Generalized Framework for the Parallel Semantic Segmentation of Multiple Objects and Posterior Manipulation

Adrian Llopart, Ole Ravn, Nils A. Andersen
Automation and Control Group
Department of Electrical Engineering
Technical University of Denmark
Lyngby, Denmark
Email: adllo@elektro.dtu.dk, or@elektro.dtu.dk, naa@elektro.dtu.dk

Jong-Hwan Kim, *Fellow IEEE*
Robot Intelligence Technology Laboratory
Department of Electrical Engineering
KAIST
Daejeon, South Korea
Email: johkim@rit.kaist.ac.kr

***Abstract*—The end-to-end approach presented in this paper deals with the recognition, detection, segmentation and grasping of objects, assuming no prior knowledge of the environment nor objects. The contributions of the paper are as follows: 1) Usage of a trained Convolutional Neural Net (CNN) that recognizes up to 80 different classes of objects in real time and generates bounding boxes around them. 2) An algorithm to derive in parallel the pointclouds of said regions of interest (*RoI*). 3) Eight different segmentation methods to remove background data and noise from the pointclouds and obtain a precise result of the semantically segmented objects. 4) Registration of the objects' pointclouds over time to generate the best possible model. 5) Utilization of an algorithm to detect an array of grasping positions and orientations based on the geometry of the objects model. 6) Implementation of the system on the humanoid robot *MyBot*, developed at the RIT Lab at KAIST. 7) An algorithm to find the bounding box of the objects model in 3D to then create a *collision object* and add it to the octomap. The collision checking between robots hand and the object is removed to allow grasping using the *MoveIt* libraries. 8) Selection of the best grasping pose for a certain object, plus execution of the grasping movement. 9) Retrieval of the object and moving it to a desired final position.**

Keywords - object detection, convolutional neural nets, semantic segmentation, pointcloud processing, grasping, humanoid robot

I. Introduction

The usage of robots in human environments has steadily gained popularity in the last years. With rapidly growing research in fields such as service and social robotics, the demand has risen for robots that can understand the environment around them to be able to interact with it. Namely, it has become essential that a wide variety of objects are recognized and manipulated in real-time.

The proposed approach focusses on the perceptive elements required for a robot to find an object and grab it. Therefore, this paper presents an end-to-end generalized framework for object handling. It starts with the recognition and detection of classes of objects in the RGB domain using a CNN, followed by the their segmentation in the spatial domain using point clouds, and ending with the actual planning and manipulation of said objects using a humanoid robot. The framework is a continuation of the work presented in [1]. No prior knowledge of the environment nor the object's model is required.

A Convolutional Neural Network, trained on the COCO dataset, is used to derive bounding boxes around 80 different classes in a live video stream obtained by a robot (III). The resulting *RoI* are then applied to the rectified depth images to generate pointclouds of the detected objects (IV-A). These result however include certain noise and irrelevant background data. Thus, they must be segmented. Eight different methods will be applied, depending on the class itself, to be able to extract the best possible representation of the object (IV-B). This process is done online and requires no human intervention. At any point in time, a user can select which class the robot has to interact with. The segmented pointclouds of the specified class will be registered over time to generate a precise and robust model of the object by removing any possible noise and multi viewpoint errors (V-B). Then, an array of grasping poses will be derived based on the geometry of the constructed model. The dimension of the robots hand are also taken into consideration (V-C). To allow grasping, the object's form must be removed from the continuously updated octomap generated by the robot. To achieve this, a 3 dimensional bounding box is found around the registered object's pointcloud and a *collision object*, from the *MoveIt* libraries, is created. Collision checking between fingers and object is then disabled to allow for the grasping (VI-B). The best grasping pose out of the whole array is chosen (which must be achievable) and a trajectory is planned. Since the humanoid robot has two arms, either can be used to reach the target pose, usually the one which has its hand closer to the object. However if the selected arm fails the grasp, the other one will be used. Finally, a humanoid robot moves its arms to reach the goal pose, the object is grasped and placed in a desired position (VI-C).

The experimental results are presented in VII, followed by concluding remarks in VIII and possible future improvements in IX.

