A Textured Object Recognition Pipeline for Color and Depth Image Data

Jie Tang and Stephen Miller and Arjun Singh and Pieter Abbeel

***Abstract*— We present an object recognition system which leverages the additional sensing and calibration information available in a robotics setting together with large amounts of training data to build high fidelity object models for a dataset of textured household objects. We then demonstrate how these models can be used for highly accurate detection and pose estimation in an end-to-end robotic perception system incorporating simultaneous segmentation, object classification, and pose fitting. The system can handle occlusions, illumination changes, multiple objects, and multiple instances of the same object. The system placed first in the ICRA 2011 Solutions in Perception instance recognition challenge. We believe the presented paradigm of building rich 3D models at training time and including depth information at test time is a promising direction for practical robotic perception systems.**

I. Introduction

Object recognition in unstructured scenes is a challenging area of ongoing research in computer vision. One important application lies in robotics, where the ability to quickly and accurately identify objects of interest is crucial for general-purpose robots to perform tasks in unstructured everyday environments such as households and offices.

The specific problem of perception for robotics has a number of unique features which differentiate it from other problems in computer vision. A general object recognition system needs to deal with a vast number of different objects. One way of dealing with this is by introducing hierarchy, doing recognition on the category level instead of the instance level. The challenge of generalizing from a few instances to an entire category of objects remains difficult, and numerous benchmarks and challenge problems like Caltech 256 [1] and PASCAL VOC [2] exist to help drive progress in this area. By contrast, for a specific robot in a specific environment, the number of unique objects is relatively small (perhaps on the order of hundreds). This makes it possible to treat it as an instance recognition problem, gathering a large amount of training data per object. In addition, a robot can take advantage of data from multiple sensing modalities such as cameras and depth sensors.

Perception for robotics also presents additional challenges which are not present in category-level object recognition benchmarks. Real world environments are highly cluttered, contain many occlusions, and frequently contain five or ten different objects in the same scene. Robots must often avoid or manipulate objects in their environment. This means that a robotic perception system needs to accurately localize objects after detecting them. Additionally, for a robot to react quickly to changes in its environment, a robotic perception system needs to operate in near real time.

In this paper we present an approach for instance recognition and object localization in cluttered everyday scenes using color images and depth information.

Our system consists of a separate, offline training stage and an online test stage. During training, we are given color images and point clouds of each object from multiple views taken from a Kinect [3] sensor. These images are used to build a full point cloud from which we construct a 3D mesh model of the object (Section III-B). Next, we extract local image features from each training image and register them to the 3D model to create object models (Section III-C). Finally, we extract global descriptors from the full point cloud (Section III-D).

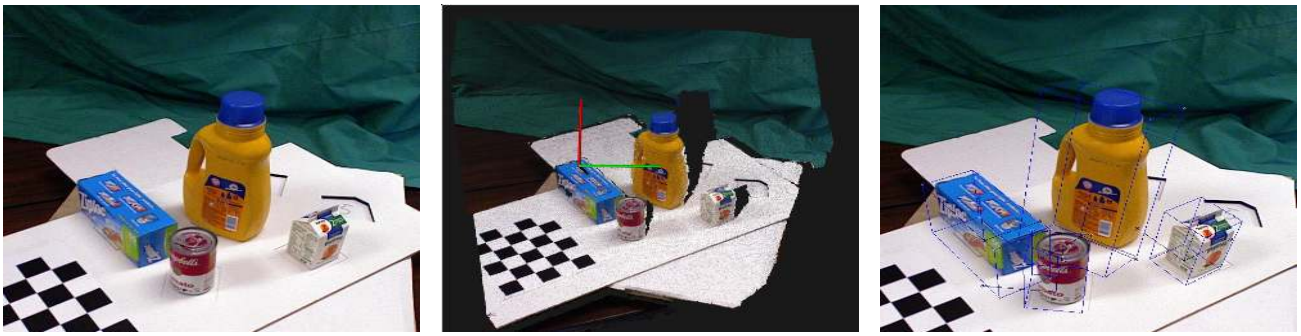
At test time, given a new color image and point cloud (possibly containing multiple objects), our approach segments the scene into individual objects by fitting a supporting plane and using depth information (Section IV-A). For each segmented object, we generate multiple object hypotheses using our extracted global and local feature models (Section IV-B). We then recover the pose of the object using our object models. (Section IV-D). Finally, we enforce global scene consistency checks to rule out geometrically incompatible hypotheses and refine our segmentation, rerunning the detection and pose recovery steps as necessary. (Section IV-F).

We present experiments (Section V) on a recent Kinect-based textured object dataset, demonstrating the benefits of simultaneous segmentation, object detection, and 3D pose recovery. An earlier version of our system placed first in the inaugural Solutions in Perception Challenge instance recognition challenge, held by Willow Garage at ICRA 2011 [4]. Figure 1 shows an example test image and test point cloud, together with the object detections and localization made by our pipeline.

