Noname manuscript No.
(will be inserted by the editor)

Multi-sensor 3D Object Dataset for Object Recognition with Full Pose Estimation

Alberto Garcia-Garcia · Sergio Orts-Escolano · Sergiu Oprea · Jose Garcia-Rodriguez · Jorge Azorin-Lopez · Marcelo Saval-Calvo · Miguel Cazorla

Received: date / Accepted: date

Abstract In this work, we propose a new dataset for 3D object recognition using the new high-resolution Kinect V2 sensor and some other popular low cost devices like PrimeSense Carmine. Since most already existing datasets for 3D object recognition lack some features such as 3D pose information about objects in the scene, per-pixel segmentation or level of occlusion, we propose a new one combining all this information in a single dataset that can be used to validate existing and new 3D object recognition algorithms. Moreover, with the advent of the new Kinect V2 sensor we are able to provide high-resolution data for RGB and depth information using a single sensor, whereas other datasets had to combine multiple sensors. In addition, we will also provide semi-automatic segmentation and semantic labels about the different parts of the objects so that the dataset could be used for testing robot grasping and scene labeling systems as well as for object recognition.

Keywords 3D computer vision · Object recognition · 3D object dataset · Kinect v2 · PrimeSense Carmine

Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Jose Garcia-Rodriguez, Jorge Azorin-Lopez, and Marcelo Saval-Calvo are with the Department of Computer Technology of the University of Alicante (email: {agarcia, sorts, soprea, jgarcia, jazorin, msaval}@dtic.ua.es).

Miguel Cazorla is with the Computer Science and Artificial Intelligence Department of the University of Alicante (email: miguel@dccia.ua.es)

Alberto Garcia-Garcia
Computer Technology Department
University of Alicante
E-mail: agarcia@dtic.ua.es

1 Introduction

Object recognition is defined as the task of identifying objects in an image. Nowadays, 3D object recognition remains a relatively unsolved problem at both category and instance levels. Usually, detecting and recognizing an object implies finding its position in the scene and also categorizing the object itself or directly giving the specific name of the object. By the position of the object we can consider only an specific coordinate in the three-dimensional space or we can also specify the 3D pose of the object in the scene. This information is becoming crucial with the advent of robotics systems for grasping real objects in cluttered environments.

During last years we have witnessed many steps forward, mainly thanks to the advent of low cost RGB-D sensors such as the Microsoft Kinect device. Besides that fact, the limits of computer vision have been pushed forward largely due to the creation of large and high quality datasets which have allowed researchers to develop and test algorithms in an efficient manner. However, there is not a clear dataset of choice for the community. Although there exist several 3D vision datasets, most of them lack some important features which are present in other datasets and vice versa. We believe that the main hurdle which is slowing the progress towards solving the 3D object recognition problem is the lack of a complete and unified dataset which is able to offer researchers all the resources they might need to develop, train, improve and test their system. In this sense, providing a single and complete solution which encompasses all the strengths and none of the weaknesses of current state of the art datasets would allow researchers to create better systems.

The main goal of this proposal is to provide users with the most complete dataset to perform 3D object

recognition, detection, 3D pose estimation and reconstruction using a single set of information. As an additional contribution, the applications developed for capturing the objects and for manual annotation will be released as an open-source package for creating custom datasets.

This paper is structured as follows. Section 2 reviews the most used datasets to pinpoint their strengths and weaknesses. Section 3 overviews the capture system setup for our proposal. Section 4 describes the components of our dataset. Sections 5 and 6 explain how the training set and the validation scenes were captured and generated. At last, Section 7 draws some conclusions about this proposal and outlines future works.

2 Related works

Before proposing this novel dataset we carefully analyzed existing ones in order to find their weak and strong aspects. The following datasets were selected for the comparison: the Washington RBGD Object Dataset V1 and V2 [1], the Object Segmentation Database [2], the Willow Garage Dataset [3], a custom dataset by Ajmal Mian *et al.* [4, 5], the ECCV2012 Dataset [6], the Big BIRD Dataset [7], the ACCV2012 Database [8] and the Bologna Descriptor Matching datasets (1, 2, 3, and 5)[9, 10, 11]. These datasets were chosen following popularity, freshness and completeness criteria.

In this analysis we have taken into account various features: the number of objects, color, per pixel labeling, bounding box labeling, category labeling, segmentation masks, 6DoF information, registered mesh, level of occlusion, 360-degree registered cloud, the number of evaluation scenes, the capture device and the file formats provided. Table 1 shows a summarized view of this comparison.

In addition, we will briefly review a subset of three of the most remarkable datasets in terms of the aforementioned features: the Washington RGB-D Object dataset V2, the BigBIRD and Willow Garage datasets. This will help us highlight the most important and requested features of those datasets to justify their inclusion in our proposal.

On the one hand, the Washington dataset is great for systems whose goals are object instance and category recognition in a hierarchical manner. Its main advantages are the huge number of objects captured from multiple viewpoints and the hierarchical organization they used to label them with categories and instances. In addition, the dataset includes a set of evaluation scenes in form of video sequences of cluttered and occluded indoor environments. They also provide a novel and effective method for reducing the complexity of the video sequence labeling task. However, the dataset lacks some features which might be quite useful for certain algorithms such as 6DoF pose information of the objects in the scenes, mesh reconstruction for each object of the dataset or fully registered point clouds of the individual objects.

