Robot In a Room: Toward Perfect Object Recognition in Closed Environments

Shuran Song   Linguang Zhang   Jianxiong Xiao
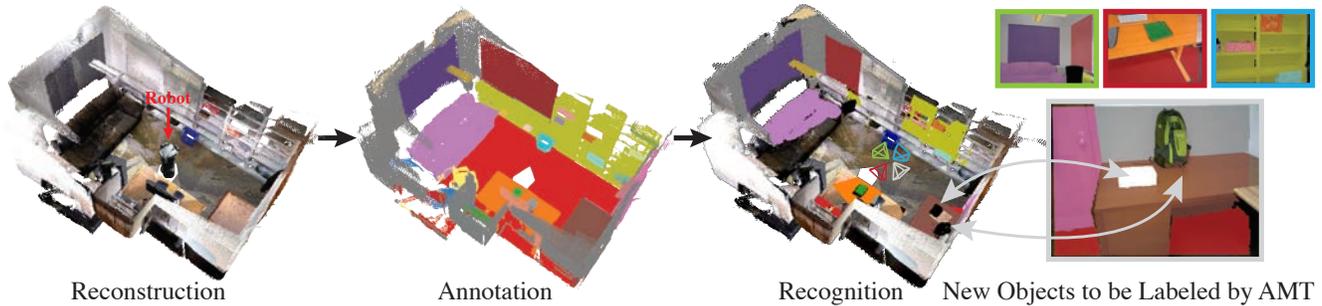Princeton University

Reconstruction   Annotation   Recognition   New Objects to be Labeled by AMT

Figure 1. **A robot that can recognize all the objects**. We propose an extremely robust mechanism to reconstruct a 3D map and use crowd sourcing to collectively annotate all objects. During testing, the robot localizes its pose, recognizes all seen objects (four images on the right from four RGB-D sensors mounted on the robot), and identifies new ones (e.g. the backpack and the box). In most cases, the robot can recognize autonomously. It can indicate reliably when it fails, and utilize crowd sourcing to fix the problem or to annotate new objects.

Abstract

While general object recognition is still far from being solved, this paper proposes a way for a robot to recognize every object at an almost human-level accuracy. Our key observation is that many robots will stay in a relatively closed environment (e.g. a house or an office). By constraining a robot to stay in a limited territory, we can ensure that the robot has seen most objects before and the speed of introducing a new object is slow. Furthermore, we can build a 3D map of the environment to reliably subtract the background to make recognition easier. We propose extremely robust algorithms to obtain a 3D map and enable humans to collectively annotate objects. During testing time, our algorithm can recognize all objects very reliably, and query humans from crowd sourcing platform if confidence is low or new objects are identified. This paper explains design decisions in building such a system, and constructs a benchmark for extensive evaluation. Experiments suggest that making robot vision appear to be working from an end user's perspective is a reachable goal today, as long as the robot stays in a closed environment. By formulating this task, we hope to lay the foundation of a new direction in vision for robotics. Code and data will be available upon acceptance.

1. Introduction

Consider the following grand challenge of computer vision for robotics: prepare a living room for a 10-person party. To enable this task, we need a robot to be able to do planning, grasping, manipulation and reasoning. But before all that, at least the robot needs to recognize the objects, such chairs, sofas, coffee tables, dishes, sodas, etc.

While we have witnessed a dramatically improvement of object recognition in the past few years (e.g. [8, 21, 12]), we are still far from having a vision system for robots to recognize objects reliably in real-world environments. The success of Internet single-image classification and detection cannot live up to its promise to visual perception in robotics. Therefore, most robot visual perception systems today only focus on recognizing a dozen of carefully pre-scanned object instances with clean background (Figure 3).

We aim to bridge the gap to enable robots to recognize realistic objects very reliably in their natural settings. Our **key observation** is that many (service) robots will stay in a relatively closed environment, and this greatly constrains the possible objects that it can encounter. For example, a household service robot will stay in a house and never go outside the house [11]. The set of possible objects that the robot can encountered is finite, and the speed of introducing a new object into a typical house is also limited. A robot doesn't have to recognize all the objects in the world to be fully capable in such a closed environment. This paper proposes to exploit this critical constraint to build a robot vision system that can reliably recognize objects in a realistic environment at an almost human-level accuracy.

Although this paper focuses on robot vision, the same idea also applies to many other movable devices that are
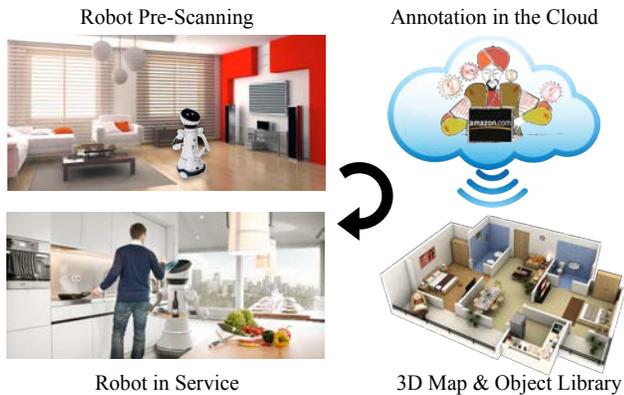
Figure 2. **User scenario.** A robot pre-scanned a house. All data is uploaded to the cloud to be annotated by human workers on crowd sourcing platform, to obtain a object library and a 3D map. After that, the robot can recognize all the objects and provide service.

only used in closed environments, such as head-mounted displays (e.g. Oculus Rift or Microsoft HoloLens) that people typically only use in their own living rooms.

