Sapienza
Università di Roma

Department of Computer, Control and Management Engineering Antonio Ruberti

Interactively picking real world objects with unconstrained natural language instructions in the ARM Challenge

Elective in Artificial Intelligence (HRI & RA)

Master program in Artificial Intelligence and Robotics

Professors:	**Students:**	
Luca Iocchi	Dennis Rotondi	1834864
Fabio Patrizi	Fabio Scaparro	1834913
Giuseppe De Giacomo	Marco Montagna	1882418

Academic Year 2022/2023

Contents

1 Introduction

As we journey further into the 21st century, the once fantastical idea of interacting with robots is swiftly transitioning from the pages of science fiction into our everyday reality. The pace of this evolution is accelerating far beyond what many could have predicted even a few years ago. Propelling this forward momentum are increasingly frequent competitions that focus on the intersection of artificial intelligence (AI) and robotics. These events are not just a testament to the blossoming era of AI and robotics, but also the pivotal role they play in pushing the boundaries of what's technically possible.

1.1 ARM Challenge

The inspiration for this project was drawn from one of the most famous robotics competitions, the ARM Challenge within RoboCup 2023. This challenge engages participants in executing a recycling task using a robotic manipulator, within a dynamic environment (Figure 1). Skills like robotic manipulation and grasping, crucial to many robotic applications, are put to the test. The challenge serves as a platform for RoboCup participants and student teams to research, develop and showcase their proficiency in manipulation perception and control in a simulated setting. These skills are later applicable to commercial robots. Participants are expected to design and implement

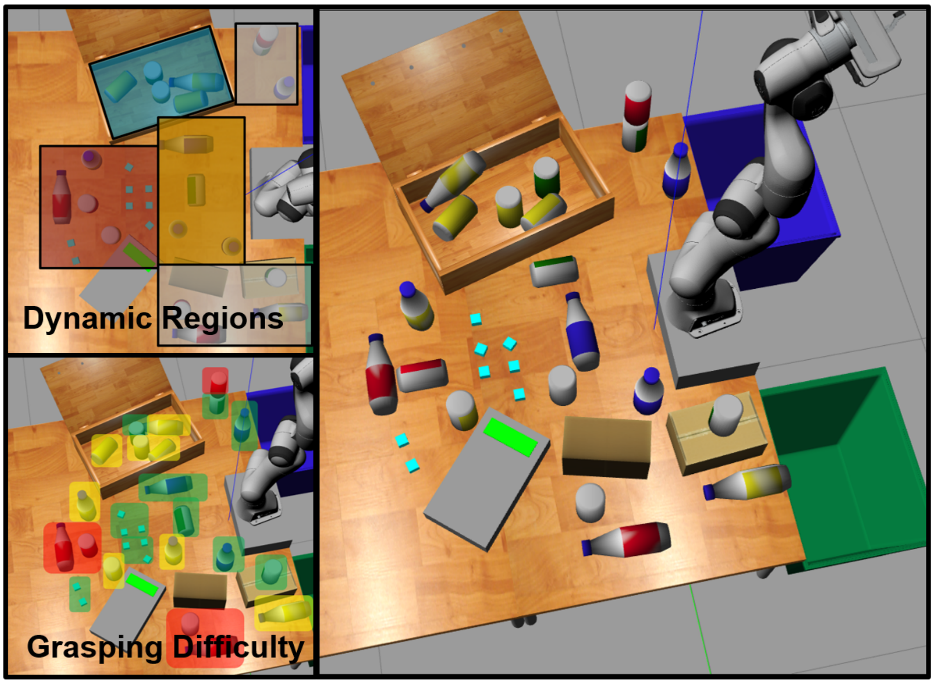
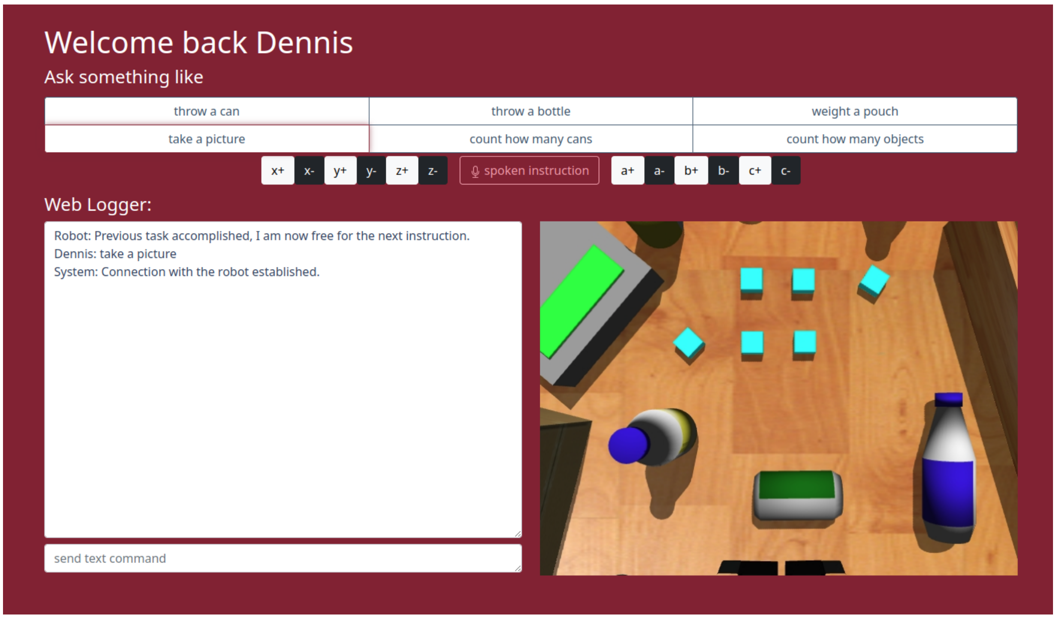


Figure 1: Picture of the challenge environment. Here the Franka Emika robotic manipulator is tasked with grasping various objects located in diverse regions. The level of difficulty and the element of randomness are contingent upon the positioning and orientation of the items.

perception and control algorithms, utilizing tools like MATLAB and Simulink, to deftly handle and maneuver objects like bottles, cans, and detergent pouches. These objects are subsequently weighted on a scale and/or disposed of in the appropriate waste bin. The performance is evaluated and points are awarded based on the algorithm's effectiveness when applied to new environments.

1.2 Motivation and project idea

In this scenario, the importance of planning becomes clear due to the unpredictability and complexity of the tasks the Franka Emika robotic manipulator has to perform. The objects to be grasped are arranged in an unknown, often random manner, and their positioning can vary widely. This requires the robot to adapt its strategy for each task, taking into account the unique characteristics of the objects and their locations. Furthermore, the robot also needs to ensure it doesn't collide with other parts of the environment during its operations. Accidental contact could lead to damage to the robot, the objects, or the environment itself. This adds another layer of complexity to the planning process, as it must incorporate spatial awareness and collision avoidance in addition to the primary goal of object manipulation. In regards to the aspect of Human-Robot Interaction (HRI), our team has decided to leverage the algorithms developed during the competition to lay the groundwork for a more interactive experience when collaborating with the robot. It's not uncommon for humans to desire a communication channel with a robot to provide specific instructions, for instance, distinguishing between various objects for disposal. In this scenario, suppose a user wishes to dispose of a specific empty bottle, rather than all bottles as stipulated in the initial challenge. Achieving this would necessitate the implementation of a mechanism that enables the robotic manipulator to simulate natural language understanding. This way, it can discern the specific verbal instructions that lead to the successful execution of the desired task. The software starter-kit provided with the challenge was not tailored to support such complex, interactive tasks, thus necessitating our development of the required infrastructure for remote robot access. This was accomplished by leveraging the rosbridge library of the Robot Operating System (ROS), which facilitated the creation of a middleware. This middleware extended the framework's intrinsic message passing functionality to web applications, thereby significantly simplifying the process of collecting voice inputs and broadcasting audio commands, ultimately improving robot-human interaction. Our robot's web interface (Figure 2) has been thoughtfully designed to accept inputs in the form of voice commands, text, and button presses. This design choice promotes easy extensibility and ensures intuitive, yet effective interaction with the manipulator. Moreover, it is capable of providing feedback even if a task cannot be completed due to factors such as the absence of the specified object or poorly formulated instructions.



