IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 June 2024, accepted 30 June 2024, date of publication 3 July 2024, date of current version 12 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3422808

RESEARCH ARTICLE

Everyday Objects Rearrangement in a Human-Like Manner via Robotic Imagination and Learning From Demonstration

ALBERTO MENDEZ, ADRIAN PRADOS, ELISABETH MENENDEZ, (Graduate Student Member, IEEE), AND RAMON BARBER, (Senior Member, IEEE)
Robotics Laboratory, Universidad Carlos III de Madrid, 28911 Madrid, Spain

Corresponding authors: Alberto Mendez (albmende@pa.uc3m.es) and Adrian Prados (aprados@ing.uc3m.es)

This work was supported by the Ministerio de Ciencia e Innovacion through the Advanced Mobile Dual-Arm Manipulator for Elderly People Attendance (AMME) under Grant PID2022-139227OB-I00.

ABSTRACT The rearrangement of objects is an essential task in daily human life. Subconsciously, humans break down such tasks into three components: perception, reasoning, and execution, which are automatically resolved. This process represents a significant challenge for robots, as they must apply complex logic to treat all the information and successfully execute the task. In this research, we propose a solution to perform this task in a human-like manner. For that purpose, we developed a modular framework that provides the capability to observe and understand the scene, imagine the best solution and execute it, following human-like reasoning. This is done by combining a zero-shot deep learning model for perception, a zero-shot large diffusion model to provide an ordered and realistic final scene and a Learning from Demonstration algorithm for execution. To test the performance, we conducted several experiments to check the correct resolution of 2D rearrangement tasks. For that purpose, we have tested the feasibility of the final generated scene, the ability to generate the trajectory by means of human demonstrations and, finally, we have carried out experiments with two different robots in a simulated and a real environment. The results obtained prove the adaptability of our framework to different environments, objects and robots. Moreover, the success rate of solutions provided and the error in the position and orientation demonstrate a significant advance in the accuracy and effectiveness of solving rearrangement tasks.

INDEX TERMS Rearrangement task, robot perception, instance segmentation, diffusion model, prompt generation, learning from demonstration, imitation learning.

I. INTRODUCTION

For humans, performing certain tasks can be so simple that we can do them automatically, almost without thinking. However, for a robot, these same tasks can pose a significant challenge. Tasks such as moving obstacles to reach an object or rearranging objects in a space are complex for robots. The task of rearranging objects in a space is particularly relevant since it is a process that humans have internalised and automated. However, it requires a series of steps for proper execution, such as locating the objects, establishing a final scene, determining the order of execution for different objects, and the precise manipulation of various objects, each with distinct sizes and shapes.

In this context, there are many approaches to this problem, which is present in a large part of people's everyday tasks. Usually, it has been solved by dividing the process into a certain number of steps, using stored final positions provided by

The associate editor coordinating the review of this manuscript and approving it for publication was Jason Gu.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

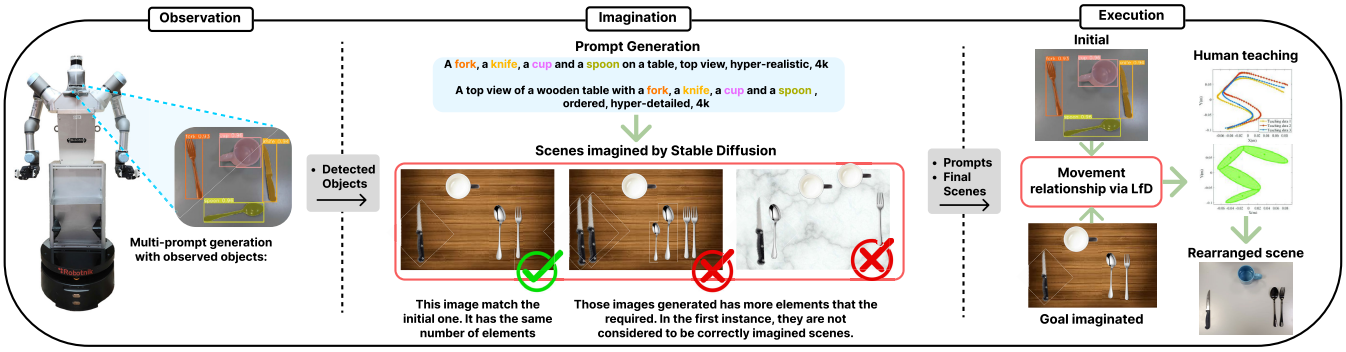

FIGURE 1. Workflow of our method which introduces a three-step process for robotic rearrangement. It begins by recognising and processing the cluttered initial scene into individual object during the *Observation phase*. Subsequently, these objects undergo a transition to the *Imagination phase*, employing Stabble DiffusionXL [4] to generate the goal positions and orientations. Finally after the matching process and the confirmation that configurations are possible, in the *Execution phase*, robotic action policies are generated by matching the initial and goal scenes learning the movement model via Learning from Demonstrations (LfD) provided just one time by the user. Our method boasts characteristics such as being open-source, adaptive and autonomous.

users [1], [2], [3] and pre-programmed movements. However, the ideal execution requires the robot to operate completely autonomously. This implies that the robot must be able to understand the initial scene, identify the objects and locate them in space, reproduce a coherent final configuration, and rearrange the objects to adopt the new configuration without human intervention. Subsequently, as an alternative to the use of human data, there is an increasing reliance on vision-based methods, which can be divided into three distinct approaches: utilising known methods of geometric and semantic target states, sequential object pose estimation, and zero-shot rearrangement with large models.

Generally, in goal-driven methods [5], the effectiveness of these preconditions has a notable impact on the precision of the reorganisation. When the target state is unavailable, these methods are not suitable for real-world applications.Additionally, in pose estimation-based approaches [6], [7], despite their well-aligned sequential design with robotic manipulations, there is a potential vulnerability to cumulative errors in autoregressive predictions. In the last type of methods, is where the use of diffusion models that are able to generate images such as OpenAI's DALL-E [8] or Stable Difussion [9] would come in. These models, having been trained through a large amount of human data both in image and video format as well as in text format coming from the Internet. This makes them capable of generating through simple prompts a large number of images similar to those that a human could imagine.

But in rearrangement tasks, it is not only the process of generating the end points that is important, but also the logic of the movements and their execution. Most of the work that focuses on this type of task neglects this aspect, making use only of pre-programmed movements, which generates sub-optimal movements that are not capable of working in a way that is similar to what a human would do [10], [11].

Hence, considering the limitations and advantages of the methods developed to date, this work introduces a new proposal for environmental rearrangement tasks. Our framework involves the use of open-source tools to solve rearrangement tasks. For that purpose, we have developed a novel framework, that is composed of several modules performing different tasks, similar to the functioning of the human mind. The first module designed is dedicated to observing and understanding the environment, detecting its elements. The second module focuses on envisioning the final configuration of these elements, and the last module controls the movements to be executed. The whole framework combines state-of-the-art and self-deployed algorithms that are interconnected in a novel way. A visual idea of our proposal is shown in Fig. 1.

The first created module combines a Convolutional Neural Network (CNN) for object detection and localization with a self algorithm to extract the poses of each detected object. To this end, we use the YOLO (You Only Look Once) [12] network with weights pretrained on COCO (Common Objects in Context) [13], a dataset with a multitude of everyday objects.

The second module is responsible for generating a description of the scene from the detected objects. Moreover, a Latent Diffusion Model (LDM) [4], capable of generating multiple images based on the description of the initial scene obtained by the preceding module is used. In this case, we have employed Stable Diffusion. The obtained images are verified to match the number of initially detected objects, and if not, they are adapted to match through an automatic selection of the most similar images and a subsequent filtering process to remove extra elements.

In the third module, the necessary movement order is verified to avoid collisions, and a Learning from Demonstration (LfD) [14] algorithm using probability fusion is applied. The learning algorithm developed by us enables learning the movement model and is responsible for generating the required movements for an specific task. To perform this process of learning from multiple demonstrations and generalizing to new situations, we have used an algorithm based on Task-Parameterized Gaussian Mixture Models

(TPGMM) [15], modified by us to automatically estimate the task parameters that are truly relevant for solving a new situation. Depending on the movement to be performed, the algorithm filters parameterized tasks and retains only the relevant parameters for each type of movement. This process is carried out through an iterative evaluation using a custom cost function developed via a probabilistic evaluation based on the Kullback-Leibler (KL) [16] divergence between the different task parameters. Based on the results of this evaluation, the algorithm decides whether to eliminate or retain the evaluated task parameters, thereby modifying the learned model.

Finally, to verify the correct performance of our proposal, a series of experiments have been conducted to assess the framework's functionality both quantitatively and qualitatively. These experiments include the rearrangement of different types of objects (utensils, fruit, office supplies), and the verification of configurations and movements generated compared to those performed by a human. Additionally, an evaluation of our LfD method has been conducted against other state-of-the-art learning algorithms. The main contributions presented and developed in the paper are:

- A modular **Python framework** designed to execute reordering tasks, ranging from visual perception to actual execution by a real-world robot, aiming to replicate human imagination capabilities.
- An **estimator of object orientations** that refines the initially detected orientation according to the pointing direction of each object.
- A **multi prompt generator** that uses the detected information to create instructions to the final scenes generation. In addition, a selector of valid among all the possible final scenes has been developed.
- A **collision checker** that generates the ordered list of objects to avoid collisions during manipulation.
- A **LfD algorithm** based on the selection of relevant frames using TPGMM has been designed for the generation of movements and the sequencing of this process trying to replicate how a human would solve these tasks.
- A combination of zero-shot techniques such as YOLO [12] and StableDiffusion [9], with our own methods to create an application capable of solving reordering tasks **without the need for specific task training**.
- A **fully autonomous framework**, in which no person is needed for the generation of the final position or the generation of relations between objects. Additionally, it can be applied to different types of robots, regardless of their physical constraints.

Our framework is open source, as we use our own code and open source tools such as YOLO or Stable Diffusion. A detailed tutorial of how it works, is fully provided on **GitHub** (https://github.com/AdrianPrados/Robotic-Rearreagment-of-Everyday-Objects). Additionally, you can watch videos of the method developed for the rearrangement tasks with the ADAM robot in simulation **(https://youtu.be/gyn1992YxhI)** and with the TIAGo robot in a real-world environment **(https://youtu.be/IwdsndI9cW0)**.

II. RELATED WORK

A. OBJECT POSE ESTIMATION

Visual perception plays a crucial role in robotics, particularly in the realm of object recognition, when addressing the management of domestic environments. The detection and estimation of object positions is of great relevance for manipulation tasks as it is necessary to know where and how the different objects to be worked with are located in the environment. The intricacies and diversity inherent in human homes demand a resilient visual perception model with the ability to differentiate among a broad spectrum of objects.

Due to re-training object detection and position estimation models can be an arduous task, it is common to use zero-shot learning or pre-trained algorithms with a large amount of data accessible to everyone. Some examples of these algorithms are [17] which allows not only to obtain the pose of a given object but also to determine which are the transformations for a given movement of this object, the algorithm ZePHyR [18] which allows new data to be entered into the dataset very quickly, taking into account factors such as colour and texture. Other algorithms use the information obtained from an object detector to estimate the pose of detected item, for instance, in the algorithm ZS6D [19] that through Vision Transformers is able to estimate the POSE of detected elements in real time, or YOLO-6D+ [20] an approach that extends YOLO to learn the 3D shape of an object from a single RGB image.

Among the different zero-shot models utilised for object recognition, the You Only Look Once (YOLO) model has become a preferred option due to its efficiency and effectiveness [21], [22]. YOLO's distinctive ability to estimate classification scores and bounding boxes directly from input images has positioned it as a valuable tool in home manipulation applications [23].

In our work, we have specifically use the YOLOv8 [24] model to identify objects without the need to train a specific model by creating a set of estimates relevant to our project. This provides us with a set of critical geometric data, which allows us to infer essential parameters such as the position, orientation, width and length of each element. This customisation enhances the utility of the model beyond mere object recognition, contributing to a more comprehensive understanding of the object's spatial characteristics for the rearrangement task.

B. OBJECT REARRANGEMENT

Object rearrangement tasks consist of two fundamental components: the estimation of an endpoint and the transition between the estimated position of the initial objects and the final point of arrival.

First of all, certain approaches for the estimation of endpoints consider the forecasting of goal poses as a classification challenge, involving the selection from a predefined set of discrete options for placing an object. In the context of rearranging spaces on a household scale, employing a pre-trained language model allows for the prediction of goal receptacles, such as tables [3], and the automatic detection of out-of-place objects [25]. On a room level, the appropriate drawer or shelf can be classified, with considerations for individual preferences [26], [27].

Another method of obtaining endpoints is based on prediction at a more detailed level from a dense set of target positions. For this, using graph neural networks [28] or a preference-aware transformer [29], we can generate high-resolution images that indicate where objects should be placed. This does not require a set of predefined discrete user-defined options to be predefined as is the case with the previously described methods and allows more accurate poses to be predicted than is possible with language.

Once the end points and the positions of the initial objects have been estimated, the second step begins, which is to obtain the transitions. One of the most common methods is scene graphs. These graphs provide a rich symbolic and semantic representation of scenes [30], allowing relationships between objects and their relative positions to be established more explicitly than language [31]. One of the most widely used ways to compactly generate these relationships is through spatial grounding [32], predicted from images [33]. Recent research in robotic manipulation presents work that leverages scene graphs in planning for rearrangement tasks [34], [35]. Other approaches of scene graphs that have been studied in current work include using optical flow, as the IFOR algorithm [5], or feature cosine similarity [36].

Another way of solving rearrangement tasks that is highly applied today are methods for predicting continuous poses of objects.These methods typically use a dataset of exemplary layouts provided by users. Those kind of algorithms can learn spatial preferences with a Variational Autoencoder graph [1]. There are also works that make use of a language-conditioned reorganisation, where an auto-regressive transformer is employed [6], or a diffusion model on poses which can be combined with learned discriminators to avoid collisions [7]. Other methods predict continuous target states by iterative de-noising [37], or perform collision avoidance during the reorganisation process using gradient fields as the TarGF algorithm [38]. Other reorganisation approaches use complete demonstrations given by users as CliPORT [39] or transporter networks presented in [40], or apply priors such as human pose context [41].

