Optics and Lasers in Engineering 54 (2014) 222–231

Contents lists available at ScienceDirect

Optics and Lasers in Engineering

journal homepage: www.elsevier.com/locate/optlaseng

Optoranger: A 3D pattern matching method for bin picking applications

Giovanna Sansoni [a,*], Paolo Bellandi [a], Fabio Leoni [a], Franco Docchio [b]

[a] Department of Information Engineering, Laboratory of Optoelectronics, University of Brescia Via Branze, 38 25123 Brescia BS, Italy
[b] Department of Mechanical and Industrial Engineering, University of Brescia Via Branze, 38 25123 Brescia BS, Italy

ARTICLE INFO

Article history:
Received 26 March 2013
Received in revised form
17 July 2013
Accepted 21 July 2013
Available online 12 August 2013

Keywords:
3D metrology
Bin picking
Robotics
3D matching
3D segmentation

ABSTRACT

This paper presents a new method, based on 3D vision, for the recognition of free-form objects in the presence of clutters and occlusions, ideal for robotic bin picking tasks. The method can be considered as a compromise between complexity and effectiveness. A 3D point cloud representing the scene is generated by a triangulation-based scanning system, where a fast camera acquires a blade projected by a laser source. Image segmentation is based on 2D images, and on the estimation of the distances between point pairs, to search for empty areas. Object recognition is performed using commercial software libraries integrated with custom-developed segmentation algorithms, and a database of model clouds created by means of the same scanning system.

Experiments carried out to verify the performance of the method have been designed by randomly placing objects of different types in the Robot work area. The preliminary results demonstrate the excellent ability of the system to perform the bin picking procedure, and the reliability of the method proposed for automatic recognition of identity, position and orientation of the objects.

1. Introduction

To maintain global competitiveness, the manufacturing industry must focus on flexibility and re-configurability, to produce customized work-pieces in a fast an efficient way [1]. Most of the operations are usually performed by Robots in the production line. One of the still unresolved issues is the ability of Robot manipulators to perform well in non-structured environments, where neither shape nor posture of the objects are predictable [2]. Machine vision, both 2D and 3D, is considered to be an essential aid in this context [3].

Fast and effective bin picking from non-organized containers is one of the present challenges for Robots, required to identify objects and to estimate their spatial location and orientation in unstructured bins [4]. Main issues of an effective bin picking are (i) acquisition of the scene through vision, (ii) scene segmentation (i.e., separation of different objects), and (iii) recognition and pose estimation of the segmented objects. These three tasks should be performed in the presence of clutter, shape variability, occlusions, object overlapping. All these aspects make bin picking a still almost unresolved problem.

Presently, scene segmentation and object recognition are based on 2D vision techniques, ideal for elements whose third dimension is negligible, lying on structured bins or on vibrating elements. Blob analysis and learning-based approaches are used, although with some limitations [5].

With complex objects, randomly oriented and partially overlapped, the 3D approach is highly preferable [6], using commercially available 3D scanners based on different principles (time-of-flight, passive and active stereo vision) [7–9]. In 3D vision, scene segmentation is based on depth discontinuities in the point cloud of the scene [10]. Other methods are based on region growing and curvature estimation, in search for planar, convex or concave continuous surfaces [11]. Edge-based approaches are more accurate in detecting border locations, however, the results need further elaboration in order to increase reliability.

Object recognition can be based on the construction of a database of 3D CAD models: every segmented element of the scene is compared with those models, to estimate their orientation [12]. Using 3D CAD models presents the drawbacks of high computational time and resources, as well as the need to create CAD models of the objects and the requirement of a good initial pose estimation to avoid local minima. An alternative model-free approach is based on the extraction of invariant geometric features from the 3D range image [13–15]: these approaches detect simple shapes in work areas with a single topology of objects. More sophisticated techniques, based on super-quadrics [16] spin images [17] and tensors [18,19] allow efficient recognition in semi-clutter scenes: however, their level of complexity might not be compatible with on-line bin picking in industrial plants.

* Corresponding author. Tel.: +39 30 371 5446; fax: +39 30 380 014.
E-mail address: giovanna.sansoni@ing.unibs.it (G. Sansoni).

0143-8166/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.optlaseng.2013.07.014

Methods based on 3D template matching, where the point cloud of a segmented object is compared to the point cloud of a template, present the ability to treat complex shapes that cannot be modelled by local features [20]. Thanks to their robustness, these approaches to object recognition in bin picking applications are being considered with increasing attention.

Our Laboratory is focused on solutions for the industrial world of Robotic manipulation and bin picking using 3D vision [21]. Our approach to satisfy industrial needs in terms of (i) short time-to-market, (ii) high cost effectiveness, and (iii) ease of implementation of the solution, is based on the use of existing hardware and software tools where available and economically sustainable, but trying to adapt some of the hardware or software tools when the commercial ones should be improved in some of their aspects.

We have recently been focused on the implementation of a full 3D solution for bin picking applications, in the presence of semi-cluttered scenes with objects characterized by both simple geometries and free-form shapes. We followed a "building brick" philosophy to simplify the development: we chose a market available laser slit as the acquisition device [22], and the *Match 3D tool* of the commercial 3D Shape Analysis Library (SAL3D) Library (AQSense Inc., Spain) to perform object identification and pose estimation [23]. *Match3D Coarse* follows a template matching approach, based on a best-fit algorithm that quickly aligns and compares 3D point clouds with their respective models (templates).

To complete the chain, a suitable segmentation tool was required, and we focused our efforts to develop a novel tool with superior performances with respect to the state of the art in terms of speed and flexibility. Our segmentation is built on the modules available from the open-source Point Cloud Library (PCL) platform [24]; this library offers an advanced and extensive approach to 3-D manipulation. In particular, we implemented an iterative algorithm based on the Euclidean distance between points in the cloud, that allows to efficiently segment the scene, in the presence of objects of different shapes, dimensions and orientations. The same tool was used to successfully treat occlusions and object overlapping of segmented elements, and this is not present in the state of the art.

In this paper the 3D acquisition procedure, the cloud segmentation and the object identification procedures are presented, together with experimental results performed to test them.

2. Workflow of operation

Fig. 1 shows the workflow of the operations implemented in the method. It is based on three main blocks. The first one performs the acquisition of the 3D scene; this is accomplished by (i) calibrating the acquisition device, (ii) scanning the work area, and (iii) pre-processing the point cloud to simplify subsequent elaboration.

The second block implements the segmentation of the 3D range map. To this aim, the point cloud is filtered to remove data belonging to the transition regions between objects that are partially overlapped or occluded. Clusters are extracted and suitably processed to select those that are represented by 3D sub-clouds corresponding to non-occluded objects. Finally, a restoring operation is carried out on each selected cluster, to maximize the visibility of the 3D features in each sub-cloud.

The last block performs the identification of the objects corresponding to the clusters, and estimates their pose for robot picking. Object identification is performed by matching the available templates to the selected clusters. The templates are 3D clouds of the objects. 3D matching is carried out by means of an alignment algorithm, implemented in the SAL3D Library. The output parameters represent the pose information, that is used as the input to the robot. In the following sections, these blocks are described in detail.

