A New Approach to 3D Object Recognition and Pose Estimation

Michela Antonelli
Dip. di Ingegneria dell'Informazione
University of Pisa
56122, Pisa, Italy
Email: michela.antonelli@iet.unipi.it

Marco Gabiccini
Dip. di Ingegneria Civile e Industriale,
University of Pisa,
56122, Pisa, Italy
email: marco.gabiccini@unipi.it

Francesco Marcelloni
Dip. di Ingegneria dell'Informazione,
University of Pisa,
56122, Pisa, Italy
Email: francesco.marcelloni@unipi.it

I. Introduction

In the last years, object recognition and pose estimation have become a key passage in several application scenarios, such as robot grasping and manipulation, scene recognition and understanding, large environment mapping and human posture recognition [1], [2], [3], [4]. The term pose estimation usually refers to the computation of a Special Euclidean transformation with 6 degrees of freedom, denoted SE(3) in the literature, between a model recognized from a database and its instance in the scene under analysis. This task is often paired with object recognition based on 3D descriptors [5], [6], [7], [8], [9].

In this paper, we focus on 3D images represented by point clouds acquired by an RGB-D depth camera. The geometry of a given object can be described by means of two different types of approaches: a local approach [10], [11], [12], [13] and a global approach [14], [15], [16], [17], [18]. The former determines some points of interest on the object, called *keypoints*, and calculates the descriptors on each spherical volume around these keypoints. The latter describes the object by means of features calculated on the point cloud taken as a whole (i.e., a single descriptor is calculated for each object). Local descriptors can be applied directly on cluttered scenes and are usually more accurate in describing the object. On the other hand, they are more complex in terms of both computational time and memory occupation than global descriptors.

In this paper, we approach the 3D object recognition and pose estimation by employing a 3D local descriptor, namely the Unique Signatures of Histograms for Local Surface Description (SHOT). In particular, we describe each object by means of a set of SHOT descriptors, one for every keypoint identified in the point cloud representing the object itself. Since each SHOT descriptor is represented by a histogram, each object is defined by a set of histograms.

Given a database of point clouds of known objects along with their set of SHOT descriptors, the classification of an unknown object is performed by, first, computing the SHOT descriptors for the object and then searching for the object in the database with the most similar set of SHOT descriptors.

To compare sets of SHOT descriptors, we introduce a new measure of similarity. We first build a similarity histogram for each object in the database by comparing the sets of SHOT descriptors of this object with the ones of the unknown object. Then, we extract five features from this similarity histogram. We use these features as input to a feed forward neural network (FFNN) that evaluates the similarity between the two objects: the unknown object is recognized as the object in the database with the highest similarity.

We tested our method on two datasets, one available online and the other acquired in our laboratory. On both the datasets, the results are quite promising.

In Section II, we introduce an overview of the system. From Section III to Section VI, we describe each module of the system in detail. Section VII presents some experimental results and finally Section VIII draws some conclusion.

II. System overwiew

Fig.1 shows an overview of the system where each step of the object recognition and pose estimation process is highlighted. The input to the system is a point cloud of a scene recorded by means of an RGB-D depth camera. The point cloud is processed by the *scene pre-processing* step. This step first performs a scene segmentation for isolating the point cloud of the unknown object (O_x). Then, it removes the outliers from the point cloud of the isolated object since these outliers could affect the estimation of local point cloud characteristics. Finally, a down-sampling of the point cloud is applied for reducing the number of points that represent each object.

The point cloud of O_x, therefore, undergoes the step of *object descriptor computation* that computes the 3D local descriptors used to characterize the point cloud. In particular, we determine a set of keypoints on the object and calculate a descriptor for each of them. The output of the object descriptor extraction is a set of 3D descriptors that characterize O_x and are used in the subsequent steps.

To recognize the unknown object, we first calculate the similarity histogram between O_x and each object contained in the object database. The similarity histogram is built by comparing the set of descriptors calculated on O_x with the set of descriptors calculated offline on the labeled objects contained in the object database; similar objects will have a similarity histogram with high values around zero. Then, a set of meaningful features are extracted from the similarity histogram: these features are used as inputs of a feed forward neural network (FFNN). The output of FFNN is the similarity degree between the two objects, which have generated the similarity histogram. Finally, the system assigns to the unknown object the label of the object that has obtained the highest similarity degree.