Although all the previously explained methods allow to solve the reorganisation problem in a correct way, there is a limitation that comes from the complexity in real environments of the graph-based methods and the need for a large amount of data. In all cases, additional data collection is necessary for each of the environments, which restricts their application to certain environments.

Our proposed framework does not necessitate the collection and training of a dataset of reordering examples. Instead, we demonstrate that leveraging existing web-scale diffusion models allows for zero-shot rearrangement. Such models are used both for the extraction of the relevant position and orientation data and for the estimation of related factors such as object relationships in a similar process to the human imagination.

C. DIFFUSION MODELS AND IMAGE GENERATION FOR ROBOTIC MANIPULATION

Diffusion models have become the state-of-the-art in image generation. They are based on the idea of learning by reserving a process and then perturbing it by introducing noise (i.e. diffusion) to generate different samples [42]. This virtue makes them perfect algorithms for image generation [43], [44]. As a result, multiple solutions for this purpose have been appearing in recent years [4], [8].

Text-to-image Diffusion Models not only work for image generation, they also support image editing, starting from an initial image with a prompt indicating the modifications to be made (i.e. img2img) [45], and generating a new image with these modifications. Moreover, changes can be made only in some areas of the image (i.e. inpainting) [46], providing the mask of what has to be changed with its descriptive prompt, obtaining as a result a new image that combines the information of the initial image with the modifications in the indicated mask.

Many applications have appeared with the use of these models. For example, there are solutions for photo editing, allowing to obtain the desired results even with elements that are not in the original image [47], [48]. Meanwhile, in the research field, their application has focused on the generation of datasets [49], [50] and providing robots with ''imagination'' [7], [51]. This last idea is one of the basic concepts of our proposal. In the task of rearranging objects, some proposals replicate the imagination by making use of the information of the objects (using Point Clouds) and high-level language goals [7], others use inpainting techniques to rearrange the objects in the scene [51].

Taking all these techniques into account, our proposal consists of using a text-to-image technique, without using additional information or intervening in the process. Adding some extra validation steps in the process, we ensure that a correct image is obtained to solve the rearrangement task. Being able to generate a large number of images, subsequently analysed, in order to use the best matching one. These images are generated by StableDiffusionXL [4] using only pre-trained weights, so we have a zero-shot configuration. That maintains the open-source spirit and non-human intervention of our method.

D. ROBOTIC LEARNING OF MANIPULATION TASKS

Classical approaches to the manipulation process in object reorganisation tasks employ pick and place actions [52],

[53] and focus on using motion planners [54], [55]. More recent work attempts to leverage non-prehensile actions for more efficient solutions [56], [57] in confined spaces like tables. The reorganisation of elements in the environment poses problems such as obstructions caused by clutter in a single-camera configuration, limitations in the face of unexpected environmental modifications, or the lack of complete knowledge of the environment [58], [59].

In an attempt to overcome these limitations, robotics is moving towards the use of learning techniques. Within this field, numerous improvements have been developed for performing human and everyday tasks [60], [61] such as cleaning and polishing, stacking, assembling parts into holes, or pick-and-place tasks [62], [63]. For rearrangement tasks, some works have proposed a deep reinforcement learning approach [64], while others have introduced transformer-based algorithms like the Transporter Network for vision-based manipulation tasks [40].

Other approaches focus on the use of learning techniques based on human data, such as Imitation Learning or Learning from Demonstrations. One example is TarGF [38], which is based on gradient descent and uses target examples provided by experts. Other examples include the work presented in [65], which uses inverse reinforcement learning for non-prehensile multi-object rearrangement, or the NeRP algorithm presented in [66], which also utilises user information for goal state selection via a neural network.

While these works have contributed significantly to robot manipulation in specific to rearrangement tasks, our work aims to design a rearrangement framework based on Learning from Demonstration. In our algorithm, a mixture of probabilities is applied based on the movement information. Each trajectory demonstrated by the users is composed by a probabilistic representation using Gaussian models. These Gaussians are combined to obtain the zones with the highest probability of movement using the means (centre of the Gaussians) and the covariances that define the size of the Gaussians. This allows the algorithm to reorder these Gaussians to learn the highest probability of the movements to be performed, thus avoiding collisions with objects in the environment and adapting to the specific characteristics of the environment where the task is performed.

E. OBJECT REARRANGEMENT IN ROBOTICS

Nowadays, different types of robotic implementations have been increasingly developed that focus on solving problems of reorganising elements of the domestic environment.

An example of robotic object rearrangement is presented in [67] a framework via multi-view fusion that is able to estimate the local similarities using a dataset an a goal image provided by the users. Another example is StructDiffusion, presented in [7], a language-guided framework that allows the user the creation of physically-valid structures using unseen objects by just generating an specific order. The framework ROSIE [68] is able to make a fusion of the techniques of the two algorithms previously explained, where the robot collects an image and the user, through a prompt, is able to generate a series of modifications on the initial image. The Knolling Bot framework [69] makes an encoding and decoding process through transformers of the elements detected by a camera and through a reordering process is able to establish an order to generate the order specified by the user.

Some other frameworks such as CACTI [70] or RoboAgent [71] focus not only on solving specific tasks, but also on task generalisation. To do so, they make use of techniques that allow data augmentation and training with this data. To do so, they make use of highly complex datasets both for the process of generating the movements and for the estimation of the correct positions of each of the elements to be reorganised.

A common factor in all of the works presented is that they are not completely autonomous. The user is still required to generate the final positions, to set the desired prompt or to validate the intake of a large amount of data from the environment. This makes those methods limited and unable to work in a truly autonomous way.

Some works in the state of the art have observed these limitations and have focused on trying to solve them. One of the most representative examples is the one presented in the SG-Bot framework [72] where, by means of an observation process, it extracts the elements on a table, and through the generation of automatic prompts, it is able to establish a scene graph where it establishes the reordering process and the relationship between the elements. Another representative example is the one presented in Dall-E-Bot [51], which, by means of camera-based element detection and without any specific training, is able to generate the position of the element.

Despite their correct operation, both works have different limitations. In the first case, the generation of graphs is associated with a series of relationships that may not be equal for all cases, and depending on certain objects in the domestic environment, they can have multiple interpretations. In the second case, for the inpainting process generation, Dall-E-Bot requires fixed elements such as a plate or tablet; therefore, it is not entirely reconfigurable. Additionally, in both methods, little importance is given to the manipulation process of the elements in the environment, and in both cases, they make use of a series of pre-programmed movements, making the relocation process less than optimal.

All these problems are addressed in our framework presented in this paper. As other state-of-the-art implementations, our framework is based on the use of different zero-shot learning methods, such as YOLO [24] and StableDiffusion [4], used to generate the final positions of the detected objects in the working environment. As a novelty, these zero-shot learning methods are combined with our own algorithms, such as an object orientation estimator, which determines the manipulation orientations of objects, an automatic prompt generator that uses this information to create descriptions of the final scenes, and a collision checker

that generates a list of the order of object manipulation to avoid collisions.

Additionally, for manipulation, we have developed an LfD algorithm based on the use of Task Parameterized Gaussian Mixture Models (TPGMM), which autonomously utilises only relevant information, thereby obtaining smooth movement paths that follow the learned human information. None of the works presented in the state of the art, such as SG-Bot [72] or Dall-E-Bot [51], on object rearrangement utilise these techniques in a combined manner.

Moreover, our method is completely autonomous, meaning it does not rely on any user-provided information. This characteristic is significant and distinguishes our work from others, as RoboAgent [71], which depend on the user selecting certain parameters for the specified task.

III. METHOD DESCRIPTION

To address the problem of object rearrangement by a robot, we present an algorithm that combines different artificial intelligence techniques based on diffusion models as well as Learning from Demonstration algorithms, allowing the entire process to be carried out in a way that resembles how a human would do it.

For this purpose, we propose to make use of a framework with different modules based on the prediction of the final positions and orientations using a method based on the observation of the environment by an RGB-D camera (using a top-down view of the environment perpendicular to the workspace, covering the workspace of the robotic arms of the models used), an imagination by zero-shot information image generation models and a Learning from Demonstration process used for the manipulation of the elements as a human would do.

The novelty of this framework that we have developed lies in the combination of different artificial intelligence techniques used in current robotics, as well as the creation of a proprietary learning algorithm that allows Learning from Demonstrations and relevant parameters such as position and orientation, determining how the robot should move in new situations. The modular pipeline presented is shown in Fig. 2.

A. OBSERVATION

This process starts with the acquisition of an top-view image $\mathcal{I}$ by the RGB-D camera (perpendicularly to the environment and covers the manipulator's workspace) where the robot will detect the elements placed on the table capturing the initial state $\mathcal{S}_o$. After this, the method will obtain in first instance the classes of the detected objects using YOLO v8. Additionally, a process of obtaining relevant geometric information (such as angle, direction and position in the image) created by us, that defines each of the objects of the classes separately will be performed. For this purpose, a mask generation process is applied and additionally, a handle detection process. This is necessary because the robot will act as a human manipulating the objects by these handles if they exist. This information will be sent to the next module for the process to continue running. This module and all its features are explained in depth in Sect. III-D

B. IMAGINATION

The data extracted from the image is received by the imagination module. With the poses received, we have developed a prompt generator that use the input information to generate a list of different prompts $\mathcal{P}$ using the quantities and different class objects detected for that purpose. These $\mathcal{P}$ prompts are passed to Stable Diffusion model which, through the information used for its training and without modifying the data. This generates a series of realistic images presenting different types of solutions on how the elements of the working environment can be rearranged. After this, we have developed an algorithm that automatically searches for the image that most closely resembles the $\mathcal{I}$. In order to do this, it will first search for images that have the same number of objects as the initial state. If it does not obtain any image that is exactly the same, it will look for the most similar image possible (with as few variations of external elements as possible) and then it will carry out a filtering process by means of an inpainting process that will eliminate these extra elements. The image generated by Stable Diffusion and selected and filtered by our algorithm will be used to establish the final positions and orientations of the final $\mathcal{S}_f$ state. This knowledge will be sent (along with the initial state data $\mathcal{S}_o$) to the trajectory execution platform. This module is fully explained in Sect. III-E

C. EXECUTION

To perform a correct rearrangement process, the objects in the $\mathcal{S}_o$ state must end up as the objects in the $\mathcal{S}_f$ state. The transition between the two states is not direct, as the rearrangement tasks are set up in a series of sub-tasks that allow the whole scene to be shaped correctly. To make this process in a human-like manner, we have developed a framework, by means of Learning from Demonstration, is able to perform each of the established sub-tasks. For this purpose, our algorithm based on TPGMM [73] has been created, to which a relevant frame selector has been added to optimise and improve the result for each of the actions and tasks to be performed. Prior to this process it is important to establish the optimal order of resolution, which is considered to be the order in which the objects between state transitions do not collide. For this purpose, we have implemented and algorithm that generates a logic order of the element before the manipulation. this algorithm select autonomously which is the best order to rearrange the list of objects to not have collisions between then when and object is placed in a final position.The Execution module and the algorithm for Learning from Demonstrations are fully defined in Sect. III-F. Once this Execution process is finished, the task is considered as successfully completed by obtaining

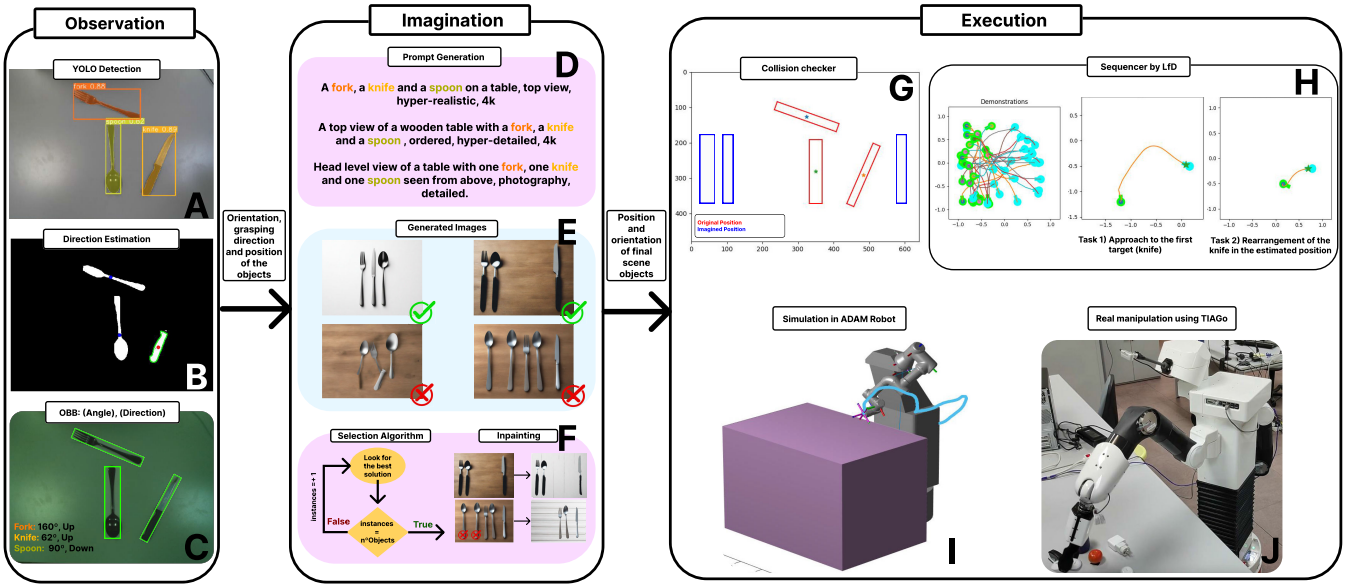

FIGURE 2. General scheme the method's functioning. The process goes: Observation → Imagination → Execution. Each of the modules send to the next information described in the image (text over the arrows).***Observation***, this module consist of object detection and localisation. The module start with and object detection (A) and a direction estimator (B and C) that gives us the position and direction to grasp. ***Imagination***, a prompt is constructed from the detected objects(D), which is used for the creation of the final scenes (E) using diffusion models. Then, the algorithm selects a matching scene, and inpaints the image to filter possible errors(F). ***Execution***, this module is composed by the collision checker (G) that generate the order to be executed the rearrangement task and the sequencer by LfD (H) that generates the path to be executed. After that, the movement information can be send them to the robot, either simulated (I) or real (J).

an organisation in the real environment equal to that obtained by the Imagination module, starting from the state obtained in the Observation module.