Robot: Previous task accomplished, I am now free for the next instruction.
Dennis: take a picture
System: Connection with the robot established.

Figure 2: This is a screenshot of our fully-functional web interface, which we crafted using standard web technologies such as JavaScript, HTML, and CSS. This interface allows for both text and voice input collection and features handy shortcut buttons for executing specific commands.

We affirm that each student has made an equal contribution to this project.

2 Related Works

Without a doubt, the quest for creating autonomous robots, capable of executing actions based on human instructions, stands as a pivotal goal for both the research community and the industry. The day we see robots not only serving coffee in cafes but also correctly recycling on request, will undeniably mark a milestone, showcasing the remarkable progress achieved in this field. It's important to note that the advancement of this technology won't emerge out of thin air; it will be the fruit of several thriving and well-established fields. These include control theory, computer vision, natural language processing, and the considerable contributions from sociology.

2.1 Human Robot Interaction in Natural Language

In a playful twist of words, we can posit that spoken natural language is the "natural" choice for commanding a robot, especially for casual users, and is certainly more intuitive than any programming language. Consequently, many researchers are focusing their efforts on deciphering human commands by interpreting the constituent words in a sentence. This process begins with the recognition and transcription of user utterances through Automatic Speech Recognition (ASR), a field in itself that has seen numerous proposed approaches over the years. For instance, the study by [1] takes on the challenge of interpreting spoken input in a conversational dialogue system. Instead of a traditional pipeline approach where the output of a speech recognizer is the input of a language understanding module, this study merges multiple speech recognition and utterance classification hypotheses into one list to be processed by a joint reranking model. On a different note, the work by [2] capitalizes on the fact that modern robotic architectures are equipped with sensors that allow for in-depth analysis of the environment. They contend that this perceptual information, modeled through semantic maps in their study, can effectively enhance the language understanding capabilities of a robot. A robust lexical mapping function based on the distributional semantics paradigm is proposed as a model for grounding language in the environment. Their findings indicate that making such information available to the underlying language understanding algorithms boosts accuracy throughout the entire interpretation process. Additionally, the authors of [3] put forth a more robust statistical method, with the goal of improving the ranking of transcription lists. This approach aims to boost the performance of ASR systems in context-specific situations and is based on a semantic grammar that incorporates semantic actions, designed to emulate typical commands in the realm of human service robotics. To effectively execute pick-and-place tasks, discerning the action and its associated object from the provided utterance text is imperative. A prevalent strategy uses morpho-syntactic analysis, as highlighted in [4]. This method calls for each transcribed utterance to undergo part-of-speech tagging and syntactic parsing. This process yields essential morphological and syntactic information, which plays a crucial role in identifying the components necessary for planning the action. It's important to consider that requests may be ambiguous when there are multiple instances of the same object. As described in [5], several objects could meet the criteria for a request like “pick the water bottle”. To resolve such ambiguity, one approach, as suggested in [6], is to confirm whether the identified object is the correct one and, if not, switch to another. Alternatively, LSTM-based neural networks could be used to take advantage of spatio-semantic expressions to disambiguate, as illustrated in [5, 7].

2.2 Planning and Control

Receiving the instruction is merely the beginning, not the pinnacle, of the project. Understanding what the robot should do is only marginally more challenging than determining how it should navigate the given scenario. Indeed, considerations should include potential obstacles that could impede a robot's actions during task execution. Control theory lends significant aid in this context, particularly in recent years as the emphasis has shifted toward creating controllers for safety-critical systems. One such potent tool is control barrier functions (CBFs) [8]. These functions, designed to ensure safety during goal achievement, represent a safety property characterized by invariance; put simply, any trajectory initiated within an invariant set is assured never to reach the set's complement, the area where unwanted events, such as collisions, might occur. However, this method may not be the most effective when the task involves grasping objects. The volume of the robot, defined as all the points within its body, will inevitably intersect with the object, and CBFs would prevent such voluntary “collisions”. To address this, more suitable methods have been explored, such as those presented in [9], which ensure safe human-robot cooperation. A noteworthy approach in this respect involves the use of an impedance controller, a control strategy used in robotics where the controller manipulates the interaction between the robot and its environment by modulating the mechanical impedance of the robot. This control approach allows the robot to adapt its movements in response to external forces, enabling smoother and safer interactions between the robot and its environment, including humans and objects. Once the robot knows how to safely move, it must formulate a plan to fulfill the instructions provided. Fundamental frameworks for constructing such a plan can be achieved through propositional dynamic logics, as demonstrated in [10], or through linear temporal logic, as depicted in [11]. These methodologies are particularly adept at managing the various regions of the workspace that pose inherent complications, which can be astutely addressed by applying the appropriate policy.

3 Solution

The architecture of our solution elegantly fuses human-robot interaction with action planning. It's been crafted with the intent of understanding a wide array of requests. Central to our approach is the use of the YOLOv4 neural network [12], which we treat as the robot's visual perception system. The robot is equipped with a camera situated at the end effector, allowing it to take snapshots of its environment during movement. This ability facilitates the initiation of planning based on these captured images, that could involve tasks like enumerating the number of specific objects in view, or confirming the presence of a requested target. The reasoning component of our system is responsible for the actual execution of requests issued by the HRI interface. It involves interpreting the external environment through the camera sensors, and then making informed decisions on how to grasp objects without causing them to fall or inadvertently inducing collisions. A comprehensive diagram of the entire architecture can be seen in Figure 3.

3.1 Human-Robot interface

Our human-robot interface takes the form of a web page, which can be hosted on a server separate from the robot and accessed from any device. Its intuitive design allows users to activate a microphone, triggering the Automatic Speech Recognition module that captures voice inputs and sends the corresponding transcriptions to the server. Alternatively, the server can receive input commands directly from a chat feature or shortcut buttons provided on the webpage. Upon receiving the text, we conduct a simplified morpho-syntactic analysis (refer to Figure 4), which involves identifying particular syntagms within the sentence that are pertinent to our application domain. For instance, nouns of interest include items like {"bottle", "can", "pouch", "photo", "move", "bin"}, while relevant verbs could be {"recycle", "throw", "take", "weigh", "count"}, and significant adjectives might be {"all", "random", "green", "blue"}. This method proves substantially more efficient than relying on strictly structured English sentences, as it facilitates the modularization of the algorithmic component tailored to the particular request, without the need to recall the exact phrasing of the command, hence promoting a more natural, user-friendly interaction. The reasoning module's output is conveyed via a synthetic voice system, which randomly selects from a pool of pre-saved sentences. This results in dynamic and engaging communication, avoiding the monotony of repetitive phrases.

3.2 Perception reasoning

The utilized manipulator is equipped with an RGBD camera, positioned in line with the end effector, allowing the robot to capture views of its immediate surroundings.