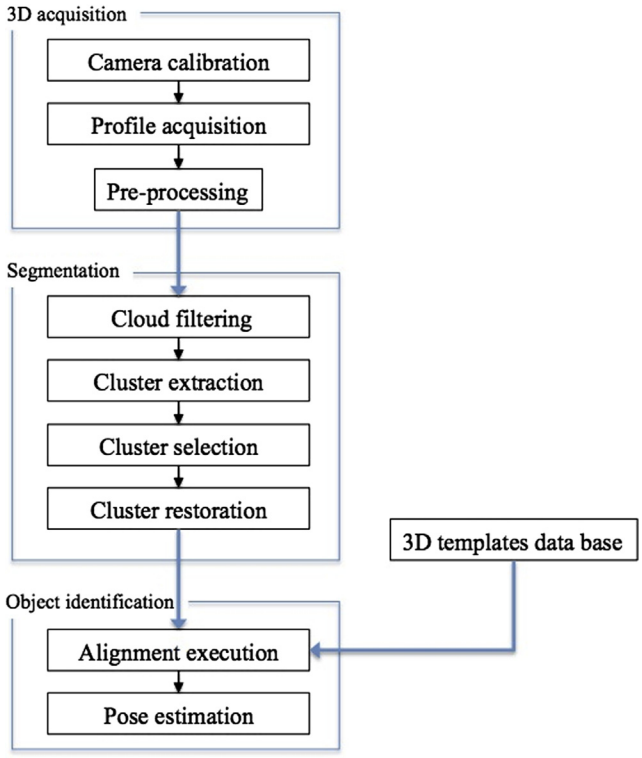

Fig. 1. Flowchart of the procedure.

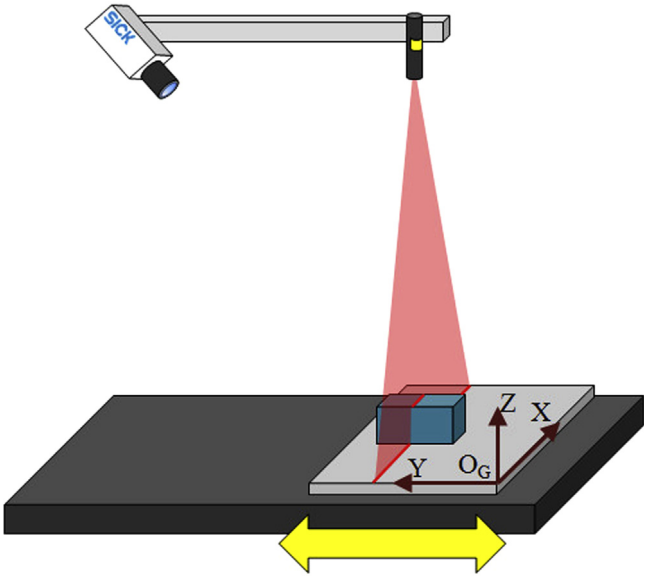

Fig. 2. Layout of the system.

3. 3D acquisition

3D scene acquisition is accomplished by using the high-speed 3D camera *Sick Ranger D50* and a laser diode at 660 nm, equipped with a cylindrical lens, angled by 45° with respect to the camera. As sketched in Fig. 2, it is mounted orthogonally to the working area and produces a very sharp line, in order to have a high contrast between the illuminated pixels and the rest of the image,

making the system insensitive to external light sources. The deformation induced by the shape of the illuminated surfaces onto the originally straight laser blade is captured by the camera; hardware designed on purpose allows the Ranger D50 camera to elaborate the light signal very fastly, and to provide the corresponding object profile in few milliseconds [22,9]. The whole scene is acquired through scanning: the Ranger D50 device acquires and elaborates the profiles into blocks of 512, at maximum speed equal to 1ms/profile. The 3D map is defined with respect to the so-called Global Reference (GR) system of coordinates (z, x, y), shown in the figure. The acquired profiles are defined in the (z, x) plane, being y the scanning direction.

3D data are sent to PC through a Gigabit Ethernet connection. The acquisition camera must be calibrated as a first step, to map the sensor coordinate system into real world coordinates. Purposely designed calibration software, called *Coordinator* is provided by *Sick*.

An example of the acquisition step is shown in Fig. 3. The field of view in Fig. 3a is equal to 120 mm by 120 mm; the scene is characterized by a number of different objects, disposed in a semi-random way. The measurement reference system is also drawn, with origin O_G in correspondence with the top-right corner of the black plane surface under the objects. The 3D raw point cloud obtained after the acquisition is presented in Fig. 3b: in this example, resolution along x and z is 6 points/mm and 0.06 mm respectively. The acquisition speed equals 5 ms/profile. The resolution along y is 10 profiles/mm. The whole point cloud is stored in a 1230 by 720 matrix. The shadows due to undercuts are well visible, as are the points belonging to the background surface. The 3D matrix encodes the shadows as invalid points; thus they are not of concern for the subsequent elaboration.

In contrast, points belonging to the x,y plane are a problem, since they would entail more computational effort in the next steps. To exclude them, a simple but efficient procedure has been developed. It consists into scanning the whole scene in the absence of the objects, immediately after the camera calibration. The resulting 3D map is obtained, and the maximum measured z_{max} value is found. This value is used as a threshold to distinguish the object points from the background in real scenes acquired subsequently.

As shown in Fig. 3b, the point cloud undergoes a translation along the y direction, with respect to the situation shown in Fig. 3a. This is a consequence of the fact that the calibration procedure does not estimate the coordinate of point O_G along y. This effect is compensated by precisely estimating the position, the velocity and the acceleration of the scanning stage at any time, and by measuring the cycle time between two subsequent scans. Fig. 3c shows the effect of these procedures: the 3D point cloud retains information only about the objects, and the origin is correctly set at point O_G.

4. Segmentation

The aim of segmentation is to assign each object to the corresponding sub-cloud. This task is not difficult when the objects in the scene are far enough from each other: this is the case of items labelled by O, G, A, and C in Fig. 3a. The corresponding sub-clouds in Fig. 3c are well separated from the neighbourhood, and a simple approach, based on the computation of the local density of the 3D map, might be very efficient to perform the operation. The problem becomes more difficult in the presence of objects very close to each other or even adjacent, such as the elements labelled by K and J in Fig. 3a. The corresponding point

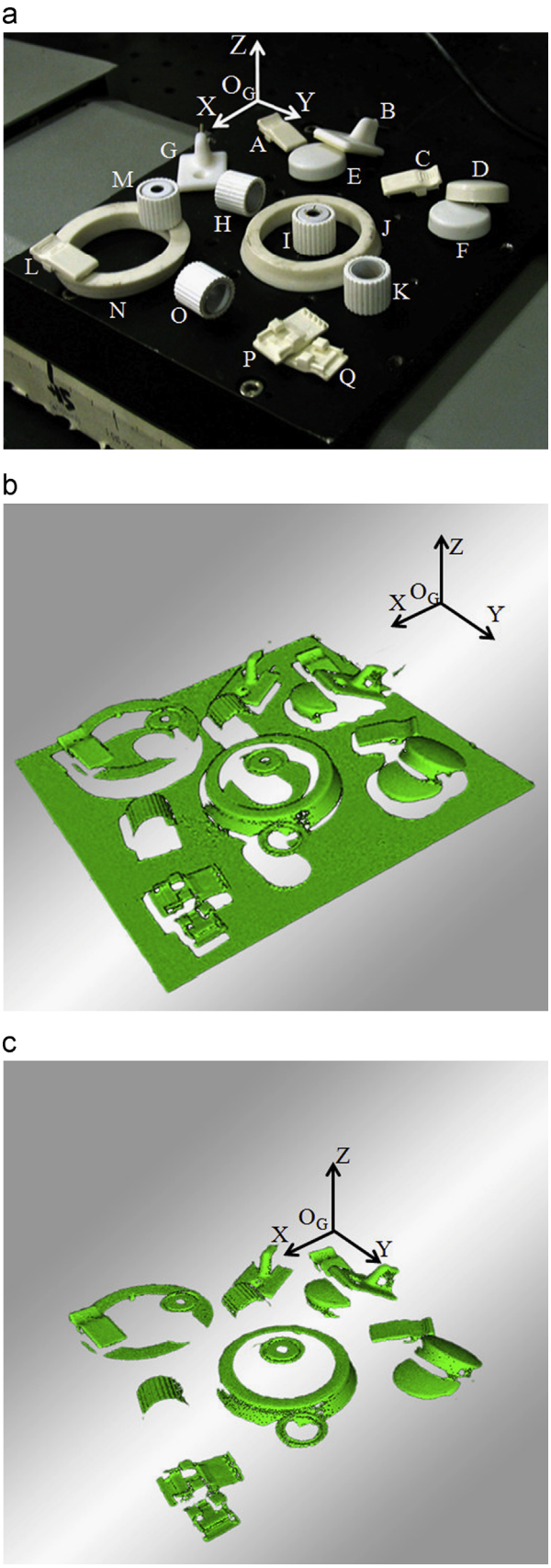

Fig. 3. Example of acquisition and 3D cloud generation. (a) Scene acquired by the 3D vision system. (b) Associated raw cloud. (c) Resulting cloud after the removal of points belonging to the plate and after the translation.

clouds in Fig. 3c are connected to each other, and cannot be distinguished as separate clouds by looking at their local density. The most demanding situation occurs when objects overlap, like M and L with respect to N, and for couples B–E and D–F in Fig. 3a. In these cases, not only the transition between the sub-clouds is not sharp, but at least one of the sub-clouds in the pair is incomplete.