A number of assumptions have been taken into account for certain aspects of the rearrangement process. The detection of the handle assumes that it must have a different colour, otherwise the detection cannot be taken into account as a handle but as part of the item itself. This assumption has been made due to the robustness of our algorithm decrease considerably in this situation. As for the selection algorithm, even though it iterates until it finds an image that can be a suitable solution even if it is necessary to discard several elements of the image (*Approx Solution*). There is always a little chance to not find a feasible solution. It is assumed that executing the imagination module again with a different seed to generate new images solves this issue. Finally, in relation to the workspace, it is defined by the maximum reachability of the robotic platform used. The camera is located in a top view configuration that covers all this workspace.

D. OBSERVATION

When humans want to manipulate something the perception of the scene initiates the action, the purpose of this Observation module is to replicate this human ability. By using different computer vision techniques and deep learning models, it is possible to make an accurate object localisation and the prompt generation, that replicates the human perception. This process will be elaborated in detail in the following sections.

The first objective of the observation module is to correctly locate the objects in the scene, as without this information, the reorganization task cannot be completed. For proper manipulation and to carry out the rearrangement process, it is essential to accurately estimate the grasping point and orientation for each object in the environment. Considering that during this rearrangement task there are no changes in the scene, if this happens the task needs to be restarted. To achieve this, we have designed an algorithm that begins by processing the image and obtaining an instance segmentation of the scene for each of its elements. Subsequently, it extracts the Oriented Bounding Box (OBB) and concludes with the estimation of the object's direction, taking into account its centroid and the object's handle, when applicable (Fig. 3).

Instance segmentation is performed using a zero-shot CNN, specifically the state-of-the-art YOLOv8 network [24]. Since our task requires precise object detection, we use the weights of the *yolov8x-seg* model, which provides the best results. The input to YOLO is the image $\mathcal{I}$ of the initial state $\mathcal{S}_o$. Class identifications and segmented masks of the objects are provided in the output. The algorithm then proceeds with OBB extraction, using the *minAreaRect()* function from the OpenCV library [74]. This function finds the rotated rectangle of minimum area enclosing the segmented mask of the detected objects, providing information about the orientation and size of the object. However, this orientation is influenced by the object's direction (i.e., the usable part of the object defines the direction), so there is an additional step to refine this value. An example would be a knife where the

blade (the part used for cutting) defines the object's direction and modifies the orientation of the OBB.

To extract the object's direction, the first question is to determine if the object has a handle or not. Handle detection is done by dividing the object's mask into two groups, assuming that the handle and the rest of the object have different colours, since most of the objects with handle share this characteristic. The chosen algorithm for clustering is *K-means* in the *HSV* colour space, using *H* (Hue) and *S* (Saturation) channels, aiming to achieve robust detection independent of scene illumination. At this point, the algorithm makes a comparison: if there is a handle, the object's centre and the handle's centroid are compared. Otherwise, the object's centroid and its centre are compared. In the first case, if the handle's centroid is below the object's centre, the object's direction is upwards. In the second case, the direction is upwards when the object's centroid is above the centre.

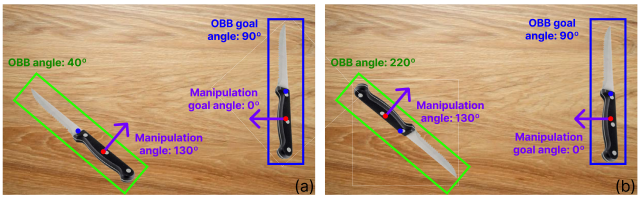

FIGURE 3. Example of object detection. Green OBB represents detected object and blue OBB represents final configuration. Blue point means centroid and red points means handle's centroid (grasping point). (a) Upwards blade example. The OBB angle (green) estimates the alignment as a function of the direction of the blade (upwards). The handling angle is obtained in this case by adding 90° to the OBB angle, (b) Downwards blade example. The OBB angle (green) estimates the alignment as a function of the direction of the blade (downwards). The handling angle is obtained in this case by subtracting 90° to the OBB angle.

Taking into account whether the object is upwards or downwards, along with the angle information provided by the OBB, it is possible to extract the actual angle of the object between 0 and 360 degrees. However, it should be noted that this angle is the one used to grasp the object, not the angle at which the movement path is calculated (denoted as manipulation angle, $\mathcal{H}_{ang}$). For this purpose, an additional adjustment of the angle measurement is made by applying a 90° clockwise or counterclockwise turn, depending on the object's position in the final state $\mathcal{S}_f$. To produce a smooth, human-like path, the start and end $\mathcal{H}_{ang}$ of the path must be opposite each other. A visual example of the object localisation process is presented in Fig. 3.

Simultaneously, this algorithm extracts the depth distance of the centroid of each object to get the $\mathcal{Z}$ coordinate of the 3D grasping point. In order to do this, the centroid points $\mathcal{C}_o$ are stored and used to find the distance between these points and the robot. This distance is measured by the Realsense D435, the RGBD sensor of the robot.

Finally the algorithm creates an array that contains all this information (class name, centroid's, size and angle) of the detected objects in the initial state $\mathcal{S}_o$. The grasping strategy

Algorithm 1 Object Localisation

Input: Image $\mathcal{I}$

Output: Detected objects data: (position $\vec{\mathcal{C}}_o$, OBB orientation $\vec{\mathcal{O}}_o$, manipulation orientation $\vec{\mathcal{H}}_{ang}$, Size, and Class Names)

1. Instance segmentation

Camera initialization
Load YOLO model: $Yolo_m$
Predict: $results \leftarrow Yolo_m.predict(\mathcal{I})$

2. Extract OBB for each object

$masks \leftarrow results.mask()$
for *mask* in *masks* **do**
  Binarize image with mask
  $contours \leftarrow mask.xy()$
  $OBB_{ang}, Obj_{centre}, Obj_{centroid} \leftarrow minAreaRect()$
  $H_{mask}, H_{centroid} \leftarrow ObjHandle(mask)$
  if H_{mask} **then**
    $\vec{\mathcal{C}}_o \leftarrow H_{centroid}$
    if $H_{centroid} > Obj_{centre}$ **then**
      $Direction = Upwards$
    else
      $Direction = Downwards$
    end if
  else
    $\vec{\mathcal{C}}_o \leftarrow Obj_{centroid}$
    if $Obj_{centroid} > Obj_{centre}$ **then**
      $Direction = Downwards$
    else
      $Direction = Upwards$
    end if
  end if
  $\vec{\mathcal{O}}_o \leftarrow AngleRefine(OBB_{ang}, Direction[i])$
  $\vec{\mathcal{H}}_{ang} \leftarrow AngleMan(\vec{\mathcal{O}}_o, Direction[i])$
end for

3. Depth estimation for grasping point

$Z \leftarrow DistanceMeasurement(\vec{\mathcal{C}}_o)$
return $\vec{\mathcal{C}}_o$,$\vec{\mathcal{O}}_o$, $\vec{\mathcal{H}}_{ang}$, $Size[i]$, Z and $ClassNames[i]$

consists of approximating straight to the object from a top position while taking into account the centroid point, its size and angle.This process is repeated one more time with the image selected as the final state $\mathcal{S}_f$. The complete algorithm is presented in Alg. 1. With the detected elements, the method is ready to skip to the next process, the imagination module.

E. IMAGINATION

After the observation, humans imagine which can be the best solution. In this module we try to replicate this human ability to create a final state that solves the rearrangement task. Here our framework generates the image of the final scene where the objects should be positioned in an ordered way (within the different ways of placing ordered elements that may exist). For that purpose, the algorithm generates a list of prompts that makes use of the detected objects and their quantities.

This prompts synthesises the necessary information to be rearranged in the final environment.

To achieve this, the images are generated by Stable Diffusion XL [4] a Latent Diffusion Model that is trained with a huge set of images and captions and which uses as input the prompts generated from the initial scene. The module consists of two parts, the first one is the selection algorithm that finds the best result between all the generated images, while the second one, is the inpainting process that is used to filter noise and irrelevant objects of the selected image. Fig. 4 presents an example of the images that can be obtained.

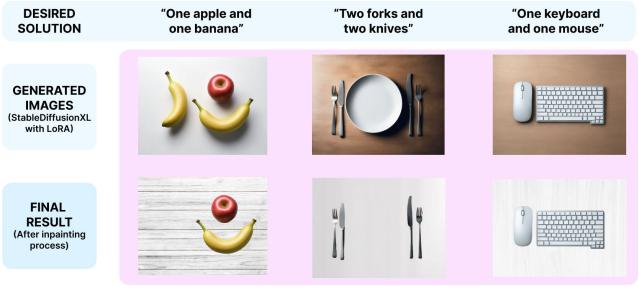

FIGURE 4. Example of images obtained with the Imagination module. There are three different scenes, cutlery, fruits and desktop, with two images for each. Top images are the output of the selection algorithm, which choose between all the generated images for the best solution. Bottom images are the output of the inpainting process done to filter extra information that do not fit with the desire solution.

1) PROMPT GENERATION

The first objective consists of a prompt generation that describes the scene. For that purpose, we have developed an algorithm that works like a *fill in the gap* exercise. After the object detection, our method has information about the class names of the objects detected, with this information it is possible to extract the count of objects that belong to the same class. Then, the algorithm makes the *fill in the gap* composition using the class names and its count with predefined sentences and adjectives that came from a database. Fig. 5 presents a graphical example of that process.

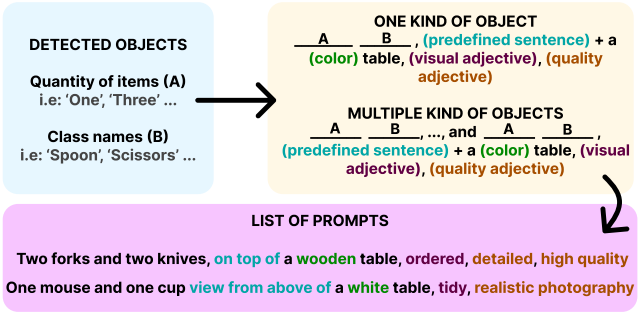

FIGURE 5. Visual scheme of the prompt generation process. The class names and count of the detected objects are the input. Then, our algorithm does a *fill in the gap* composition to generate the prompts. At the end, the algorithm returns a list that contains all these prompts.

It should be noted that this database can be modified by the user, and adapt the application to other setups or environment specifications. Moreover, these prompts are combined with a set of negative prompts (that are taken into account in order not to generate elements or configurations that impede the correct understanding of the scene) in order to improve the final result.

2) SELECTION ALGORITHM

Although Stable Diffusion (SD) generates multiple valid solutions of organized environments, it sometimes produces images that are not entirely suitable for the rearrangement task. These images may contain elements that overlap in the image, making object detection difficult. SD can also generate unwanted duplicate elements or additional elements that were not explicitly requested by the input prompt. This is why an image selection algorithm has been developed to obtain the option that best matches and adapts to the prompt specified by the algorithm. The process is divided into two steps. The first step is based on searching for an exact solution (i.e., with the exact same number of elements). If none of the images generated by Stable Diffusion can generate such an exact solution, the algorithm then searches for images with the minimum possible difference.

The selection algorithm begins by iterating over each image generated by Stable Diffusion $\mathcal{I}_f$, searching for an exact match using a *checkExact* function. Within this function, object detection is performed using YOLOv8 on the study image, generating masks for each of the objects in that image. After this, we obtain a list of objects that is compared with the original list of objects obtained in the initial image $\mathcal{I}$. In case they exactly match in both the number of objects and class identities, it is considered an exact match, indicating that Stable Diffusion has generated an image with the same objects as the required image. If this occurs, the algorithm follows the same process described in Sect III-D, obtaining all the necessary information for the motion generation process through imitation learning. If the two lists do not match in length or have different identifiers for their elements, the algorithm continues iterating through the rest of the generated images to find a correct option. If, after iterating through all the images, there is no solution, it moves to the second part of the process, where it looks for an image with the minimal possible variations compared to the initial image $\mathcal{I}$.

To achieve this, it uses the *checkAprox* function, which iteratively searches through all images, incrementing the length of the list by one. For example, if it is searching in a list of dimension 3, consisting of a fork, a knife, and a spoon, the algorithm looks for an image that has an additional element, making it a dimension 4 list. The additional object can be a repeated element, in which case the algorithm only takes into account the mask of the first detected element and the rest of the duplicated objects are discarded, or a different one, which is not taken into account. The first image that meets this criterion is chosen as the most approximate option for $\mathcal{I}_f$. This process is iteratively repeated indefinitely. The algorithm tends to converge with at least one approximate solution in the vast majority of cases. Since it starts by looking

for exact solutions and then slightly varying approximate solutions, the algorithm iterates quickly, allowing for an efficient search in terms of time.

After this process, and once the image has been selected as valid (as the ones presented in Fig. 4), the information retrieval process for the required data is carried out, followed by a filtering process using an inpainting process to obtain an image with exactly the same characteristics as those required by the input prompt.

3) INPAINTING

Once the selection algorithm is finished, the imagination module continues with an inpainting process to filter all the non-relevant objects and noise that can exists in the selected image. This step is necessary due to a YOLO limitation caused by the number of classes that is able to detect, for example a plate or noise elements created by StableDiffusion. The inpainting process is done with the *cv2.inpaint()* function of OpenCV using the mask extracted with YOLO and one image randomly selected from a database that can be modified by the user. A visual result of this step can be seen on Fig. 4.

After completing the inpainting process, it is necessary to rescale the image generated by Stable Diffusion to maintain the aspect ratios with the image of the initial scene. This process allows for easier alignment between both images, thus reducing the potential source of error in the relationship between the real and imagined images. When this process is done, the framework is ready to run the execution module, extract the order of movements with the collision checker and the path needed to complete the rearrangement task.

