Object Classification using RGB-D data for Vision Aids

Apples and Oranges

Kyle Chiang, Anthony Pelot, and Trisha Lian

I. INTRODUCTION

With the increased availability of cheap and reliable depth sensors, imaging systems can now use depth information to better detect and locate objects in a scene. New augmented reality (AR) systems, such as the Microsoft HoloLens, are now mounting depth sensors on their glasses to improve functionality.

The motivation behind our project is to perform object recognition using RGB-D (color and depth) data for a vision aid being developed at Stanford. This vision aid consists of a pair of AR goggles with an Asus Xtion depth sensor mounted on top of it. We would like to use the RGB-D data from this sensor to train a classifier that can recognize household items from a database. Information on the classified object can then be relayed to a user through the vision aid. This system has the potential to help the visually impaired navigate and perform everyday tasks more efficiently.

Our training data consists of an RGB-D data set from a team at the University of Washington. **We used two different 3D descriptors to extract feature vectors that describe each frame of RGB-D data. The input features to our algorithm are these descriptor vectors. We then use SVM to train a classifier that can output the predicted class of new RGB-D images.** Using test data we collected with our own experimental setup, we evaluated the effectiveness of our classifier.

The best cross validation results gave 89% accuracy while using the SHOTCOLOR descriptors and a subset of tested items. Experimental results from this model had a prediction accuracy of 83% when presented with testing data obtained from our experimental setup.

II. RELATED WORK

Object detection and recognition using RGB-D data is an ongoing research area in computer vision. In particular, there are a wide range of 3D descriptors that can be used as feature vectors[1]. For example, Lai et al. used a combination of SPIN descriptors on point clouds and SIFT descriptors on corresponding RGB images to train three classifiers (linear support vector machine (LinSVM), gaussian kernel support vector machine (kSVM), and random forest (RF)) using RGB-D data [2]. Their classifiers, however, were only tested on their own data set and not on data collected from different experimental setups. In addition, their feature vectors required extensive pre-processing of descriptors.

Instead of choosing standard descriptors, Bo et al. introduced a collection of novel features that utilize the depth map, the RGB image, and their corresponding point cloud [3]. These features capture specific aspects such as shape, edges, or relative object size. These features performed better then standard descriptors when trained with LinSVM. In a different paper, the same authors used sparse coding to learn hierarchical feature representations from raw RGB-D data [4]. In addition to object recognition, classifiers using RGB-D data have also been used for other computer vision tasks. For example, Goswami et al. use HOG descriptors and entropy maps extracted from RGB-D data to train an RF classifier to recognize faces.

III. DATASET AND FEATURES

A. Training Set

We use a dataset collected by a team at the University of Washington for research on RGB-D data [2]. This dataset contains 300 household objects grouped into 51 categories. The dataset was recorded using a Microsoft Kinect while the objects were rotated on a turn table to obtain multiple angle views. The camera was also mounted at different heights to obtain viewpoints of different angles from the horizon. Objects are segmented from the background and presented as textured point clouds. Each viewpoint and its corresponding point cloud acts as a single training example.

We trained on a subset of the 300 object categories in this set. Each object category contains data from a variety of different objects. For example, in the apple category there are different types of apples (e.g. Red, Green, Fiji). In total, we have around 3000 training examples per category (600 view points from 4-6 different objects in each category).

Figure 1 shows a small example of items contained in the training dataset.

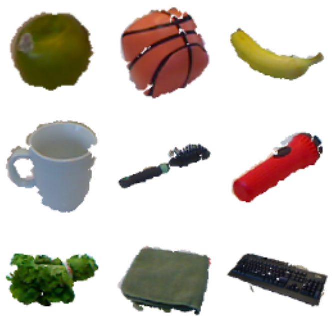

Fig. 1. Example images from our training data.

B. Testing Set

In order to test our classifier with completely new inputs, we collected our own RGB-D data. First, we mounted a calibrated Asus Xtion RGB-D sensor on a raised platform overlooking a table. Next, we performed a calibration step to obtain average depth and RGB images of the static background. We then placed objects in front of the sensor and captured RGB-D images. The object was automatically segmented from the image by selecting points that differ significantly from the static background. Multiple objects were separated into individual point clouds using connected component approaches. The depth and RGB information was then projected back out into 3D space to form a textured point cloud. This data matches the type of input we obtain from our training data set.

We collected textured point cloud data with our own objects from each category. We rotated the object by hand to obtain different viewpoints of each object. Like the training data, each viewpoint and its corresponding point cloud was a single testing example. In total, we had around 10-15 views of each of the 5 objects.