In the following sections, each step of the system is described in more detail.

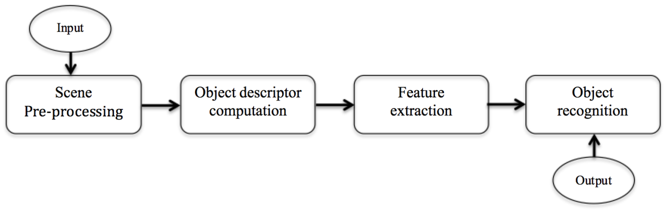


Fig. 1. System overview

III. Scene pre-processing

The *scene pre-processing* step is applied to a 3D image recorded by an RGB-D depth camera. The scene is set up by a table with few objects on the top. An example of an image of a scene containing a table with a cup, a funnel and a container on the top is shown in Fig. 2. The scene pre-processing step first removes the table from the scene, then applies a filter to eliminate the outliers and finally segments the scene for extracting the single object. Since this paper mainly focuses on recognition of objects and their pose estimation, we consider simple scenes consisting of only few objects on the table without any occlusion. Further, to extract the point clouds related to each object we follow the approach introduced in [19] by applying the functions defined in the PCL library [20]. In particular, we employ the following procedure:

- we find the area related to the table by looking for clusters of points with planar shapes. In our experiments, the clusters with planar shapes will be related to both the floor and the table. Between these two clusters, we identify the cluster representing the table as the one with the lower number of points;
- we calculate the average value of the height of the table (i.e., the distance between the floor and the table);
- we remove the table from the scene;
- we apply the filter to remove the outliers;
- we find all other clusters related to the other objects in the scene;
- we identify the objects as the clusters that have the same height of the table.

The process of outlier removal is based on the method proposed in [21]. An outlier is a point that is distant from other points: an outlier could arise from sensor noise or simply from a bad acquisition (bad light). To detect an outlier, we use a statistical analysis of each point with respect to its neighborhood.

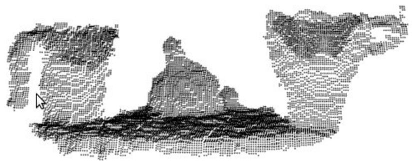


Fig. 2. An example of RGB-D image of a scene containing a table with a cup, a funnel and a container on the top (from the right to the left)

Let p, P^K and $\overline{d_p}$ be a point of the point cloud P, the set of k nearest points of p, and the mean distance between p and each point in P^K, respectively. We calculate the mean μ_p and the standard deviation σ_p of the k nearest points of p and remove from the point cloud the points that are not in the set $P^* = \{p^* \in P | (\mu_p - \alpha * \sigma_p) \leq d^* \leq (\mu_p + \alpha * \sigma_p)\}$

The parameter α is a weight that controls the width of the interval, making the filtering process more or less restrictive (we consider $\alpha = 1$ in our experiments).

Figure 3 shows the results of the pre-processing step applied to the scene represented in Fig. 2. In particular, Figs. 3(a), 3(b) and 3(c) show the point clouds after removing the points belonging to the table, after the application of the outlier removal filter and after the segmentation of the three detected objects, respectively. We observe that the points belonging to the point cloud of the three detected objects have different colors. After the segmentation, the point clouds of the objects are processed independently of each other, and undergo to object descriptor extraction.