1.1. User scenario

A user buys a robot, brings it home and turns it on. And the journey of its life begins, as illustrated in Figure 2. In the first few hours, the robot does nothing but wanders around the house, scanning the environment by taking videos to reconstruct a 3D map of the place. The robot picks some frames from the video to cover the space, and asks human workers in the cloud to label all the objects in these frames, using online crowd-sourcing platforms (e.g. Amazon Mechanical Turk). For a typically apartment, this costs about $100 and takes about an hour. The robot uses the annotation to build a 3D object library and it is ready for service.

Now the robot can navigate within the apartment and locate itself, relying on the main structure of the rooms that are not movable (e.g. no home remodeling). Given the location, the limit set of possible objects, and the knowledge of their appearance specific to this apartment, the robot can recognize the objects almost perfectly. Of course, the robot will break miserably if it leaves the specific apartment, but it can avoid going outside the known territory. If there is unrecognizable change (e.g. new object), it will reconnect to crowd-sourcing platform again and ask humans to annotate the frame (e.g. with a monthly subscription fee of $10). In such a way, a robot can recognize all the objects in the house perfectly to support its missions.

1.2. Challenges and our solutions

To build such a system, there are several technical challenges that we have to overcome (Figure 1). First, we need an extremely robust way to reconstruct a 3D map for any given environment. In this paper, we propose two coupled mechanisms to make this happen: a panoramic RGB-D camera array and a special robot path to ensure significant

Figure 3. **Current object recognition for robotics** mostly focuses on recognizing objects that are completely inside the field of view (FOV) on a clean background. The objects are from a very limited set with similar sizes suitable for those algorithms. No new objects are allowed, and all objects are completely scanned using a careful setup. We advocate to recognize objects in their natural scenes without change to the environments. (image source: [13, 34])

view redundancy for loop closing. Second, we need a way to enable multiple humans to collectively and efficiently label all objects in a place at the instance level. We propose an algorithm to select frames to maximize coverage, and a robust way to associate instances in 3D. Third, we need a reliable way to recognize objects during testing time. We propose to use the 3D pose of the robot to retrieve the unmovable background structure of the environment (e.g. floor, walls, and ceiling) in order to reliably subtract the background, which significantly reduces the difficulty of recognizing foreground objects. For each of the foreground movable objects, we build a mixture of 3D models merging from multiple views to increase view invariance. Finally, we need a reliable way to fill in holes of old objects when the occlusion situation changes to avoid requesting human annotation too frequently, but at the same time to avoid overly propagate to uncertain areas or new objects. To this end, we propose to integrate object color, normal distribution, shape bounding box, spatial continuity, to eliminate impossible object labels for reliable propagation.

Besides these technical contributions, there are several new concepts as well. First, this project provides a way to bring the success of robotic object recognition from tabletops to real-world scenes. Instead of constraining the background to be very artificial, we recognize objects in realistic environments but we are still able to use background subtraction to ease recognition. Second, unlike Internet images, there is ample domain knowledge we can use to reduce the difficulty of perception in robotics applications. This task puts computer vision algorithms to a real test. By integrating all useful cues, it is a great exercise to let us clearly know how far we are from making robot vision work. Third, our user scenario may potentially incubate a new business model for robot perception, by leveraging automatic recognition and online crowd sourcing. Fourth, our formulation of the user scenario opens up a new research direction in vision for robotics. By constructing a realistic offline benchmark directly related to the goal, we can test algorithms without a robot but reliably reflexing the true performance in real world. Last, we have built a working system with all the components. Experiments shows that

we can already make object recognition working most of time without any human involvement to enable autonomous operation.

1.3. Related works

There is a vast literature on object recognition from 2D, 3D, RGB-D, video in computer vision and robotics. For category-level recognition, the state-of-the-art object detectors are [12, 8, 24], and [36, 17] for RGB-D images. [29, 25, 16, 4, 23] are popular semantic segmentation systems. However, category-level recognition is still far from human performance. For instance-level recognition, well-known approaches include [30, 28]. For RGB-D images, the state-of-the-arts [37, 13] focus on recognizing table-top objects on a clean artificial background, with object models carefully pre-scanned from all view angles [22, 34] (Figure 3). Our approach is built on top of these success, extending them to realistic scenes.

Our 3D mapping is related to RGB-D reconstruction [26, 19, 39, 27, 5, 41] and localization [33, 18]. Our algorithm is closest to the RGB-D Structure from Motion (SfM) from [41]. We extend their algorithm to utilize four RGB-D sensors and encode the camera height as a hard constraint. We design a special trajectory to control the robot to move in a way with significant redundancy to favor loop closing to ensure good 3D reconstructions. Toward semantics, there are several seminal works on combining 3D mapping and object recognition on RGB-D scans [9, 32, 10, 40, 20]. There are also several seminal works in image domain as well [3, 2, 6, 15].

Our object annotation is built on the success of many predecessors [31, 38, 14]. The most related one is the SUN3D annotator for RGB-D videos [41], which uses object annotation to correct reconstruction errors by object-based loop closing. Because our 3D reconstruction is much better, we label individual frames to reduce the labeling difficulty. We also propose a novel 3D instance association algorithm to link object instances in 3D automatically so that multiple humans can collectively label the same sequence.