Figure 2 shows a few examples of our testing data gathered from our experimental setup. Figure 3 shows one textured point cloud of a mug from the testing data.

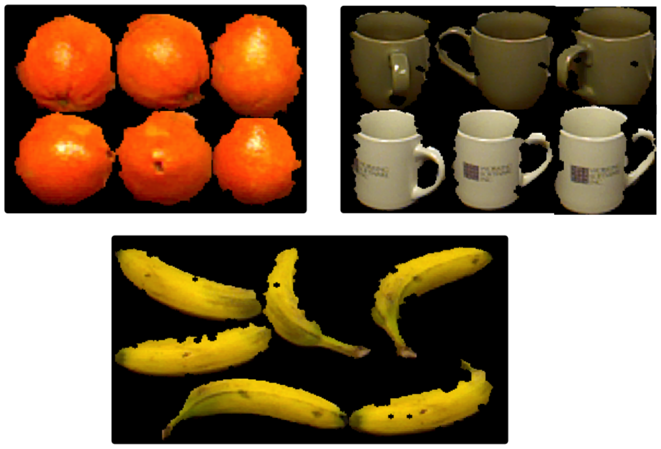
Fig. 2. Examples of several objects and different viewpoints from our test set.

C. 3D Descriptors

In order to represent our textured point clouds with a feature vector, we use 3D descriptors. 3D descriptors reduce a point cloud into a vector or histogram that captures the key aspects of the object. Although there are many available 3D descriptors, we focus on two: Viewpoint Feature Histograms (VFH) [5] and Signature of Histograms of Orientations with Color (SHOTCOLOR) [6]. Descriptors are calculated and manipulated using the Point Cloud Library (PCL) [7].

VFH captures both the viewpoint information of the point cloud as well as its geometry. It does the former by first calculating a vector between the viewpoint and the point cloud's centroid. It then bins the angle between this vector and the normals of each point into a histogram. To capture

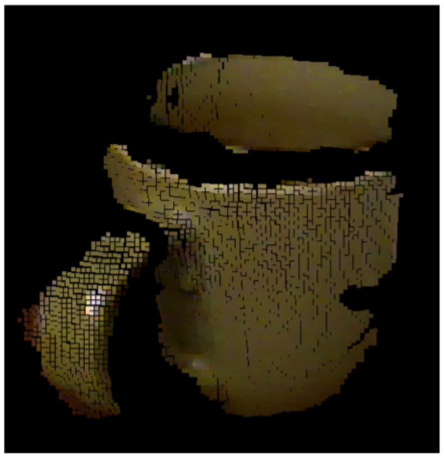
Fig. 3. Example of a textured point cloud from our testing set.

the geometry, it calculates a Fast Point Feature Histogram (FPFH) descriptor using the object's centroid. FPFH pairs neighboring points and bins each pair's euclidean distance and angular difference between their normals. This results in 4 histograms: 1 for the viewpoint, 3 for each angle in FPFH, and 1 more for the distance in FPFH. The final feature vector contains 308 values.

While VFH captures both geometry and viewpoint, it does not utilize the RGB data. SHOTCOLOR is a descriptor that incorporates both RGB texture and geometry. A spherical support structure is constructed around the point of interest (keypoint) and divided into 32 volumes. For each volume, the angle between the keypoint and each point is binned into a histogram. A similar procedure is performed for the texture information; color is converted into a vector (using the CIElab color space) and the angle between the keypoint and each point is also binned into a histogram. This results in a feature vector with 1353 values.

VFH is a global descriptor and therefore encodes the entire 3D geometry of a point cloud. On the other hand, SHOTCOLOR is a local descriptor and only describes the geometry around a single point. In order convert our local descriptor into a global one for our implementation, the centroid of each point cloud was found and the maximum distance between the centroid and each point was calculated. We then use the centroid as a single keypoint and set the radius of the descriptor to the be the maximum distance. Both descriptors are invariant to scale and SHOTCOLOR is also invariant to rotation.

We expect SHOTCOLOR to perform more accurately in situations where object geometry is similar but color is not. For example, apples and oranges are both spherical, but the former tends to be red (and occasionally green) while the latter is always orange. Despite the lack of color, VFH's inclusion of viewpoints might be advantageous when an object has unique geometry when viewed from a single angle. In addition, our method of converting a local descriptor to a global one for SHOTCOLOR is not standard and may reduce the effectiveness of the descriptor.