F. EXECUTION

Once the initial and final configuration of each of the objects in the environment is known, it is necessary to carry out the process of physical manipulation of these objects. To do this, the execution module developed first estimates the correct order to solve the task and then, by means of an algorithm based on the mimicry of human movements through the learning of different trajectories, performs the execution of the different elements of the environment.

1) COLLISION CHECKER

Before carrying out the reordering process of the elements, it is necessary to establish that the movements to be performed are possible and do not generate collisions. For this, we have designed an algorithm that, before the execution of the rearrangement orders, iteratively checks each of the final locations that the objects will have (Fig. 6).

Our method generates the execution order that establish which objects will move first, checking that the estimated final position of each of the objects does not collide with another object in the scene. To achieve this, having the data obtained from the initial state $\mathcal{S}_o$ and final state $\mathcal{S}_f$, we evaluate each of the possible actions to be taken. In this process, we use the size of the Oriented Bounding Box (OBB) of each of the masks for individual objects in both $\mathcal{S}_o$ and $\mathcal{S}_f$. These OBBs are positioned in their initial state, where the centroid $\mathcal{C}_o$ establish the position and the orientation $\mathcal{O}_o$ establish the angle of the mask. The initial elements follow a predetermined order based on the detection order. Starting with the first element on the initial list, it is placed at the required final position $\mathcal{C}_f$ with the final arrival orientation $\mathcal{O}_f$. After positioning it, collision is checked by examining whether the OBBs of the moved element and the remaining elements in the list intersect with each other, or if the repositioned element is placed within another element in the list. If this does not occur, the algorithm considers the positioning as correct, leaves the checked element in its position in the list, and proceeds with the rest of the elements in the list.

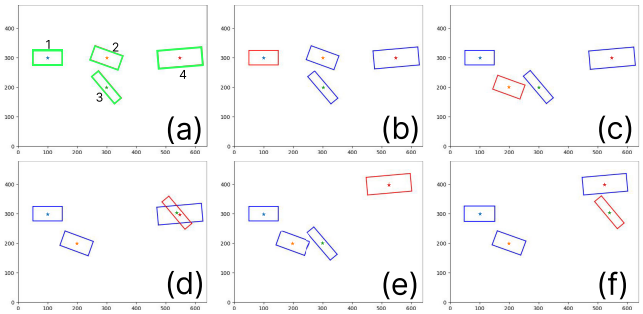

FIGURE 6. Collision Checker execution. In (b-f), the red object indicates the desired final pose and the blue ones the rest of the elements with which it is compared. (a) Initial state with an specific ordered list $\mathcal{L} = [1, 2, 3, 4]$ estimated by the ordered detected with YOLO, (b) Checking object 1: no collision. The object is left in the final position for further comparisons (in this case coincides with the initial position), (c) Checking object 2: no collision. The object is left in the final position for further comparisons, (d) Checking object 3: collision. The object is not moved from its initial position for further comparisons and is placed at the end of the list to move the other elements to ensure the possible solution, (e) Checking object 4: no collision. The object is left in the final position for further comparisons, (f) Re-checking object 3: no collision. The object is re-evaluated, ensuring that there is no collision as all other objects have been moved. Final list $\mathcal{L} = [1, 2, 4, 3]$.

In case of a collision, the object is placed at the end of the list since the algorithm considers it necessary to reorder the other object first to avoid the collision. The algorithm will continue iterating through the rest of the objects until it comes back to reevaluate the collision of that first object. The process will be repeated until the entire list is reordered, obtaining an order in which there are no collisions between objects. The complete process of operation is shown in Fig. 7 applying the algorithm in a real environment for the rearranging of 3 different objects: scissors, a remote, and a cell phone.

First, the algorithm receives an initial list generated by the detection order of YOLO. This list may be correct or not, and this is verified with the collision check. The algorithm first evaluates the order of the list given by YOLO (in this case, cell phone, scissors, and remote) and attempts to move the objects to the final position (obtained from the image generated by Stable Diffusion) one by one following this

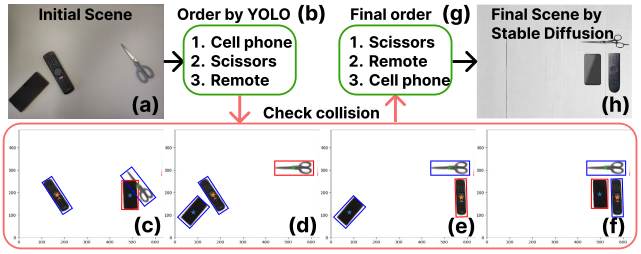

FIGURE 7. Example of the execution of the reordering algorithm by detecting objects in the real environment (a) using YOLO, to obtain the final positions generated by Stable Diffusion. The list obtained with YOLO (b) is used to check the collision order (c-f). Once the algorithm selects the order (g) it is send to the manipulation module to reorder the final scene as the one created with Stable Diffusion (h).

order. If the object, when moved (red-coloured OBB), to the final position does not collide, the movement will be possible, and therefore, its order in the list will not be modified. On the contrary, if there is a collision, the object that has been attempted to move will be placed at the end of the list to rearrange the rest of the elements before it.

The list given by YOLO indicates that the first element to be evaluated is the cell phone. This object is evaluated in its final position, checking for collisions with the rest of the objects. Since there is a collision with the scissors, that movement is not possible until the scissors are displaced, so the cell phone will not move from its initial position and will be rearranged at the end of the list. This process is iterative, so it will be repeated for each of the objects in the environment that need to be rearranged.

2) PROGRAMMING BY DEMONSTRATION WITH TPGMM

The task-parameterized Gaussian mixture models (TPGMM) [73] have been proposed in recent decades as an approach to robot programming through demonstration. In this process, TPGMM models, probabilistically encode the relevance of candidate frames of reference, which can change during the task. This relevance also considers the learning of orientation data along with position data, i.e., the complete pose [75], allowing for a much more comprehensive parameterization of tasks. In the field of learning by demonstration in robotics, several works based on this idea have been developed, as presented in [15] where has been used to encode the complete pose of the end-effector of a soft robot, and in [76], in which a TPGMM synthetic data generation algorithm is presented to cope with partial availability of task parameters.

In TPGMM, the task parameters are treated as P coordinate systems, defined at time step t by $\{b_j, A_j\}_{j=1}^{P}$, where b_j and A_j represent, respectively, the origin of the j-th reference frame and a set of basis vectors $\{e_1, e_2, \ldots\}$ forming the transformation matrix $A = [e_1 e_2 \ldots]$. All task parameters are specified in advance by the operator, according to the task and based on the operator's prior knowledge. In this way, the user decides where to establish and specifies that all chosen parameters are relevant for solving a specific task.

Each of the demonstrations, denoted as $m \in 1, \ldots, M$, contains a quantity of T data points of dimension D $\{\xi\} \in \mathbb{R}^{D\times T}$, which are encoded in P different reference frames, resulting in a third-order tensor dataset: $\mathbb{R}^{D\times T\times P}$. This dataset is composed of P trajectory samples projected onto P candidate frames, corresponding to matrices composed of D-dimensional observations at T time steps. The parameters of the TPGMM model are defined as $\{\pi_i, \{\mu_i^{(j)}, \Sigma_i^{(j)}\}_{j=1}^{P}\}_{i=1}^{K}$, where π_i are the mixing coefficients, $\mu_i^{(j)}$ and $\Sigma_i^{(j)}$ are the centre and covariance matrix of the i-th Gaussian component in frame j in a TPGMM with K components.

The learning of TPGMM model parameters involves maximising the log-likelihood under the constraint that data in the reference frames originate from the same source. This leads to the Expectation-Maximization (EM) algorithm (defined by Equation. 1, 2, 3) for iteratively updating the model parameters until convergence. In these equations, the superscript t represents EM iterations. The initialisation of model parameters is done using a k-means procedure.

E-step:

$$\gamma_{t,i} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}(X_j^t | \mu_i^{(j)}, \Sigma_i^{(j)})}{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{P} \mathcal{N}(X_t^j | \mu_k^{(j)}, \Sigma_k^{(j)})} \tag{1}$$

M-step:

$$\pi_i = \frac{\sum_{t=1}^{T} \gamma_{t,i}}{T}, \quad \mu_i^{(j)} = \frac{\sum_{t=1}^{T} \gamma_{t,i} X_t^{(j)}}{\sum_{t=1}^{T} \gamma_{t,i}} \tag{2}$$

$$\Sigma_i^{(j)} = \frac{\sum_{t=1}^{T} \gamma_{t,i} (X_t^{(j)} - \mu_i^{(j)})(X_t^{(j)} - \mu_i^{(j)})^T}{\sum_{t=1}^{T} \gamma_{t,i}} \tag{3}$$

The learned model can be used to reproduce new trajectories for new situations (new positions and orientations of reference frames), thus generalizing the learned optimal model. To perform this process, in the first step, the model retrieves a Gaussian Mixture Model (GMM) at each time step t by calculating the product of linearly transformed Gaussians:

$$\mathcal{N}(\mu_{t,i}, \Sigma_{t,i}) \propto \prod_{j=1}^{P} \mathcal{N}(A_{t,j}\mu_i^{(j)} + b_{t,j}, A_{t,j}\Sigma_i^{(j)} A_{t,j}^T) \tag{4}$$

Afterward, Gaussian Mixture Regression (GMR) [77] is used to reproduce the new trajectory based on the learned model for the new task parameters. The use of GMR allows leveraging the joint probability density function of the data (input and output) described by TPGMM. The sets of dimensions spanned by the input variables ($\mathcal{I}$) and output variables ($\mathcal{O}$) are defined, respectively, at each time iteration t. As a result, the data point ξ_t is decomposed into input data $\xi_t^{\mathcal{I}}$ and output data $\xi_t^{\mathcal{O}}$. Consequently,ξ_t, μ_i, and Σ_i can be redefined as:

$$\xi_t = \begin{bmatrix} \xi_t^{\mathcal{I}} \\ \xi_t^{\mathcal{O}} \end{bmatrix}, \quad \mu_i = \begin{bmatrix} \mu_t^{\mathcal{I}} \\ \mu_t^{\mathcal{O}} \end{bmatrix}, \quad \Sigma_i = \begin{bmatrix} \Sigma_i^{\mathcal{I}} & \Sigma_i^{\mathcal{I}\mathcal{O}} \\ \Sigma_i^{\mathcal{O}\mathcal{I}} & \Sigma_i^{\mathcal{O}} \end{bmatrix}. \tag{5}$$

The temporal GMM deduced in Equation. 4 encodes the joint distribution $P(\xi^{\mathcal{I}}, \xi^{\mathcal{O}}) \sim \sum_{i=1}^{K} \pi_i N(\mu_i, \Sigma_i)$ of the

dataset ξ. If we apply this definition for each reproduction step t, $P(\xi_t^{\mathcal{I}}|\xi_t^{\mathcal{O}})$ can be computed as the conditional distribution:

$$\mathcal{P}(\xi_t^{\mathcal{I}}|\xi_t^{\mathcal{O}}) \sim \sum_{i=1}^{K} h_i(\xi_t^{\mathcal{I}})\mathcal{N}(\hat{\mu}_i^{\mathcal{O}}(\xi_t^{\mathcal{I}}), \hat{\Sigma}_i^{\mathcal{O}}), \tag{6}$$

where:

$$\hat{\mu}_i^{\mathcal{O}}(\xi_t^{\mathcal{I}}) = \mu_i^{\mathcal{O}} + \Sigma_i^{\mathcal{O}\mathcal{I}}\Sigma_i^{\mathcal{I}-1}(\xi_t^{\mathcal{I}} - \mu_i^{\mathcal{I}}), \tag{7}$$

$$\hat{\Sigma}_i^{\mathcal{O}} = \mu_i^{\mathcal{O}} - \Sigma_i^{\mathcal{O}\mathcal{I}}\Sigma_i^{\mathcal{I}-1}\Sigma_i^{\mathcal{I}\mathcal{O}}, \tag{8}$$

and

$$h_i(\xi_t^{\mathcal{I}}) = \frac{\pi_i\mathcal{N}(\xi_t^{\mathcal{I}}|\mu_i^{\mathcal{I}\mathcal{O}}, \Sigma_i^{\mathcal{I}\mathcal{O}})}{\sum_{k=1}^{K}\pi_k\mathcal{N}(\xi_t^{\mathcal{I}}|\mu_k^{\mathcal{I}\mathcal{O}}, \Sigma_k^{\mathcal{I}\mathcal{O}})} \tag{9}$$

Following this process, the estimated $\hat{\xi}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\xi_t^{\mathcal{I}})\mu_i^{\mathcal{O}}$ can be applied as a command for the position of the robot's end-effector to follow. The covariance information could also be used to determine the stiffness of the robot's end-effector. This allows for controlling the generation of movements of a robotic manipulator through learning an optimal movement model using TPGMM and a set of user-provided data, along with its relevant parameters.

3) MODIFIED TPGMM ALGORITHM BASED ON RELEVANT TASK PARAMETERS

The use of TPGMM is highly widespread in the field of learning tasks through human imitation. Despite this, it is not a perfect method, and in recent years, methods based on this idea have been developed to optimize the algorithm. The main limitations of the method arise from the user's choice of relevant parameters. Manual selection can lead to suboptimal learning of movement policies, as the user may choose task-irrelevant frames for solving a specific task. Another major limitation is the presence of redundant frames. Redundancy negatively affects the selection of an optimal model since TPGMM is unable to distinguish between two possible solutions to the same problem, resulting in movements that attempt to consider both solutions equally at the same time.

To address those problems, works such as those presented in [78] and [79] have sought to select relevant frames by generating an importance score based on the covariance matrices of each frame. Frames with a value below a threshold are categorized as irrelevant frames.

Other algorithms [80] leverage this idea and rely on generating probabilistic weights to categorize the relevance of frames, allowing for the filtering of necessary information. In both cases, the algorithm focuses solely on filtering out irrelevant data, potentially overlooking redundant frames that could negatively impact solutions.

