“Push-That-There”: Tabletop Multi-robot Object Manipulation via Multimodal ‘Object-level Instruction’

Keru Wang
New York University
New York, NY, USA
keru.wang@nyu.edu

Zhu Wang
New York University
New York, NY, USA
zhu.wang@nyu.edu

Ken Nakagaki
University of Chicago
Chicago, IL, USA
knakagaki@uchicago.edu

Ken Perlin
New York University
New York, NY, USA
perlin@cs.nyu.edu

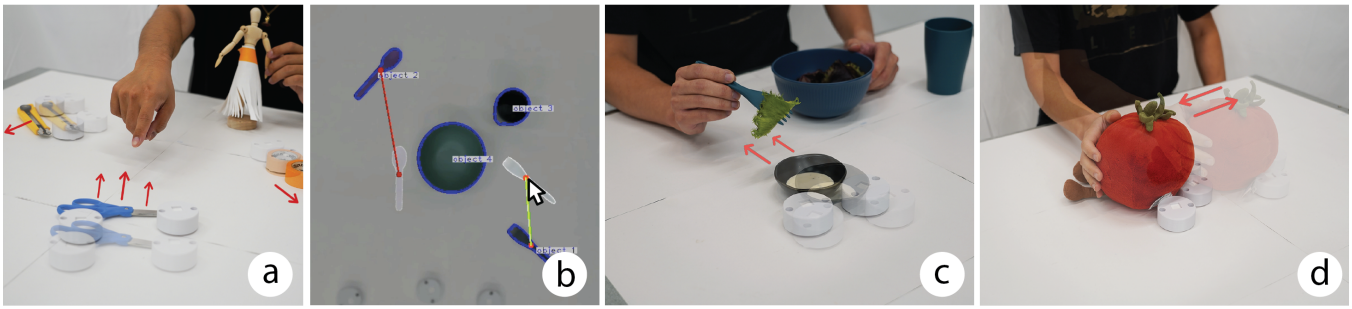

Figure 1: Push-That-There (a) Push-That-There is an interactive multi-robot system that autonomously handles free-form objects on tabletops according to the user’s object-level instruction. (b) The system is integrated with multimodal input, such as Graphical User Interface. (c) The system can help users with daily object-handling tasks, such as bringing dipping sauce to the user when they are eating. (d) The system can add force to passive objecs, turning them into active tangible interactive interface.

CCS CONCEPTS

• **Human-centered computing** → **User interface design**; *Haptic devices*; **Interaction techniques**.

KEYWORDS

Multi-Robot Control, Tangible Interface, Human-Robot Interaction, Object-level instruction, Multi-robot UI

ACM Reference Format:
Keru Wang, Zhu Wang, Ken Nakagaki, and Ken Perlin. 2024. “Push-That-There”: Tabletop Multi-robot Object Manipulation via Multimodal ‘Object-level Instruction’. In *Designing Interactive Systems Conference (DIS ’24), July 01–05, 2024, IT University of Copenhagen, Denmark.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3643834.3661542

We present “Push-That-There”, an interaction method and system enabling multimodel object-level user interaction with multi-robot system to autonomously and collectively manipulate objects on tabletop surfaces, inspired by “Put-That-There”. Rather than requiring users to instruct individual robots, users directly specify how they want the objects to be moved, and the system responds by autonomously moving objects via our generalizable multi-robot control algorithm. The system is combined with various user instruction modalities, including gestures, GUI, tangible manipulation, and speech, allowing users to intuitively create object-level instruction. We outline a design space, highlight interaction design opportunities facilitated by “Push-That-There”, and provide an evaluation to assess our system’s technical capabilities. While other recent HCI research has studied interaction using multi-robot system (e.g. Swarm UIs), our contribution is in the design and technical implementation of intuitive object-level interaction for multi-robot system that allows users to work at a high level, rather than needing to focus on the movements of individual robots.

DIS ’24, July 01–05, 2024, IT University of Copenhagen, Denmark

ACM ISBN 979-8-4007-0583-0/24/07
https://doi.org/10.1145/3643834.3661542

1 INTRODUCTION

Researchers in Human-Computer Interaction (HCI) have long explored the development of interactive systems designed to transport passive physical objects in everyday environments, which has led to varied approaches to the mechanism of actuation. For example, some researchers have used pin-based shape displays to move and animate objects [17, 39, 52], while others have used a robotic arm [31] or wheeled robots [55] to facilitate passive objects manipulation. Some other researchers investigate using acoustic levitation to manipulate small objects without physical contact [41]. Each approach tends to optimize for particular goals, such as enhancing the expressiveness of passive objects, providing haptic feedback for interaction with virtual environments, or automating repetitive tasks. This area of research is often motivated by a vision of the future in which our physical surroundings can dynamically reconfigure themselves to meet users’ needs and requirements via computational, interactive, robotic systems.

Over the past several decades, there has been growing interest in multi-robot systems. The systems used a number of coordinated robots for passive object handling on tabletops to perform tasks such as interactively arranging furniture [48], relocating phones [28], transporting tools [38], and cleaning clusters of objects on tabletop surfaces [57]. In the robotics field, some researchers use the term "swarm robots" to describe these systems. They define a robot swarm as 3+ robots working cooperatively under limited human control [3, 15, 27]. However, some other researchers have a more specific definition which characterizes a robot swarm as a group of self-organizing robots with decentralized control and minimal synchronization [40, 65]. To clarify the distinction and avoid potential confusion, we use ***"multi-robot systems"*** to describe these systems in general, and only use ***"swarm"*** to specify those with decentralized control and minimal synchronization.

While research in multi-robot system has led to proof-of-concept demonstrations, showcased intriguing opportunities, and shown potential for handling objects through the use of tabletop swarm robots, it has often fallen short in providing concrete implementations that satisfy two critical goals: 1. Merging arbitrary object tracking module with a shape-independent object handling algorithm to let the multi-robot system be used for dynamic, varied interactive environments efficiently, and 2. Executing autonomous and collective multi-robot actions in response to users' high-level instructions. To bridge the gap between conceptual exploration and practical applications in multi-robot-based object-handling interactions, there is a need to develop comprehensive technical methodologies and pragmatic implementations.

To address this challenge, we introduce Push-That-There, an interactive system that manages a mobile multi-robot system to push unmarked passive objects on tabletops to specific positions and orientations collectively and autonomously. The system controls the robots by directly interpreting users' high-level instructions about where each object should be moved to, which we term ***"object-level instruction"***, in contrast to ***"robot-level instruction"***, where users need to specify every actionable operation for individual robots to handle the objects [20, 27].

The interaction modality and research approach of object-level instruction are affected by "Put-That-There" [6], one of the most influential HCI papers in multi-modal interaction published in 1980 by Richard Bolt. In the paper, Bolt introduced his invention for users to instruct a computer with a large digital screen through high-level instruction combining pointing gestures and speech. This system was innovative in the way that users never needed to care about how the computer was rendering the graphics, or learn to use any special interfaces (e.g., mouse or keyboard) to instruct the computer. Users simply pointed to digital graphics and said "put that there", which is an illuminating example of object-level instruction on a digital screen. In this spirit, our work intends to build and enable object-level instructions and multimodal inputs, but with a focus on facilitating autonomous multi-robot control to manipulate physical objects, while the user never needs to be concerned about individual robot's movement. Similar to the impact Bolt made, we wish to establish a system that allows users to intuitively and seamlessly instruct their environments and objects assisted by a multi-robot system that is flexibly and dynamically reconfigured.

Object-level interactions offer potential benefits over robot-level interactions (a method to instruct individual robots' movements) [23, 25, 27, 57]: 1. It is easier for users to carry out actions requiring multiple robots to handle multiple objects simultaneously. 2. It allows users to concentrate on the task's objectives, minimizing the cognitive load to assign and plan individual robots' movements. And 3. The simplicity of object-level instruction makes it easy to be fused with various user input modalities (speech, graphical user interface, gesture interaction, and tangible manipulation), as the system only requires information from users about how the objects should be moved.

We implement a tabletop multi-robot system that can push free-form unmarked objects to target positions and orientations autonomously and simultaneously based on object-level instructions. To make this system adaptable for everyday items, we deploy a webcam-based tracking module for identifying objects' contour and design a multi-robot object handling algorithm independent of the object's shape. We also include special features to enhance the interactive tabletop experience, such as disregarding human hand interference, dynamically allocating robot resources to objects, and instructing robots to leave the limited workspace upon task completion. The system provides dynamic robot allocation, parallel object handling, and a scalable robot control algorithm. Compared with an object handling system employing hardware such as robotic arms, our multi-robot system offers distinct advantages, including decreased intrusiveness, flexibility, scalability, and enhanced fault tolerance to a single robot's dysfunctionality [58].

We integrated multiple user input modalities with our multi-robot control system to demonstrate the system's ability to work with general user interfaces via object-level instructions. The user interface includes gesture interaction, graphical user interface (GUI), tangible manipulation, and speech. This shows our system's potential to facilitate a user-centric multi-robot interaction experience for intuitive object manipulation that accommodates the user's preferred input method.

By experimenting with our implemented system prototype, we explore the design space for interactive object handling on tabletops with a multi-robot system. The design space leverages the advantages of haptic and tangible interfaces. Moreover, we demonstrate applications including cooking/dining assistance, remote communication, and immersive haptic experiences with the limited robots in our lab. We further discuss some potential applications as the system is scaled up. Our work offers a generalizable approach to interactive multi-robot object-handling systems, which can potentially benefit HCI researchers.

Our contributions include:

- An introduction of object-level instruction for multi-robot system user interfaces, and its design space to layout the research opportunity in HCI.
- An implementation of a multi-robot object handling system designed for an interactive tabletop environment. It features an object-level multi-robot control algorithm integrated with an arbitrary object-tracking module. The system interprets object-level instructions and intelligently coordinates numerous robots to autonomously manage and adjust to varying

numbers of objects on a tabletop, ensuring efficient operation in dynamic settings and the system's scalability.
- Integration of multimodal user input modalities (gesture interaction, GUI, tangible manipulation, and speech) with the multi-robot control system via object-level instruction to show our system's generalizability and interactivity.
- A technical evaluation of the system to measure performance and efficiency in handling various everyday objects on tabletops, both individually and collectively.
- The exploration of object-level instructions in a wide range of application scenarios.

2 RELATED WORK

Push-That-There builds upon previous work on multi-robot and swarm user interactions, tabletop interactive object actuation, and multi-robot object handling control. It advanced these fields by implementing and merging autonomous multi-robot system with a multimodal interface in a tabletop setting. This lets users give object-level instructions for more natural and efficient interactions.

2.1 Multi-Robot and Swarm User Interactions

HCI researchers have shown a growing interest in using multi-robot system as Tangible User Interfaces (TUIs) for various applications [28]. For example, Asteroids [32] and HoloBots [20] used desktop robots for remote communication and guidance. SwarmHaptics [27] developed a multi-robot system to generate haptic patterns on the human body. (Dis)Appearables [38] explored multi-robot applied to making TUIs actively appear and disappear from users' vision and attention. Some research has used multi-robot systems as tangible representations of physics simulations [33] and virtual information [57]. Other research [56, 60] has utilized a group of robots to provide haptic feedback in an XR environment. Previous work has also explored intuitive interface designs, such as hand gestures, for directly commanding the movement of groups of robots [23, 26].

