D Object Dataset, such as the object representations in Figure 2. In this case, valuable information is information that assists the object recognition system in distinguishing any pair of instances that belong to different categories.

In this study, we extract a total of 13 features, which are later used as input to the learning algorithms. Nine of the features are derived from the RGB color histograms. For each of the red, green, and blue color histograms of an object, the statistical mean, standard deviation, and skew are computed, giving rise to a total of 9 color-based features. Three geometric features extracted for this study include geometric class correlations for a plane, a cylinder, and a sphere. Essentially, these three features describe how well the array of 3D data points in the PCD files can be reconstructed with a plane, a cylinder, and a sphere. One volumetric feature extracted for this study is the number of voxels occupied by the point cloud. Voxels provide a convenient measure of volume that is invariant to distance from the sensor (the raw number of data points corresponding to the object, for example, will change based on the distance to the sensor due to decreasing sensor sensing density at increasing distances).

The geometric features used in this experiment are extracted from the query object using RANdom SAmple Consensus (RANSAC)-based 3D object segmentation [6]. A geometric model for each of the classes is fit to the collection of points for the query object within a certain threshold (our fitting threshold is the euclidean distance of the point to the geometric model, which we set to 0.01 meters). To create a measure of model fitness, we divide the number of points belonging to the best-fit RANSAC model for each geometric class by the total number of points in the query point cloud. This normalized model fitness for each of the geometric classes is used as the classification feature in our experiments.

Our identification approach is intended to be used for real-time applications, thus we chose to utilize features that are bounded in terms of computation time. RANSAC based segmentation can be bounded in such a way by setting the maximum number of iterations to an amount that can be performed in an acceptable duration of time. For each of our geometric classes (plane, cylinder, and sphere), we limit the maximum execution time to 0.2 seconds. This guarantees that the total time allotted for computation of the gemoetric features is bounded to 0.6 seconds, which provides sufficient accuracy for our experiments. While this does not guarantee that the solution will converge to a best-fit model within the given time duration, we need only to obtain the maximum number of points that sufficiently fit any model within the distance tolerance. Each of the geometric features can be considered to be a response to that particular class, which we are able to utilize for identification.

3.3 Supervised Learning Algorithms

We model the problem of object recognition as a multi-class classification problem, where each object type, belongs to one of a set of predefined classes (categories). To classify each object, we follow the supervised learning paradigm, i.e. we first train a classifier with a set of objects of known labels (classes), and then we use that classifier to classify any new object to one of the classes encountered during the training process.

We have experimented with three of the most popular learning algorithms used in the computer vision literature, namely Adaboost, Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs). Each of these algorithms have their own advantages and disadvantages, however, as we will see in the experimental results section, the SVM turned out to be the most promising method for this problem, providing the best balance of speed and classification accuracy.

In the following subsections we give a brief description of each algorithm, and how we used it in our experiments.

3.3.1 AdaBoost

The application of the AdaBoost algorithm centers on a classification problem that involves instances, labels, and classifiers. In this problem, an instance is defined as a description used to derive a predicted classification. The space of all possible instances is known as the instance space. A label is a name that indicates the correct classification of an object. An instance together with an associated label is defined as an example. A classifier is a function that maps instances to labels, and can be viewed as a prediction rule [16].

Our implementation of AdaBoost takes as input a matrix containing the set of all observed features for each example, a vector containing the class labels corresponding with each example, and an integer value containing the number of training rounds to perform. During a given round k, AdaBoost aims to correctly classify the examples that were misclassified in rounds 1 through k-1. After AdaBoost trains for the number of rounds specified as an input argument, it outputs a classification model. During each round of training, AdaBoost determines a weak classifier which derives class predictions for each training example. The classification model is a vector of "best" weak classifiers, which is described in detail in the following section. The classification model also associates an alpha value with the best weak classifier for each training round, each corresponding with a best weak classifier. The alpha value α for a weak classifier describes that weak classifier's prediction accuracy. Alpha is calculated as follows:

$$\alpha = 0.5 \cdot log((1 - \epsilon)/\epsilon) \tag{1}$$

where ϵ is the error associated with the current round's best weak hypotheses. Once alpha has been calculated for the current training round, the set of boosted hypotheses can be updated by using both the weak hypotheses derived by the best weak classifier and alpha:

$$H = H + \alpha \cdot h \tag{2}$$

where H is the set of boosted hypotheses, and h is the set of weak hypotheses. The greater the value of alpha, the greater the effect of the weak hypotheses have when changing the boosted hypotheses. Before the round of training terminates, the distribution of weights across the training examples must be updated to reflect the updated boosted hypotheses:

$$W = \frac{exp(-H^T \cdot y)}{\sum_{i=1}^{N} exp(-H^T \cdot y)} \tag{3}$$

By making these updates, AdaBoost assures that each training round, the examples that have been most frequently misclassified will have the greatest weight. In other words, AdaBoost maintains a reward system where the greatest reward is achieved when examples misclassified in previous training rounds are correctly classified.

The AdaBoost algorithm boosts hypotheses that are binary classifications of instances, either +1 or -1. However, when there are more than two classes to predict from, it is necessary for these base hypotheses to express varying degrees of confidence. To achieve this, we implement an extension of the basic AdaBoost algorithm where the base learners report self-rated confidence scores that estimate the reliability of each binary prediction, and a one-versus-all approach to perform multi-class classification.

Weak Learners: During each round of training, the weak learner of our object recognition system creates a weak classification rule by using a decision stump, which is a decision tree with one test at the root [14]. Our weak learner takes as input a vector containing a single object feature extracted from all the examples, a vector containing the class labels corresponding with each example, and a vector containing the training weights, or importances, corresponding with each example. It outputs a weak classifier.

The weak classifier is a structure that consists of a best threshold, a best error, and a direction. The threshold is a floating point number value between the minimum and maximum feature values of the feature vector input. The threshold is found by stepping through possible threshold values with a step size proportional to the difference between the maximum and minimum feature values. The best threshold is chosen and updated when a threshold is found that divides the weighted training examples in a way such that the misclassification error is less than the previous best error. In such a case, the threshold being observed is recorded as the new best threshold, the corresponding error is recorded as the new best error, and a direction, either 1 or -1, is chosen based on whether or not the new best error is less than 0.5, respectively.

3.3.2 Multilayer Neural Network

The second learning algorithm used to construct a classification model is the multilayer artificial neural network (ANN) [14]. A neural network is composed of many perceptrons, forming an input layer, several hidden layers, and an output layer. At any given perceptron j, some number n of input features has the potential effect of activating j's output channel. Whether or not j's output channel is activated depends on on the result of several calculations. One can think of a perceptron as a training example in the dataset because the number of perceptrons corresponds to the number of training examples. First, a sum is taken of the i-dimensional input vector, multiplied by a vector of i weights corresponding with the i inputs. Mathematically, this sum over j's inputs is denoted as in_j, and is defined as:

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{4}$$

With this expression, one can see how the vector of inputs resembles neural signals and how the vector of weights resembles the strengths, or connectivity, of each signal.

Second, using the value computed for in_j, an activation function is used to determine how the perceptrons to the inputs, potentially propagating the input's information further through the neural network. The activation function for given perceptron j is defined as:

$$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j} a_i) \tag{5}$$

The activation function can be defined in many different ways, but the majority of the time, it is simply the logistic function. Thus, the activation function can usually be expressed as:

$$a_j = g(in_j) = Logistic(in_j) = \frac{1}{1 + e^{-in_j}} \tag{6}$$

Third, using the output of the activation function, some small constant α (i.e. 0.05), and the true label y of example j corresponding with perceptron j and the set of inputs used to compute (4), the feature input weights can be updated. During this update, features that are better category separators are assigned more weight and features that are poor category separators are assigned less weight. Mathematically, this weight update for a given feature i is defined as:

$$w_i = w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x})) \times x_i \qquad (7)$$

Throughout the training process, Equations (4), (6) and (7) are applied to update the weight distribution over the features at every layer of the neural network for a specified number of epochs. An epoch can be defined as an iteration where the weight distribution over the attributes is updated once for each example in the training set. For the current work, the number of epochs defined such that the neural network will continue training for more epochs until the weight distribution over the features converges, or 1000 epochs have been reached.

3.3.3 Support Vector Machine

The third machine learning algorithm used to construct a classification model is the Support Vector Machine (SVM) algorithm. The SVM algorithm works by constructing a model which represents the training or testing instances in a hyperplane. The SVM algorithm then generates support vectors which separate the instances by their categories. The SVM algorithm generates support vectors and hence makes predictions that maximize the distance of correctly classified instances from the support-vector defined decision boundaries.