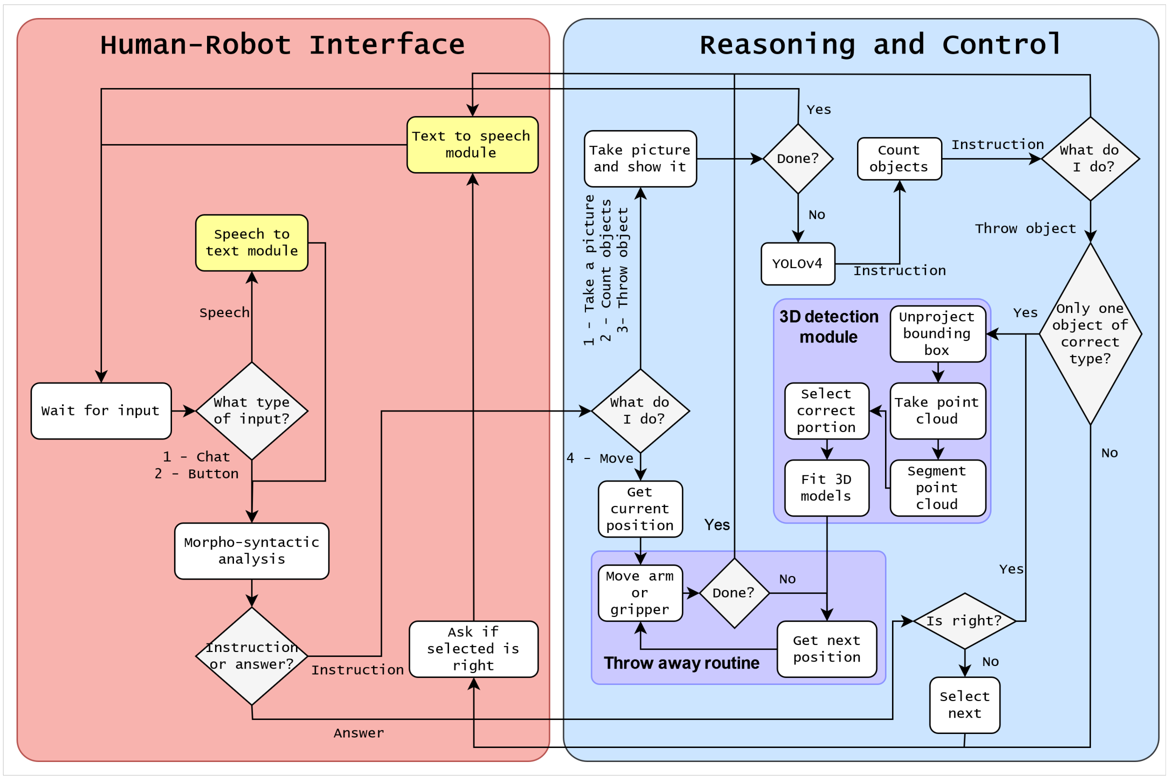


Figure 3: The flowchart showing how the system takes decisions. The HRI module takes various inputs in the form of voice commands, button presses, and written instructions. After an instruction is passed to the system the model decides in autonomy how to follow it. The yellow modules were taken as is, while all other parts of the project were implemented during this work. It's important to note that within the 3D detection module, there is an example of a method we have implemented. Depending on the region, it's possible to bypass certain steps due to pre-existing knowledge about the situation.

With this camera, the robot possesses the capability to take a snapshot of the scene directly in front of it, serving as its only means of perceiving the tables and the objects on them. By combining the RGB color information and the depth data from the camera, the robot gains a powerful visual understanding of its environment. Armed with knowledge about its own precise position and orientation, as well as the camera matrix with the camera's parameters, the RGBD information proves sufficient to not only detect the objects present on the table but also accurately locate them within the appropriate frame of reference. This knowledge enables the robot to effectively plan its grasp and manipulation strategies, ensuring that it can pick up objects with precision.

3.2.1 YOLOv4

One of the most significant challenges for a robot is interpreting the objects in its field of vision. Although seemingly a simple task for humans, it is indeed a complex endeavor

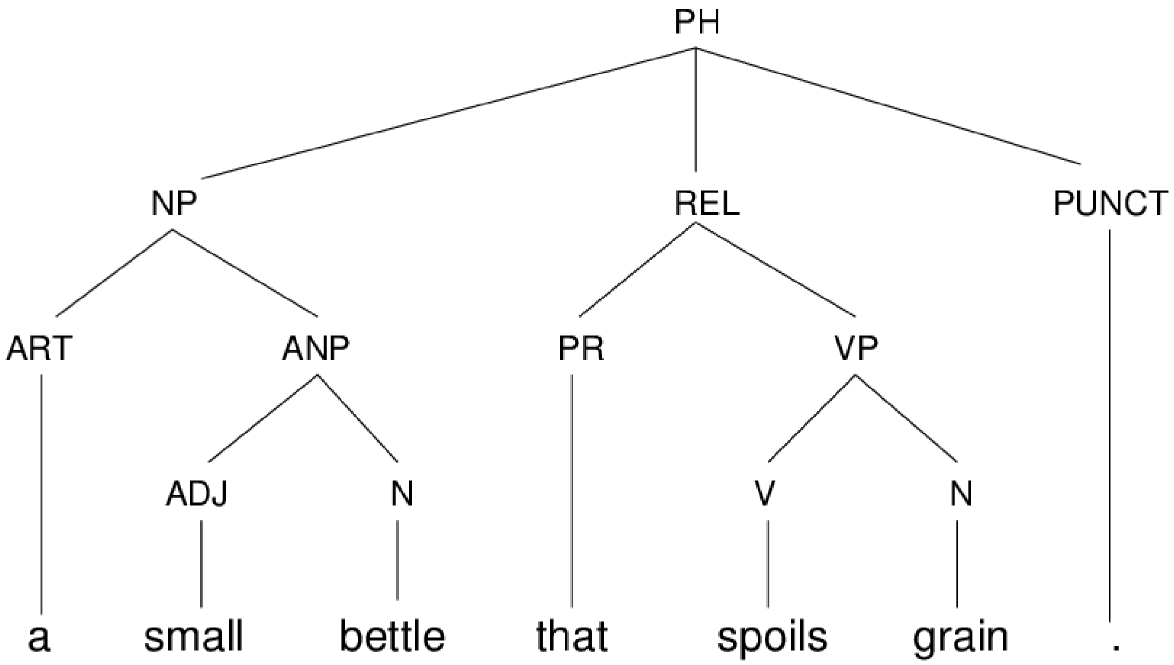


Figure 4: A morpho-syntactic tree is obtained from a lexical and grammatical analysis of the text. In this tree, ART represents an article, PUNCT indicates punctuation, V stands for verb, N is a noun, VP signifies a verb phrase, REL represents a relative pronoun, PH indicates a phrase, NP stands for noun phrase, ANP denotes an adjective noun phrase, PR signifies a preposition and ADJ denotes an adjective.

for a computer program to discern objects from the raw data matrices returned by a camera. Fortunately, recent advances in computer vision, particularly in the domain of deep learning, provide some formidable tools to address this challenge. Deep learning solutions, specifically for image recognition, have made tremendous strides in recent years. Therefore, we decided to leverage such a deep learning model to process the RGB images and detect the objects on the table. A noteworthy advantage of this approach is that color information from the objects can be utilized to differentiate between cans and bottles; only cans can be green, while only bottles can be blue. For yellow and red cans and bottles, their shapes often provide sufficient distinction. Among the plethora of available computer vision models for segmentation, we elected to employ YOLOv4 [12]. Firstly, because YOLOv4 is easy to train and it exhibits a remarkable ability to achieve reliable segmentation results, even with a limited set of labeled examples. This property significantly eases the burden of collecting and annotating a vast amount of training data. In addition to its performance, YOLOv4 is also known for its speed and efficiency, which are crucial aspects in real-time applications such as ours.

