frontiers in ICT

ORIGINAL RESEARCH
published: 28 September 2018
doi: 10.3389/fict.2018.00025

Force Push: Exploring Expressive Gesture-to-Force Mappings for Remote Object Manipulation in Virtual Reality

Run Yu[1,2]* *and Doug A. Bowman*[1,2]

[1] *Center for Human-Computer Interaction, Virginia Tech, Blacksburg, VA, United States,* [2] *Department of Computer Science, Virginia Tech, Blacksburg, VA, United States*

This paper presents Force Push, a novel gesture-based interaction technique for remote object manipulation in virtual reality (VR). Inspired by the design of magic powers in popular culture, Force Push uses intuitive hand gestures to drive physics-based movement of the object. Using a novel algorithm that dynamically maps rich features of hand gestures to the properties of the physics simulation, both coarse-grained ballistic movements and fine-grained refinement movements can be achieved seamlessly and naturally. An initial user study of a limited translation task showed that, although its gesture-to-force mapping is inherently harder to control than traditional position-to-position mappings, Force Push is usable even for extremely difficult tasks. Direct position-to-position control outperformed Force Push when the initial distance between the object and the target was close relative to the required accuracy; however, the gesture-based method began to show promising results when they were far away from each other. As for subjective user experience, Force Push was perceived as more natural and fun to use, even though its controllability and accuracy were thought to be inferior to direct control. This paper expands the design space of object manipulation beyond mimicking reality, and provides hints on using magical gestures and physics-based techniques for higher usability and hedonic qualities in user experience.

Keywords: hand gesture, object manipulation, transfer function, physics-based manipulation, controllability, virtual reality

OPEN ACCESS

Edited by:
Anthony Steed,
University College London,
United Kingdom

Reviewed by:
Alexander Kulik,
Bauhaus-Universität Weimar,
Germany
Géry Casiez,
Lille University of Science and
Technology, France

****Correspondence:***
Run Yu
runyu@vt.edu

Specialty section:
This article was submitted to
Virtual Environments,
a section of the journal
Frontiers in ICT

Received: *22 May 2018*
Accepted: *04 September 2018*
Published: *28 September 2018*

Citation:
Yu R and Bowman DA (2018) Force Push: Exploring Expressive Gesture-to-Force Mappings for Remote Object Manipulation in Virtual Reality. Front. ICT 5:25.
doi: 10.3389/fict.2018.00025

INTRODUCTION

Many people fantasize about remotely manipulating real-world objects, as we can see from the frequent appearance of "telekinesis" as a magic power in popular media. In virtual reality (VR), however, remote manipulation is commonplace and even necessary for some applications (Bowman et al., 2004; LaViola et al., 2017).

Existing VR interfaces for remote manipulation typically use a "simple virtual hand" metaphor, which is simply a direct extension of how we grab and move objects in the real world. The user first grabs the object using a remote selection technique (e.g., ray-casting or arm extension), then the movement of the object follows the movement of the hand using a zero-order, positional mapping between the two (Bowman and Hodges, 1997; Zhai, 1998; Bowman et al., 2004). The rationale behind this design pattern is simple: previous literature consistently shows that a zero-order mapping provides superior performance and a more intuitive user experience compared to mappings, such as rate- or acceleration-based control (Zhai, 1998). This is not only due to the fact that lower-order mappings are inherently more controllable, but also because the drag-and-drop metaphor is relatable to how we manipulate objects in the real-world and in 2D user interfaces.

Although this technique provides excellent usability, it is not perfect given the specific requirements of VR interaction. Many VR experiences intend to create a strong sense of presence, which requires that the objects in the virtual environment behave in a plausible and realistic manner (Slater, 2009). When the object's behavior is directly driven by a zero-order mapping from the hand's movement, its physically based properties are (at least partially) taken away. Its movement does not have realistic acceleration or deceleration, but instead is sudden and abrupt, as if the object has negligible weight. Such behaviors might still be reasonable if the object is being held in the hand. However, they are less appropriate in the context of remote manipulation where there is no physical contact between the hand and the object. Moreover, it becomes difficult to define the object's interaction with other virtual objects in the scene. If the object is still affected by collisions, it might be detached from the hand when bumping into other objects—a behavior that contradicts the interaction mapping. If it stays in the hand, it might penetrate into other objects or knock over a much heavier object, both of which contradict real-world phenomena.