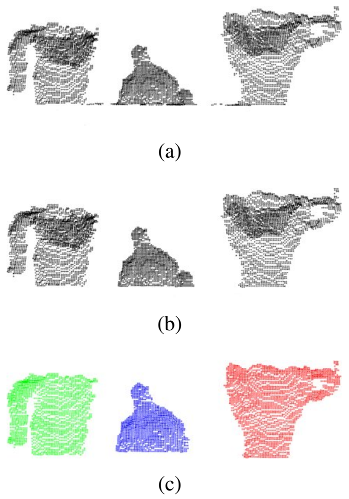


Fig. 3. (a) Point cloud of the scene without the table, (b) point cloud after the application of the filter for removing the outliers, (c) segmented objects

IV. Object descriptor computation

Once the point cloud of the unknown object has been isolated, we compute a set of descriptors which characterize the point cloud of the object. To this aim, we use a local approach, which determines some keypoints on the object and calculate the descriptors on a spherical volume around these keypoints. In particular, we use the SIFT Keypoint Detector algorithm [22] to identify the keypoints from the point cloud. This algorithm uses a cascade filtering approach. First, a difference-of-Gaussian function is used to identify potential points of interest that are invariant to scale and orientation. Then the points that have low contrast or are poorly localized along an edge are rejected. Figure 4 highlights in red the keypoints extracted from the point cloud of a bottle.

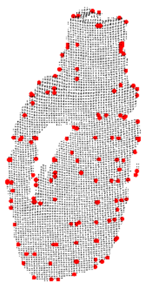


Fig. 4. Keypoints (in red) extracted from the point cloud of a bottle

For each keypoint, we compute the Signatures of Histograms of Orientation (SHOT) descriptor [23]. Following the categorization introduced in [23], the state-of-the-art 3D descriptors can be grouped into two main categories: Signatures-based methods and Histograms-based methods. The first methods describe the neighbourhood of a keypoint (i.e., the support of a keypoint) by encoding geometric measurements calculated with respect to the coordinates according to an invariant local reference frame (RF) defined on the support. The second methods describe the support of a keypoint by encoding counters of local geometrical or topological measurements into histograms. SHOT belongs to the intersection of these two categories. Indeed, beside encoding histograms of the normals within the support, it also adds geometrical information regarding the location of the points within the support by computing a set of local histograms on a 3D volume generated by superimposing a 3D grid on the support. In particular, let $K = \{K_1, K_2, \ldots, K_n\}$, S_{K_i} and $\mathbf{n_l}$ be the set of keypoints, the set of points within a sphere of radius R centered into K_i and the normal at the *l-th* point of the point cloud. For each $k_i \in K$, the SHOT descriptor is calculated as follows:

1) superimpose a spherical grid S_{K_i}, centered in K_i and divided into 32 volumes (see Fig. 5): each volume results from 4 azimuth, 2 radial and 2 elevation divisions of the sphere;
2) calculate the local RF using all the points in S_{K_i};
3) for each volume V_j, with $j \in \{1, \ldots, 32\}$, calculate a local histogram of 11 bins by accumulating point counts depending on the angle θ_q between the normal n_l at the *l-th* point within V_j and the normal at the keypoint K_i (see Fig. 5);
4) merge together all the local histograms obtaining a descriptor length of 352;
5) normalize the descriptor to Euclidean norm.

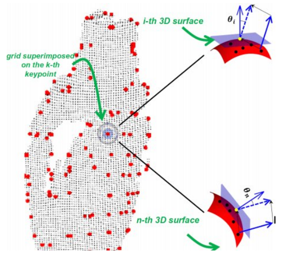


Fig. 5. Computation of the SHOT descriptors on the point cloud of a bottle

V. Feature extraction

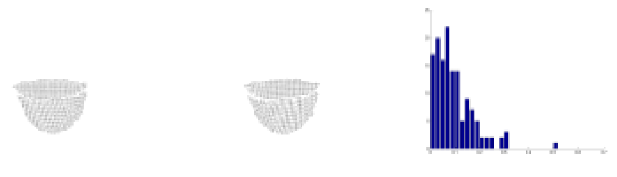
After executing the object descriptor computation, each object is characterized by a set of SHOT descriptors, one for each keypoint detected in the point cloud. Let us assume that a database of known objects along with the corresponding sets of descriptors is available. Then, the descriptors of an unknown object (*query object*) are compared with the sets of descriptors of all the objects in the database: the query object is labeled with the class of the known object with the most similar sets of descriptors to the ones of the query object. Since each object is described by a set of histograms, we have to introduce a measure of similarity between sets of histograms.