On the other hand, the BigBIRD database is an excellent dataset for training a system whose focus is the object instance recognition problem, its main strengths are a considerable number of objects with high quality images taken from a large number of viewpoints, together with reconstructed meshes and pose and calibration information. It makes use of a high-quality 3D scanning system and provides a lot of software components for the data collection process. However, the main drawback of the dataset is that it contains no evaluation scenes so it is not suitable for testing object recognition systems in scenes under cluttered conditions or presence of occlusions.

At last, the Willow Garage dataset stands out of the crowd due to its large number of evaluation scenes, however it lacks many of the aforementioned mentioned features. The rest of the datasets are not particularly remarkable at any point except the custom one by Mian *et al.* which was captured with a Minolta Vivid 910 and provides high-quality fully reconstructed meshes and registered clouds with ground truth pose.

Taking all of this into account, our contribution focuses on providing a single dataset which takes the better features of all of them, providing a reasonable number of objects and evaluation scenes with various file formats and also using multiple sensors like the PrimeSense Carmine and the recently introduced Kinect V2.

3 Capture system overview

The capture system consists of a Microsoft Kinect v2 sensor device mounted on a tripod which is 1.3 meters tall and positioned at a fixed distance of 0.9 meters to the turntable center with an inclination angle of approximately 30 degrees with respect to the turntable plane. Additionally, a PrimeSense sensor was mounted on the same tripod; the PrimeSense Carmine was positioned at the same height but closer to the platform (approximately 0.7 meters away from the turntable center).

Besides the sensor device, the main component of the capture system is the rotating platform or turntable **(see Figure 2)** and its control unit. The movement of the turntable is controlled by an Arduino Uno device which receives commands over a serial port, including orders for rotating the platform step by step, changing

Fig. 1 All the objects of the dataset placed on the turntable at the starting position. The color images correspond to the ones captured by the Kinect V2 device. From left to right and top to bottom: cocacola_paper_glass, cola_cao_container, air_freshener, plastic_plate, plastic_knife, coffee_cup1, coffee_cup2_edinburgh, danone_yogurt, wireless_telephone, telephone, medicine1_supradyne, milk, tea_package_camomile, medicine2_enalapril, medicine3_almax, medicine4_dacortin, medicine5_metformina, medicine6_drsos, honey_pot, chocolate_syrup, blue_vase, plant, saccharine, big_dice, tasmanian_figure, banana, hand_cream and toilet_paper.

Table 1 Comparison of current state of the art datasets and definition of our dataset proposal.

Name	No. Objects	Color	Per pixel labelling	Bounding Box Labelling	Category labelling	Segmentation masks	6DoF pose information	Registered mesh	Level of oclusion	360 Registered cloud	No. Evaluation scenes	Device	Format
Washington RGBD Object dataset V1 [1]	300	Yes	No	Yes	Yes	Yes	No	No	No	No	8	Kinect	RGBD + PCD
Washington RGBD Object dataset V2 [1]	300	Yes	Yes	No	Yes	Yes	No	No	No	No	14	Kinect	RGBD + PCD
Object Segmentation Database [2]	N/A	Yes	Yes	No	Yes	No	No	No	No	No	111	Kinect	RGB + PCD
Willow Garage Dataset [3]	N/A	Yes	Yes	No	No	Yes	No	No	No	No	176	Kinect	RGB + PCD
Ajmal Mian et al. [4, 5]	5	No	No	No	No	No	Yes	Yes	No	Yes	50	Minolta Vivid 910	PLY
ECCV2012 [6]	35	No	No	No	No	No	Yes	Yes	No	Yes	50	Kinect	PCD, PLY
BigBird [7]	125	Yes	Yes	Yes	No	Yes	Training	Yes	No	Yes	0	Carmine 1.09 sensor + HD-RGB	PCD, RGB
ACCV 2012 [8]	15	Yes	No	No	No	No	Yes	Yes	No	Yes	18000	Kinect	RGB + OWN DEPTH
Bologna Descriptor Matching (1 & 2) [9, 10, 11]	6	Yes	No	No	No	No	Yes	No	No	No	45	Spacetime Stereo	PLY
Bologna Descriptor Matching (3) [9, 10, 11]	8	Yes	No	No	No	No	Yes	No	No	No	15	Spacetime Stereo	PLY
Bologna Descriptor Matching (5) [9, 10, 11]	6	Yes	No	No	No	No	Yes	No	No	No	16	Kinect	PLY
Our dataset	30	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	15	Kinect V2, Carmine	PCD + PLY + RGBD

the step size and time, returning the platform to a home point and calibrating the rotation.

Fig. 2 Our turntable with an object placed near its center, note the blue color of the platform to perform chroma keying operations.

4 Dataset composition

The dataset is composed by two different parts: the training set and the validation scenes. The training set consists of acquired 360-degrees views of each object using the previously described capture system. Validation scenes were acquired in household environments, considering different levels of occlusion and a great various common objects.

For this initial version of the dataset [1] we have included 28 objects in the training set **(see Figure 1)** and 9 evaluation scenes. We intend to increase the number of both training objects and validation scenes if the dataset gains popularity.

[1] `http://www.dtic.ua.es/~agarcia/dataset`

In order to reach most people from the research community, we will provide data in different image and point cloud formats. Subsections 4.1 and 4.2 describe the information provided with the dataset in a detailed manner.

