Mimic: In-Situ Recording and Re-Use of Demonstrations to Support Robot Teleoperation

Karthik Mahadevan
University of Toronto
karthikm@dgp.toronto.edu

Yan Chen
University of Toronto
yanchen@dgp.toronto.edu

Maya Cakmak
University of Washington
mcakmak@cs.washington.edu

Anthony Tang
University of Toronto
tonytang@utoronto.ca

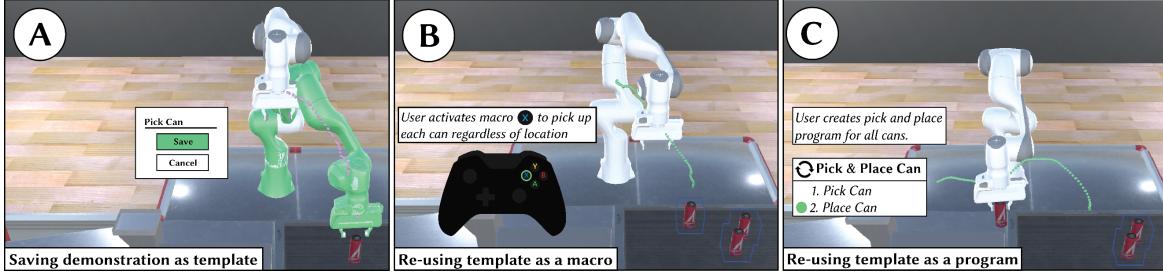Tovi Grossman
University of Toronto
tovi@dgp.toronto.edu

Figure 1: Mimic is a system that allows robot users to record arbitrary robot trajectory demonstrations and (A) save them as templates. Later, they can re-use templates for remotely teleoperating the robot to perform manipulation tasks through: (B) macros, parametrized templates activated by buttons, and (C) programs, sequences of parametrized templates that operate autonomously.

ABSTRACT

Remote teleoperation is an important robot control method when they cannot operate fully autonomously. Yet, teleoperation presents challenges to effective and full robot utilization: controls are cumbersome, inefficient, and the teleoperator needs to actively attend to the robot and its environment. Inspired by end-user programming, we propose a new interaction paradigm to support robot teleoperation for combinations of repetitive and complex movements. We introduce Mimic, a system that allows teleoperators to demonstrate and save robot trajectories as templates, and re-use them to execute the same action in new situations. Templates can be re-used through (1) macros—parametrized templates assigned to and activated by buttons on the controller, and (2) programs—sequences of parametrized templates that operate autonomously. A user study in a simulated environment showed that after initial set up time, participants completed manipulation tasks faster and more easily compared to traditional direct control.

CCS CONCEPTS

• **Human-centered computing** → Human computer interaction (HCI); Interactive systems and tools.

KEYWORDS

Robot teleoperation, end-user robot programming

1 INTRODUCTION

Robot teleoperation is necessary in many cases where robots cannot operate fully autonomously, such as for surgery and manufacturing [12, 36, 58]. In these scenarios, teleoperators need to attend to, and actively make decisions about the robot's movement based on an understanding of its environment gained through sensors (e.g., cameras). Joysticks or controllers [24, 27] are a primary method for commanding robots through combinations of complex and repetitive movements [37, 41]. However, joysticks typically have fewer degrees-of-freedom (DoF) compared to the high degrees-of-freedom that robots use to plan movements [9, 27], among other difficulties [14].

Automating repetitive tasks has the potential to relieve the teleoperator's workload. We are inspired by how collocated end-user

robot programming supports the automation of repetitive tasks through pre-defined movements [22] or recorded demonstrations [3, 29]. However, teleoperation fundamentally differs from collocated scenarios due to its unpredictability [14]. We apply principles from end-user robot programming to improve the teleoperator's experience.

We propose Mimic, a novel system that allows users to *record* and *re-use* demonstrations in the context of teleoperating a tabletop robotic arm with a joystick interface. In the recording phase, users can manually control the robot and provide a single demonstration of an arbitrary trajectory to the robot. The system generalizes the recorded movement, learned using dynamic movement primitives (DMPs) [30, 53], and can then generate trajectories for arbitrary start and end locations. Users can *parametrize* these trajectories (such as on the type of object to pick up), and save them as a *template* for re-use through *macros* or *programs*. Macros let users assign parametrized templates (such as a custom "pick" or a "place") to individual buttons on the joystick for subsequent single-click activation. Programs allow users to combine multiple templates, which can then execute autonomously. Mimic is realized in a simulated environment through the Unity game engine and ROS.

Mimic was evaluated in a remote user study where 12 participants first tried the recording and re-use features of the system, and then completed a grocery bagging task using direct control, macros, and programs. The results show that, despite an initial setup time to create the templates, participants completed the task significantly faster with macros (35%) and programs (69%) compared to direct control, and overwhelmingly preferred the macros and programs. The system also allowed teleoperators to maintain control and fine tune the robot's movements when needed.

This work advances knowledge on designing mechanisms to improve the teleoperation experience. We contribute:

- A proof-of-concept simulated robotic system, Mimic, demonstrating the ability to record and apply arbitrary robot trajectories in real-time.
- An evaluation showing that Mimic improves the user's teleoperation performance and user experience.

2 RELATED WORK

2.1 Combating the Challenges of Teleoperating Robots

Prior work has aimed to mitigate the numerous challenges associated with robot teleoperation (see [14] for a survey). For instance, research has explored increasing the field of view that operators can see such as through perspective folding [63] or stereo cameras [55]. Alternatively, additional stationary [48] or dynamic [50, 61] viewpoints can be provided. Other research has proposed simpler direct control schemes such as object-centric navigation for drones [33] and relaxing constraints between the user and robotic arm [49]. AR visualizations have also been used to support teleoperators, such as controlling a virtual surrogate of the physical robot [64] or streaming what they can see [26].

Despite these improvements, the teleoperator still needs to provide continuous commands to the robot. Researchers have also proposed various shared control schemes to assist teleoperators,

such as by auto-completing their movement if it matches a pattern [66] or shared control by predicting user intent using different approaches [2, 16, 17, 21, 25]. Shared control is usually activated automatically, but our work targets the *deliberate activation* of automation by the user.

2.2 End-User Robot Programming

End-user robot programming (EUP) has emerged as a way to enable users with limited programming experience to program robots (see [3] for a survey), particularly for scenarios where humans and robots are collocated. Here we review work that focuses on manipulation tasks.

Some approaches provide the user with pre-programmed primitives like "pick up" to create programs [20, 28]. Although such high-level primitives greatly simplify the programming task, it is difficult to provide users with robust and generalizable primitives that can function in every possible scenario. Other approaches allow users to demonstrate trajectories and learn a generalizable model of the action from these demonstrations. This is typically achieved by representing trajectories relative to reference frames of objects or the environment [6, 7, 18, 29, 46]. The reference frame can be adapted to a new scenario by the user or autonomously via robot perception [34, 35].

To our knowledge, end-user robot programming has seen limited application to support teleoperation. Senft et al. [57] propose an approach where teleoperators can select from a list of pre-defined movements and choose actions to be applied on individual objects (such as picking up an individual screw). Our work expands on this approach in a number of important ways. In particular, we allow users to record and save trajectories contextualized in the task environment, which can then be generalized to allow the user to specify trajectory characteristics that are unique to the task (e.g., picking up a screw from a specific approach angle due to possible collisions). Our proposed approach also supports the impromptu deployment of templates whereas end-user programming work usually necessitates the exact specification of the sequence of actions (e.g., pick up the first screw and place it in the bin). This is especially valuable for teleoperation scenarios since planning far ahead is not always possible. Finally, our approach augments automation with direct control from the user as needed.