Let O_d and O_x be a known object of the dataset and the query object, respectively. To define the similarity between O_d and O_x, first of all, we compute a histogram, the *similarity histogram*, as follows. For each keypoint K_i of O_x, we search for the nearest keypoint K_t of O_d and store the distance d_{it}. The distance between the keypoints is computed as the distance between the corresponding histograms. This distance is calculated by considering the histograms as vectors (each value of the bin is an element of the vector) and therefore applying the Euclidean distance between them. After computing all the distances d_{it} for each keypoint K_i, we plot these distances in a histogram. The x-axis of the histogram represents the distances from 0 to D_{MAX}: D_{MAX} is the maximum distance we consider (in the experiments, $D_{MAX} = 0.6$). If a distance is larger than D_{MAX}, this distance is inserted into the last bin. The x-axis is partitioned into M equi-width bins. If the query object O_x and the object O_d are close to each other, the histogram is skewed right since distances close to 0 are more frequent than large distances; otherwise, it should be skewed left. Figures 6(a) and 6(b) show two examples of similarity histograms calculated, respectively, on two equal and two different objects. As expected, the distribution of the distances is different in the two cases.

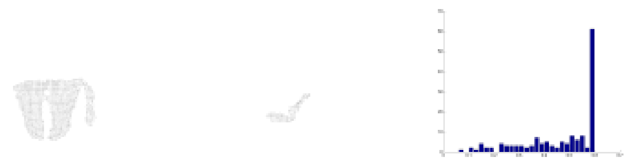
From each similarity histogram, we extract a set of features that are used as inputs to the FFNN. Let $D = \{d_1, \ldots, d_M\}$ be the set of bins of the similarity histogram. Let $Hs = \{h_1, h_2, \ldots, h_M\}$ be the set of occurrences of the distances in the corresponding bins. We extract the following features:

i) Expected value:

$$\mu = \frac{1}{NK_{O_x}} \sum_{i=1}^{N} h_i \cdot d_i \qquad (1)$$



(a)



(b)

Fig. 6. Similarity histograms calculated on two point clouds (a) of the same object in the same pose and (b) of different objects

where $NK_{O_x=\sum_{i=1}^{N}h_i}$ is the number of SHOT descriptors considered, which coincides with the number of keypoints of the unknown object O_x.

ii) Variance:

$$\mu = \frac{1}{NK_{O_x}} \sum_{i=1}^{N} h_i \cdot d_i^2 - \mu^2 \tag{2}$$

iii) Standard Deviation:

$$\sigma = \sqrt{\frac{h_i}{NK_{O_x}} \cdot (d_i - \mu)^2} \tag{3}$$

iv) Symmetry:

$$\gamma = \frac{1}{NK_{O_x} \cdot \sigma^3} \sum_{i=1}^{N} (d_i - \mu)^3 \cdot \frac{h_i}{N_D} \tag{4}$$

v) Different number of keypoints between the two objects:

$$ND = |NK_{O_x} - NK_{O_d}| \tag{5}$$

where NKO_d is the number of keypoints calculated on the reference object O_d of the database and $||$ is the absolute value operator.

This feature takes into account that similar objects should have point clouds characterized by a similar number of keypoints.

For each pair (O_x, O_d), the five features are used as input to a feed forward neural network (FFFN). The FFNN outputs a value in [0,1] which determines the similarity between O_x and O_d.

VI. Object Recognition

In order to calculate the similarity between two objects we exploit the FFNN of Fig. 7.

The input layer contains as many neurons as the number of features used to describe the similarity histogram calculated between object O_d and object O_x. Each input neuron is connected to all the neurons of the hidden layer, which in their turn are connected to the unique neuron of the output layer.