Given feature vectors $\mathbf{x}_i$ and corresponding labels $y_i \in \{-1, 1\}$ for some arbitrary number of instances l, the SVM algorithm solves the optimization problem

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \xi_i \qquad (8)$$

subject to the constraints $y_i(\mathbf{w}^\mathbf{T}\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$.

Often, it is not easy to find an optimal linear separator in the input space $\mathbf{x}$, but it is possible to find linear separators in a feature space with higher dimensionality. To map the original input space to the new, high-dimensional feature space we make us of a kernel function $\phi(\mathbf{x})$. We use the (Gaussian) radial basis function (RBF) kernel. The RBF kernel on two samples $\mathbf{x}$ and $\mathbf{x}'$, represented as feature vectors in some input space, is defined as

$$K(\mathbf{x}, \mathbf{x}') = \exp(\gamma||\mathbf{x} - \mathbf{x}'||_2^2) \qquad (9)$$

where $\gamma = -\frac{1}{2\sigma^2}$.

To achieve optimal performance, the optimal values of parameters C and γ above, need to be determined. Before building classification models for the various-sized classification tasks, we optimize those input parameters using an exhaustive grid search, by cross-validation on the training set.

Since classification of the instances within the RGB-D Object Dataset involves multiple categories, the values of y, $y = -1$ and $y = 1$, correspond with the instance not being a member of some specified category and being a member of some specified category, respectively. This "one-versus-all" approach proves effective, and yielded the highest performance of any classifier, as it can be seen in the experiments section.

Prediction Accuracy vs. Runtime and Task Size		
Categories	Accuracy	Test Time Per Instance (s)
10	0.9419	0.030515
20	0.8519	0.062318
30	0.8246	0.093466
40	0.7685	0.12494
51	0.7146	0.16078

Table 1: AdaBoost performance.

Prediction Accuracy vs. Runtime and Task Size		
Categories	Accuracy	Test Time Per Instance (s)
10	0.9474	0.00001170526
20	0.8213	0.0000036827381
30	0.7824	0.000016679368
40	0.6765	0.00000674687
51	0.5157	0.000083871

Table 2: Artificial Neural Network (ANN) performance.

4. EXPERIMENTAL RESULTS

We evaluate the performance of our object recognition approach using the RGB-D Object Dataset described in section 3.1. In our experiments, we evaluate the performance of our object recognition system by recording the classification accuracy and the instance classification (testing) times, using the *HoldOut*, cross-validation option in Matlab, which returns logical index vectors for cross-validation of N observations by randomly selecting $P * N$ (approximately) observations to hold out for the evaluation set. We used $P = 0.3$. The training times of the system are not of interest to a real-time application since the training happens off-line, in advance, and thus are not reported. We have recorded these values for various size classification problems using 10, 20, 30, 40, and 51-categories from the dataset, to demonstrate how the classification accuracy and classification times scale as the size of the dataset and the number of classes increase.

Tables 1, 2 and 3 show the performance of Adaboost, ANN and SVM algorithms respectively. For each table, the first column lists the number of categories (classes) that were used in that experiment, the second column shows the classification accuracy in the range $[0, 1]$ (i.e. number of correctly classified instances over total number of instances), and the third column shows the average classification time per instance, in seconds.

Prediction Accuracy vs. Runtime and Task Size		
Categories	Accuracy	Test Time Per Instance (s)
10	0.974036	0.001079599
20	0.93888	0.0010226605
30	0.934808	0.002284086
40	0.916956	0.004052287
51	0.895562	0.00649148

Table 3: Support Vector Machine (SVM) performance.

4.1 Classification Accuracy

As it is easily noticeable, the three different learning algorithms show different behavior in terms of classification accuracy and speed. SVM turns out to be the most accurate algorithm, while ANN is the fastest one. Adaboost, at least with the current set of features, does not manage to excel in any of the two categories. SVM manages to achieve a 97.4% accuracy for the 10-class classification problem, and it continues to maintain a relatively high accuracy of 89.6%, all the way up to the 51-class classification problem. For comparison, a randomly guessing classifier would perform with approximately 10%, 5%, 3.3%, 2.5%, and 2% accuracy for the 10, 20, 30, 40, and 51 category classification tasks, respectively.