IV. Methods

A. SVM

To train our machine learning algorithm, we decided to use a multi-class SVM. A binary SVM classifier works by establishing a separating hyperplane, or decision boundary, within the n-dimensional representation of n features. This hyperplane is located such that it creates a constant margin between positive and negative training examples. This hyperplane can then be recreated for prediction purposes by only considering those training examples that exist exactly along the margin for the hyperplane. These training examples are called the support vectors. The optimal margin classifier can be solved via the following equation:

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, ..., m \end{aligned}$$

This constrained optimization problem can be solved by utilizing the Lagrangian dual problem where the primal consists of the form:

$$\min_{w} \theta_{\mathcal{P}}(w) = \min_{w} \max_{\alpha,\beta:\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

The dual can be similarly formed as:

$$\max_{\alpha,\beta:\alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha,\beta:\alpha_i \geq 0} \min_{w} \mathcal{L}(w, \alpha, \beta)$$

By finding w^*, α^*, and β^* that satisfy the Karush-Kuhn-Tucker (KKT) conditions, we find the solution to this dual optimization problem and the support vectors are found to be any training point in which the KKT condition of $g_i(w^*) \leq 0$ is actively constrained.

Additional complexity can be introduced by utilizing kernels that map features into a higher dimensional feature space. This allows much more flexibility in the fitting of the features, especially when the dimensionality of the feature set is low or irregular enough that a linear classifier cannot capture the complexity present in the training data.

To expand the binary classification to a multi-classification system, two approaches can be used. In the first, called One vs All, each class is compared to all the other classes, resulting in n classifiers for n classes. During prediction, the classifier with the greatest margin is determined to be the correct class. The second method, called One vs One, the classes are divided into pairs resulting in $\frac{n(n-1)}{2}$ total classifiers for n classes. This method can be much more computationally intensive, but tends to give better performance.

After obtaining descriptors of the training set, we implemented a multi-class SVM network using LIBSVM, which uses One vs One and supports a variety of kernels [8]. Kernels considered were the radix (RBF) kernel, a polynomial kernel, the sigmoid kernel, and a basic linear classification. Of the various kernels, the RBF kernel was determined to be the best to start with since it performs very similarly to the sigmoid kernel which can be invalid under certain paramaters and has fewer hyperparameters than a polynomial kernel [9].

However, when investigating the usage of RBF kernels, the support vectors would take hours to compute. Furthermore, with a training set of around 20,000 data points of dimension 1353, we would obtain over 1000 support vectors. The high number of support vectors and poor performance on cross validation data suggested that we were overfitting the data. To fix this problem, we switched to a linear classifier using LIBLINEAR, which uses One vs All, and the performance on cross validation data improved significantly [10].

B. Cross Validation

Because of the way our dataset was collected, there is a lot of structure and correlation between the various datapoints. For this reason, naively setting aside the last 10% of our data or a random 10% of our data for cross validation would not accurately represent the quality of our classifier. Because our data was comprised of a small number of objects with many views of each object, we implemented a version of k-fold holdout cross validation by holding out all views of each item. This way we can be assured that the cross validation data is independent of the training data.

Looking at the visual of class predictions for the 2 different descriptors we used in Figure 4 and of the confusion matrices in Figures 5 and 6 for SHOTCOLOR and VFH respectively, we confirmed our initial hypothesis that the additional color data captured by the SHOTCOLOR descriptor would give us better prediction results than that using the VFH descriptors.

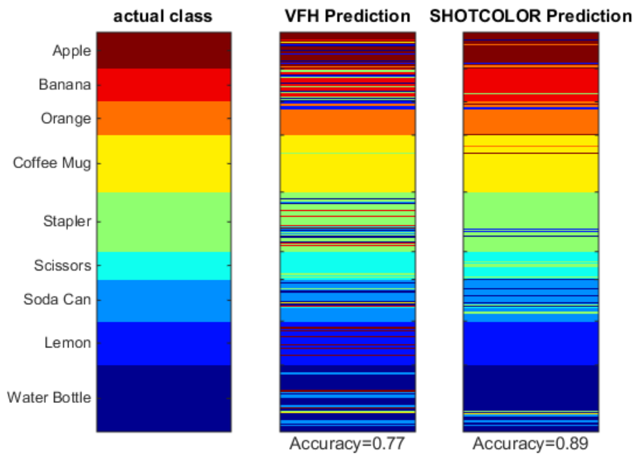

Fig. 4. Visualization of VFH and SHOTCOLOR classification. Each row is a point of training data and the color represents the class in which that training point was classified as in the holdout cross validation