To tackle this, there are works, like those presented in [81], that, using a reinforcement learning process, estimate relevant and non-redundant frames that minimize a specified cost function. This process is slow and cannot be applied in real-time. Other works such as [82] introduce a visual selection of relevant frames based on the morphology of the object, but it requires slow and costly offline training using reinforcement learning techniques to obtain a applicable model.

To address the issues of irrelevant and redundant frames, we have created a proprietary algorithm based on the use of Kullback-Leiberg Divergence (KL divergence) [16]. As a novelty, we have implemented a cost function based on this widely used mathematical tool in pattern recognition that allows the method to establish the similarity between two probability density functions denoted as $f(x)$ and $g(x)$, such that:

$$D(f||g) = \int f(x)\log\frac{f(x)}{g(x)}dx \tag{10}$$

Applying that equation for two gaussians $\hat{f}$ and $\hat{g}$, the KL divergence has a closed expression where $\log\frac{|\sum_{\hat{g}}|}{|\sum_{\hat{f}}|}$ represents the difference in variability is between the two distributions through the covariance matrices, $Tr[\sum_{\hat{g}}^{-1}\sum_{\hat{f}}]$ represents the trace of the multiplication of the inverse of the covariance matrix of $\hat{g}$ by the covariance matrix of $\hat{f}$, δ denotes the distribution dimension (expressed by the number of characteristics) and $(\mu_{\hat{f}} - \mu_{\hat{g}})^T\sum_{\hat{g}}^{-1}(\mu_{\hat{f}} - \mu_{\hat{g}})$ allows to estimate the difference between the mean of each distribution. Our algorithm uses this equation to estimate, through a cost function, the values of relevant frames and also eliminate data redundancy. The complete algorithm is presented in Alg. 2.

$$D(\hat{g}||\hat{f}) = \frac{1}{2}[\log\frac{|\sum_{\hat{g}}|}{|\sum_{\hat{f}}|} + Tr[\sum_{\hat{g}}^{-1}\sum_{\hat{f}}] - \delta + (\mu_{\hat{f}} - \mu_{\hat{g}})^T\sum_{\hat{g}}^{-1}(\mu_{\hat{f}} - \mu_{\hat{g}})] \tag{11}$$

The algorithm starts with the initialisation of parameters providing information about the number of samples (*nbSamples*), the amount of path data (*nbData*), and the number of frames (*nbFrames*). Depending on the type of movement (left to right or vice versa) and the distance between the frames detected by the algorithm, a specific dataset is used for each movement. To achieve this, four different datasets have been recorded by the human, each with 25 demonstrations. The datasets are divided into two groups: data from left to right and from right to left. Within these groups, there is a further division into two subgroups: those considered long movements ($nbData = 200$) and short movements ($nbData = 50$). The use of oriented datasets (name we have given to those datasets that uniquely store a series of movements depending on the starting point of their initial frame) greatly reduces errors, especially related to redundant movements, thereby minimising a potential source of error. Choosing two movements based on distances helps avoid problems derived from the operation of TPGMM in generating the solution. This algorithm always generates a path of length *nbData*. Therefore, if we use paths with 200 points for very close points, the generated solutions try to introduce 200 path points into very confined spaces,

Algorithm 2 Optimization of Task-Parameterized Movements

Input: $\mathcal{S}_o$ data: (position $\vec{\mathcal{C}}_o$ and initial manipulation angle $\mathcal{H}_{oang}$), $\mathcal{S}_f$ data re-ordered: (position $\vec{\mathcal{C}}_f$, and final manipulation angle $\mathcal{H}_{fang}$), distance d and direction θ
Output: Movement path: $\mathcal{M}$

1. Initialization and data aggregation
Initialization: $nbSamples, nbFrames, nbData$;
Demo selec: $nbData, nbSamples \leftarrow dataset(d, \theta)$;
for $t = 1 \ldots nbData$ **do**
  for $m = 1 \ldots nbSamples$ **do**
    TPs from users: $slist \leftarrow \{b_j, A_j\}_{j=t}^{nbFrames}$
  end for
end for
2. Model fitting
Set number of Gaussianas $nbStates$
$mTPGMM \leftarrow initTPGMM(nbStates, nbFrames)$
$mTPGMM \leftarrow \{\pi_i, \{\mu_i^{(j)}, \Sigma_i^{(j)}\}_{j=1}^{nbFrames}\}_{i=1}^{nbStates}$
3. Extraction of irrelevant and redundant frames
New paramaters P_o & P_f: $nslist \leftarrow \{b_j, A_j\}_{j=1}^{nbFrames}$;
EM & GMR: $\mathcal{M}_{trial} \leftarrow repro(nslist, mTPGMM)$;
for $m = 1 \ldots nbSamples$ **do**
  for $s = 1 \ldots nbSates$ **do**
    KL div.:$kl \leftarrow D(mTPGMM(m, s)||\mathcal{M}_{trial}(m, s))$
  end for ▷ Eq. 11
end for
Filter redundant frames (P_{red}) by multicollinearity
Specified relevant frames (P_r) and irrelevant frames (P_i)
Update: $newmTPGMM \leftarrow \{\pi_i, \{\mu_i^{(j)}, \Sigma_i^{(j)}\}_{j=1}^{P_r}\}_{i=1}^{nbStates}$;
4. Reproduction
EM & GMR: $\mathcal{M} \leftarrow repro(nslist, newmTPGMM)$;
return $\mathcal{M}$

resulting in sub-optimal solutions with many curvatures that can negatively affect the generation of movements on the actual robot. To address this issue, four datasets covering all options in our environment have been generated. A series of practical demonstrations, to prove the efficiency, is presented in detail in Sect. IV-B.

After the selection based on distance and direction of movement (data extracted in previous modules), we read the data previously collected by users and store it in *slist*. Data collection in this algorithm can be done kinesthetically (moving the robot arm directly and collecting data from the end effector) [83], through a proprietary algorithm for detecting and recognising human movements through an RGBD camera [84], [85] or by using a program that allows generating data by painting a path with the mouse on the screen.

Once this process is finished, the algorithm performs model fitting, obtaining an initial model that uses all the frames from the demonstrations as relevant. After this model fitting process, the elimination of irrelevant and redundant frames is carried out. To do this, we use the initial model and, using the data obtained by the camera and Stable Diffusion, establish the new frames on which we generate an initial solution. This solution, $\mathcal{M}_{trial}$, is the one we will use to estimate the KL divergence. To do this, first, we compare each of the Gaussians that make up the solution for the new frames with each of the Gaussians from the frames used to generate the model.

The result of this calculation can give two outcomes: (a) if KL returns that the covariance matrix is not positive definite. This error is derived from the multicollinearity of the data, which indicates a high correlation between the studied Gaussians. If this is the case, we are dealing with a redundancy in the frames, meaning that when comparing each frame with each other, we find that the influence on the result we are generating is the same, but we have two distinct solutions for the same case. In this case, those frames are filtered as redundant frames (P_{red}). (b) if KL returns a numerical value. The greater the dependence between the Gaussians, the closer the KL divergence value will be to 0 (0 being exactly the same Gaussian). These values are summed up to obtain an average value for each demonstration based on the relevance of the solution for the new frames. To distinguish the relevance between relevant frames (P_r) and irrelevant frames (P_i), a variable threshold is used that aims to reduce values above 60% of the maximum value for each case. This threshold has been obtained experimentally to maintain a balance between relevant and irrelevant frames, regardless of the number of added demonstrations. This way, we keep (normalized between 0 and 1) values below 0.6, with 1 normalized over the maximum value of the values for each solution path. This allows the threshold to adapt automatically without depending on the user.

After obtaining the relevant frames, we relearn the model, but this time considering only those frames. We generate the solution for the same points detected by the camera and generated by Stable Diffusion, resulting in the final path $\mathcal{M}$. In Sect. IV-B, a series of experiments are presented to verify the efficiency of the Learning from Demonstrations algorithm proposed in this work.

IV. EXPERIMENTAL RESULTS

A. STABLE DIFFUSION EXPERIMENTS

In this section the imagination module will be tested in order to check its performance. To this end, there are two main experiments that make a quantitative analysis. First, the selection of the mode used to generate the images will be discussed. Second, a test to get the minimum execution time maintaining a certain successful rate. These experiments were made in a computer with the Intel i9 12900K CPU and a NVIDIA 3080 GPU with 10GB of vRAM.

The selection of the mode used in the deep learning model during the image generation is a crucial decision, because it affects the final result and the execution time. In our proposal, we use two different modes using *base model* and with *LoRA* acceleration [86]. The comparison consist of the generation of 100 images, using the same hardware and checking the

solutions provided. The results of the test are collected in Table 1. In this test, the images are classified into exact images, approx images and invalid images. *Exact images* refers to an image that contains the same objects that the objects detected on the initial state ($\mathcal{S}_o$). Otherwise, *Approx images* refers to an image that presents a solution containing the objects detected in the initial state ($\mathcal{S}_o$), with one of those objects repeated. Finally, *Invalid images* will be those in which the required elements are not generated (for example, missing any of the detected objects) or those in which the generated images are not realistic enough and cannot be correctly detected by the algorithm. For this experiment we have defined that a correct solution is the union between *Exact and Approx images*. In Table 1, can be seen that the successful rate is similar in both cases, around a 30%, that means that 1 of every 3 images provide a valid solution. With LoRA, the algorithm generates more exact images and the execution time is significantly lower, as much as 3.5 times.

TABLE 1. Comparison of the success rate of the modes used to generate 100 images with each mode.

	Base	LCM-XL
No. Exact Images	10	14
No. Approx Images	20	18
Success rate (%)	30	32
Execution time (s)	2246	758

The main objective of this work is to achieve the best solution in the shortest amount of time. For this purpose, in the second test with LoRA acceleration we make a few images generation with different objects and different batch size. Fig. 8 presents a bar chart that collects the results obtained. In this test there are four scenes, *Scene 1* correspond to *One spoon, one fork and one knife*. *Scene 2* refers to *Two forks and two knives*. *Scene 3* is for *One mouse and one apple*. And *Scene 4* correspond to *Two apples*. About the configuration, is the same throughout the test except in the **number of inference steps** used to generate the images.

This value increases from one image to the next according to an step that can add 1 or 2 to the inference steps used to generate the image. Also there are three batch sizes: 30, 20 and 10. The execution time is similar in all the scenes, it only depends on the batch size and the step used to generate the images. For a batch size of 30 images, the execution time takes about 126*s* with a step of 1, while with a step of 2 takes 141*s*, being the longest. In the case of a batch size of 20 images, the execution time with a step of 1 is 80*s* or 87*s* using a step of 2. For a batch size of 10 images, the execution time is the shortest, taking about 39*s* with a step of 1, and 40*s* with a step of 2. Analysing the results presented in Fig. 8, can be concluded that work with a small batch size for the image generation is better. The success rate

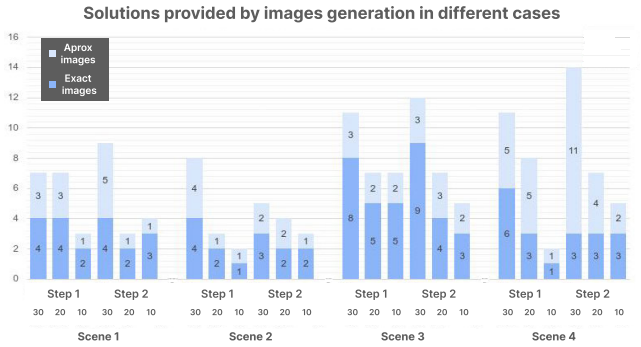

FIGURE 8. Summary graph with image generation results with different objects and batch sizes.

outreach a value over 38% in almost every cases and the inference time is the shortest. From the results obtained we conclude that the success rate does not increase in proportion to the number of images generated. Using a batch size greater than 10, we observe that the number of solutions increases slightly while the execution time is considerably longer. That makes the execution time-success rate ratio less efficient. All these rates are collected in Table. 2. The results provided generating only 10 images with an step of 2 for increasing the inference steps between images fits for our proposal. Since our objective is to have only one valid solution for the rearrangement task, and the average success rate among the four scenes with this configuration is 42%, providing in almost all cases around 4 possible solutions we use a batch size of 10 images.In other applications, where multiple solutions are necessary and execution time is not a constraint, the best batch size is the highest. This is because it provides a larger number of valid solutions, even though the overall success rate decreases slightly.

B. LEARNING FROM DEMONSTRATIONS EXPERIMENTS

In this section, the advantages of the presented algorithm over the widely used classical Task Parameterized Gaussian Mixture Models (TPGMM) have been empirically tested through different experiments. Firstly, in Fig. 9, the results of applying the described algorithm considering all parameters as relevant and considering the relevant parameters after filtering through the study of their Kullback-Leibler divergence are presented. The experiment has been conducted with the same dataset for both cases. To assess effectiveness, 100 examples have been generated, taking into account initial and final positions within a workspace ranging from [-1.2, 1.2] for both axes and orientations ranging from symbol 0° to 360°. Our algorithm shows better results in the execution of the generated tasks, as it is able to achieve higher efficiency in terms of obtaining feasible solution paths. These paths are those that do not generate movements beyond the workspace limits, implying reaching the physical range limits of the arm's movement and considering the input and output frame orientations. Our algorithm is capable of generating a greater number of correct data, achieving a success rate of 96%

TABLE 2. Success rates of image generation sets in different scenes and conditions.

Scene	Step	Set	Success rate(%)	Scene	Step	Set	Success rate(%)
Scene 1	1	30	23,3	Scene 2	1	30	36,7
		20	35,0			20	35,0
		10	**30,0**			**10**	**70,0**
	2	30	30,0		2	30	40,0
		20	15,0			20	35,0
		10	**40,0**			**10**	**50,0**
Scene 3	1	30	26,7	Scene 4	1	30	36,7
		20	15,0			20	40,0
		10	**20,0**			**10**	**20,0**
	2	30	16,7		2	30	46,7
		20	20,0			20	35,0
		10	**30,0**			**10**	**50,0**

efficiency in the tests conducted, while the experiment carried out with the classical TPGMM has an efficiency of 47%. This can be observed in more detail in Figure. 9c and Fig. 9d, it is apparent how the classical TPGMM is unable to generate a valid solution for those frames, while our algorithm is capable of providing a correct, executable, and natural solution, mimicking what a human could perform. In cases where TPGMM can generate a valid solution (Fig. 9b), our algorithm tends to improve the results by generating much smoother and more natural solutions.