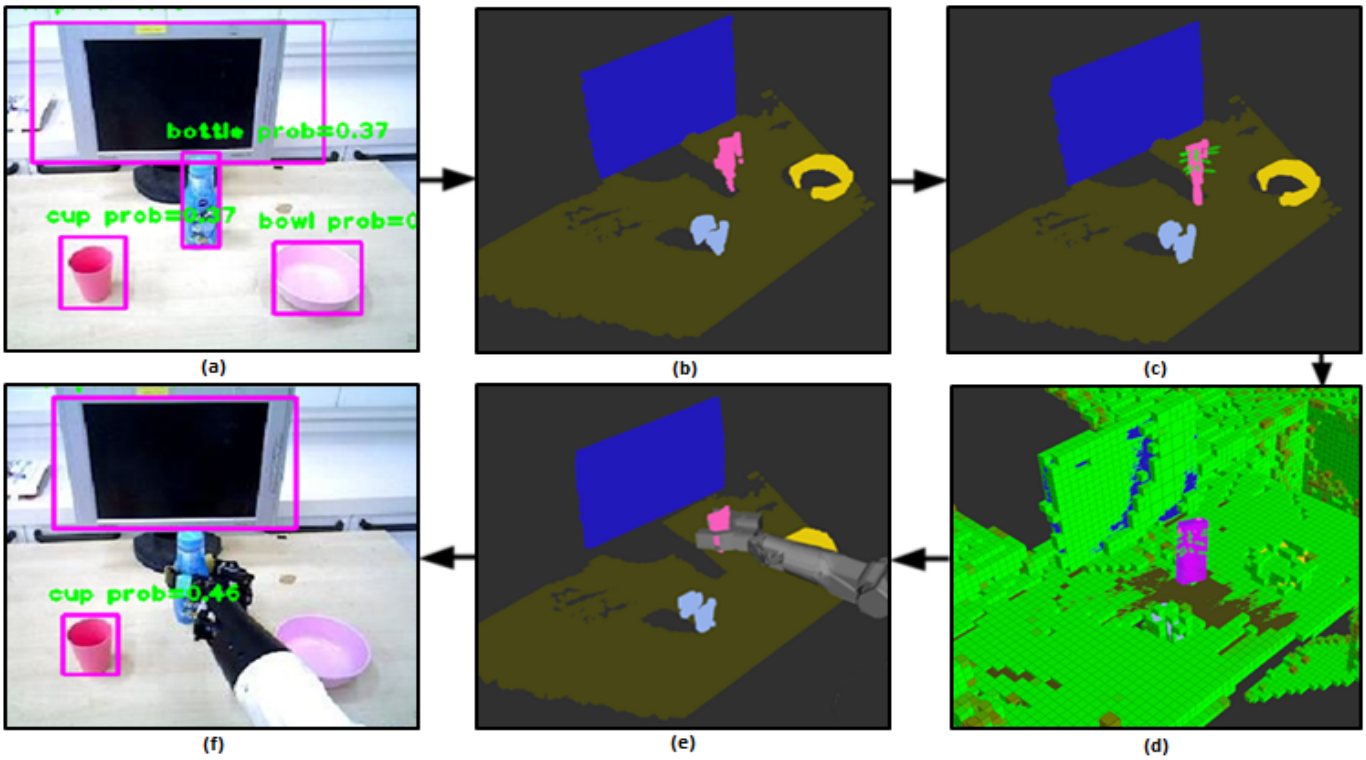

Fig. 1: End-to-end pipeline of the presented approach: **(a)** Object detection using CNN **(b)** Segmented objects **(c)** Array of grasp poses **(d)** Collision object and octomap **(e)** Planning to target pose **(f)** Execution of movement

II. Related Work

One of the presented pipeline's strengths is that it requires no prior knowledge of the environment nor the class models to be segmented. This contrasts with other *state-of-the-art* approaches which utilize geometric verifications to match precomputed models and grasping poses of the objects onto the scene ([2], [3], [4], [5]). Namely, they require offline meshes of all objects; which means that the approach is not generalizable.

Multi-viewpoint of a scene, as opposed to one single view, helps improve robustness of objects models hypotheses ([3], [6], [7] and [8]) and becomes essential when applying SLAM-techniques

Asif *et. al.* [9] presents a novel method, *STEM*, which encodes the appearance and structural characteristics of an RGB-D point cloud in terms of five feature maps. It allows for RGB-D object recognition and grasp detection using hierarchical cascaded forests, without predefined object models.

III. CNN for object recognition and detection

The pipeline presented in this paper initiates with the usage of a Convolution Neural Network to recognize and detect specific pre-trained classes from images. The goal is to extract *RoI's*, in real time, from a live video stream coming from the robot. Hence, limiting the data into useful class information.

The proposed CNN must be able to predict bounding boxes around classes based on their probability extremely quick and with a high degree of reliability and precision. This optimization is achieved by employing a unified architecture: the YOLO Real-Time Object Detection System ([10], [11]).

It is important that model is accurate, real-time and able to detect multiple objects simultaneously. To attain these goals, the *yolo-coco* model is used. It has been trained on the COCO dataset provided by Microsoft, which includes 80 different classes. However, in this paper special emphasis is put on kitchenware, food and office supplies.

A great benefit of this system is that it is easily integrated with *ROS*. On a GeForce GTX 1070 it can run at an average of 16 FPS. The result given by the detection system includes not only the bounding boxes around the diverse classes, but also the probabilities for each one. Some results can be seen in Fig. 1, Fig. 2, Fig. 7 and Fig. 8.

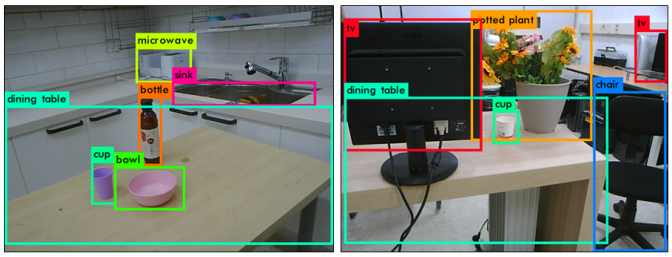

Fig. 2: Results after CNN is applied to images collected by *Mybot*.

