

Clonemator: Composing Spatiotemporal Clones to Create Interactive Automators in Virtual Reality

Yi-Shuo Lin*

r10922069@csie.ntu.edu.tw
National Taiwan University
Taipei, Taiwan

Ching-Yi Tsai*

ching-yi.tsai@hci.csie.ntu.edu.tw
National Taiwan University
Taipei, Taiwan

Lung-Pan Cheng

lung-pan.cheng@hci.csie.ntu.edu.tw
National Taiwan University
Taipei, Taiwan

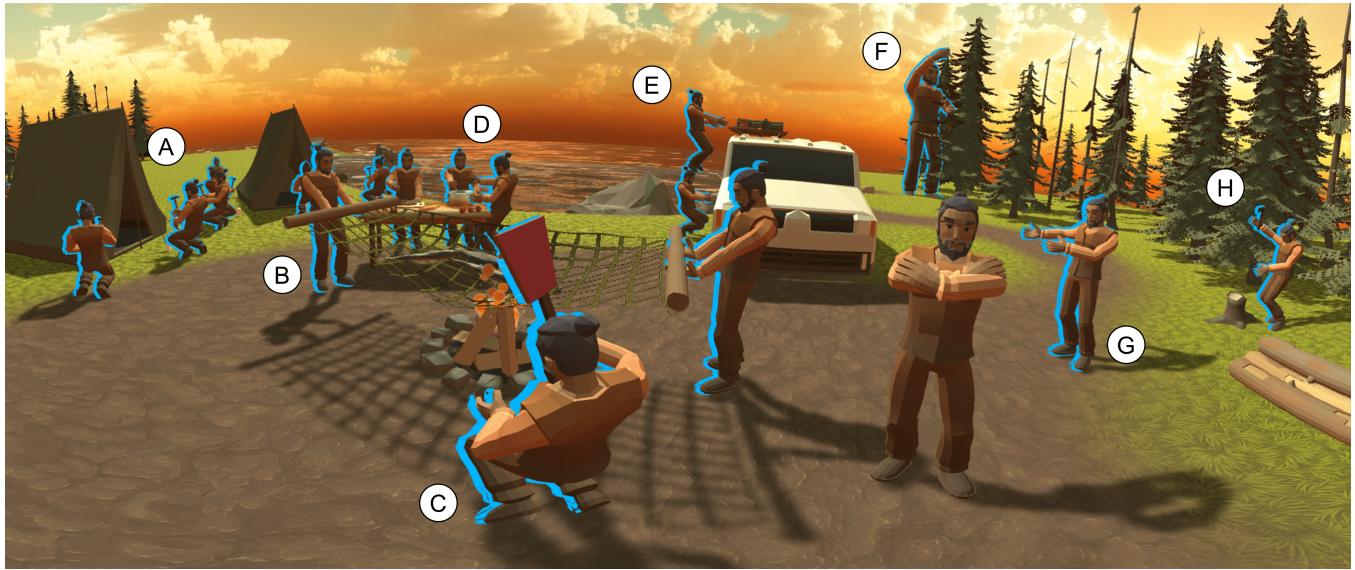


Figure 1: Clonemator allows a user to clone their avatars in various spatiotemporal configurations and collaborate with them to achieve complex tasks in virtual reality: (A) a group of clones synchronously follow the user to hammer the tent pegs; (B) a clone mirrors the user’s movement to help spread or fold the net; (C) a clone is fanning the fire while (D) the user is chopping food with a group of self-recorded clones helping cook; (E) the user steps on a clone; (F) a giant clone; (G) a body-sign clone; and (H) a remote clone replaying the logging experience previously performed by the user.

ABSTRACT

Clonemator is a virtual reality (VR) system allowing users to create their avatar clones and configure them spatially and temporally, forming automators to accomplish complex tasks. In particular, clones can (1) freeze at a user’s body pose as static objects, (2) synchronously mimic the user’s movement, and (3) replay a sequence of the user’s actions in a period of time later. Combined with traditional techniques such as scaling, positional rearrangement, group selection, and duplication, Clonemator enables users to iteratively develop customized and reusable solutions by breaking down complex tasks into a sequence of collaborations with clones. This bypasses implementing dedicated interaction techniques or scripts while allowing flexible interactions in VR applications. We demonstrate the flexibility of Clonemator with several examples and validate its usability and effectiveness through a preliminary user study. Finally, we discuss the potential of Clonemator in VR applications such as gaming mechanisms, spatial interaction techniques, and multi-robot control and provide our insights for future research.

*These authors contributed equally to this research.

CCS CONCEPTS

- Human-centered computing → Virtual reality.

KEYWORDS

virtual reality, clone, beyond-real interaction, automator

1 INTRODUCTION

Virtual reality (VR) has gained popularity with the commercialization of head-mounted displays. Over the years, many researchers have proposed several interaction techniques to enhance experiences, including the Go-Go technique [24], the Worlds-in-Miniature technique [29], and portals [12].

In recent years, researchers have started exploring the potential of augmenting the human body in VR with duplicated body parts, such as a sixth finger [8], a third arm [5], or supernumerary hands [16]. Studies even show that extra limbs can improve performance [25]. More recently, researchers also have tried to replicate full bodies in VR, and indicated that it can affect task performance and reduce physical movements [19].

While each VR technique works well by itself, there is hardly any chance of combining them to create new solutions to solve

complex tasks as they are usually dedicated to certain types of jobs. Technically speaking, it is challenging to integrate all these VR techniques into a single system due to the complexity of development and unexpected exceptions. Also, once the VR designers decide, users have little chance to adjust the settings or parameters of these VR techniques.

In this paper, we look into finding a more general approach that enables a user to decompose complex tasks into smaller, solvable problems by combining the users themselves. Our work is analogous to previous work on automation by demonstration, such as Sikuli [35] and SUGILITE [14] that configure automation through actions within naive users' understandability, and come up with an intuitive and enjoyable way to tackle dynamic challenges. We present Clonemator, a VR system that empowers users to create clones in VR. Configuring and collaborating with spatiotemporal clones offers an understandable approach to tackling each decomposed problem, as users can utilize solutions they've used before. Clonemator also visually preserves spatiotemporal contexts with clones, offering a way of visualizing the entire solving procedure.

Figure 1 depicts an overview of a user who uses Clonemator at a campsite. In the following section, we use this as our example walkthrough to demonstrate how the key components of Clonemator work.

2 EXAMPLE WALKTHROUGH

At the start of the experience, the user finds himself alone at a campsite. His goals are to set up camp with the help of Clonemator, including pitching a tent, preparing food, catching fish and building a campfire.

As the user walks to the tents, he notices many loose pegs surrounding the tents that need to be hammered. The user grabs a hammer and spawns clones in front of the pegs using *Auto Spawning* (Figure 2A). Clones are then spawned with the same offset from the corresponding pegs, allowing the user and the clones to hammer the pegs simultaneously (Figure 2B). Note that all clones are outlined in blue for easy distinction.

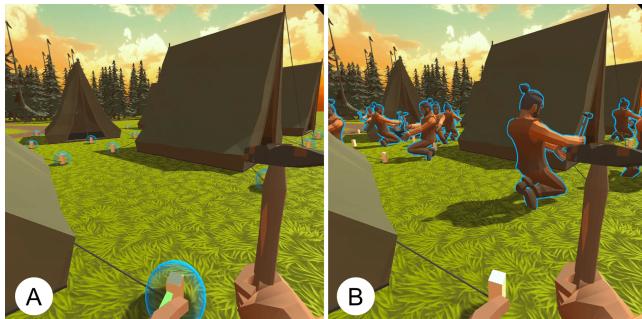


Figure 2: (A) When the user hovers on a pegs, other pegs will also be highlighted. (B) The clones will be spawned based on the offset between the user and the selected peg. Our system calculates this offset and applies it to each peg to spawn the corresponding clone, ensuring that all the clones can hammer the peg precisely.

Next, the user decides to catch fish for dinner but finds it difficult to control the fishing net alone. To solve this problem, he uses *Relative Spawning* to spawn a clone in front of the handle on the other side of the net, ensuring that the offsets between the avatars and the handles are the same so that they can grab the two handles at the same time. After creating a synchronous clone, the user mirrors its movement by flipping it along the vertical axis. When the user moves backward, the clone moves backward, making spreading the fishing net easier. On the other hand, when the user moves to the right, the clone moves to the left, allowing them to move in the same direction since they are facing each other. To rotate the net, the user mirrors the clone's movement again to flip it back, allowing them to move synchronously and rotate in the same direction.