II. Related Work

There have been several previously published approaches to joint object recognition and pose recovery in the literature. Gordon and Lowe [5] use SIFT features and structure from motion to register a camera pose against a known object model. Our work is inspired by the MOPED system by Collet et al. [6], [7], which uses SIFT features to build sparse, 3D local descriptor representations of textured objects at training time; at testing time, SIFT features are extracted, matched, and used to obtain a pose estimate. Our approach incorporates depth information at multiple stages of the

Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94709 {arjun, pabbeel, jietang, sdmiller,}@eecs.berkeley.edu



(a) (b) (c)

Fig. 1. (a) Example test image (b) Example test point cloud (c) Sample object detections.

TABLE I
SUMMARY OF NOTATION

Symbol		Description
I	$\sim$	a color image (III-A)
C	$\sim$	a point cloud (III-A)
p	$\sim$	a 6DOF pose for an object (III-A)
y	$\sim$	a training label for an object (III-A)
$\mathbb{C}$	$\sim$	a full object point cloud (III-B)
$\mathbb{D}$	$\sim$	a 3D mesh model (III-B)
x	$\sim$	a 2D point of interest (III-C)
z	$\sim$	a 3D point of interest (III-C)
f	$\sim$	a 128-dimensional SIFT descriptor (III-C)
$\mathbb{M}$	$\sim$	a 3D object model (III-C)
H	$\sim$	global object descriptors (III-D)

processing pipeline. During training we build 3D mesh models in addition to 3D metric feature representations, and at test time we use depth to segment objects and verify scene consistency. Our framework supports additional global descriptors such as color in the classification process.

A number of different approaches have been presented for incorporating depth information. Several approaches rely on extracting feature descriptors from the depth data, including spin images [8], point feature histograms [9], [10], or histograms of oriented gradients on the depth image [11], [12].

Several recent datasets have been created using the Kinect sensor to gather color and depth images of household objects. The textured object training data and testing data used in this paper comes from the Solutions in Perception challenge at ICRA 2011 [4]. Lai et al. [11] recently presented a larger color and depth image dataset for category and object recognition, containing both textured and untextured objects.

III. MODELING OBJECTS

A. Overview

An overview of our object recognition and pose recovery pipeline is given in Figure 2. A summary of the notation used in the remainder of this paper is given in Table I.

At training time, we require a set of N_I labeled training instances $(\{I_i, C_i\}, \{p_i, y_i\})$, $i = 1, ..., N_I$, where y_i specifies an object label, I_i is a color image containing one object, the object y_i, C_i is an associated 3D point cloud containing position and color information, and p_i is the 6DOF pose of the camera and depth sensor in a known reference frame. Ideally, for each unique object y_l, $l = 1, ..., N_L$, there exist multiple training instances with $y_i = y_l$ together covering all visible areas of the object. Using the image and point cloud data, we create a 3D point cloud $\mathbb{C}^{y_l}$, a 3D mesh model $\mathbb{D}^{y_l}$, a 3D object model $\mathbb{M}^{y_l}$ mapping image features to 3D locations, and a set of global object descriptors H^{y_l} (hue histograms) for each of the N_L unique objects in our training set.

B. Building 3D Models

Since we are given the camera pose p_i associated with each image and point cloud pair (I_i, C_i), we can create a complete 3D point cloud model $\mathbb{C}^{y_l}$ by combining the known camera poses to register all point clouds belonging to object y_l into a single coordinate frame. We then segment away the table plane and perform Euclidean distance-based clustering to extract the actual object[1], keeping the largest cluster.

This 3D point cloud model $\mathbb{C}^{y_l}$ is at best a noisy representation of the actual object due to sensor errors and camera pose measurement errors.[2] To address this, we used an off-the-shelf Poisson surface reconstruction tool [14] to construct a 3D mesh model $\mathbb{D}^{y_l}$ for each object y_l. Poisson reconstruction smoothly regularizes inconsistencies in the full 3D point cloud and fills in small unobserved gaps in the model.

C. 3D Object Models

Given the 3D mesh model $\mathbb{D}^{y_l}$, we can use our known camera poses p_i to project $\mathbb{D}^{y_l}$ onto each training image I_i which contains object $y_l = y_i$ in our training set. This projection is used as an accurate segmentation mask for the object.[3] After applying the mask to get a segmented image, we extract 2D SIFT [15] interest points $\{x_{ir}\}, r = 1, ..., N_R$ and associated feature descriptors $\{f_{ir}\}, r = 1, ..., N_R$[4], and project them onto $\mathbb{D}^{y_l}$ to get a 3D location $z_{ir} =$

[1]We used an off the shelf Euclidean clustering algorithm available in PCL [13]

[2]This initial alignment could in principle be improved by aligning the point clouds using e.g. an iterative closest points (ICP) algorithm, but the 3D mesh modeling process already performs some level of denoising.

[3]In practice we also include a buffer region of size N_{Segment} around the edge of the object

[4]We used OpenCV's SIFT library in our implementation [16].