While these works have contributed to the field by exploring novel hardware design, swarm movement control interfaces, and innovative applications of multi-robot as TUIs, our paper focuses on how an object-level instructed multi-robot system can interact with free-form physical objects in a tabletop environment to enrich tangible interactions. Push-That-There provides user interface design for object-level tasks and lets the multi-robot system autonomously manage the movement of individual robots. Even though our implementation doesn't fit the strict definition of a swarm system due to its centralized control, its user interface design can still be extended to give high-level object manipulation tasks in swarm systems.

2.2 Tabletop Interactive Object Actuation

HCI researchers have broadly investigated manipulating passive objects through various actuating hardware to design novel tangible user experiences via dynamically animated objects.

For example, actuating magnets on an XY gantry [2, 42, 62] or electromagnet arrays [43, 61] hidden under tabletop surfaces have been employed to actuate and control ferromagnetic passive objects for interaction design and HCI applications. Ultrasonic acoustic transducers are another approach often employed for object actuation, specifically to move very light objects on tabletop surfaces [35], or even to levitate them in 3D space [41, 45]. Pin-based shape display hardware has also been demonstrated to move balls and smartphones [17], to flip cards [59], to assemble blocks [52], or to animate passive handcrafts [39].

Our work emphasizes using tabletop mobile robots in passive object actuation. It leverages the mobility and simplicity inherent to multi-robot systems. In contrast to the above systems, these systems are generally applicable to moving a wide range of objects without requiring those objects to be either ferromagnetic or extremely light in weight. Multi-robot system provide cost-effective, fault-tolerant, and reliable solutions for numerous automated applications [58]. They are characterized by scalability, robustness, and adaptability [51]. Compared to systems based on robotic arms, mobile desktop robots utilize compact hardware, potentially making the system less obtrusive and more portable. These characteristics open up possibilities to use tabletop multi-robot systems for broader and more natural interactions in object actuation.

While using multi-robot system to move passive objects has been broadly explored and envisioned for pushing smartphones [28], assembling blocks [63], cleaning tables [57], activating passive mechanisms [37], and dropping balls from ceilings [34], none of these systems were intended to identify free-form objects on tabletops automatically, and then to coordinate a group of robots to move those objects to target destinations collectively. While researchers have envisioned mobile robots to be ubiquitously embedded in everyday environments, we believe that building a system that can intelligently manipulate unmarked free-form objects on tabletop surfaces is crucial for seamless future swarm interaction with everyday objects. One of our specific goals was to enable *object-level control*, whereby users can instruct the system to move objects to the target destination rather than needing to specify individual robots' movement. Users can focus more on high-level object management and directly engage with the physical environment, thereby enhancing the user-centric interaction experience.

2.3 Multi-robot Object Handling Control

Object handling with multi-robot system has been a popular topic in robotics research. Multi-robot control leverages interaction among robots to exhibit collective behaviors [54], inspired by social insects such as ants transporting food [13]. Many object-handling work in robotics focused on improving algorithm control evaluated in simulation [24]. However, some multi-robot control algorithms are feasible in simulation or large open spaces but are not easy to use in an interactive desktop environment with random objects. For example, Chen et al. introduced a strategy relying solely on each robot's local information for object handling, which is incapable of object collision avoidance and is limited in handling parallel tasks [10]. Gebhardt et al. presented control algorithms for small swarm robots to stack up all in one direction when manipulating objects [18]. This approach occupies a large amount of space surrounding the items, which might not be suitable for a tabletop workspace because it limits the table's usable area for other human activities and greatly reduces the number of potential items the table can hold. Some systems only focus on handling objects with specific shapes [5, 19]. They may not function efficiently in

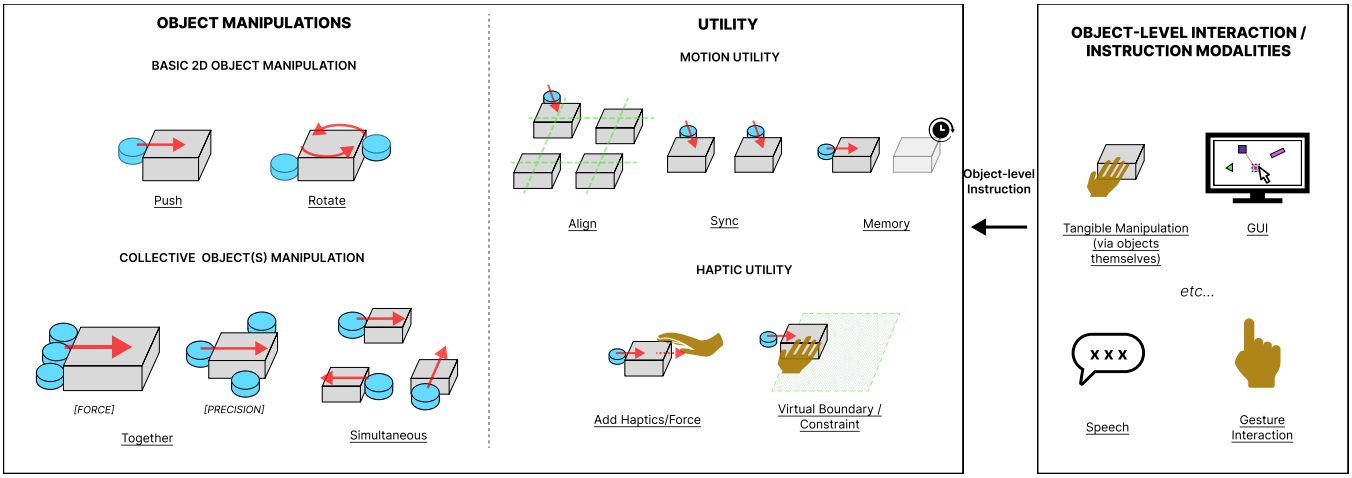

Figure 2: Design Space of Push-That-There

dynamic tabletop environments with arbitrary free-form objects to interact with. More importantly, none of the above contributions elaborated on the integration with human interaction.

Our system centers on designing a multi-robot system specifically for an interactive tabletop environment, and we focus on providing a generalizable system that can be easily integrated with various user interfaces via object-level instructions. We developed a multi-robot control algorithm for managing the positions and orientations of various free-form objects simultaneously. The system includes features specifically designed for interactive tabletop settings, such as "hand region removal" to enable human hands to engage with objects, and robots' "exit" behavior to leave more space for users' activities. Furthermore, we have integrated a multimodal user interface that translates human input into object-level instructions for the autonomous robot control system.

3 DESIGN SPACE OF PUSH-THAT-THERE

The design space of Push-That-There, as depicted in Figure 2, encompasses three primary aspects: 1. basic and collective object manipulation methods with multi-robot system, 2. motion-based and haptic-based utility of the object manipulation, and 3. user instruction and interaction modalities for the system. The design space comprehensively overviews opportunities in Push-That-There's object handling capabilities, utility, and interactivity. Together with the scalability of using multi-robot control, Push-That-There offers insights into its potential as a generalizable contribution.

3.1 Object Manipulation

The first aspect of the design space is *Object Manipulation*, which is the core functionality and fundamental purpose of Push-That-There: the manipulation and control of passive objects on a 2D plane. While pushing objects with a multi-robot system has been explored in prior literature [28, 38, 57], this section comprehensively reviews such capabilities to ground our work to interactively manipulate objects with a multi-robot system.

Basic 2D Object Manipulation. Autonomous object manipulation on a 2D plane includes *pushing* objects to specific locations and *rotating* them around their center to certain orientations. These fundamental functionalities are building blocks for more advanced features and can be extended to support collective manipulation and interactive utility.

Collective Object Manipulation. In tabletop settings, objects vary in size. Our multi-robot control is scalable and can intelligently assign a variable number of robots to each object dynamically, optimizing the system's efficiency based on the object's dimensions and the available robots. Multiple robots can work together to manipulate a single object by combining their effort for *increased force* (e.g., multiple robots push a large object together from the same side, as demonstrated in [28, 37]), and *increased precision* (by assigning more robots across the object's edge, the robots can better stabilize objects during movement, reducing unintended deviations from the desired path). This dynamic approach enhances the system's scalability by accommodating multiple tasks with a proper number of robots. Furthermore, Push-That-There can handle multiple objects simultaneously, improving efficiency in managing diverse objects.

3.2 Utility

Based on its basic 2D object-handling capabilities, Push-That-There enables several utility functions oriented toward handling passive objects. These utilities can be categorized into basic motion utilities and haptic utilities.

3.2.1 *Basic Motion Utilities.*

Object Alignment. The system can align objects with each other based on their relative positions by utilizing pushing and rotating actions automatically (referred to as *Auto-arrangement Mode* in the later sections). While it is common in 2D digital CAD UI to align graphical objects via center, left/right, or grid alignment, our system extends this utility to physical objects on tabletop surfaces.

This can be useful in use cases such as automatically sorting items on a table to a more organized configuration.

Object Synchronization. The system can synchronize the positions and orientations of different pairs or sets of objects. When the user manipulates an object, the system can assist the user in replicating the same movement on another object. This feature improves efficiency in performing repetitive tasks, such as setting up tableware for multiple dinner guests or facilitating remote communication with synchronized passive objects [8, 9, 30].

Arrangement Memorization. The system can remember the configuration of a workspace by storing the previous positions and orientations of objects. It can then use this information as a target configuration for objects in subsequent interactions, allowing the system to restore the desktop configuration to previous states automatically. This feature facilitates such use cases as preparing the table for different occasions based on users' past habits. While storing and retrieving physical configurations has been presented in earlier work in conceptual form [29, 59], Push-That-There's object-level coordination and interactive functionalities make it practical for passive objects on tabletop surfaces.

3.2.2 Haptic and Tangible Utilities.

Adding force to passive objects. Tabletop robots can exert force feedback on passive objects, counteracting user movements by pushing against the objects, as demonstrated in [27]. This transforms everyday objects into bi-directional force feedback tangible interfaces [36], which can enhance haptic sensations, particularly in immersive embodied experiences.

Defining boundaries/constraints. The system can help establish boundaries for objects, either confining them within or outside the boundaries or constraining their movement along specific directions. As discussed in [44], these constraints enable the system to perform essential configuration maintenance tasks. For example, they ensure that objects remain at a safe distance from table edges, preventing accidental falls. Additionally, these constraints create immersive and tangible experiences for users. For instance, these constraints enable users to interact with an object in a way that emulates the functionality of another object, such as mimicking the behavior of a sliding knob by restricting the motion of a cylindrical-shaped object to a specific direction.

3.3 Interaction Methods

In alignment with a user-centric design approach, object-level instructions enable users to guide multiple robots to move passive objects rather than directly controlling individual robots. This versatility allows users to engage through various input modalities, enhancing the system's intuitiveness, ease of use, and adaptability to user preferences. We connect several interaction modalities to our multi-robot control system, including directly physically manipulating tabletop objects (tangible manipulation), desktop GUI controls, gesture interaction, and speech commands, showcasing our system's potential to easily integrate with different user interaction interfaces through object-level instructions, helping to provide intuitive user experience and to extend the system's accessibility.

The interaction and instruction modalities listed below are explored in our Push-That-There prototype.