3.2.2 Dataset

The dataset used to train the model consisted of 130 images, labeled by hand. To ensure good performance and generalization capability the dataset was gathered with

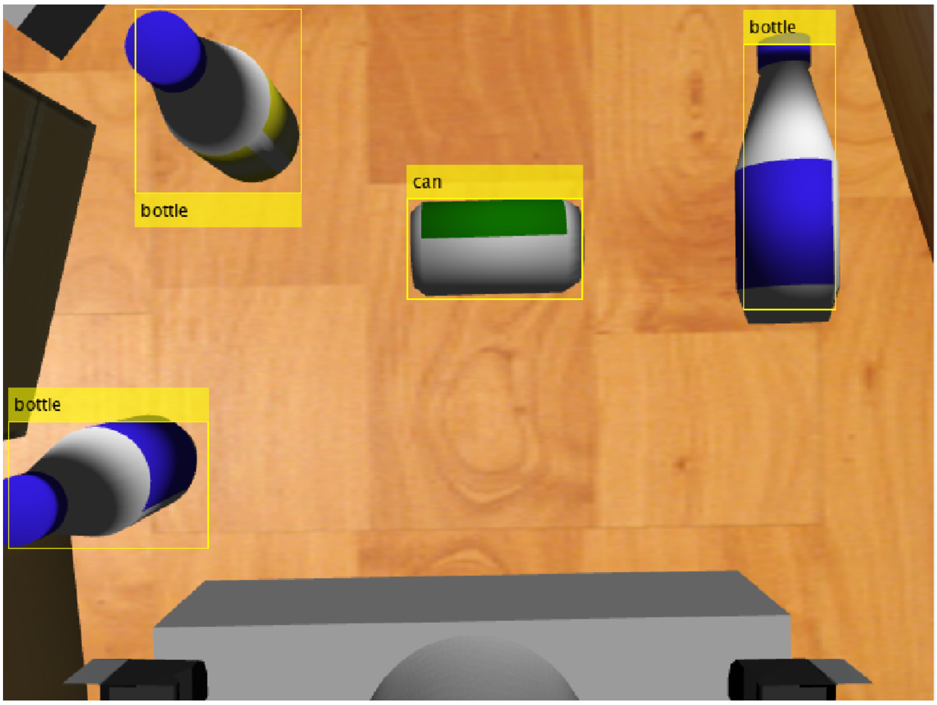


Figure 5: The output of the YOLOv4 network. The network was able to detect and identify correctly all the objects in the image.

two approaches. The first set of pictures of the dataset was taken randomizing the positions and orientations of the objects on the table and taking the pictures from predefined positions. Since these poses were close to what one would expect to utilize during the use of the robot this was an obvious choice to ensure good performances. By randomizing the positions of the objects but not the poses the aim was to aid the model in learning the important visual characteristic of the cans, bottles, and pouches while learning to ignore things like the table, the scale, and the boxes. The second part of the dataset was taken from the same environment, thus keeping the positions of the objects on the table the same, but by changing randomly the pose of the robot. This results in images that are very varied and ensures that the model is trained on images that contain elements that one might not have initially considered. For example, in some images, the robot pointed toward itself, or toward the floor. These images, while not strictly necessary for the task, improve the robustness of the model and their inclusion proved to be useful. In fact, they improved the performance of the robot in all those cases that one might not initially expect during normal operations. For example, before their introduction, the robot was biased towards always detecting at least one object in every picture and would fail to realize when no objects were present. The first set of images was also consistently looking at the table and the algorithm would sometimes hallucinate when the edge of the table was present in the pictures. On average each picture had 5 objects between cans, bottles, and pouches but the number of objects varied all the way from zero to more than ten for some pictures. The dataset used for training proved to be adequately large to attain impressive performance using YOLOv4. We incorporated conventional data augmentation techniques to enhance the diversity of samples, eliminating the need for labeling additional images. Figure 5 presents an example of the network's output. While the produced bounding boxes may not be perfect, these results are more than sufficient for our system, as the information gathered from the RGB camera will be further integrated with depth information, ensuring high-accuracy object localization.

3.2.3 Depth perception

As previously mentioned, the program relies on the depth information acquired from the camera to achieve a precise estimation of the position and orientation of the detected objects. When YOLO identifies an object, the program finds the center of the associated bounding box to determine its coordinates in the world frame. This process utilizes the depth values present in the captured image, in combination with the camera matrix and the pose of the robot. By integrating these components, the program is able to translate the relative position of the object within the image frame to its corresponding spatial coordinates in the robot's operational environment. This operation is commonly referred to as *unprojection.* Given a 2D point $\boldsymbol{p}_c$ expressed in the camera frame and given d its relative depth, the corresponding point in the world frame can be expressed as

$$\boldsymbol{p}_w = {}^{w}\boldsymbol{R}_c \boldsymbol{K}^{-1} \begin{pmatrix} d * \boldsymbol{p}_c \\ d \end{pmatrix} + {}^{w}\boldsymbol{t}_c \,,$$

where ${}^{w}\boldsymbol{R}_c$ and ${}^{w}\boldsymbol{t}_c$ represent the orientation and translation of the camera in world frame and $\boldsymbol{K}$ is the matrix that encapsulates the intrinsic parameters of the camera projection model. Note that in this environment the considered camera model is the pinhole one. Upon pinpointing the bounding box's center, the program goes a step further by focusing on the corresponding depth image to extract the associated point cloud data. This point cloud, essentially a collection of 3D points, provides a more detailed depiction of the object's spatial structure. With the known pose of the arm, the program can transform the extracted point cloud into world coordinates. Instead of solely relying on a single point cloud extracted from the image used for object detection, our system adopts a technique to capture a wider scope of the objects' structure. It utilizes multiple point clouds, obtained from different angles, for each distinct region on the table. The inclusion of point clouds from various perspectives allows the system to attain a more comprehensive understanding of the objects' geometric and spatial properties. This strategy enhances object perception as it accounts for potential occlusions and the varying appearances of objects when viewed from different angles. Following the point clouds' merging, points that evidently do not pertain to the objects of interest can be removed, specifically points associated with the table and the wooden box. This filtering is easily accomplished by evaluating the coordinates of the points and only selecting those within specific regions. After these processes, the remaining point cloud should primarily consist of the points of objects identified by YOLO, along with some other objects present on the table, such as the scale and cardboard boxes.

3.2.4 Point cloud segmentation

One of the main challenges when working with point clouds is devising a dependable method to segment them, such that each point cloud corresponds to an individual object. Various techniques exist for point cloud segmentation, and this project employs two of them: when the number and position of objects are known but not their orientation, a situation achieved using YOLO, we opted to use the K-means algorithm [13] for segmentation. The K-means algorithm is a widely used clustering technique designed to partition a given dataset into distinct groups or clusters based on their similarities and, possibly, an initial guess. The algorithm begins by randomly picking K initial cluster centers from the dataset. It then assigns each data point to the closest cluster center using Euclidean distance, aiming to minimize the total within-cluster variance. The algorithm recalculates cluster centers by averaging all points in each cluster, repeating this process until cluster centers stabilize or a pre-set stopping criterion is met. The final result is a partitioning of the dataset into k clusters, with each data point assigned to the cluster that corresponds to its nearest cluster center. While powerful, this approach does have limitations. It is best suited for clusters where points have comparable distances from the center. However, objects like bottles, which are elongated and narrow, present a challenge because their top and bottom are farther from the center than their sides. Furthermore, as the clustering problem is inherently NP-hard, the algorithm sometimes settles on sub-optimal solutions. In the context of our application, this can lead to inaccurate object identification and potentially unsuccessful object pick-ups. Consequently, we opted to use this approach for segmenting point clouds associated with pouches, as their number, positions, and cube-like shape are more compatible with the strengths of the K-means algorithm. The second technique employed for segmenting the point clouds is known as "Region Growing Segmentation". This method is based on the proximity of the points. It begins with a set containing a single point, and all the points that fall within a specified distance threshold are incorporated into the set. This process is iteratively repeated for all the newly added points. If no additional points meet the criteria, the set is stored and a new point is selected from the remaining point cloud. The process continues until the point cloud is completely segmented. This approach boasts the advantage of dealing with an unknown quantity of objects with arbitrary shapes, provided that the scans don't possess substantial missing regions. However, a primary drawback lies in the necessity to set an optimal maximum distance between points. A large threshold can lead to improper segmentation of closely placed objects, whereas a small one might separate points from the same object. Optimal performance was achieved with a relatively small maximum distance and by only considering point clouds with more than 100 points as valid. This method proved reliable particularly when the point cloud contained unexpected object parts. For instance, if part of a scale is present in a point cloud, the YOLO algorithm would only detect cans, bottles, and pouches in the image and return an underestimated number of objects for segmentation. This discrepancy could lead the K-means algorithm to incorrectly cluster the scale's points with another object. However, the Region Growing Segmentation method, based on the maximum distance, can effectively identify the scale as a separate object.