To give the reader a better overview of the misclassification errors that occurred during the classification process, in Figure 3 we plot the confusion matrix of the SVM performance for the 20-category problem[1]. The confusion matrices provide helpful information for identifying reasons for misclassification. When observing the accuracy and error percentages for arbitrary category i, notice that row i and column i provide different insights about the classification of this category. The index of the row gives the output class (prediction) of the category, while the index of a column gives the target class (actual label) of a category. Essentially, the sum of errors along row i gives the expectation of accuracy level for a prediction made to category i for instances from all categories, and the sum of errors along column i gives the average accuracy of all categorical label predictions for instances with true category label i. As one can notice, the most common misclassifications occur between object of similar size and shape, e.g. food can vs. food jar, or phone vs. camera, etc.

Adaboost had the second best of overall accuracy in our experiments, while the Artificial Neural Network was the worst performer among the three algorithms test, especially when tested with a larger number of classes. We should note here that we experimented with different parameters and different number of hidden layers for the ANN. The reported levels of accuracy are the best we could obtain, and that was when using 20 hidden layers.

For comparison, Lai et al. [8] report 83.8% classification accuracy using SVM with Gaussian Kernel as a learning algorithm in 2011, whereas the same team [3] reports a maximum of 87.5% accuracy for the 51-category classification problem, using their unsupervised feature learning approach, in their most recent work.

4.2 Classification Speed

One of the main goals of this work was to create and object recognition system that can be used in real-time applications. To achieve that goal we rely on a small set of features, described in section 3.2, which can be extracted fast and also allow for fast instance classification using existing learning algorithms, while at the same time providing enough information to achieve a high classification accuracy.

Tables 1, 2 and 3 list the test times for each algorithm and number of categories evaluated. Test time refers to the time that it takes for a classifier to classify one object to one of the predefined classes (categories). The reported times show the average time (in seconds) required to classify one object, however, the differences between different objects are negligible, since we use the same 13-dimensional feature vector type for every object, differentiating only the values of the features. The experiments were ran on a machine with 2 GHz Intel Core i7 processor and 4 GB 1333 MHz DDR3 memory. We used the Adaboost and ANN implementations in Matlab and the LibSVM [4] implementation of SVM.

As it can be clearly seen in the tables, ANN was by far the fastest classification algorithm, achieving times in the order of a few microseconds, regardless of the number of classes. SVM was the second fastest algorithm, taking a few milliseconds to classify each object, while the time seems to be increasing linearly with the number of classes, and subsequently the total number of objects in the training dataset. Adaboost was proven to be considerably slower, requiring about 0.16 seconds to classify each object for the 51-category set. Taking into consideration SVM's superior performance in terms of classification accuracy, it appears to be the learning algorithm of choice in this case.

When compared to the object recognition system of Lai and colleagues in [9] and Bo and colleagues in [2], it is evident that the our approach outperforms the state of the art object recognition systems with regard to classification times. While Lai and colleagues report a classification time of approximately 1.8 seconds, and Bo and colleagues report a classification time of 0.51 seconds, the SVM classifier in combination with our proposed set of features takes approximately 0.006 seconds, on average, to classify a tested object among all the categories in the RGB-D Dataset. This guarantees that even with a larger dataset and number of classes, the classification time will still remain well under a second.

For a real-time system, on top of the classification time, we should add the feature extraction time for each object, reported in section 3.2, which is independent of the type of the object and the number of objects in the training dataset.