Graphic User Interface (GUI). GUI is useful for precise object layout planning and remote object control. We explored connecting the GUI with Push-That-There's multi-robot control system by live streaming a video feed of the desktop environment to users. Users can interact with physical objects through actions such as dragging and dropping the objects in the video to relocate them, as well as scrolling or pressing keys to rotate them. All the operations can be executed using input devices such as a computer mouse or touchscreen. We provide visual feedback for users in our GUI design. As shown in figure 6, the computer vision module can identify the target object that the user selects, and then a semi-transparent replica of the object's cutout can be displayed at the user-defined position and orientation. This replica serves as a visual reference for the desired object handling result. While the tangible manipulation modality requires a complete physical setup on the user's side, the GUI control modality provides users with the convenience of remotely interacting with objects using everyday devices such as laptops, tablets, or smartphones.

In addition to directly providing target placement (we call this feature *Target-Position Control Mode* in later sections), the GUI can enable users to provide more complex object-level instruction, such as defining a path for the item to be pushed along (*Target-Trajectory Control Mode*) by sketching it with the mouse or touch screen. The robots can then automatically figure out how to move the object along the user-defined path.

Gesture Interaction. To facilitate moving objects over large distances or beyond reach, users can employ gestures to give commands. As shown in Figure 7, we integrate a simple gesture interface with Push-That-There. We cast a ray from the user's hand, and render its intersection point with the tabletop surface for the user as a cursor, suggesting where they are aiming. Users can target objects on the table, and use mid-air pinching gestures to confirm their target selection. They can also define the object's desired position in the same way.

Speech. As speech input becomes increasingly prevalent in our everyday environments, driven by the widespread adoption of consumer voice assistant interfaces (e.g. Siri, Alexa, etc.), Push-That-There has the potential to offer users speech modes for object-level instructions. This hands-free interaction approach can be helpful when the user's hands are occupied, or when they need assistance in retrieving objects beyond their reach. Users can refer to the objects by name, using speech to instruct the system how to handle the objects. For example, users could simply say commands such as "move Object 1 to me". While this type of user interface modality has been explored in gestural graphical systems [7] or human-robot interaction (HRI) systems [53], Push-That-There aims to prototype this concept within the context of user interfaces specifically designed for passive objects on a 2D surface.

Tangible Manipulation. Push-That-There autonomously arranges objects on a tabletop into desired positions, aligning with users' direct manipulation of other objects on the same or separate tabletops. This approach extends the concept of "programming by demonstration" within the realms of robotics and actuated user

interfaces [11, 46]. It captures object positions in real-time and uses multi-robot system to replicate these arrangements, allowing simultaneous task execution by humans and robots in shared spaces. This provides a fast and intuitive method for demonstrating object movement. To enable this interaction modality, the system captures real-time object placements and replicates them onto other objects using the pushing capabilities of multi-robot system. This could be a real-time movement replication between two objects (referred to as *Movement Replication* in the later sections) or a replication on the layout of a set of objects (*Layout Replication*) building on top of the system's *Arrangement Memorization* feature.

4 IMPLEMENTATION OF PUSH-THAT-THERE

This section overviews the implementation of our system, following a list of criteria specifying the required technical elements for Push-That-There.

4.1 Criteria and Goal of the System

We aim to create an interactive multi-robot system capable of autonomously moving free-form objects in response to users' object-level instruction in a tabletop environment.

To achieve this objective, we need the following components:

(1) **Arbitrary and On-Demand Object Detection:** a module to identify and detect free-form passive objects on tabletop surfaces. Specifically, it needs to track objects' positions, orientations, and contours in order for the multi-robot system to manipulate the objects adaptively.
(2) **Autonomous Multi-robot Control:** an autonomous control mechanism designed for mobile tabletop robots. It orchestrates detected objects' movement and aligns objects with their target positions and orientations. Importantly, this control system needs to be adaptable to varying numbers of robots and objects, enabling efficient object manipulation and adaptive to dynamic tabletop environments.
(3) **System Adjustment for Tabletop Setting:** a multi-robot control system tailored to fit the tabletop environment in which space is limited, objects are dynamically brought out or taken into the workspace, and people's hands frequently interact with the system.
(4) **Multimodal User Inputs:** multimodal user interfaces for users to provide object-level instructions. We integrate diverse interaction methods to cater to different user preferences and scenarios, showing the system's potential to integrate with different user interfaces to make the system accessible and user-friendly.

The implementation of our control algorithm is engineered for generalizability, making it compatible with diverse multi-robot systems. The system's ability to process object-level instructions simplifies its integration with various user interfaces, as it automatically computes the multi-robot behavior based on users' intentions for interacting with objects.

4.2 Overall Hardware Setup

Figure 3 shows our overall hardware setup. For the robots, we employed a commercially available (currently in Japan and China) two-wheeled robotic toy, *toio* [12], developed by *Sony Interactive Entertainment*. These robots can localize their 2D positions relative to toio mats, often used in recent HCI research [37, 56]. For Push-That-There, we 3D printed cylinder shells for the toio robots to make them round, making the pushing behavior more predictable and reproducible when the robots contact and push the objects, as a round shape allows for uniform contact with objects. We installed metal sheets under the toio mat to increase the torque of the toio robots. A *Logitech Brio 4K Ultra HD Webcam* tracks objects on the tabletop surfaces. The camera is mounted to face the tabletop with a fixed upside-down angle from a distance of 1200mm. The tabletop surface is 840$mm \times 891mm$ and is fully covered by toio mats, representing the size of a common tabletop.

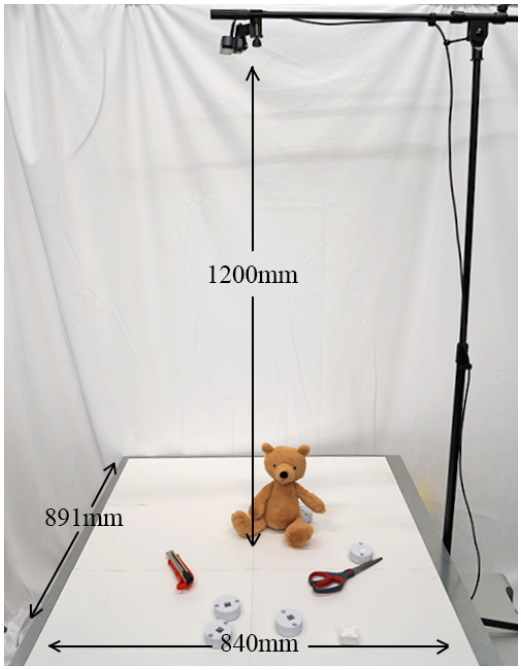

Figure 3: Hardware setup of Push-That-There

4.3 Software

The software can be divided into three parts: computer vision (CV), multi-robot object handling system, and multimodal interaction interface (see Figure 4).

4.3.1 *Computer Vision.*

Object Detection and Tracking. The system employs OpenCV to detect and track contours for all objects within the video feed. These contours are first numbered and ordered based on their size and spatial relationships so the system can maintain naming consistency for each object in subsequent frames while tracking the object's movements. We take the average sum of the object's contour point position to calculate the object's position. We perform Principle Component Analysis (PCA) on the contour to calculate object orientation.

Contour Simplification. The contours of all the objects within the video feed are transmitted to the robot control system as point

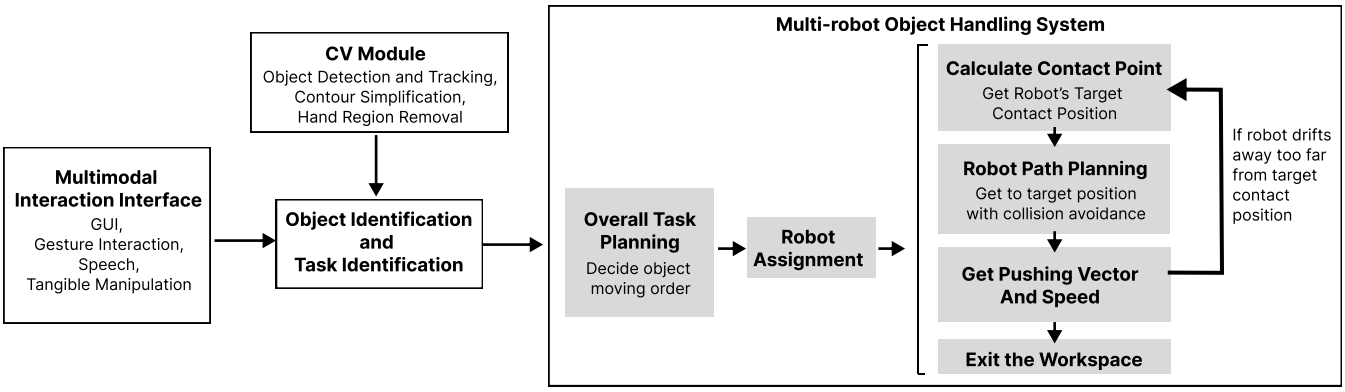

Figure 4: System software pipeline

arrays that outline their boundaries. Before the transmission, the Ramer-Douglas-Peucker (RDP) algorithm is applied to streamline the contours by describing each contour with fewer points [14, 47]. This lessens data transmission load and enhances performance.

Hand Region Removal. Notably, the system excludes the user's hand from detecting and tracking by using color segmentation in the HSV color space. This ensures that the robots can accurately locate target object boundaries with less interference from human hands, making the system more adaptive to an interactive desktop environment. The hands are excluded during object contour processing, but are still included in the robots' path planning computation, avoiding collisions between the robots and the user's hands.

4.3.2 Multi-robot object handling system. We aim to develop a multi-robot system for autonomously handling objects based on object-level instructions. It has to intelligently manage free-form objects simultaneously, allocating robots according to their size, and ensuring tasks like moving multiple items are done without collisions. The system will conserve limited desktop space by keeping robots out of the main workspace when not in use.

To do so, we need to perform the following steps: 1. If the task involves moving multiple objects simultaneously, plan the object manipulation task to avoid collision; 2. Decide how to assign the robots to the manipulated objects; 3. Determine the robot's contact point with the object when pushing it; 4. Do robot path planning to guide each robot to the contact point without collision; 5. When the robots reach the contact point, determine their pushing vector and speed to handle the objects according to the object-level instructions; 6. When the robots finish handling the object, let them leave the center of the workspace.

Technical implementation details are provided below.

Overall Task Planning. If the tasks involve moving multiple objects, we need to do overall task planning first to figure out whether we need to move some objects in sequence to avoid object collision. We first utilize the Axis-Aligned Bounding Box (AABB) algorithm [4] to determine whether an object lies in the path of another object. This path is computed by extruding the object's bounding box from its current position to its target position with the assumption that the object will be pushed in a straight line. Given two objects, i and j, if the bounding box of object i currently intersects the path of object j, then object i is moved first. In cases where the object is undergoing rotation, we resort to its "bounding circle", a circle centered at the object's centroid with a diameter equal to the longest edge of the bounding box. The task planning updates in real-time to accommodate changes in the interactive environment.

Robot Assignment. We need to dynamically assign robots to objects so that the system can adapt to handling different numbers of objects. The system allocates available robots to objects based on the following criteria:

(1) If an object is prioritized for movement, it is assigned robots first.
(2) For objects that can be moved simultaneously, robots are assigned to the larger object first.

The number of robots allocated to an object is contingent on the object's dimensions. Specifically, the system evaluates whether the object can accommodate robots where each robot must maintain a distance of at least $\frac{1}{3}$ of the robot's diameter (d) from its neighbors for the pushing and aligning phases. For the rotation phase, this distance is set to be the robot's radius based on how its contact points are calculated in equation 1.