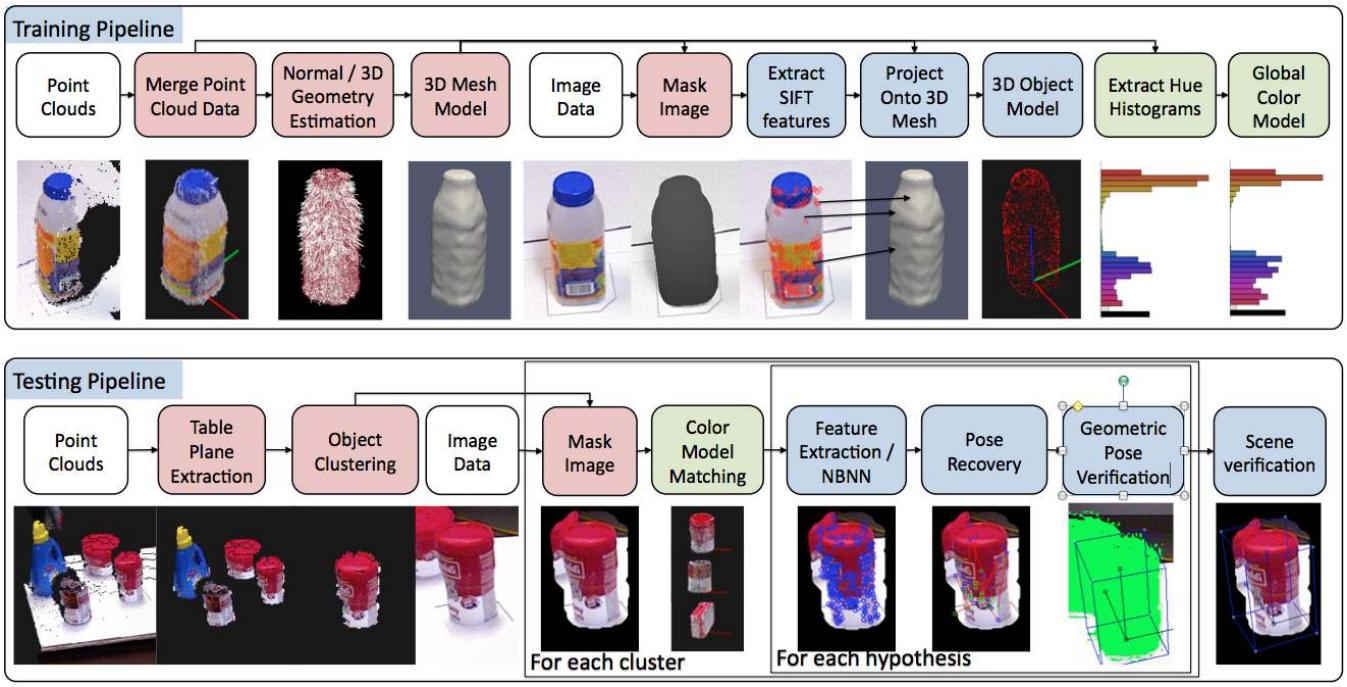


Fig. 2. Overview of our training and testing pipelines. Red boxes correspond to steps which make use of point cloud and depth information. Green boxes correspond to steps which make use of global color information. Blue boxes correspond to steps which make use of local SIFT feature information.

$\text{Proj}(x_{ir}, p_i, \mathbb{D}^{y_i})$.[5] We ignore interest points which do not project onto $\mathbb{D}^{y_l}$.

The collection of all such SIFT descriptor and 3D point pairs $\{(z_{ir}, f_{ir})\}$ forms an object model $\mathbb{M}^{y_l}$ which consists of a sparse collection of 3D locations with associated SIFT feature descriptors. The SIFT features can be used in a bag-of-words-style object classifier [18], while the 3D locations enable fast, accurate pose recovery.

D. Hue Histogram Descriptors

Given the full 3D point cloud $\mathbb{C}^{y_l}$ for object y_l, we simulate synthetic views of the point cloud from N_J camera poses $\{\widehat{p}_j\}$, $j = 1, ..., N_J$ around the object.[6] For each simulated pose $\widehat{p}_j$, we construct a global color histogram descriptor H_j using only points visible from that view. We first convert RGB to HSV. Points in the cloud which have very low or very high saturation have unreliable hue readings and are therefore counted in one of two special low saturation or high saturation bins, respectively. For points with medium saturation, we construct a 25-dimensional histogram of hue values using a soft binning scheme.[7]

[5]In practice we precompute the 2D locations of every face in $\mathbb{D}^{y_l}$ visible from camera pose p_i, and check to see which face contains x_{ir} using a point-in-polygon algorithm [17]. We then intersect a ray from p_i through x_{ir} with the given face to get 3D location z_{ir}.

[6]In practice, we generate synthetic poses 0.5m away from the object, at 36 different azimuths ϕ from 0 to 2π and 9 different inclinations θ from $\pi/4$ to $3\pi/4$.