IV. Semantic segmentation of objects

Once the classes have been recognized and detected in the colour domain, the knowledge learnt (i.e. the semantic labelling of objects) must be transferred into the spatial domain. In other words, segment 3 dimensional representations of the objects.

A. Generation of pointclouds based on Region of Interest

Initially, the meta data (location and size) of the *ROI's* found by the CNN is transmitted to the depth domain. This can be done because the RGB and Depth images have been previously rectified, using *ROS Indigo* libraries, and match pixel-to-pixel. By removing all data from a depth image which is not inside the *RoI*, performance of the system and quality of the results are substantially increased. Pointclouds of each object are created based on the cropped depth images; one pointcloud for every object detected. These pointclouds will encode not only the typical data necessary in a pointcloud message, but also the label of the object it represents. The new message type will be called *LabelledPointcloud*.

B. Multiple segmentation methods

Even though the objects have been generated in the spatial domain, they still render useless for a correct interaction with the robot. The reason is that these pointclouds generally include noise and background data. Therefore, to obtain the best estimate of the object, these pointclouds must be segmented.

A total of 8 different segmentation methods are employed in this paper. They have been applied using *PCL* libraries. The algorithms are:

1) Planar segmentation
2) Planar segmentation (only vertical surfaces)
3) Planar segmentation (only horizontal surfaces)
4) Outlier segmentation from planes
5) Cylindrical segmentation
6) Spherical segmentation
7) Region growth segmentation
8) No segmentation

The name of the methods does indeed describe very clearly what they do. For instance, the planar segmentation finds all the points within a point cloud that support a plane model. The second and third methods restrict the found planes to only vertical and horizontal surfaces, respectively. Additionally, the vertical planar segmentation method includes also a handle segmentation because some objects that fall into this category require it (for instance refrigerators), similarly to [1]. To achieve this, only points that fulfill the conditions of not being inliers in the plane, being located between the plane and the robot (i.e. not behind the plane) and being less than 10 cm to the plane, will be segmented as handles.

The fourth method, goes one step further as the first one. It is used to segment objects that are found always on top of desks or other surfaces (also known as *tabletop* objects). The method finds planes but only stores the remaining points (*outliers*). This method has been enhanced to save only points located between robot and plane, eliminating everything else.

In cylindrical and spherical segmentation, cylinder and sphere models are fitted respectively into the data. The last method, region growth, the points that are close enough in terms of the smoothness constraint are merged forming multiple clusters. The largest cluster will be selected as the best descriptor of the object.

The decision of which algorithm has to be applied to which object is done *a priori* following logical criterion. Special emphasis is put on the fact that this is the only part of the whole pipeline that has been hard coded by the authors. Objects such as bottles and cups will obtain better results using a cylindrical segmentation. Monitors, laptops and refrigerators should use the vertical planar segmentation method. Sports balls, oranges and apples work best with spherical segmentation. Objects with no predefined form, like animals or backpacks will be segmented using region growth. And this is done for each of the 80 classes the CNN is able to recognize.

Hence, given an incoming *LabelledPointcloud* message, the label is checked and the correct segmentation method is selected. The result is a second *LabelledPointcloud* message which differs from the first in that all noise and unnecessary background data has been removed. Results can bee seen in Fig. 1 , Fig. 4, Fig. 6(b) and Fig. 8.

It is worth mentioning too that the segmentation of pointclouds is done in parallel thanks to the *ROS* interface. Individual nodes have been created for every distinct segmentation method to allow parallel processing, therefore the results update at a much faster rate, allowing real time processing.

V. Derivation of Grasping poses

The interaction between robot and objects is one of the key features presented in this paper. As aforementioned, several objects can now be detected and segmented correctly from the environment, which opens the possibility of them being manipulated by a robot. To achieve this, an array of correct and precise grasping poses (positions and orientations) must be obtained so the robot can grab the object and move it around. These poses will be found depending on: **(1).** the geometry of a model of the object, which is created by registering segmented pointclouds of the object over time, and **(2).** the form of the robots hand.

A. User input for the selection of object

The first step to interacting with an object is, indeed, selecting it. In the presented approach, this decision is made by the user, with a simple command stating the name of a class. This can be done at any point in time since the recognition and segmentation of classes is done continuously. However, in future iterations of the proposed pipeline, this decision might be made by the robot itself using artificial intelligence methods.

Fig. 3: Registration over time of different pointclouds (grey) representing a cup. The result appears in red. Its growth is clearly visible over time as more and more points become registered.

B. Registration of pointclouds to create model of the object

After the user chooses a specific class to handle, its grasping poses must be found. These will be highly dependant on the geometry of the objects model, which must be extremely precise. Since there is no prior knowledge of the objects form, it is essential to create the model online based on the segmented pointclouds.

The major issue with the segmented pointclouds is that they vary greatly depending on the viewpoint of the robot; additionally, they sometimes include noise. Thus, the results may vary from one iteration to another. This problem can easily be turned into an advantage by registering the pointcloud over a certain amount of time. This will allow for the removal of noise and the generation of a model based on pointclouds from multiple viewpoints. In other words, the more the robot looks at and object, and from different positions, the better and more definite model of it is achieved.

