A Software Architecture for Object Perception and Semantic Representation

Luca Buoncompagni and Fulvio Mastrogiovanni

University of Genoa
luca.buoncompagni@edu.unige.it, fulvio.mastrogiovanni@unige.it

Abstract. In the near future, robots are expected to exhibit advanced capabilities when interacting with humans. In order to purposely understand humans and frame their requests in the right context, one of the major requirement for robot design is to develop a knowledge representation structure able to provide sensory data with a proper semantic description. This paper describes a software architecture aimed at detecting geometrical properties of a scene using a RGB-D sensor, and then categorising the objects within to associate them with a proper semantic annotation. Preliminary experiments are reported using a Baxter robot endowed with a Kinect RGB-D sensor.

Keywords: Perception, Semantic knowledge, RGB-D sensor, Software architecture

1 Introduction

Advanced human-robot interaction processes in everyday environments are expected to pose a number of challenges to robot design, specifically as far as perception, knowledge representation and action are concerned. Examples where advanced capabilities in robot cognition play a central role include robot companions [9] and robot co-workers [4], just to name a few.

It is expected that a major role in robot cognitive capabilities for human-robot interaction will be played by a tight connection between robot perception processes and their semantic representation. The latter is expected to provide robot *percepts* with explicit contextual knowledge, which is implicitly assumed to be present when two humans interact and, after an adaptation process, reach a so-called *mutual understanding* state [11].

In the long-term, we consider human-robot interaction processes where a robot and a human share a workspace, and have to interact (physically, verbally or by means of gestures) in order to perform a certain joint operation. Examples of these processes include object handovers, joint manufacturing tasks, or typical household activities. In this paper, we focus on robot perception capabilities: a robot is able to detect and track the objects present in the shared workspace and, if they belong to known categories, to provide them with semantic meaning. To this aim, the proposed software architecture provides two main functionalities:

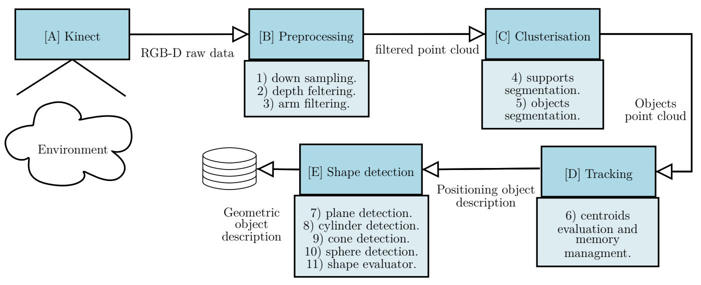

Fig. 1. Abstract representation of the system's architecture: components (blue squares) represent information sources and computational units, exchanged data (arrows) are related to the internal information flow.

- *Clustering, tracking and categorisation.* Starting from RGB-D data, the scene is processed to detect individual clusters in the point cloud. The position of each cluster, independently of its *shape* (in terms of the configuration of constituent points) is tracked over time. If a cluster can be mapped to a known basic geometric class (i.e., plane, cone, cylinder or sphere, a cube being given by six planes in a specific configuration), they are labelled accordingly using the Random Sample Consensus (RANSAC) algorithm [8].
- *Semantic description.* When object categories are determined, an ontology is dynamically updated to associate objects in the workspace (i.e., instances) with their semantic description (i.e., concepts). To this purpose, Description Logics (DLs) are used [1]. Once objects are classified, it is possible to tune robot behaviours accordingly, for example associating them with specific grasping behaviours, grounding them with proper verbal tags, using them in action planning processes.

The paper is organised as follows: Section 2 describes the proposed architecture; Section 3 discusses preliminary results; Conclusions follow.

2 System's Architecture

The proposed software architecture is based on the *computational* design pattern [2], i.e., a sequence of basic computational steps carried out in sequence according to a pipeline structure (Figure 1). First, a raw point cloud is acquired by the Kinect component ([A] in the Figure) and, for each scan, depth data are preprocessed ([B]). Then, the Clusterisation component ([C]) detects object supports (e.g., tables or shelves), and segments the objects above them, by generating a point cluster for each object. Such a *mid-level* representation is used by the Tracking component ([D]), which compares each cluster in the current point cloud with clusters already present in memory, updates their position and, if a new cluster is detected, registers it in memory. Finally, the Shape detection component ([E]) provides an estimate of the object basic primitive shape, as well as its parameters.