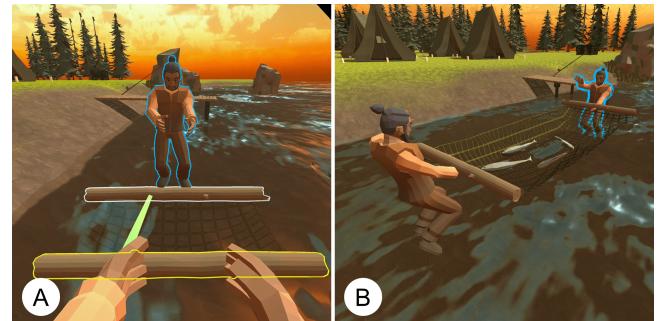


Figure 3: (A) The user spawns a clone using *Relative Spawning*, which ensures that the clone can grab the handle of the other side (outlined in white) when the user grabs the handle in the front (outlined in yellow). (B) The user creates a synchronous clone and mirrors its movement, enabling them to move in the same direction and catch the fish together.

Once the user catches the fish, he sets the clone to static mode and performs *Direct Spawning* to create another clone at the exact position. The two clones each help hold one handle of the fishing net. To move the fish and net near the table, the user can group the two clones first. This allows the user to directly grab one of the clones and move them together without altering the spatial relationships between the clones. Then, the user realizes that he needs to get a slice of beef on top of the van, which is too high to reach. The user crouches and performs *Direct Spawning*, leaving the original body as a clone on the ground. The clone can then serve as a stationary step stool, allowing the user to step onto it and reach the beef (Figure 4). Note that in our current implementation, stepping onto a clone is achieved by teleporting to the clone's head since the user's legs are not tracked.

The user brings the beef to the table and starts cutting it with a knife but notices that the campfire is almost extinguished. Since the user prefers not to wave his hands twice, he wants to cut the beef and fan the campfire at the same time. The user first spawns a clone near the campfire using *Indirect Spawning* (Figure 5A) and switches to it to grab the fan (Figure 5B). The user then switches back to the original body and grabs the knife from the table. Finally, the user updates the clone near the campfire to synchronous mode so that it can mimic the user's movement (Figure 5C). This allows

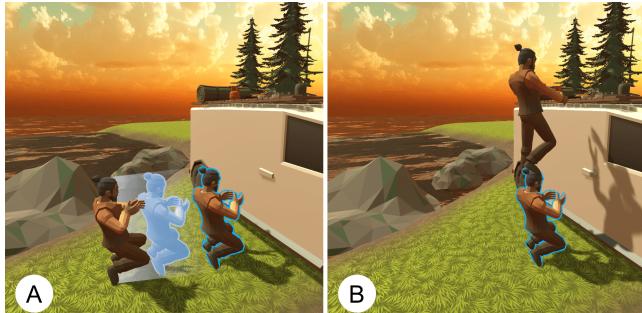


Figure 4: (A) The user crouches and performs *Direct Spawning* to create a static clone. Note that the figure has been edited to visualize the avatar's moving trajectory for easier understanding. (B) The clone now functions as a step stool, enabling the user to step onto it and reach the desired object.

the user to cut the beef and fan the campfire simultaneously since both interactions share similar movements (Figure 5D).



Figure 5: (A) The user spawns a clone at the desired location using *Indirect Spawning*. (B) The user switches to the clone to grab the fan from the ground. (C) The user changes the clone's interaction mode from static to synchronous using a ray. (D) With the assistance of the clone, the user can now cut the beef while simultaneously fanning the campfire.

Then, the user wants to add ingredients such as apples and canned food to the boiling soup while keeping it stirred to prevent burning. By recording himself stirring the pot and applying the recorded actions to a clone, the user can focus on adding ingredients while the clone continues to stir the soup (Figure 6).

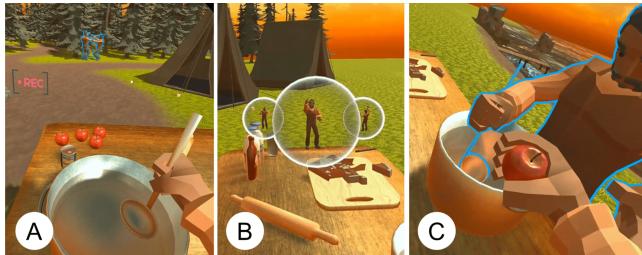


Figure 6: (A) The user records himself stirring the pot. During the recording stage, an icon appears in the upper left corner. (B) The user can preview and select previous recordings using a user interface inspired by [34]. (C) Once the user applies the recorded actions to the clone, he can delegate the task of stirring the pot to it and focus on other tasks such as adding ingredients.

When the user wants to collect more apples, he constructs a robotic arm by arranging multiple synchronous clones in a row and having them grab each other (Figure 7). By adjusting the number and size of the clones, the user can achieve different control-display ratios, enabling him to customize the arm's reach and precision to suit his needs. The flexibility of the simulated robotic arm can be further enhanced by altering the interaction modes of specific clones. For instance, changing parts of clones to static results in a joint-like movement, allowing the user to bend and twist the robotic arm at different angles.

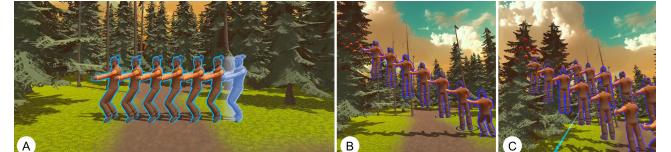


Figure 7: (A) The user performs *Direct Spawning* several times to form a chain of clones. (B) By having adjacent clones grab each other, the clones become a robotic arm that enables the user to reach apples on the tree. (C) The user can duplicate the robotic arm pattern to reuse it in other scenarios.

Suddenly, a cabin near the campsite catches fire, and the user must extinguish it quickly. He stacks four clones vertically and sets up the clones so that the first and third clones move in sync while the second and fourth clones mirror their movements, forming a vertical bucket brigade to transfer water to the roof efficiently (Figure 8).

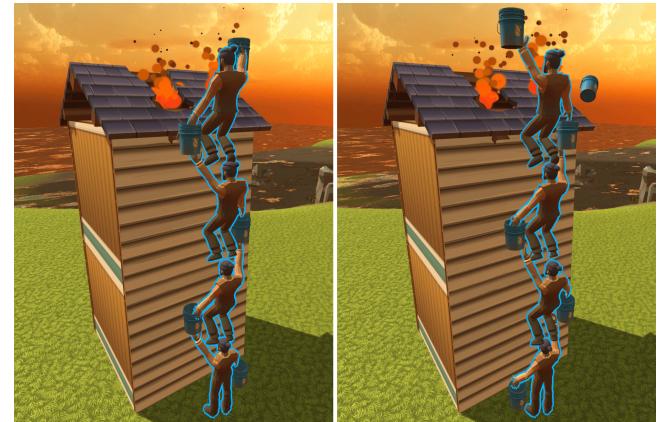


Figure 8: The user creates synchronous clones and forms a vertical bucket brigade by mirroring adjacent clones' movements. As one clone raises its left hand and lowers its right hand, the adjacent clone mirrors this movement by lowering its right hand and raising its left hand. It allows a single user to pass buckets to extinguish a fire efficiently.

After extinguishing the fire, the user decides to relax by dancing. By assigning recorded actions sequentially to multiple clones, users can delay the clones' movements and create a phase shift between them. This allows the user to perform the Thousand-Hand Bodhisattva Dance with the clones (Figure 9A).

When the user encounters an intersection in the forest, he leaves a static clone as a road sign to help him navigate more easily (Figure 9E). By creating clones that perform different postures, the user can create personalized gestures that are more intuitive than traditional flags or icons. In addition, the user can record and replay his body movements to create even more expressive guides, allowing him to convey more detailed information.

Finally, the user takes advantage of the unique properties of clones to create painting in VR. By using a rotoscoping technique on a clone, the user can easily sketch any desired posture (Figure 9B). He can also create symmetrical paintings collaboratively with synchronous clones (Figure 9C, D).

2.1 Summary of the Walkthrough

While we demonstrate the effectiveness of Clonemator within a camping scenario, it is important to note that all the showcased spatiotemporal interactions are general and applicable to other scenarios and tasks as well. To provide a comprehensive overview of the interactions enabled by Clonemator, we summarize their taxonomy in Figure 10.