2.3 Learning from Demonstration

Prior work has proposed learning generalizable robot behaviors through demonstration (LfD) [10, 13, 52], typically for automating robot behaviors. Some modern methods take demonstrations represented as keyframes [5, 43], while most methods take continuous time series of joints or end effector poses. There are many ways in which skills are represented within LfD research. Most commonly, behavioral cloning methods learn a mapping between observed states and actions, i.e., a *policy*, without needing to know the demonstrator's latent objectives and using only a few demonstrations. Dynamic movement primitives (DMPs) [30] are an example of a behavioral cloning model that support learning from a single demonstration. Based on a single example, a DMP can generate trajectories that match the characteristics of the original trajectory in a new context. They have become a standard representation of
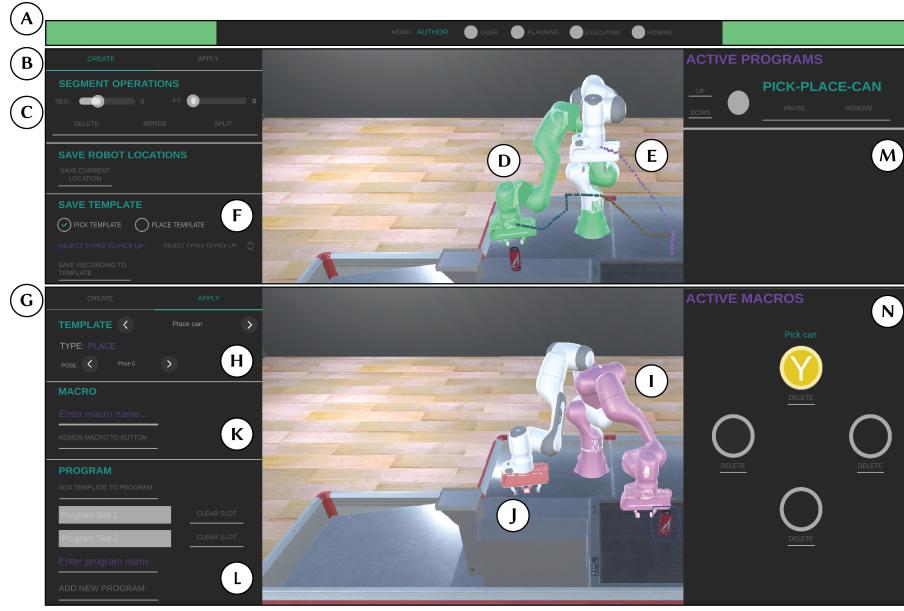
Figure 2: Mimic's user interface in *author mode*. The *awareness widget* (A) shows the state of the interface and the robot. *The create tab* (B) allows the user to see a recorded demonstration, visualized with a robot (D) and Cartesian path (E). The recorded demonstration can be edited using segment operations (C), and *parametrized* and saved as a *template* (F). The *apply tab* (G) allows the user to browse previously saved templates (H) visualized with a robot (I) and its saved drop-off location (J). Re-use can be achieved by creating *macros* (K) and *programs* (L). Active macros (N) can be seen regardless of the mode of operation and currently available programs (M) are visible.

robot movement [44], and successfully applied in many domains [53] such as for assembly tasks [8] and human-robot collaboration [1, 59]. Some researchers have also worked on allowing skills to be maintained or adapted later on, such as through AR-based interfaces [38]. We propose to leverage LfD in a novel context—supporting teleoperators by recording trajectories which are represented using DMPs, to quickly learn from and generate new trajectories.

3 MIMIC SYSTEM

3.1 Overview and Design Goals

We developed Mimic to explore the idea of recording and re-using arbitrary robot trajectories to enable operators to blend autonomous movement with direct control (Figure 1). Mimic is inspired by recent end-user programming interfaces for various human-robot interaction scenarios such as collaboration [54] and co-located end-user robot programming [4, 56]. Mimic was designed with the following design goals in mind, to address some of the constraints discussed in our review of the prior literature:

- **DG1:** Allow the user to record demonstrations in-situ (during task execution) and modify them later. This ensures that the user can quickly capture task-specific movements for immediate re-use.
- **DG2:** Support parametrization of recorded demonstrations for generalization through templates. This is critical to ensure that users can create scenario-specific movements (e.g.,

pick up certain object types using a particular recorded demonstration).
- **DG3:** Support generalizability of demonstrations by producing trajectories with similar movement characteristics despite changes in task parameters (e.g., object types or goal locations).
- **DG4:** Allow the user to easily execute templates during task execution. This ensures that the user can re-use templates to adapt to evolving task requirements.
- **DG5:** Give the user the ability to quickly "step in" through direct control as necessary. This ensures that the user has flexibility in switching between manual and autonomous behavior depending on task demands.

3.2 Implementation Environment

The system is prototyped using the Unity game engine and ROS on the Ubuntu operating system to present a simulated virtual robot which the user controls. A game controller is used to control the robot, and a keyboard and mouse is used for interface operations. The implementation assumes teleoperation scenarios where multiple cameras provide access to diverse viewpoints (e.g., [50]). It also assumes the ability to augment a camera stream with spatial AR visualizations such as waypoints and user interface elements (e.g., [60]). We elected to utilize a virtual simulation of a robot as our focus is on the teleoperator interface, and not the implementation issues of a physical robot. Considerations of a physical implementation are discussed later in the paper.

3.3 Interactions

3.3.1 Modes of Operation. Mimic supports two operation modes: *control* and *author. Control mode* allows the user to teleoperate the robot using a game controller (e.g., Xbox or PlayStation) representing Cartesian end effector velocities in 6DoF. The user can open and close the robot's gripper or reset its joints to the neutral state. They can switch viewpoints between looking from behind and looking from the front of the robot. A secondary, first-person camera attached to the robot's gripper is also available. In *author mode*, the user can record new templates through demonstration or apply existing templates through macros and programs. Figure 2 illustrates the user interface.

3.3.2 Recording Demonstrations. Mimic allows recording demonstrations *automatically* or *manually* **(DG1)**. When automatic recording is on, continuous robot joint states are captured for up to one minute (a system parameter). Manual recordings begin with a button press. The recording stops when the user presses the record button again or switches to *author mode*.

3.3.3 Editing and Saving Trajectories as Templates. In *author mode*, the user can preview a recorded trajectory through the *create tab* (Figure 2B). Mimic visualizes the trajectory as a list of Cartesian poses which form a *Cartesian path* (Figure 2E). The poses are then segmented to meaningful components (such as a "pick" or "place") through unsupervised path segmentation [32] and visualized with different colors (Figure 2E). A *visualization robot* (Figure 2D) also appears in-situ and replays the recorded trajectory in an animated loop.

The user can inspect and edit the Cartesian path (Figure 2C) using several interactions **(DG1)**. These features provide fine-grained control over the recording, such as trimming unwanted portions (e.g., failing to pick up an object in the first attempt). Editing a segment modifies the underlying joint positions comprising the trajectory. When active, a segment can be *deleted, merged*, or *split*.

Once the user is satisfied with any needed edits, they can save the remaining trajectory as a *template* (Figure 2F) by parametrizing it using *movement patterns* **(DG2)**. Mimic currently supports two types of patterns, *pick* and *place*, as they represent a large class of movement types today's robots can perform. Before saving the trajectory as a template, the user can select the pattern that best represents it.

For *pick templates*, Mimic supports specifying one or more *object types*. This means that a template can be created to pick up specific object types if desired. In Mimic, grasping happens from the top—though there are approaches to dynamically generate grasp poses [39]. The robot closes its gripper after executing a pick template.

For *place templates*, the user can specify a drop-off location for picked up objects. The user can save a new location using the gripper's current location. The robot will open its gripper after executing a place template.

After parameterization, a template can be saved to store trajectory information and its parameters—this triggers the learning of the trajectory's characteristics for re-use **(DG3)**.

Retrieving and Re-Using Templates

The user can re-use a saved template in three ways: as a single instance, a *macro*, and a *program* **(DG4)**. In all cases, the first step involves accessing *author mode* and selecting the *apply tab* of the UI (Figure 2G) which displays previously saved templates. The stored parameters of a template are visible when active, but can be overridden. For example, a place template with a saved drop-off location (Figure 2J) can be swapped to a new location. When a saved template is browsed, the robot's environment is augmented with a *visualization robot* that replays the trajectory (Figure 2I). Depending on the type of re-use, the next step varies:

Macro: When the user recognizes that a task they are doing requires repetitive movements, they can create a *macro* rather than execute the template just once (Figure 2K). For instance, a *pick* template can be assigned as a macro to work for all cylindrical objects. If the conditions required to activate a macro are true, the robot plans and executes it. After selecting a template and possibly overriding its parameters, the user can name the macro and assign it to a controller button (Figure 2N). To activate a macro, the user presses the corresponding controller button which prompts the robot to begin checking if it can be used. When active, the macro slot is highlighted (Figure 2N). An active macro can be halted at any time by pressing the same button.

Program: If the user wants the robot to act autonomously, a *program*, composed of one or more templates (Figure 2L), can be used. Saved templates can be added to fill *program slots*. Currently, two slots can be filled in a single program to represent pick and place scenarios but this can be extended to integrate more complex movements (e.g., picking up an object, performing an action, and placing it somewhere).

Creating a program populates it in the list of available programs (Figure 2M). A program runs when it is active—i.e., when the conditions of the first template comprising the program is met. An active program is visualized by a green icon. Running a program means planning and executing trajectories for each template comprising it. The user can pause a program at any point using the *pause* button (Figure 2M). When multiple programs are available the user can re-order them to change the priority in which they are executed.

3.3.4 Overriding the Robot's Movement. The user can seamlessly override the robot's current movement by providing an input command (e.g., flicking the controller thumbstick), to transition back to direct control **(DG5)**. This can be useful if the environment has changed, or an autonomous movement causes an unexpected behavior. User input immediately halts an active macro whereas the robot resumes checking for programs when there is no active input for more than one second (a system parameter).