[7]Our soft binning scheme divides the count for a given hue value between the two closest histogram bins. If x is the hue value, and x_i, x_j are the centers of the closest and 2nd closest histogram bins i, j, respectively, we would add weight $\frac{x-x_i}{x_j-x_i}$ to bin i and $\frac{x_j-x}{x_j-x_i}$ to bin j. We normalize the final hue histogram.

Our final color descriptor is normalized and has 27 dimensions, 25 for hue and 2 for low and high saturation. For each object y_l, the collection of hue histograms H_j for each pose $\widehat{p}_j$ forms the global descriptor model H^{y_l}.

IV. Object Detection

At testing time, we are given a color image and point cloud (I, C) (possibly containing multiple objects), and the goal is to recover the set of labels and poses $\{y_k, p_k\}$, $k = 1, ..., N_K$ for each of the N_K objects in the input data. Our system first segments and clusters the scene into potential objects using supporting plane and depth information. Next, we extract local SIFT features and a global hue histogram from the image, and use a Naive Bayes Nearest Neighbor classifier to match them against our object models and global color models $\{\mathbb{M}^{y_l}, H^{y_l}\}$, $l = 1, ..., N_L$. This yields a likelihood for each potential object label y_l. For the best scoring object label hypotheses, we use the SIFT feature matches and our 3D object models $\mathbb{M}^{y_l}$ to estimate a pose using RANSAC. We then run a series of geometric verification steps to match our learned 3D models $\mathbb{M}^{y_l}, \mathbb{D}^{y_l}$ against the observed scene. High probability object detections are removed from the test image and point cloud. The detection pipeline is repeated on any remaining object clusters to recover from undersegmentation errors. Finally, oversegmentations are handled by merging together consistent object hypotheses which occupy the same region of space.

A. Segmentation

Given a color image I and point cloud C of a scene, we first attempt to locate large planar surfaces in C using RANSAC [19]. Large planar surfaces are interesting because

we assume objects must be supported by a table or ground plane. Each candidate plane is represented as a surface normal plus an offset. If multiple planar surfaces share the same normal and offset (within a tolerance ϵ_{Plane}), we merge the two surfaces into one. For each planar surface, we find what parts of the point cloud (if any) lie above the plane. All supporting plane hypotheses with more than N_{Plane} points lying above it are treated as possible supporting planes.

The remainder of the pipeline processes each supporting plane hypothesis separately. For each supporting plane hypothesis, we remove all points which do not lie above the plane, and run a Euclidean distance-based clustering algorithm on the remaining point cloud to obtain individual object point clouds $\{C_m\}$, $m = 1, ..., M$. These point clouds are reprojected onto the original test image I to obtain M masked image and point cloud pairs $\{I_m, C_m\}$ as segmentation candidates for the object classification stage.

B. Object Classification using NBNN

For each test frame, the segmentation phase produces a set of segmentations $\{I_m, C_m\}$. For each segmented object we extract SIFT interest points and descriptors (x_{mr}, f_{mr}) from the masked image I_m, and a hue histogram H_m (Section III-D) from the point cloud C_m.

We use a Naive Bayes Nearest Neighbor (NBNN) classifier [20] to match the hue histogram and SIFT descriptors against our trained object models $\{\mathbb{M}^{y_l}, H^{y_l}\}$. The likelihood of an object label y given a set of descriptors $\{f_r\}, H$ is given by

$$\begin{aligned} P(y|\{f_r\}, H) &\propto P(y)P(\{f_r\}, H|y) \\ &\propto P(H|y)\prod_r P(f_r|y) \end{aligned}$$

Here we have assumed a uniform prior over object labels and that feature descriptors are conditionally independent given an object label (the Naive Bayes assumption). To model $P(f_r|y)$, we use kernel density estimation: if $f_{l1}, ..., f_{lN_R}$ are the descriptors in the feature model $\mathbb{M}^{y_l}$ for object y_l, then

$$P(f_r|y_l) = \frac{1}{N_R}\sum_{i=1}^{N_R} K(f_r - f_{li})$$

where $K(f_r - f_{li}) = \exp(-\|f_r - f_{li}\|_2^2/(2\sigma^2))$ is a Gaussian Parzen kernel function.

The core insight behind NBNN is that SIFT descriptors are high dimensional and are therefore sparsely distributed throughout the space. Since the kernel function $K(\cdot)$ drops off exponentially with distance, the distance between f_r and its nearest neighbor $f_{lNN(r)}$ can be used to compute an efficient approximation of $P(f_r|y_l)$. Therefore, the log likelihood of each object label can be computed as

$$\log P(y_l|\{f_r\}, H) = \alpha\|H - H_{lNN}\|_2^2 + \sum_r \|f_r - f_{lNN(r)}\|_2^2$$

where $f_{lNN(r)} = \arg\min_i \|f_r - f_{li}\|_2^2$ is the nearest neighbor to f_r in $\mathbb{M}^{y_l}$ (H_{lNN} is defined analogously for the hue histogram descriptor). The above log-likelihood allows us to create a ranked list of the most likely object hypotheses $\{y_{mn}\}$, $n = 1, ..., N_{\text{SIFT}}$ for each input segmentation m.