The registration will be accomplished using the Iterative Closest Point (ICP) algorithm. Pointclouds will only be registered if the transformation between one and another is smaller than a certain threshold. Due to this, noise is easily removed and, most importantly, pointclouds that have been wrongly segmented will be discarded. If the difference between the current registered pointcloud and following new points is too big, the registering process is halted and restarted.

All in all, the resulting pointcloud will increase over time as more and more correct points are registered, creating a robust and precise model of the object. The registration process can be visualized in Fig. 3. Additional results for the registered pointcloud can be seen as red pointclouds in Fig. 4 and inside the generated *collision object* in Fig. 6(d).

C. Array of grasping poses based on geometry of the object

Much emphasis has been put on creating the best possible model of the object because the grasping poses will be highly dependant on its geometry. This is due to the fact that the *agile_grasp* package [12] is used. Moreover, the algorithm also takes into account the dimensions of the robotic hand.

The usage of this approach is fundamental for the presented framework. One of the key concepts shared by both projects is that there must be no prior knowledge of the manipulated objects, as opposed to other *state-of-the-art* methods that include offline models of the objects and predefined grasping positions (i.e. [3] and [6]). The grasping algorithm [12] has two steps. The former, sampling a large set of grasping hypothesis based on the condition that for a grasp to exist, the hand must be collision-free and part of the objects surface must be contained between the fingers. The latter, using machine learning (Support Vector Machine) to classify the validity of those hypothesis checking whether the grasping points are *antipodal*. Thus, Pas *et. al.* [12] report an average grasp success rate in their experiments of 88% when grasping novels objects presented in isolation and 73% when presented in dense clutter. Even though the methodology has been developed for 1 DoF , two finger hands, the results are reliable enough to still use this method with a 3DoF, 3 fingered hand with which *Mybot* is equipped (VI-A).

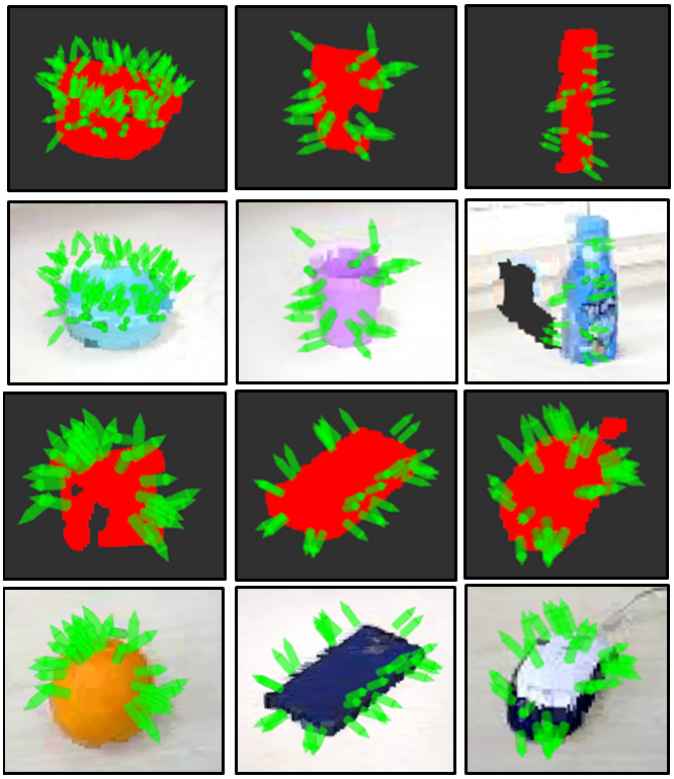

Fig. 4: Array of grasp poses (green vectors) derived from the registered pointclouds of objects (red). These results are also shown overlapped on top of the original coloured pointclouds: cup, bowl, bottle, orange, cell phone and mouse, respectively

Some results of good grasping positions for diverse objects can be seen mainly in Fig. 4, but also in Fig. 1(c).

VI. Implementation on humanoid robot

The proposed framework has been carried out on the humanoid robot *MyBot*. The pipeline has been expanded (as shown in sections VI-B and VI-C) to be able to port the entire system onto the robot, namely recognition, detection, segmentation and manipulation of objects.

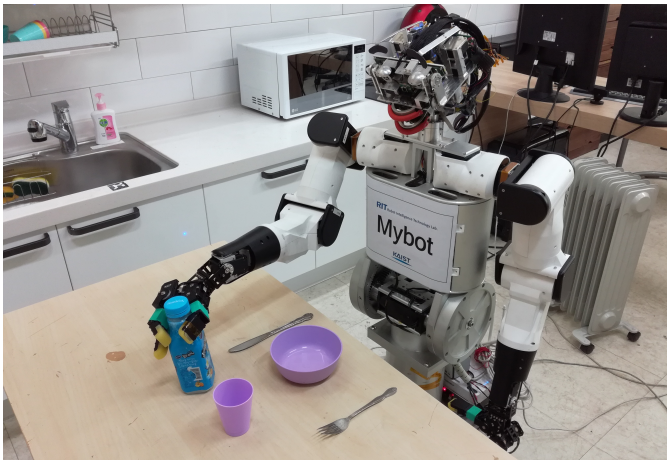

Fig. 5: *MyBot* humanoid robot grasping a cup.