A possible solution to this is to drive the object's behaviors using physics-based simulation. With this approach, the object is moved by force and the corresponding acceleration, so it can exhibit realistic motion and correctly collide with the environment. In fact, this is usually how an object moves when telekinesis is depicted in popular culture in order to make the scene look believable. Following this observation, we were motivated to use a completely physics-based model. The particular super power that inspired us is the "Force Push" from the *Star Wars* movie franchise. Jedi masters can remotely move objects using the Force: gestural input from the hand (usually either pushing or pulling) applies a certain amount of force to a remotely located object, which throws it in the direction indicated by the gesture using a physics-based process. If a strong force is exerted, a single Force Push can easily move a heavy object by a large distance. This sort of gesture-based interaction seems to be a powerful and fun way of influencing the physical world, while still being understandable and plausible due to its coherent (though non-realistic) physical laws. With this research, we intended to design such a novel interaction technique that moves the object by a physics-based simulation while connecting its movement to hand input in an intuitive way.

Although we were motivated by improving subjective user experience of the interaction, the first and foremost task is to make the technique *controllable*. Using a physics-based model implies that objects are moved by force, which could be inherently more difficult than using a zero-order mapping. Moreover, the user needs to precisely control the force exerted on the object through gestural input, which requires a novel transfer function that accurately interprets the user's intention through hand gestures. In this paper, we focus on developing a novel gesture-to-force mapping algorithm that makes this technique usable even in extremely difficult cases, and present an initial experiment as a first step in understanding its usability and subjective user experience.

Our research addresses the following two questions:

(1) How can we design a gesture-to-force mapping technique that provides gesture-based, physics-driven remote manipulation and makes it controllable in VR?
(2) Does this technique lead to a superior user experience compared to a direct position-to-position mapping?

To address question 1, we designed Force Push, a gesture-based interaction for physics-driven remote object manipulation. We present the iterative design process we used to make the technique intuitive and controllable. To begin to address question 2, we ran an initial user study that compares Force Push to a more conventional direct mapping. The study focused on task performance and controllability, but also measured some subjective aspects of user experience.

The contributions of this paper are:

- The concept of using the rich features of dynamic gestures to allow users to naturally control complex object movements, which led to a novel, controllable gesture-to-force mapping algorithm for remote object manipulation.
- Empirical results of an initial study demonstrating both the performance and broader user experience qualities of Force Push as compared to a traditional direct control mapping.

RELATED WORK

Gesture-Based Interaction

Gesture-based interaction has drawn increasing attention in recent years, as it is an essential part of the emerging "natural user interface" paradigm (Bowman et al., 2004; Wigdor and Wixon, 2011; LaViola et al., 2017). Many have tried to find the best set of gestures for a certain task by understanding human preference through user studies. One such example is the classification system for mid-air gestures by Aigner et al. in which they analyzed human-to-human gestures in terms of hand usage and gesture type (Aigner et al., 2012). Norton et al. investigated design strategies for full body gestures in video games by studying human preference when given complete freedom of choosing gestures (Norton et al., 2010). In fact, there is a whole family of user studies called "gesture elicitation" that tries to derive a suitable gesture set from the users themselves (Ortega et al., 2017). Overall, these efforts usually focus on finding the best mapping from a group of gestures to a group of actions required by the task. Instead, we take the approach of identifying gestures used in the real-world and carefully designing a usable and understandable mapping based on their expressive properties.

van Beurden et al. compared the pragmatic and hedonic qualities between gesture-based and traditional interfaces (van Beurden et al., 2011). They found that "more embodied interaction reveals higher scores in terms of hedonic quality and fun than mouse-based interaction." On the other hand, more embodied interaction could result in more body fatigue. This finding implies that gesture-based interaction may have some inherent qualities that lead to superior user experience on aspects other than performance alone.

Transfer Functions for Object Manipulation

Card et al. presented a framework for analyzing input devices, in which "expressiveness" was used to measure the capability of input-to-output transfer function (Card et al., 1990). The general consensus is that lower-order mappings between hand and object provide better user experience and performance compared to indirect mappings such as rate control (Zhai, 1998; Hinckley et al., 2004). The most common technique for 3D object manipulation in VR is based on the "simple virtual hand" metaphor, in which the entire hand is treated as one rigid body (usually through a hand-held controller) and the object is directly attached to it (Poupyrev et al., 1996). With the addition of a remote selection technique such as ray-casting or arm extension, this technique can be extended to break the isomorphic mapping and enable remote control (e.g., Poupyrev et al., 1996, 2000; Bowman and Hodges, 1997). However, such techniques are still based on a zero-order mapping between the relative movements of the hand and the object, and few existing techniques focus on the effect of using physics-based movement for remote manipulation in VR.