C. Pose Recovery

For each of the N_{SIFT} most likely object hypotheses, we next determine the most likely corresponding 6DOF pose. Given an object hypothesis y_l, for each of the SIFT interest points and descriptors (x_{mr}, f_{mr}) extracted from the color image I_m, we first find the nearest neighbor $f_{lNN(r)}$ of f_{mr} in our 3D object model $\mathbb{M}^{y_l}$. Since each descriptor $f_{lNN(r)}$ has a corresponding 3D location $z_{lNN(r)}$, this lets us match each 2D interest point x_{mr} with its true 3D location $z_{lNN(r)}$.

If we knew the true pose p_m of the object, it would satisfy $x_{mr} = \text{Proj}(z_{lNN(r)}, p_m)$, i.e. each 2D interest point x_{mr} is the projection of the 3D location $z_{lNN(r)}$ onto the image plane of a camera located at p_m.

However, because we may have incorrect 2D to 3D matches, instead of looking for an exact matching pose, we find the pose p_m which minimizes the squared error between our 2D points and the 2D reprojections of the corresponding 3D locations. More concretely, if we let $x = \text{Proj}(z, p)$ be the 2D projection of a 3D point z onto the image plane given by camera pose p, our optimization problem becomes:

$$p_m = \min_p \sum_r \|x_{mr} - \text{Proj}(z_{lNN(r)}, p)\|_2^2 \qquad (1)$$

This problem is nonlinear because of the projection operator. We solve this using a Levenberg-Marquardt nonlinear least squares algorithm [21], [22].[8]

Because we use nearest neighbors in SIFT descriptor space to generate our 2D / 3D correspondences, some of our correspondences are likely to be incorrect. We account for this using RANSAC. For each iteration of RANSAC, we solve Equation (1) on a random subset of our 2D / 3D correspondences. For each pose p_m, we count the number of correspondences which are explained by p_m by evaluating

$$f(p_m) = \sum_r \mathbf{1}(\|x_{mr} - \text{Proj}(z_{lNN(r)}, p_m)\|_2^2 < \epsilon_{\text{RANSAC}})$$

and retain the pose p_m with the highest number of matching correspondences after N_{RANSAC} iterations.

D. Geometric Pose Verification

Following the pose fitting step, each candidate pose $\{(y_{mn}, p_{mn})\}$ (where (y_{mn}, p_{mn}) corresponds to the nth candidate object label and pose for the mth segmentation) is subject to a geometric pose verification step which attempts to account for SIFT features present in the image I_m which were not matched to the label y_{mn}. We project the 2D locations x_{mr} of *all* SIFT features f_{mr} detected in I_m onto

[8] In practice, we use a fixed position (0,0,1) and a random quaternion whose components are chosen uniformly $\in [-1, 1]$ as the initial guess for Levenberg-Marquardt optimization.

the 3D object model $\mathbb{D}^{y_{mn}}$ (using our candidate pose p_{mn}) to obtain a 3D position z_{mr} for each descriptor f_{mr}. This includes SIFT features which did not match the object during the initial pose recovery step. We then search the 3D object model $\mathbb{M}^{y_{mn}}$ for a 3D feature point / SIFT descriptor pair (z_{mnr}, f_{mnr}) such that z_{mr} lies within a local neighborhood of z_{mnr} and f_{mr} lies within a local neighborhood of f_{mnr}.[9] Poses are sorted by the number of matches between the 3D positions of the test image features and the 3D object model.

The intuition for this pose verification step is to ensure that most SIFT features found in the image can be explained by the given object and pose hypothesis. This allows us to correct for SIFT features which were incorrectly matched during the initial classification step. It also helps reject errors in object classification and pose recovery.

The output of the pose verification step is a pose hypothesis (y_m, p_m) for segmentation m. We run a final round of Levenberg-Marquardt pose optimization on p_m (Equation (1)) using all consistent 2D / 3D feature matches.

If multiple object hypotheses (y_{mn}, p_{mn}) pass the geometric pose verification test, we take the object hypothesis (y_m, p_m) which yields the most consistent 2D / 3D feature matches in the test image I_m.

E. Recovering from Undersegmentation

Undersegmentations result in a candidate object which actually consists of several objects contained in the same cluster. When this occurs, the classification and pose verification steps can match only one of the objects correctly.

To amend this, after we have finalized our correct object hypotheses y_m we remove the points in the test point cloud C contained in the bounding volume of any object. We then re-segment the modified point cloud C' to get new candidate objects $\{I'_m, C'_m\}$, and run them through the classification, pose-fitting, and geometric verification stages of our pipeline.