To cope with these problems, a segmentation algorithm based on four steps has been developed. In the first one, called *Cloud filtering*, a dedicated procedure to filter out points belonging to the transitions areas among the objects has been designed, to enhance their separation in the point cloud. The second step, called *Cluster extraction*, implements a suitably designed algorithm able to find dense regions in the cloud, which are expected to identify distinct objects. The third step, called *Cluster selection*, is aimed at removing clusters corresponding to occluded objects. The last step, called *Cluster restoration*, compensates for unavoidable data losses caused by the preceding three steps.

4.1. Cloud filtering

This algorithm is based on the analysis of the neighbourhood of each point in the cloud, to find and remove points whose mean distance from their neighbours is greater than a given threshold. This algorithm is structured as follows:

(a) the neighbourhood of each point of the cloud is determined: given point q and the number of neighbours n, the goal is to find the points $\{p_1,\ldots,p_n\}$ in the cloud that are the closest to q. This is a typical application of the *nearest neighbours search* (*NNS*) problem, which is a hard problem and one of the most computationally expensive components in many computer vision algorithms. To cope with this aspect, a number of approximate methods have been developed; they are proven to be a good-enough approximation in practical applications and are orders of magnitude faster than the algorithms performing the exact search. The most widely used algorithm for nearest-neighbour search is the k–d tree, a space-partitioning data structure that stores a set of k-dimensional points in a tree structure [25]. In our work, we selected this algorithm to perform this computation; in particular, the *Fast Library for Approximate Nearest Neighbours (FLANN)*, which is part of the Point Cloud Library (PCL), was selected at the implementation level [26].

(b) Distances $\{d_i\}$ $(i=1,\ldots,n)$ among point q and points $\{p_1,\ldots,p_n\}$ and the mean value d_{mean} over values $\{d_i\}$ are computed;

(c) The distribution of distances d_{mean} obtained for each point q is assumed to be a Gaussian distribution, with mean μ and standard deviation σ. As an example, Fig. 4 shows the histogram of distances d_{mean}, obtained considering a neighbourhood of $n=20$ elements, for the cloud shown in Fig. 3c (the semi-logarithmic scale is used for clarity). The mean value μ of the distribution is estimated to correspond with the lowest d_{mean} value, and the standard deviation σ is chosen in correspondence with the 68.27% of the integral area of the histogram.

(d) This distribution gives an indication about the density of points in the cloud. In fact, points q with distance d_{mean} almost equal to μ are kept in the filtered point cloud P_f, as they belong to dense regions, whereas points q having d_{mean} greater than a given threshold are filtered out, as they correspond to sparse, if not empty, regions. The threshold is chosen equal to $d_{\text{ThF}}=\mu+\alpha\sigma$ where α, the standard deviation multiplier, is the degree of freedom of the algorithm: low values of α result into a strong separation among portions of points in the cloud, because a considerable number of points are filtered out. Referring to the example Fig. 4, it is evident that most of the points in the cloud belong to dense regions, because the histogram is compressed toward the mean, and the number of points decreases sharply for increasing values d_{mean}. The cloud resulting from the filtering process with $\alpha=0.05$ is shown in Fig. 5: less densely populated regions are removed and homogeneous groups of points are clearly separated.

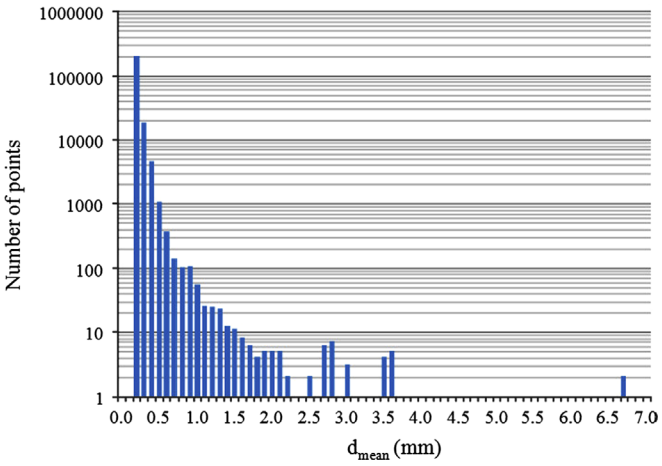

Fig. 4. Histogram of mean distances d_{mean} from 20 neighbors for the cloud shown in Fig. 3c.

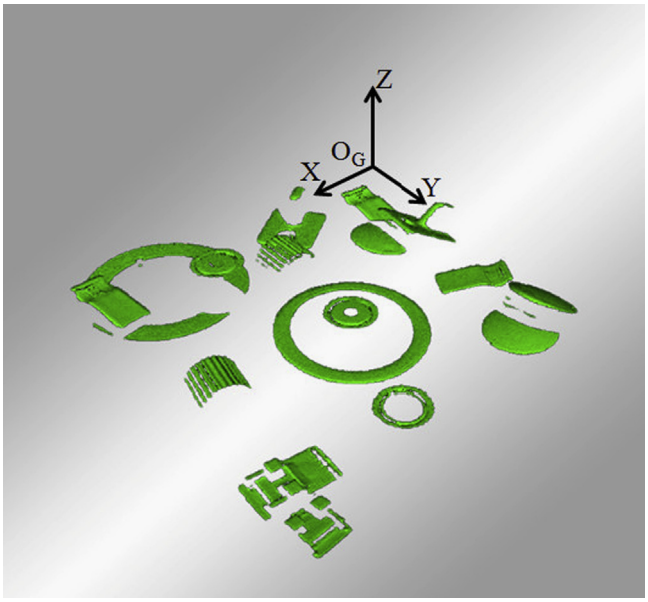

Fig. 5. Filtering procedure applied to the cloud shown in Fig. 3c.

4.2. Cluster extraction

The aim of this procedure is to decompose cloud P_f into clusters, i.e., sub-clouds that are supposed to identify distinct objects. From a mathematical viewpoint, two clusters, defined by sets of points $O_i=\{q_i\in P_f\}$ and $O_j=\{q_j\in P_f\}$, are distinct from each other if:

$$\min\|q_i-q_j\|_2\geq d_{ThE} \tag{1}$$

where d_{ThE} is the distance threshold. In other words, Eq. (1) states that if the minimum distance between the points in the set O_i and the points in the set O_j is larger than threshold d_{ThE}, then points q_i belong to cluster O_i and points q_j to cluster O_j. In our algorithm, the basic idea is to find the neighbours of each point q in cloud P_f, by

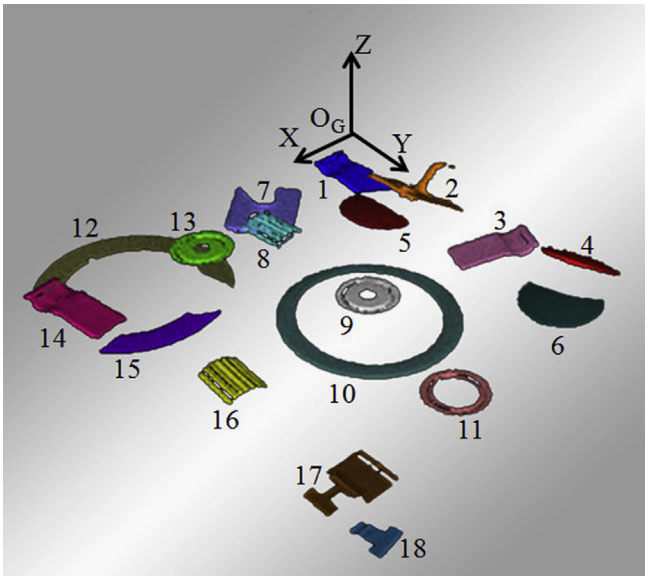

Fig. 6. Clustering algorithm applied to the 3D cloud in Fig. 5.

using a NNS approach similar to the one described in Section 4.1. However, in this case, the parameter that determines the neighbourhood is the distance threshold d_{ThE}.