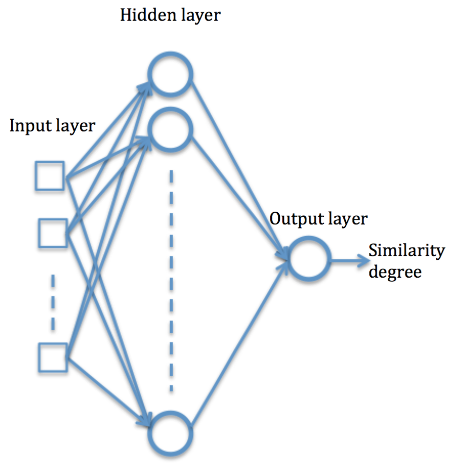


Fig. 7. Structure of the FFNN

The neurons of the hidden layer perform a weighted sum of the inputs and their outputs are calculated by applying an activation function to this weighted sum. More precisely, let x_s, with $s = 1, \ldots, 5$, and $w_{s,h}$ be the inputs to the network and the weights on the connections between the neurons of the input layer and the $h - th$ neuron of the hidden layer, respectively. The output y_h of the $h-th$ neuron in the hidden layer is calculated by applying the activation function $f(\cdot)$ to the weighted sum of the inputs, that is,

$$y_h = f(\sum_{s=1}^{t} x_s \cdot w_{s,h}) \tag{6}$$

In the experiments we have used as activation function the sigmoid function $f(x) = \frac{2}{1+e^{-2x}} - 1$.

All the H neurons of the hidden layer are connected to the single neuron n_o of the output layer. The output O of this neuron, that is the similarity degree between the unknown object O_x and one object O_d of the database, is determined by applying the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ to the weighted sum calculated as:

$$O = \sum_{h=1}^{H} y_h \cdot w_{h,o} \tag{7}$$

where y_h is the output of the $h-th$ hidden neuron, and $w_{h,o}$ is the weight of the connection between the $h - th$ hidden neuron and the output neuron. In our experiments, we adopted $H = 30$ hidden neurons.

To train the network we use a training set made up by pairs of both equal and different objects. Furthermore, as explained in the experimental part, we also consider pair of equal objects but in different pose. The output of the network is 1 either when the two objects are equal and in the same pose, or when the two objects are equal and have a symmetrical shape. We calculate the error between the output of the FFNN and the desired output and we use the back-propagation algorithm to update the weights.

The query object O_x is recognized to be the same object and in the same pose of the object O_d with the highest degree of similarity with O_x.

VII. Experimental part

The proposed approach has been tested on two datasets of household objects. The first, called the Willow Garage dataset [24], contains 31 objects. The second, called Ikea dataset, is composed of 23 objects whose point clouds have been acquired in our laboratory.

In both datasets, each object is described by 36 point clouds. Indeed, since we are interested in recognizing also the pose of the objects, each object is rotated along the vertical axis at steps of 10° and a point cloud is acquired at every step. In the following subsections, we introduce in more detail the two datasets and the obtained results.

A. The Willow garage dataset

This dataset contains 1116 point clouds (31 objects, each acquired in 36 rotations). We divide the dataset into two sets, S_{tr}, and S_{ts}, to be used for training and testing, respectively.

S_{tr} contains, for each object, the point clouds corresponding to the rotations multiple of 20° and 30° starting from 0° (0°, 20°, 30°, 40°, 60°,. . . , 330°, 340°). The training set contains 713 point clouds.

The test set contains the point clouds corresponding to all the remaining rotations (10°, 50°, 70°, . . . , 350°), for a total of 403 point clouds.

Since the inputs to the FFNN are pairs of objects and the output is a similarity value, the training and test sets of the FFNN have been generated as follows. As regards the training set, we made up every possible pair of point clouds contained in S_{tr} (for a total of 553,536 pairs). For each pair we use as input to the FFNN the 5 features extracted from the similarity histogram and the output of each pair is assigned using the following criterion.

We group each object into categories (cups, bottles, books, etc.) since the resolution of the RGB-D camera is not able to appreciate small differences. Further, as the final aim of this object recognition system is to infer how the object can be grasped, we do not need to distinguish between very similar objects.

For each pair of the training set we assign an output equal to 1 (equal point clouds) in the following cases:

- if they represent two objects of the same category in the same pose (same rotation along the vertical axes).
- if they represent two objects of the same category and the difference between the rotation angles is lower than ±20°.
- if they represent two symmetrical objects of the same category independently of the rotation. We consider as symmetrical axes the vertical axes used to rotate the object.