3.3.5 Guided Trajectory Execution. In certain scenarios, autonomous execution of the generated movement is not desirable (e.g., when there are items nearby the robot could collide with). Instead, the user can *step through* individual points of the generated trajectory by pressing a controller button **(DG5)**. This lets them maintain fine control over execution without needing to manually maneuver the robot.

3.3.6 Awareness Features. Mimic informs the user about the system and robot state. An *awareness widget* (Figure 2A) indicates the current mode (green denotes *author mode* and red denotes *control mode*). The widget lets the user know when (1) they are controlling the robot, (2) the robot is moving to the neutral position, (3) the
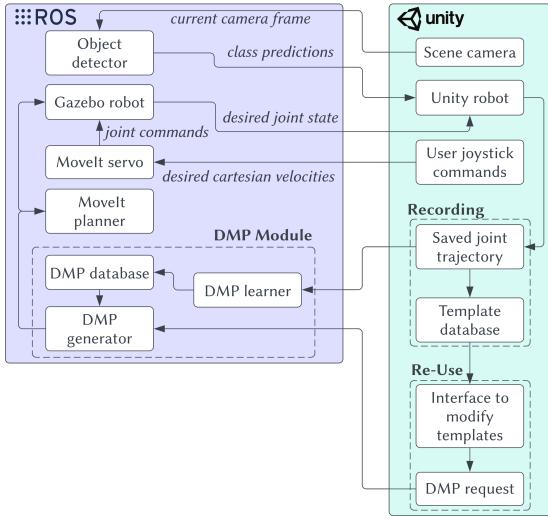
Figure 3: Mimic's system architecture—showing ROS-specific components (left) and Unity-specific components (right).

robot is planning a trajectory, and (4) the robot is executing a macro or program. An in-situ *trajectory trail* appears each time the robot plans a trajectory (green if successful and white otherwise) until the execution finishes.

3.4 Implementation

Mimic is prototyped using Unity and ROS (Figure 3). The physics of the robot are simulated in Unity (as articulation bodies) and Gazebo [67]. MoveIt [68] supports the robot's real-time movement through the MoveIt Servo package and motion planning. The Franka Panda 7DoF robot arm is used in the simulation but Mimic is robot-agnostic. Unity's proprietary solution is used to communicate between Unity and ROS through messages and services. The *Unity robot* subscribes to joint state changes on the *Gazebo robot* to keep the two robots synchronized.

3.4.1 Teleoperation Architecture. In *control mode*, the user's raw controller commands are received in Unity, converted to velocity commands, and sent to MoveIt Servo. They are later converted to joint commands and sent to the Gazebo robot. MoveIt Servo provides many parameters that can be tuned to achieve real-time performance. We used parameters from a sample configuration file for the Franka Panda robot [69].

3.4.2 Segmenting and Visualizing Joint Trajectories. When the user wishes to edit and save a recorded trajectory, Mimic converts it to a 6DoF Cartesian path of poses through forward kinematics using a ROS node. To simplify creating demonstrations, the path is automatically segmented through a custom implementation of an unsupervised segmentation algorithm [32]. Using Gaussian Mixture Models after reducing the 7-dimensional joint trajectory with Principal Component Analysis, the algorithm predicts the number of segments that could belong to the trajectory. The Cartesian path and segments are later sent back to Unity and visualized.

3.4.3 Learning and Applying Templates Using DMPs. Saving a template after parametrizing it involves storing the joint trajectory, Cartesian path, and meta-information (such as what object types to pick up for a pick template) into a JSON file. Once saved, the Cartesian path is used by the DMP learning module on the ROS side. DMPs work by modelling a trajectory (or path) as a second-order non-linear dynamical system akin to a spring-damper model [30, 44]:

$$\tau^2 \ddot{x} = \propto_x (\beta_x (g - x) - \tau \dot{x}) + f$$

Here, x refers to the state of the system, f is the forcing function manipulated to achieve non-linear behavior, $\propto_x$ and β_x are constants expressing the damping and spring behavior, τ is a constant that modulates temporal behavior, and g represents the desired goal state. The forcing function influences the system's behavior from its current state to the goal state and is composed of: (1) a phase variable z that exponentially decays to zero from an arbitrary initial state and (2) Gaussian basis functions $\psi_i(z)$. The forcing term is:

$$f(z) = (g - x_0) \sum_{i=1}^{M} \psi_i(z) w_i z$$

After choosing the appropriate number of Gaussian functions, their widths, and centers, the weight vector w (for the basis functions) can be learned to minimize the loss between the robot's imitated trajectory and the user demonstrated trajectory. This is a typical supervised learning problem whose solution can be estimated using regression. Later, changing the initial state x_0 and goal state g produces new trajectories. This is repeated for each degree of freedom. An inherent limitation of DMPs for Cartesian space learning is singularities associated with Euler angles representing rotations, which are not independent of each other (unlike position coordinates). This means that the robot may orient its gripper in undesired angles while moving from the start to the goal state. Hence, we use Cartesian DMPs [62] which handle the positions and quaternions of the trajectory separately to generate a stable trajectory (or path).

Cartesian DMPs are implemented using an open source Python library [70]. When saving a template on the ROS side, the trajectory is sent to the DMP learning module which imitates the demonstrated trajectory and saves the learned weights (and hyperparameters of the DMP) to a JSON file. When applying a template as an instance, macro, or program, Unity sends a DMP generation request with the name of the associated template and the desired start and end goal states as 6DoF Cartesian poses to the DMP generator module. This results in the generation of a Cartesian path representing the desired movement. However, since DMPs are robot-agnostic, the path needs to be converted into a feasible robot-executable joint trajectory.

3.4.4 Motion Planning of the Robot's Movement. When applying templates, the output of the DMP generation module, a Cartesian path, is sent to MoveIt to plan and execute robot trajectories. This path is converted into a robot-executable joint trajectory using inverse kinematics. However, the default planner can generate trajectories that the robot may not be able to execute due to velocity and acceleration limit violations. By retiming the trajectory [51], a trajectory respecting the robot's velocity and acceleration bounds is generated. The trajectory is then executed on the Gazebo robot and

mirrored on the Unity robot. We assume that the robot's trajectory can be executed at velocities and accelerations up to the robot's maximum joint limits.

3.4.5 Active Perception of the Robot's Environment. For the pick movement pattern, the robot makes decisions based on the type of object. To support active perception of the robot's environment, a pre-trained YOLOv5 object detection model [31] was fine-tuned by creating a synthetic image dataset using virtual grocery models [71] and Unity's perception package [72] to recognize 8 classes of items. For detection, a virtual camera next to the robot streams images of the scene every two seconds (a system parameter) to a ROS node that sends it to the object detector. The model returns a list of bounding boxes and class label predictions which are sent back to Unity. We assume access to the 6DoF poses of all scene objects for motion planning, but one could estimate 6DoF poses of objects when such data is not available [47].

3.4.6 Macros and Programs. When the user activates a macro with a button, the *macro manager* handles its execution by first checking if any macros are active. If not, it activates the corresponding macro and looks for conditions that must be met to execute it. For pick macros, the manager tries to find object types matching those specified in the template (e.g., a pick macro for rectangular objects only). If found, the manager requests a plan from the DMP generator. For place templates, the conditions are met if the gripper's current pose is available. To execute, a motion plan with the current pose as the start and a drop-off location as the end is generated. If a planning request fails after three attempts (a system parameter), the macro is deactivated.

Each program instance has its own checking functionality to see if it can become active. Programs meeting the prerequisite conditions are *available* to be used. The *program manager* checks all available programs and activates them in the order in which they were created, assuming no other program is running. Once active, the program will aim to plan and execute all program slots (containing templates) in sequence. If a planning request fails after three attempts (a system parameter), the program becomes inactive and the manager looks for new programs to execute.

For pick templates executed as macros and programs, Mimic selects the object *closest* to the robot's gripper whose object type matches the template. User intervention through commands issued on the controller terminate the current macro instance and reset the active program.

4 USER STUDY

We conducted a controlled remote user study to understand whether users can quickly learn to use Mimic, whether they can efficiently use it to complete manipulation tasks, and how the use of macros and programs compare to the direct control baseline condition. In particular, we were interested to know if the cost of setting up macros and programs outweighed the speed gains from automating repetitive movements. The study task required utilizing the core features of Mimic (e.g., recording templates and re-using them as macros and programs), but some advanced features (e.g., editing segments or guided execution) were omitted.
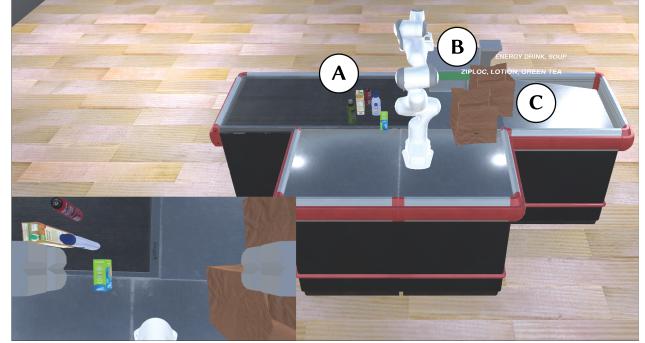

Figure 4: The evaluation task environment showing (A) grocery items to be manipulated, (B) where items needed to be scanned, and (C) bags where items needed to be placed.