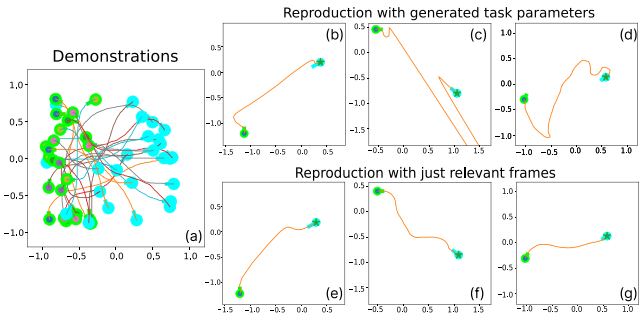

FIGURE 9. Demonstration of optimization of task-parameterized movements.(a) Demonstrations generated by the user. Green circles represent the initial frame and blue circles the final frames, (b-g) Examples of execution of our algorithm. Top images (b,c and d) represents the different cases using classic TPGMM. It can be seen that some cases don't generate a correct path. Bottom images (e,f and g) represents the same frames after our filtering process to eliminate irrelevant and redundant frames. Our algorithm is able to generate a correct path where classical TPGMM can't and also is able to filter the paths to generate smoother solutions.

The algorithm presented in this work has two additional characteristics related to the selection of demonstration data: the choice of movement orientation and the distance of the generated path. When working with environment rearrangement processes, we have considered the division into two major groups: left-to-right movements and right-to-left movements. We categorize movements based on the zone from which they start (left or right), regardless of whether they end in the opposite zone or the same. The division of demonstration data into two groups and their selection through the algorithm (Alg. 2) shows improvements in data filtering and avoids problems arising from the quantity of data with opposite movements, which generates highly redundant data. If we teach parametrized tasks to perform a movement in one direction and also a similar movement oriented in the opposite direction, the algorithm is unable to recognize it as a redundant frame and therefore cannot be filtered unless it is irrelevant for task resolution. It is due to this issue that the proposal was made to divide into two datasets and choose which one to use based on the frames and where they are detected. The efficiency of this division can be observed in Fig. 10.

To perform the comparison, 100 tests (for each case) have been conducted with frames with random positions and orientations within the workspace (similarly to the previous experiment). The comparisons were made by recording three different datasets: one with left-to-right oriented movements, another with right-to-left oriented movements, and finally a dataset with mixed movements. To check the efficiency of the dataset orientation choice, the same number of data points was used in all cases (25 in each dataset). For the tests of left-to-right oriented movements, it can be observed that using a dataset with oriented data (Fig. 10a) generates correct data with a success rate (estimated similarly to the previous case) of 96%. If we look at the case with the mixed dataset (Fig. 10b), we observe that, despite using our algorithm that removes redundancies and irrelevant elements, it is unable to find a solution for all cases. Teaching situations of similar movement with frames oriented in the opposite direction generates redundancies (since the generated movement or path is the same), but they are not detected by the algorithm, resulting in suboptimal solutions that prevent proper functionality. The success rate of using combined datasets drops to 68%, indicating that using separated datasets

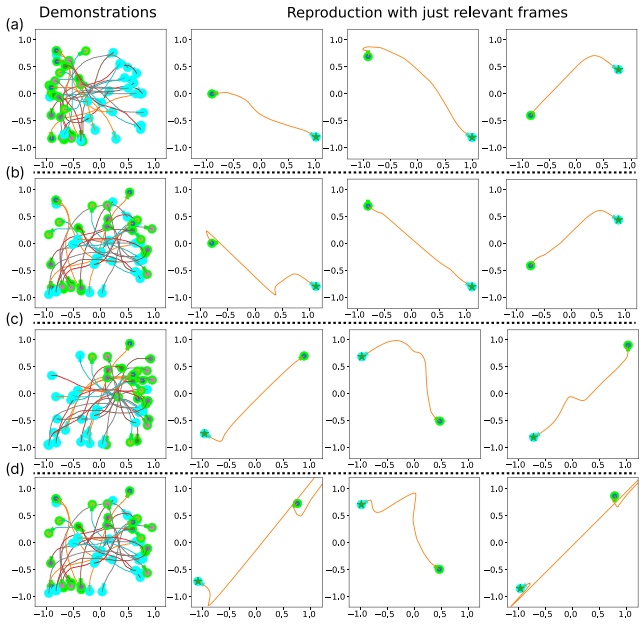

FIGURE 10. Comparison between oriented and not-oriented datasets. Each column represent a different task parameters. Each row represents the same task. Green circles represent the initial frame and blue circles the final frames. (a) Solutions for left to right tasks, (b) Solutions with mixed data for left to right tasks, (c) Solutions for right to left tasks, (d) Solutions with mixed data, same as used in (b), for right to left tasks.

significantly improves the results. Additionally, tests have been conducted with right-to-left movement data, using a dataset with oriented data (Fig. 10c) and another with the same mixed data as in the previous case (Fig. 10d). The result obtained in this case is the same as in the left-to-right movement, where the use of oriented datasets shows better results in terms of success rate and optimization of movements for use in the robot.

In addition to the use of oriented movement datasets, it is crucial, for the correct development of rearrangement tasks, to generate paths of the correct length. Since the TPGMM algorithm generates a solution of the same length as the data provided in human demonstrations, it happens that for very close data points, the algorithm generates a path where it is physically impossible to fit the specified number of points. This results in paths starting to create curves and solutions that are not optimal. In our case, the paths in the datasets have a length of 200 points each. The solution to enable shorter movements is to reduce the number of points per path. In this case, it was decided to reduce it to 50 points per path, attempting to have a single dataset. This creates the opposite situation, where for very long movements, the generated paths are very straight and with spikes, making them unnatural. To address this issue, it was decided to use both datasets with long lengths (and oriented) and datasets with short lengths (and oriented). Depending on the distance between the initial frame and the final frame, the choice would be made to use the long ones (200 points per solution) or the short ones (50 points per solution), being more adaptable for different cases. The use of dynamic data (mixing data with 50 points and 200 points) is not possible since the model is unable to perform the TPGMM model initialization process. The result of using short versus long datasets for situations where the frames are close can be observed in Fig. 11.

To assess its efficiency, we conducted 100 tests with different frame positions and orientations for each oriented dataset. The results show varying outcomes depending on the test distances (Table 3). If the distance between frames is less than 100 units, the algorithm using long datasets is unable to generate any correct solutions. If we use values up to 150 units of distance, the algorithm using long datasets can generate a 75% success rate. Beyond 200 units, the path solution returns to its 96% success rate. Therefore, experimentally, it is found that when the distance between frames is less than 200 units, it is optimal to use the short dataset as it generates higher success percentages, as described in Table 3.

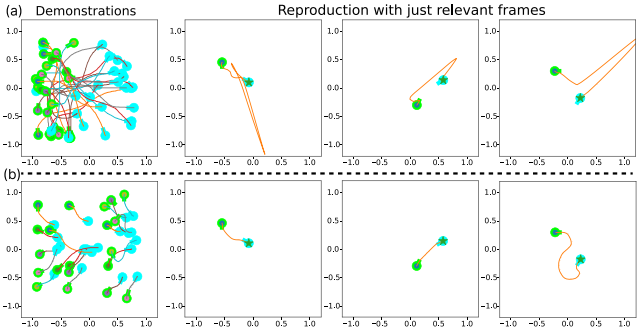

FIGURE 11. Comparison between short and long length datasets. Each column represent a different task aprameters. Each row represents the same task. Green circles represent the initial frame and blue circles the final frames. (a) Examples of solutions using long length datataset. The method is unable to generate a solution that fit in the space between frames. (b) Examples of solutions using sort length datataset. The method is able to generate a correct solution that fit in the space between frames.

TABLE 3. Comparison of the success rate of the datasets as a function of the distance in units (un.) between frames.

	50 un.	75 un.	100 un.	150 un.	200 un.
Right Short	**98%**	**98%**	**98%**	**95%**	93%
Right Long	0%	0%	12%	75%	**96%**
Left Short	**98%**	**98%**	**98%**	**96%**	94%
Left Long	0%	0%	14%	75%	96%

To assess the efficiency of the LfD presented, we conducted a quantitative comparison against widely used algorithms within the LfD domain using a dataset with real-world data captured through movement capture for two tasks in a real environment: the *shooting puck* task and the *sweeping* task. The algorithms against which we have compared include the base TPGMM, Probabilistic Movement Primitives (ProMPs) [87], that directly learns a probabilistic model from demonstrated trajectories, local Kernelized Movement Primitives (KMPs) [88] that reduce the use of manual basis functions using kernel techniques, and Task-Parameterized Probabilistic Movement Primitives

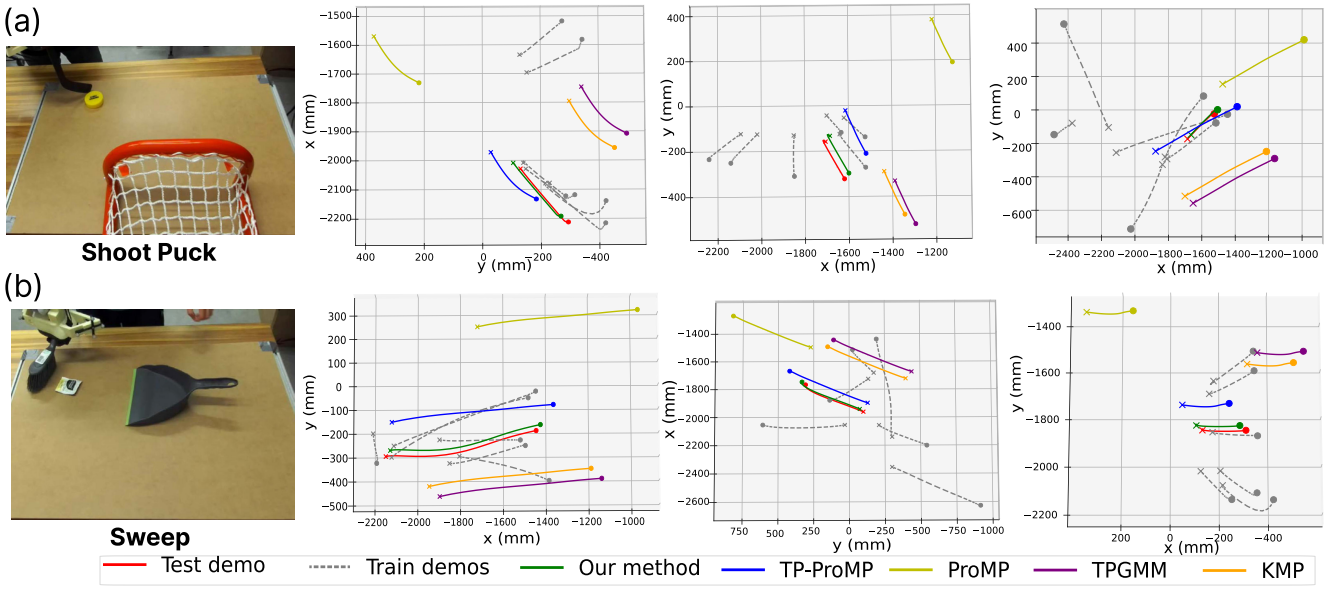

FIGURE 12. Experiment with real data from [89]. Each of the rows shows different experiments between the LfD algorithms compared against our implementation. (a) Three different examples for *Shooting Puck* task (full row), (b) Three different examples for *Sweeping* task (full row).

(TP-ProMP) [89], which relies on using multiple local references to a ProMP model.

For the evaluation of tasks in the real environment, all methods were compared for two real-world tasks. For the *Shooting Puck* task, demonstrations were captured by the user shooting the puck towards a net on the table. For the *Sweeping* task, demonstrations were solely performed with a specific oriented movement (from right to left), simulating the task of dragging trash towards a user-placed receptacle. The visual results of all algorithms can be observed in Fig. 12, where each method is depicted based on the captured demonstrations.

To validate the efficiency of the algorithms, we selected N demonstrations from the dataset as the training set and 1 demonstration as the test trajectory, replicating the experiments presented in [89]. To evaluate the predictive accuracy, we employed the Euclidean distance as a position similarity measure. For orientation, the norm of the quaternion difference was used as the metric. The mean of these metrics over the trajectory is reported, providing a straightforward and comprehensive reflection of performance throughout the entire trajectory. The position similarity measure is presented at critical points (start and end). This process was iterated 100 times, and the average, along with the standard deviation, was computed for a robust assessment of predictive performance. The results are presented in Table 4. These results show a significant improvement compared to the rest of the methods, both at critical points (start and end) and in the overall trajectory mean and final orientations. By considering only the truly relevant information for the new frames of the task to be performed and by eliminating existing redundancies between the data, the presented imitation learning method is capable of generating trajectories with very small errors, effectively solving tasks with new endpoints and starting points not demonstrated by the user.

C. ROBOT EXPERIMENTS

In order to test the real performance of our proposal, some experiments have been conducted with two different robots to prove the ability of the method to be used in different platforms. One of them has non-anthropomorphic arms (ADAM) and the other has anthropomorphic arms (TIAGo). The experiments involve solving various object rearrangement tasks in both a simulated and a real environment.

The simulated environment is where the test with the ADAM robot [90], [91] is carried out. This robot is a mobile bimanipulator platform designed to provide personal assistance to elderly individuals in performing physical tasks indoors. ADAM serves as a development platform for various tasks, such as social navigation [92], detection and filtering of reflective surface for mapping indoor environments [93] or learning manipulation through user demonstrations [83]. The robot has two UR3 arms positioned at the top of the robot torso, arranged in a configuration similar to human despite having non-anthropomorphic manipulators.