F. Scene Consistency

After all candidate objects $\{I_m, C_m\}$ have been processed, our system checks all accepted object hypotheses (y_m, p_m) for overlaps. If two hypotheses have the same object label, and their bounding volumes overlap, we merge the two hypotheses by keeping the pose with more consistent SIFT feature matches. This helps eliminate spurious matches due to oversegmentation, where an object that is split into two object clusters generates two object hypotheses with roughly equivalent poses. We do not modify object hypotheses which overlap but do not share the same object label.

In addition, a separate check ensures that each object hypothesis (y_m, p_m) is consistent with the original point cloud C.[10]

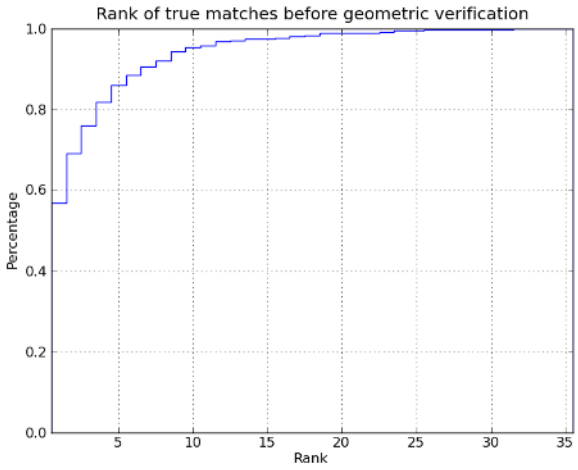


Fig. 5. Cumulative histogram of the rank of the true object after global and local feature matching but before geometric pose verification on the Willow challenge dataset. The true object lies in the top 15 over 95% of the time.

V. Experiments

A. Datasets

We evaluated our training and testing pipeline on two textured household object datasets (which we refer to as Willow and Challenge) and one synthetic textured object dataset (which we refer to as NIST) used for the Solutions in Perception Challenge [4]. The household object datasets contained 35 rigid, textured, household objects provided by Willow Garage (Figure 3). These objects were imaged using a Kinect sensor on a calibrated turntable rig, providing 360 degree coverage at a single azimuth angle. The Willow household object data set was released before the competition and contained roughly 1000 training instances. Each instance consisted of a Kinect point cloud, a color image, and a ground truth pose for a particular object at a particular azimuth. The Willow dataset also contained roughly 500 test instances consisting of Kinect point clouds and color images. Each of these frames contained multiple objects. The Challenge dataset was used for the challenge itself, and also contained roughly 1000 training instances of the same 35 household objects as the Willow dataset, together with 120 test instances (Kinect frames) containing a total of 434 objects. Finally, the synthetic textured object dataset consisted of 15 synthetic textured objects provided by NIST (Figure 4). This NIST dataset was used only for the challenge and consisted of 450 training instances (Kinect frames, color images, and ground truth pose of a single object) and roughly 400 test instances containing 831 object instances.

[9] More precisely, we require that $\|z_{mnr} - z_{mr}\|_2^2 \leq \epsilon_{\text{Pose}}$ and $\|f_{mnr} - f_{mr}\|_2^2 \leq \epsilon_{\text{Feature}}$.

[10] This is done by projecting the 3D bounding volume of each object hypothesis onto the observed test image. A percentage P_A of the area of the image occupied by the projection of the 3D bounding volume must also be occupied by the projection of the observed point cloud. For our experiments we use $P_A = 30\%$.

Fig. 3. Five of the thirty-five textured household objects from the Willow and Challenge dataset [4].

Fig. 4. Five of the fifteen synthetically made NIST challenge objects.

Figure V-B shows some examples of the test data for both Willow and NIST datasets. [11]

Building a single object model using our training pipeline on 35 training image / point cloud pairs takes about 7 minutes.[12] Running our object recognition pipeline on a single color image / point cloud pair (with a 35 object database) takes about 20 seconds. All tests were run on a 6-core 3.2GHz Core i7 with 24GB of RAM.

B. Instance Recognition

We first evaluate the performance of our pipeline in a single object recognition and pose recovery setting. For this experiment, we used the training data from the Willow dataset to build our object models, and the training data from the Challenge dataset as the test set. Each frame of the test set contains exactly one object in an unknown pose.

Our detection pipeline achieved a precision of 96.72% at a recall of 97.44% at this task. Out of the 35 household objects, 23 were correctly detected 100% of the time. For the other 12 objects, Table II shows the object-object confusion matrix. The vertical axis shows the true object label, and the horizontal axis shows the label our algorithm reported. Entries along the diagonal represent the percentage of correct classifications; off-diagonal entries represent misclassifications.

C. Multiple Instance Recognition

We also evaluate the performance of our pipeline on the Willow, Challenge, and NIST testing data. Each of the test sets can contain multiple objects in the same frame. We report our precision-recall results in Table III as Willow, Challenge, and NIST respectively. We also report the precision and recall of our older version, which was the winning entry to the Solutions in Perception challenge at ICRA 2011 on the Challenge dataset. Our contest entry did not handle multiple table hypotheses, lacked the final scene consistency check, and did not attempt to recover from undersegmentation.