4.1 Tasks

The study was divided into two sections. The first section was used as a feature exploration—to get participants accustomed to manual controls for the robot, and to use Mimic and gain insight into its features. The evaluation section was used to formally compare three control conditions: direct control, macros, and programs. For the evaluation, pre-made templates were provided as the goal was to assess the cost savings of using templates for teleoperation. Since each user demonstrates large variance in creating the same template (e.g., pick) which is tied to their ability to manually operate the robot, we instead evaluated the user experience of creating templates in the feature exploration section. All tasks were grounded in a grocery bagging robot scenario as there have been recent attempts to use robots in this context [73].

Feature Exploration: The feature exploration section was split into four tasks. The *manipulation task* let participants become accustomed to picking and placing objects using the controller, requiring them to place four grocery items into a bag. In the *recording task*, participants recorded templates for picking and placing one grocery item into a bag. For the *macro task* and *program task*, participants were provided pre-made templates and used them to create macros and programs respectively. The tasks involved placing four items (two of each type) into two bags (one for each type). The pre-made templates included one pick template and two place templates (one for each bag).

Evaluation: The evaluation section (Figure 4) served to compare three conditions (direct control, macros, and programs) and involved picking up a *set* of five grocery items from the check-out counter, scanning them, and placing them in one of two bags (each with its own item types). The task was divided into three phases: Phase 1 – *setup bagging task*, Phase 2 – *repeat bagging task*, and Phase 3 – *bagging with user intervention task*. The task was divided to separate the analysis of the different functionality that Mimic offers – the ability to create macros and programs, the ability to use the macros and programs (once created), and the ability to interleave autonomous execution with user input.

In Phase 1, participants were provided one template for picking up grocery items and one template each for scanning and placing grocery items into a grocery bag, generated with a single demonstration via Mimic's recording capabilities. Participants used the

controls for the associated condition to clear the first set of items by scanning and placing them in the correct bag. Once these were cleared, Phase 2 began with a new set of the same type and number of items appearing in a semi-randomized fashion. Phase 3 began when the next set of items were cleared. In this phase, the location of each grocery bag changed. In the macro and program conditions, manual intervention was required at the end to ensure items were placed in the correct bags.

For item randomization, each set contained the same number and type of items (five items with one of each type) placed in specific areas on the table depending on the set. Within each set, objects were shuffled between five locations (120 combinations) and displaced by +/- 0.01 m. A balanced Latin Square was used to counterbalance the condition ordering.

Conditions: Three conditions represented the method participants used to complete the task. The *direct control* condition represented a baseline, where the game controller was used to manually control the robot. In the *macro* condition, participants had to create and use macros to complete the task. In the *program* condition, participants had to create and use programs to complete the task.

Task Progress Awareness: Visual feedback provided participants awareness of task progress. In all tasks, bags were labelled with the types of items that could be placed inside. Correctly placed items were outlined in green. For the evaluation, scanning a grocery item outlined it in yellow and caused the scanner to briefly change color from green to red.

Task Characteristics: In both the feature exploration and evaluation, tasks were designed so that the participant never had to rotate the robot's gripper; they only performed translations in world coordinates. This ensured that the tasks were not too difficult to complete with direct control as rotations can be particularly difficult [14].

4.2 Robot Behavior & Assumptions

Our simulation supports physics for the robot (in Gazebo and Unity) and grocery items (in Unity). However, in pilot studies, we noticed that objects were sometimes grasped by the robot and dropped due to the stochasticity of physics parameters such as friction. Hence, to maintain consistency across trials, objects grasped by the robot stayed grasped until the gripper opened to release them. In the *evaluation task*, the robot did not attempt to pick previously scanned and bagged items in the *macro* and *program* conditions.

4.3 Participants

Twelve participants (7 male, 5 female) with a mean age of 27.9 (std 5.1) were recruited through research networks. All participants had experience using game controllers: two used their controllers daily, three used them once a week, four used them two to six times a week, and three did not use them on a weekly basis. One participant had previously teleoperated a mobile robot as part of a robotics competition.

4.4 Measures

In-application data was collected throughout the study, including task completion time and time spent on different aspects of the task. A post-study questionnaire and short interview solicited information about participants' experience with the various features of Mimic and their overall feedback. The screen streaming the application was recorded throughout the study.

4.5 Procedure

Participants completed the study remotely through a Zoom call in their homes using their own game controllers. Participants took control of the experimenter's PC to run the application using the Moonlight streaming application [74]. Participants provided consent prior to the study and completed a demographics survey. Participants were given four tutorial videos explaining different system components which they watched before each relevant task.

Prior to the *teleoperation* task, participants familiarized themselves with controlling the robot using a controller. Then they stepped through each task as described and only stopped to watch tutorial videos associated with the specific feature task. During the feature exploration section, the experimenter intervened to provide additional guidance if the participant struggled with a specific step.

For the evaluation, the experimenter reset an object's position if the participant toppled it over (usually during manual control). Participants repeated the evaluation task in each condition one at a time. In the *macro* condition, participants were given the suggestion to either create individual macros for items belonging to each bag or create a single macro for all items. Similarly, in the program condition, participants were given the suggestion to create two programs, one for each bag. In Phase 3 of the evaluation for the *macro* and *program* conditions, participants were informed that they needed to takeover at the end of the place movement as the prepared macros and programs did not recognize that the bags had changed locations.

5 FINDINGS

The findings focus on the qualitative aspects of participants' experience using Mimic as well as the performance of the underlying techniques, macros and programs, in comparison to direct control. One participant's data was omitted from quantitative analysis as an outlier, as their completion time in the *direct control* condition for Phase 2 and Phase 3 was greater than 2.5 times the standard deviation. Their qualitative reflections were still captured and analyzed.

5.1 Evaluating Task - Performance

Completion time was analyzed through one-way repeated measures ANOVA tests with Greenhouse-Geisser corrections for sphericity. Given that each phase of the task assessed different functionality, the analysis was performed separately per phase. Figure 5 shows the completion time of participants in each phase of the task over each condition. Figure 6 provides a detailed breakdown of the time spent—including operating the robot, robot idle time, authoring macros and programs in the respective conditions, and the time spent by the robot acting autonomously.

Phase 1 – Setup Bagging Task: There was a significant effect of the technique used on completion time ($F_{(2, 20)} = 8.74$, $p < .005$). Pairwise tests revealed that the *macro* condition resulted in a significantly slower completion time than the *direct control* and *program* conditions. Figure 6B shows that authoring time added
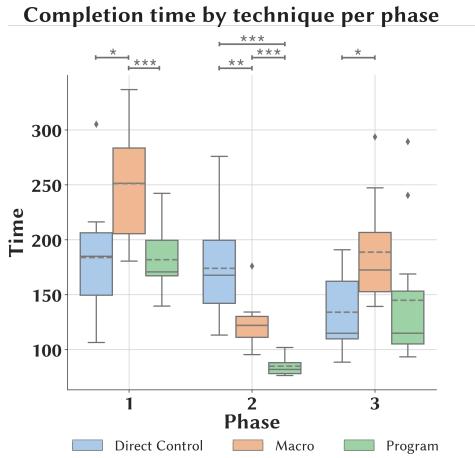
Completion time by technique per phase

Figure 5: Evaluation task completion times by condition for each phase. Dashed lines represent the mean and the solid lines represent the median. Asterisks indicate pairwise comparisons with significance differences (p < .05*, p < .01**, p < .001***).
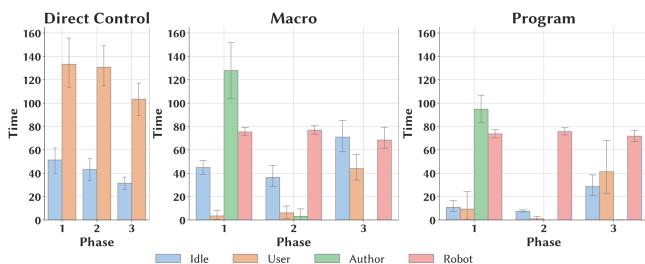

Figure 6: Breakdown of time spent by participants in each technique condition per phase of the evaluation task: (1) Robot idling (waiting for user input). (2) Participant operating the robot. (3) Participant authoring macros and programs. (4) Robot planning and executing trajectories.
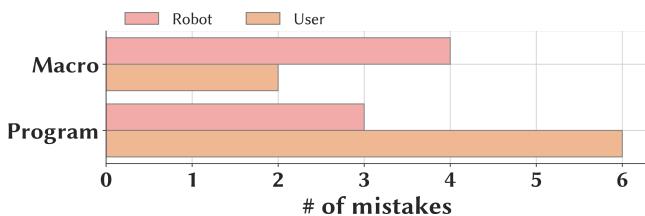

Figure 7: Number of mistakes made by the robot and the user when using macros and programs.

a substantial amount of time to completion time. This varied as some participants opted to create one macro for all object types whereas others found it easier (from a mental model standpoint) to create individual macros per bag. There were also instances where participants (P2, P12) created macros but re-created them right before starting as they felt it made more sense to use a single macro to pick up all the items. This was not the case for programs where

the obvious solution was to create one program per bag. There was no significant difference in completion time between *direct control* and *program* (Figure 5).