V. Lab Tests

A. Testing Model

After training an SVM model on the training set, we proceeded to validate the quality of the model using our own test data captured in the lab. Unfortunately, we could not obtain all the objects in our original training set and our camera was unable to pick up objects with too many transparent or shiny surfaces like the water bottle and soda

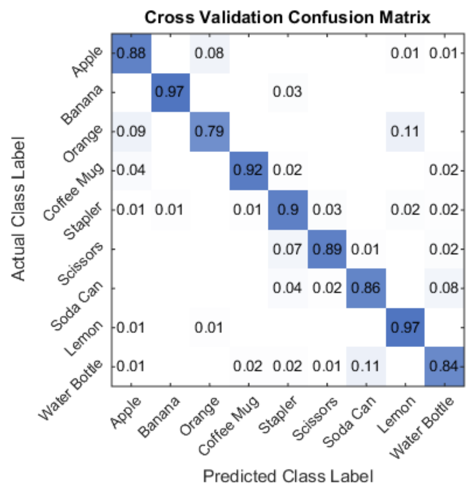

Fig. 5. Visualization of confusion matrix for cross validation when using SHOTCOLOR

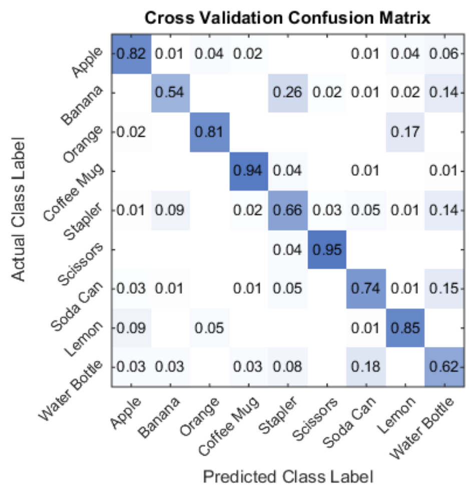

Fig. 6. Visualization of confusion matrix for cross validation when using VFH

can. For this reason, the model we used for our test set was smaller than that used to select the linear kernel for the SVM, classifying 5 objects instead of 9.

B. Results

Testing on our own data, we indeed showed that the SHOTCOLOR descriptor for training resulted in more accurate predictions compared to VFH. A visualization of the test predictions corresponding to each of these descriptors is shown in figure 7. Figures 8 and 9 show the confusion matrices for categorizing our test data. Overall experimental accuracy was 78% for VFH and 83% for SHOTCOLOR.

While the difference in accuracy between the two is not as drastic as the difference in the cross validation (78% to 83% in experimental tests versus 77% to 89% in cross-validation tests), it can be seen that SHOTCOLOR does a much better job categorizing globular fruit than VFH. In the testing set, VFH simply categorized nearly all the globular fruit as oranges, whereas SHOTCOLOR was able to differentiate them correctly with about a 60% to 70% accuracy. This difference can be improved if the categories were more finely split among colors. With green apples and red apples all in the training set under the same category, color doesn't help separate apples and oranges as well as if red apples and green apples were categorized as separate classes. Unfortunately doing this would give poor results because there are too few independent red and green apple data points in the training set.