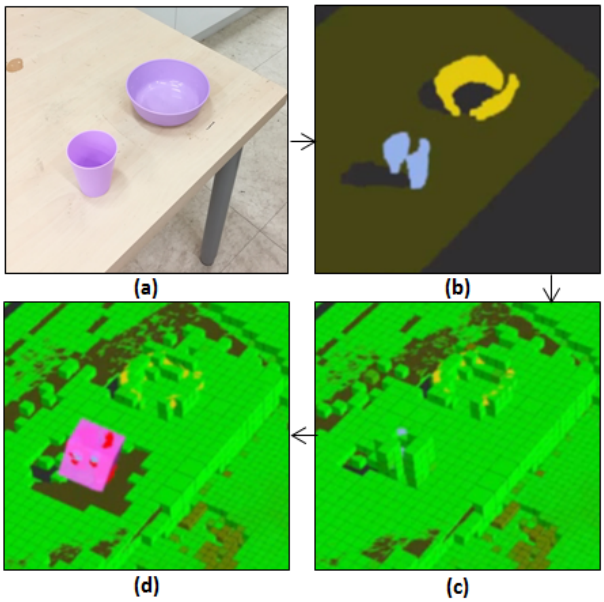

Fig. 6: Creation of a collision object based on the 3D bounding box around the registered pointcloud (red): **(a)** Original RGB image **(b)** Segmented objects **(c)** Current state of the Octomap **(d)** Generated collision object (pink) removes points from octomap

A. Description of the robot

MyBot is a robot developed in the Robot Intelligence Technology (RIT) Laboratory at KAIST. It makes use of an Odroid XU board, Ubuntu 14.04 and *ROS Indigo*. A *Xtion ASUS* sensor is employed to obtain rectified RGB and Depth images of the surroundings. It is located on the head of the robot, which includes 2 *DoF* to look around. The robot has a mobile platform for navigation and two 7 *DoF* arms with 3 finger hands. Additionally, the torso of the robot includes another 2 *DoF* for panning and tilting.

B. Removal of collision checking between fingers and object

An octomap is a 3D occupancy grid mapping approach that models environments into occupied, free or unknown areas. This becomes essential for robot navigation, that is, knowing where certain actions can take place and where the robot will surely collide with obstacles.

A problem of this approach is that the segmented objects will have surely been encoded as occupied space. This means that the system might fail at planing movements to the grasping positions since collisions may appear between fingers of the robot and objects whilst moving. Additionally, the system will certainly fail when grasping the object because grasping obviously requires contact.

The way to solve this is to remove the collision checking between object and fingers. This can be done only because the proposed approach [12] minimizes contacts during movement through an optimal grasping pose array and the *MoveIt* libraries. To remove collision checking, a *collision object* must be created first around the object. It could be created from an objects mesh, but there is no prior knowledge of it. Thus the only solution is to create it from primitive shapes, specifically rectangular prisms.

The proposed contribution is as follows: since the geometry of the object is mostly known thanks to the registered model of the object, a 3D bounding box can be generated around it. Firstly, the centroid, the normalized covariance and the eigenvectors of the registered pointcloud are found; these represent the principal directions. Then all points are moved onto the found reference frame. The maximum and minimum point values for every axis can be easily found: these represent the width, height and depth of the bounding box. All results are finally converted back to the original frame.

The values obtained from the bounding box are used to create a rectangular prism *collision object*, which removes all points inside it from the octomap. This can be seen in Fig. 1(d) and Fig. 6(d). Lastly, the collision checking between fingers and the newly created *collision object* is removed, which allows the robot to plan a trajectory to the object and finally grab it.

C. Manipulation of object using MoveIt libraries

Interaction between segmented objects and the robot is the final step of the framework and, most importantly, requires all previous sections to have worked correctly. At the end of the day, what truly matters is that when the user states an object's name, the robot is able to, autonomously, find it in a cluttered new environment and be able to pick it up.

Since the robot has two arms, it is important to select carefully which one to manipulate the objects with before even starting to plan a trajectory. Naturally, the hand that is already closer to the object should be the one used because it will require less planning time. For this reason, the positions of all the found poses in the grasp array are averaged and the distance between that result and the reference frame of both hands is calculated based on the 3D Pythagorean theorem. The hand with the shortest distance will be used to grab the object.

In spite of this, sometimes, selecting the closest hand to an object might not result in the optimal solution. Especially