Calculate Contact Point. To determine the contact points for the robots to push the object, we first need to determine the edges where these contact points are situated. We observe that pushing on edges where the angle between their normal directions and the object's desired moving directional vector (the vector difference between the object's target position and its center) exceeds 90° can bring the object closer to its target. We aim to distribute robot contact points along these edges uniformly to ensure a stable pushing process. Contact points are determined based on intersections with lines parallel to the pushing direction, as shown by the small red points on the object's edge in Figure 5 (a-2).

As shown in Figure 5 (a-1), we also introduce padding to adjust the spacing for the robots at the outermost positions to make effective contact with the object. In our system, we set padding to at least $0.25d$ to ensure this edge contact.

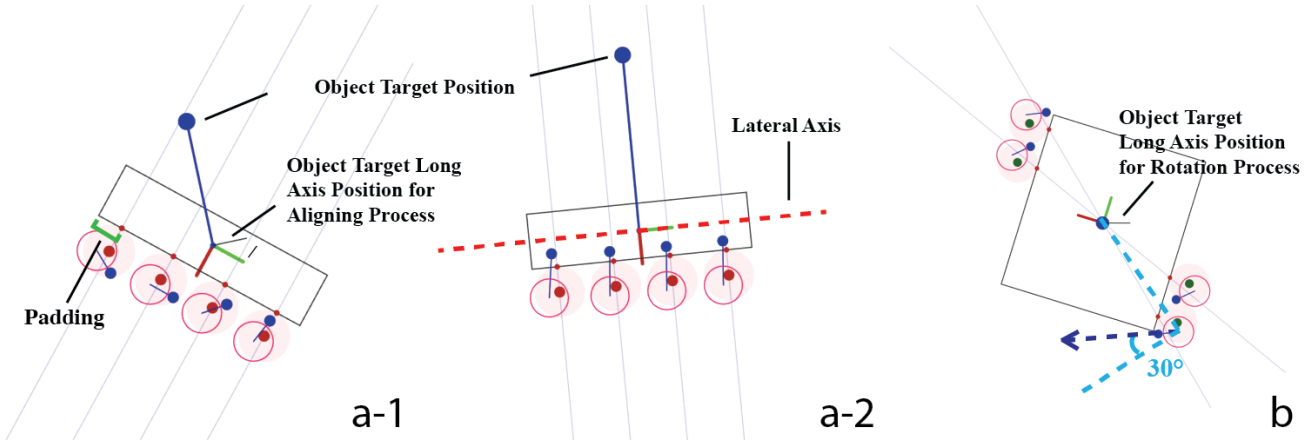

Figure 5: Contact points and robot's pushing vector calculation illustration. (a) Aligning (be prepared to push) process and pushing process. (a-1) Aligning the long object to the optimized pushing orientation before pushing. (a-2) Pushing the object. We use our novel lateral axis (represented by the dashed red line) alignment control to adjust the robot's speed. (b) Rotating the object. We calculate the pushing vector for robots using the vector that is 30° inward of the object from the vector obtained by subtracting the object's center from the robot's position.

As illustrated in Figure 5 (b), to identify contact points when the object is rotating, we use an angle defined by

$$\text{angle} = \arctan\left(\frac{\text{object\_length}}{\text{object\_width} - (1.5 \times i \times d + \text{padding})}\right) \quad (1)$$

and then we calculate where lines spaced at this angle from the object's principal longitudinal axis intersect with the object's contour, where i (integers ranging from 0 to $the\_number\_of\_robots - 1$) represents the index of the lines needed to find points to all the robots assigned to the object. This ensures that robots on the same side are spaced approximately $0.5d$ apart.

For objects with an elongated shape, where the difference between width and height is significant (we use a threshold where the ratio of the bounding box's long edge to its short edge exceeds 2:1), it will be much easier to push if the robots can be allocated more on the long side. Consequently, we first align the object's PCA short edge axes with the intended moving direction, positioning the pushable edges primarily along the longer axis. As is shown in Figure 5 (a-1), the method for determining the contact points during the align operation resembles that for the pushing operation. However, instead of lines parallel to the directional vector, we use lines parallel to the principal short edge axis. We opt for this approach over using the rotation operation for alignment because, during the alignment process, objects don't need to rotate around their centers. This lets us position the contact points on one side of the object, facilitating a more rapid transition from the alignment to the pushing phase.

Robot Path Planning. Given the dynamic changes in the environment as objects are moved, we do not use optimal calculation methods for global robot path planning. Instead, we use the robot's local information and design several rules to prevent the robots from running into obstacles or the other robots when they are approaching their target contact points with the object:

- **Avoidance of Objects:** If a robot approaches too close to an object, and its moving direction is toward the object, the robot will follow the contour of that object until its path toward its target position is no longer blocked by that object.
- **Avoidance of Fellow Robots:** When two robots are in proximity, the following rules apply:
  (1) If one robot has reached its target, it remains stationary while the other navigates around it.
  (2) If both robots move in the same direction, the leading robot has the right of way.
  (3) If two robots are moving toward each other, they navigate around one another to avoid a collision.

Calculate the robot's pushing vector and speed. The short blue vectors on each robot in Figure 5 represent the robot's pushing vector. In the pushing phase, to maintain efficient and stable movement, the robots should push the object along its direction vector while maintaining proximity to their initial contact points. To achieve this, we combine the robot's movement vector — from its current position to the target — with the object's directional vector. The mix ratio of these two vectors depends on the robot's deviation from the optimal contact point, ensuring precise alignment and movement towards the target.

When the robot drifts too far away from its target contact position, we stop the pushing process, realign the object if needed, and redirect the robots back to their target contact positions. This strategy is also applied to the object rotation and alignment processes.

We also aim to suppress unintended object rotation to promote straight-line movement. This can be accomplished by maintaining a constant distance from the contact point to the object's *lateral axis* (the axis perpendicular to the direction of the object's movement, illustrated in Figure 5 (a-2). We adjust the robot's speed accordingly — increasing it when this distance increases, and decreasing it when

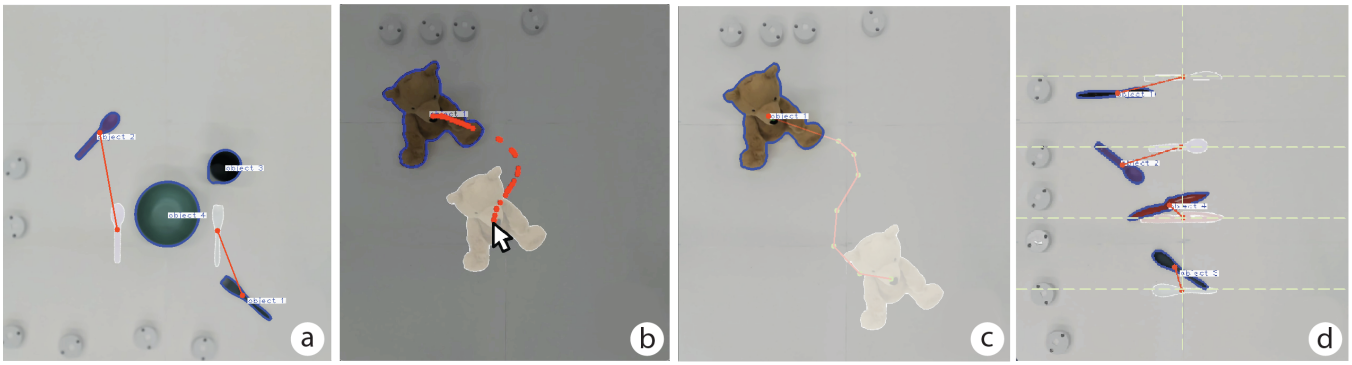

Figure 6: Push-That-There GUI design (a) Target-Position Control Mode. (b) Target-Trajectory Control Mode, where the user can draw the trajectory of the object with a cursor, and the robots figure out how to push the object along the path automatically. (c) The system simplified the trajectory into straight line segments before pushing. (d) Auto arrangement mode

this distance decreases or if the robot crosses to the other side of the object's lateral axis.

During the rotation and alignment phases, we adjusted the robot's pushing direction to be a vector that angles 30° inward towards the object from a baseline vector. This baseline is calculated by subtracting the object's center point from the robot's current position, as depicted in Figure 5 (b).

Exit the Workspace. When the robots finish handling the objects, they will pull back from the object and move to the edge of the table to avoid occupying the main workspace. This feature is useful to leave more space for users in the table's limited area.

4.3.3 Multimodal Interaction Interface. We developed a multimodal user interface, including gesture interaction, graphic user Interface (GUI), speech, and tangible manipulation, to demonstrate our multi-robot system's ability to easily integrate with various user interactions through object-level instructions.

GUI in 3 Modes. The GUI, controlled by the computer mouse, enhances the control and visualization of object movements (See Figure 6). The GUI is based on the top-view video stream of the workspace, and responds to mouse-press on the object, allowing users to create a semi-transparent copy of the object, which can then be placed, when the mouse is released, to represent the target position. The orientation of the object can be rotated by scrolling. There are three modes in the GUI:

(1) **Target-Position Control Mode.** The GUI visualizes the target position, the real-time object position, and the connecting line between them (Figure 6 (a)). The object will be pushed to the target position in a straight line.
(2) **Target-Trajectory Control Mode.** The mouse trajectory can be recorded and rendered in simplified line segments (also processed through RDP as introduced in section 4.3.1) as a means of path planning, which enables users to define a customized path (Figure 6 (b, c)). The object will be pushed to follow the simplified path.
(3) **Auto-arrangement Mode.** The GUI can help users define a grid layout for the automatic arrangement of objects based on the specified layout (Figure 6 (d)). The object's target position will be mapped to the line intersection in the grid, and the object's rotation will be set to 0°.

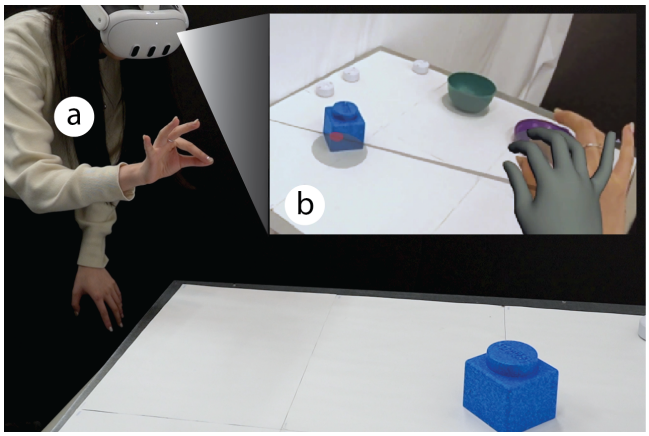

Figure 7: Push-That-There gesture interaction (a) We use Meta Quest 3 with video passthrough to track the hand gesture and render the visual cue. (b) This is what the user sees inside the headset. Users can cast a ray from their hand and pinch to select objects and define their target position.

Gesture Interaction. Users can make gesture interaction with Push-That-There using a wearable XR device, such as Meta Quest 3 with video passthrough. We use the headset's hand-tracking feature to read users' gestures and interpret where the user is targeting by casting a virtual ray from the user's palm to the tabletop surface and calculating the intersection point. To manipulate an object, users target the object and do a pinch gesture; to set a new location, they aim and pinch again at the desired spot immediately after selecting the object. This is shown in Figure 7.