4.1 Training set

Each object of the training set was scanned using the previously described capture system and performing 64 captures at different and equally spaced points of view (360 degrees turn with a full capture each 5.625 degrees). This process was performed using the three aforementioned sensor devices. All the information provided by the capture system was later post-processed to generate the following data for each object:

- 64 Color images in PNG format.
- 64 Depth images in PNG format.
- 64 Color structured point clouds in PCD format.
- 64 Color point clouds in PLY format.
- 64 Segmentation masks in PBM format.
- 360-registered point cloud in PCD and PLY format.
- Reconstructed object mesh in PLY format.
- Category and instance labels.
- Semantic component labels.
- Acquisition settings in a TXT file.

The 64 raw colored PCD point clouds for each one of the points of view are provided in an organized format in order to allow users to take advantage of the matrix organization to accelerate algorithms that perform computations over this data.

The whole process followed for generating this set of data using the information provided by the capture system is described in **Section 5.**

4.2 Validation scenes

A set of 9 validation scenes has been included in our dataset. The validation scenes were acquired in household environments, where the distribution of the objects is messy. Therefore, occlusions may occur between the different objects. This fact will be a challenge in order to properly recognize each one of the objects in the scene. For each validation scene we will provide similar data as the one we provided for the training set (color and depth maps, structured PCD and PLY but obviously no 360-registered cloud nor reconstructed mesh). Nevertheless, we will also provide labeled color images of the scenes where each object is identified by a label. This information will be reflected in a TXT file where image pixels will be labeled. We will describe the labeling process in Section 6.

5 Training set generation

In the following section we will describe all the steps that have been taken to generate the training set, from the information capture to the final object reconstruction including the registration of all the point clouds. **For the sake of simplicity, we will describe the process using only the Kinect V2 sensor.**

5.1 Information capture

The first step for the dataset generation is the information capture in which the data streams provided by the sensors are processed to create pieces of useful information. In our case, each object is captured by the camera 64 times in a full 360 degree turn of the platform. For each capture, the direct processing of the Kinect V2 data streams provides us three different information sources: a 1920x1080 RGB color image (Figure 3), a 512x424 depth map, and a 512x424 infrared image (Figure 4). Those three sources are converted to RGB and grayscale PNG images respectively as a part of the information of the dataset.

By using the color information and the depth map, a colored point cloud is generated by projecting the color on the depth data, **as shown in Figure 5**. The object is then segmented in that point cloud by using a chroma key operation to remove the platform and a set of depth thresholds and a bounding box to eliminate non-keyed points. Our dataset generator application allows the user to fully customize the parameters of the previously mentioned processes in order to obtain a perfect manual segmentation.

Fig. 3 RGB color image (1920x1080) in PNG format outputted by our application using the information captured by the Kinect v2 device.

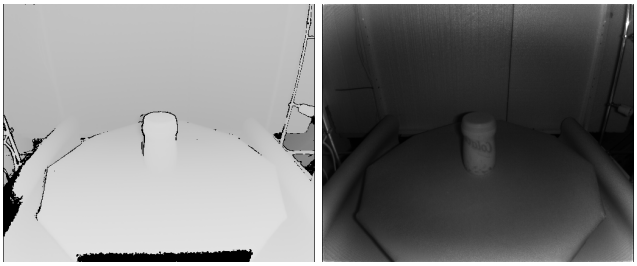

Fig. 4 Depth map in preview mode with inverted color representation (left) and infrared image (right) both 512x424 PNG images outputted by our application using the data provided by the Kinect v2.

The points which are segmented out are not actually removed from the point cloud, instead they are expressed as *NaN* so that the cloud is kept in an organized format. This way, the algorithms that might be applied to those clouds can exploit the matrix organization to accelerate the processing of certain operations. As a result, the colored point cloud is exported as PLY and PCD files and a segmentation mask with the removed points in PBM format is also generated.

This process was repeated with the objects and the aforementioned setup using the Primesense Carmine 1.09 sensor.

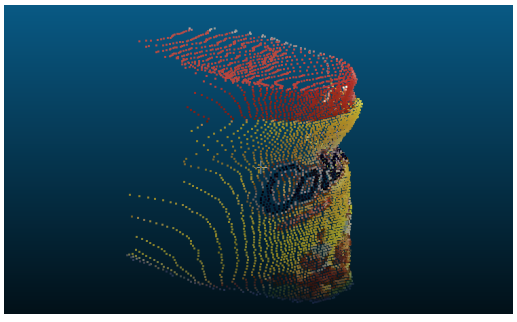

Fig. 5 Point Cloud of a capture generated by our application by combining the color information (Figure 3) and the depth map (Figure 4) and with object segmentation applied.

5.2 Cloud registration

The cloud registration stage refers to the process of aligning the 64 captured point clouds of the object from different points of view integrating all of them into a single 360-degre cloud.

It is important to note that the depth maps provided by the sensor devices contain measurement errors in form of noisy data and outliers which could eventually produce fails when applying certain 3D data processing methods such as normal estimation or point cloud registration. In particular, this noise is very characteristic in the captures performed with the Kinect V2 device and it often consists of artifacts generated at the borders of the objects in form of trails. This kind of noise is due to the time of flight technology used by the Kinect V2 sensor. We show some examples of this noise in Figure 6.