In the context of bare-hand gesture-based manipulation in VR, previous research has focused on advancing tracking technologies and finding "natural" gestures to control objects. In many cases, the gestures are only used as a binary on/off form to activate the control, while the manipulation itself is still based on the simple virtual hand metaphor (e.g., Segen and Kumar, 1998; O'Hagan and Zelinsky, 2000; Sato et al., 2001). Alternative mappings between the movements of the hand and the object do exist in 3D UI. In the bimanual interface presented by Levesque et al., the object was attached to a ray cast from the user's right hand, and "the distance from the hand to the object remains constant throughout the whole interaction" (Lévesque et al., 2011). Benko et al. designed a "DepthTouch" interface that was placed above an interactive surface, where translations of the hand along certain dimensions were mapped to the rotational angle of the object (Benko and Wilson, 2008). Similarly, Song et al. used a "handle bar" metaphor, where circular movements of the hand could be mapped to the angle of rotation of the object (Song et al., 2012). Notice that even though these cases differ from the naïve direct position-to-position mapping, the transfer function is still zero-order and none of these techniques emphasizes on utilizing physics-driven movement.

On the other hand, previous research in 2D UI has put particular focus on using physics-based actions to enhance usability. Specifically, the interaction of using a flick gesture to "throw" an object to far-away locations has been widely adopted to move an icon beyond arm's reach (Geißler, 1998). Moyle et al. analyzed the physical properties of the flick gesture itself when using mouse and pen input (Moyle and Cockburn, 2002). Aliakseyeu et al. evaluated the effectiveness of the flick interaction for scrolling through documents with pen interfaces (Aliakseyeu et al., 2008). Reetz et al. proposed the "Super Flick" technique that adds an optional correction phase to the simple flick interaction to improve its accuracy (Reetz et al., 2006). Similarly, techniques such as "drag-and-throw" and "push-and-throw" were introduced to help the user accurately define the trajectory of throwing (Collomb et al., 2005; Hascoët, 2008). Compared to its application in 2D UI, physics-driven movement may be especially desirable in VR, since realism and plausibility is a first priority in many VR experiences. Thus, although physics-based control has been researched in 2D UI, investigating its application to VR could still provide useful knowledge in this specific context.

To overcome the anatomical limits of the human hand such as hand instability and limited range of movement, enhancements of direct mappings have been proposed. Kopper et al. let the user toggle between two separate Control/Display ratios (C/D ratios) in order to achieve both ballistic and precise control, which is very similar to the direct control interface employed in our study (Kopper, 2011). Frees et al. proposed a solution called PRISM that dynamically adjusted the C/D ratio by interpreting the user's intention from the hand's movement speed, which is similar to our approach that uses the features of user input to adjust the property of output action (Frees et al., 2007).

DESIGNING FORCE PUSH

Metaphors

Inspired by the magic power of telekinesis in popular culture, such as the *Star Wars* Force Push, we set out to design a remote manipulation technique in VR that was based on hand gestures and a gesture-to-force mapping. In designing the details of the mapping, however, we found additional metaphors with roots in reality.

People often use a hand waving gesture to indicate desired movement to others at a distance. The path of the hand motion along with the orientation of the palm typically indicates the intended movement direction, which creates easy-to-understand commands such as "come closer," "go further," or "move more to the left." Think about the case of communicating with a person parking a vehicle: people naturally wave their hands to express "straight back quickly," "turn slightly this direction," or "slow down." This set of simple gestures seen in human-human interaction are not based on direct movement specification using a positional mapping. At the same time, they are commonly seen and widely accepted, which makes them natural and easy to understand.

Another metaphor based on physical phenomena may be at play in such gestures. The user may think of this as pushing the air with hand motions, so that the airflow moves the target object. One may also think of these gestures as "nudging" actions applied remotely to the target object, similar to the motions used to tweak the position or orientation of an object sitting on a table or floor—rather than picking up the object and placing it down again, the user simply pushes the object with a finger or hand. In both scenarios, the user's movements are being translated into force, rather than positions.

The high-level design of Force Push follows these inspirations and metaphors: with the hand in an open posture (all fingers stretched out), a quick burst of hand movement along the direction that is perpendicular to the palm would translate the object in the same direction. **Figures 1A,B** shows this hand gesture when it's moving the object on the vertical dimension.