3.2.5 Identifying objects pose

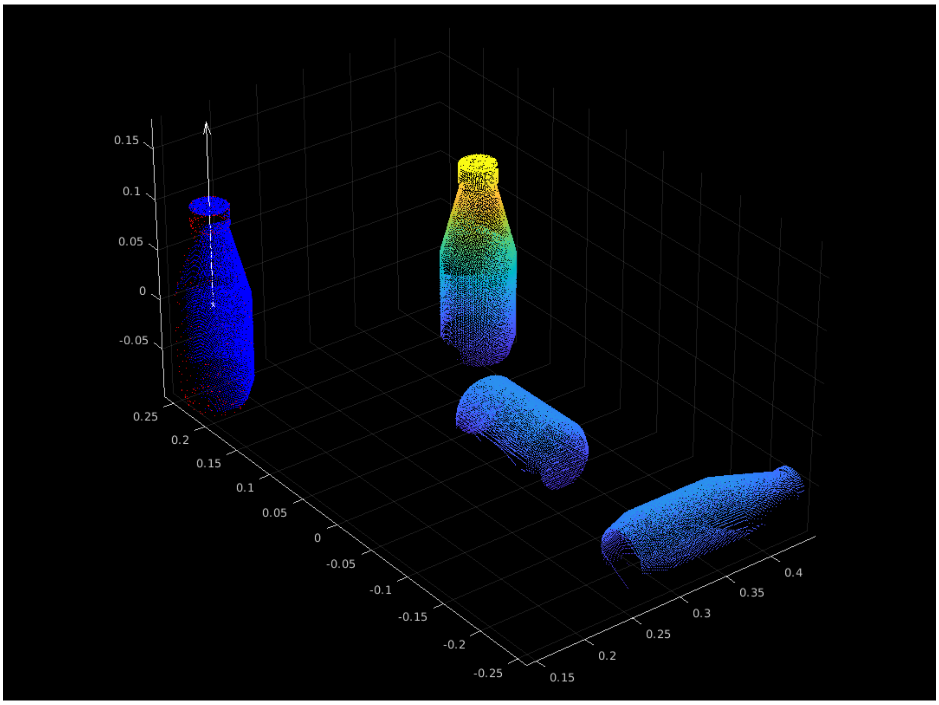


Figure 6: The point cloud of the 4 objects in front of the robot. The arrow in white starts from the detected center of the object and points toward the detected direction.

With the availability of distinct point clouds for each object within a given region, the process of selecting the appropriate one becomes straightforward, as it simply involves identifying the point cloud that is closest to the unprojected point obtained from the bounding box's center returned by YOLO. However, determining an object's precise position and orientation using its respective point cloud requires more finesse. For this purpose, we opted to employ point clouds generated from specially crafted 3D models for both bottles and cans. These models are then adjusted to align with the corresponding points in the point cloud data. By applying the resulting transformation to the known center and orientation of the 3D models, we can ascertain the exact position and orientation of the objects within the real-world environment. This method exhibits inherent robustness, delivering reliable results even when working with incomplete point clouds. Moreover, this approach has an additional advantage—it allows us to fit both can and bottle models to the points obtained. By assessing the mean square error of the fitting results, we can determine the specific type of object present. Hence, when the YOLO predictions lack confidence, this more robust strategy effectively overrides the initial object type detected by YOLO. This leads to improved accuracy and reliability in both object classification and localization. An example of the point cloud generated from the area in front of the robot is illustrated in Figure 6.

3.2.6 Detecting pouches

Pouches, introduced in the latest edition of the challenge, present a distinctive set of difficulties for the robot. They are small cube-like objects colored in cyan that have to be picked up and thrown just like the other objects. In addition to that, it is required that the pouch is weighted on a weighting scale before being thrown away to score more points. Importantly, pouches are destined for the same trash bin as cans. However, unlike cans and bottles, pouches have unique dimensions due to their cube-like shape. This means that in order to grasp them correctly, the robot's end effector must align its fingers with the square surfaces of the pouch. Successfully achieving this precise alignment necessitates the estimation of the object's orientation, adding an extra degree of complexity to the task. To determine the orientation of the pouches, the algorithm follows a different approach compared to the bottles and cans. After identifying the portion of the point cloud that represents a pouch using K-means, as described in previous sections, the algorithm proceeds to find the pose of the object in the world. Since there is no 3D point cloud model available specifically for the pouches, the algorithm attempts to find the diagonals of the square surface to compute the orientation of the pouch. It should be noted that this method may be more fragile and less effective outside of simulation, but it provides satisfactory results for the current needs of the challenge. To enhance the accuracy of the orientation estimation, one potential improvement is to formulate a least squares optimization problem using the Iterative Closest Point (ICP) algorithm [14]. By applying ICP, a 3D point cloud model could be fitted to the pouches, which has the potential to yield more precise results. Grasping the pouches requires the robot to position its end effector close to the surface of the table and exercise extra caution to ensure a successful grasp. The weighting task involves the robot moving above a fixed and known weighting scale, dropping the pouch onto the scale, re-grasping it, and finally throwing it into the correct bin. Interestingly, dropping the pouch from a small height enables the robot to re-grasp it without requiring an additional detection step. By simply lowering the end effector's position on the z-axis, the dropped pouch can be successfully re-grasped. However, it is acknowledged that this approach may not achieve the required robustness for real-life scenarios. To address this limitation, an additional step could be added to re-detect the dropped pouch using YOLO or another point cloud segmentation step.

3.2.7 Graduated Difficulty Levels across Competition Zones

The competition environment is divided into four zones (see Figure 1), each presenting its own set of challenges. This subdivision was designed to ensure a gradual increase in difficulty, requiring participants to employ different reasoning strategies in each zone. The *white* zone, the simplest of all, houses objects with fixed positions and orientations, maintaining the same object types. For tasks focusing on this zone, we can bypass additional detection or classification and direct the robot toward its destination using predefined world coordinates.

Next, the *yellow* zone introduces an element of variability. Though object positions remain constant, their type can change, shifting between bottles and cans. This necessitates accurate object classification before initiating the grasp-and-discard process, thus requiring a more intricate approach involving object recognition and classification algorithms.

The *red* zone escalates the complexity by allowing objects to rotate freely. Successful interaction with this zone necessitates accurate pose estimation of objects for the robot to align its gripper in correspondence with the object's orientation, enabling a precise grasp.

The pinnacle of difficulty is embodied by the *light blue* zone, or the box area. Here, objects are not just free to rotate, but also translate randomly. This demands a sophisticated handling process that includes precise object detection, classification, pose estimation, and meticulous motion planning to avoid collisions during the object grasp, ensuring safe and successful operations.

3.3 Motion planning and control

The most common use case for robots is by far the industrial setting. In this highly structured environment accuracy, repeatability and speed are the most valued characteristics desired in a robot. However, if we want to extend the range of tasks executable by a robotic application, we must develop new strategies to allow the manipulator to operate in less structured environments, without causing harm to the external world and especially humans. One of the most successful strategies for doing so is monitoring and controlling the exchange of interaction forces. Among various techniques we used impedance control. The main idea behind this approach is to impose a spring-mass-damper behavior to the robot arm when in contact, so that the other receiving end of the interaction does not experience a collision with the full rigid chain but instead the perceived inertia is much lower. This comes at the cost of more control effort and added complexity of the controller, both in computation and in sensing (additional external sensors might be required to correctly estimate the