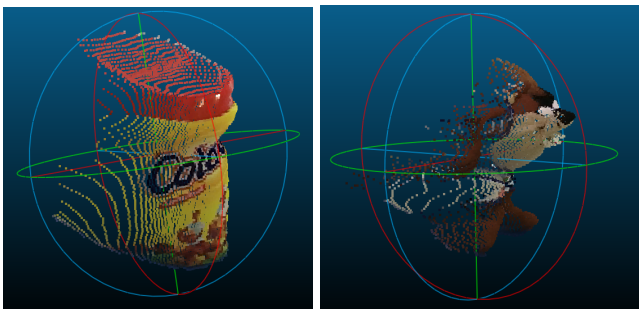

Fig. 6 Noisy point clouds of two objects of our dataset generated by combining the depth and color maps provided by the Kinect v2 sensor.

The first step to perform before registering consists of reducing the noise levels to increase the performance and quality of the alignment methods. In order to do that we have applied multiple filters implemented in the Point Cloud Library (PCL) [12]. The most effective one is the Statistical Outlier Removal (SOR) filter which is able to deal with the trails generated by the Kinect v2 device (see Figure 6) as we can observe in Figure 7.

As we can see, the Statistical Outlier Removal filter does a nice job removing the trails but it tends to generate clusters due to the removal of sparse points which might not be noise. The Radial Outlier Removal (ROR) filter was also applied to the objects but its results were quite similar to the ones obtained by using the SOR one.

Assuming that those clusters are noise, we can remove them by applying a Euclidean Cluster Extraction (ECE) operation. We assume too that small clusters tend to be isolated noise information so we keep the k largest ones as valid information using the previously

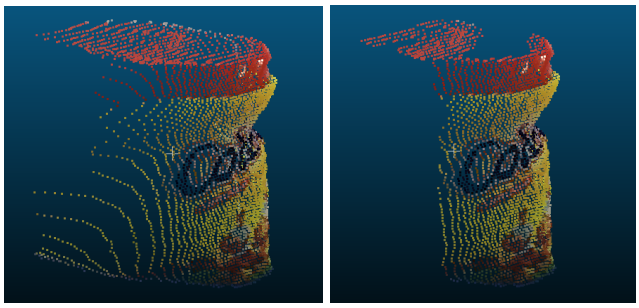

Fig. 7 Noisy point cloud of an object of the dataset (left) and Statistical Outlier Removal filter applied to that point cloud (right) using 50 points for the mean distance estimation and a standard deviation multiplier threshold of 0.9. Note that the trail is removed but two clusters of points are formed.

mentioned filter. Figure 8 shows an example of application of the ECE filter to the SOR filtered point cloud shown in Figure 7.

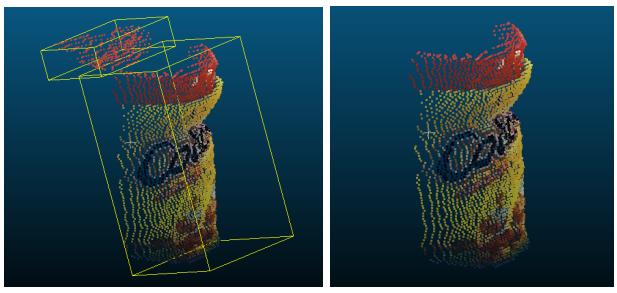

Fig. 8 SOR filtered cloud of an object of the dataset (left) with two clusters identified with yellow boxes and the same cloud after an ECE operation keeping the largest cluster (right).

However, this combination of SOR and ECE is not perfect and fails when the aforementioned assumptions are not true. Even when multiple clusters are generated, not only the largest has to be the only valid one so determining a proper value for the k parameter turns out to be critical.

As an alternative for the statistical and radial approaches, we designed a simple point removal operator named Cut filter. This filter determines the maximum and minimum Z values (Z_{max} and Z_{min} respectively) of the target point cloud and then removes all the points whose Z value exceeds a certain limit set within the range $[Z_{min}, Z_{max}]$. This limit is expressed as a percentage $[0, 100]$ so that applying the Cut filter with a cut amount of 50% implies removing all the points whose Z value is out of the range $[Z_{min}, (Z_{min}+(Z_{max}-Z_{min})/4)]$.

This custom filter behaves nicely if the object has a simple a convex shape as we can see in Figure 9. However, it fails when dealing with complex and concave

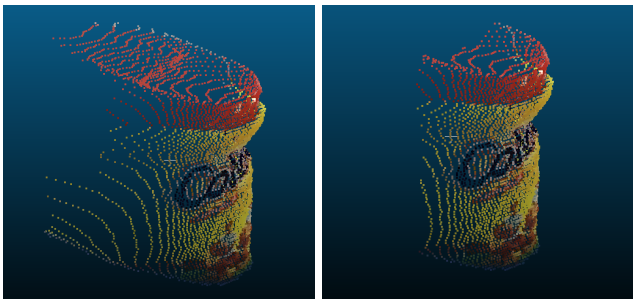

Fig. 9 Noisy point cloud of an object of the dataset (left) and Cut filter applied to that point cloud (right) using an amount of 50%. Note that half part of the cloud which contains the noise is removed.

objects with multiple shapes as we can observe in Figure 10. In this sense, a combination of the three filters has been applied to remove the noise from the clouds of the dataset so that we could achieve the best result for each particular object depending on its shape.