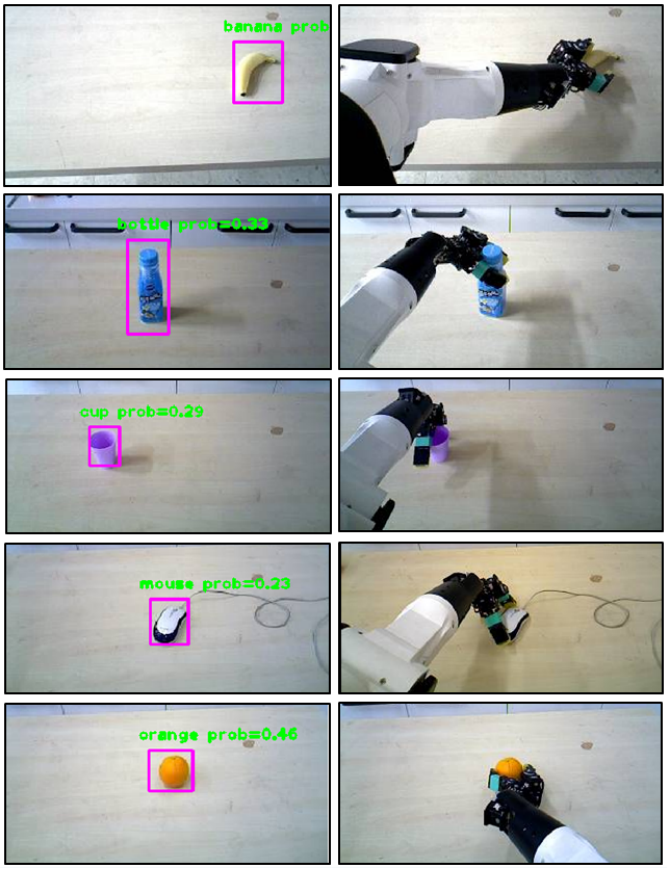

Fig. 7: *Left column:* Results from CNN applied to several classes. *Right column:* Grasping poses achieved by the humanoid robot.

if there are obstacles to avoid between the end and goal poses. This is why, if the planning with the first arm fails, the second arm is utilized. Generally, one of the plans will succeed; however, if both of them do fail, a new perception iteration takes place. Namely, registration of pointclouds, grasp derivation, generation of collision object and planning. This occurs until the object has finally been grabbed or the user halts the process. Further robot cognition could be added at this point to allow more autonomy and better results, like the one presented by *Veiga et al.* [13]. Another example of this is changing the robots position with respect to the object if the trajectory planning has failed several times and then approach it from a completely different perspective.

Assuming the planning to the grasping pose has succeeded, the robot will execute the movement thanks to the *MoveIt* libraries. This leads to the robots gripper being positioned right next to the object. The last step is to simply close the gripper and move the object to a desired position or perform a specified task. The created *collision object* is attached to the robot so that when moving the object around, it never crashes into obstacles. The end-to-end perceptive and manipulative framework has, therefore, finished. Various grasping poses, achieved using one or the other arm, for different objects can be seen mainly in Fig. 7 and, alternatively, in Fig. 1(f).

	Class	Trials	Recognition (%)	Grasp (%)
Single	Banana	10	100	20
	Bottle	10	100	90
	Bowl	10	80	20
	Cup	10	90	80
	Mouse	10	70	50
	Orange	10	90	80
Clutter	Bottle	10	100	80
	Cup	10	100	70
	Orange	10	80	70
	Chair	10	90	-
	Monitor	10	100	-
	Refrigerator	10	90	-
Average			**90.1**	**62.2**

TABLE I: Results for the recognition and grasping blocks for 9 different classes, both in single-object and cluttered scenes.

VII. Experimental Results

The proposed framework has been tested in real life indoor scenarios, that is a kitchen and an office. The conducted experiments are divided into two blocks: the former, the object perception block, which includes detection and registration of objects via CNN and the generation and segmentation of the corresponding pointclouds. The latter, the grasping block, includes the pointcloud registration of a specified object, the derivation of an array of grasping positions, the creation of a *collision object* based on the 3D bounding box around the registered pointcloud, the selection of which arm to use and, finally, the execution of the grasping trajectory with a humanoid robot.

A. Experimental Setup

A correct object recognition occurs when the objects bounding box is precisely found in the RGB domain, with a proper semantic labelling. This must be followed by a strict segmentation of at least half of the object. Similarly, a good grasping of the desired object occurs when the objects geometry is correctly found using pointcloud registration, a legitimate *collision object* is generated, several grasping poses are derived and the robot plans a trajectory to one of them to ultimately grab the object. All steps in each block must be fulfilled for the test to be considered positively.

The conducted tests are divided into 2 types: single object scenes and cluttered scenes. The reason is to test how highly dense scenes affect both the perceptions of the objects and the manipulation. The classes used in this experiment are also divided into two groups: those that can be both recognised and grasped (cup, bottle, bowl, banana, orange, mouse), and those that can only be recognised (chair, monitor, refrigerator).

For single object perception, the relative position between robot and objects will be changed randomly after every trial. This will help evaluate the system in front of different viewpoints and trajectory distance until grasp. Equivalently, in cluttered scenes, multiple objects will be arbitrarily relocated in the scene. However, special care will be taken so that the goal object is still visible, at least half of it, so that occlusions do not affect the results.