Overall, perhaps the most relevant work is Google's self-driving car [7]. The great success of the car relies on the pre-annotated maps that contains 3D scan of all the streets that the car can drive to. The maps are precisely annotated by humans for all stationary objects, such as the location of all traffic signs and lights. During driving, based on the car's current location, the data is looked up to serve as virtual infrastructures, and it already gets to know a great deal about the environment. For example, without even looking, the car already knows where a traffic light is, and just needs to classify the color of the light. In such a way, the very difficult street scene understanding task is transformed into a much easier instance-level task, and human-level recognition accuracy is achieved. This paper proposes to build
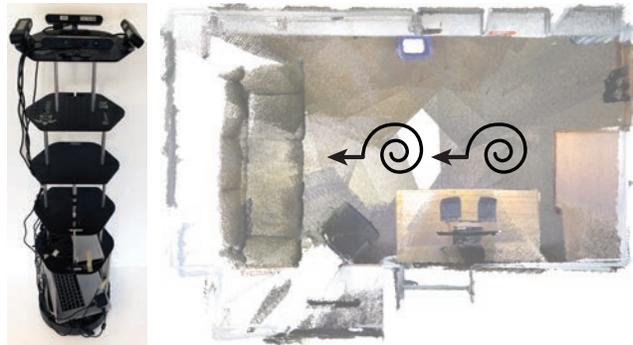

Figure 4. **Robot and path.** On the left we show our robot prototype equipped with four RGB-D sensors at the top close to a typical human height. On the right we illustrate the robot path during training phase, where it rotates two circles at each location and moves 0.3 meter to the next location.

an indoor version of Google's car by utilizing the **power of pre-annotated data**. We show that it is a reachable goal today for a general-purpose robot vision system to recognize every object in a closed environment.

2. Formulation

To make it concrete, we formulate the goal as the following task. There are two phases: training and testing. During **training** phase, RGB-D videos are captured when the robot moves around in the environment that it is supposed to spend its life in. None of the objects is allowed to move during this phase. The 3D map of the environment is reconstructed, and a small number of frames are picked to be labelled by human workers on Amazon Mechanical Turk. During **testing** phase, for each of the RGB-D frames captured by the robot, the task is to estimate the 6D camera pose for the frame, and recognize the object categories and 6D object poses for all the objects presented in the images. Objects may be at different locations between training and testing, and among different testing images. Some objects may disappear and some new objects may appear. The goal is to produce a very high quality recognition result with a reliable confidence indicator, to enable autonomous operation in most cases. The algorithm should also be able to tell whether it is necessary to bother human annotators, if the recognition lacks confidence or when some new object is identified. Note that our focus is on the vision component, and other components of the robot decide its trajectory during testing phase, depend on its missions.

3. Training phase

During training, we desire a robust way to reconstruct the 3D map that the robot will spend its life in. The success of whole system depends on the 3D map, and it is crucial to be able to have a very robust algorithm for any environment. To this end, we propose two major mechanisms to make reconstruction become much easier. First, we mount four RGB-D sensors on top of the robot to have a panoramic
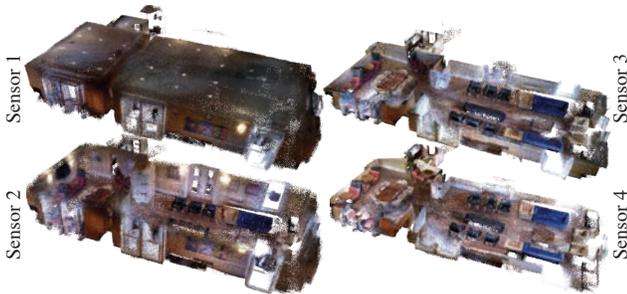
Figure 5. **3D reconstruction from diffferent sensors.** Sensor 1 sees the ceiling, while sensor 4 sees the floor.

without fixed camera height
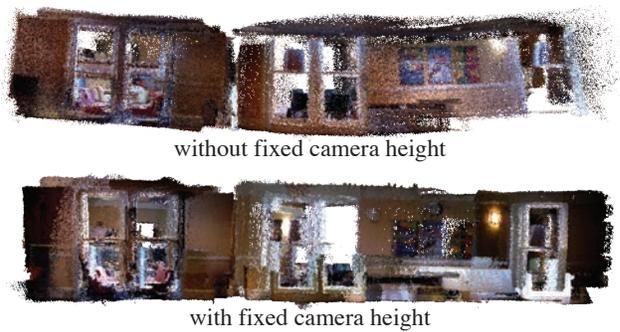

with fixed camera height

Figure 6. **Camera height as hard constraints in bundle adjustment.** We can see that by constraining the camera height, the floor gets flat and the reconstruction gets better.

RGB-D input. For example, if the robot can measure the distance to the four walls of a typical rectangular room, it is much easier to localize the robot to obtain a good 3D camera pose. Because the robot has a fixed height, we can also enforce the camera height as a hard constraint. Second, we let the robot to move in a way that it sees a lot of duplicate views to enable loop closing. When bundle adjustment is overly constrained with many correspondences, 3D reconstruction will have no choice but to converge to the right solution.

3.1. RGB-D camera array

We use an off-the-shelf TurtleBot 2 (Figure 4) and a Macbook Pro. The laptop controls the robot wheels and collects data from four ASUS Xtion Live Pro sensors. We use four sensors to stream four RGB-D videos at 640×480 30fps, reaching the maximal USB bandwidth of the laptop. We mount them on a shelf that is 1.4 meter tall, because most indoor spaces are designed for human heights. All the components of our robot can be purchased off-the-shelf from the Internet, and cost about $2000 (excl. laptop).