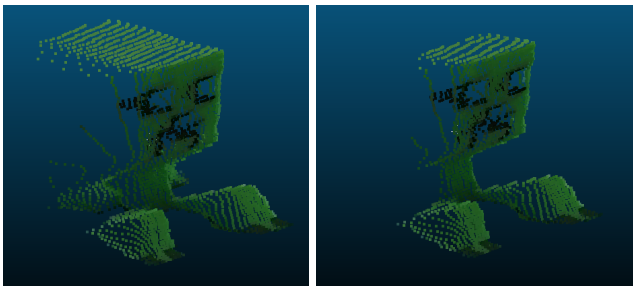

Fig. 10 Noisy point cloud of an object of the dataset (left) and Cut filter applied to that point cloud (right) using an amount of 50%. Note that half part of the cloud which contains the noise is removed but also some part of correct information.

Once all the noise filtering techniques have been applied, the next step is providing a coarse estimation for the alignment. In this case, we can provide a good approximation by rotating the cloud the same degrees that the platform has rotated until the capture has been performed. In order to do that, we first need to transform all the clouds to a common coordinate system. This is achieved by translating the clouds to a fixed point in the platform considered as its center. This point is extracted by first performing a capture of the platform without objects and then manually selecting the center in the point cloud and getting its coordinates as shown in Figure 11. The actual center was marked in the platform to obtain the ground-truth information.

After the cloud is translated to the new origin we need an axis to rotate it: the normal of the platform in which the object is placed. In order to obtain that axis, we fitted a plain to a considerably large section of the platform extracted from the empty cloud, then we computed the normal vector to that plane. Figure 12 shows the plane fitting result. This vector, orthogonal to the platform, will allow us to rotate the object around it in the same way the turntable rotates thus providing a good estimation for the subsequent alignment operations. Each cloud will be rotated a certain amount depending on the steps performed by the turntable, i.e., the first capture will not be rotated and the second one will be rotated 5.625 ° in the same sense the turntable does.

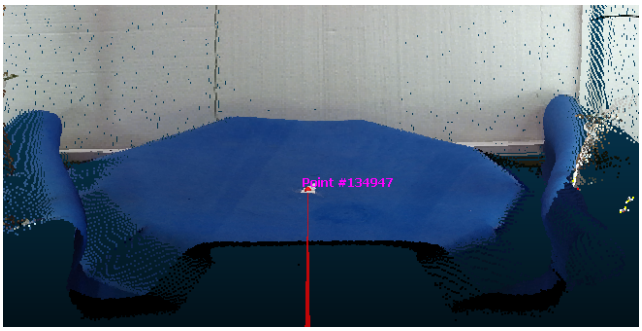

Fig. 11 Point cloud of the empty platform with the point at its center selected. Note that the actual center of the table is marked with a black dot.

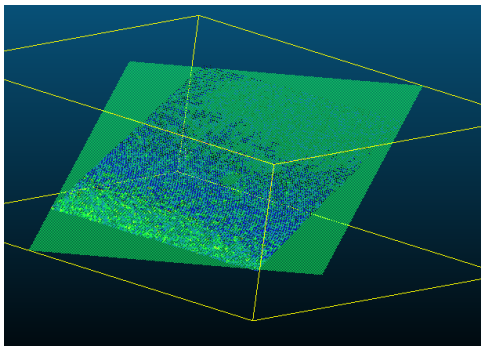

Fig. 12 Plane (green) fitting a section of the platform (blue) to obtain a stable normal vector of the turntable where the object is placed.

Once the cloud has been rotated, the final alignment can be performed. We apply the Iterative Closest Point (ICP) [13][14][15] algorithm implemented in PCL to align the current cloud with the last processed one (obviously, the ICP is not applied to the first cloud). This algorithm aligns two clouds iteratively refining the transformation until a certain stop criteria is reached, either a maximum number of iterations or some error threshold. Our application allows us to set both of them.

Figure 13 shows an example of aligment using the ICP algorithm to integrate the 64 views of an object of the dataset.

Fig. 13 64 point clouds aligned with ICP after applying all the noise removal operations, translation and rotation. Front view (left) and perspective (right).

5.3 Object reconstruction

After the 64 point clouds are registered, they are concatenated into a single one. This cloud is downsampled by means of the Voxel Grid filter implemented by the PCL. Depending on the selected leaf size the resulting cloud will have more or less points. Our application allows the user to specify the leaf size as a parameter in a flexible manner.

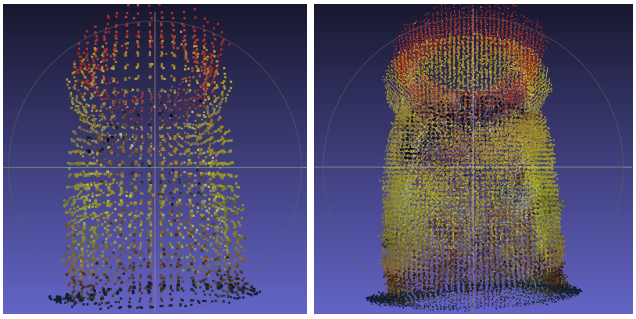

Fig. 14 Voxel Grid filter applied to a fully registered point cloud of an object of the dataset with a leaf size of 0.008 (left) and 0.003 (right).