Speech. We employ the Web Speech API [1] for speech detection, which can identify the user's spoken sentences. In our proof-of-concept speech interaction, keywords from these sentences are extracted for further processing.

As Section 4.3.1 mentions, the system automatically assigns a default name tag (e.g. "object 1") to each detected object. Users can alter these tags through the GUI or by employing keyword-based voice commands such as "rename object 1 to knife", Once renamed, users can employ voice commands referencing the updated name to interact with and manipulate the objects.

In addition, the computer vision model tracks the user's position, enabling speech commands such as "hand object 1 to me".

Commands such as "bring", "move away", "to the left", "to the right", and "rotate xxx degrees" are recognized by the system and translated into object-level instructions for the robots. If no specific values are mentioned, the default values used for movement and rotation are 10cm and 30°, respectively.

Tangible Manipulation. When prototyping mirroring of object arrangements between two remote locations, we identify similar objects based on contour size and bounding box ratio similarities, to determine which objects need to be synchronized for our proof-of-concept interaction.

There are two modes of interaction:

(1) **Movement Replication:** For tracking a demo object, we detect the object with which the user's hand interacts. This interaction is identified when there are shared points between the object's contour and the user's hands. If there is no hand interaction for 3 *seconds*, the transformation offset from its original position is recorded and mirrored to the target object.

(2) **Layout Replication:** The relative positions of a set of objects within a predefined region (currently hard-coded for proof-of-concept) are mirrored. If the user's hand ceases to interact with objects within a specific region for 5 *seconds*, the system assumes the configuration to be finalized. It then attempts to mirror this configuration in the other location.

5 TECHNICAL EVALUATION

We conducted a series of technical evaluations to validate the system's performance in pushing and moving everyday freeform tabletop objects. Our evaluation assessed: 1. the speed and precision of handling individual items of various sizes and shapes, and 2. the system's ability to handle multiple objects concurrently.

The objects used in the evaluation include a plastic bowl, scissors, a plastic knife, sunglasses, a phone, a computer mouse, and a paper box (detailed in Figure 8 and Table 1). These items offer a representative sample of everyday objects of various shapes and sizes.

5.1 Evaluation of Individual Object Handling

In this section, we present an evaluation of the time and accuracy of individual object handling operations (i.e., *pushing* and *rotating*). We tested with different numbers of robots ranging from one to four to show the object handling algorithm's scalability and to provide insights into the performance. To gain a comprehensive understanding of the system's efficiency, we conducted ten trials for each object. Each trial had randomized initial and target positions/orientations. We defined a pushing operation as completed when the object was positioned within a 10mm distance from the target location and defined a rotation operation as completed when

Figure 8: The items used in the object handling evaluation.

Item	Dimensions (cm)	Weight (g)
Plastic knife	$19 \times 2 \times 0.5$	15
Sunglasses	$14.5 \times 14.5 \times 8$	33
Plastic Bowl	$14.5 \times 14.5 \times 8$	68
Scissors	$21 \times 7 \times 1$	85
Mouse	$13 \times 7 \times 3$	86
Phone	$16 \times 7.5 \times 0.7$	203
Paper Box	$19 \times 11 \times 9$	596

Table 1: Dimensions and weight of the items. The items are listed in ascending order of their weights.

	1 robot	2 robots	3 robots	4 robots	1 robot	2 robots	3 robots	4 robots	1 robot	2 robots	3 robots	4 robots
items	push time (s)				rotate time (s)				post rotation error (mm)			
knife	27.61 [5.31]	5.95 [0.37]	5.63 [0.41]	†	2.84 [0.76]	2.13 [0.28]	2.41 [0.26]	2.53 [0.44]	32.46 [11.28]	6.43 [3.71]	5.13 [1.95]	5.68 [2.31]
plastic bowl	10.87 [1.94]	4.43 [1.05]	4.084 [0.89]	†	*	*	*	*	*	*	*	*
sun glasses	13.56 [3.52]	5.49 [0.97]	5.51 [0.36]	†	3.85 [0.49]	2.74 [0.62]	2.75 [0.24]	2.18 [0.73]	28.91 [10.04]	14.81 [3.25]	9.36 [2.6]	5.75 [2.17]
scissor	24.97 [5.45]	6.58 [1.10]	5.95 [0.22]	†	4.59 [0.84]	2.99 [0.35]	2.43 [0.29]	2.37 [0.51]	38.6 [13.74]	15.27 [4.89]	12.16 [3.24]	10.54 [4.14]
mouse	14.83 [3.2]	4.81 [0.47]	†	†	3.75 [0.82]	3.12 [0.91]	3.34 [0.82]	3.15 [0.79]	31.51 [9.82]	10.32 [3.46]	8.76 [2.97]	6.91 [2.52]
phone	7.28 [1.12]	5.63 [0.45]	4.83 [0.57]	†	3.13 [0.51]	2.17 [0.86]	2.53 [0.5]	2.41 [0.33]	29.27 [11.1]	9.24 [3.12]	7.81 [1.79]	7.26 [2.04]
box	‡	13.50 [1.14]	9.93 [0.98]	8.12 [1.26]	7.21 [1.48]	4.02 [0.43]	4.00 [0.49]	3.92 [0.56]	40.28 [11.9]	10.66 [3.97]	10.64 [3.81]	8.39 [3.79]

Figure 9: The technical evaluation result. The items are listed in ascending order based on their weight. The value in the cell represents the average result of all the trials under that condition, followed by the standard deviation. Cells with smaller average values will have shorter colored bars in them. Different coloring bars have different scales. In the table, † represents the cases where the objects are too small to the target numbers of robots; ∗ means the target operation does not apply to the objects (e.g. try to orient central-symmetric objects, such as a bowl); ‡ means the object cannot be moved by the assigned numbers of robots.

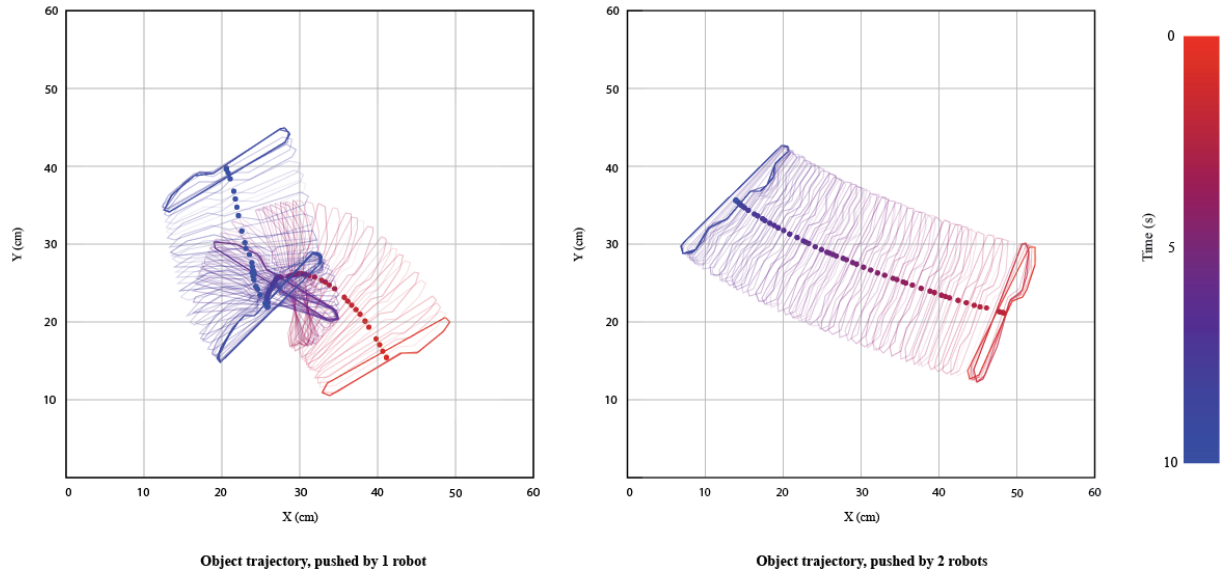

Figure 10: Comparison of object trajectories when pushing a knife with one robot versus two robots. Trajectories are color-coded by time, as indicated by the color bar to the right of the figure. Dots represent the moving object's center point at various moments.

the object's orientation was within 5° from the target orientation. These thresholds are tied to the intrinsic precision level of computer vision detection, robot control, and the size of the used tabletop objects. Despite minor deviations, the thresholds play a crucial role in enhancing system stability and efficiency, preventing excessive sensitivity, and ultimately ensuring that each robot can reach its target without overshooting.

5.1.1 Procedure.

Pushing. To evaluate the performance of pushing an object to the target position, we randomly positioned individual items within the system's workspace and set their target position to be 40*cm* away in arbitrary directions. We measured the duration from the moment the robots initiated their movement to when the object reached its destination (referred to as *push time* in Figure 9). In addition, we recorded the average time required for the robots to travel from random starting positions to their contact points with the objects (around 4*s*).

Rotation. In addition to pushing, we investigated the performance of rotation operation. Each item was given a random initial orientation, and we set a random target orientation that was between +180° and −180° from its initial orientation. We tracked the duration required for the robot to initiate and complete the object's rotation (referred to as *rotate time* in Figure 9). For each rotation trial, we measured the angular accuracy and how much the distance deviated from its initial position (referred to as *post rotation error* in Figure 9) to investigate the positional deviation that occurs during the rotation.

5.1.2 Results and Discussion. We provide our experimental results in detail in Figure 9.

Robot number. Pushing an object with a single robot often proves inefficient, even for lightweight objects like a 15g plastic knife. The main challenge arises from stability issues. When only a single robot is involved in the pushing task, it encounters significant difficulty as it attempts to balance the object. This struggle often leads to the object deviating from the intended direction. Consequently, the robot is forced away from its target contact points, demanding continual realignments and repositioning to effectively resume pushing. Figure 10 shows the trajectories of object movement during a pushing stage with different numbers of robots. In this stage, the system does not consider the object's final orientation. The left diagram uses only a single robot, showcasing the knife's undesired rotational movement during the push resulting from the stability issue previously mentioned. In contrast, the diagram on the right, which represents two robots working in unison, demonstrates that the knife can be moved much more stably (since the trajectory is much smoother without unwanted rotation).

When pushing lightweight objects, such as the 15g plastic knife, there is no significant difference in pushing time when using 2 or 3 robots simultaneously. However, for heavier objects such as the filled paper box with a weight of 596g, assigning more robots can indeed increase the speed of the operations.