We configure the four cameras to have a panoramic field of view. They are arranged at $0°$, $90°$, $180°$ and $270°$ horizontally, and $5°$ up, $10°$ down, $25°$ down and $40°$ down vertically, covering $90°$ vertical field of views. Figure 5 contains point clouds reconstructed from the cameras with different tilt angles. We didn't use a checkerboard to calibrate of the transformation between cameras, because our cameras have no view overlapping at all. Instead, we use the hardware setting to initialize the calibration and adjust it during reconstruction.

3.2. Scanning path

To make object recognition more robust, we desire to densely capture objects from as many different views as the robot can possibly navigate to. Also, to produce a good reconstruction, we control the robot to move in a way that enforces significant view overlapping to ensure ample loop-closing correspondences. As shown in Figure 4, we let the robot move 0.3 meter, stop to rotate two circles, and then move to the next location. Rotating two circles lets every camera see exactly the same view twice. In this way, every frame of the RGB-D videos will have many keypoint correspondences with other frames.

3.3. Panoramic SfM

Similar with the state-of-the-art RGB-D SfM [41], we use a frame-based approach with 3D keypoint-based bundle adjustment. For each RGB-D frame, we detect SIFT keypoints on the color images, and use depth map to obtain the 3D coordinates for the keypoints. For a pair of RGB-D images, we use RANSAC with 3 points to estimate the 6D rotation and translation between the two images and obtain the inlier correspondences. We use this pairwise alignment routine to align every two consecutive frames from the video captured by the same camera to obtain initial camera poses. We use the initial calibration among the cameras to align the 3D reconstructed from the four cameras as the initial poses for bundle adjustment. To obtain more correspondences, we compute bag-of-word for each image, and choose pairs of frames from any camera with high dot product values. For each pair, we use the pairwise alignment routine to obtain inlier correspondences to add into the bundle adjustment.

For the n-th frame, we model the robot base by its (x_n, z_n) location and rotation angle θ_n, with $y_n = 0$ because the robot has a fix height ($\vec{y}$ is the gravity direction). We model the extrinsics of each camera as a 6D rotation $\mathbf{R}_c$ and translation $\mathbf{t}_c$ w.r.t. the robot base. For a 3D point $\mathbf{X}$ from the c-th camera of n-th frame, its world coordinate is $R_{\vec{y}}(\theta_n)(\mathbf{R}_c\mathbf{X}+\mathbf{t}_c)+[x_n, 0, z_n]^T$, where $R_{\vec{y}}(\cdot)$ is a 3×3 rotation matrix for rotating around $\vec{y}$ axis. For bundle adjustment [1], we model these extrinsic parameters $\{x_n, z_n, \theta_n\}$ and the camera calibration $\{\mathbf{R}_c, \mathbf{t}_c\}$ as variables, with the default intrinsic parameters from OpenNI as constants. We did try to keep the camera calibration $\{\mathbf{R}_c, \mathbf{t}_c\}$ as constants during bundle adjustment. But the reconstruction results are always better when we model them as variables.

To speed up reconstruction of long sequences, we break down a sequence into segments of 1,000 frames. We reconstruct each segment independently. Then, we link these segments together by aligning the consecutive frames among them. For each frame in a segment, we look for a frame
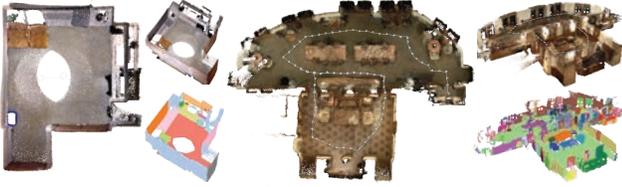
Figure 7. **Reconstruction & annotation.** The kitchen in on the left, and the tea area is on the right. Semantics is encoded as colors.
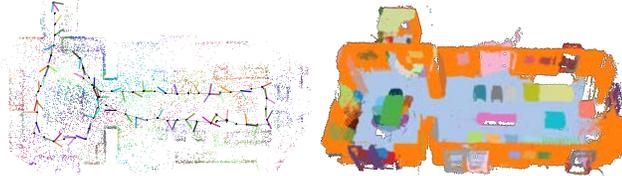

Figure 8. **Annotation.** Left shows the selected images for annotation from the top view. The polyline shows the camera trajectory, and the colored sticks are the camera direction of selected views. The 3D points are the voxel centers colored based by their membership to selected views. Right is the annotated 3D point could, color coded by object instances after automatic association.

with significant view overlapping from another segment, and use the pairwise alignment routine to add more correspondences. Finally, we bundle adjust the whole reconstruction again. Figure 6 compares the reconstruction without and with camera height as a hard constraint. It takes about 2 hours to reconstruct this sequence using 10 CPU cores. We tried various environment and found that our system is extremely robust. Figure 7 shows more results.

3.4. Collectively annotation

It is critical to annotate all the objects in the space at a high quality for reliable recognition. SUN3D [41] provides a way to label RGB-D videos. But because their reconstruction from hand-held captured videos has significant drifting, they can only rely on local stitching. In our case, given the high quality reconstruction, our goal is to minimize human efforts by embracing 3D information as much as possible.

Our algorithm first picks a small number of frames to cover the whole space. Then, we use Amazon Mechanical Turk to annotate all the objects in each frame by polygons and object categories. Finally, our algorithm figures out the association of polygons to the same instance of objects.