3 CONTRIBUTIONS

The key benefit that Clonemator brings is its capability to maintain spatial and temporal **continuity** [27, 30], and thus result in the high **understandability** for the users to build functional interactions. While there have been methods for composing VR interaction techniques, the predominant approach involves employing dedicated programming languages within a graphical user interface (GUI) and separately observing the results in 3D spatiotemporal environments. This discrepancy results in a discontinuity between the spatiotemporal context within VR and the process of constructing interactions (GUI coding). In this paper, we argue that Clonemator is the interface for maintaining seamless spatiotemporal continuity for building VR interaction techniques. While every interaction technique includes **users** itself and a corresponding input action, Clonemator empowers users to temporally reuse a previous **action** and spatially configure the **users** themselves (in the form of clone). By combining these spatial and temporal manipulations, Clonemator enables users to build complex solutions and to adapt interaction techniques, all originating from the fundamental components of simple demonstrations of themselves, using the most familiar interaction they know.

In this paper, we make the following contributions:

- We explore the possibilities and potential for interactions between clones by providing several examples and discussing our insights.
- We develop a VR system that allows users to create clones with different interaction modes, including static, synchronous and replayed. We also integrate several additional techniques such as duplicate, mirror, and group to offer users flexibility in composing clones and building their own automators.
- We conduct a preliminary user study to validate the intuitiveness and effectiveness of the key components of Clonemator.

4 CLONEMATOR

The key design space of Clonemator is spatial and temporal controls of clones.

In this section, we begin by outlining the essential components necessary for enabling Clonemator and how we implement them. Then, we elaborate on how these key components can act as unifying operations for composing other existing interaction techniques for forming new, dynamic, and reusable automators or interactions.

4.1 Key Component #1: Spatial Manipulations

The core spatial manipulations facilitating Clonemator encompass three fundamental aspects: (1) **Spawning Method** and **Duplication** for clones, (2) **Switching Control** among differently located clones, and (3) **Grouping** a configured set of clones.

Spawning Method. The fundamental atomic action in Clonemator is cloning the users themselves. In our current implementation, anticipating that the spawning method would be the most frequently used, we offer four variations of clone spawning. These four methods essentially represent the same function through different polymorphisms: Direct, Indirect, Auto, and Relative.

- *Direct Spawning*: allows the user to create a clone at the exact position and in the same posture as the current avatar. When triggering *Direct Spawning*, our system duplicates the current avatar and smoothly moves the user backward by a customized offset, simulating the feeling of leaving the original body. The original body then becomes a clone, facilitating precise spawning. It is useful when the user wants to preserve the context of his current interaction, such as when holding a ladle (Figure 6A). In this case, the clone will continue to hold the ladle while the user can move away and perform other actions.
 - *Indirect Spawning*: enables users to immediately spawn a clone at the desired location using a ray. Users can adjust the clone's rotation along the vertical axis by pressing the thumbstick to the left or right.
 - *Auto Spawning* and *Relative Spawning*: exhibit a higher level of context awareness. Using *Auto Spawning*, the user can select an object and spawn clones in front of other objects of the same kind while preserving the relative position and rotation offsets. In our current implementation, we detect objects of the same kind by searching for objects with the same tag that is manually assigned beforehand. *Relative Spawning* can be used when the user wants to spawn clones in front of arbitrary objects. When selecting a reference object, Clonemator calculates the offset between the reference object and the user. Then, as the user selects the next object, Clonemator spawns a clone based on the position of the selected object and the calculated offset. Keeping the same offset between objects ensures that the user can interact with those objects simultaneously.
- When spawning a clone while holding an object using *Auto Spawning* or *Relative Spawning*, the object is also duplicated and distributed to the clone's hand since we aim to allow users to perform the same interactions with their clones (Figure 2B). We also set the default interaction modes of the clones spawned by these methods to synchronous.

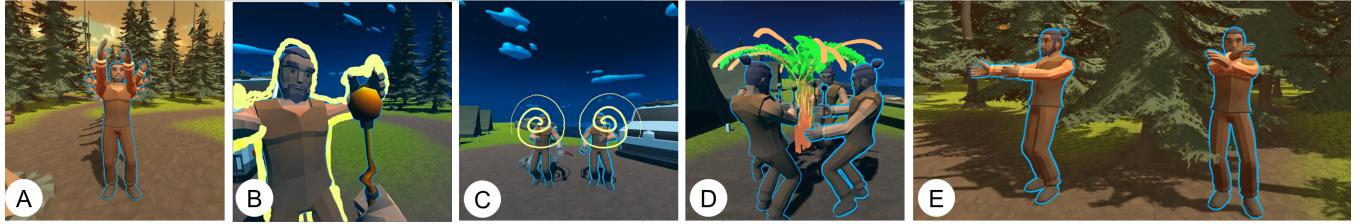


Figure 9: (A) A group of clones with delayed movements allows the user to perform the Thousand-Hand Bodhisattva Dance. When painting in VR, the user can (B) roscope on a clone with any postures, (C) create symmetrical paintings by mirroring the clone’s movement, and (D) create symmetrical paintings by arranging the clones’ positions and orientations. (E) The user can also generate customized and expressive road signs by creating clones with different postures.

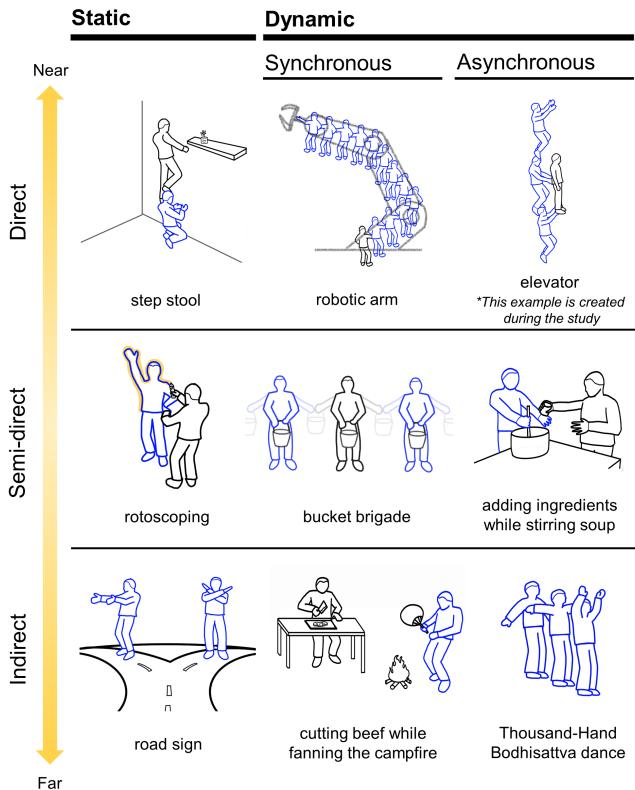


Figure 10: The solution space provided by Clonemator and the corresponding walkthrough examples. Depending on the distance between the clones and the user, the interactions between them can be categorized as either (1) direct, (2) semi-direct, or (3) indirect. In terms of the temporal aspect, the clones can be (1) static, (2) synchronous, or (3) asynchronous. These spatial and temporal dimensions together define the possibilities of Clonemator.

Switch Control. While users can create numerous clones simultaneously, human’s ability to manage multiple entities with multiple perspectives is limited [19]. Consequently, implementing a feature that enables users to transition between clones becomes imperative. Our current implementation allows users to switch between

clones by selecting the desired clone using a ray. After switching to a clone’s body, the user can gain first-person control over its movements, enabling more accurate adjustments and fine-tuning of the clone’s position and interaction. To minimize potential disorientation after switching, we implement an interpolation method to smoothly transition the positions and rotations of the user’s viewpoint between the original avatar and the clone. We also reduce the field of view during the transition to mitigate motion sickness. While this implementation employs one perspective for the users at the same time, an alternative approach could involve a multi-perspective system such as OVRlap [26].

Group. Group locks the spatial relationship between multiple clones. In our current implementation, users can group multiple clones by selecting them with a ray. Once the clones are grouped, they will be visually distinguished with the same outline color. Users can manipulate them by moving, applying interaction modes, duplicating, and removing them. Grouping serves to maintain the existing spatial relationships between clones. Without grouping, every time users wish to reuse a previous automator (*i.e.* a group of clones), it would potentially lead to the reconfiguration of the relationship between them, such as relative positions or rotational offset.