Once the full cloud has been downsampled, a Moving Least Squares filter is applied to smooth, resample and reduce the noise of the registered cloud. We can specify multiple parameters of the filter such as the search radius, polynomial fit order and its upsampling method, and the radius and step size in order to obtain the best result for the current object and optimize its performance.

Before applying a reconstruction method to generate a mesh out of the point cloud, we estimated the normals of each point in the cloud using a Kd-tree search method to accelerate the nearest neighbor searching process. The search radius of the Kd-tree can be customized through a parameter.

At last, the Poisson Mesh Reconstruction algorithm is applied to the processed cloud. This method is also implemented in the PCL and our application allows the user to select the depth for the Poisson algorithm, varying this value will induce different levels of complexity in the reconstructed mesh **(see Figure 15)**. Finally, the mesh is exported to a PLY file.

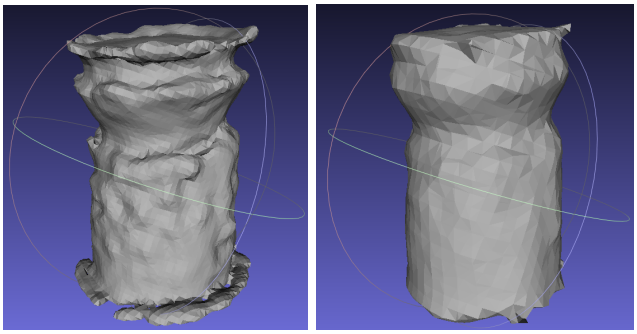

Fig. 15 Reconstructed meshes of the object shown in Figure 14 using the Poisson algorithm with a depth of 7 (left) and 5 (right), note the varying complexity of the mesh which increases with the depth.

6 Validation scenes generation

In this section we will describe the dataset labeling process. For this task we will use our own application developed in C++ with Qt and OpenCV [16]. This application will allow us to label the different objects in a scene by assigning a determined label to each pixel of the RGB image. Once the RGB images are annotated, the labels will be mapped to the depth maps and consequently to the point clouds. At last, we will register the different objects in the scene and manually refine that registration to annotate the scenes with the full pose of each one of them.

The main input data used in labeling process are the RGB color images obtained by our capture application. To label all the objects in those images, we will need to select and tag all the pixels of a particular object and repeat the same process for all the objects in the scene. This is a tedious task that through our application we try to automate as much as possible by applying image processing and artificial and computational intelligence algorithms.

6.1 GrabCut foreground extraction

In order to simplify the labeling process we will extract the foreground of each scene for an easier object segmentation. Many algorithms could be used for this purpose. Some of them use texture information such as

the Magic Wand, or edge/contrast such as Intelligent Scissors [17], other are based on matting, e.g., Bayes Matting [18][19] or Knockout 2. In our case, we will use a graph-cutting based one which combines both texture and edge information: the GrabCut algorithm [20], which is an extension of the traditional graph-cut approach [21] [22] with an improved version of the iterative optimization, a simplified user interaction process, and a brand new border matting algorithm.

The GrabCut algorithm allows us to extract accurate foreground alpha mattes with certain degree of user effort. It combines an automatic hard segmentation process by iterative graph-cut optimization with border matting to deal with blur and mixed pixels on boundaries. In addition, the user can manually mark conflictive pixels as background or foreground to help the algorithm and refine the results.

Figure 16 shows an example scene of the validation set, in which certain objects are present, and the resulting image after applying the GrabCut segmentation algorithm that was integrated into our application using the implementation provided by the OpenCV library.

Fig. 16 RGB image of a sample scene (left), and the same image after GrabCut segmentation (right).

6.2 Image preprocessing

After foreground segmentation, we will need to select the region of the image that we want to label. If we consider necessary, we proceed to preprocess the selected region in order to better discriminate and group similar pixels. This step can be helpful in the case of colorful images with a variety of similar pixel colors and different textures. For that, we will use a color quantization algorithm that reduces the number of colors in an image. Color quantization is the same as vector quantization, but applied in a three-dimensional space. There are different techniques for this purpose, in our case we will use k-means clustering where the numbers of clusters corresponds to the number of colors that the resulting image will have. Figure 17 shows an example of k-means clustering for color quantization. We can observe a clear reduction in the number of colors in the image. The advantage of this preprocessing step will be observed when selecting the pixels belonging to each object individually.

Fig. 17 Segmented scene after color quantization with 8 clusters (left), and after quantization with 48 clusters (right).

If we decrease the number of clusters, there exists the risk that objects with similar colors end up having the same color. In this case, we will not be able to properly segment the objects using the color information.

With our application we can also apply smoothing filters for images such as bilateral or median filters. Both are used to perform some kind of noise reduction and obtain smoother images. This will also help us to select the pixels individually because the color change between them will be reduced. Consequently, when selecting a pixel in a region, neighboring pixels will have similar color and we will be able to automatically select them by applying a small threshold in the color segmentation process.

6.3 Pixel selection and image segmentation

After the preprocessing step has finished, we will start selecting pixels and properly segmenting the image. In order to do that, we will start manually selecting pixels from the object we want to segment and label. Since selecting pixels one by one would be too tedious, once we select a pixel we will automatically include neighboring ones in that selection whose colors are similar considering a configurable threshold. That threshold can be changed on-the-fly so that the user can properly refine the segmentation by adding or removing groups of similar pixels.