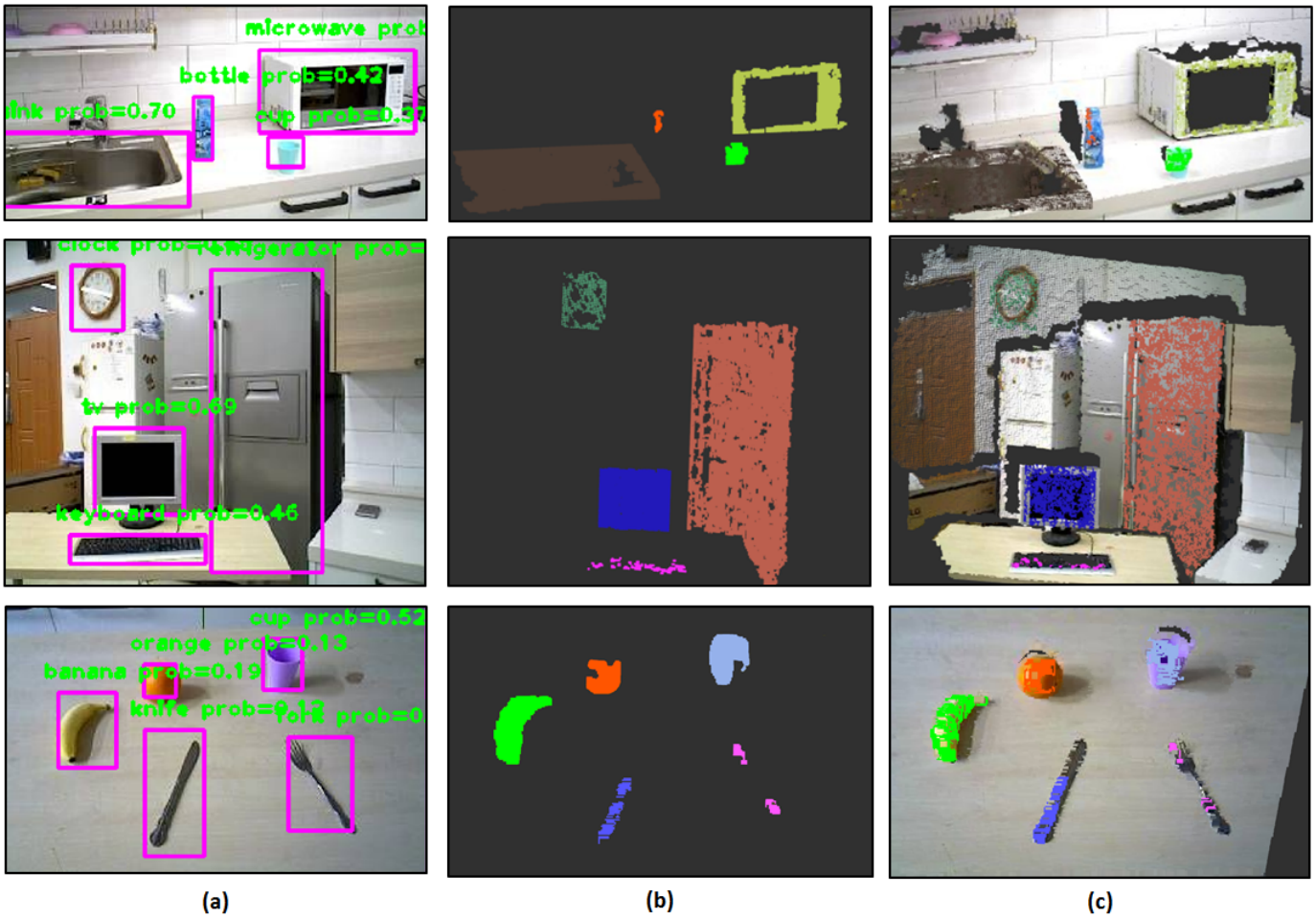

Fig. 8: Experimental results: **(a)** Object detection using CNN **(b)**, Segmented objects **(c)**, Overlap over original pointcloud

B. Results, causes and consequences

The most important conclusion to be extracted from the experiments is that the recognition block achieves highly positive results. With a **90.1%** of correctly recognized, detected and segmented objects from the COCO Dataset, the approach presented in this paper is proven to be competitive in front of other *state-of-the-art* methods. Comparatively, Asif *et al.* [9], using their STEM methodology, achieves results for object recognition of 93.0% and 91.9% grasp success rate.

Concerning the grasping block, the results (**62.2%** grasp success rate) show that some objects can be handled quite easily, whilst others tend to cause more failures. There are 3 main reasons for this: goal position and orientation errors, slippery surfaces and collisions prior to the grasp.

Firstly, if the final grasping pose is not precise enough due to noise in the registration process, then even when the robot reaches the goal position, it will end up either grasping thin air or grabbing inconsistently a small part of the object. Similarly, if the orientation for grasping is not reliable enough, the fingers of the robot might en up touching the object when moving to the goal position and pushing it out of the way. A clear example of this is when trying to grasp cups and bowls from the top. For it to work, the thumb has to end up on one side of the vertical surface of the object, whilst the rest of fingers have to be placed on the opposite side. if the orientation is not precise enough, the fingers will collide with the top of the object and the grasp will be unsuccessful.

The second problem for grasping failures is the lack of friction between fingers of the robot and certain objects. If the surface is smooth, the robot will commonly grab the objects but it will just slip away when lifting it. This becomes evident with objects with polished surfaces such as mouses or bowls. In addition, these objects have convex surfaces which makes the whole process of grabbing them even more difficult.

The last issue when grasping objects is small collisions occur between the fingers and the environment before grasping. This problem becomes evident when dealing with objects that have small grasping surfaces. For instance, bananas and mouses can be handled only in parts of their surface that are really close to the table they usually are located on. The experiments show that even when the robot correctly positions the hand around the object, when trying to grab it, the fingers collide with the table, blocking the grasp.