View selection. To minimize the human effort, we design an algorithm to choose a minimal number of views to achieve 100% coverage. First, we represent the 3D space by voxel grids of 0.01^3 meter^3 and obtain a list of surface voxels that contains more than five 3D points. We maintain a counter on each voxel to count how many times this voxel has been seen so far. In each iteration of the algorithm, we pick one frame from the remaining ones that maximizes the number of voxels with count transition from $0 \rightarrow 1$. When a frame is picked, we increase the count for each voxel on the depth map of the picked frame. This greedy selection ends when no new frame can be picked to further increase

Figure 9. **Instance association.** Each row shows seven views of the same object automatically associated by our algorithm. The focus objects are highlighted in red.
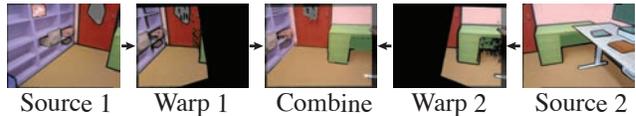

| Source 1 | Warp 1 | Combine | Warp 2 | Source 2 |

Figure 10. **Label propagation to other view.**

the coverage. Figure 8 shows an example of view selection.

Instance association. For the each selected view, we ask human workers on AMT to label all the objects in this frame, using a LabelMe-style interface [31]. We require a instance level segmentation for each frame. But associating instances across many different images is a very challenging and time consuming task, especially when annotators work collectively. Therefore, we propose an automatic algorithm to associate the polygons to objects instances, utilizing our good 3D reconstruction. For each pair of images with overlapping views, we try to establish association between the two sets of polygons. Using their relative pose and depth maps, we warp the label from one image to another, to obtain the ID mapping pair between the two label masks at each pixel in the common overlapping area. We count the frequencies for each matched label pairs if their object names are the same. We use a greedy algorithm to go through all pairs one by one in the descending order of their frequencies, until it cannot find more pairs. For each pair, we accept it only if its frequency is more than 50 pixels, and none of the two polygons has been paired to others so far. In this way, from each of two images, we can get a partial association list. We merge all the lists into one, and use connected component to obtain the final object association for the whole sequence. We evaluate the instance association by the "office" sequence. Figure 9 shows some associated polygons across images. It makes no mistake of merging two objects, but it is slightly over-fragment and cannot associate 10 polygons that it should be able to. Figure 7 and 8 show some point clouds colored with instance association.

Label propagation. Because only a very small portion of frames are labeled by humans, and we will have more invariance if more views of an object are seen, we propagate the label to other frames, making use of our 3D reconstruc-
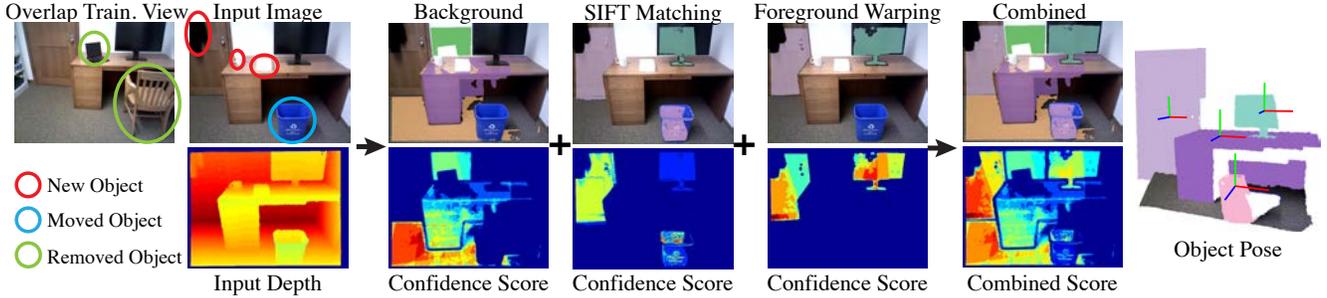
Figure 11. **Recognition pipeline during testing.** Details are explained in Section 4.1 and 4.2.

tion. For an unlabeled image, we warp[1] the depth, color, and label maps of the annotated images to the target image. For each pixel, if the depth and color are consistent, we cast a vote to the label. We accumulate votes from all annotated images and take the majority to obtain a label (Figure 10).

4. Testing phase

During testing, given a frame with four RGB-D images from the sensors, the task is to estimate the pose of the robot, recognize all the previously seen objects, and identify any unseen new objects in these images. Assuming spatial continuity, we also take the robot pose from the previous frame as an input. We classify objects as movable or non-movable[2], by the annotation of object category.

4.1. Localization and background subtraction

As shown in Figure 11, given the initial pose, we can identify all training frames with view overlapping. For each of these training images, we warp the unmovable areas to the testing view using the initial camera pose. We check the depth consistency (with 0.1 meter as the threshold) to identify the unmovable area in the testing frame. If the area is bigger than 50 pixels, we run the SIFT keypoint detection in these areas and use our pairwise alignment routine (Section 3.3) to extract correspondences between the unmovable areas of these two frames. We collect all correspondences from all training frames, and use RANSAC to find a better pose for the current testing frame.

Using this refined pose, to extract background, we repeat the same steps again to warp the unmovable areas of the training frames to the testing frame. For each pixel in this area with consistent depth and color, we accumulate a vote and threshold the vote count to identify the unmovable areas. In this way, we can recognize the unmovable objects as background in the testing frame. The remaining unrecognized areas are the foreground to be recognized next.