The algorithm includes the following steps:

1. set up an empty list of clusters C and an empty queue of points Q;
2. for any point in P_f
   a) add point q_i to the current cluster Q;
   b) determine the neighbours $\{p_1,\ldots,p_n\}$ of q_i in a sphere with centre in q_i and radius $r < d_{ThE}$;
   c) add points $\{p_1,\ldots,p_n\}$ to Q, if they are not already present in the cluster;
   d) for any point of Q, repeat steps b and c;
   e) save Q in the list of clusters C;
   f) reset Q;
3. Go to step 2 and repeat until all the points in P_f have been processed.

Using this algorithm, each cluster is defined by points whose mutual distance is less or equal to radius r; steps from (b) to (d) are in a loop: the exit condition occurs when no more elements can be appended to the queue. This condition inherently occurs when the points at the boundaries of the cluster have been included. Subsequent clusters are detected by applying this searching algorithm to the *residual* points of the cloud. The degree of freedom of this procedure is the distance threshold d_{ThE}: if its value is too high, multiple objects can be grouped into a single cluster; on the other hand, every point of the cloud is assigned to a single cluster if the threshold value is by far lower than d_{ThE}.

Clusters with a very small number of points are removed as they are supposed to correspond to noise. Fig. 6 presents the result of this procedure applied to the 3D cloud of Fig. 5. Eighteen clusters, each showed in a different colour, have been detected. Referring to the objects in Fig. 3a, it is evident that most clusters (i.e., clusters 1–11, and 16–18) have been correctly assigned to a single object: the algorithm only fails in correspondence to object N, which has been assigned to two different clusters, (labelled by 12 and 15 in the figure) because of objects L and M, that stand on it. Objects E, F and Q, which are partially occluded, are assigned to clusters 5, 6 and 18; however, the corresponding point clouds are incomplete and any subsequent matching with the template representing the real object would fail, preventing the robot from correctly picking them up. For this reason, clusters corresponding to occluded objects must be detected. To this aim, a dedicated procedure has been developed. It is presented in the following section.

4.3. Cluster selection

Our method is based (i) on the calculation of the bounding box of each cluster, i.e., the smallest rectangular region in the *xy* plane that includes all the cluster points, and (ii) on the detection of overlaps between bounding boxes taken in pairs. Step (i) is very simple, since clusters share the same matrix which represents cloud P_f. Hence, denoting by Col and Row the matrix indexes, a simple sorting operation allows us to calculate, for each cluster, the highest values (maxCol, maxRow) and the lowest values (minCol, minRow) along columns and rows respectively of their bounding boxes. Step (ii) is performed by considering that, given two clusters Q_i and Q_j, there are sixteen overlapping combinations between the corresponding bounding boxes, schematically presented in Fig. 7. The conditions that must be checked to detect these combinations are shown in Table 1.

Whenever an overlap is detected, the occluded cluster has to be identified. To this aim, points in clusters Q_i and in Q_j, belonging to the region of overlap, are selected, and mean values $\overline{z}_i$ and $\overline{z}_j$ over corresponding depth values $\{z_i\}$ of Q_i and $\{z_j\}$ of Q_j are evaluated. If $\overline{z}_i > \overline{z}_j$, then Q_j is occluded by Q_i, if $\overline{z}_i < \overline{z}_j$ Q_j occludes Q_i. Occluded clusters are removed from the cluster list C. In case that the area of intersection does not include any of the two clusters, both remain in the list.

An example of the performance of this procedure is shown in Fig. 8: with respect to the situation in Fig. 6, clusters 5, 6, 12, 15 and 18 have been correctly excluded because they are occluded by clusters 2, 4, 13, 14 and 17, respectively. The elaboration of clusters 9 and 10 deserves a special comment: in this case, the corresponding bounding boxes do intersect (both conditions 2 and 3 in Table 1 hold); however, there are no points in cluster 10 that also belong to the intersection, and the algorithm correctly keeps both clusters in the list.

4.4. Cluster restoration

Cluster 7 in Fig. 8 gives a poor representation of object G: this is well evident by comparing the corresponding point cloud to the one in Fig. 3c. This behaviour is a consequence of the procedures presented in Sections 4.1 and 4.2. Matching this cluster with the template model, which represents the real object, might be unsuccessful. Cluster restoration deals with this problem: for each cluster in list C, the corresponding point cloud is replaced with the original one, which was acquired by the 3D scanning system: thus it can be more robustly aligned with the template.

5. Object identification

The core of this procedure is based on the *Match3D Coarse* function, belonging to the commercial suite of routines *SAL3D* (*3D Shape Analysis Library*), which contains a set of tools for the development of applications based on analysis and processing of range maps and point clouds.

Match3D Coarse follows a template matching approach. It provides three parameters. These are (i) the parameters of the 3D rigid transformation necessary to obtain the alignment, (ii) the Quality Factor (*QF*), which is a measure of how good the alignment is between the template and the cluster, and (iii) the disparity map (*DM*),

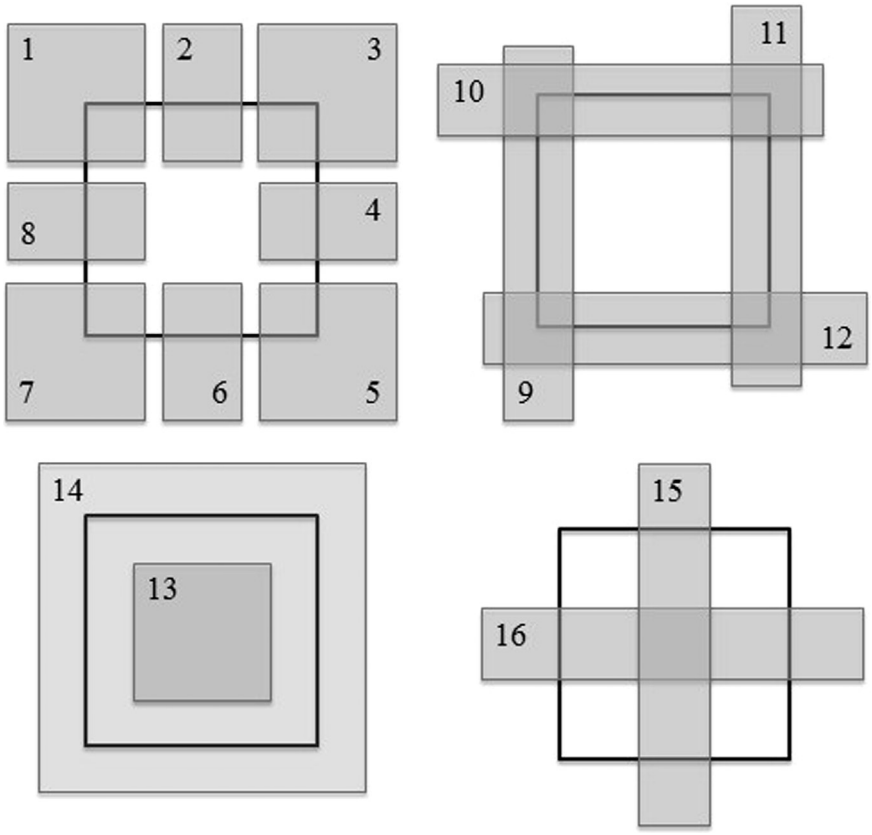

Fig. 7. Sketch of the possible overlaps between the bounding boxes of two clusters. Cluster Q_i is represented by the transparent rectangle, bold framed; clusters Q_j are represented by semi-transparent, grey rectangles.