The Willow test set was significantly more difficult than the Challenge test set. Many test frames contained 6 different objects, creating more instances of partially occluded objects, distractor objects which were not part of the training data, and objects lacking in distinctive texture. The actual dataset used in the challenge was hand-curated to only contain objects with significant texture.

Our current algorithm performed the best on the NIST test set. The presence of distinctive synthetic color and texture on the NIST objects makes it an easier benchmark than the real world Willow objects.

Our current algorithm also performs very well for object recognition for textured household objects, achieving precision and recall rates above 90% for challenging scenes with multiple objects and occlusions.

Figure 7 shows a histogram of our translation and rotation errors for correct detections on the Challenge dataset. Our approach excels at accurately recovering pose, consistently yielding translation and rotation errors of under 5cm and 10 degrees.[13]

VI. Conclusion and Future Work

In this paper we have presented an end-to-end instance recognition approach for textured objects. Our system exploits depth information at training by constructing 3D mesh and feature models, and verifying object hypotheses by matching them against the observed geometry during testing.

[11] For all of our experiments, we used the following parameter settings: $N_{\text{Segment}} = 15, \epsilon_{\text{Plane}} = 0.1, N_{\text{Plane}} = 100, \alpha = 2, N_{\text{SIFT}} = 15, \epsilon_{\text{RANSAC}} = 16, N_{\text{RANSAC}} = 750, \epsilon_{\text{Feature}} = 0.5, \epsilon_{\text{Pose}} = 0.25cm$

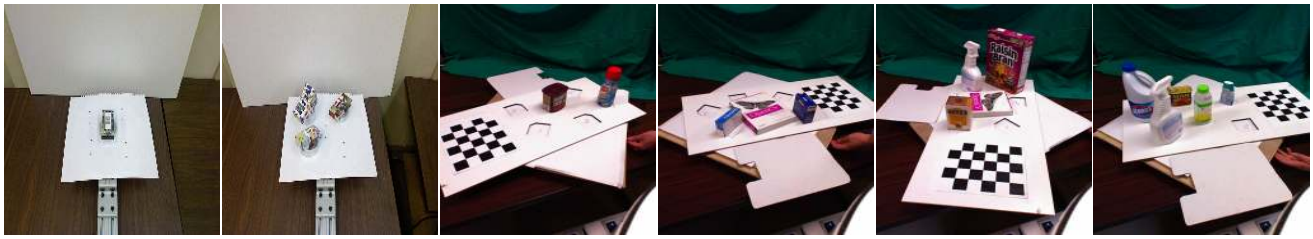
[12] About 95% of the time is spent building the 3D object model (extracting SIFT interest points and descriptors and determining their 3D location).

[13] The evaluation metric used for the ICRA 2011 challenge weighted precision, recall, and pose error together into a single number. See the Solutions in Perception website [4] for more details.

TABLE II

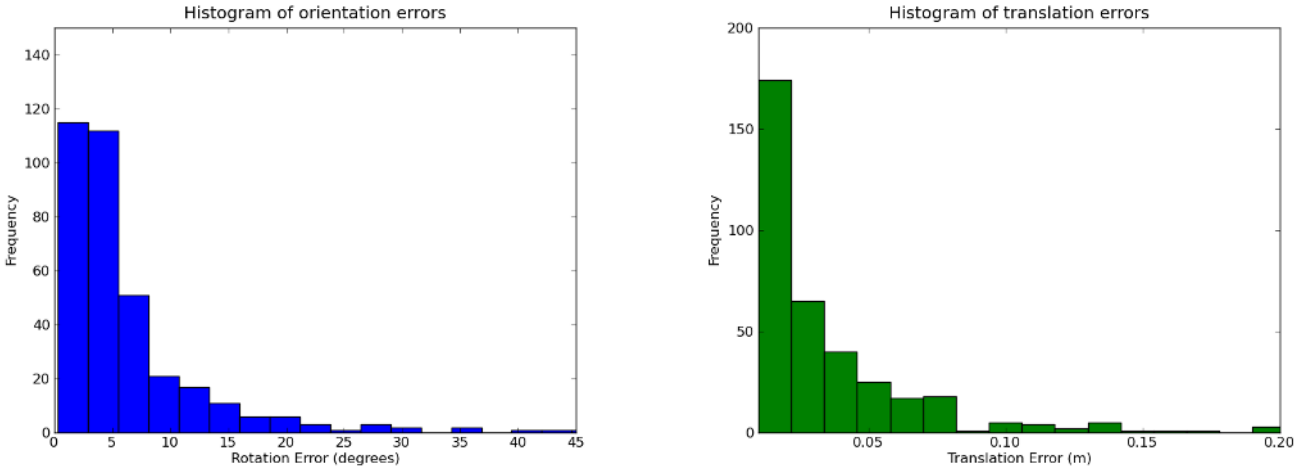
CONFUSION MATRIX FOR THE SINGLE OBJECT INSTANCE RECOGNITION EXPERIMENT. RESULTS ARE REPORTED FOR THE THIRTY-FIVE WILLOW TEST OBJECTS. THE VERTICAL AXIS SHOWS THE TRUE OBJECT LABEL, AND THE HORIZONTAL AXIS SHOWS THE LABEL OUR ALGORITHM REPORTED. MISSING OBJECT LABELS WERE CORRECTLY DETECTED 100% OF THE TIME.