For the development of tests in simulation, an environment has been created in Matlab where the simulated model of the robot has been used to replicate the movements of the solution path. A *robotiq 2F85* gripper has been added to the end effector of the robot to verify that the grip orientation configurations were correct. To generate the configurations, a custom inverse kinematics algorithm has been applied based on searching for the optimal arm configuration from both exact solutions of inverse kinematics and solutions that do

TABLE 4. Mean ± SD distance from ground truth for LfD algorithms.

		ProMP	TPGMM	KMP	TP-ProMP	Our Method
Start (mm)	Shoot	269.8 ± 146.7	76.3±46.9	72.0±42.1	42.2±25.0	**17.5±6.2**
	Sweep	638.1±324.2	107.0±58.8	97.6±45.5	68.9±61.9	**14.1±4.1**
End(mm)	Shoot	175.3±92.7	55.2±29.5	51.7±29.4	52.9±27.6	**15.3±7.4**
	Sweep	240.4±111.9	189.2±107.3	180.4±95.6	110.5±85.0	**13.3±5.5**
Avg.(mm)	Shoot	237.0±123.9	67.9±24.3	65.8±24.2	57.5±24.3	**16.4±7.2**
	Sweep	479.9±216.3	152.6±64.2	139.3±59.7	106.4±49.0	**16.3±8.4**
Ori.(^{o})	Shoot	10.1±3.5	13.2±4.0	8.9±2.2	6.3±5.3	**3.3±2.0**
	Sweep	9.4±2.7	14.4±4.2	8.8±1.8	5.8±4.7	**2.7±1.0**

TABLE 5. Table of errors in positioning and orientation of rearrangement experiments.

		Simulation			Real Environment		
		Error x(cm)	Error y(cm)	Error θ (rad)	Error x (cm)	Error y (cm)	Error θ (rad)
Exp1	Orange	0.65	0.98	0.003	1.25	1.12	0.032
	Computer mouse	0.45	0.65	0.042	0.95	1.4	0.04
	Cup	0.55	0.25	0.0014	0.78	0.74	0.028
Exp2	Fork	1.02	0.987	0.012	1.41	1.01	0.035
	Knife	0.987	0.84	0.008	1.00	1.24	0.056
	Spoon	0.6987	0.7846	0.015	1.24	1.87	0.04
Exp 3	Remote Control	1.00	0.547	0.009	1.35	1.245	0.009
	Scissors	0.987	0.365	0.0074	1.32	1.12	0.051

not generate collisions with the elements of the environment or the robot's own body. For the simulation, demonstration data has been collected using the TAICHI algorithm [84], [85], through which a series of different data points have been captured for each type of movement using an RGBD camera. It is important to note that, since the TPGMM operates in a simulated environment, a re-targeting process of the path points for the physical and mechanical limits of the ADAM robot is necessary. The result of the simulation, illustrating the functioning of the entire framework, can be observed in the following **video: https://youtu.be/gyn1992YxhI**. In the simulation experiment, we focused on visualising the paths and hid the object models to avoid interference with this process. These models were used to calculate the efficiency of the method presented in the Table 4.

The tests in the real environment have been conducted using the TIAGo robot [94]. This robotic platform is one of the most widely used for household task resolution [95] and has also been extensively utilised for the development and testing of imitation learning algorithms [96]. The robot has two arms with an anthropomorphic composition, having a structure practically identical to that of the human arm. In this case, the demonstrations were performed through a kinesthetic process, where we directly captured the data using TIAGo's own arm. Once demonstrations dataset was generated, the process was carried out in two different scenarios. The operation of an example of one test on the real environment can be observed in the following **video: https://youtu.be/IwdsndI9cW0**. An example of the tests carried out in simulation and in the real environment is presented in Fig. 13.

The experiments with the robot in the real environment were carried out with the TIAGo robot base fixed and facing the work table. This allows us to have a stable global reference with respect to the robot itself. Additionally, it is necessary to carry out a configuration generation process (both for the ADAM robot and for TIAGo). This process is specific to each robot and allows the paths generated to be adapted to the configurations of the different robotic arms that exist in the state of the art, which allows them to be generalised. This has not been included as part of the framework so that any user can make use of our method regardless of the robot used.

To assess efficiency in various tests conducted both in simulation and in the real environment, we have evaluated the error in the final placement of objects, considering the image generated by StableDiffusion as the ground truth. In this regard, Table 5 presents the position errors for each object and the orientation error for three of the conducted tests.

The position errors in both X and Y do not exceed 2 cm in any case, ensuring that the robots can replicate with great fidelity the learned paths, easily achievable by robotic arms. Additionally, it is noted that the maximum error obtained in the experiments is 0.056 radians, indicating that the robots can accurately follow the orientations of different objects, and the developed algorithm generates feasible solutions for robotic arms. In this way, we observe in a general sense that the robots are capable of solving various types of tasks with objects of different shapes, sizes, and textures in a stable

manner, obtaining systematic and small errors in the majority of experiments.

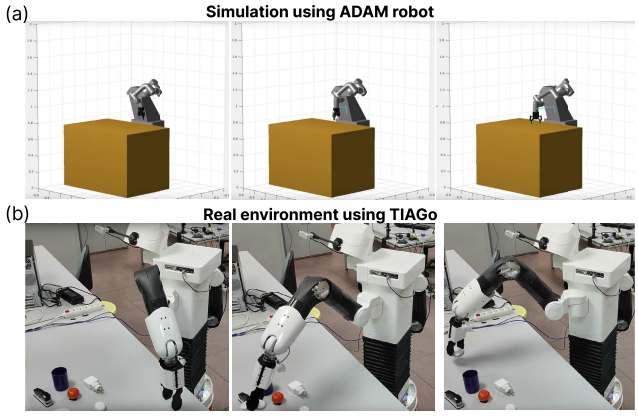

FIGURE 13. Tasks performed in simulation with the ADAM robot presented in top row (a) and in the real environment presented in down row with the TIAGo robot (b). For simplicity, objects are not displayed in simulation. In the real environment experiment, the entire process operation is observed, including the object grasping procedure.

V. CONCLUSION AND FUTURE WORK

This work introduces an object reconfiguration framework in a robot's working environment. The proposed framework novelty comes from the combination of state-of-the-art zero-shot models for image detection and generation with self-developed algorithms. Our own implementations in this work are the object direction estimator, the multi prompt generator, the final scenes selector, the collision checker and the Learning from Demonstrations algorithm for path generation. Moreover, the modular design of our framework allows it to be optimised or used individually, adapting it to various tasks or projects. To test its performance, we have made experiments with the ADAM robot (simulation) and the TIAGo robot (real environment). These results include accurate object detection, generation of realistic solutions, and smooth and natural movements. During both test, experimental errors in position and orientation were low, with a position error lower than 2 cm and an orientation error smaller than 0.056 radians. We also highlight the virtues of each modules:

- The perception module, which uses YOLOv8 and our own developed algorithms, is effective in identifying object positions and orientations. Its robustness is maintained under various conditions, such as different object types, sizes, distances and partial occlusions.
- The imagination module, based on our automatic generator of prompts and StableDiffusion accelerated by LoRa, generates useful instructions to obtain ordered images. Moreover, the self-developed algorithm selects the best solution without user intervention.
- The execution module, with our Learning from Demonstration algorithm, ensures safe and precise movements done in a human-like way, surpassing traditional methods in a variety of tasks.

Despite the advantages and innovations presented, each module has its limitations and can be improved. In the observation module, the combination with other neural networks could increase the accuracy of object detection. In the imagination module, the main limitation is to generate cues that faithfully represent all features of the environment. One possible approach is the semantic extraction of spatial relationships between objects (e.g., 'next to', 'on', 'left'), which could generate more controlled solutions. In addition, the selection of duplicate objects could be improved by an algorithm that chooses the best positioned one and eliminates the remaining ones. In the execution module, the use of both arms could increase efficiency in repositioning, allowing the robot to consider the position of objects and execute the movement more optimally and quickly.

Other relevant improvement consist of an integration of a human-robot interaction. This could be done using voice instructions that tell the robot how to set the scene (refining the generated prompt), or directly interacting with the robot in a collaborative loop where the user can place an object to help the robot rearrange the remaining objects.

ACKNOWLEDGMENT

(Alberto Mendez and Adrian Prados contributed equally to this work.)

REFERENCES

[1] I. Kapelyukh and E. Johns, "My house, my rules: Learning tidying preferences with graph neural networks," in *Proc. Conf. Robot Learn.*, 2022, pp. 740–749.

[2] M. Kang, Y. Kwon, and S.-E. Yoon, "Automated task planning using object arrangement optimization," in *Proc. 15th Int. Conf. Ubiquitous Robots (UR)*, Jun. 2018, pp. 334–341, doi: 10.1109/URAI.2018.8442210.

[3] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," in *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*. Springer, Oct. 2022, pp. 355–373, doi: 10.1007/978-3-031-19842-7_21.

[4] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," 2023, *arXiv:2307.01952*.

[5] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "IFOR: Iterative flow minimization for robotic object rearrangement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 14767–14777, doi: 10.1109/CVPR52688.2022.01437.

[6] W. Liu, C. Paxton, T. Hermans, and D. Fox, "StructFormer: Learning spatial structure for language-guided semantic rearrangement of novel objects," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 6322–6329, doi: 10.1109/ICRA46639.2022.9811931.

[7] W. Liu, Y. Du, T. Hermans, S. Chernova, and C. Paxton, "StructDiffusion: Language-guided creation of physically-valid structures using unseen objects," 2022, *arXiv:2211.04604*.

[8] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," 2022, *arXiv:2204.06125*.

[9] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10674–10685.

[10] S. Hun Cheong, B. Y. Cho, J. Lee, C. Kim, and C. Nam, "Where to relocate: Object rearrangement inside cluttered and confined environments for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 7791–7797, doi: 10.1109/ICRA40945.2020.9197485.

[11] K. Ren, L. E. Kavraki, and K. Hang, "Rearrangement-based manipulation via kinodynamic planning and dynamic planning horizons," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 1145–1152, doi: 10.1109/IROS47612.2022.9981599.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V*. Springer, Sep. 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[14] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009, doi: 10.1016/j.robot.2008.10.024.

[15] M. Malekzadeh and J. J. Steil, "Learning the end-effector pose from demonstration for the bionic handling assistant robot," in *Proc. Workshop Hum.-Friendly Robot.*, 2016, pp. 101–107.

[16] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2007, p. 317, doi: 10.1109/ICASSP.2007.366913.

[17] W. Goodwin, S. Vaze, I. Havoutis, and I. Posner, "Zero-shot category-level object pose estimation," in *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*. Springer, Oct. 2022, pp. 516–532, doi: 10.1007/978-3-031-19842-7_30.

[18] B. Okorn, Q. Gu, M. Hebert, and D. Held, "ZePHyR: Zero-shot pose hypothesis rating," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 14141–14148, doi: 10.1109/ICRA48506.2021.9560874.

[19] P. Ausserlechner, D. Haberger, S. Thalhammer, J.-B. Weibel, and M. Vincze, "ZS6D: Zero-shot 6D object pose estimation using vision transformers," 2023, *arXiv:2309.11986*.

[20] J. Kang, W. Liu, W. Tu, and L. Yang, "YOLO-6D+: Single shot 6D pose estimation using privileged silhouette information," in *Proc. Int. Conf. Image Process. Robot. (ICIP)*, Mar. 2020, pp. 1–6, doi: 10.1109/ICIP48927.2020.9367354.

[21] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digit. Signal Process.*, vol. 126, Jun. 2022, Art. no. 103514, doi: 10.1016/j.dsp.2022.103514.

[22] J. Terven and D. Cordova-Esparza, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," 2023, *arXiv:2304.00501*

[23] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," 2018, *arXiv:1809.10790*.

[24] *Ultralytics Docs*. Accessed: Feb. 4, 2024. [Online]. Available: https://docs.ultralytics.com/es

[25] G. Sarch, Z. Fang, A. W. Harley, P. Schydlo, S. Gupta, and K. Fragkiadaki, "TIDEE: Tidying up novel rooms using visuo-semantic commonsense priors," in *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*. Springer, Oct. 2022, pp. 480–496, doi: 10.1007/978-3-031-19842-7_28.

[26] M. J. Schuster, D. Jain, M. Tenorth, and M. Beetz, "Learning organizational principles in human environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3867–3874, doi: 10.1109/ICRA.2012.6224553.

[27] N. Abdo, C. Stachniss, L. Spinello, and W. Burgard, "Robot, organize my shelves! Tidying up objects by predicting user preferences," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1557–1564, doi: 10.1109/ICRA.2015.7139396.

[28] Y. Lin, A. S. Wang, E. Undersander, and A. Rai, "Efficient and interpretable robot manipulation with graph neural networks," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2740–2747, Apr. 2022, doi: 10.1109/LRA.2022.3143518.

[29] V. Jain, Y. Lin, E. Undersander, Y. Bisk, and A. Rai, "Transformers are adaptable task planners," in *Proc. Conf. Robot Learn.*, 2023, pp. 1011–1037.

[30] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1–26, Jan. 2023, doi: 10.1109/TPAMI.2021.3137605.

[31] J. Johnson, A. Gupta, and L. Fei-Fei, "Image generation from scene graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1219–1228, doi: 10.1109/CVPR.2018.00133.

[32] J. Wald, H. Dhamo, N. Navab, and F. Tombari, "Learning 3D semantic scene graphs from 3D indoor reconstructions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3960–3969, doi: 10.1109/CVPR42600.2020.00402.

[33] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari, and C. Rupprecht, "Semantic image manipulation using scene graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5212–5221, doi: 10.1109/CVPR42600.2020.00526.

[34] B. Tang and G. S. Sukhatme, "Selective object rearrangement in clutter," in *Proc. Conf. Robot Learn.*, 2023, pp. 1001–1010.

[35] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 6541–6548, doi: 10.1109/ICRA48506.2021.9561548.

[36] W. Goodwin, S. Vaze, I. Havoutis, and I. Posner, "Semantically grounded object matching for robust robotic scene rearrangement," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 11138–11144, doi: 10.1109/ICRA46639.2022.9811817.