In the case of rotating objects, using more robots does not significantly reduce overall rotation time. However, it does contribute to better balancing during movement and more accurate alignment in position and orientation (as reflected by the post-rotation error in Figure 9). The additional robots not only contribute to the pushing

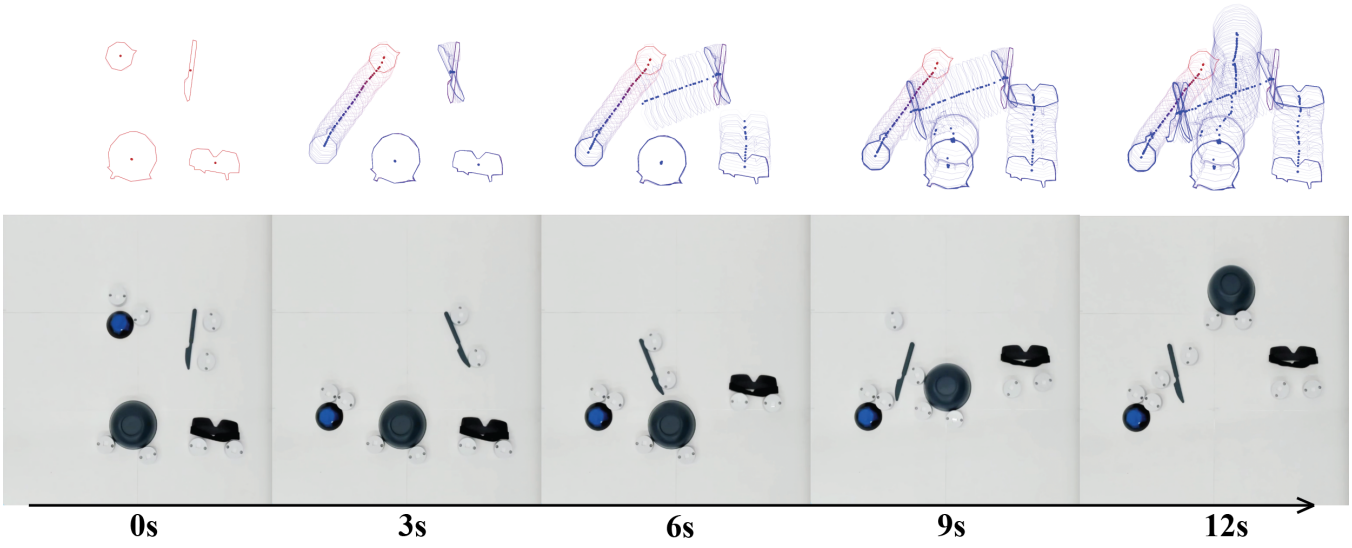

Figure 11: Example task planning in time sequence. The sequence at the top shows the objects' trajectory, and the sequence under it is the top view of the physical workspace.

force but also act as "stopping blocks", constraining the object to avoid excessive undesired drifting.

Different objects. Objects with asymmetric shapes, such as scissors, typically take longer to push to the target position compared to more symmetric objects. The longer duration potentially results from the need to align the object's orientation optimally before initiating a push. In addition, robots need to reposition themselves when there are deviations between the calculated positions and the actual positions of the contact points. Pushing objects with asymmetric shapes is inherently less stable, as reflected by the standard deviation in Figure 9, especially when only a single robot is involved.

For an asymmetrical object, because its center of mass doesn't always align with its geometric center, it is challenging to maintain rotations around its geometric center point. This is reflected by a higher post-rotation error.

It is worth noting that the success rate of handling an object to target position and orientation is 100% for all applicable cases where the assigned robot can move the object in our evaluation.

5.2 Evaluation of Multiple Object Handling

5.2.1 Procedure. Theoretically, our system can manage multiple objects through overall task planning if the task can be done by pushing the objects in straight lines while prioritizing moving some objects before others.

We conducted 30 trials with those theoretically possible situations. Each trial involved four random tabletop objects to represent the dynamic nature of the tabletop environment. We chose to use two robots to handle each object since this is proven to be efficient based on the individual object handling evaluation. The objects were positioned with random initial position and orientation. The target position and orientation were also randomly chosen, with a distance ranging from 10cm to 50cm from their starting position and 0° to 180° from their current orientation. We ensured that at least two paths intersected to test the overall task planning. We then recorded the success rate and computed the average time taken for task completion. Figure 11 shows an example of task planning in time sequence, where the system pushed different objects to targets when their paths crossed each other.

5.2.2 Results and Discussion. The overall task planning algorithm managed multiple objects in most cases (success in 26 out of 30 trials). The completion time ranges from around 18s to 31s with an average completion time of around 24s.

However, certain instances revealed limitations. Failures often happen when objects deviate from their optimal pushing path (the straight path by lining up their initial and target positions). Such deviations led to object collisions that caused the computer vision algorithm to group two objects into one, which caused errors in object tracking and robot assignments. This might also leave insufficient room for the robot to maneuver to its designated contact point on the object.

The trajectory deviations often result from misrepresentation of the contour caused by perspective variation. For instance, objects can appear skewed if positioned far from the camera's central axis. This may lead the system to incorrectly recognize the side faces as part of the object's contour. This can cause the robot to push at non-optimal points, or even at points that are not on the object's real push-able contour, resulting in unpredictable behaviors. Also, objects with deviated weight centers, such as scissors, will be harder to balance during the object handling than asymmetrical objects, such as bowls, making the pushing path difficult to align with the targeted straight line.

We will discuss potential ways to improve these in the Section 7.

Figure 12: Cooking and Dining Application (a) Robots bring a far-away peeler to the user when he needs to peel an apple. (b) The system sets the table for the user by bringing and arranging the tableware in a desired configuration. (c) After dining, robots push the dirty dishes to the edge of the table so they fall into the dish-cleaning bin.

6 APPLICATIONS

Based on our proof-of-concept implementation, we demonstrate potential applications of the Put-That-There, harnessing object-level instruction to manipulate physical objects with tabletop multi-robot system. They each leverage different features, shown in Design Space (Figure 2). Also, Section 6.1 and Section 6.2 were fully developed using our system, while Section 6.3 is shown with a mock-up demo.

6.1 Cooking and Dining Assistant

Push-That-There assists everyday tabletop activities through object-level interaction, facilitating daily tasks such as cooking and dining. The system can act as a helping hand for fetching objects, automating repetitive tasks, and tidying the table, which helps reduce the user's workload. Its versatile user input modalities allow users to choose their preferred interaction methods under different situations.

As depicted in Figure 12 (a), the multi-robot system can proactively assist users by delivering ingredients and utensils in response to speech commands or gestures, particularly when the items are distant or the user's hands are occupied with cooking tasks. If the system is integrated with a recipe application (like in [50]), robots can help users figure out what ingredients or cooking tools are the best to use, and the multi-robot system can deliver that to the user automatically.

When setting the dining table for multiple guests, our system can use its *arrangement memorization* feature to remember and replicate tableware arrangements. The user can demonstrate the arrangement of one set of tableware via Push-That-There's tangible manipulation modality, and the system will automatically apply it to others. During the dining phase, Push-That-There can assist individuals in retrieving food, seasonings, or utensils (Figure 12 (b)) via speech and gesture commands. Moreover, there is great potential to enhance the dining experience through dynamic actuation, such as mimicking diverse cultural elements like rotating sushi displays or Chinese turntables.

Lastly, when guests finish their meal, the multi-robot system can help clean up the table by pushing the dirty plates and utensils to the edge of the table.

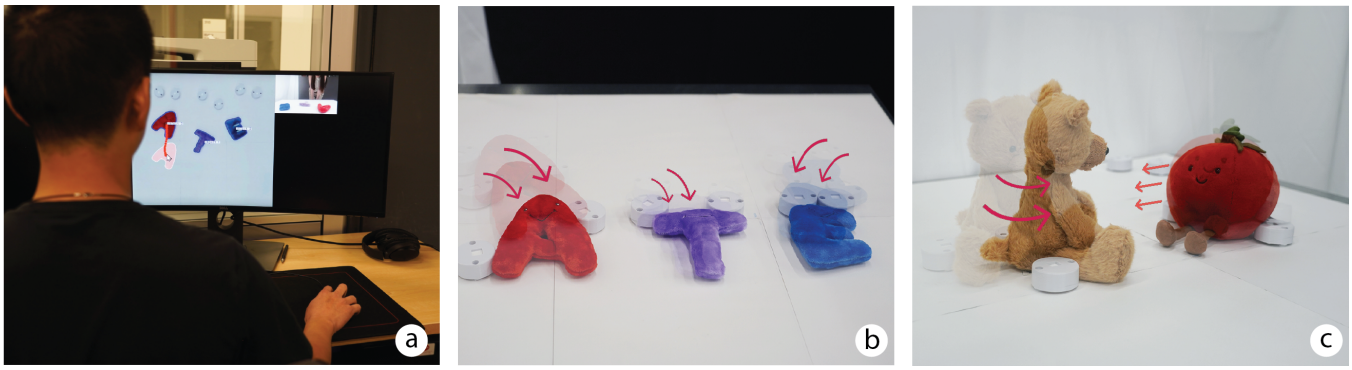

Figure 13: Remote Communication Application (a) John (distant parent) is manipulating the alphabet toys through Push-That-There GUI for remote English teaching on his computer. He can see the live video feed of Mary in the upper-right corner of his display. (b) On Mary's side, the system moves the alphabet toys based on John's remote control. (c) John remotely manipulates Mary's favorite toys for storytelling.

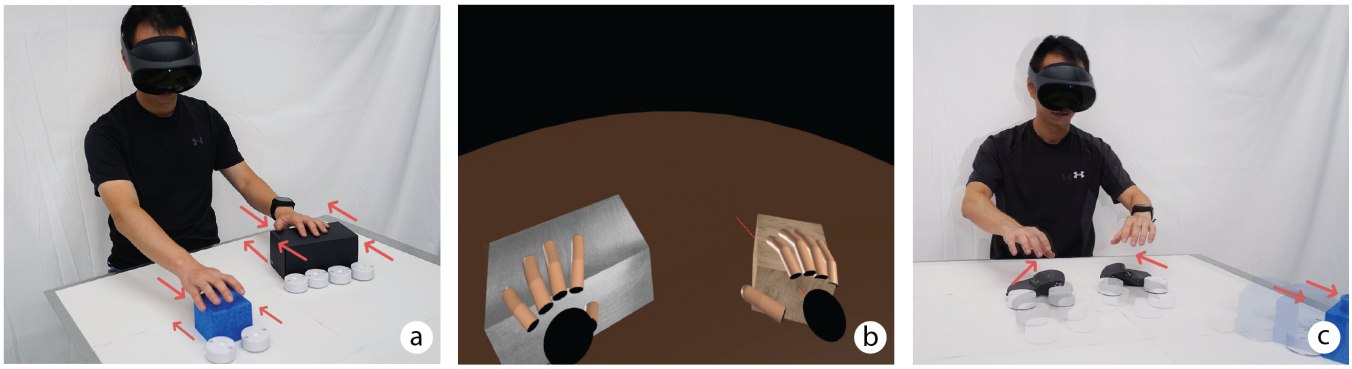

Figure 14: Tangible UI for Immersive Environment Application (a) Push-That-There can provide haptic feedback to a VR user by pushing physical proxies to align with the VR content while adding force feedback. (b) What the user sees in VR. (c) After the user is done playing, the system can remove the physical proxies and bring VR controllers to the user.

We prototype and demonstrate this cooking and dining scenario with a small group of robots accessible to our lab as an example of daily object-handling tasks. It can also be applied to other similar settings, such as handling and arranging tools for a tabletop assembly task. Since our robot control method is scalable, with more robots available, it can help with tasks such as setting a table for 20 guests automatically and simultaneously, which helps reduce manual effort by automating large-scale repetitive tasks.

6.2 Object-mediated Remote Communication

While synced physical actuated devices have been developed for facilitating remote tangible communication [8, 30], our system enables such interaction via everyday passive objects. This, for example, can be used in remote physical education/collaboration applications. In our prototype, shown in Figure 13 (a), we demonstrate a daughter (Mary [she/her]) and a distant parent (John [he/him]) who are interacting via her toys. Mary can bring her favorite toys at home on a table equipped with our system, thanks to Push-That-There's ability to handle free-form unmarked objects, while John controls her toys remotely via the GUI during a video call. As shown in Figure 13 (a), for example, John can use the GUI's *Target-position Control Mode to* place alphabet toys to construct words, facilitating English learning via tangible play. While Mary can flexibly move the alphabet toys, using tactile sensation to better engage in tangible learning.