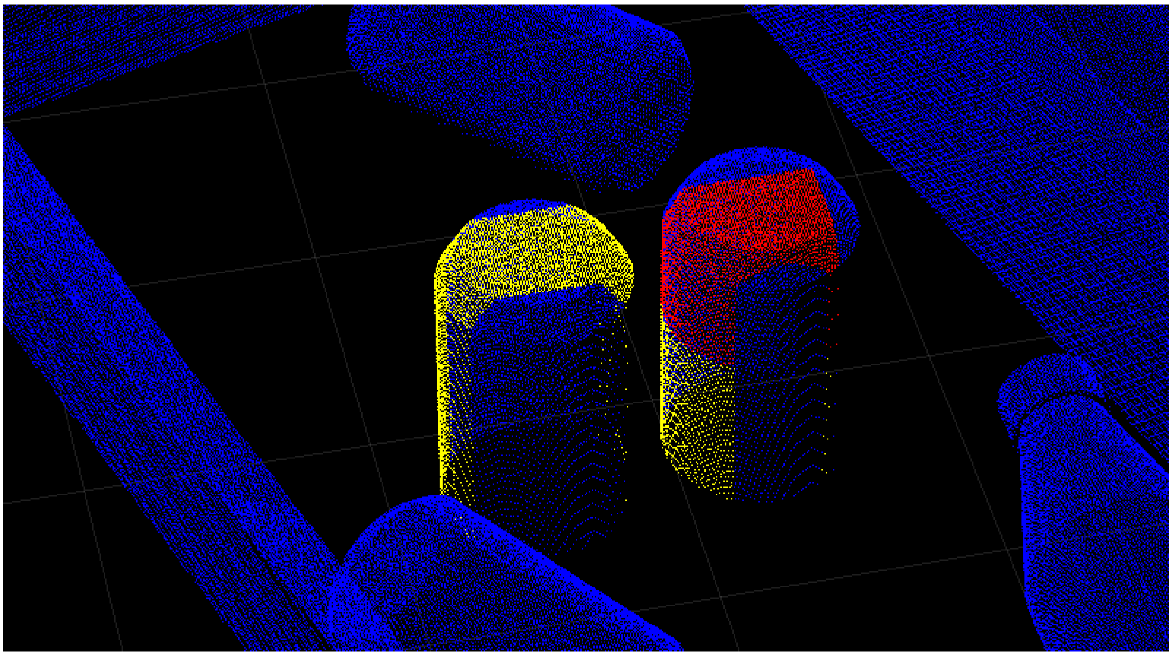


Figure 7: An example from the algorithm to find the best orientation to grasp an object. The target object is the can on the left, and the red points signify that a possible obstacle is detected. In this case, another orientation will be tried.

interaction). The control of the robot's pose is managed using Cartesian coordinates. Goals, comprising the desired position and orientation of the end effector, are passed to the controller, which subsequently manipulates the robot's joints to achieve the intended pose. A wrapper function, `moveTo`, is employed to publish these goals and control the position of the end effector. If the desired position is attained, the function confirms the successful completion of the move. However, should the function be unable to achieve the desired position within a specified time frame (which can be configured as a function parameter), it continually checks the end effector's position and velocity at one-second intervals. If the end effector halts, the function ceases its attempts and returns an unsuccessful result. This alone does not equip the robot with obstacle avoidance capabilities, but limits the damages in case of unwanted collisions. When moving long distances the controller tends to overshoot the target before correcting and reaching the desired position. For this reason, the operations that require fine movements are done in steps. When the robot needs to pick up an object the end effector firstly moves above the target and then it descends, with the gripper open, in small steps. Based on the type of the detected object the algorithm then automatically decides the right bin to go to and drop the object into. The orientation of approach when picking up an object necessitates careful consideration, especially in cases where objects are situated closely together. In such scenarios, the gripper risks colliding with an adjacent object while descending to grasp the intended object. This issue could be easily remedied by rotating the gripper by $\pi/2$ around the vertical axis. In light of this, the algorithm, after identifying the object's position and orientation, uses the surrounding point cloud to examine a rectangular area that is oriented to match the current desired gripper orientation. If points unrelated to the target object are detected within this area, the target gripper orientation is rotated by $\pi/4$, and a new collision check is initiated. An illustrative example of this algorithm is depicted in Figure (7). The points belonging to the region around the object are represented in blue. The points within the currently considered rectangular region are highlighted in yellow, and the points unrelated to the target object are marked in red. This approach ensures safe and efficient object grasping, avoiding potential collisions and enhancing the overall performance of the system. Based on the human user's request, the robot evaluates its current position, the location of the target object (that we can define as its state), and the ultimate objective of the action to perform optimally. We structured the program logic based on the framework detailed in this section, progressing in the order of the proposed solution. Once the robot possesses complete situational awareness, we initiate the functions that address the task at hand.

3.4 PDDL Planning

To solve the challenge of picking up all the objects from the table and throwing them out in the correct bin one interesting approach is to use a planner. This way a plan can be automatically generated and executed by the arm. Since we are dealing with a non-trivial challenge it would be quite complex to formalize it using standard planning languages like PDDL capturing all the details as in Figure 3. Consequently, in order to manage this complexity, we have decided to present a higher-level implementation of this solution. Our approach entails a planning formulation that uses PDDL but within an environment that is slightly less intricate than the one presented up to this point. As a matter of fact, we have hidden the detection pipeline under the action *detect* which gives the robot perfect knowledge about the object types and positions. We also decided to simplify the grasping of the objects with the action *pick* which can be performed when the robot is in the same region as the object. The solver that we decided to use is the Planner for Relevant Policies (PRP) [15]. The core elements in the problem are cans, bottles, and pouches that can be located in any of the four distinct regions on the table. The robot can then interact with the objects only if they are in the same region. This arrangement facilitates the environment's initialization while maintaining a non-trivial setup. Below we report the PDDL implementation for the domain and problem together with the plan formulated by the solver.

```
(define (domain arm_domain)
    (:requirements :strips)
    (:predicates (busy ?r)(hold ?x ?r)(at ?r ?x)(robot ?r)
                 (object ?o)(area ?a)(position ?p)(bottle ?b)
                 (can ?c)(pouch ?p)(detected ?o))

    (:action move
    :parameters (?r ?from ?to)
    :precondition (and (robot ?r)(position ?from)(position ?to)
                  (at ?r ?from))
    :effect (and (at ?r ?to)(not (at ?r ?from)))
    )
    (:action pick
    :parameters (?a ?x ?r)
    :precondition (and (at ?r ?x)(at ?a ?x)(robot ?r)
                  (object ?a)(detected ?a)(area ?x)(not (busy r)))
    :effect (and (hold ?a ?r)(not (at ?a ?x))(busy r))
    )
   (:action detect
    :parameters (?a ?x ?r)
    :precondition (and (at ?r ?x)(at ?a ?x)(robot ?r)(object ?a)
                  (not (detected ?a)))
    :effect (and (detected ?a ))
    )
    (:action throw
    :parameters (?a ?x ?r)
    :precondition (and (at ?r ?x)(hold ?a ?r)(robot ?r)(object ?a)
                  (position ?x))
    :effect (and (not(hold ?a ?r))(at ?a ?x)(not (busy ?r)))
    )
)
```

Listing 1: Domain description file in PDDL

```
(define (problem arm_problem)
    (:domain arm_domain)
    (:objects rob bin_bottles bin_cans pouch1  pouch2 pouch3 bottle1
              bottle2 bottle3 bottle4 can1 can2 can3 yellow_region
              red_region grey_region box_region)
    (:init (robot rob)(position bin_bottles)(position bin_cans)
           (position yellow_region)(area yellow_region)
           (position red_region)(area red_region)
           (position grey_region)(area grey_region)
           (position box_region)(area box_region)
           (object pouch1)(object pouch2)(object pouch3)
           (object bottle1)(object bottle2)(object bottle3)
           (object bottle4)(object can1)(object can2)(object can3)
           (pouch pouch1)(pouch pouch2)(pouch pouch3)(bottle bottle1)
           (bottle bottle2)(bottle bottle3)(bottle bottle4)(can can1)
           (can can2)(can can3)(at rob yellow_region)
           (at bottle1 yellow_region)(at bottle2 yellow_region)
           (at bottle3 grey_region)(at bottle4 box_region)
           (at can1 box_region)(at can2 red_region)
           (at can3 grey_region)(at pouch1 red_region)
           (at pouch2 yellow_region)(at pouch3 grey_region)
   )
    (:goal (and (at bottle1 bin_bottles )(at bottle2 bin_bottles )
                (at bottle3 bin_bottles )(at bottle4 bin_bottles )
                (at can1 bin_cans)(at can2 bin_cans)
                (at can3 bin_cans)(at pouch1 bin_cans)
                (at pouch2 bin_cans)(at pouch3 bin_cans) )
    )
)
```