Duplicate. The duplicate function is indispensable to facilitate the reuse of previous automators (*i.e.* a group of clones), which enables users to reuse complete automators with configured properties. In our current implementation, users can grab and pull off a single clone or a group of clones from a distance using a ray to make a duplication, which preserves the original recorded interactions or properties of the original automator. Although the interaction of creating a duplicate through pulling off an avatar is similar to that in SpaceTime [34], the main difference in Clonemator is that the duplicated clone can also interact with the virtual world. Notably, duplicating a single clone is equivalent to spawning one. Therefore, an alternative implementation of Clonemator involves retaining only the duplication function, as duplicating is inclusive of spawning.

4.2 Key Component #2: Temporal Manipulations

To enable Clonemator, our temporal manipulations offer three interaction modes for clones: (1) **Static**, (2) **Synchronous**, and (3) **Replayed**.

Static Mode. In *Static Mode*, the clone freezes in time and remains stationary. It is valuable when the user needs a stable platform or needs help holding objects in place. For example, the user can hang a lantern on the clone to illuminate a dark forest. This frees up the user’s hands and allows the user to perform other tasks such as gathering woods. Additionally, a static clone can serve as a reference point. For example, the user can leverage a static clone’s body as a measuring tool for comparing the height of a door when building a house. While *group* locks the spatial relationship between multiple clones, *static* mode locks the temporal property of a single clone.

Synchronous Mode. In this mode, the clone follows the user’s movements precisely, creating a one-to-one replication of the user’s actions. Users can create a group of synchronous clones to perform repetitive tasks or collaborate with them synchronously. It is also useful for activities that require synchronization among multiple clones, such as passing objects over a long distance using the bucket brigade technique.

Replayed Mode. Clonemator allows a clone to perform previously recorded sequences of actions in a loop repeatedly. This allows the users to record their physical movements and their interaction events. Users can access a 3D carousel menu that shows previews of all previous recordings to choose the desired one and apply it to an existing clone or the users’ own avatar themselves. This mode is useful when the user needs to repeat certain interaction in the future. One way to conceptualize the replayed mode is as a form of temporal grouping that captures a series of past user actions and interactions. When a sequence of movements is recorded and replayed, the time offset between these movements remains fixed. Moreover, it allows users to collaborate to achieve more complex interactions with their past selves. Currently, users can set a clone’s interaction mode in our implementation by pointing a ray.

These modes represent various temporal behaviors for clones: Static mode keeps the clone fixed at a single moment, Synchronous mode allows the clone to perform future actions alongside the user, and Replayed mode enables the clone to reenact past actions.

4.3 Composing Automators through Spatiotemporal Manipulation

In this subsection, we explain how Clonemator can create an automaton that acts as a new interaction technique beyond the system’s original set of interactions by combining existing interactions spatiotemporally in the VR system. As illustrated in Figure 11, let’s consider a VR environment where the fundamental components of Clonemator are implemented, along with an additional technique called *remove* (which allows the removal of a group or a single clone).

To establish an automaton enabling a teleportation technique (where users can instantly appear at a different location), the user can initiate the recording function in replay mode (temporal manipulation). They can then spawn a clone in front of themselves, switch control to the newly created clone (spatial manipulation), turn around and *remove* the avatar that has been deactivated, and finally, return to the original forward orientation. After completing

this sequence of actions, the user will find themselves at a new location, at which point they can stop the recording.

From that point forward, whenever the user needs to perform a teleportation, they can trigger these automated recordings from the replayed 3D carousel menu, applying them to themselves. This results in a reusable and functional *teleportation* technique. Notably, each step falls within the system’s original capabilities, making it highly understandable, as they consist of basic, straightforward spatiotemporal operations.

When a system contains a different interaction set, then Clonemator may provide the possibility to form a different resulting solution space. For example, if we take *remove* out of the aforementioned VR system and add *scale* function to it (where users are allowed to resize the avatar and clones), then we can possibly create a World-in-Miniature function [29] with **synchronous** control between the users and a large giant clone.

5 IMPLEMENTATION

We implement Clonemator using Unity version 2020.3.18f1, running on an HP VR Backpack G2 computer equipped with a NVIDIA GeForce RTX 2080 graphics card. The VR content is streamed to a Meta Quest 2 headset via Quest Link over a USB cable. The display resolution is 1832×1920 per eye, and the refresh rate is set to 72 Hz. The user holds two Meta Quest 2 controllers as input devices.

The tracking data from the headset and controllers are distributed to both the original avatar and its clones. Specifically, we map the positions and the rotations of the headset and the controllers to the corresponding joints (i.e., the neck and the wrists) on a humanoid avatar and use Unity’s built-in inverse kinematics system to animate the avatar’s full body movements. Hence, one limitation of our system is that users can only record their upper body movements and can not control their legs to form more dynamic poses.

5.1 User Interface

To help users get started faster, we have implemented a 2D menu that can be toggled by pressing the controller button. The menu is mounted on the user’s left hand and provides access to all the commands and functions.

Additionally, Clonemator offers voice command functionality for experienced users. The voice recognition system is developed using Unity’s built-in keyword recognizer, which can respond to pre-defined keywords. The microphone of the headset captures the voice. Users can initiate a voice command anytime as the microphone is always active. Users can also access a list with all available voice commands inside the menu. It allows users to seamlessly control Clonemator faster without breaking their immersion. In situations where a 2D menu is not feasible, such as when the user’s hands are fully occupied with other tasks, voice commands become a valuable alternative.

5.2 Added Interaction Technique

While we have covered the essential functions enabling Clonemator in Section 4, its true potential lies in its capacity to integrate and combine existing interactions that were accommodated within the current virtual world, thereby generating novel, adaptable, and

Record Sequence

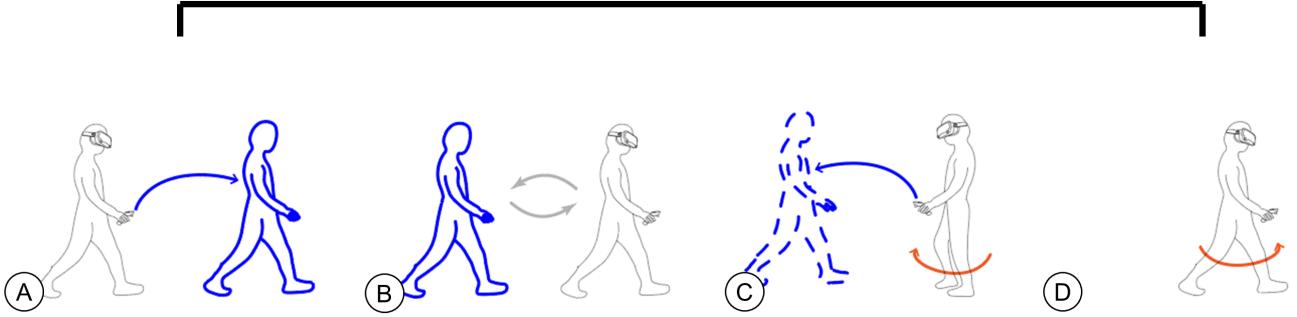


Figure 11: A teleportation technique can be assembled through the spatiotemporal arrangement of Clonemator with an additional remove technique. A user can record their sequence of actions, including (A) spawning a clone, (B) switching control to the newly spawned clone, (C) turning back and remove the original avatar, and (D) returning to the original orientation. Subsequently, they can easily apply this stored, recorded sequence to themselves whenever they wish to teleport without redoing each decomposed step. This creates an automated process for achieving teleportation.

understandable solutions. A different collection of these added interaction techniques may span a different solution space. Here, we provide an overview of the added interaction techniques we have incorporated into our current implementation. Note that the selection of these interaction techniques is for illustrative purposes and the following user study, the concept of Clonemator allows VR designers to determine their own set of added interactions for enabling a different possible solution space for the users to explore.

- Snap: Although *Indirect Spawning* provides a quick way to spawn a clone at a distance, it lacks precision in determining the spawning position. To further ease the user’s effort when positioning the clone, Clonemator introduces two snapping methods, which can be toggled using the controller button.

Grid Snapping divides the world into a grid, with each cell having a length equivalent to the user’s arm length. It ensures that the spawned clones can interact with each other if they are located in two adjacent cells. This method is convenient for maintaining consistent offsets between avatars.

Nearest Object Snapping snaps the clone to the closest object while maintaining an arm’s length offset. This method is useful when spawned clones need to interact with a specific object. Both snapping methods can be customized by using a slider to adjust the offsets according to different situations.