4.2. Object model and recognition

To recognize the foreground movable objects, for each object, we build a 3D point cloud model with SIFT descriptors, similar to the state-of-the-art RGB-D instance recognition systems (e.g. [37]). For each object, we maintain

a mixture of clusters, each of which is a 3D point cloud merged from multiple RGB-D images, with a SIFT descriptor attached to each 3D point extracted from the image. We do not directly use camera pose from reconstruction to merge keypoints, because the reconstruction may not be perfectly accurate locally, especially for small objects. Starting from the SIFT keypoints of the object from each frame, we agglomeratively merge two point clouds, if there are enough SIFT inliers from RANSAC. The hierarchical merging is done very conservatively, so that we can obtain accurate 3D shapes with camera poses overfitted to each object. Each object may be represented as a mixture of clusters, instead of only one cluster. During testing time, for the remaining foreground areas, we extract SIFT keypoints with 3D locations, and use RANSAC to match these 3D points with SIFT descriptors in each cluster. If a good matching is found, not only the category and instance label, but also the 6D object pose (Figure 11 right) are recognized.

Although movable objects may move, they may also stay at the same location as it was during training. Therefore, we reuse the same algorithm for background subtraction to warp the movable objects using the refined pose, and check consistency for both depth and color to confirm the guess. We also accumulate a voting count to the object instance label. This step is run in parallel with the object model matching, and the voting count is accumulated together. Finally, we choose the label with maximal count for each pixel.

4.3. Conservative propagation & new object

As shown in Figure 12(b), there may be areas that couldn't be recognized or with wrong boundaries, due to various small imperfection from human annotation or the algorithm. More significantly, there may be holes in the recognition result because there was another object occluding this object in the training set (the yellow circles in Figure 12(a)). Now that the occluder moves away, a new part of the object appears, and we desire to the algorithm to be able to propagate the label to these missing areas. However, this propagation must be very conservative, because we do not want the label to propagate to cover any new object (e.g. the white box in Figure 12(f)). The correct label for new objects should be "no label", so that the system can send it to ask for annotation from crowd sourcing.
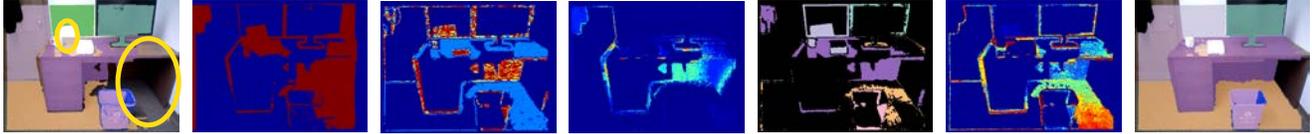
---

[1]We triangulate the mesh and render them in OpenGL.
[2]For simplicity, we assume there is no deformable objects (e.g. human).

| (a) Before propagation | (b) Region need to fill | (c) Normal and color likelihood (desk) | (d) Possibility map | (e) Ordering map | (f) Result |

Figure 12. **Conservative propagation.** Yellow circles in (a) highlight the holes created by the removal of occluders after the training phase.

	train	labeled	obj	test	labeled	area (sq.ft.)
kitchen	432×4	33	29	339×4	18	110
office	637×4	58	40	415×4	24	180
tea area	21,407×4	182	329	6,779×4	40	1,870
meeting place	24,059×4	136	82	7,001×4	40	1,530

Table 1. **Benchmark statistics.** Col 1: number of training frames. Col 2: number of training frames being labeled. Col 3: number of object instance labeled in the training set. Col 4: number of testing frames. Col 5: number of labeled testing frames with ground truth for evaluation. Col 6: the area of the space.

To handle wrong labels near the boundaries, we first shrink the recognition mask by image erosion (with a 5-pixel disk as kernel), and then propagate the labels to better align with the image and depth boundary.[3] We treat the shrunk labels as hard constraints during label propagation.

For the propagation, we first identify a list of possible object instance labels for each unrecognized pixel (Figure 12(d)), by considering color, normal, and 3D bounding box of an object. For each object, we can obtain their color and surface normal distributions, and use these two distributions to estimate the likelihood of each pixel belongs to this object (Figure 12(c)).[4] We also obtain a 3D bounding box for each object using the labeled 3D point cloud from the training set. In testing, we position the bounding box using the estimated object pose, and check whether this pixel is outside the box and therefore should not belong to this object.

A good propagation should have a smooth transition of color, normal and depth. And it should be able to stop if it is a new object. We also desire to enforce the spatial continuity and want the propagation to be connected to the hard constrained label mask. To this end, we maintain a max-heap of all unlabeled pixels adjacent to the areas with labels. In each iteration, we will choose one pixel from the heap that has the most similar color, normal, and depth with one of its neighboring labeled pixel to propagate the label from this neighbor, without violating the label possibility constraints estimated above. When a pixel is chosen to propagate the label, its unlabeled neighbors will be put into the heap, or their key values will be updated (i.e. increase-key) by the similarity score with this newly labeled pixel. The iterations stop when the maximal similarity score is below

a set threshold. Figure 12(e) shows the pixel ordering of the propagation. In such a way, we make use of color, normal, depth, size, continuity to propagate conservatively, and stop when new object(s) are introduced (Figure 12(f)).

After propagation, if there are still a large unrecognizable area, the image will be sent to crowd sourcing platform to request for annotation. Otherwise, the recognition is considered successful autonomously. In both cases, the label mask is integrated to the object model. The whole pipeline for testing takes about 10 mins per frame for a typical scene.

5. Experiments

For this new task, we construct a benchmark to evaluate the algorithm carefully. The benchmark also enables offline comparison without a robot, and eases follow-up research.