Listing 2: Problem description file in PDDL

```
(detect bottle1 yellow_region rob)
(detect bottle2 yellow_region rob)
(detect pouch2 yellow_region rob)
(pick bottle1 yellow_region rob)
(move rob yellow_region bin_bottles)
(throw bottle1 bin_bottles rob)
(move rob bin_bottles grey_region)
(detect bottle3 grey_region rob)
...
(move rob bin_bottles red_region)
(detect can2 red_region rob)
(detect pouch1 red_region rob)
(pick can2 red_region rob)
(move rob red_region bin_cans)
(throw can2 bin_cans rob)
...
```

Listing 3: Plan found by deterministic planner

The strategy, as presented in Snippet 3, involves maneuvering the robot across the various areas, identifying all objects within them, and then sequentially disposing of each object. However, the challenge environment is inherently non-deterministic due to the possibility of failure in the simulated detection pipeline. Consequently, the success of grasping is compromised when an object is incorrectly detected, as the goal grasping position may not be optimal, increasing the likelihood of failure. The PRP solver becomes a crucial asset as it allows us to broaden the scope of robotic actions to include non-deterministic results. In relation to the current task, we implemented this solver to address situations where the robot's initial object detection might be unsuccessful. Nevertheless, the planner makes certain that the robot can revisit the area, re-detect the object, and successfully pick it up. For this purpose, we have defined a second domain description using actions that can have *probabilistic* outcomes, i.e. an object may fail to be picked up. When non-deterministic actions are involved the framework is called PPDDL (probabilistic-PDDL). In these situations, deterministic planners are not able to give an optimal plan because the preconditions of an action do not entirely characterize the effects and there is some degree of randomness involved.

We concluded that if the robot failed to pick up an object, there was a risk of the object being displaced. Hence, a failure in object pickup results in the object no longer being detected.

```
(define (domain arm_domain)
    (:requirements :strips :probabilistic-effects)
    (:predicates (busy ?r)(hold ?x ?r)(at ?r ?x)(robot ?r)
                 (object ?o)(area ?a)(position ?p)(bottle ?b)
                 (can ?c)(pouch ?p)(detected ?o))

    (:action pick
    :parameters (?a ?x ?r)
    :precondition (and (at ?r ?x)(at ?a ?x)(robot ?r)
                  (object ?a)(detected ?a)(area ?x)(not (busy r)))
    :effect (probabilistic 0.8 (and (hold ?a ?r)(not (at ?a ?x))
                                    (busy ?r))
                                0.2 (and (not (detected ?a))))
    )
    ...
)
```

Listing 4: Domain description file in PPDDL. The action *pick* is non-deterministic and if it fails it also makes the object be no longer detected, simulating the fact that the robot might have moved it while trying to pick it up. The other actions were kept the same as in the deterministic case.

Of course, the plan generated is no longer a deterministic plan and this makes it more difficult for humans to interpret it. PRP offers the possibility to set a particular flag, `--debug-output 1`, so that the planner simulate one particular plan in which the non-deterministic actions are carried out and their outcomes are chosen with the specified probabilities. Below we reported part of the plan generated in which it is clear that the robot failed to pick up an object (bottle2) and needs to find it again before picking it up.

```
detect bottle1 yellow_region rob (6)
pick_DETDUP_0_WEIGHT_4 bottle1 yellow_region rob (29)
move rob yellow_region bin_bottles (6)
throw bottle1 bin_bottles rob (6)
move rob bin_bottles yellow_region (6)
detect bottle2 yellow_region rob (6)
pick_DETDUP_0_WEIGHT_4 bottle2 yellow_region rob (29)
detect bottle2 yellow_region rob (6)
pick_DETDUP_0_WEIGHT_4 bottle2 yellow_region rob (29)
move rob yellow_region bin_bottles (6)
throw bottle2 bin_bottles rob (6)
...
```

Listing 5: The plan in the non-deterministic domain.

As can be seen in this particular instance the robot failed to pick up the second bottle and it had to find the bottle again and retry. Using PPDDL the problem can also be formulated in a probabilistic way, by assigning probabilities to the initial predicates as reported in Snippet 6.

```
(define (problem arm_problem)
    (:domain arm_domain)
    (:objects rob bin_bottles bin_cans pouch1  pouch2 pouch3 bottle1
              bottle2 bottle3 bottle4 can1 can2 can3 yellow_region
              red_region grey_region box_region)
    (:init
           (probabilistic 0.34 (at bottle1 yellow_region)
                          0.33 (at bottle1 red_region)
                          0.33 (at bottle1 box_region)
           ...
   )
    (:goal ...
    )
)
```

Listing 6: Problem description file in PPDDL. The different possible initial configurations can be defined together with the corresponding probability.

4 Implementation

4.1 Web page

To create a dynamic and visually engaging interface for human-robot interaction, we make use of the `Bootstrap` and `jQuery` libraries. `Bootstrap`, a popular front-end development framework, offers an extensive suite of pre-designed components and responsive layout classes, streamlining the process of designing visually compelling interfaces. The inclusion of `jQuery`, a robust and efficient JavaScript library, improves interactivity and simplifies the execution of complex tasks within our web page. The synergistic combination of `Bootstrap` and `jQuery` allows us to develop an aesthetically pleasing, responsive, and interactive interface with relative ease. In our quest to incorporate speech-to-text and text-to-speech capabilities within our project, we strategically opted to leverage existing robust solutions. Specifically, we employed the Web Speech API, an efficient tool furnished by Mozilla Developer Network (MDN), known for its comprehensive voice interaction capabilities. Of the various features offered by this API, the SpeechSynthesis interface was particularly advantageous since it offers a simple methodology for transforming textual information into natural-sounding and expressive audio output, thereby greatly enhancing user interaction. The SpeechRecognition interface, instead, allows us to easily translate voice inputs into text, which is then passed as instructions to the robot. For enabling effective communication between our web application and the server's ROS environment, we employ the `rosbridge` suite and `roslibjs` libraries. The `rosbridge` suite functions as a middleware, facilitating the flow of data and commands between the web-based interface and the ROS framework. Concurrently, `roslibjs`, acting as the client-side equivalent, seamlessly integrates with our web application. It provides a comprehensive collection of intuitive JavaScript features and classes, which empower us to effortlessly publish and subscribe to ROS topics, thereby augmenting the overall interactivity and efficacy of our system without cumbersome code.

4.2 MATLAB server

The Perception, Algorithm, and Control modules were developed using MATLAB, a decision informed by the RoboCup competition's specifications, which necessitated participant engagement with the ROS environment through MATLAB as the event's principal sponsor. We employed the `ROS Toolbox`, offering a robust set of tools for interacting with the simulation environment via the designated IP address. This toolbox not only facilitates the straightforward publishing and listening to ROS environment messages but also deftly manages user inputs from the web interface through a ROS callback mechanism. This function triggers the corresponding module that employs reasoning algorithms to fulfill the user request. We processed text in MATLAB to identify the syntactic structures necessary to comprehend the user's request. With this information parsed, we employed several MATLAB toolboxes for further processing and task execution such as the `Aerospace Toolbox` that provides straightforward quaternion functionalities, essential for executing point cloud rotations, crucial for our robot's orientation adjustment. We relied on the `Computer Vision Toolbox`, `Deep Learning Toolbox`, `Image Processing Toolbox`, and `Statistics and Machine Learning Toolbox` for training and implementing the YOLOv4 model, as well as leveraging probabilistic machine learning algorithms like ICP and K-means for point cloud segmentation and object identification. Specifically, the YOLOv4 model demanded a manually labeled dataset, which we trained on a local machine outfitted with a high-performance Nvidia GeForce RTX 3080 GPU. The robot's operation was simulated using the Gazebo simulation environment, while RViz was utilized for the visualization and testing of frame and point cloud messages, ensuring an accurate representation and understanding of the robotic operations.