- Avatar Rotation and Teleportation: We enable users to navigate easily by rotating their bodies or teleporting using the controller. Since Clonemator currently synchronizes physical movements, users can use avatar rotation and teleportation to prevent unintentional movements of synchronous clones. For example, when users rotate themselves using the controller, the clone will remain stationary, enabling them to establish a rotational offset.
- Scale: Users can adjust the size of clones by pressing the thumbstick backward and forward during *Indirect Spawning* or when grabbing an existing clone. Users can also switch between clones of different sizes to perform interactions at varying scales [10].

- Mirror: Mirroring allows the user to flip a clone’s movement horizontally, creating a mirror image of their own actions. For instance, when the user raises their right hand, the clone will raise its left hand, and vice versa. It provides intuitive control, especially when the user and the clone face each other as their movements are reflected, much like looking into a mirror. Mirroring is also helpful for activities that require symmetric movements, such as swinging a long jump rope together or performing synchronized dance moves.

- Remove: Users can select and remove a clone using a ray. If the clone is holding an object, the object will dissolve and appear near the user. The mental model behind this feature is that the removed clone will merge back with the user. For example, the user can assign a clone to gather wood and then remove the clone to obtain the collected wood.

- Undo: Clonemator also provides an Undo feature, which allows the user to cancel previous spawning commands. It is especially useful when the user wants to remove multiple clones created by *Auto Spawning* at the same time. Our system also allows users to undo grouping and duplication commands, reducing manual effort.

6 PRELIMINARY USER STUDY

Since Clonemator can possibly utilize infinite supporting interaction techniques to form an even larger solution space, it was challenging to identify a particular collection of supporting interactions to show its full potential. As the first step to understanding Clonemator, we conducted a preliminary study whose main evaluation is around Clonemator’s key components only, including user experience and flexibility in solution-making. To allow participants to focus on the key components of Clonemator, we utilized the smaller supporting interaction set employed in our walkthrough and implementation, and we restricted the replay function to record only the user’s physical actions.

6.1 Design, Tasks, and Participants

The study includes 9 tasks in the VR camping scene presented earlier in Section 2, and is composed of two sessions: a first session with example solutions and a second session without.

In the first session, participants completed five tasks, including four specific tasks from the walkthrough in Section 2 (hammering multiple pegs, catching fish, cooking while adding ingredients, and fanning the campfire while cutting beef) and a free-exploration task. Prior to the five tasks, a video tutorial was given, explaining the tasks' goal, providing the sample solutions from the walkthrough, and demonstrating the necessary UI elements and actions. Participants had to replicate the sample solutions in the VR scene after viewing the video. The free-exploration task lets participants test the operation not covered in the first four tasks, such as remove, duplicate and group, and allowed participants to explore the scene freely with all operations available. The first session familiarized participants with the system and provided a general understanding of Clonemator's experience.

In the second session, participants had to complete the other four tasks, including:

- Moving a table that is too heavy for a single person, so the participants have to collaborate with or utilize multiple clones
- Passing multiple basketballs as fast as possible from a starting point to a target that is 9 meters away without teleportation
- Fetching a slice of beef at 2.5 meters height from the top of a van
- Fetching an apple at 7.5 meters height from the top of a tree, where a distinct solution from fetching beef was required.

Participants were only given the task goal and requirement without any further instructions provided. These tasks are designed for utilizing more complex clone manipulation, and since participants had to come up with their own solutions, this session is aimed at testing not only the user experience but also the richness of the solutions that Clonemator can support.

We recruited 12 participants (8 male, 4 female) who had no prior knowledge of Clonemator's details, aged from 19 to 27 ($M = 23.8$, $SD = 2.4$) through word of mouth. 2 had VR experience more than once a week, another 3 had VR experience about once a month, 1 had VR experience about once every three months, and the remaining 6 had very rare VR experience (about once a year or less). The apparatus used for the study is the same as mentioned earlier in Section 5.

6.2 Solutions Generated by Participants

During the second session of the study, we observed that participants were able to come up with different solutions for the given tasks. For example, P6 used a synchronous clone to catch a basketball he threw (Figure 13A), while P5 and P11 recorded themselves throwing a ball and applied the recording to a clone, catching the ball at the destination by themselves. P2 successfully built a bucket brigade made of clones using the *Direct Spawning* technique and set their interaction modes to synchronous (Figure 13C). The first clone grabbed a ball with its right hand and passed it to its left hand, while the next clone used its right hand to grab the ball held by the left hand of the previous clone, forming a pipeline to pass multiple balls efficiently.

For fetching a high object (including fetching beef and fetching an apple), P4 used *Relative Spawning* to spawn a clone in front of the apple by setting the reference object to the gas cylinder next to her and then switched to the clone to get the apple. P3, P6, P7 and P11 directly threw a clone to the top of the tree and switched to it. P5 recorded the action of grabbing and lifting objects and applied it to vertically stacked clones with different phase shifts, forming an elevator that allowed the clones to pass him to the tree top (Figure 13B). P8 created a step stool by positioning a static clone near the camper van and used it as a platform to reach for the beef.

Overall, participants created 5 different solutions for fetching a high object and 4 different solutions for passing basketballs, which indicate Clonemator's flexibility in providing dynamic, adaptable solutions.

6.2.1 Overall Experience. On “*the overall experience of using the system is enjoyable*,” participants reported an average of 6 points ($SD = 1.13$) on a 7-point Likert scale (1- “*strongly disagree*”, 7- “*strongly agree*”). Participants responded positively about the experience of completing tasks with Clonemator, including “*It's pretty cool. It's interesting and fun.*” [P1], “*It's a fun game.*” [P2, P7, P10], “*It's pretty fun, even though it's my first time using it.*” [P3], “*It's quite fun. The experience is interesting.*” [P5], “*You can do many things due to variability and combinations, so it's pretty interesting.*” [P11].

6.2.2 Understandability and Usefulness. Regarding the system's usefulness, participants noted “*Even without instructions, one should be able to figure it [the solution] out.*” [P1] and “*I can think of a solution easily.*” [P12]. For learning the overall control of the system, participants added “*It is not difficult to learn, there are many things but all the operations are basic.*” [P2]. “*Once you're familiar with what you can do with the clones, it's quite simple.*” [P11]

Regarding spawning the clones, participants express it's natural as “*The method of creating clones is quite intuitive.*” [P3]. Specifically, “*Out-of-body is convenient and most intuitive*” [P1, P12] ... “*because one can determine the position of their clone by his body.*” [P1], and provides their view on tradeoff between different spawning methods as “*It's harder to image its result for auto-spawning but it has huge potential for mass deployment.*” [P12] In terms of controlling the clones, “*simultaneously waving the wind and cutting foods [using synchronous mode] is intuitive because the movements are consistent.*” [P1], and P3 mentioned her controlling metaphor, by “*to think about how to operate different bodies at the same time, it's like controlling a machine remotely.*” [P3]

6.2.3 Use Cases and Further Applications. The participants have also suggested additional features. For example, editing or managing replayed clones “*Replay could benefit from naming, searching, and categorizing. Also, I hope to extract certain actions during a replay sequence*” [P7], as well as more dexterous control with “*complex actions such as finger movement.*” [P11]

Although we make sure the keyword or notion of automation is never mentioned in our study instruction, P2 drew a comparison between Clonemator and such concept, by “*It [the system] is like writing code modules and microservices. Using the switch or replay to trigger pre-written functions is like a game plugin*” and “*similar to a macro or script but for repeated tasks in VR.*”

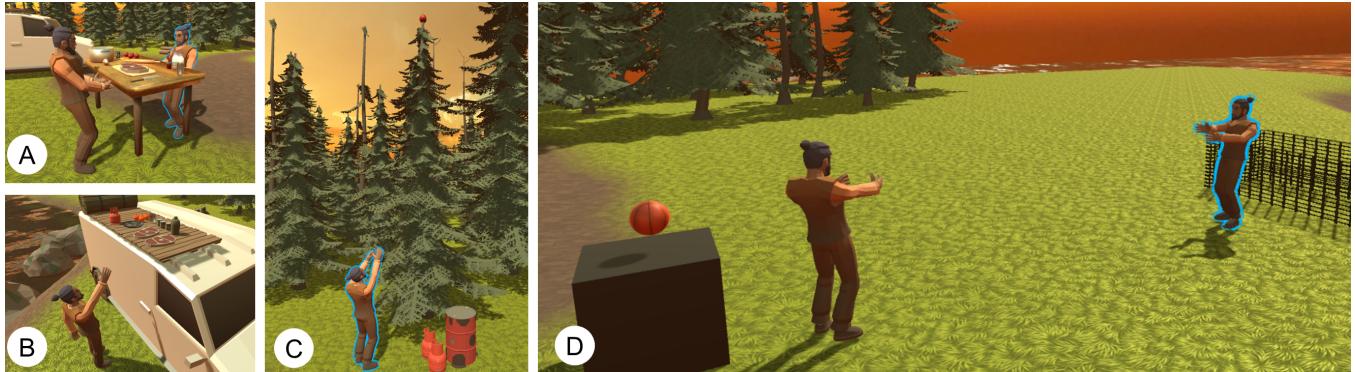


Figure 12: Study tasks without sample solutions. Participants have to complete four challenges without explicit instructions. These challenges included: (A) lifting a heavy table, (B) fetching a slice of beef from the top of a van, (C) fetching an apple from the top of a tree using a solution different from that used for fetching the slice of beef, and (D) passing basketballs across a distance without teleportation as fast as possible.