5.1. Benchmark evaluation

We construct a benchmark of four places: an office, a kitchen, a tea area, and a meeting place. Table 1 gives some basic statistics. The kitchen is a single room in an office building. The office is a professor's office. The tea area contains two connected rooms: a tea room for social events with a small kitchen, and a lounge with sofas and a coffee table. The meeting place contains three connected rooms: a meeting room with a comfortable setting for discussion groups of 20-25 people, a large living room to provide a cozy space, and a kitchen. For the training phase, the robot exhaustively scans the places using rotate-and-move scheme. For the testing phase, we move some old objects naturally and introduce some new objects. For labeling, we experiment with AMTurk and we can obtain good results with quality control mechanisms from [35]. But to construct a high quality benchmark, we label all data in house.

We evaluate the recognition result by semantic segmentation accuracy. We use recall to measure the ratio of pixels with correct labels to the total number of pixels. We only evaluate on the set of pixels $\mathcal{I}$ with valid depth and annotation. We use precision to measure the percentage of correct pixels among all the predicted labels. In our case, we desire an almost perfect precision with a high recall, so that robot can avoid requesting humans to annotate too frequently. When a new object appears, we desire the algorithm to predict "no label" at the region, and prediction of any label will be considered as wrong. For the i-th pixel, let t_i be its ground truth label, and l_i be the predicted label ($l_i = \emptyset$ when it predicts no label). Let $\mathcal{N}$ denote the set of new objects. Therefore, the correctness of a predicted label is $\delta_i = (t_i \notin$

---

[3]During erosion, thin objects might completely disappear and not be able to recover during propagation. So we use image opening to identify these thin objects and only shrink the labels for the thick objects.

[4]For a given pixel, we compute the likelihood by counting the number of pixels within the object mask with similar color or normal directions. The count is thresholded into binary decision to indicate whether this pixel can be possibly labeled as the object.

| Image | Depth | NN pose | [29] pre-train | [29] re-train | background | before prop. | after prop. | with human |

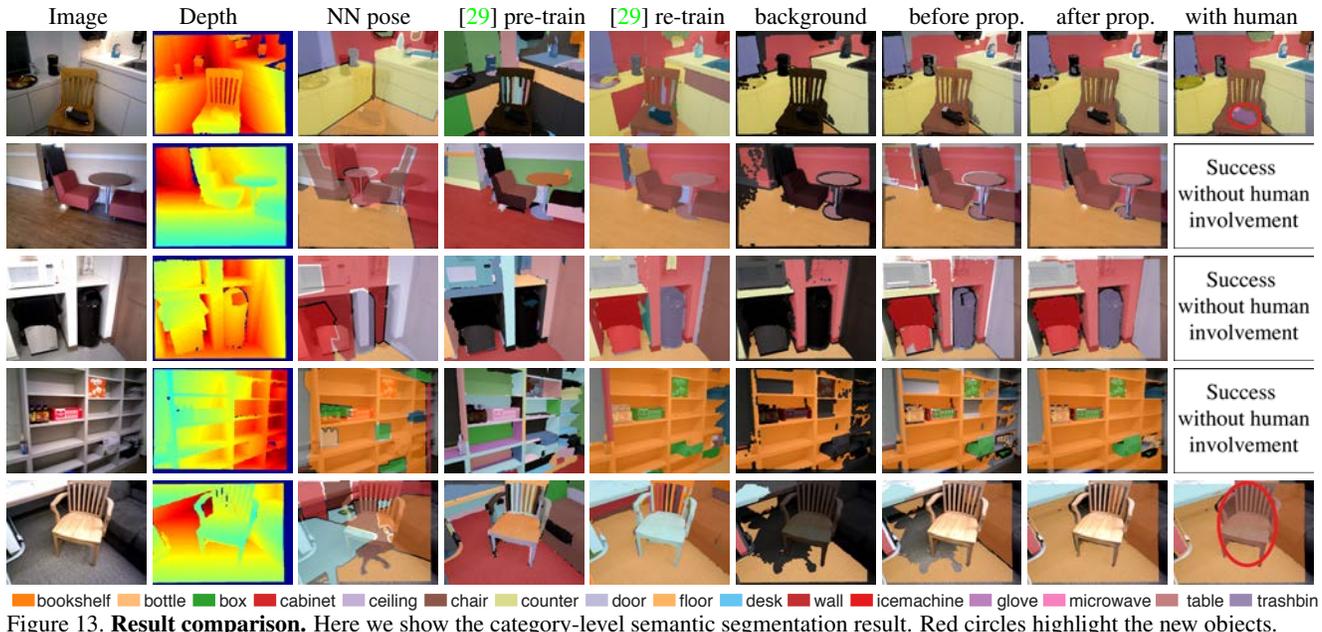| bookshelf | bottle | box | cabinet | ceiling | chair | counter | door | floor | desk | wall | icemachine | glove | microwave | table | trashbin |

Figure 13. Result comparison. Here we show the category-level semantic segmentation result. Red circles highlight the new objects.

		[29] pre-train	[29] re-train	NN gist	NN pose	bg+fgwarp	bg+SIFT	SIFT	bg+fgwarp+SIFT	+ propagate	+human	no new
instance	recall	-	84.45	62.12	63.95	70.15	60.72	67.48	74.01	86.58	91.01	86.59
	precision	-	84.45	66.67	68.56	97.61	97.97	96.82	97.44	96.03	96.24	96.23
category	recall	6.67	86.24	62.12	63.95	70.15	60.72	67.48	74.01	86.58	91.01	87.36
	precision	6.67	86.24	67.96	69.85	98.5	98.45	97.95	98.35	96.89	97.13	97.08

Table 2. **Performance evaluation.**

$\mathcal{N} \wedge l_i = t_i) \vee (t_i \in \mathcal{N} \wedge l_i = \emptyset)$. And the recall is $\frac{\sum_{i \in \mathcal{I}} \delta_i}{|\mathcal{I}|}$, and the precision is $\frac{\sum_{i \in \mathcal{I}} \delta_i}{\sum_{i \in \mathcal{I}} (t_i \notin \mathcal{N} \wedge l_i \neq \emptyset) \vee (t_i \in \mathcal{N} \wedge l_i = \emptyset)}$.