	1	3	4	5	6	7	10	11	13	16	17	20	23	24	25	27	29	30	31	32	33	34	35
1	98.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.4	0	0
3	0	98.6	0	0	1.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	1.4	98.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	98.4	0	0	0	0	0	0	0	0	0	0	0	0	0	1.6
17	0	0	0	2.8	0	0	0	0	0	2.8	87.8	1.9	0	0	0	0	0	1.9	0	0	0	0	2.8
20	0	0	0	0	0	1.7	0	0	0	0	0	93.2	0	0	0	0	0	0	1.7	0	3.4	0	0
24	0	0	0	0	0	1.4	0	0	1.4	0	0	0	0	94.4	0	0	0	1.4	0	0	0	0	1.4
25	0	0	0	0	0	0	0	0	0	0	0	0	2.7	0	97.3	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	1.4	0	0	0	94.3	0	0	0	0	4.3	0
30	0	0	3.3	0	0	0	0	0	3.3	0	0	0	0	0	0	0	0	93.4	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.4	0	0	0	97.2	0	1.4	0
33	1.4	0	0	0	0	0	0	0	0	0	0	0	0.9	0	0	0	0	0	0	0	96.8	0.9	0



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

Fig. 6. Sample test data for the NIST (a), (b) and Willow (c), (d), (e), (f) data sets.



(a) (b)

Fig. 7. Histograms of localization errors on the challenge data set for (a) rotation (b) translation. Results are reported for the contest entry on the Challenge data set.

This paradigm for incorporating depth information allows us to incorporate invariance to 3D transformations into the training procedure, and allows for effective, practical instance recognition. Taking this approach further, we would like to investigate extensions to non-rigid or textureless objects. We would also like to investigate high fidelity 3D rendering approaches to verification.

VII. ACKNOWLEDGMENTS

This work was supported in part by NSF under award IIS-0904672 and by Intel. J.T. was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. We thank Ziang Xie for his contributions.

REFERENCES

[1] G. Griffin, A. Holub, and P. Perona. The Caltech-256. Technical report, California Institute of Technology, 2007.

TABLE III

PRECISION AND RECALL RESULTS FOR THE CURRENT PIPELINE AND THE ICRA 2011 CONTEST ENTRY.

	Precision	Recall
Willow (Current System)	88.75%	64.79%
Challenge (Current System)	98.73%	90.23%
NIST (Current System)	97.24%	97.70%
Challenge (ICRA 2011 Contest)	95.30%	84.10%

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[3] Microsoft Kinect. http://www.xbox.com/en-us/kinect.

[4] Solutions in Perception Instance Recognition Challenge, ICRA 2011. [web page] http://opencv.willowgarage.com/wiki/SolutionsInPerceptionChallenge.

[5] Iryna Gordon and David G. Lowe. What and Where: 3D Object Recognition with Accurate Pose. In Jean Ponce, Martial Hebert, Cordelia Schmid, and Andrew Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2006.

[6] Alvaro Collet, Dmitry Berenson, Siddhartha S. Srinivasa, and Dave Ferguson. Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation. In *IEEE International Conference on Robotics and Automation*, pages 48–55, Kobe, May 2009. IEEE. Best Vision Paper Award Finalist.

[7] Manuel Martinez Torres, Alvaro Collet Romea, and Siddhartha Srinivasa. MOPED: A Scalable and Low Latency Object Recognition and Pose Estimation System. In *Proceedings of ICRA 2010*, May 2010.

[8] A.E. Johnson and M. Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433 –449, may 1999.

[9] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *The IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 05/2009 2009.

[10] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D Recognition and Pose Using the Viewpoint Feature Histograms. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 10/2010 2010.

[11] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *The IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 05/2011 2011.

[12] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. volume 1, pages 886–893, 2005.

[13] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.

[14] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*, volume 256 of *ACM International Conference Proceeding Series*, pages 61–70. Eurographics Association, 2006.

[15] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[16] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[17] W. Randolph Frankin. PNPOLY - Point Inclusion in Polygon Test. [web page] http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html, Dec. 2009.

[18] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[19] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[20] Oren Boiman, Eli Shechtman, and Michal Irani. In Defense of Nearest-Neighbor Based Image Classification. In *CVPR*. IEEE Computer Society, 2008.

[21] Jorge Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. In G. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin / Heidelberg, 1978. 10.1007/BFb0067700.

[22] M.I.A. Lourakis. levmar: Levenberg-Marquardt Nonlinear Least Squares Algorithms in C/C++. [web page] http://www.ics.forth.gr/~lourakis/levmar/, Jul. 2004. [Accessed on 31 Jan. 2005.].