5 Results

We are extremely pleased with the performance of our algorithm. Its robust capabilities have enabled us to successfully tackle a global challenge, securing our place among the top four university teams worldwide. This achievement has granted us the exciting opportunity to participate in the grand finals set to take place in Bordeaux, France, this coming July. The solution we presented for the competition was fully automated, developed directly from the successful outcomes of this project. It involved the robot systematically addressing each zone on the table one at a time, utilizing its RGBD camera to detect the presence of any objects. Upon object detection, the robot would gather the point cloud of the region, process it to ascertain the object's position and orientation, and then proceed to pick it up and discard it. Notably, pouches were placed on the scale before being discarded, as per the competition's requirements. After completing an initial sweep of a region, the robot would take another image to ensure all objects were collected. If additional objects were discovered, the procedure would simply recommence. The impressive results from the competition serve as further affirmation of our approach's quality and effectiveness.

Our web interface and its logic were thoughtfully designed with remote robot operation in mind, where the only feedback from the robot comes via the images it provides. This approach has proven its great effectiveness, as highlighted in our demonstration video. The system offers a personalized user experience by recognizing and remembering previous interactions based on user login data. Moreover, it provides immediate feedback upon receiving voice commands by displaying the understood message on the screen, which also allows the user to ensure speech-to-text translation accuracy. Text input, on the other hand, allows users to explore a range of commands effortlessly and is particularly beneficial for users who may be less comfortable speaking English. One of the key delights of interacting with this robotic system is witnessing it execute the commanded actions – it's a rewarding experience to watch it perform intricate tasks like picking up an object and disposing of it. In addition to this, the system's text-to-speech feature is invaluable for requesting further details or for the robot to inform users of its availability or current engagement. While text responses are displayed on the console logger, auditory feedback from the robot enhances the immersive user experience and aids in maintaining the user's engagement during interaction. This mix of interactive modalities makes interacting with the robot both captivating and efficient. As demonstrated in the video, the success of our project hinges heavily on the YOLO model, given its critical role in precise object recognition via the RGB camera. It underpins a multitude of our system's functions - from quantifying the number of objects, discerning if the object of interest aligns with the user's inquiry, to pinpointing their coordinates for successful grasping. Our robot's ability to understand and act upon the nuances of language is one of its distinguishing features. When instructed not to perform a certain action, it will confirm that it "won't do it," reflecting its understanding of negation in the sentence, rather than resorting to a simpler response that ignores this subtlety. The robot can discern requests for the selection of random objects, consequently generating a random number to independently choose from the objects in view. The object's physical properties guide the robot's execution plan for grasping procedures. Factors such as the grasp amplitude and the spatial position are considered, leveraging a 3D model of the object it possesses. Moreover, when asked to discard an item appearing only once in its field of view, the robot will not seek confirmation but will proceed with the task immediately, making logical inferences from its observations. The robot's state mechanism enables it to verify its availability (always confirmed on the GUI upon task completion), recognize when it's busy (responding that it can't take on other tasks until it's free), or ascertain if it's in a waiting state pending user confirmation. Moreover, our robot embraces robust interactions, maintaining clarity and usefulness in its responses. If asked to count or dispose of items not present in its environment, like "dogs" or "cats", it will inform the user that it doesn't see such items and needs to be repositioned. This provides valuable information to the user about the robot's 6-degree freedom mobility. Similarly, if the robot repeatedly receives a "NO" answer when seeking object confirmation, it will deduce that there are no more of that type of object within its snapshot. This clever capability might even help the user realize they misidentified an object themselves. We have also observed that fitting 3D models to the gathered data not only provides exceptional accuracy but also ensures a highly dependable method for precise object localization. Harnessing the prowess of the YOLO model for object detection and coupling it with the precise localization enabled by our algorithm's proficiency in fitting 3D models, we've created a perception system that allows our robot to interact with its surroundings with remarkable precision and efficiency. This combination proves particularly effective for grasping and discarding objects, while simultaneously ensuring safe navigation throughout the environment.

6 Conclusions

Undoubtedly, this project stands as one of the most educational and challenging experiences throughout our entire master's degree program. Despite the integration of Robotics and Artificial Intelligence in our degree's name, no other course offered a clear path for the convergence of these two methodologies. More often than not, courses in Natural Language Processing encouraged us to delve into the nuances of the specific domain, while Robotics courses inclined us towards the usage of traditional controllers. It was only upon the practical application of the knowledge we had acquired during the degree that we began to grasp the intricacies and allure of these disciplines. Moreover, we understood the necessity of one for the other. Machine learning point cloud analysis is indeed remarkable, but without a dependable controller, it is challenging to navigate the world smoothly without causing damage to surrounding objects. On the same note, without the comprehensive training of models like YOLO, identifying multiple objects in an image and reprojecting their centroids for accurate object grasping is an arduous task. Though we likely utilized knowledge from every course we've taken over these two years, we believe that there's room for extending the project further.

For instance, developing additional vision features to interpret when a person intends to interact with the robot, as discussed in class [16], would be one way. Similarly, incorporating non-verbal cues, such as gestures indicating the object we want the robot to move, could help mitigate ambiguity that might arise from merely naming the object. For the planning aspect, exploiting the temporal relationships inherent in the Linear Temporal Logic (LTL) formalism could lead to more complex instruction handling, such as "recycle a bottle and then a can". At present, this instruction would result in the recycling of either a can or a bottle, but not both.

References

[1] Fabrizio Morbini, Kartik Audhkhasi, Ron Artstein, Maarten Van Segbroeck, Kenji Sagae, Panayiotis Georgiou, David R. Traum, and Shri Narayanan. A reranking approach for recognition and classification of speech input in conversational dialogue systems. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 49–54, 2012.

[2] Emanuele Bastianelli, Danilo Croce, Roberto Basili, and Daniele Nardi. Using semantic maps for robust natural language interaction with robots. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association*, pages 1393–1397, 2015.

[3] Andrea Vanzo, Danilo Croce, Emanuele Bastianelli, Roberto Basili, and Daniele Nardi. Robust spoken language understanding for house service robots. *Polibits*, 54:11–16, 07 2016.

[4] Emanuele Bastianelli, Danilo Croce, Andrea Vanzo, Roberto Basili, and Daniele Nardi. Perceptually informed spoken language understanding for service robotics. In *Proceedings of the IJCAI-2016 Workshop on Autonomous Mobile Service Robots*, 2016.

[5] Mohit Shridhar and David Hsu. Grounding spatio-semantic referring expressions for human-robot interaction. *CoRR*, abs/1707.05720, 2017.

[6] Rohan Paul, Jacob Arkin, Derya Aksaray, Nicholas Roy, and Thomas M. Howard. Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms. *The International Journal of Robotics Research*, 37(10):1269–1299, 2018.

[7] Jun Hatori, Yuta Kikuchi, Sosuke Kobayashi, Kuniyuki Takahashi, Yuta Tsuboi, Yuya Unno, Wilson Ko, and Jethro Tan. Interactively picking real-world objects with unconstrained spoken language instructions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3774–3781, 2018.

[8] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, 2019.

[9] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi. An atlas of physical human–robot interaction. *Mechanism and Machine Theory*, 43(3):253–270, 2008.

[10] Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: Logical framework and implementation. In *KR-00*, pages "678–689". MK, 2000.

[11] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. From structured english to robot motion. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717–2722, 2007.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[13] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

[14] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[15] Christian Muise, Sheila A McIlraith, and J Christopher Beck. Improved non-deterministic planning by exploiting state relevance. In *The 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[16] Sebastian Loth, Kerstin Huth, and Jan De Ruiter. Automatic detection of service initiation signals used in bars. *Frontiers in Psychology*, 4, 2013.

[17] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007.

[18] G. Oriolo, B. Siciliano, L. Sciavicco, and L. Villani. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.