Phase 2 – Repeat Bagging Task: There was a significant effect of technique on completion time (F(2, 20) = 23.87, p < .001). Pairwise comparisons showed that the *program* condition resulted in significantly faster completion time than the *macro* and *direct control* conditions, whereas the *macro* condition resulted in significantly faster completion time than *direct control*. As participants had already created the macros and programs needed to complete the task in Phase 1, the advantage of using them (when little or no failures occur) became clear. Although both macros and programs were significantly faster, using macros resulted in more idle time than programs (Figure 6 middle and right). With macros, we observed that participants needed to (1) map the next action they wanted the robot to perform with the macros they created and (2) recognize when the robot could be issued new commands.

Phase 3 – Bagging with User Intervention Task: There was a significant effect of technique on completion time (F(2, 20) = 3.91, p < .05). Pairwise comparisons showed that *direct control* resulted in significantly faster completion time than the *macro* condition. In this phase, participants had to intervene at the end of the scan and place routine of the robot in the macro and program conditions—hence, user time (Figure 6) increased. Of note is the average time spent controlling the robot in the *macro* and *program* conditions which was nearly equal (Figure 6) but the variance for programs was higher. This may be explained by the time-sensitive nature of the takeover interaction. A majority of participants (6/11) were able to take over and place objects in the correct bag in the *macro* and *program* conditions while the remaining (5/11) had trouble (Figure 7). When using macros, robot errors were more common than user errors—such as when the user did not take over before the robot incorrectly placed items in the wrong bag (or outside the bag).

In contrast, when using programs, user errors were more common—such as taking over and placing the item in the wrong bag (Figure 7). This aligns with expectations as programs operate more autonomously so the user may have been less prepared or forgot the task context after takeover. In addition, since the robot did not pick up or scan a previously placed item (since it was already in the bag), failing to take over meant that participants had to correct the mistake through direct control. However, as programs resumed automatically, the robot spent less time idling than when macros were used and tried to remain productive.

5.2 User Experience

We asked participants to rate their experience using Mimic. Figure 8 summarizes participants' Likert responses about the different components of Mimic while Figure 9 summarizes participants' Likert responses comparing the techniques (direct control, macros, and programs). Overall, each of Mimic's features received generally positive responses (Figure 8). When comparing conditions to *direct control*, macros and programs received better ratings for both Ease of Use and Overall Experience (Figure 9).

5.2.1 Templates. Most participants felt that templates required low effort to create and found the UI intuitive to use (Figure 8). However, many participants felt overwhelmed when creating their
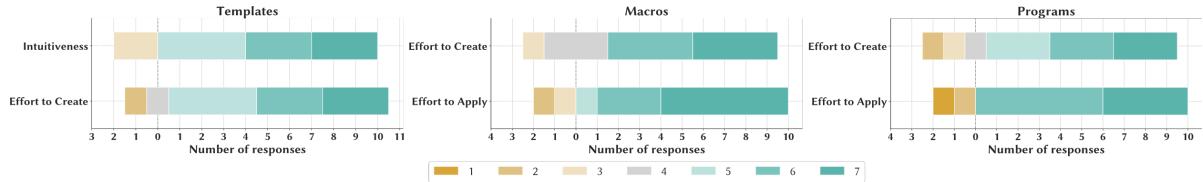
Figure 8: Participants' responses to Likert questions regarding creating templates, macros, and programs respectively. For effort questions, the scale is 1: very high and 7: very low, and for other questions, the scale is 1: strongly disagree to 7: strongly agree.
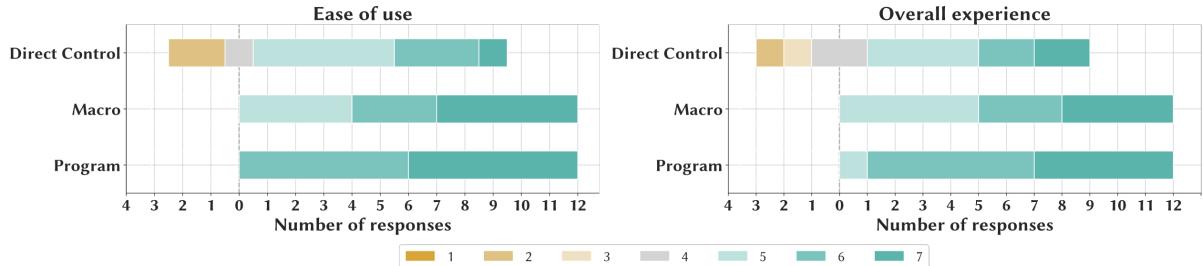

Figure 9: Participants' responses to Likert questions comparing each of the techniques in the user study for (1) ease of use and (2) overall experience. Here, the scale is 1: strongly disagree to 7: strongly agree for questions about ease, multitasking, and movement, while the scale for overall experience is 1: very poor to 7: excellent.

first template. For their first pick template, most participants (8/12) missed steps including: forgetting to start or stop, or overriding a recording, forgetting to select a parameter such as the object type(s) for pick templates, and forgetting to save the template. Most participants (10/12) did not miss steps of the template creation workflow the second time (*start recording > demonstrate > stop recording > switch to author mode*). However, most participants (7/12) forgot to save a drop-off location before assigning it, possibly since it was their first attempt creating a place template. Some participants (3/12) stated that creating templates became easier after creating the first template: *"If you do it once, I find that it is intuitive after that."* (P8).

Almost all participants (11/12) found creating templates or custom movements useful. They appreciated the flexibility of being able to create scenario-dependent templates: *"As long as there is some repetition in the task, then sure there is definitely value in there."* (P10). However, P4 stated that it was easier to use pre-recorded templates rather than creating new ones as manually operating the robot was difficult.

5.2.2 Macros. Overall, most participants rated macros as requiring low effort to create and apply (Figure 8). A majority of participants (7/12) did not have difficulty completing the *macro task* which involved creating macros to pick and place items into a grocery bag from the provided templates. Some participants (5/12) missed minor steps, for example, selecting the wrong menu (hitting the *create* tab instead of the *apply* tab), forgetting to name a macro, or forgetting to assign it to a button. After setting up macros, some participants (4/12) occasionally activated the wrong macro (e.g., applying a place macro without an item in the robot's gripper). All participants felt that macros were more useful than manual control, but found them less useful than programs. Some participants (4/12)

remarked that macros gave them more control over task execution. One participant liked that macros could be built and utilized to support the user's mental model of the task sequence: *"The fact that I could have different macros that fit the mental model of how I would do the task was useful."* (P10).

Most participants (9/12) noted the possibility to mistake the mapping between the button pressed and the macro it activated. This could partly be explained by the default behavior of macros being assigned to the first available slot so participants had to be deliberate in assigning each macro. For instance, P4 stated that at one point, their macros for the right and left bags in the *evaluation task* were flipped.

5.2.3 Programs. Participants felt that creating and applying programs required low effort (Figure 8): *"Because the templates were provided, it was extremely easy by comparison. I could really quickly get out of the menu and watch it go in motion."* (P6). Half the participants (6/12) did forget minor steps while creating programs, including forgetting to fill the second program slot, attempting to add the same template to multiple slots, or selecting the wrong menu. All participants felt the idea of programs was useful and felt that programs would save them from the workload and difficulty of manual control: *"The programs were great for taking care of everything."* (P11). However, it was noted that in noisy and unpredictable environments, programs may require more intervention: *"The only issue with programs is that it does involve a little bit of thinking ahead of time to plan."* (P11).

5.2.4 Ability to Override. Most participants (11/12) found it useful to override macros and programs at any time. Some participants (3/12) noted that the transition of control was unclear, particularly after they had finished the takeover and the robot did not immediately take back control. One user felt that macros were best suited

to overriding: *"You can do overriding in tandem with macros because you are already pushing buttons anyway and you're layering input on top."* (P6).

5.2.5 Awareness. Participants primarily maintained awareness of the control mode using the *trajectory trail*. P6 mentioned that knowing who was in control was easy since they had to stop touching the controller in order for the robot to take back control. P11 stated that they observed the stream of the camera mounted to the gripper to predict when things were about to happen. However, they were generally positive about awareness indications that let them know when a macro or program was active.