Table 1
Conditions to be checked at the bounding boxes (first column) and corresponding overlaps (second column).

Checking conditions	Overlap combinations
$minCol_i \le minCol \le maxCol_i \wedge maxCol_j \le maxCol_i$	2, 6, 13, 15
$minCol_j \le minCol_i \le maxCol_j \wedge maxCol_i \le maxCol_j$	10, 12, 14, 16
$minCol_j \le minCol_i \le maxCol_j \wedge maxCol_i > maxCol_j$	1, 8, 7, 9
$minCol_i \le minCol_j \le maxCol_i \wedge maxCol_{ij} > maxCol_i$	3, 4, 5, 11

which stores disparities Δz of corresponding points in the two aligned surfaces.

The parameters of the 3D rigid transformation are expressed by the following matrix:

$$\mathbf{T} = \begin{bmatrix} & \mathbf{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

in Eq. (2), **R** is the 3×3 rotation matrix and **t**, of components T_x, T_y, T_z, is the 3D translation vector.

The *Match3D Coarse* tool is very robust against initial misalignment of the clouds; considering a typical situation, characterized by a number of point clouds corresponding to the objects in the work area, their position can be efficiently estimated by matching them to suitable templates, whose orientation in the GR system is known. Whenever a matching is found, the point cloud is assigned to the object identified by the template (object identification), and parameters in matrix **T** are used to tell the robot where the object is located and how it is oriented, for optimal picking.

This approach requires the creation of a database of 3D templates. Each template is a point cloud generated by acquiring the real object from a specific viewpoint, using the 3D scanning system in Fig. 2. The object shape determines the number of the templates corresponding to a single object; in addition, the templates of different objects can be stored, depending on the particular application. As an example, Fig. 9 shows the templates of the objects in Fig. 3a. Each point cloud is expressed in the GR system, with matrix **R** and vector **t** in Eq. (2) equal to the Identity matrix and to (0, 0, 0) respectively.

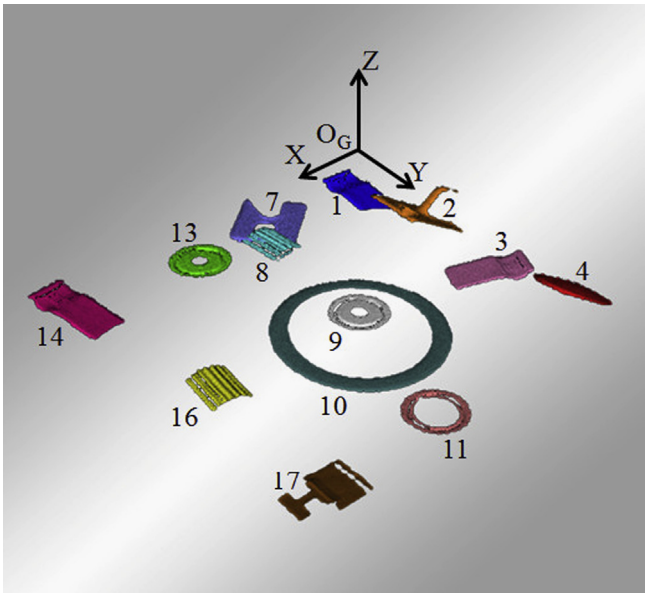

Fig. 8. Cluster selection method applied to the segmented point cloud in Fig. 6.

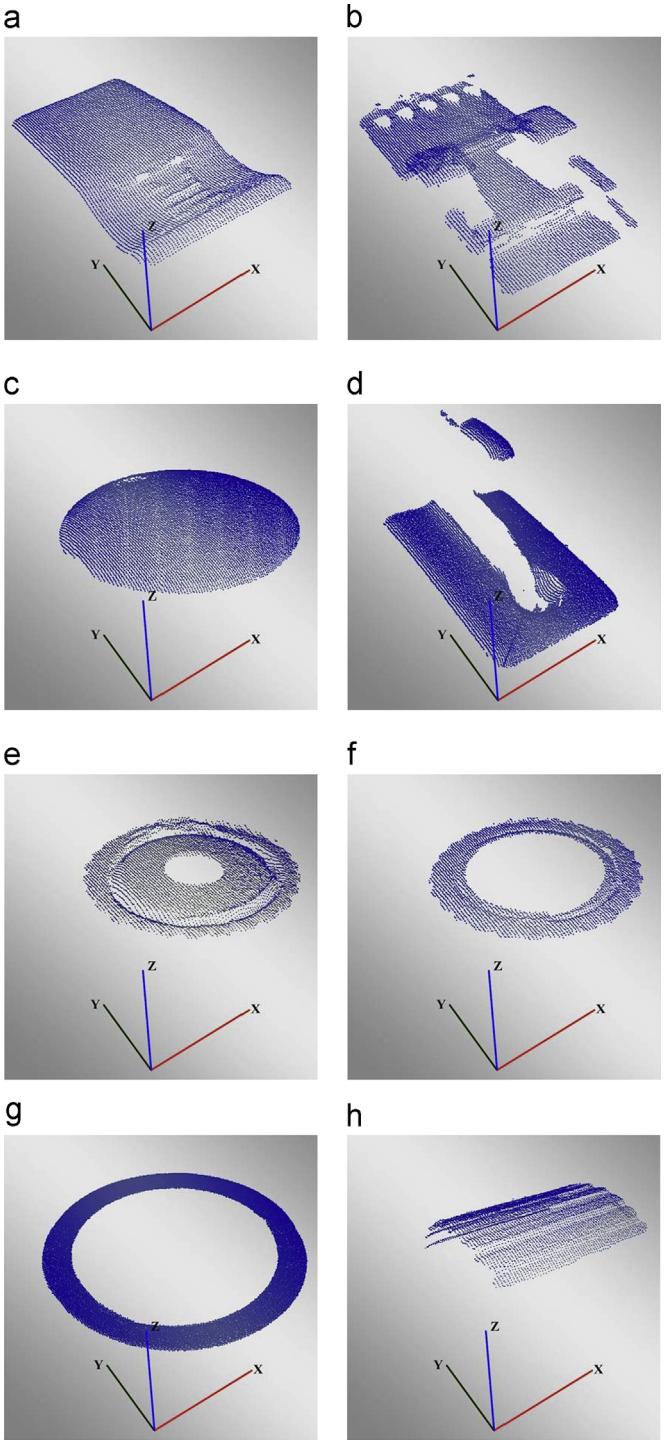

Fig. 9. Templates in the GR system, corresponding to the objects in Fig. 3a. (a) template corresponding with objects A and C; (b) template corresponding with objects P and Q; (c) template corresponding with objects D, F and E; (d) template corresponding with objects G and B; (e) template corresponding with objects I and M; (f) template corresponding with objects K; (g) template corresponding with object I; (h) template corresponding with H and O.