Algorithm 1: The *Tracking* component

```
Input: A vector $C$ of $n$ clusters belonging to the same support; a vector $T$ of $m$ tracked
       clusters; a matrix $D$ of $n \times m$ distances between current and tracked clusters; a
       vector $U$ of $m$ counters to check for how many scans a cluster has not been updated;
       a vector $F$ of $m$ Boolean values to keep track of which clusters have been updated.
Parameters: The radius $\epsilon \in \mathbb{R}$ and the threshold $\eta \in \mathbb{N}$.
1  For each $f_k \in F$, $f_k \leftarrow false$
2  foreach $c_i \in C$ do
3      For each $i = 1, \dots, n$ and $j = 1, \dots, m$, $D_{i,j} \leftarrow \infty$
4      foreach $t_j \in T$ do
5          $d_{i,j} \leftarrow dist(c_i, t_j)$
6          if $d_{i,j} < \epsilon$ then
7              $D_{i,j} \leftarrow d_{i,j}$
8      if $\exists D_{i,j}$ such that $D_{i,j} = \infty$ then
9          create $t_k \in T$ using $c_i$
10         add and initialise $u_k \in U$ such that $u_k \leftarrow 0$, $f_k \in U$ such that $f_k \leftarrow 0$
11     else
12         $d_o = d^*_{i,j} \leftarrow argmin_{i,j}(D)$.
13         if $f_o = false$ then
14             update centroid and point cloud of $t_o$ using weighted average between $c_i$ and $t_j$
15             $f_o \leftarrow true$
16             $u_o \leftarrow 0$
17 foreach $f_k \in F$ such that $f_k = false$ do
18     $u_k \leftarrow u_k + 1$
19     if $u_k > \eta$ then
20         delete $t_k \in T$, $u_k \in U$, $f_k \in F$
```

The *Preprocessing* component performs a sequence of steps sequentially. First, a downsampling step is carried out to decrease the number of points in the point cloud provided by the RGB-D sensor [10]. Then, vectors normal to all the surfaces are computed [6]. The data is limited to all the points belonging to a semi-sphere around the Kinect-centred reference frame, which allows for focusing on a particular area of the workspace thereby drastically improving the overall system performance. Finally, the point cloud is also filtered to remove the points related to the robot's arms. This is done by enveloping the links related to robot arms in bounding boxes, and checking for a point-in-parallelepiped inclusion.

The *Clustering* component recursively applies RANSAC to find all the horizontal planes in the scene. This information is used to determine the points belonging to the objects located on planes (i.e., acting as *supports*). Finally, an Euclidean clustering algorithm is applied to segment the objects in the scene [7]. As a result, the component generates for each support a set of clusters related to objects located above it. Each cluster i is represented by its *visual centroid* $c_i^v = (x_i^v, y_i^v, z_i^v)$, computed as the mean of all the points in i.

Although many approaches to obtain a robust tracking are available (see the work in [3] and the references therein), in this preliminary work we adopted a simple geometrical approach. Our aim is to obtain and evaluate a hybrid geometric/symbolic representation of objects. Previously detected objects are stored in memory, specifically using their visual centroid and an associated list of cloud points. Our current implementation of the *Tracking* component is depicted

%	*Empty*	*Plane*	*Sphere*	*Cone*	*Cylinder*
Empty	87.40	11.65	0.20	1.06	1.60
Plane	12.50	82.54	6.63	7.22	8.70
Sphere	0.00	0.00	93.17	0.00	0.00
Cone	0.00	0.32	0.00	66.04	0.11
Cylinder	0.10	5.49	0.00	25.68	89.59

Table 1. Confusion matrix describing the error distribution of the RANSAC-based shape detection.

in Algorithm 1. After an initialisation phase (lines 1-10), first an association between current and tracked clusters is performed (lines 11-16), then old clusters are removed (lines 17-20). Given two clusters i and j detected at time instants t_1 and t_2, we refer to their visual centroids as $c_i^v(t_1)$ and $c_j^v(t_2)$. We assume that j is an updated representation of i if $c_j^v(t_2)$ is located within a sphere of radius ϵ centred on $c_i^v(t_1)$. A tracked cluster is removed from memory if it is not updated for η consecutively scans.