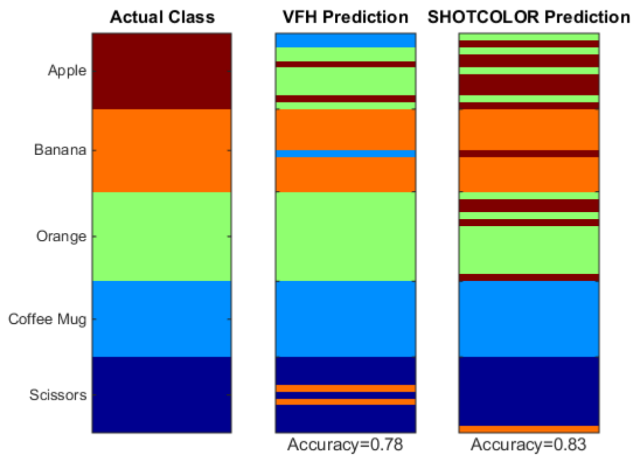

Fig. 7. Visualization of classification of test data

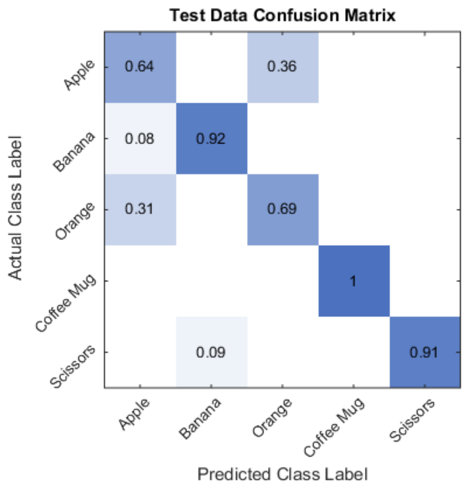

Fig. 8. Visualization of confusion matrix for test data when using SHOTCOLOR

VI. Future Work

In order to accurately identify a larger amount of household items, a much larger training set must be used. This can be done by generating additional data in the same way that the training set currently used was generated. Another option is to investigate the usage of widely available 3D models of objects. To use standard 3D models, they would first need to be converted to 3D point clouds before they could be

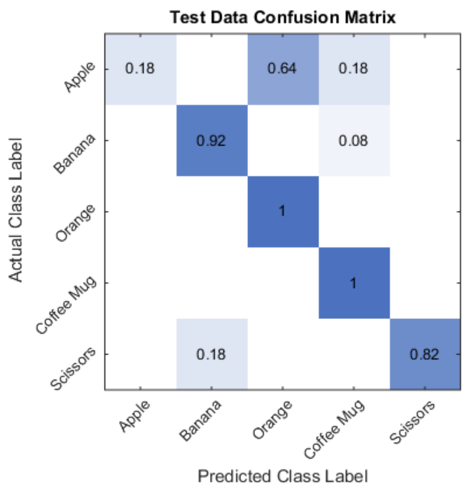

Fig. 9. Visualization of confusion matrix for test data when using VFH

converted to the appropriate feature set with 3D descriptors. One important difference in this process would be that a single model would represent a 360 degree view of the object and thousands of training examples per object for each view would no longer be required. This precludes utilizing view based descriptors such as VFH and may introduce additional complexities that need to be investigated. If successful, this would allow a training set to be compiled using freely available databases of 3D models.

Another important work to be addressed in the future is recognition within a scene. Currently, due to background subtraction, only the object introduced is considered for recognition. In a real-time system, calibration for background subtraction of each individual object in the room is not feasible. Additional algorithms must be introduced that will allow recognition of objects within the entire scene in order for true real-time operation to be possible.

VII. Conclusion

This investigation sought to implement a machine learning algorithm to accurately recognize household objects for the purpose of assisting the visually impaired using an AR system. A training set of 3D point clouds for household items was converted to appropriate features using 3D descriptors in PCL. Both VFH and SHOTCOLOR 3D descriptors were tested for comparison with and without considering color. A model based on a linear SVM multi-classifier was trained, cross-validated, and tested against data collected in our own experimental setup for each item. Using VFH, which does not consider color, cross-validation results were 77% and experimental results were 78% accuracy. Using SHOTCOLOR, which takes advantage of RGB data, cross-validation results were 89% and experimental results were 83% accuracy.

References

[1] Alexandre, Lus A. “3D descriptors for object and category recognition: a comparative evaluation.” Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal. Vol. 1. No. 2. 2012.

[2] Lai, Kevin, et al. “A large-scale hierarchical multi-view rgb-d object dataset.” Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011.

[3] Bo, Liefeng, Xiaofeng Ren, and Dieter Fox. “Depth kernel descriptors for object recognition.” Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[4] Bo, Liefeng, Xiaofeng Ren, and Dieter Fox. “Unsupervised feature learning for RGB-D based object recognition.” Experimental Robotics. Springer International Publishing, 2013.

[5] Rusu, Radu Bogdan, et al. “Fast 3d recognition and pose using the viewpoint feature histogram.” Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, 2010.

[6] Tombari, Federico, Samuele Salti, and Luigi Di Stefano. “A combined texture-shape descriptor for enhanced 3D feature matching.” Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE, 2011.

[7] Rusu, Radu Bogdan, and Steve Cousins. “3d is here: Point cloud library (pcl).” Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011.

[8] Chang, Chih-Chung, and Chih-Jen Lin. ”LIBSVM: A library for support vector machines.” ACM Transactions on Intelligent Systems and Technology (TIST) 2, no. 3, 2011.

[9] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. ”A practical guide to support vector classification.” 2003.

[10] Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. ”LIBLINEAR: A library for large linear classification.” The Journal of Machine Learning Research, 9, 2008.