5.2.6 Feeling of Collaboration. When asked if they felt in collaboration with the robot, most participants (8/12) agreed to varying extents. For instance, P10 assumed that the robot was an extension of their arm and used it as a tool whereas P11 described the interaction as a dialogue between them and the robot. A few participants (2/12) stated the collaboration was evident regardless of condition, while some (3/12) believed they were collaborating when activating macros and taking control for macros and programs, and a few (3/12) only felt collaboration when taking over for macros and programs.

6 DISCUSSION

Overall, the results indicate that recording and re-using templates in-situ is feasible and effective for real-time teleoperation. However, if users failed to take over when needed, the robot made errors resulting in time delays.

6.1 Intertwining Direct Control and Automation

With direct control, users found themselves constantly attending to the robot's environment to make decisions about its movement. Macros and programs made their job substantially easier; they only needed to pay attention in between template execution (for macros) and moments during which intervention was needed (for programs). Still, participants had a strong preference for programs (10/12 agreed) but their explanations were more nuanced. Though programs performed best once set up, they necessitated time-sensitive takeover because the templates were executed immediately in sequence, so the cost of mistakes was higher. Even when the user successfully takes over, user mistakes could occur (Figure 7). There is also an additional cost with programs: a mistake during template execution could affect subsequent templates. Without additional sensing, the robot would not be able to recognize a mistake, and would continue. In these cases, the user would need to grab control.

We think macros may be more suitable than programs as they abstract the complexity of fine-grained direct control but preserve the ability to override without the fear of cascading errors. P5 said, *"I felt that macros offered the right level of abstraction".* In environments where the robot's missteps do not substantially affect task success, programs could be valuable, as beyond setting up and occasionally providing fine control, the user can be a *"silent observer that manipulates it (robot) if and when needed."* (P1). There is also the option to use macros as fallbacks to programs when the task is straightforward but has some stochasticity. In a grocery store

scenario, a macro to press a physical button to advance the conveyor belt for out-of-reach items is useful. Macros or programs can also be interleaved with direct control—such as to fill a box with items according to the user's requirements— here automation minimizes user time and effort but lets the user dictate exactly where to place items.

6.2 Improving Robot Takeover Models

Takeover is time-sensitive and costly when a mistake occasionally happens, so other interaction models could be pursued. In Mimic, takeover results in immediate stoppage of a macro and program but programs resume after a fixed period (and can also be manually paused). Providing finer control over autonomous execution is promising. For example, if the user fails to takeover and the robot drops an item, the user could "rewind" the program's state so that it re-attempts the place template after the user picks up the item instead of cancelling the program. Or, the system could provide additional awareness cues of the robot's next actions (beyond the *trajectory trail*) to ensure takeover happens, based on prior work on handover for semi-autonomous driving and state awareness for mobile robots (e.g., [11, 42]). For instance, providing a visual timer to let the teleoperator know when an autonomous movement will finish could better prime them for takeover. In addition to the teleoperator, providing awareness cues to people sharing the physical space and collaborating with the robot could be beneficial.

6.3 Limitations and Future Work

Despite promising results, there are some limitations. Mimic and the study results are grounded in a simulation that does not capture the unpredictability of the real world (although we included a user intervention evaluation task). Some assumptions underpin Mimic such as accurate object pose detection, the lack of a need for collision detection (as all pick up templates grasp from the top), and the ability to operate the robot at its maximum capabilities. In the study, grasped objects were not dropped again but this could happen in reality. Hence, a physical implementation is needed to confirm the benefits observed in the study. However, we would expect Mimic to outperform manual teleoperation since our approach records the robot's movements in the real-world. Hence, the generated movements would already account for real-world considerations such as sensor noise in recording the joint angles. In addition, Mimic lets users intervene to make fine adjustments to the generated movement when there is no perfect model of the teleoperation environment. In contrast, manual teleoperation necessitates a higher user workload and lower efficiency through continuous user input.

The evaluation tasks were short (two to three minutes per phase). Despite that, macros and programs performed very well. Longer and more complex tasks involving several objects or robots will likely demonstrate their benefits further especially if the robot's gripper needs to be manually rotated (unlike in the study tasks). Template creation was not included in the evaluation task as the goal was to assess the cost savings once templates were set up. However, it would be interesting to compare task performance when users can create their own templates.

The current behavioral cloning model (DMPs) is powerful and worked well. However, as it only uses a single demonstration to

learn and generate movements, poor demonstrations could lead to worse generated movements. DMPs have been extended to let users specify via-points, or points that generated trajectories must traverse [53] to reduce the reliance on good demonstrations. Further, DMPs occasionally produce out-of-reach or collision-laden trajectories. This can be partly resolved by automatically switching to standard motion planning approaches to plan an arbitrary point-to-point movement without collisions as a fallback option. Newer DMP formulations can also account for collisions and generate modified trajectories even if they were not part of the initial demonstrations [23]. Alternatively, models other than DMPs could be explored, such as ProMPs [45] which increase the user's burden at recording time but can produce better trajectories despite the variance in user demonstration quality. More novel approaches such as one-shot visual imitation [19] and transformers [15] are also viable, though they require a priori knowledge of the tasks to be completed.

Mimic currently allows to create pick and place movement patterns but any arbitrary patterns such as cyclical movements (e.g., shaking a can) can also be learned. At present, macro slots are limited by available buttons on the control interface but more macros could be accessed through button combinations or double tapping. Macros could also be chained in advance instead of individually activated. For programs, Mimic could support conditionals (e.g., if/else statements) to express logic. It could also be expanded to further support ease of re-use, such as automatically triggering macros or programs based on the user's task [65, 66] or suggesting sequences of templates to the user [40].

Mimic allows specifying a few parameters at present—but expanding this range to other sensors (e.g., touch or weight) or properties beyond object detection (e.g., user annotation of the camera feed or detecting object relationships) could make it more powerful. We are also interested in scenarios where the teleoperator and robot work with a collocated person (e.g., for remote healthcare). This requires accounting for human preferences and dynamics through more advanced parametrization (e.g., human trajectory prediction in the shared space).

Mimic's current interface is usable but could be improved. For example, the user needs to utilize a joystick for teleoperation and a keyboard and mouse for authoring. A unified approach on a single input device could reduce the user's workload. Some features such as editing individual keyframes of a long trajectory can be cumbersome with a mouse. More novel configurations such as VR present new opportunities to improve features such as trajectory (keyframe) editing or browsing the library of recorded templates, which could be made easier through spatial interactions as opposed to screen-based interactions. Lastly, devices such as VR headsets and controllers could enhance teleoperator immersion and result in better demonstrations to learn from.

7 CONCLUSION

Today's teleoperation interfaces typically require users to manually command robots to achieve combinations of complex and repetitive movements which is cumbersome. We present Mimic, which demonstrates the idea of supporting teleoperators by allowing them

to record and re-use arbitrary trajectories through macros and programs. The evaluation suggests that the techniques helped users to complete manipulation tasks more efficiently and easily after an initial setup time. We hope that Mimic inspires future work into building automated mechanisms to better assist teleoperators in completing real-time manipulation tasks.

ACKNOWLEDGMENTS

REFERENCES

[1] Ahmed Faisal Abdelrahman, Alex Mitrevski, and Paul G. Plöger. 2020. Context-Aware Task Execution Using Apprenticeship Learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 1329–1335. DOI:https://doi.org/10.1109/ICRA40945.2020.9197476

[2] Henny Admoni and Siddhartha Srinivasa. 2016. Predicting User Intent Through Eye Gaze for Shared Autonomy. In *AAAI Fall Symposia*.