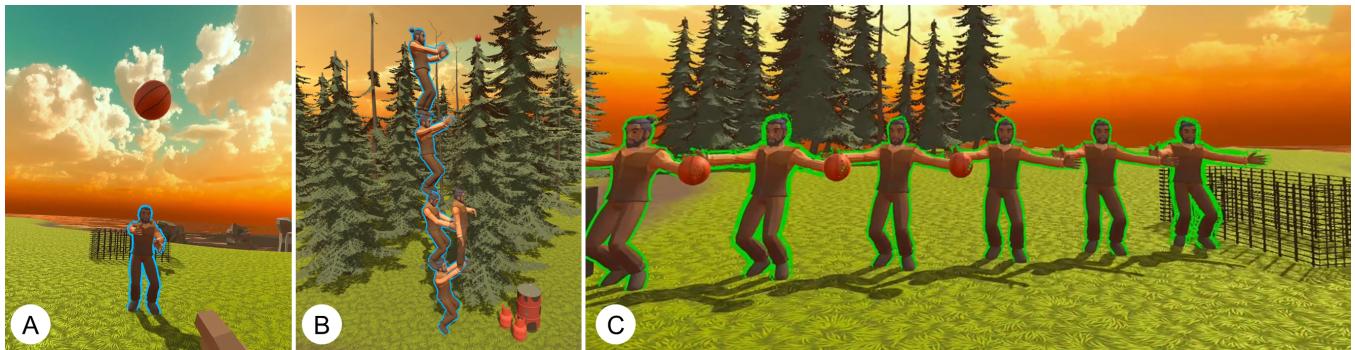


Figure 13: Solutions come up by participants: (A) The user throws a ball and catches it with a synchronous clone. (B) The user replays the lifting movement to multiple clones stacked vertically, forming an elevator. (C) A bucket brigade formed by a chain of clones with synchronous movement.

Furthermore, participants proposed additional potential applications of Clonemator, which encompassed “3D modeling in VR” [P1], “gaming” [P1, P2, P3, P5, P8, P10, P11, P12], and “demonstration or teaching” [P6, P7, P12]. Specifically, P7 suggested that “[Clonemator] can be used for teaching and even demonstrate tasks that require collaboration or teamwork but by one’s self.” [P7], while participant P2 mentioned the composition of clones “can improve the experience of games like Story of Seasons, which involves a lot of repeated farming tasks.”

6.3 Study Summary

The study results indicate that Clonemator provides diverse and adaptable functionalities for various tasks, and the key operations of Clonemator deliver a user-friendly and enjoyable experience with intuitive controls.

7 RELATED WORK

The presented work draws on several areas of previous research, including Replicating Movement in the Real World, Beyond-Real Interaction, and Programming by Demonstration.

7.1 Replicating Movement in the Real World

Previous studies in teleoperation have investigated the potential of replicating control or movement in real-world scenarios. This involves remotely controlling one or multiple robots by one single user. For instance, Glas et al. [7] designed a coordination system that enables a user to operate four robots at the same time. Researchers have also explored ways of controlling two robotic arms synchronously or asynchronously through the use of EMG signals and gaze control [20]. Additionally, in Transfantome [10], users can control two robots concurrently at different scales by embodying two proxy bodies that represent each robot. Takada et al. [31] utilized parallel embodiment to allow users to simultaneously operate two robot arms to play ping-pong against two opponents. These studies demonstrate the potential of duplicating control and movement in enhancing physical operation.

However, these methods are restricted by real-world hardware resources, leading to limitations in the number of clones or replications. In Clonemator, we systematically explore such a concept in virtual reality environments, allowing users to operate an unlimited number of clones of varying sizes, timing, and replayability, thereby fully realizing the potential for movement cloning and replication.

7.2 Beyond-Real Interaction in VR

VR offers a unique platform for exploring interactions that are impossible to experience in the physical world. Such interactions have been coined as "Beyond-real interaction" [2] by Parastoo et al. Researchers have investigated a variety of beyond-real interactions, including dynamically changing arm length for reaching objects [24], scaling the body for navigation [1], and using spherical proxies to interact with distant objects [22].

Researchers have also begun exploring the possibility of multi-embodiment or duplicating body parts in VR. Studies have shown that users can adapt to the six-digit hand while maintaining a sense of ownership and agency with it [8]. Similar results can also be applied to having a third arm [5] and even multiple bodies [19]. Furthermore, users can develop an implicit dual motor adaptation when switching between two virtual bodies [32].

Beyond the experience of having supernumerary body parts, researchers have also investigated the resulting changes in performance. For example, it has been shown that users can increase task efficiency and reduce physical movement by leveraging additional hands [25] or bodies [26]. Smith et al. have also demonstrated that users can coordinate two bodies simultaneously and develop different strategies for handing off a cube [28]. However, while controlling multiple clones, sharing multiple perspectives (*i.e.* more than two) simultaneously could also increase control complexity, thus adding task completion time [19].

Clonemator is situated within the "duplication" category of Beyond-Real Interaction's taxonomy [2]. Compared to previous work, our research further explores the interaction between the user and their clones, leveraging the ability to create clones with various temporal and spatial properties to give users greater control. This approach builds upon previous research and takes it to the next level by enabling the possibilities for user-clone interactions.

7.3 Programming by Demonstration

Previous research has investigated Programming-by-Demonstration (PbD) [15] to enhance the accessibility of end-user programming. PbD liberates users from the constraints of dedicated programming languages or scripts by enabling them to create their automated solutions by demonstrating "an example of what they wanted it to do" directly to the system [15].

For instance, Sikuli [35] allows users to utilize screen-captured elements as their script and search input, bypassing the need for expert knowledge of the parameters or names of the corresponding UI. Additionally, SikuliBot [11] extends the same concept to physical interfaces by enabling users to take real-world images of physical buttons or touchscreen positions, and a robotic actuator automatically operates these physical UI elements based on the users' arrangement of the images. In addition to images or screenshots, recording and replaying a sequence of users' interactions further enhances the automation process with PbD. For instance, Ringer [3] allows non-coders to record a sequence of their website interaction, which generates a script that interacts with the page as demonstrated by the users. Similarly, Mau'es et al. [18] enables users to perform a smartphone task that they would like to automate, and their system generates the automation based on these latest actions. Moreover, SUGILITE [14] combines voice input

and app demonstration to enhance the automation pipeline for non-programmers.

While these approaches have lowered the barrier to automation for desktop, web, and mobile applications, Clonemator explores this concept in the context of virtual reality. Specifically, Clonemator builds upon the PbD concept for task automation by allowing users to record and replay their VR demonstration, which can be rearranged, distributed, and composited with each other in the form of clones in VR. Users can create and control clones to perform specific actions as if they were controlling their own bodies, without requiring dedicated crafting of different interaction techniques for different situations.

8 DISCUSSION AND FUTURE WORK

8.1 Snapping and Alignment

Composing multiple clones to create custom automators often requires precise alignment of the clones. Clonemator addresses this by offering several alignment techniques, such as *Auto Spawning* and *Relative Spawning*, as well as *Grid Snapping* and *Nearest Object Snapping* modes when spawning a clone.

While these techniques can handle situations in our case, more advanced techniques are necessary for complex scenarios, such as when the user wants to spawn a clone in front of an object that is significantly smaller than the object in front of the user since the required offsets may not be the same. This can lead to misalignment, making it difficult for the clone to interact with the smaller object simultaneously.