		Target	
Output	0	21138	3695
	1	5821	23433

TABLE I
Average confusion matrix calculated on the training set

Specificity	78.4
Sensitivity	86.4
Accuracy	82.4

TABLE II
Specificity, Sensitivity and accuracy calculated on the training set

Further, as examples of different objects (output of the network equal to 0), we include in the training set only the pairs made up by point clouds of objects belonging to different categories. In this way, we avoid to give wrong information to the network. Indeed, two point clouds of objects of the same category could be very similar also when the rotation is greater than ±20°. For this reason, to train the network we prefer to use only pairs of objects whose equality/disequality is unambiguous.

By using the above criteria we generate a training set made up by 476,928 and 33,804 pairs whose output is equal to 0 (negative samples) and 1 (positive samples), respectively. Being this training set unbalanced, we randomly choose only 33,804 pairs among the negative samples and we remove the others from the training set. In this way we end up with a training set of 67,608 samples, 33,804 positive samples and 33,804 negative samples. This process is repeated ten times so as to avoid any bias on the random choice of the negative samples.

As regards the test set, we generate every possible pair between each point cloud of S_{ts} and the point clouds contained in S_{tr} (for a total of $403 \cdot 713 = 287,339$ pairs). We automatically assign the output using the same criteria used for the training set.

In Tab. I we show the average confusion matrix obtained on the ten trials and on the training set after 50 epochs. Table II shows the corresponding values of specificity, sensitivity, and accuracy.

In Tab. III, we show the average results calculated on the test set. To assign the output to the pairs of the test set, we consider the same criterion used for the pairs of the training set.

The results are encouraging, mainly considering that the output on some test samples could be considered wrong but, due to a particular symmetry of the object, the two point clouds could be, in fact, very similar. By improving the way of assigning the output to each pair of point clouds, the results should improve further.

Object correctly recognized	98.2%
Object correctly recognized and rotation recognized with an error of ±10°	76.3%
Object recognized but wrong rotation	21.9%
Object not recognized	1.8%

TABLE III
RESULTS ON THE TEST SET

		Target	
Output	0	13106	3594
	1	4755	14348

TABLE IV
AVERAGE CONFUSION MATRIX CALCULATED ON THE TRAINING SET

Specificity	73.4
Sensitivity	80.4
Accuracy	76.7

TABLE V
SPECIFICITY, SENSITIVITY AND ACCURACY CALCULATED ON THE TRAINING SET

B. *The Ikea dataset*

We acquired the point clouds of 23 objects with the aid of a DoF turntable, with vertical axis. Objects like mugs, various containers, plates, glasses, spoons and kitchen utensils were put on the turntable center, with Xtion Pro sensor pointing at it and located at a fixed position. Figs. 8 and 9 show, respectively, the objects contained in this dataset and an example of point cloud acquired by means of the turntable.

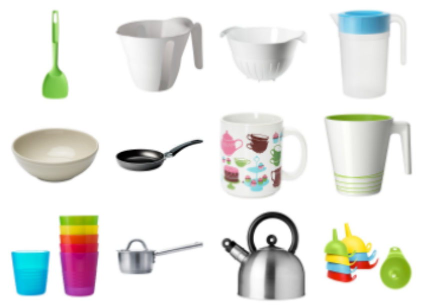


Fig. 8. Objects used in the second experiment

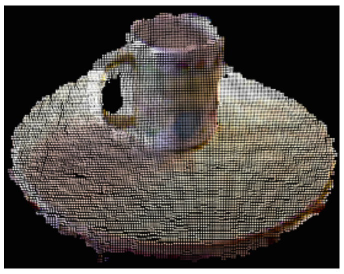


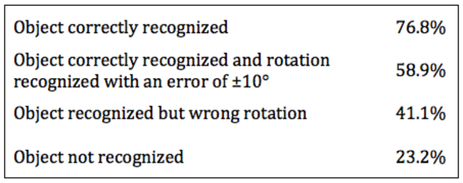
Fig. 9. Example of point cloud acquired by using the turntable

The turntable was rotated at steps of 10°: at each step the sensor acquired a point cloud, thus collecting 36 point clouds per object (828 point clouds in total). The overall process was repeated in different three days, so as to have three distinct rounds of acquisition. Since the quality of acquisition strongly depends on the ambient light, the point clouds acquired for the same object and the same orientation result to be different from one round to another.