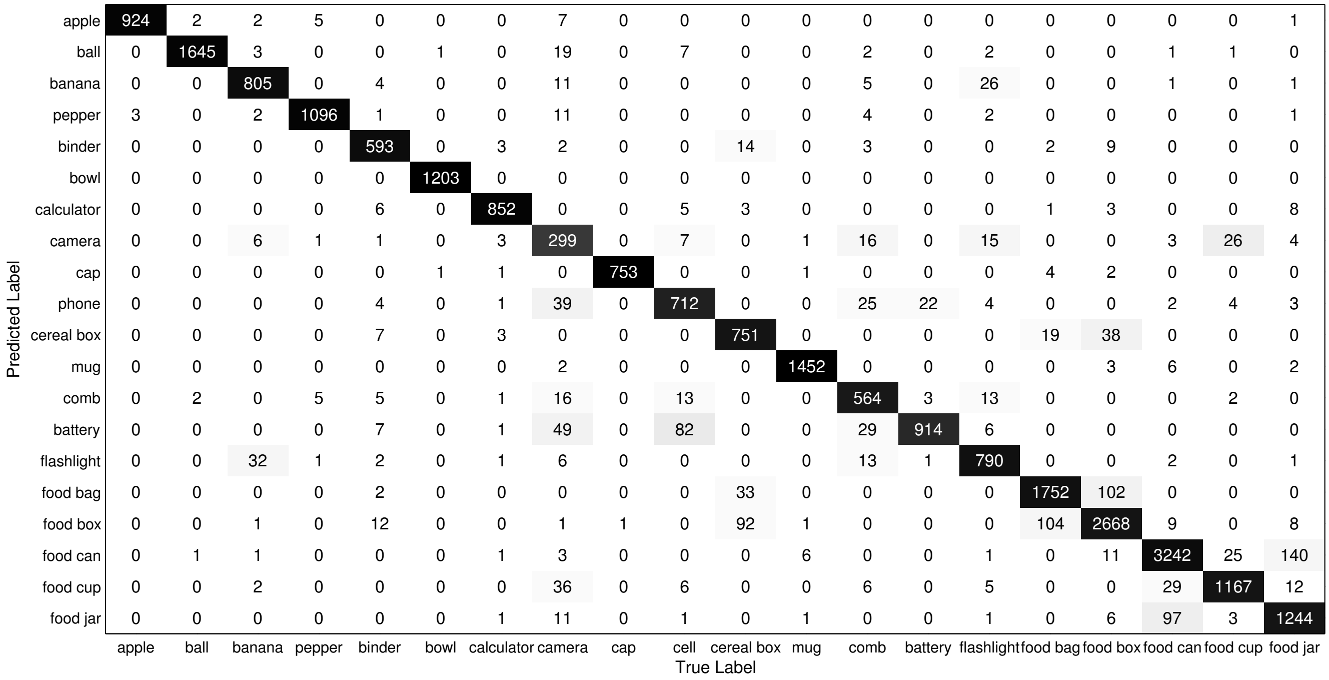
5. CONCLUSION AND FUTURE WORK

In this work have presented our proposed approach to creating an accurate and fast object recognition system, based on RGB-D imaging data. We introduce a novel feature extraction methodology and we evaluate it in combination with well-known supervised learning algorithms, on a publicly available RGB-D Object Dataset. Our system manages to overtake existing state-of-the-art approaches both in terms of classification accuracy and speed, allowing for a real-time object recognition system.

As it becomes obvious, the feature extraction process and the final set of extracted features, are of major importance to achieving high classification accuracy and high classification speed at the same time. Our system extracts color, geometric and volumetric features which are proven to be very successful in categorizing the majority of the objects in the dataset. However, as it can be observed from the confusion matrix, for objects which are similar in color, shape and volume (e.g. camera and cell-phone), we see higher misclassification rates.

One direction of future work that would confront this problem would be adding scale-invariant feature transform (SIFT) features to the feature space by extracting symmetry and texture information from the point cloud instances. For example, a camera may not be distinguishable by its color or geometric shape, but it often contains a protruding

[1] We chose to visualize 20 categories instead of 51 for readability reasons.