Once we have obtained a satisfactory set of pixels for the target object, we have to choose a color for the object in the image, and assign it a label – the background will be labeled as -1 whilst the objects will be identified by positive integers consistenly throughout all the validation scenes. By doing this, we will map all the se-

lected pixels to a certain label and a color. This information will be stored in a TXT file, containing this manually determined ground-truth. This process is repeated for all the objects in the scene until it is fully annotated. Figure 18 shows an example of an individually tagged object and the fully labeled scene from Figure 16.

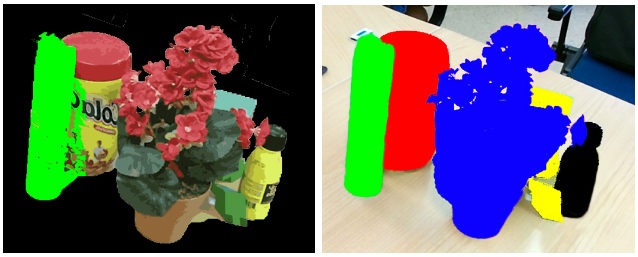

Fig. 18 Example of a tagged object in green (left) the result is refined later by tagging all the other objects of the original scene after segmentation (right).

The quality of this ground-truth heaviliy depends on the refinement level applied by the user. Our tool provides the means to perfectly segment and label the different objects of the scene. Although this task could be speeded up by using automatic algorithms, the result will not be nearly as perfect as this manual approach.

6.4 Depth maps and point clouds labeling

Once the RGB images have been labeled we can easily map the labels to the depth map and obtain a TXT file with each pixel and its label. This process is trivial when the sensor depth resolution is the same as the RGB camera one, but requires a more complex mapping when the color image has a different resolution, e.g., when using the Kinect 2. In the same fashion, the labeled pixels of the depth image can be mapped to the points of the scene clouds. By doing this, we obtain fully annotated RGB and depth maps, and also point clouds.

6.5 Pose annotation

In addition to pixel and point labeling, our dataset provides full pose annotation for each object in a scene. This process is semi-automatic and it is done using the open-source software CloudCompare[2] for point cloud manipulation and visualization.

[2] `http://www.danielgm.net/cc/`

Firstly, both scene and object clouds are loaded and a point-pair registration method is used to obtain a coarse estimate of the transformation which aligns the model to the object instance in the scene. Figure 19 shows this process in which we manually select points from both clouds that will be used for registration. The result of that process for a sample scene and a model is shown in Figure 20.

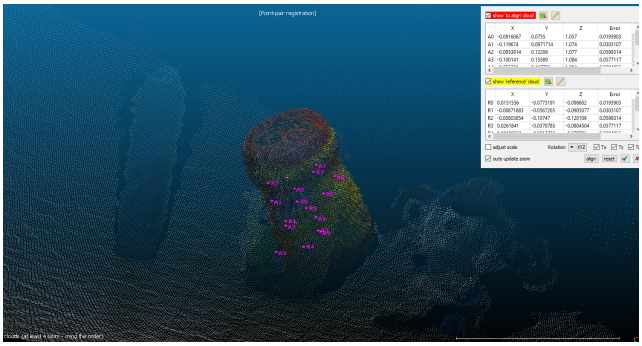

Fig. 19 Point-pair registration of a model cloud to a reference scene using CloudCompare to obtain a coarse estimate of the pose. The points of both clouds (shown in pink) are manually selected.

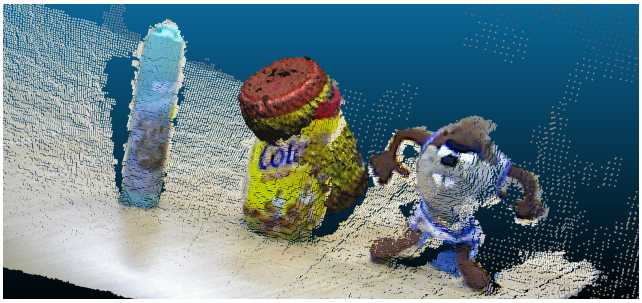

Fig. 20 Result of the point-pair registration performed to obtain a coarse estimate of the pose of the object model.

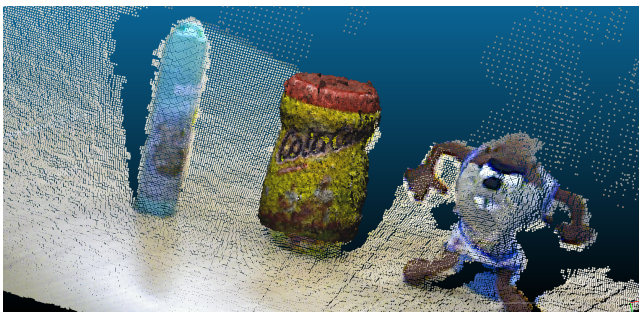

Fig. 21 Registration using ICP with the coarse estimate of the previous step as the initial transformation.