With other types of toys, such as dolls, John can move Mary's favorite toys to make an *in-situ* remote puppet show as in Figure 13 (c). He can actively control the movement of the toys for storytelling using the GUI's *Target-trajectory Control Mode.* In such a way, the adaptability of our system to manipulate free-form objects on the table based on user instruction has great potential for letting users easily animate everyday objects remotely, and the relatively simple and minimal tabletop robot setup has the potential to allow users or children better focus on the objects, especially compared with bulkier systems, such as those that use robot arms.

6.3 Tangible Interaction in Immersive Environments

As prior works have explored employing pre-defined everyday objects as passive haptic props in VR [16, 64], Push-That-There can be a great method to not only track but also actuate arbitrary unmarked haptic props to enrich immersive experiences. For example, by pushing props against the user, the props can be augmented with haptic force feedback to replicate variable weights of virtual objects, as shown in Figure 13 (a) and (b). The number of robots can be dynamically configured to apply different forces to different props, simulating the different resistances of sliding a light wooden cube and a heavy metal box on the table.

When the user decides to switch to controllers from passive props, Push-That-There can promptly deliver the controllers to the user's hands while efficiently pushing away the physical proxies, ensuring a seamless transition (Figure 14 (c)).

7 LIMITATION AND FUTURE WORK/OPPORTUNITIES

We developed Push-That-There as a proof-of-concept system to explore interactions with desktop autonomous multi-robot system via 'object-level instruction'. Here we discuss the limitations of our current approach and potential future work to share research opportunities with fellow researchers.

7.1 Technical Improvements

7.1.1 Object Detection and Classification. Our system captures the object's contour from a top view using a camera positioned above the workspace. However, this approach does not consistently provide accurate object boundaries that robots can push. For example, for tall objects with a wide top but a narrow bottom, this approach results in a detected contour larger than their actual pushable area, making it hard for the robots to handle the objects efficiently. Similarly, objects featuring concave contours with sharp inward angles may make contact points inaccessible to the robots. Future work could explore alternative technologies for accurately identifying

accessible contact points on object contours. For example, the system could integrate LiDAR sensors into the robots, allowing them to estimate boundaries more accurately as they approach objects.

Besides that, leveraging data-driven methods for object classification can prove advantageous to help the system gain a semantic understanding of the environment, allowing users to provide instructions more intuitively. For example, incorporating deep-learning object detection technologies, such as YOLO [21], could provide informative name tags for the objects, thereby enabling users to describe objects by their class names more efficiently and naturally when using the speech UI. Also, as it is mentioned in Section 5.2.2, the current contour detection method might recognize multiple objects as a single entity if they are directly contacting with each other, which messes up the object tracking and robot assignments. Semantic object detection can potentially reduce this issue.

7.1.2 Object Handling Algorithm. In our present object handling algorithm, robots handle object positioning and orientation separately, which limits potential applications. Enabling simultaneous positioning and orientation could be useful in many scenarios, such as moving a camera across a table while filming, where the camera needs to consistently face its subject.

Currently, the robot control algorithm is designed to move objects in straight lines. While we can mimic curved paths by linking straight segments as we did for GUI *Target-Trajectory Control Mode*, this method is inefficient because it requires frequent robot repositioning around the object.

In addition, the system directly pushes the object toward user-defined target position/path without any detours. This makes it hard to handle objects when they are closely placed, or there are other static objects on their way to their target. Incorporating advanced path planning for the objects would improve Push-That-There's ability to maneuver objects with fewer collisions, especially in cluttered environments.

7.2 Hardware Design and Choice of Robot

While the current cylindrical shape of our robots is optimized for applications that push rigid objects on 2D surfaces, round objects (e.g. ping pong balls), lightweight thin objects (e.g. a piece of paper), or deformable objects (e.g. clay, rope) cannot be stably pushed. Advanced hardware designs could incorporate additional actuators, such as equipping each robot with multiple end effectors [34] to provide adaptive object manipulation. Besides, applying object-level control for a 3D spatial multi-robot system (e.g. drones) would be another exciting direction as it can potentially handle 3D manipulation.

Additionally, the robots employed, Sony Toio two-wheel robots, harness a non-holonomic drive that has limited locomotion ability. While we employed this robot due to ease of control, future research could look into holonomic-drive robots for omnidirectional movements, which would minimize the time to move objects. Our control method and algorithm would be largely applicable to such hardware.

7.3 User Study

While we have established a foundation of technical functionality and a basic design space, more work is needed to better understand the system's usability, user acceptance, and overall user experience. Conducting user studies can shed light on the intuitiveness of our interface and further evaluate the system's efficiency and efficacy in real-world contexts. By comparing the cognitive load, efficiency, and user preferences between *object-level instruction* and *robot-level instruction* (whereby users manually guide each robot), we can gain invaluable insights for designing user-centric multi-robot interaction systems.

7.4 Interaction Interface

As a first step to showcase our system's ability to integrate with diverse user interfaces through object-level instruction, we choose the basic interfaces and interaction modules including gestures, GUI, TUI, and Speech UI. Inspired by the "Put-That-There" [6] approach of blending gestures and speech for content manipulation, exploring how to combine these multimodal inputs for a smoother dynamic user experience presents an exciting future direction.

In addition, one of the advantages of having object-level instruction is that the multi-robot control system can easily be connected with programs that can efficiently process object-level information. For example, we can integrate recent advances in Natural Language Processing AI to process complex multi-object instructions from users. Users can give the AI a more general object manipulation command, such as "sort my stationery in order of size", or "arrange the apples into a star shape", and let the AI figure out where to put individual objects and pass that detailed object-level command to our multi-robot system.

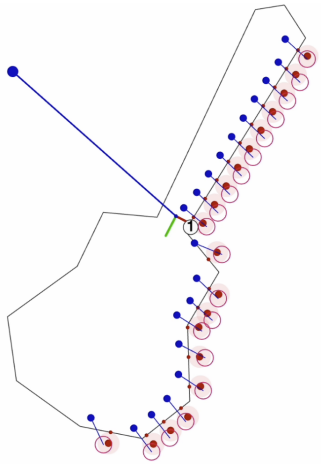

Figure 15: Simulation of our robot assignment algorithm with larger objects and more robots. In this figure, we are assigning 20 robots to push a guitar.

7.5 Scalability

Similar to research on miniature robots such as Kilobot [49] or larger robots such as iRobot Roomba [22], it is essential to explore the system's scalability to assess its adaptability to varying quantities and sizes of robots. This exploration can facilitate the manipulation of objects across a wide range of scales. Figure 15 shows the results

of our conducted preliminary simulation test to evaluate how our control may be able to handle large-scale objects (e.g. a guitar model - approximately 1 meter in length), employing 20 robots. These initial tests demonstrate the adaptability of our system to larger-scale objects and increased robot quantities. Further experimentation, algorithmic refinements, and technological advancements (such as scalable object detection) will be crucial for optimizing the system's scalability across different numbers and sizes of robots and objects, beyond tabletop scales. The general adaptability of our approach in different scales can contribute toward building future environments where hundreds or thousands of micro-robots facilitate interaction with various passive objects in our everyday environment.

8 CONCLUSION

In this paper, we presented Push-That-There, an interactive multi-robot system to handle objects on a tabletop using object-level instructions. We elaborated on how the multi-robot system integrates computer vision techniques with a generalizable robotic control algorithm to handle free-form objects autonomously and collectively. With versatile user instruction modalities including gesture interaction, GUI, tangible manipulation, and speech, we have enabled a wide range of applications from hands-free object manipulation to autonomous tabletop reconfiguration, and allowed users to choose their preferred way of working with the system. Our technical evaluation demonstrated the system's efficiency and feasibility in handling various everyday desktop objects individually and simultaneously. We also outlined current limitations and described possible avenues for future research. We aspire to offer the research community an accessible technological solution and to pave the way for a user-centric multi-robot ecosystem through our *object-level interaction* approach.

REFERENCES

[1] Julius Adorf. 2013. Web speech API. *KTH Royal Institute of Technology* 1 (2013).

[2] Takafumi Aoki, Takashi Matsushita, Yuichiro Iio, Hironori Mitake, Takashi Toyama, Shoichi Hasegawa, Rikiya Ayukawa, Hiroshi Ichikawa, Makoto Sato, Takatsugu Kuriyama, et al. 2005. Kobito: virtual brownies. In *ACM SIGGRAPH 2005 emerging technologies*. 11–es.

[3] Ross Arnold, Kevin Carey, Benjamin Abruzzo, and Christopher Korpela. 2019. What is a robot swarm: a definition for swarming robotics. In *2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (uemcon)*. IEEE, 0074–0081.

[4] Gino van den Bergen. 1997. Efficient collision detection of complex deformable models using AABB trees. *Journal of graphics tools* 2, 4 (1997), 1–13.

[5] Filippo Bertoncelli, Mario Selvaggio, Fabio Ruggiero, and Lorenzo Sabattini. 2022. Task-Oriented Contact Optimization for Pushing Manipulation with Mobile Robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1639–1646. https://doi.org/10.1109/IROS47612.2022.9982177

[6] Richard A. Bolt. 1980. "Put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (Seattle, Washington, USA) *(SIGGRAPH '80)*. Association for Computing Machinery, New York, NY, USA, 262–270. https://doi.org/10.1145/800250.807503

[7] Richard A Bolt. 1980. "Put-that-there" Voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*. 262–270.

[8] Scott Brave and Andrew Dahley. 1997. inTouch: a medium for haptic interpersonal communication. In *CHI'97 extended abstracts on human factors in computing systems*. 363–364.

[9] Scott Brave, Hiroshi Ishii, and Andrew Dahley. 1998. Tangible interfaces for remote collaboration and communication. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. 169–178.

[10] Jianing Chen, Melvin Gauci, Wei Li, Andreas Kolling, and Roderich Groß. 2015. Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots. *IEEE Transactions on Robotics* 31, 2 (2015), 307–321. https://doi.org/10.1109/TRO.2015.2400731

[11] Allen Cypher and Daniel Conrad Halbert. 1993. *Watch what I do: programming by demonstration*. MIT press.

[12] Sony Design. n.d.. Toy platform toio™ / stories / sony design. https://www.sony.net/SonyInfo/design/stories/toio/

[13] Pollyanna G Faria Dias, Mateus C Silva, Geraldo P Rocha Filho, Patricia A Vargas, Luciano P Cota, and Gustavo Pessin. 2021. Swarm robotics: A perspective on the latest reviewed concepts and applications. *Sensors* 21, 6 (2021), 2062.

[14] David H Douglas and Thomas K Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization* 10, 2 (1973), 112–122.

[15] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. 1993. A taxonomy for swarm robots. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, Vol. 1. IEEE, 441–447.

[16] Cathy Mengying Fang, Ryo Suzuki, and Daniel Leithinger. 2023. VR Haptics at Home: Repurposing Everyday Objects and Environment for Casual and On-Demand VR Haptic Experiences. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–7.