5.2. Comparisons

To see the benefits from constraining a robot to stay in a closed environment, in Table 2, we compare the performances of an state-of-the-art algorithm [29] pre-trained on SUN-RGBD [35] vs. re-trained on our training set. The bad performance of the pre-trained model shows that general object recognition is very difficult. And the huge performance gap between them suggests that constraining in a closed environment makes the problem much easier. Different from our method, this approach mostly relies on appearance, without any other cues such as object movability, environment map and camera pose. Note that this approach [29] has the same recall and precision, because it makes prediction on every pixel regardless of its confidence, which is undesirable in our scenario. We also design another set of baselines using Nearest Neighbor (NN), based on appearance or camera pose. For appearance-based NN, we extract GIST features on RGB-D image [35] to find the NN. For camera-pose-based NN, we use the initial pose to find the training frame that has the largest overlapping view. In both cases, we directly copy the NN's label map as the result. Table 2 and Figure 13 shows the comparison with our algorithm. By using all domain cues, our algorithm significantly outperforms all these baselines.

To evaluate the importance of each component in our algorithm, we decompose our pipeline and evaluate them separately. "bg+fgwarp" assumes none of the objects moves and it simply warps the label from training frames using refined camera poses. "bg+SIFT" uses warping to explain the non-movable objects and uses SIFT to match all the remaining regions. "SIFT" assumes that all objects are movable and directly matches them with object models using SIFT. "bg+fgwarp+SIFT" combines all above the components. "+ propagate" is our full pipeline that also propagates the label to increase prediction coverage. The results in Table 2 suggest that every component in our pipeline is very reliable with a very high precision. Our final result combines different components to increase the coverage of the prediction, in order to achieve a better recall. "+human" shows what happens if the algorithm decides to ask a human annotator to fix the result using crowdsourcing. We can see that human performance is not 100%, which is caused by the imperfection of annotation on boundaries. Last but not least, "no new" shows the performance when there is no new object (we exclude the new object area in evaluation). From here, we can see that the automatic algorithm performs almost as good as humans. This suggests that when there is no new object, the level of accuracy is high enough for fully autonomous operation without any human involvement.

6. Conclusion

We observe that most robots will stay in a closed environment (e.g. a house). Based on this, we propose a way to enable a robot to recognize all objects at an almost perfect accuracy, leveraging 3D maps and crowd sourcing. By formulating and constructing a benchmark, we hope to lay the foundation of a new direction in vision for robotics.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. 4

[2] S. Y. Bao, M. Bagra, Y.-W. Chao, and S. Savarese. Semantic structure from motion with points, regions, and objects. In *CVPR*, 2012. 3

[3] S. Y. Bao and S. Savarese. Semantic structure from motion. In *CVPR*, 2011. 3

[4] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, 2013. 3

[5] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 3

[6] W. Choi, Y. W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *CVPR*, 2013. 3

[7] G. Erico. How google's self-driving car works. *IEEE Spectrum*, 2013. 3

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010. 1, 3

[9] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard. Toward lifelong object segmentation from change detection in dense rgb-d maps. In *ECMR*, 2013. 3

[10] N. Fioraio, G. Cerri, and L. Di Stefano. Towards semantic kinectfusion. In *ICIAP*. 2013. 3

[11] B. Gates. A robot in every home. *Scientific American*, 2007. 1

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 3

[13] J. Glover and S. Popovic. Bingham procrustean alignment for object detection in clutter. In *IROS*, 2013. 2, 3

[14] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 3

[15] A. Gupta, M. Hebert, T. Kanade, and D. M. Blei. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010. 3

[16] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*. 2013. 3

[17] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*. 2014. 3

[18] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. Multi-output learning for camera relocalization. In *CVPR*, 2014. 3

[19] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *ISER*, 2010. 3

[20] E. Herbst, P. Henry, and D. Fox. Toward online 3-d object segmentation and mapping. *ICRA*, 2014. 3

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[22] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011. 3

[23] K. Lai, L. Bo, X. Ren, and D. Fox. Detection-based object labeling in 3d scenes. In *ICRA*, 2012. 3

[24] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 3

[25] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 3

[26] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 3

[27] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *TOG*, 2013. 3

[28] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007. 3

[29] X. Ren, L. Bo, and D. Fox. RGB-D scene labeling: Features and algorithms. In *CVPR*, 2012. 3, 8

[30] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *IJCV*, 2006. 3

[31] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2008. 3, 5

[32] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013. 3

[33] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013. 3

[34] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. BigBIRD: A large-scale 3d database of object instances. In *ICRA*, 2014. 2, 3

[35] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 7, 8

[36] S. Song and J. Xiao. Sliding Shapes for 3D object detection in depth images. In *ECCV*, 2014. 3

[37] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data. In *ICRA*, 2012. 3, 6

[38] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2013. 3

[39] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. 2012. 3

[40] T. Whelan, M. Kaess, R. Finman, M. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald. 3d mapping, localisation and object retrieval using low cost robotic platforms: A robotic search engine for the real-world. 3

[41] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013. 3, 4, 5