Another factor affecting data quality is the presence of shining or semi-transparent surfaces, since these types of surface could scatter the infrared signal, causing large chunks of the point cloud to be missing or appearing very cluttered. To cope with this issue, we covered sensible surfaces, like metal, plastic, and shining ceramics, by using an opaque film with the aim of reducing the signal scattering. However, intrinsic noise in the depth sensor still lowers the quality of point clouds for certain objects, as highlighted in Figure 9. We decided to keep those bad quality acquisitions in order to simulate realistic situations met by an autonomous robot. Data diversification is also ensured by the unavoidable human error in positioning the object on the table. This could lead to inaccuracies in objects acquired at the same orientation, across different rounds of acquisition. For example, it is possible that a mug, acquired at an orientation of 10° in a round, is more similar to the same mug, acquired at 20° in another round. As explained in the previous sections, before saving the point cloud we apply some pre-processing on the images in order to extract from the scene only the object.

We used, in turn, one of the three rounds of acquisition as training set, and the others as test sets (defined as R_{TR} and R_{TS}, respectively). Each round contains 828 point clouds (23 objects times 36 rotations). As explained in the previous subsection, for building the training and test sets we consider, respectively, every possible pair of the point clouds contained in R_{TR} (685,584 pairs) and every possible pair formed by the point cloud contained R_{TS} and the point clouds contained R_{TR} (1,371,168 pairs). The output of each pair has been assigned as described in the previous subsection. Tables IV and V show the average confusion matrix and the values of specificity, sensitivity and accuracy obtained on the training set.

As regards the test set, the results are shown in Tab. VI. We observe that our method obtain better results on the first dataset. Indeed, for the Ikea dataset, the recognition performance drops for both objects and rotations. This can be due to several aspects: the point clouds of the first dataset contain a lower amount of noise than the ones contained in the Ikea dataset. Further, the Ikea dataset contains a larger variety of objects and the acquisitions are performed in different conditions of light.



Object correctly recognized	76.8%
Object correctly recognized and rotation recognized with an error of ±10°	58.9%
Object recognized but wrong rotation	41.1%
Object not recognized	23.2%

TABLE VI
RESULTS ON THE TEST SET

The recognition performance of the pose is quite low (58.9%) due mainly to the automatic procedure of assigning the output. Indeed, two point clouds representing the same object acquired in two different rotations, could be similar even if the difference between the two rotations is greater than 20°. Fig. 10 shows examples of pairs of point clouds (one in red and the other in green) considered erroneously equal by the system. The reader can easily realize that actually these point clouds are very similar and the error is only due to the automatic labeling procedure used to build the training and test sets.