VIII. Conclusion

The presented framework includes the whole pipeline for object manipulation with a humanoid robot: from the detection of objects in the RGB domain, to their segmentation and, finally, their manipulation.

The approach differs from many *state-of-the-art* due to the fact that no prior knowledge of the environment or objects is required. This means that no pre-computed models and grasping poses are known. This is precisely why the proposed methodology can be applied to multiple objects (which, however must fall into one of the 80 classes recognizable by the CNN) independently of their size, form or colour. The system can be greatly expanded by applying, at the beginning of the pipeline, a larger CNN capable of detecting even more classes. The usage of several segmentation methods and the registration over time of the objects pointcloud, allows building a consistent model of the unknown object. As aforementioned, the grasping poses are determined based on the online generated geometry of the object. The downside of this is that the grasps will be of a poorer quality than if thy were pre-selected. In spite of this, the approach has the important upside that it is generalizable in front of any object's geometry, making it very powerful when dealing with unknown objects.

The experimental results have proved that the system works consistently from end-to-end. The recognition block achieves highly competitive results whilst the grasping block obtains good results, specially considering the lack of prior knowledge of the objects, Despite this, the grasp pose estimation could be improved using alternative methods. It is worth mentioning too that trying to grasp objects with a simpler shape and larger surface, such as boxes, than the proposed objects would entail an increase in the successful grasping trials. This is due to finding more robust grasping poses and the non-existance of slippery surfaces and collision with the environment. This is seen in the fact that cups and bottles (cylinders) and oranges (sphere) have higher success rates than the rest of objects.

All in all, the final outcome is a system that allows a humanoid robot to find certain classes in an unexplored environment, segment the objects from the scene, calculate correct poses and, finally, manipulate them.

IX. Future improvements

Even though the presented framework achieves good results considering its performance in front of novel objects and surroundings, there is always room for improvement. Right now the system can detect up to 80 classes which is a good number but not large enough for a real life implementation. *Yolo9000* [11] is a CNN model trained to recognize and detect up to 9000 different classes with real time performance. It would be a very nice addition to the presented pipeline. It would surely mean including many more segmentation methods (like the one presented in [14]) or combinations of them to be able to obtain adequate results for the large number of new classes (e.g. colour-based region growth segmentation).

At the moment, the selection of the object to grasp is done via user input. This could change with some sort of robot cognition which selects objects based on the future task to be executed, as proposed by [13]. Additionally, when registering the selected objects pointcloud, much better results would be achieved with a multi-viewpoint perspective since a more reliable model would be generated and so more consistent grasping poses would be found (as presented in [3], [6], [7] and [8]). Similarly, if planning to a grasping pose has failed several times, approaching the object from a new perspective might solve the problem. These methods require some level of artificial intelligence.

A force-torque sensor could be added to the fingers to be able to know precisely if the grasp has failed or, otherwise, how good the quality of the grasp is. Finally, SLAM techniques and loop closure could be added to generate full semantic maps over time, as shown in [2] and [6].

References

[1] A. Llopart, O. Ravn, and N. Andersen, "Door and Cabinet Recognition Using Convolutional Neural Nets and Real-time Method Fusion for Handle Detection and Grasping," in *IEEE International Conference on Control, Automation and Robotics (ICCAR)*, Nagoya, Japan, April 2017.

[2] K. Tateno, N. Navab, and F. Tombari, "When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 16-21 2016.

[3] C. Li, H. Xiao, K. Tateno, F. Tombari, N. Navab, and G. Hager, "Incremental Scene Understanding on Dense SLAM," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, October 9-14 2016.

[4] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. D. Stefano, and M. Vincze, "Multimodal Cue Integration through Hypotheses Verification for RGB-D Object Recognition and 6DOF Pose Estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6-10 2013.

[5] S. Gupta, P. Arbelaez, R. Girshick, and J. Malik, "Aligning 3D Models to RGB-D Images of Cluttered Scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, June 7-12 2015.

[6] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks," [ArXiv]. Arxiv, 7 pp., 7 pp.

[7] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense slam," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sept 28 - Oct 14 2015.

[8] A. Djelouah, J. Franco, and E. Boyer, "Multi-View Object Segmentation in Space and Time," in *IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, Sydney, Australia, December 3-6 2013.

[9] U. Asif, M. Bennamoun, and F. Sohel, "RGB-D Object Recognition and Grasp Detection Using Hierarchical Cascaded Forests," *IEEE Transactions on Robotics*, vol. PP, no. 99, pp. 1–18, January 2017.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Boston, USA, p. 779788, June 2016.

[11] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," Dec. 2016, arXiv:1612.08242.

[12] A. Pas and R. Platt, "Using Geometry to Detect Grasp Poses in 3D Point Clouds," in *Proc. International Symposium on Robotics Research (ISRR)*, Genova, Italy, September 2015.

[13] T. Veiga, P. Miraldo, R. Ventura, and P. Lima, "Efficient Object Search for Mobile Robots in Dynamic Environments: Semantic Map as an Input for the Decision Maker," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, October 9-14 2016.

[14] U. Asif, M. Bennamoun, and F. Sohel, "Unsupervised segmentation of unknown objects in complex environments," in *Autonomous Robots*. SPRINGER, 2015, ch. 40 (5), p. 805829.