In our procedure, the *3D Match* tool is applied (on a pairwise basis) between each template *T* and each cluster *Q* belonging to list *C*. Correspondingly, the *QF* parameter is calculated. *QF* ranges between 0 (no matching) and 1 (perfect alignment), and is of help in all cases where a *single* template is matched to a *single* cluster; this typically occurs in quality control applications, where the goal is to check the tolerances of the work-pieces with respect to a

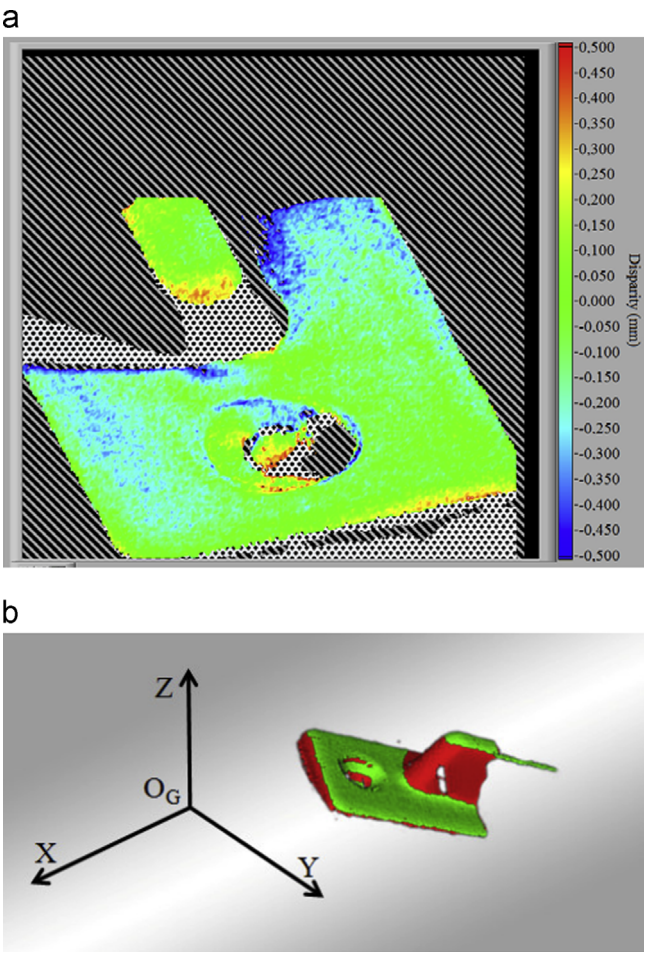

Fig. 10. Output of the Match 3D tool. (a) Disparity Map; (b) corresponding alignment.

model. However, in our application, the scenario is quite different, since the crucial point is to select the *best* template-cluster (*T*–*Q*) alignment, among all possible *T*–*Q* combinations. Very critical situations occur when (i) a single template matches with two or more clusters, or (ii) a single cluster matches with two or more templates. In these cases, the highest value of parameter *QF* represents a necessary but not sufficient condition for the best alignment.

5.1. Finding the best alignment

To remove any ambiguity of the alignment and to establish which template *best* represents a *given* object, the disparity map provided by each alignment is used. This map is a signed float matrix, storing the depth differences Δz for each point of coordinates (x,y) in the template and in the cluster. As an example, Fig. 10a shows the disparity map corresponding to the alignment of Fig. 10b, between the template in Fig. 9d and cluster 2 in Fig. 8. Disparities Δz are shown in mm on the right scale (a colour code is used for displaying); texturized areas are also visible in the map. The striped areas correspond to regions of the *DM* where neither the template, nor the cluster, is present; dotted areas correspond with points belonging to either the template or to the cluster. No valid Δz elements belong to texturized areas. The dimension along the rows and the columns of *DM* are *W* and *H* respectively.

In our approach, two new parameters are evaluated from each *DM* (i.e., for each alignment): the former is called *OF* (Overlap

Table 2
Performance of the procedure applied to the clusters in Fig. 8. T_x, T_y, T_z: components of vector **t**; D_{xy}: distance on the *x,y* plane; Roll, Pitch and Yaw: rotation angles estimated from matrix **R**.

Cluster	*OF* (%)	*EF* (mm)	T_x (mm)	T_y (mm)	T_z (mm)	D_{xy} (mm)	Roll (°)	Pitch (°)	Yaw (°)
1	90.6	0.041	10.82	20.12	2.09	22.84	−0.36	−1.08	−3.31
2	91	0.051	0.42	45.34	3.55	45.34	0.89	−29.71	−10.98
3	97.1	0.047	11.51	77.5	1.88	78.35	0.44	−0.31	−32.8
4	89.7	0.106	15.43	100.64	1.15	101.82	30.34	−18.41	−56.4
7	90.3	0.088	29.47	2.93	1.8	29.62	0.11	0.17	32.6
8	83.1	0.037	65.73	44.66	1.69	79.47	−0.27	−0.85	171.11
9	91.8	0.092	59.3	64.6	1.7	87.69	0.04	0.2	10.1
10	88.7	0.244	35.73	70.14	1.63	78.72	−0.4	−0.5	−40.9
11	93.6	0.164	69.61	105.31	1.1	126.24	3.65	1.83	−48.09
13	93.9	0.932	81.81	50.15	8.73	95.96	−1.57	−0.21	−132.5
14	96.4	0.03	103.51	16.88	7.73	104.88	−0.9	−4.51	11.24
16	87.4	0.057	110.81	73.41	1.56	132.92	−0.16	−0.15	163
17	71.9	0.176	115.89	93.15	5.18	148.69	−1.92	5.02	87.49

Factor), and is calculated as follows:

$$OF = 100 \times \left(1 - \frac{N_{Inf}}{(W \times H) - N_{NaN}}\right)\% \tag{3}$$

in Eq. (3), N_{Inf} and N_{NaN} are the number of points in *DM* that belong to the dotted and to the striped regions respectively; the higher *OF*, the better the intersection between the template and the cluster on plane *x,y*. When $OF = 100\%$, the template is perfectly overlapped to the cluster along *z*.

The latter parameter is called *EF* (Error Factor) and is defined as follows:

$$EF = \sqrt{\frac{\sum_{i=1}^{N_A} (\Delta z)^2}{N_A}} \tag{4}$$

in Eq. (4), N_A is the number of elements Δz in *DM*. Squared values of disparities Δz have been used, to emphasize large errors. The lower the *EF* value, the better the quality of the alignment.

Considering the set of templates and the disparity maps resulting from their alignment with a single cluster *Q*, parameters *OF* and *EF* are evaluated by means of Eqs. (3) and (4) and used to find the template that best aligns to cluster *Q* by means of the following algorithm:

(1) The subset *M* of templates for which is $OF \geq 70\%$ is determined;
(2) The template that shows the lowest value of *EF* among those of subset *M* is selected as the one which best aligns to cluster *Q*.

If subset *M* is empty, the sub-cloud corresponding to cluster *Q* cannot be assigned to any object, and the classification fails. In the other cases, the condition that the template and the cluster share a considerable subset of points must be satisfied at first. Then, the pair showing the best quality of the alignment is selected.

6. Experimental results

The implemented procedure has been tested to evaluate its performance. The first test deals with the object identification from the clusters in Fig. 8. For each cluster in Table 2 the values of parameters *OF* and *EF* corresponding to the best *T*–*Q* alignment are shown. Correspondingly, the parameters of the 3D rigid transformation are presented: T_x, T_y, T_z are the components of vector **t**, and angles Roll, Pitch and Yaw are derived from rotation matrix **R**. Parameter D_{xy} is evaluated as $D_{xy} = \sqrt{T_x^2 + T_y^2}$. A qualitative comparison of these values with the actual position of the objects in the scene in Fig. 3.a shows that vector **t** and matrix **R** do hold the required information to tell the robot where to move to pick objects up. For example, object A is the closest to origin O_G in GR; its distance D_{xy} from O_G equals 22.84 mm and is the lowest among D_{xy} values in the table. Values T_z of clusters 13 and 14 are the highest among values T_z in the table; this is consistent with the fact that they are placed on object N in the scene. As a last example, cluster 10 presents values of angles Roll and Pitch very close to zero, which is perfectly in accordance with the fact that it is placed on plane *x,y*. The value of angle Yaw is immaterial, due to its symmetry with respect to axis *z*.