Finally, the *Shape detection* component associates each cluster with a possible primitive shape (i.e., plane, cone, cylinder or sphere, a cube being given by six planes in a specific configuration), as well as its geometrical coefficients. To this aim, we employ RANSAC to find the best fitting shape based on the relative number of points belonging to those primitives. Once a cluster i is associated with a primitive shape, its representation can be *augmented* with a shape tag (i.e., a category), its coefficients (e.g., the axis for cones or the radius for cylinders), and the *geometrical centroid* c_i^g, which is computed using the primitive shape rather than the point cloud. It is noteworthy that, in principle, c^g is more accurate than c^v, since it considers not only the visible part of the object but its full reconstruction provided by RANSAC.

Currently, knowledge about primitive shapes is maintained within an OWL-based ontology [5], where all the geometric object properties can be described. Two classes are used to model objects, namely VisualObj and GeomObj. The former models objects in the form of clusters, whereas the latter represents the associated primitive shape. GeomObj has a number of disjoint subclasses related to primitive shapes, including SphereObj, ConeObj, CylinderObj, and PlaneObj. Two data properties are used to describe visual and geometric centroids, namely hasVisualCen and hasGeomCen, as well as properties to describe shape-specific coefficients, e.g., a SphereObj has a radius specified using hasGeomRadius. As a consequence, a description corresponding to a cluster i is an instance of VisualObj if its property hasVisualCen contains a valid visual Centroid c_i^v, and it does not contain any valid description related to hasGeomCen. In formulas: $\mathsf{VisualObj} \sqsubseteq \exists \mathsf{hasVisualCen.Centroid} \sqcap \neg \exists \mathsf{hasVisualCen.Centroid}$. A similar description holds for VisualObj.

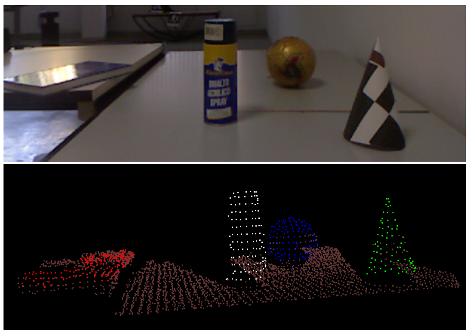

Fig. 2. Top: Experiment set up (top) and data visualization (bottom), where colours identify shapes and supports.

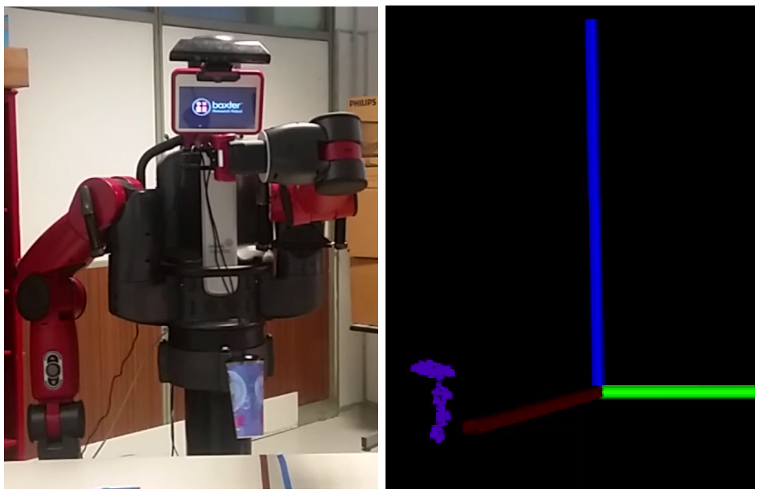

Fig. 3. Baxter holding an oscillating cone.

3 Experimental Results

Two experiments have been set-up, the first aimed at evaluating the performance of the architecture in static conditions, the second on a set-up involving a Baxter robot. The system has been implemented in ROS and the Point Cloud Library (PCL) [7].