8.2 Cloning with objects in hands vs. without

One confusion we observed during the study and our own implementation is whether the object held in hand should be cloned when spawning a user's clone. If not, then interactions such as hammering multiple pegs simultaneously can't be easily achieved. However, if we clone the held objects every time, excessive objects would be produced unnecessarily.

We believe this problem essentially pertains to defining the boundary of a clone's ownership. Since we currently see the need for both alternatives, we suggest that future systems allow users to adjust what they want to include in the cloning and what they don't. While our implementation currently achieves this by setting different object cloning modes for different spawning methods (*i.e.* while auto and relative spawn clone the object in hands, the rest don't), we see a complete decoupling between object cloning and spawning method as a better future option.

8.3 Further Supporting Techniques

With only the key components of Clonemator, the system can possibly play with any interaction techniques and enable a bigger solution space. However, we have identified several supporting interaction techniques closely related to the concept of Clonemator itself, which we believe is not necessary to enable Clonemator but will improve the automation process. We list these interaction features here to inspire future research on the system.

- **Temporal Transformation:** While we have incorporated interaction techniques about spatial transformations (*e.g.* scaling a

clone or group) in our current implementation, the comparable future supporting technique could be temporal editing of clones. This could potentially involve trimming unnecessary replayed segments, extracting replayed clips to create new sequences, and implementing fast-forward or rewind functionalities within a replay.

- **Visualization and Mesh Control:** While we currently present the user avatar and clones with full-body mesh, visualizing only the upper body of the avatar or even just the hands could reduce the visual burden when many automators are working at the same time (e.g. utilizing multiple clones to create a pantograph mechanism for achieving adjusted CD gain in fetching). Additionally, allowing users to change the coloring of clones would make it easier for them to manage them using color labels systematically. Moreover, changing the mesh or texture of a clone could enable a customized form of automators. For instance, when using a string of static clones to build a row of fences, giving them a wooden texture or simplified mesh would be more realistic and aesthetically pleasing.
- **Shortcut Bindings:** In our existing implementation, to reuse a previous recording, users must initially access a carousel menu. To duplicate a group of clones, they need to target an existing group and execute the duplication. While this design sufficiently facilitates automated tasks, an enhanced interaction could enable users to bind the reuse or duplication of specific actions or groups to a button shortcut. This would prove especially convenient for frequently used interaction techniques.

8.4 Merging a Clone's Experience

The Shadow Clone Technique heavily inspires us in the popular anime, Naruto. One of the key features is the user's ability to gain the clones' experiences when they disperse. In Clonemator, we have implemented a similar feature where a memory orb, inspired by the film *Inside Out*, appears in front of the user upon removing a clone. By grabbing the memory orb and placing it into his chest, the user can relive the clone's experience by watching a replay captured by the clone's eye (Figure 14).

However, a question remains: can the user experience a clone's sensations other than sight? For example, imagine removing a clone that has been stirring a pot for half an hour and instantly feeling the soreness in their arm. This presents an interesting design space for future exploration.

8.5 Beyond Problem Solving

Although, for our VR study, we implemented and demonstrated Clonemator with its problem-solving capacity, we believe the work has further potential to be utilized in other areas.

To begin with such extension, we believe Clonemator can serve greatly for social scenarios such as VRChat [33] or BeanVR [4]. With Clonemator, in social platforms, players can compose versatile body motions with single or multiple replayed clones, forming groups of clones with intriguing postures as signs, or deploying multiple dynamic dancing clones as social gestures such as Figure 9. This extension also hints at the potential for Clonemator to function as a creative system where individual users can harness the

cloning operations to generate diverse artistic expressions, including forms that traditionally may require multiple users, such as movies (multiple actors/ actress) and dances (multiple dancers).

Another prospective area is a multi-robot system. While our system is conceptually similar to the "Distributed Autonomous System" [21] that utilizes multiple independent robot units to work coordinately and achieve a more complex task, it has the potential to serve as a control mechanism for future swarm user interfaces [13] or multiple robotic arms. By leveraging the flexibility of Clonemator, users can control multiple elements as intuitively as controlling their clones. For example, when a user spawns a clone, a corresponding robot will appear in the real world at the same position. It can also facilitate users in switching between multiple robots [10] or recording and replaying actions for teleoperation [17].

Besides, previous video games have already leveraged different aspects of clone-based mechanisms to create novel gaming experiences. For example, Quantum League [6] and Time Rifters [9] allow players to team up with their past selves in a shooting game. More recently, The Last Clockwinder [23] enables players to create clones that replay their actions and construct a pipeline to maximize the resource harvesting throughput. While Clonemator systematically extends the possibilities of interactions between clones by providing multiple interaction modes and showcasing how they can be combined with supporting techniques for even more complicated tasks, we believe our work will inspire game designers to explore and develop more engaging and innovative gaming mechanisms.

9 CONCLUSION

We have presented Clonemator, a VR system that allows a user to create and collaborate with clones to accomplish complex tasks. We have demonstrated the potential of Clonemator by showcasing concrete examples based on our systematic exploration of spawning clones, clones' properties, and interactions with traditional techniques. We also have shown from our preliminary study that participants were able to intuitively, creatively, and also enjoyably use Clonemator. With Clonemator, we see the user's avatar could be the generic automator waiting to be dispatched.

ACKNOWLEDGMENTS

REFERENCES

- [1] Parastoo Abtahi, Mar Gonzalez-Franco, Eyal Ofek, and Anthony Steed. 2019. I'm a Giant: Walking in Large Virtual Environments at High Speed Gains. 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/10.1145/3290605.3300752>
- [2] Parastoo Abtahi, Sidney Q. Hough, James A. Landay, and Sean Follmer. 2022. Beyond Being Real: A Sensorimotor Control Perspective on Interactions in Virtual Reality. 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 358, 17 pages. <https://doi.org/10.1145/3491102.3517706>
- [3] Shaon Barman, Sarah Chasins, Rastislav Bodik, and Sumit Gulwani. 2016. Ringer: Web Automation by Demonstration. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Amsterdam, Netherlands) (OOPSLA 2016). Association for Computing Machinery, New York, NY, USA, 748–764. <https://doi.org/10.1145/2983990.2984020>
- [4] BeanVR. 2017. BeanVR. https://store.steampowered.com/app/638920/BeanVRThe_Social_VR_APP/
- [5] Adam Drogemuller, Adrien Verhulst, Benjamin Volmer, Bruce H. Thomas, Masahiko Inami, and Maki Sugimoto. 2019. Remapping a Third Arm in Virtual

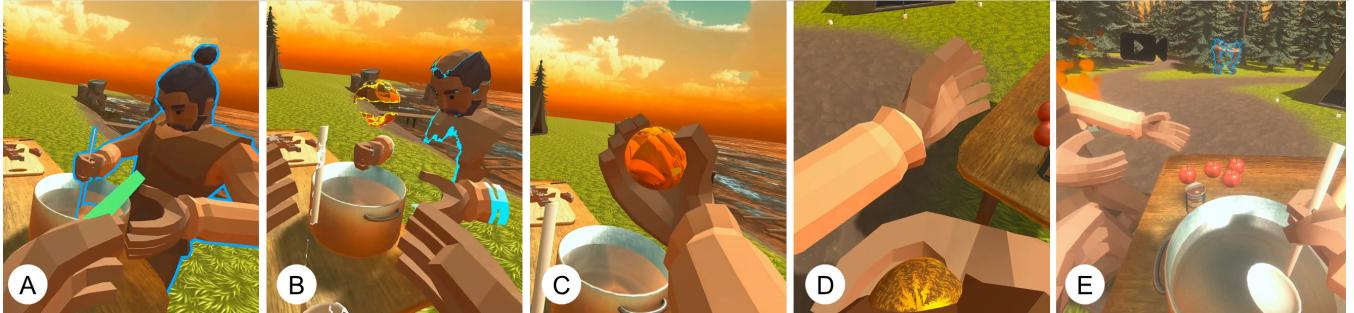


Figure 14: A proof-of-concept feature allowing users to relive a clone's experience: (A) The user removes a clone and (B)-(C) a memory orb appears. (D) The user places the memory orb into his chest and (E) experiences the clone's memory.

- Reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 898–899. <https://doi.org/10.1109/VR.2019.8798197>
- [6] Nimble Giant Entertainment. 2021. Quantum League. https://store.steampowered.com/app/651150/Quantum_League/.
- [7] Dylan F. Glas, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. 2012. Teleoperation of Multiple Social Robots. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42, 3 (2012), 530–544. <https://doi.org/10.1109/TSMCA.2011.2164243>
- [8] Ludovic Hoyet, Ferran Argelaguet, Corentin Nicole, and Anatole Lécuyer. 2016. “Wow! I Have Six Fingers!”: Would You Accept Structural Changes of Your Hand in VR? *Frontiers in Robotics and AI* 3 (2016). <https://doi.org/10.3389/frobt.2016.00027>
- [9] Proton Studio Inc. 2014. Time Rifters. https://store.steampowered.com/app/270010/Time_Rifters/.
- [10] Atsushi Izumihara, Tomoya Sasaki, Masahiro Ogino, Reona Takamura, and Masahiko Inami. 2019. Transfomation: Transformation into Bodies of Various Scale and Structure in Multiple Spaces. In *ACM SIGGRAPH 2019 Emerging Technologies* (Los Angeles, California) (*SIGGRAPH ’19*). Association for Computing Machinery, New York, NY, USA, Article 27, 2 pages. <https://doi.org/10.1145/3305367.3327980>
- [11] Jeeeon Kim, Mike Kasper, Tom Yeh, and Nikolaus Correll. 2014. SikuliBot: Automating Physical Interface Using Images. In *Adjunct Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) (*UIST ’14 Adjunct*). Association for Computing Machinery, New York, NY, USA, 53–54. <https://doi.org/10.1145/2658779.2659110>
- [12] Ioannis Kotziapasis, Nathan Sidwell, and Alan Chalmers. 2003. Portals: Increasing Visibility in Virtual Worlds. In *Proceedings of the 19th Spring Conference on Computer Graphics* (Budmerice, Slovakia) (*SCCG ’03*). Association for Computing Machinery, New York, NY, USA, 257–261. <https://doi.org/10.1145/984952.984995>
- [13] Mathieu Le Goc, Lawrence H. Kim, Ali Parsaei, Jean-Daniel Fekete, Pierre Dragicevic, and Sean Follmer. 2016. Zoids: 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>
- [14] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (*CHI ’17*). Association for Computing Machinery, New York, NY, USA, 6038–6049. <https://doi.org/10.1145/3025453.3025483>
- [15] Henry Lieberman. 2001. *Your wish is my command: Programming by example*. Morgan Kaufmann.
- [16] Paul Lubos, Gerd Bruder, and Frank Steinicke. 2014. Are 4 Hands Better than 2? Bimanual Interaction for Quadmanual User Interfaces. In *Proceedings of the 2nd ACM Symposium on Spatial User Interaction* (Honolulu, Hawaii, USA) (*SUI ’14*). Association for Computing Machinery, New York, NY, USA, 123–126. <https://doi.org/10.1145/2659766.2659782>
- [17] Karthik Mahadevan, Yan Chen, Maya Cakmak, Anthony Tang, and Tovi Grossman. 2022. Mimic: In-Situ Recording and Re-Use of Demonstrations to Support Robot Teleoperation. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) (*UIST ’22*). Association for Computing Machinery, New York, NY, USA, Article 40, 13 pages. <https://doi.org/10.1145/3526113.3545639>
- [18] Rodrigo de A. Maués and Simone Diniz Junqueira Barbosa. 2013. Keep Doing What iJust Did: Automating Smartphones by Demonstration. In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Munich, Germany) (*MobileHCI ’13*). Association for Computing Machinery, New York, NY, USA, 295–303. <https://doi.org/10.1145/2493190.2493216>
- [19] Reiji Miura, Shunichi Kasahara, Michiteru Kitazaki, Adrien Verhulst, Masahiko Inami, and Maki Sugimoto. 2021. MultiSoma: Distributed Embodiment with Synchronized Behavior and Perception. In *Proceedings of the Augmented Humans International Conference 2021* (Rovaniemi, Finland) (*AHs ’21*). Association for Computing Machinery, New York, NY, USA, 1–9. <https://doi.org/10.1145/3458709.3458878>
- [20] Yukiya Nakanishi, Masaaki Fukuoka, Shunichi Kasahara, and Maki Sugimoto. 2022. Synchronous and Asynchronous Manipulation Switching of Multiple Robotic Embodiment Using EMG and Eye Gaze. In *Proceedings of the Augmented Humans International Conference 2022* (Kashiwa, Chiba, Japan) (*AHs ’22*). Association for Computing Machinery, New York, NY, USA, 94–103. <https://doi.org/10.1145/3519391.3522753>
- [21] Jun Ota. 2006. Multi-agent robot systems as distributed autonomous systems. *Advanced Engineering Informatics* 20, 1 (2006), 59–70. <https://doi.org/10.1016/j.aei.2005.06.002>
- [22] Henning Pohl, Klemen Lilija, Jess McIntosh, and Kasper Hornbæk. 2021. Poros: Configurable Proxies for Distant Interactions in VR. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI ’21*). Association for Computing Machinery, New York, NY, USA, Article 532, 12 pages. <https://doi.org/10.1145/3411764.3445685>
- [23] Pontoco. 2022. The Last Clockwinder. <https://www.oculus.com/experiences/quest/4837365566303714>
- [24] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology* (Seattle, Washington, USA) (*UIST ’96*). Association for Computing Machinery, New York, NY, USA, 79–80. <https://doi.org/10.1145/237091.237102>
- [25] Jonas Schjerlund, Kasper Hornbæk, and Joanna Bergström. 2021. Ninja Hands: Using Many Hands to Improve Target Selection in VR. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) (*CHI ’21*). Association for Computing Machinery, New York, NY, USA, Article 130, 14 pages. <https://doi.org/10.1145/3411764.3445759>
- [26] Jonas Schjerlund, Kasper Hornbæk, and Joanna Bergström. 2022. OVRLap: Perceiving Multiple Locations Simultaneously to Improve Interaction in VR. 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 355, 13 pages. <https://doi.org/10.1145/3491102.3501873>
- [27] Anne Schlottmann. 1999. Seeing it happen and knowing how it works: how children understand the relation between perceptual causality and underlying mechanism. *Developmental psychology* 35, 1 (1999), 303.
- [28] James Smith, Xinyun Cao, Adolfo G. Ramirez-Aristizabal, and Bjoern Hartmann. 2023. Dual Body Bimanual Coordination in Immersive Environments. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) (*DIS ’23*). Association for Computing Machinery, New York, NY, USA, 230–243. <https://doi.org/10.1145/3563657.3596082>
- [29] Richard Stoakley, Matthew J. Conway, and Randy Pausch. 1995. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (*CHI ’95*). ACM Press/Addison-Wesley Publishing Co., USA, 265–272. <https://doi.org/10.1145/223904.223938>
- [30] Benjamin Straube and Anjan Chatterjee. 2010. Space and time in perceptual causality. *Frontiers in Human Neuroscience* 4 (2010). <https://doi.org/10.3389/fnhum.2010.00028>
- [31] Kazuma Takada, Midori Kawaguchi, Akira Uehara, Yukiya Nakanishi, Mark Armstrong, Adrien Verhulst, Kouta Minamizawa, and Shunichi Kasahara. 2022. Parallel Ping-Pong: Exploring Parallel Embodiment through Multiple Bodies by a Single User. In *Proceedings of the Augmented Humans International Conference*

- 2022 (Kashiwa, Chiba, Japan) (*AHs '22*). Association for Computing Machinery, New York, NY, USA, 121–130. <https://doi.org/10.1145/3519391.3519408>
- [32] Adrien Verhulst, Yasuko Namikawa, and Shunlchl Kasahara. 2022. Parallel Adaptation: Switching between Two Virtual Bodies with Different Perspectives Enables Dual Motor Adaptation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 169–177. <https://doi.org/10.1109/ISMAR55827.2022.00031>
- [33] VRChat. 2014. VRChat. <https://hello.vrchat.com/>
- [34] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. 2018. Space-time: Enabling Fluid Individual and Collaborative Editing in Virtual Reality. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (Berlin, Germany) (UIST '18)*. Association for Computing Machinery, New York, NY, USA, 853–866. <https://doi.org/10.1145/3242587.3242597>
- [35] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (Victoria, BC, Canada) (UIST '09)*. Association for Computing Machinery, New York, NY, USA, 183–192. <https://doi.org/10.1145/1622176.1622213>