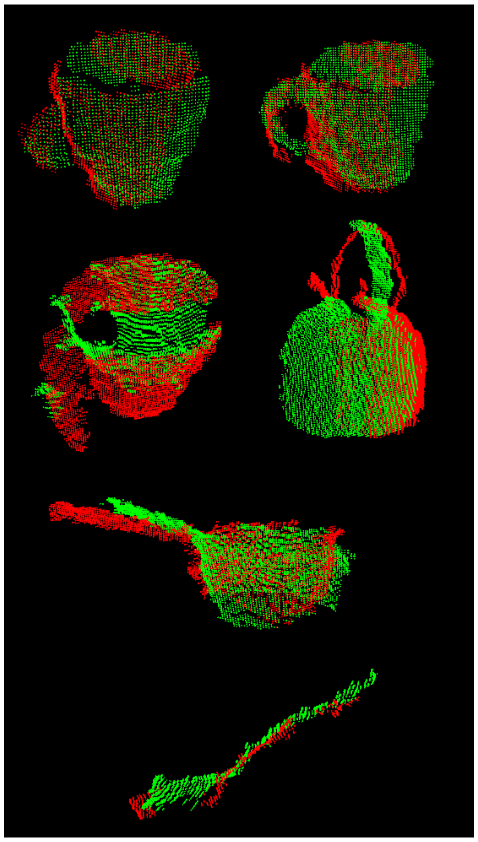


Fig. 10. Objects used in the second experiment

REFERENCES

[1] C. English, G. Okouneva, P. Saint-Cyr, A. Choudhuri, and T. Luu, "Real-time dynamic pose estimation systems in space: Lessons learned for system design and performance evaluation," *Special issue on Quantifying the performance of intelligent systems, International Journal of Intelligent Control and Systems*, vol. 16, no. 2, pp. 79–96, 2011.

[2] J. Bandouch, F. Engstler, and M. Beetz, "Accurate human motion capture using an ergonomics-based anthropometric human model," in *Articulated Motion and Deformable Objects*, ser. Lecture Notes in Computer Science, F. Perales and R. Fisher, Eds. Springer Berlin Heidelberg, 2008, vol. 5098, pp. 248–258.

[3] H. Ning, W. Xu, Y. Gong, and T. Huang, "Discriminative learning of visual words for 3d human pose estimation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[4] P. Doliotis, V. Athitsos, D. Kosmopoulos, and S. Perantonis, "Hand shape and 3d pose estimation using depth data from a single cluttered frame," in *Advances in Visual Computing*. Springer, 2012, pp. 148–158.

[5] S. May, D. Droeschel, D. Holz, C. Wiesen, S. Fuchs *et al.*, "3d pose estimation and mapping with time-of-flight cameras," in *International Conference on Intelligent Robots and Systems (IROS), 3D Mapping workshop, Nice, France*, 2008.

[6] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam, "Voting-based pose estimation for robotic assembly using a 3d sensor," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1724–1731.

[7] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 731–738.

[8] H. Kollnig and H.-H. Nagel, "3d pose estimation by directly matching polyhedral models to gray value gradients," *International Journal of Computer Vision*, vol. 23, no. 3, pp. 283–302, 1997.

[9] M. Andriluka, S. Roth, and B. Schiele, "Monocular 3d pose estimation and tracking by detection," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 623–630.

[10] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent Point Feature Histograms for 3D Point Clouds," in *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10), Baden-Baden, Germany*, 2008.

[11] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 356–369. [Online]. Available: http://dl.acm.org/citation.cfm?id=1927006.1927035

[12] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3212–3217.

[13] A. Johnson, "Spin-images: A representation for 3-d surface matching," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[14] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 47–54.

[15] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, Dec 2011, pp. 2987–2992.

[16] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 2155–2162.

[17] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, and G. Bradski, "Cad-model recognition and 6dof pose estimation using 3d cues," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Nov 2011, pp. 585–592.

[18] A. Aldoma, F. Tombari, R. Rusu, and M. Vincze, "Our-cvfh oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation," in *Pattern Recognition*, ser. Lecture Notes in Computer Science, A. Pinz, T. Pock, H. Bischof, and F. Leberl, Eds. Springer Berlin Heidelberg, 2012, vol. 7476, pp. 113–122.

[19] R. B. Rusu, *Semantic 3D Object Maps for Everyday Robot Manipulation*, ser. Springer Tracts in Advanced Robotics. Springer, 2013, vol. 85.

[20] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *ICRA*. IEEE, 2011.

[21] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927 – 941, 2008.

[22] D. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] S. Salti, F. Tombari, and L. D. Stefano, “Shot: Unique signatures of histograms for surface and texture description,” *Computer Vision and Image Understanding*, vol. 125, no. 0, pp. 251 – 264, 2014.

[24] J. Tang, S. Miller, A. Singh, and P. Abbeel, “A textured object recognition pipeline for color and depth image data.” in *ICRA*. IEEE, 2012, pp. 3467–3474.