[3] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. 2022. A Survey on End-User Robot Programming. *ACM Comput. Surv.* 54, 8 (November 2022), 1–36. DOI:https://doi.org/10.1145/3466819

[4] Gopika Ajaykumar, Maia Stiber, and Chien-Ming Huang. 2021. Designing user-centric programming aids for kinesthetic teaching of collaborative robots. *Robotics and Autonomous Systems* 145, (November 2021), 103845. DOI:https://doi.org/10.1016/j.robot.2021.103845

[5] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L. Thomaz. 2012. Keyframe-based Learning from Demonstration. *Int J of Soc Robotics* 4, 4 (November 2012), 343–355. DOI:https://doi.org/10.1007/s12369-012-0160-0

[6] Sonya Alexandrova, Maya Cakmak, Kaijen Hsiao, and Leila Takayama. 2014. Robot Programming by Demonstration with Interactive Action Visualizations. In *Robotics: Science and Systems*. DOI:https://doi.org/10.15607/RSS.2014.X.048

[7] Sonya Alexandrova, Zachary Tatlock, and Maya Cakmak. 2015. RoboFlow: A flow-based visual programming language for mobile manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Seattle, WA, USA, 5537–5544. DOI:https://doi.org/10.1109/ICRA.2015.7139973

[8] Daniel Angelov, Yordan Hristov, Michael Burke, and Subramanian Ramamoorthy. 2020. Composing Diverse Policies for Temporally Extended Tasks. *IEEE Robotics and Automation Letters* 5, 2 (April 2020), 2658–2665. DOI:https://doi.org/10.1109/LRA.2020.2972794

[9] Brenna D. Argall. 2018. Autonomy in Rehabilitation Robotics: An Intersection. *Annual Review of Control, Robotics, and Autonomous Systems* 1, 1 (2018), 441–463. DOI:https://doi.org/10.1146/annurev-control-061417-041727

[10] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57, 5 (May 2009), 469–483. DOI:https://doi.org/10.1016/j.robot.2008.10.024

[11] Kim Baraka and Manuela M. Veloso. 2018. Mobile Service Robot State Revealing Through Expressive Lights: Formalism, Design, and Evaluation. *Int J of Soc Robotics* 10, 1 (January 2018), 65–92. DOI:https://doi.org/10.1007/s12369-017-0431-x

[12] T. Barfoot, J. Burgner-Kahrs, E. Diller, A. Garg, A. Goldenberg, J. Kelly, X. Liu, H. E. Naguib, G. Nejat, A. P. Schoellig, F. Shkurti, H. Siegel, Y. Sun, and S. L. Waslander. 2020. Making Sense of the Robotized Pandemic Response: A Comparison of Global and Canadian Robot Deployments and Success Factors. *arXiv:2009.08577 [cs]* (September 2020). Retrieved from http://arxiv.org/abs/2009.08577

[13] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. 2008. Robot Programming by Demonstration. In *Springer Handbook of Robotics*, Bruno Siciliano and Oussama Khatib (eds.). Springer, Berlin, Heidelberg, 1371–1394. DOI:https://doi.org/10.1007/978-3-540-30301-5_60

[14] Jessie Chen, Ellen Haas, and Michael Barnes. 2007. Human Performance Issues and User Interface Design for Teleoperated Robots. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 37, (December 2007), 1231–1245. DOI:https://doi.org/10.1109/TSMCC.2007.905819

[15] Henry M. Clever, Ankur Handa, Hammad Mazhar, Kevin Parker, Omer Shapira, Qian Wan, Yashraj Narang, Iretiayo Akinola, Maya Cakmak, and Dieter Fox. 2021. *Assistive Tele-op: Leveraging Transformers to Collect Robotic Task Demonstrations*. Retrieved July 19, 2022 from https://ui.adsabs.harvard.edu/abs/2021arXiv211205129C

[16] Anca D Dragan and Siddhartha S Srinivasa. 2013. A policy-blending formalism for shared control. *The International Journal of Robotics Research* 32, 7 (June 2013), 790–805. DOI:https://doi.org/10.1177/0278364913490324

[17] Anca D. Dragan, Siddhartha S. Srinivasa, and Kenton C. T. Lee. 2013. Teleoperation with intelligent and customizable interfaces. *J. Hum.-Robot Interact.* 2, 2 (June 2013), 33–57. DOI:https://doi.org/10.5898/JHRI.2.2.Dragan

[18] Sarah Elliott, Russell Toris, and Maya Cakmak. 2017. Efficient programming of manipulation tasks by demonstration and adaptation. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 1146–1153. DOI:https://doi.org/10.1109/ROMAN.2017.8172448

[19] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. 2017. One-Shot Visual Imitation Learning via Meta-Learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, 357–368. Retrieved July 19, 2022 from https://proceedings.mlr.press/v78/finn17a.html

[20] Samir Yitzhak Gadre, Eric Rosen, Gary Chien, Elizabeth Phillips, Stefanie Tellex, and George Konidaris. 2019. End-User Robot Programming Using Mixed Reality. In *2019 International Conference on Robotics and Automation (ICRA)*, 2707–2713. DOI:https://doi.org/10.1109/ICRA.2019.8793988

[21] Ming Gao, Jan Oberländer, Thomas Schamm, and J. Marius Zöllner. 2014. Contextual task-aware shared autonomy for assistive mobile robot teleoperation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3311–3318. DOI:https://doi.org/10.1109/IROS.2014.6943023

[22] Yuxiang Gao and Chien-Ming Huang. 2019. PATI: a projection-based augmented table-top interface for robot programming. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ACM, Marina del Ray California, 345–355. DOI:https://doi.org/10.1145/3301275.3302326

[23] Michele Ginesi, Daniele Meli, Andrea Calanca, Diego Dall'Alba, Nicola Sansonetto, and Paolo Fiorini. 2019. Dynamic Movement Primitives: Volumetric Obstacle Avoidance. In *2019 19th International Conference on Advanced Robotics (ICAR)*, 234–239. DOI:https://doi.org/10.1109/ICAR46387.2019.8981552

[24] Deepak Gopinath, Siddarth Jain, and Brenna D. Argall. 2017. Human-in-the-Loop Optimization of Shared Autonomy in Assistive Robotics. *IEEE Robotics and Automation Letters* 2, 1 (January 2017), 247–254. DOI:https://doi.org/10.1109/LRA.2016.2593928

[25] Kris Hauser. 2013. Recognition, prediction, and planning for assisted teleoperation of freeform tasks. *Auton Robot* 35, 4 (November 2013), 241–254. DOI:https://doi.org/10.1007/s10514-013-9350-3

[26] Hooman Hedayati, Michael Walker, and Daniel Szafir. 2018. Improving Collocated Robot Teleoperation with Augmented Reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, Chicago IL USA, 78–86. DOI:https://doi.org/10.1145/3171221.3171251

[27] Laura V. Herlant, Rachel M. Holladay, and Siddhartha S. Srinivasa. 2016. Assistive teleoperation of robot arms via automatic time-optimal mode switching. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 35–42. DOI:https://doi.org/10.1109/HRI.2016.7451731

[28] Gaoping Huang, Pawan S. Rao, Meng-Han Wu, Xun Qian, Shimon Y. Nof, Karthik Ramani, and Alexander J. Quinn. 2020. Vipo: Spatial-Visual Programming with Functions for Robot-IoT Workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ACM, Honolulu HI USA, 1–13. DOI:https://doi.org/10.1145/3313831.3376670

[29] Justin Huang and Maya Cakmak. 2017. Code3: A System for End-to-End Programming of Mobile Manipulator Robots for Novices and Experts. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, Vienna Austria, 453–462. DOI:https://doi.org/10.1145/2909824.3020215

[30] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. 2013. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation* 25, 2 (February 2013), 328–373. DOI:https://doi.org/10.1162/NECO_a_00393

[31] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. 2022. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. DOI:https://doi.org/10.5281/zenodo.6222936

[32] Sang Hyoung Lee, Il Hong Suh, Sylvain Calinon, and Rolf Johansson. 2015. Autonomous framework for segmenting robot trajectories of manipulation task. *Auton Robot* 38, 2 (February 2015), 107–141. DOI:https://doi.org/10.1007/s10514-014-9397-9

[33] Jiannan Li, Ravin Balakrishnan, and Tovi Grossman. 2020. StarHopper: A Touch Interface for Remote Object-Centric Drone Navigation. In *Proceedings of Graphics Interface 2020* (GI 2020), Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 317–326. DOI:https://doi.org/10.20380/GI2020.32

[34] Ying Siu Liang, Damien Pellier, Humbert Fiorino, and Sylvie Pesty. 2019. End-User Programming of Low- and High-Level Actions for Robotic Task Planning. *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (October 2019), 1–8. DOI:https://doi.org/10.1109/RO-MAN46459.2019.8956327

[35] Ying Siu Liang, Damien Pellier, Humbert Fiorino, Sylvie Pesty, and Maya Cakmak. 2018. Simultaneous End-User Programming of Goals and Actions for Robotic

[36] Shelf Organization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain. DOI:https://doi.org/10.1109/IROS.2018.8593518

[36] Jeffrey I. Lipton, Aidan J. Fay, and Daniela Rus. 2018. Baxter's Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing. *IEEE Robotics and Automation Letters* 3, 1 (January 2018), 179–186. DOI:https://doi.org/10.1109/LRA.2017.2737046

[37] Yugang Liu and Goldie Nejat. 2013. Robotic Urban Search and Rescue: A Survey from the Control Perspective. *J Intell Robot Syst* 72, 2 (November 2013), 147–165. DOI:https://doi.org/10.1007/s10846-013-9822-x

[38] Matthew B. Luebbers, Connor Brooks, Carl L. Mueller, Daniel Szafir, and Bradley Hayes. 2021. ARC-LfD: Using Augmented Reality for Interactive Long-Term Robot Skill Maintenance via Constrained Learning from Demonstration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 3794–3800. DOI:https://doi.org/10.1109/ICRA48506.2021.9561844