[37] Q. A. Wei, S. Ding, J. J. Park, R. Sajnani, A. Poulenard, S. Sridhar, and L. Guibas, "Lego-net: Learning regular rearrangements of objects in rooms," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 19037–19047, doi: 10.1109/CVPR52729.2023.01825.

[38] M. Wu, F. Zhong, Y. Xia, and H. Dong, "TarGF: Learning target gradient field for object rearrangement," 2022, *arXiv:2209.00853*.

[39] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and where pathways for robotic manipulation," in *Proc. Conf. Robot Learn.*, 2022, pp. 894–906.

[40] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, and V. Sindhwani, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Proc. Conf. Robot Learn.*, 2021, pp. 726–747.

[41] Y. Jiang, M. Lim, and A. Saxena, "Learning object arrangements in 3D scenes using human context," 2012, *arXiv:1206.6462*.

[42] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon, "Text-to-image diffusion model in generative AI: A survey," 2023, *arXiv:2303.07909*.

[43] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. G. Lopes, B. K. Ayan, and T. Salimans, "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 36479–36494.

[44] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," 2021, *arXiv:2112.10741*.

[45] A. Raj, S. Kaza, B. Poole, M. Niemeyer, N. Ruiz, B. Mildenhall, S. Zada, K. Aberman, M. Rubinstein, J. Barron, Y. Li, and V. Jampani, "DreamBooth3D: Subject-driven text-to-3D generation," 2023, *arXiv:2303.13508*.

[46] J. Ploennigs and M. Berger, "AI art in architecture," *AI Civil Eng.*, vol. 2, no. 1, p. 8, Aug. 2023, doi: 10.1007/s43503-023-00018-y.

[47] *Clipdrop*. Accessed: Dec. 11, 2023. [Online]. Available: https://clipdrop.co/

[48] *Adobe Firefly*. Accessed: Dec. 11, 2023. [Online]. Available: https://www.adobe.com/es/products/firefly.html

[49] J. J. Bird and A. Lotfi, "CIFAKE: Image classification and explainable identification of AI-generated synthetic images," 2023, *arXiv:2303.14126*.

[50] Y. Chen and J. Zou, "TWIGMA: A dataset of AI-generated images with metadata from Twitter," 2023, *arXiv:2306.08310*.

[51] I. Kapelyukh, V. Vosylius, and E. Johns, "DALL-E-bot: Introducing web-scale diffusion models to robotics," *IEEE Robot. Autom. Lett.*, vol. 8, no. 7, pp. 3956–3963, Jul. 2023, doi: 10.1109/LRA.2023.3272516.

[52] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," *IEEE Trans. Robot.*, vol. 32, no. 4, pp. 960–972, Aug. 2016, doi: 10.1109/TRO.2016.2588879.

[53] K. Hang, J. A. Stork, N. S. Pollard, and D. Kragic, "A framework for optimal grasp contact planning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 704–711, Apr. 2017, doi: 10.1109/LRA.2017.2651381.

[54] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23, nos. 7–8, pp. 729–746, Aug. 2004, doi: 10.1177/0278364904045471.

[55] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3327–3332, doi: 10.1109/ROBOT.2007.363986.

[56] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 3940–3947, doi: 10.1109/ICRA.2016.7487583.

[57] J. E. King, V. Ranganeni, and S. S. Srinivasa, "Unobservable Monte Carlo planning for nonprehensile rearrangement tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4681–4688, doi: 10.1109/ICRA.2017.7989544.

[58] D. Schiebener, J. Morimoto, T. Asfour, and A. Ude, "Integrating visual perception and manipulation for autonomous learning of object representations," *Adapt. Behav.*, vol. 21, no. 5, pp. 328–345, Oct. 2013, doi: 10.1177/1059712313484502.

[59] A. Reichlin, G. L. Marchetti, H. Yin, A. Varava, and D. Kragic, "Learning geometric representations of interactive objects," in *Proc. ICLR*, 2022, pp. 1–13.

[60] L. Berscheid, P. Meißner, and T. Kröger, "Robot learning of shifting objects for grasping in cluttered environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 612–618, doi: 10.1109/IROS40897.2019.8968042.

[61] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "Robosuite: A modular simulation framework and benchmark for robot learning," 2020, *arXiv:2009.12293*.

[62] A. Kramberger, E. Shahriari, A. Gams, B. Nemec, A. Ude, and S. Haddadin, "Passivity based iterative learning of admittance-coupled dynamic movement primitives for interaction with changing environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 6023–6028, doi: 10.1109/IROS.2018.8593647.

[63] W. Amanhoud, M. Khoramshahi, M. Bonnesoeur, and A. Billard, "Force adaptation in contact tasks with dynamical systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 6841–6847, doi: 10.1109/ICRA40945.2020.9197509.

[64] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-reinforcement learning for robotic industrial insertion tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 9728–9735, doi: 10.1109/IROS45743.2020.9340848.

[65] F. Bai, F. Meng, J. Liu, J. Wang, and M. Q.-H. Meng, "Hierarchical policy with deep-reinforcement learning for nonprehensile multiobject rearrangement," *Biomimetic Intell. Robot.*, vol. 2, no. 3, Sep. 2022, Art. no. 100047, doi: 10.1016/j.birob.2022.100047.

[66] A. H. Qureshi, A. Mousavian, C. Paxton, M. C. Yip, and D. Fox, "NeRP: Neural rearrangement planning for unknown objects," 2021, *arXiv:2106.01352*.

[67] D. Huang, C. Tang, and H. Zhang, "Efficient object rearrangement via multi-view fusion," 2023, *arXiv:2309.08994*.

[68] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, K. Hausman, and F. Xia, "Scaling robot learning with semantically imagined experience," 2023, *arXiv:2302.11550*.

[69] Y. Hu, Z. Zhang, R. Liu, P. Wyder, and H. Lipson, "Knolling bot: A transformer-based approach to organizing a messy table," 2023, *arXiv:2310.04566*.

[70] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar, "CACTI: A framework for scalable multi-task multi-scene visual imitation learning," 2022, *arXiv:2212.05711*.

[71] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, "RoboAgent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking," 2023, *arXiv:2309.01918*.

[72] G. Zhai, X. Cai, D. Huang, Y. Di, F. Manhardt, F. Tombari, N. Navab, and B. Busam, "SG-bot: Object rearrangement via coarse-to-fine robotic imagination on scene graphs," 2023, *arXiv:2309.12188*.

[73] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Service Robot.*, vol. 9, no. 1, pp. 1–29, Jan. 2016, doi: 10.1007/s11370-015-0187-9.

[74] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ With the OpenCV library*. Sebastopol, CA, USA: O'Reilly Media, 2016.

[75] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell, "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 464–470, doi: 10.1109/IROS.2015.7353413.

[76] A. Prados, S. Garrido, and R. Barber, "Learning and generalization of task-parameterized skills through few human demonstrations," *Eng. Appl. Artif. Intell.*, vol. 133, Jul. 2024, Art. no. 108310, doi: 10.1016/j.engappai.2024.108310.

[77] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Syst., Man Cybern., B*, vol. 37, no. 2, pp. 286–298, Apr. 2007, doi: 10.1109/TSMCB.2006.886952.

[78] T. Alizadeh and M. Malekzadeh, "Identifying the relevant frames of reference in programming by demonstration using task-parameterized Gaussian mixture regression," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Dec. 2016, pp. 453–458, doi: 10.1109/SII.2016.7844040.

[79] T. Alizadeh and N. Karimi, "Exploiting the task space redundancy in robot programming by demonstration," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Aug. 2018, pp. 2394–2399, doi: 10.1109/ICMA.2018.8484455.

[80] A. Sena, B. Michael, and M. Howard, "Improving task-parameterised movement learning generalisation with frame-weighted trajectory generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 4281–4287, doi: 10.1109/IROS40897.2019.8967688.

[81] Y. Huang, J. Silvério, L. Rozo, and D. G. Caldwell, "Generalized task-parameterized skill learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5474–5667, doi: 10.1109/ICRA.2018.8461079.

[82] S. E. Zaatari, Y. Wang, Y. Hu, and W. Li, "An improved approach of task-parameterized learning from demonstrations for cobots in dynamic manufacturing," *J. Intell. Manuf.*, vol. 33, no. 5, pp. 1503–1519, Jun. 2022, doi: 10.1007/s10845-021-01743-w.

[83] A. Prados, A. Mora, B. López, J. Muñoz, S. Garrido, and R. Barber, "Kinesthetic learning based on fast marching square method for manipulation," *Appl. Sci.*, vol. 13, no. 4, p. 2028, Feb. 2023, doi: 10.3390/app13042028.

[84] B. Lopez, A. Prados, L. Moreno, and R. Barber, "TAICHI algorithm: Human-like arm data generation applied on non-anthropomorphic robotic manipulators for demonstration," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2023, pp. 1–7, doi: 10.1109/ecmr59166.2023.10256343.

[85] A. Prados, B. López, R. Barber, and L. Moreno, "Tracking humano visual aplicado a manipuladores no antropomórficos para imitación," in *Proc. 44th Jornadas de Automática*. Zaragoza, Spain: Univ. da Coruña, 2023, pp. 714–719, doi: 10.17979/spudc.9788497498609.714.

[86] S. Luo, Y. Tan, S. Patil, D. Gu, P. von Platen, A. Passos, L. Huang, J. Li, and H. Zhao, "LCM-LoRA: A universal stable-diffusion acceleration module," 2023, *arXiv:2311.05556*.

[87] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Auto. Robots*, vol. 42, no. 3, pp. 529–551, Mar. 2018, doi: 10.1007/s10514-017-9648-7.

[88] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 833–852, Jun. 2019, doi: 10.1177/0278364919846363.

[89] X. Yao, Y. Chen, and B. Tripp, "Improved generalization of probabilistic movement primitives for manipulation trajectories," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 287–294, Jan. 2024, doi: 10.1109/LRA.2023.3333741.

[90] A. Mora, A. Prados, A. Mendez, G. Espinoza, P. Gonzalez, B. Lopez, V. Muñoz, L. Moreno, S. Garrido, and R. Barber, "ADAM: A robotic companion for enhanced quality of life in aging populations," *Frontiers Neurorobotics*, vol. 18, Feb. 2024, Art. no. 1337608, doi: 10.3389/fnbot.2024.1337608.

[91] R. Barber, F. J. Ortiz, S. Garrido, F. M. Calatrava-Nicolás, A. Mora, A. Prados, J. A. Vera-Repullo, J. Roca-González, I. Méndez, and Ó. M. Mozos, "A multirobot system in an assisted home environment to support the elderly in their daily lives," *Sensors*, vol. 22, no. 20, p. 7983, Oct. 2022, doi: 10.3390/s22207983.

[92] A. Mora, A. Prados, A. Mendez, R. Barber, and S. Garrido, "Sensor fusion for social navigation on a mobile robot based on fast marching square and Gaussian mixture model," *Sensors*, vol. 22, no. 22, p. 8728, Nov. 2022, doi: 10.3390/s22228728.

[93] A. Mora, A. Prados, P. González, L. Moreno, and R. Barber, "Intensity-based identification of reflective surfaces for occupancy grid map modification," *IEEE Access*, vol. 11, pp. 23517–23530, 2023, doi: 10.1109/ACCESS.2023.3252909.

[94] J. Pages, L. Marchionni, and F. Ferro, "TIAGo: The modular robot that adapts to different research needs," in *Proc. Int. Workshop Robot Modularity*, vol. 290, 2016, pp. 1–4.

[95] Z. Wang, G. Tian, and X. Shao, "Home service robot task planning using semantic knowledge and probabilistic inference," *Knowl.-Based Syst.*, vol. 204, Sep. 2020, Art. no. 106174, doi: 10.1016/j.knosys.2020.106174.

[96] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín, "Error-aware imitation learning from teleoperation data for mobile manipulation," in *Proc. Conf. Robot Learn.*, 2022, pp. 1367–1378.

ALBERTO MENDEZ was born in Santa Cruz de Tenerife, Spain, in 1999. He received the B.S. degree in electronics and automatic engineering from the University of La Laguna (ULL), in 2021, and the M.S. degree in robotics and automation from the Universidad Carlos III de Madrid (UC3M), in 2023, where he is currently pursuing the Ph.D. degree with the Program of Electrical, Electronic and Automation Engineering.

He is collaborating as a Researcher with the Robotics Laboratory, UC3M. His research interests include computer vision and deep learning, including the detection and localization of elements in the environment, artificial intelligence, and 3D perception.

ADRIAN PRADOS was born in Leganes, Spain, in 1999. He received the B.S. degree in industrial electronics and automation engineering and the M.S. degree in robotics and automation from the Universidad Carlos III de Madrid (UC3M), in 2021 and 2023, respectively, where he is currently pursuing the Ph.D. degree with the Program of Electrical, Electronic and Automation Engineering.

He is collaborating as a Researcher with the Robotics Laboratory, UC3M. His main research interests include imitation learning and learning from demonstration, including generalization of learned task and adaptation to new environment, applied on manipulation tasks. He also works in terms related with control of mobile manipulators, deep learning, optimization techniques, and assistive robots.

ELISABETH MENENDEZ (Graduate Student Member, IEEE) received the B.S. degree in industrial engineering from the University of Oviedo, Spain, in 2015, and the M.S. degree in robotics and automation from the Universidad Carlos III de Madrid (UC3M), Spain, in 2019, where she is currently pursuing the Ph.D. degree in electrical, electronic and automation engineering, specializing in human–robot interaction with a focus on object manipulation. Her research interests include gaze-based and decision-driven interactions in robot manipulation, aiming for human-like movements to enhance safety and naturalness in human–robot collaboration.

RAMON BARBER (Senior Member, IEEE) received the B.Sc. degree in industrial engineering from the Polytechnic University of Madrid, in 1995, and the Ph.D. degree in industrial technologies from the Universidad Carlos III de Madrid, Spain, in 2000. He is currently an Associate Professor with the System Engineering and Automation Department, Universidad Carlos III de Madrid. His research interests include mobile robotics including perception of the environment, environment modeling, planning, localization, and navigation tasks, considering geometrical, topological, and semantic representations. He is a member of the International Federation of Automatic Control (IFAC).