The second test aims at quantitatively assessing the quality of the alignment. Fig. 11.a shows the experimental situation. The effect of the segmentation procedure is visible in Fig. 11b and in Fig. 11c. The cluster selection algorithm presented in Section 4 segments clusters from 1 to 4, which correspond with objects A–D at the first iteration. Clusters 5 and 6 are segmented at the second iteration, due to occlusions. To identify the objects, templates (e), (f) and (h) in Fig. 9 have been input to the *Match 3D* tool and aligned with clusters 1–4 first, and with clusters 5 and 6 afterwards. For each cluster, three values of parameters *OF* and *EF* have been obtained. The algorithm developed to choose the best *T*–*Q* alignment provides the values shown in Table 3.

To quantitatively evaluate the quality of the alignment, the following procedure has been implemented. Clusters in Fig. 11 have been given as inputs in the IMAlign software, belonging to the market available PolyWorks suite of programs [27]. This environment is one of the most powerful market-available products specifically designed for multi-view registration of 3D point clouds, mesh modelling, dimensional control and CAD applications.

The IMAlign module performs the pair-wise alignment between two point clouds defined in the same reference system. One point cloud is 'locked', i.e., its rotation matrix is set to the Identity matrix, and the translation vector is set to zero. The alignment algorithm, which is based on a semi-automatic, very sophisticated iterative closest point algorithm, provides the rotation matrix and the translation vector that allows the free cloud to align to the locked one [28]. The quantitative evaluation of the quality of the alignment is given by the Std.Dev parameter, which represents the standard deviation among corresponding points in the two clouds after the alignment.

In our test, clusters were locked, and templates were aligned to them, so that the alignment parameters output by the IMAlign module could be directly compared to the parameters estimated by our procedure. As an example, Fig. 12 shows the result of the alignment of the template in Fig. 9h to cluster 2 in Fig. 11b: the two point clouds are aligned very precisely as demonstrated by the value of the Std.Dev parameter which is 0.044 mm. This procedure was repeated for all the templates and all the clusters: the results

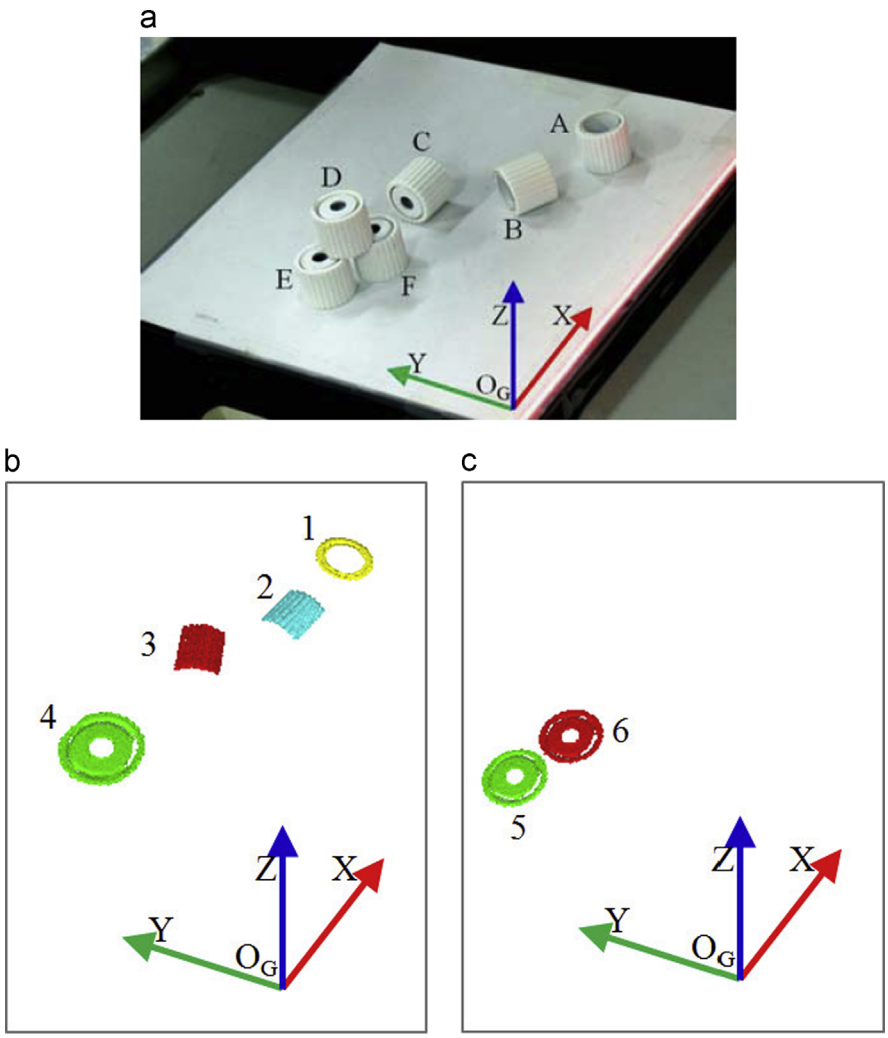

Fig. 11. Experimental test used to quantitatively assess the performance of the alignment.

Table 3
Performance of the implemented procedure applied to the scene in Fig. 11a. T_x, T_y, T_z: components of vector **t**; D_{xy}: distance on the x,y plane; Roll, Pitch and Yaw: rotation angles estimated from matrix **R**.

Cluster	*OF* (%)	*EF* (mm)	T_x (mm)	T_y (mm)	T_z (mm)	D_{xy} (mm)	Roll (°)	Pitch (°)	Yaw (°)
1	96.00	0.05	90.13	27.62	−0.76	94.27	4.61	−1.46	0.21
2	89.00	0.13	69.76	43.86	−1.27	82.40	24.70	−0.26	−26.17
3	84.00	0.10	55.25	54.57	0.07	77.66	−0.58	−1.19	−1.81
4	95.00	0.10	21.49	62.73	11.58	66.31	0.75	−0.23	−0.61
5	95.60	0.08	28.75	61.11	0.11	67.54	0.22	−0.43	−0.35
6	93.10	0.10	29.11	61.23	0.01	67.80	0.63	−0.64	−0.19

are shown in Table 4. For each cluster, the value of the Std.Dev parameter provided by the IMAlign module is reported: ΔT_x, ΔT_y, ΔT_z, ΔRoll, ΔPitch and ΔYaw are the differences between parameters T_x, T_y, T_z, Roll, Pitch and Yaw estimated by the IMAlign module and by our procedure respectively. The values shown in this table clearly confirm that the performance of the proposed procedure is suitable for bin picking applications.

As far as the elaboration time is concerned, the following considerations can be done: the acquisition of the 3D point cloud is very fast: the 3D Ranger D50 device is able to acquire one profile/ms. In realistic situations, where the work area might be up to 1×1 m^2, a 3D point cloud of 1230×780 can be acquired in one second, scanning at a speed of 1 m/s, with a resolution of 1 profile/mm, provided that the Region of Interest of the camera is reduced to 64 rows. This limitation can be removed using a more performing device, like the 3D Ranger E50. Resolutions along x and z are scaled proportionally depending on the optical layout.

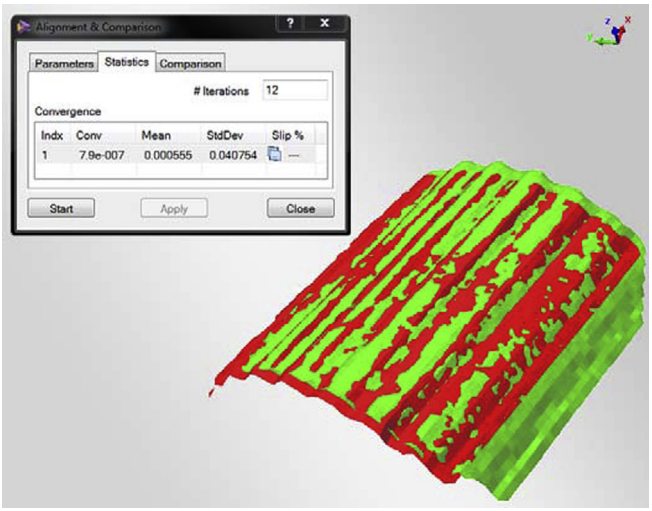

Fig. 12. Result of the pairwise alignment of the template in Fig. 9h to cluster 2 in Fig. 11b.