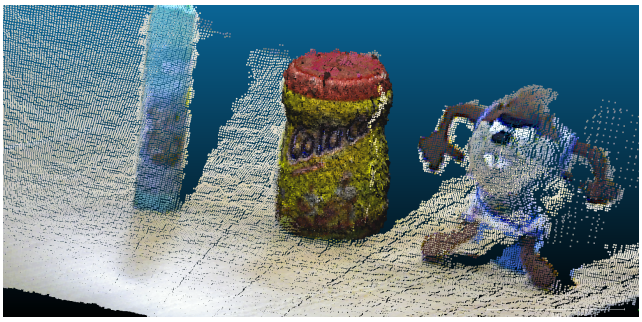

Fig. 22 Fine alignment performed manually after the ICP registration.

After that coarse transformation is obtained, the ICP algorithm is applied to refine that alignment. Figure 21 shows the same example as before after applying this registration method. As we can observe, the alignment of the model and its instance in the scene is significantly improved.

At last, we manually refine the pose by applying translations and rotations to increase the quality of the alignment of both clouds. The final result for this example is shown in Figure 22.

In the end, the final model position and orientation in the scene are stored in a TXT file together with its label. The process is repeated for every other object in the scene.

7 Conclusions

3D object recognition with full pose estimation is a challenging and relevant topic, specially for mobile robotics where a lot of applications require this kind of cognitive information about the environment. Several methods have been proposed to deal with this problem, and there is still room for improvement.

In this paper, we present a dataset which can be used as a benchmark for comparison and evaluation of different 3D object recognition methods. The dataset has been proposed to capture the strengths of already available datasets: a moderate number of objects, color information, per pixel labeling, bounding box labeling, category labeling, segmentation masks, 6DoF pose information, registered clouds and reconstructed objects, level of occlusion for each object, multiple devices, various formats, and a reasonable number of evaluation scenes. In addition, the tools which were developed to generate the dataset are available open-source [3].

As a future work, we will extend the dataset to include more objects and scenes with greater variability and also special data or situations that might be challenging for existing 3D object recognition methods.

[3] `https://github.com/Blitzman/multisensor-dataset-tools`

Acknowledgements This work was partially funded by the Spanish Government DPI2013-40534-R grant. This work has also been funded by the grant *"Ayudas para Estudios de Mster e Iniciacin a la Investigacin"* from the University of Alicante

References

1. Lai K, Bo L, Ren X, Fox D (2011) A large-scale hierarchical multi-view rgb-d object dataset. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, 1817–1824, IEEE
2. Richtsfeld A, Morwald T, Prankl J, Zillich M, Vincze M (2012) Segmentation of unknown objects in indoor environments. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, 4791–4796, IEEE
3. Ciocarlie M, Bradski G, Hsiao K, Brook P (2010) A dataset for grasping and manipulation using ros. In IROS Workshop: RoboEarth-Towards a World Wide Web for Robots, Taipei, Taiwan
4. Mian A, Bennamoun M, Owens R (2006) Three-dimensional model-based object recognition and segmentation in cluttered scenes. Pattern Analysis and Machine Intelligence, IEEE Transactions on 28(10):1584–1601
5. Mian A, Bennamoun M, Owens R (2010) On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. International Journal of Computer Vision 89(2-3):348–361
6. Aldoma A, Tombari F, Di Stefano L, Vincze M (2012) A global hypotheses verification method for 3D object recognition. In Computer Vision–ECCV 2012, 511–524, Springer
7. Singh A, Sha J, Narayan K, Achim T, Abbeel P BigBIRD: A Large-Scale 3D Database of Object Instances
8. Hinterstoisser S, Lepetit V, Ilic S, Holzer S, Bradski G, Konolige K, Navab N (2013) Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In Computer Vision–ACCV 2012, 548–562, Springer
9. Tombari F, Salti S, Di Stefano L (2010) Unique signatures of histograms for local surface description. In Computer Vision–ECCV 2010, 356–369, Springer
10. Tombari F, Salti S, Di Stefano L (2011) A combined texture-shape descriptor for enhanced 3D feature

matching. In Image Processing (ICIP), 2011 18th IEEE International Conference on, 809–812, IEEE
11. Salti S, Tombari F, Di Stefano L (2014) SHOT: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding 125:251–264
12. Rusu R, Cousins S (2011) 3D is here: Point Cloud Library (PCL). In IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China
13. Chen Y, Medioni G (1991) Object modeling by registration of multiple range images. In Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, 2724–2729, IEEE
14. Besl PJ, McKay ND (1992) Method for registration of 3-D shapes. In Robotics-DL tentative, 586–606, International Society for Optics and Photonics
15. Rusinkiewicz S, Levoy M (2001) Efficient variants of the ICP algorithm. In 3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on, 145–152, IEEE
16. Bradski G, Kaehler A (2008) Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc."
17. Mortensen E, Barrett W (1995) Intelligent scissors for image composition. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, 191–198, ACM
18. Ruzon M, Tomasi Cea (2000) Alpha estimation in natural images. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, vol. 1, 18–25, IEEE
19. Chuang Y, Curless B, Salesin D, Szeliski R (2001) A bayesian approach to digital matting. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 2, II–264, IEEE
20. Rother C, Kolmogorov V, Blake A (2004) Grabcut: Interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG) 23(3):309–314
21. Greig D, Porteous B, Seheult A (1989) Exact maximum a posteriori estimation for binary images. Journal of the Royal Statistical Society Series B (Methodological) 271–279
22. Boykov Y, Jolly M (2001) Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol. 1, 105–112, IEEE