The first experiment is aimed at estimating the errors in shape detection in a static environment with multiple supports (Figure 2). Acquisition is made 500 times for every shape. Results are reported in the confusion matrix shown in Table 1. It is possible to see that system's performance is reliable specifically for planes and spheres. Slightly lower recognition scores are present for cones and cylinders.

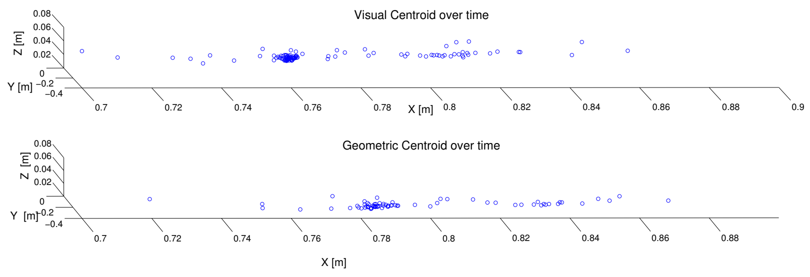

Fig. 4. Visualisation of c^v (top) and c^g (bottom) during a tracking experiment.

The second experiment focuses on the performance of the tracker as well as the visual and geometrical centroids (Figure 3 on the left hand side). A cone has been fixed to the Baxter's end-effector through a wire, in order to mimic a pendulum-like behaviour. When the robot moves the arm, the cone oscillates. The wire's length and the cone's mass are unknown to the robot. Figure 3 on the right hand side shows the tracked point cloud. It is noteworthy that the cluster does not represent the object completely, which affects the visual centroid, whereas the geometrical representation of the object allows for computing a more accurate centroid. Figure 4 shows the tracking of the two centroids. Intuitively, it can be noticed that the variance associated with c^v is higher than that of c^g, since the visible part of the object changes while the object oscillates. Moreover, it can be observed that between the two plots there is an offset due to the geometric properties of the real object and its visible part.

4 Conclusions

This paper describes an architecture to model and track a few (both geometrical and semantic) properties of objects located on a table. The system is still work-in-progress. One the one hand, we are interested in exploring the possibility of using symbolic-level information to model high-level object features, such as affordances. On the other hand, we believe that the interplay between the two representation levels can be exploited to increase the overall system's capabilities.

References

1. Baader, F., Galvanise, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic handbook: theory, implementation, and applications. Cambridge University Press, Cambridge, MA (2007)
2. Brugali, D.: Software Engineering for Experimental Robotics. Springer, Heidelberg, Germany (2007)
3. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D mapping with an RGB-D camera. IEEE Transactions on Robotics 30(1), 177–187 (2013)

4. Haddadin, S., Suppa, M., Fuchs, S., Bodenmuller, T., Albu-Schäffer, A., Hirzinger, G.: Towards the robotic co-worker. In: Proceedings of the 2009 International Symposium on Robotics Research (ISRR 2009). Lucerne, Switzerland (September 2009)
5. McGuinness, D.L., Harmelen, F.V.: OWL web ontology language overview. W3C Recommendation (2004-03) (2004)
6. Rusu, R.: Semantic 3D object maps for everyday robot manipulation. Springer, Heidelberg, Germany (2013)
7. Rusu, R., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA 2011). Shanghai, China (May 2011)
8. Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum 26(2), 214–226 (2007)
9. Walters, M.L., Syrdal, D.S., Dautenhahn, K., I. Boekhorst, K.L.K.: Avoiding the uncanny valley: robot appearance, personality and consistency of behaviours in an attention-seeking home scenario for a robot companion. Autonomous Robots 24(2), 159–178 (2008)
10. Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J.: Kintinuous: spatially extended KinectFusion. In: Proceedings of the 2012 RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras. Sydney, Australia (July 2012)
11. Zlotowski, J.A., Sumioka, H., Nishio, S., Glas, D.F., Bartneck, C., Ishiguro, H.: Persistence of the uncanny valley: the influence of repeated interactions and a robot's attitude on its perception. Frontiers in Psychology 6, 883 (2015)