[17] Sean Follmer, Daniel Leithinger, Alex Olwal, Akimitsu Hogge, and Hiroshi Ishii. 2013. InFORM: Dynamic Physical Affordances and Constraints through Shape and Object Actuation. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 417–426. https://doi.org/10.1145/2501988.2502032

[18] Gregor H.W. Gebhardt, Kevin Daun, Marius Schnaubelt, and Gerhard Neumann. 2018. Learning Robust Policies for Object Manipulation with Robot Swarms. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 7688–7695. https://doi.org/10.1109/ICRA.2018.8463215

[19] B Hichri, Lounis Adouane, J-C Fauroux, Youcef Mezouar, and Ioan Doroftei. 2016. Cooperative mobile robot control architecture for lifting and transportation of any shape payload. In *Distributed Autonomous Robotic Systems: The 12th International Symposium*. Springer, 177–191.

[20] Keiichi Ihara, Mehrad Faridan, Ayumi Ichikawa, Ikkaku Kawaguchi, and Ryo Suzuki. 2023. HoloBots: Augmenting Holographic Telepresence with Mobile Robots for Tangible Remote Collaboration in Mixed Reality. *arXiv preprint arXiv:2307.16114* (2023).

[21] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. 2022. A Review of Yolo algorithm developments. *Procedia Computer Science* 199 (2022), 1066–1073.

[22] Joseph L Jones. 2006. Robots at the tipping point: the road to iRobot Roomba. *IEEE Robotics & Automation Magazine* 13, 1 (2006), 76–78.

[23] Florian Kennel-Maushart, Roi Poranne, and Stelian Coros. 2023. Interacting with Multi-Robot Systems via Mixed Reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 11633–11639. https://doi.org/10.1109/ICRA48891.2023.10161412

[24] A Khozaee and A Ghaffari. 2009. Moving chain: a surround-and-push method in object pushing using swarm of robots. *Journal of Applied Sciences* 9, 1 (2009), 15–26.

[25] Lawrence H. Kim, Daniel S. Drew, Veronika Domova, and Sean Follmer. 2020. User-defined Swarm Robot Control. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Honolulu</city>, <state>HI</state>, <country>USA</country>, </conf-loc>) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376814

[26] Lawrence H. Kim, Daniel S. Drew, Veronika Domova, and Sean Follmer. 2020. User-defined Swarm Robot Control. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Honolulu</city>, <state>HI</state>, <country>USA</country>, </conf-loc>) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376814

[27] Lawrence H. Kim and Sean Follmer. 2019. SwarmHaptics: Haptic Display with Swarm Robots. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300918

[28] Mathieu Le Goc, Lawrence H. Kim, Ali Parsaei, Jean-Daniel Fekete, Pierre Dragicevic, and Sean Follmer. 2016. Zooids: Building Blocks for Swarm User Interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 97–109. https://doi.org/10.1145/2984511.2984547

[29] Daniel Leithinger. 2010. *Design and implementation of a relief interface*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[30] Daniel Leithinger, Sean Follmer, Alex Olwal, and Hiroshi Ishii. 2014. Physical telepresence: shape capture and display for embodied, computer-mediated remote collaboration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 461–470.

[31] Jiahao Li, Alexis Samoylov, Jeeeun Kim, and Xiang'Anthony' Chen. 2022. Roman: Making Everyday Objects Robotically Manipulable with 3D-Printable Add-on

Mechanisms. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–17.

[32] Jiannan Li, Maurício Sousa, Chu Li, Jessie Liu, Yan Chen, Ravin Balakrishnan, and Tovi Grossman. 2022. ASTEROIDS: Exploring Swarms of Mini-Telepresence Robots for Physical Skill Demonstration. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 111, 14 pages. https://doi.org/10.1145/3491102.3501927

[33] Jiatong Li, Ryo Suzuki, and Ken Nakagaki. 2023. Physica: Interactive Tangible Physics Simulation Based on Tabletop Mobile Robots Towards Explorable Physics Education. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) *(DIS '23)*. Association for Computing Machinery, New York, NY, USA, 1485–1499. https://doi.org/10.1145/3563657.3596037

[34] Ting-Han Lin, Willa Yunqi Yang, and Ken Nakagaki. 2023. ThrowIO: Actuated TUIs that Facilitate "Throwing and Catching" Spatial Interaction with Overhanging Mobile Wheeled Robots. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[35] Mark Marshall, Thomas Carter, Jason Alexander, and Sriram Subramanian. 2012. Ultra-tangibles: creating movable tangible objects on interactive tables. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2185–2188.

[36] Ken Nakagaki, Daniel Fitzgerald, Zhiyao Ma, Luke Vink, Daniel Levine, and Hiroshi Ishii. 2019. inforce: Bi-directionalforce'shape display for haptic interaction. In *Proceedings of the thirteenth international conference on tangible, embedded, and embodied interaction*. 615–623.

[37] Ken Nakagaki, Joanne Leong, Jordan L. Tappa, João Wilbert, and Hiroshi Ishii. 2020. HERMITS: Dynamically Reconfiguring the Interactivity of Self-Propelled TUIs with Mechanical Shell Add-Ons. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 882–896. https://doi.org/10.1145/3379337.3415831

[38] Ken Nakagaki, Jordan L Tappa, Yi Zheng, Jack Forman, Joanne Leong, Sven Koenig, and Hiroshi Ishii. 2022. (Dis)Appearables: A Concept and Method for Actuated Tangible UIs to Appear and Disappear Based on Stages. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 506, 13 pages. https://doi.org/10.1145/3491102.3501906

[39] Ken Nakagaki, Udayan Umapathi, Daniel Leithinger, and Hiroshi Ishii. 2017. AnimaStage: hands-on animated craft on pin-based shape displays. In *Proceedings of the 2017 Conference on Designing Interactive Systems*. 1093–1097.

[40] Iñaki Navarro and Fernando Matía. 2013. An introduction to swarm robotics. *Isrn robotics* 2013 (2013), 1–10.

[41] Yoichi Ochiai, Takayuki Hoshi, and Jun Rekimoto. 2014. Pixie Dust: Graphics Generated by Levitated and Animated Objects in Computational Acoustic-Potential Field. In *ACM SIGGRAPH 2014 Posters* (Vancouver, Canada) *(SIGGRAPH '14)*. Association for Computing Machinery, New York, NY, USA, Article 83, 1 pages. https://doi.org/10.1145/2614217.2614219

[42] Masayuki Orihara and Koji Tsukada. 2019. PartsSweeper: interactive workbench to casually organize electronic parts and tools. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. 339–341.

[43] Gian Pangaro, Dan Maynes-Aminzade, and Hiroshi Ishii. 2002. The Actuated Workbench: Computer-Controlled Actuation in Tabletop Tangible Interfaces. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology* (Paris, France) *(UIST '02)*. Association for Computing Machinery, New York, NY, USA, 181–190. https://doi.org/10.1145/571985.572011

[44] James Patten and Hiroshi Ishii. 2007. Mechanical constraints as computational constraints in tabletop tangible interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 809–818.

[45] Diego Martinez Plasencia, Ryuji Hirayama, Roberto Montano-Murillo, and Sriram Subramanian. 2020. GS-PAT: high-speed multi-point sound-fields for phased arrays of transducers. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 138–1.

[46] Hayes Solos Raffle, Amanda J Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 647–654.

[47] Urs Ramer. 1972. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing* 1, 3 (1972), 244–256.

[48] D. Rosenfeld, M. Zawadzki, J. Sudol, and K. Perlin. 2004. Physical objects as bidirectional user interface elements. *IEEE Computer Graphics and Applications* 24, 1 (2004), 44–49. https://doi.org/10.1109/MCG.2004.1255808

[49] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. 2012. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*. 3293–3298. https://doi.org/10.1109/ICRA.2012.6224638

[50] Ayaka Sato, Keita Watanabe, and Jun Rekimoto. 2014. Shadow cooking: situated guidance for a fluid cooking experience. In *Universal Access in Human-Computer Interaction. Aging and Assistive Environments: 8th International Conference, UAHCI 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part III 8*. Springer, 558–566.

[51] Michael Schader and Sean Luke. 2023. Exploring Planner-Guided Swarms Running on Real Robots. *Submitted for publication* (2023).

[52] Philipp Schoessler, Daniel Windham, Daniel Leithinger, Sean Follmer, and Hiroshi Ishii. 2015. Kinetic Blocks: Actuated Constructive Assembly for Interaction and Display. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 341–349. https://doi.org/10.1145/2807442.2807453

[53] R. Stiefelhagen, C. Fugen, R. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel. 2004. Natural human-robot interaction using speech, head pose and gestures. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3. 2422–2427 vol.3. https://doi.org/10.1109/IROS.2004.1389771

[54] H. Sugie, Y. Inagaki, S. Ono, H. Aisu, and T. Unemi. 1995. Placing objects with multiple mobile robots-mutual help using intention inference. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, Vol. 2. 2181–2186 vol.2. https://doi.org/10.1109/ROBOT.1995.525583

[55] Ryo Suzuki, Hooman Hedayati, Clement Zheng, James L Bohn, Daniel Szafir, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. 2020. Roomshift: Room-scale dynamic haptics for vr with furniture-moving swarm robots. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–11.

[56] Ryo Suzuki, Eyal Ofek, Mike Sinclair, Daniel Leithinger, and Mar Gonzalez-Franco. 2021. HapticBots: Distributed Encountered-Type Haptics for VR with Multiple Shape-Changing Mobile Robots. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 1269–1281. https://doi.org/10.1145/3472749.3474821

[57] Ryo Suzuki, Clement Zheng, Yasuaki Kakehi, Tom Yeh, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. 2019. Shapebots: Shape-changing swarm robots. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 493–505.

[58] Elio Tuci, Muhanad H. M. Alkilabi, and Otar Akanyeti. 2018. Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art. *Frontiers in Robotics and AI* 5 (2018). https://doi.org/10.3389/frobt.2018.00059

[59] Luke Vink, Viirj Kan, Ken Nakagaki, Daniel Leithinger, Sean Follmer, Philipp Schoessler, Amit Zoran, and Hiroshi Ishii. 2015. Transform as adaptive and dynamic furniture. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 183–183.

[60] Keru Wang, Zhu Wang, Karl Rosenberg, Zhenyi He, Dong Woo Yoo, Un Joo Christopher, and Ken Perlin. 2022. Mixed Reality Collaboration for Complementary Working Styles. In *ACM SIGGRAPH 2022 Immersive Pavilion*. 1–2.

[61] Malte Weiss, Florian Schwarz, Simon Jakubowski, and Jan Borchers. 2010. Madgets: Actuating Widgets on Interactive Tabletops. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. Association for Computing Machinery, New York, NY, USA, 293–302. https://doi.org/10.1145/1866029.1866075

[62] Junichi Yamaoka and Yasuaki Kakehi. 2013. dePENd: augmented handwriting system using ferromagnetism of a ballpoint pen. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 203–210.

[63] Yiwei Zhao, Lawrence H Kim, Ye Wang, Mathieu Le Goc, and Sean Follmer. 2017. Robotic assembly of haptic proxy objects for tangible interaction and virtual reality. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 82–91.

[64] Qian Zhou, Sarah Sykes, Sidney Fels, and Kenrick Kin. 2020. Gripmarks: Using hand grips to transform in-hand objects into mixed reality input. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

[65] Bing Zhu, Lihua Xie, Duo Han, Xiangyu Meng, and Rodney Teo. 2017. A survey on recent progress in control of swarm systems. *Science China Information Sciences* 60 (2017), 1–24.