Table 4
Performance of the developed procedure. Cluster: the number of the clusters in Fig. 11b and c; Std.Dev: standard deviation parameter output by the IMAlign module; ΔT_x, ΔT_y, ΔT_z, ΔRoll, ΔPitch and ΔYaw: differences between parameters T_x, T_y, T_z, Roll, Pitch and Yaw estimated by the IMAlign module and by our procedure respectively.

Cluster	Std.DEV (mm)	ΔT_x (mm)	ΔT_y (mm)	ΔT_z (mm)	ΔRoll (°)	ΔPitch (°)	ΔYaw (°)
1	0.05	−1.10	1.12	−0.15	0.97	−0.45	−9.99
2	0.04	−1.46	6.12	0.50	−12.50	0.50	−2.02
3	0.03	0.12	−0.04	0.04	−0.06	0.93	3.89
4	0.07	−1.67	2.40	0.03	−0.78	−0.17	−19.39
5	0.08	−0.23	0.18	0.01	−0.28	0.05	−1.95
6	0.08	−0.63	0.11	0.11	−0.68	0.26	−2.52

The elaboration time required by the segmentation procedure mainly depends on the dimension of point cloud P_f. In our preliminary tests, where the filtered clouds had about 110,000 points, average values of 2 s, on a Intel Core i5 3350P hardware, have been observed.

The object identification procedure is the most demanding. In fact, the time required for a single *T*–*Q* alignment depends on both the template shape and the position of the template with respect to the cluster: in our tests, we observed that, on average, this time is 70 ms. The overall time depends on the number of *T*–*Q* alignments that must be carried out by the *Match 3D* tool. The most favourable situation (which in practice is also the most common) occurs when a single topology of objects is considered, like in the test presented in Fig. 11. In this case, only the templates in Fig. 9c, f and h have been matched to the four clusters in Fig. 11b at the first iteration, and to the two clusters in Fig. 11c at the second iteration, for an overall time of 0.84 ms and 0.42 ms respectively. In case the objects are of different topology, the time required to identify them increases: referring to the scene in Fig. 3a, the time required by the procedure is 7.8 s, since the number of matches is $13 \times 8 = 104$, being thirteen the clusters in Fig. 8, and eight the templates in Fig. 9.

7. Conclusions

A novel method for the identification of free-form objects in scenes affected by clutter and occlusions has been presented. It can be proposed as a satisfactory compromise between effectiveness and complexity for the bin picking problem.

The implemented method consists of three stages: the generation of the 3D cloud by means of triangulation-based laser scanning, the segmentation based on filtering and calculation of distance between point pairs, followed by a matching process realized starting from a commercial tool and a database of model clouds. We illustrated the features and the goal of these algorithms and we evaluated the performance generating random scenes with different types of objects.

The set of experiments demonstrated the accuracy and the reliability of the method for the identification and localization of randomly placed objects in the work area without prior assumptions about their shape and the need of neglecting the occluded ones. We are aware that the presented results are preliminary: a more complete assessment of the procedure is under development, using a robot gripper for picking-up objects randomly disposed in a bin. The performance of the whole system will be the subject of a subsequent paper.

Acknowledgments

The authors are grateful to Mr. Gabriele Coffetti for his continuous technical support during the development of this project.

References

[1] Sakakibara S. The robot cell as a re-configurable machining system. In: Dashchenko A, editor. Reconfigurable manufacturing systems and transformable factories. Berlin, Heidelberg: Springer Verlag; 2006. p. 259–72.
[2] Herakovic N. Robot vision in industrial assembly and quality control processes. In: Ude A, editor. Robot vision. InTech; 2010. p. 501–34.
[3] Hutchinson S, Hager GD, Corke PI. A tutorial on visual servo control. IEEE Trans Rob 1996;12:651–68.
[4] Tudorie CR. Different approaches in feeding of a flexible manufacturing cell. Lect Notes Comput Sci 2010;6472:509–20.
[5] Saxena A, Driemeyer J, Kearns J, Ng AY. Robotic grasping of novel objects. Neural Inf Process Syst. 2006:1209–16.
[6] Xie SQ, Cheng D, Wong S, Haemmerle E. Three-dimensional object recognition system for enhancing the intelligence of a KUKA robot. Int J Adv Manuf Technol 2008;38:822–39.
[7] van Dijck H, vdHeijden F. Object recognition with stereo vision and geometric hashing. Pattern Recognit Lett 2003;23:137–46.
[8] Kjellander JAP, Rahayem M. Planar segmentation of data from a laser profile scanner mounted on an industrial robot. Int J Adv Manuf Technol 2009;45:181–90.
[9] Blais F. Review of 20 years of range sensor development. J Electron Imaging 2004;13:231–40.
[10] Bellon ORP, Silva L. New improvements to range image segmentation by edge detection. IEEE Signal Process Lett 2002;9:43–5.
[11] Kirkegaard J, Moeslund TB. Bin-picking based on harmonic shape contexts and graph-based matching. In: Proceedings of the 18th international conference on pattern recognition (ICPR) 2 2006, pp. 581–584.
[12] Park IK, Germann M, Breitenstein MD, Pfister H. Fast and automatic object pose estimation for range images on the GPU. Mach Vision Appl 2010;21:749–66.
[13] Yang MY, Förstner W. Plane Detection in Point Cloud Data, ⟨http://www.journalogy.net/Publication/13234513⟩.
[14] Rahayem MR, Kjellander JAP. Quadric segmentation and fitting of data captured by a laser profile scanner mounted on an industrial robot. Int J Adv Manuf Technol 2011;52:155–69.
[15] Taylor G, Kleeman L. Robust range data segmentation using geometry primitives for robotic applications. In: Proceedings of the IASTED international conference on signal and image processing, 2003, pp. 467–472.
[16] Biegelbauer G, Vincze M, Wohlkinger W. Model-based 3D object detection. Efficient approach using superquadrics. Mach Vision Appl 2010;21:497–516.
[17] Johnson AE, Hebert M. Using spin-images for efficient object recognition in cluttered 3-D scenes. IEEE Trans Pattern Anal 1999;21:433–49.
[18] Mian AS, Bennamoun M, Owens R. Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans Pattern Anal 2006;28:1584–601.
[19] Del Bimbo A, Pala P. Content-based retrieval of 3D models. ACM Trans Multimedia Comput 2006;2:20–43.
[20] Dhome F. Real time robust template matching. In: Proceedings of BMVC 2002, British Machine vision conference, 2002, pp. 124–132.
[21] Sansoni G, Bellandi P, Docchio F. Combination of 2D and 3D vision systems into robotic cells for improved flexibility and performance. In: Proceedings of the fourth IEEE international workshop on advances in sensors and interfaces, 2011, pp. 22–30.
[22] ⟨http://www.sick-automation.ru/images/File/pdf/DIV01/ranger.pdf⟩.
[23] ⟨http://www.aqsense.com/products/sal3d⟩.
[24] Rusu RB, Cousins S 3D is here: Point Cloud Library (PCL). In: Proceedings of the IEEE international conference on robotics and automation, 2011, pp. 1–4.
[25] Moore A. An introductory tutorial on kd-trees. Computer Laboratory, University of Cambridge; 1991 (Technical Report no. 209).
[26] Muja M, Lowe DG. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: Proceedings of the international conference on computer vision theory and applications, 2009, pp.67–71.
[27] ⟨http://www.innovmetric.com/polyworks/3D-scanners⟩.
[28] Böhnke K, Gottscheber A. Fast object registration and robotic bin picking. In: Proceedings of the Eurobot conference, 2009, pp. 23–37.