Predicted Label
True Label

	apple	ball	banana	pepper	binder	bowl	calculator	camera	cap	cell	cereal box	mug	comb	battery	flashlight	food bag	food box	food can	food cup	food jar
apple	924	2	2	5	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	1
ball	0	1645	3	0	0	1	0	19	0	7	0	0	2	0	2	0	0	1	1	0
banana	0	0	805	0	4	0	0	11	0	0	0	0	5	0	26	0	0	1	0	1
pepper	3	0	2	1096	1	0	0	11	0	0	0	0	4	0	2	0	0	0	0	1
binder	0	0	0	0	593	0	3	2	0	0	14	0	3	0	0	2	9	0	0	0
bowl	0	0	0	0	0	1203	0	0	0	0	0	0	0	0	0	0	0	0	0	0
calculator	0	0	0	0	6	0	852	0	0	5	3	0	0	0	0	1	3	0	0	8
camera	0	0	6	1	1	0	3	299	0	7	0	1	16	0	15	0	0	3	26	4
cap	0	0	0	0	0	1	1	0	753	0	0	1	0	0	0	4	2	0	0	0
phone	0	0	0	0	4	0	1	39	0	712	0	0	25	22	4	0	0	2	4	3
cereal box	0	0	0	0	7	0	3	0	0	0	751	0	0	0	0	19	38	0	0	0
mug	0	0	0	0	0	0	0	2	0	0	0	1452	0	0	0	0	3	6	0	2
comb	0	2	0	5	5	0	1	16	0	13	0	0	564	3	13	0	0	0	2	0
battery	0	0	0	0	7	0	1	49	0	82	0	0	29	914	6	0	0	0	0	0
flashlight	0	0	32	1	2	0	1	6	0	0	0	0	13	1	790	0	0	2	0	1
food bag	0	0	0	0	2	0	0	0	0	0	33	0	0	0	0	1752	102	0	0	0
food box	0	0	1	0	12	0	0	1	1	0	92	1	0	0	0	104	2668	9	0	8
food can	0	1	1	0	0	0	1	3	0	0	0	6	0	0	1	0	11	3242	25	140
food cup	0	0	2	0	0	0	0	36	0	6	0	0	6	0	5	0	0	29	1167	12
food jar	0	0	0	0	0	0	1	11	0	1	0	1	0	0	1	0	6	97	3	1244

Figure 3: SVM Confusion Matrix, 20-Category Classification

lens on one of its faces that serves as an asymmetry that distinguishes it from other small digital objects such as the calculator and cell phone. Of course any new set of features added, should be also efficiently extracted and should not add a big overhead to the classification algorithm.

Acknowledgments

This work is supported in part by the National Science Foundation under award numbers NSF-1258500, NSF-1035913, NSF-1041637, NSF-1338118. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 4 (2002), 509–522.

[2] Bo, L., Ren, X., and Fox, D. Unsupervised feature learning for rgb-d based object recognition. In *13th International Symposium on Experimental Robotics (ISER)* (2012).

[3] Bo, L., Ren, X., and Fox, D. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics* (2013), Springer, pp. 387–402.

[4] Chang, C.-C., and Lin, C.-J. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*, 3 (2011), 27.

[5] Cignoni, P., Corsini, M., and Ranzuglia, G. Meshlab: an open-source 3d mesh processing system. *Ercim news 73* (2008), 45–46.

[6] Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[7] Khoshelham, K., and Elberink, S. O. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors 12*, 2 (2012), 1437–1454.

[8] Lai, K., Bo, L., Ren, X., and Fox, D. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)* (2011).

[9] Lai, K., Bo, L., Ren, X., and Fox, D. Detection-based object labeling in 3d scenes. In *IEEE International Conference on Robotics and Automation (ICRA)* (2012).

[10] Lowe, D. G. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* (1999), vol. 2, Ieee, pp. 1150–1157.

[11] McMurrough, C., Rich, J., Conly, C., Athitsos, V., and Makedon, F. Multi-modal object of interest detection using eye gaze and rgb-d cameras. In *Proceedings of the 4th Workshop on Eye Gaze in Intelligent Human Machine Interaction* (2012), ACM, p. 2.

[12] McMurrough, C., Rich, J., Metsis, V., Nguyen, A., and Makedon, F. Low-cost head position tracking for gaze point estimation. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA)* (2012).

[13] McMurrough, C. D., Metsis, V., Rich, J., and Makedon, F. An eye tracking dataset for point of gaze detection. In *Proceedings of the Symposium on*

Eye Tracking Research and Applications (ETRA) (2012).

[14] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Pearson Education, Inc., 2010.

[15] Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems 56* (2008).

[16] Shapire, R. E., and Freund, Y. *Boosting: Foundations and Algorithms.* Massachusetts Institute of Technology, 2012.

[17] Shi, L., Kodagoda, S., and Ranasinghe, R. Fast indoor classification using 3d point clouds. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)* (2011).

[18] Sural, S., Qian, G., and Pramanik, S. Segmentation and histogram generation using the hsv color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (2002), vol. 2, IEEE, pp. II–589.

[19] Van De Weijer, J., and Schmid, C. Coloring local feature extraction. In *Computer Vision–ECCV 2006.* Springer, 2006, pp. 334–348.