[39] Jeffrey Mahler and Ken Goldberg. 2017. Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences. In *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, 515–524. Retrieved April 6, 2022 from https://proceedings.mlr.press/v78/mahler17a.html

[40] Simon Manschitz, Jens Kober, Michael Gienger, and Jan Peters. 2014. Learning to sequence movement primitives from demonstrations. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Chicago, IL, USA, 4414–4421. DOI:https://doi.org/10.1109/IROS.2014.6943187

[41] Henrique Martins and Rodrigo Ventura. 2009. Immersive 3-D teleoperation of a search and rescue robot using a head-mounted display. In *2009 IEEE Conference on Emerging Technologies & Factory Automation*, IEEE, Palma de Mallorca, Spain, 1–8. DOI:https://doi.org/10.1109/ETFA.2009.5347014

[42] Roderick McCall, Fintan McGee, Alexander Meschtscherjakov, Nicolas Louveton, and Thomas Engel. 2016. Towards A Taxonomy of Autonomous Vehicle Handover Situations. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, Ann Arbor MI USA, 193–200. DOI:https://doi.org/10.1145/3003715.3005456

[43] Carl Mueller, Jeff Venicx, and Bradley Hayes. 2018. Robust Robot Learning from Demonstration and Skill Repair Using Conceptual Constraints. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6029–6036. DOI:https://doi.org/10.1109/IROS.2018.8594133

[44] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. 2018. An Algorithmic Perspective on Imitation Learning. *FNT in Robotics* 7, 1–2 (2018), 1–179. DOI:https://doi.org/10.1561/2300000053

[45] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. 2013. Probabilistic Movement Primitives. In *Advances in Neural Information Processing Systems*, Curran Associates, Inc. Retrieved April 4, 2022 from https://proceedings.neurips.cc/paper/2013/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html

[46] Chris Paxton, Felix Jonathan, Andrew Hundt, Bilge Mutlu, and Gregory D. Hager. 2018. Evaluating Methods for End-User Creation of Robot Task Plans. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6086–6092. DOI:https://doi.org/10.1109/IROS.2018.8594127

[47] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. 2019. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Long Beach, CA, USA, 4556–4565. DOI:https://doi.org/10.1109/CVPR.2019.00469

[48] Pragathi Praveena, Luis Molina, Yeping Wang, Emmanuel Senft, Bilge Mutlu, and Michael Gleicher. 2022. Understanding Control Frames in Multi-Camera Robot Telemanipulation. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction* (HRI '22), IEEE Press, Sapporo, Hokkaido, Japan, 432–440.

[49] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. 2017. A Motion Retargeting Method for Effective Mimicry-Based Teleoperation of Robot Arms. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction*, 361–370. DOI:https://doi.org/10.1145/2909824.3020254

[50] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. 2018. An Autonomous Dynamic Camera Method for Effective Remote Teleoperation. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, Chicago IL USA, 325–333. DOI:https://doi.org/10.1145/3171221.3171279

[51] Francisco Ramos, Mohanarajah Gajamohan, Nico Huebel, and Raffaello D'Andrea. 2013. Time-Optimal Online Trajectory Generator for Robotic Manipulators. (2013), 6 p. DOI:https://doi.org/10.3929/ETHZ-A-007611532

[52] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. 2020. Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems* 3, 1 (2020), 297–330. DOI:https://doi.org/10.1146/annurev-control-100819-063206

[53] Matteo Saveriano, Fares J. Abu-Dakka, Aljaz Kramberger, and Luka Peternel. 2021. Dynamic Movement Primitives in Robotics: A Tutorial Survey. *arXiv:2102.03861 [cs]* (February 2021). Retrieved March 29, 2022 from http://arxiv.org/abs/2102.03861

[54] Andrew Schoen, Curt Henrichs, Mathias Strohkirch, and Bilge Mutlu. 2020. Authr: A Task Authoring Environment for Human-Robot Teams. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, ACM,

Virtual Event USA, 1194–1208. DOI:https://doi.org/10.1145/3379337.3415872

[55] David R. Scribner and James W. Gombash. 1998. *The Effect of Stereoscopic and Wide Field of View Conditions on Teleoperator Performance.* Army Research Lab Aberdeen Proving Ground, MD Human Research and Engineering Directorate. Retrieved April 1, 2022 from https://apps.dtic.mil/sti/citations/ADA341218

[56] Emmanuel Senft, Michael Hagenow, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. 2021. Situated Live Programming for Human-Robot Collaboration. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (UIST '21), Association for Computing Machinery, New York, NY, USA, 613–625. DOI:https://doi.org/10.1145/3472749.3474773

[57] Emmanuel Senft, Michael Hagenow, Kevin Welsh, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. 2021. Task-Level Authoring for Remote Robot Teleoperation. *Frontiers in Robotics and AI* 8, (2021). DOI:https://doi.org/10.3389/frobt.2021.707149

[58] Yang Shen, Dejun Guo, Fei Long, Luis A. Mateos, Houzhu Ding, Zhen Xiu, Randall B. Hellman, Adam King, Shixun Chen, Chengkun Zhang, and Huan Tan. 2021. Robots Under COVID-19 Pandemic: A Comprehensive Survey. *IEEE Access* 9, (2021), 1590–1615. DOI:https://doi.org/10.1109/ACCESS.2020.3045792

[59] Antonis Sidiropoulos, Yiannis Karayiannidis, and Zoe Doulgeri. 2021. Human-robot collaborative object transfer using human motion prediction based on Cartesian pose Dynamic Movement Primitives. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 3758–3764. DOI:https://doi.org/10.1109/ICRA48506.2021.9562035

[60] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. 2022. Augmented Reality and Robotics: A Survey and Taxonomy for AR-enhanced Human-Robot Interaction and Robotic Interfaces. *Proceedings of the 2022 CHI Conference on Human Factors in Computing System* (2022), 33. DOI:https://doi.org/10.1145/3491102.3517719

[61] Ryotaro Temma, Kazuki Takashima, Kazuyuki Fujita, Koh Sueda, and Yoshifumi Kitamura. 2019. Third-Person Piloting: Increasing Situational Awareness using a Spatially Coupled Second Drone. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, ACM, New Orleans LA USA, 507–519. DOI:https://doi.org/10.1145/3332165.3347953

[62] Aleš Ude, Bojan Nemec, Tadej Petrić, and Jun Morimoto. 2014. Orientation in Cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2997–3004. DOI:https://doi.org/10.1109/ICRA.2014.6907291

[63] Martin Voshell, David D. Woods, and Flip Phillips. 2005. Overcoming the Keyhole in Human-Robot Coordination: Simulation and Evaluation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 49, 3 (September 2005), 442–446. DOI:https://doi.org/10.1177/154193120504900348

[64] Michael E. Walker, Hooman Hedayati, and Daniel Szafir. 2019. Robot Teleoperation with Augmented Reality Virtual Surrogates. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 202–210. DOI:https://doi.org/10.1109/HRI.2019.8673306

[65] Mohammad Kassem Zein, Majd Al Aawar, Daniel Asmar, and Imad H. Elhajj. 2021. Deep Learning and Mixed Reality to Autocomplete Teleoperation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 4523–4529. DOI:https://doi.org/10.1109/ICRA48506.2021.9560887

[66] Mohammad Kassem Zein, Abbas Sidaoui, Daniel Asmar, and Imad H. Elhajj. 2020. Enhanced Teleoperation Using Autocomplete. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 9178–9184. DOI:https://doi.org/10.1109/ICRA40945.2020.9197140

[67] Gazebo. Retrieved April 1, 2022 from http://gazebosim.org/

[68] MoveIt Motion Planning Framework. Retrieved April 1, 2022 from https://moveit.ros.org/

[69] moveit2/panda_simulated_config.yaml at main · ros-planning/moveit2. *GitHub*. Retrieved April 6, 2022 from https://github.com/ros-planning/moveit2

[70] dfki-ric/movement_primitives: Dynamical movement primitives (DMPs), probabilistic movement primitives (ProMPs), spatially coupled bimanual DMPs. *GitHub*. Retrieved April 6, 2022 from https://github.com/dfki-ric/movement_primitives

[71] Free samples: Synthetic datasets. Retrieved April 1, 2022 from https://resources.unity.com/unity-computer-vision-datasets-hub

[72] Unity-Technologies/com.unity.perception: Perception toolkit for sim2real training and validation in Unity. *GitHub*. Retrieved April 6, 2022 from https://github.com/Unity-Technologies/com.unity.perception

[73] The robot shop worker controlled by a faraway human - BBC News. Retrieved April 6, 2022 from https://www.bbc.com/news/business-54232563

[74] Moonlight Game Streaming: Play Your PC Games Remotely. Retrieved April 2, 2022 from https://moonlight-stream.org/