Following the same idea of utilizing intuitive gestures, we also implemented a gesture for rotation: a circle drawn by one finger around a certain axis would rotate the object in that direction by applying torque, as shown in **Figure 1C**. In the remainder of this paper, however, we focus only on the translation part of the Force Push interface.

Gesture Detection

In order to formalize the Force Push technique, several important questions needed to be answered:

- How to detect the gesture
- How to determine the exact direction of translation
- How to determine the quantity of translation (distance, speed)

In answering the first two questions, a naïve design would allow "pushing" at an arbitrary direction: Any time a hand moves along its palm's normal, the gesture would be activated and start translating the object along that direction. This naïve interpretation would create too many false positive cases for the gesture recognition algorithm, as we often accidentally move a hand along the direction perpendicular to the palm without the intention of activating the gesture. Moreover, we found that people tended to be inaccurate when defining a direction through either a trajectory of hand movement or the palm's orientation in our early tests. To overcome these issues, we added some constraints to the possible translation directions that made it easier to control:

- Limit the gesture's moving directions to be along the left/right and forward/back axes of the camera's coordinate system, plus the up/down axis of the world coordinate system. From the user's perspective, this limited the choice of directions in translating the object to six directions that were aligned with his egocentric point of view.
- Among these six options, the one that was closest to the direction that the palm was facing (normal of the palm) was chosen. This effectively "snapped" the hand gesture direction to the closest option that was aligned with the egocentric view.

Figure 2 shows how the user controls the translation direction of the object using hand orientation and hand choice (the right hand is used to make the left translation gesture, and vice-versa). With the constraints added to gestural movement directions, a simple gesture recognition algorithm was implemented: a ring buffer was built to record the most recent 10 frames of hand motion; it pushes in a new frame's data every 0.1 s (a sliding window of 1 s). The gesture detection algorithm continuously analyzes the recorded hand motion and determines whether to activate the gestural interaction based on two criteria:

- We first examine whether the palm is generally facing one of the six directions aligned with the egocentric view. This is done by examining the similarity between the palm's normal and these directions. In every frame, we calculate the Euclidean distance between the value of the palm's normal and the value of the normalized vector pointing to each of these six directions. We sum this distance over the past 10 frames and see if it's smaller than a certain threshold. This threshold was empirically determined to be 8.0 in the Unity engine.
- If the above criterion is met, we inspect whether the fingers' motion exhibits a burst of movement along that direction. To do this, we examine the similarity between a long vector pointing to this direction and the fingers' relative translation. Specifically, we calculate the Euclidean distance between the value of a normalized vector (with a length of 1 m in Unity) pointing in that direction and the value of each finger tip's translation in every frame. We sum this distance over all five fingers and over the past 10 frames to see if it's larger than a threshold. This threshold was empirically determined to be 49.5 in the Unity engine.

Physics-Driven Control

Having determined the gesture detection algorithm and the direction of object translation, next we needed to decide how to map hand movement to the movement of the object. In an earlier gesture-based manipulation prototype, we actually started with a position-to-position mapping: the travel distance of the hand along the palm's normal direction was applied to move the object by the same distance. However, we found that this naïve mapping created a jerky movement for the object. As the hand traveled forward, for example, the object would be translating forward at the same pace, while as the hand traveled backward to reset

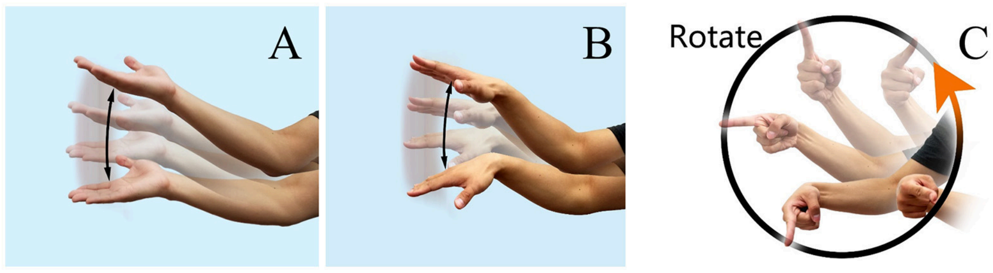

FIGURE 1 | The Force Push gestures. **(A)** Translation that moves the object up. **(B)** Translation that moves the object down. **(C)** Rotation.

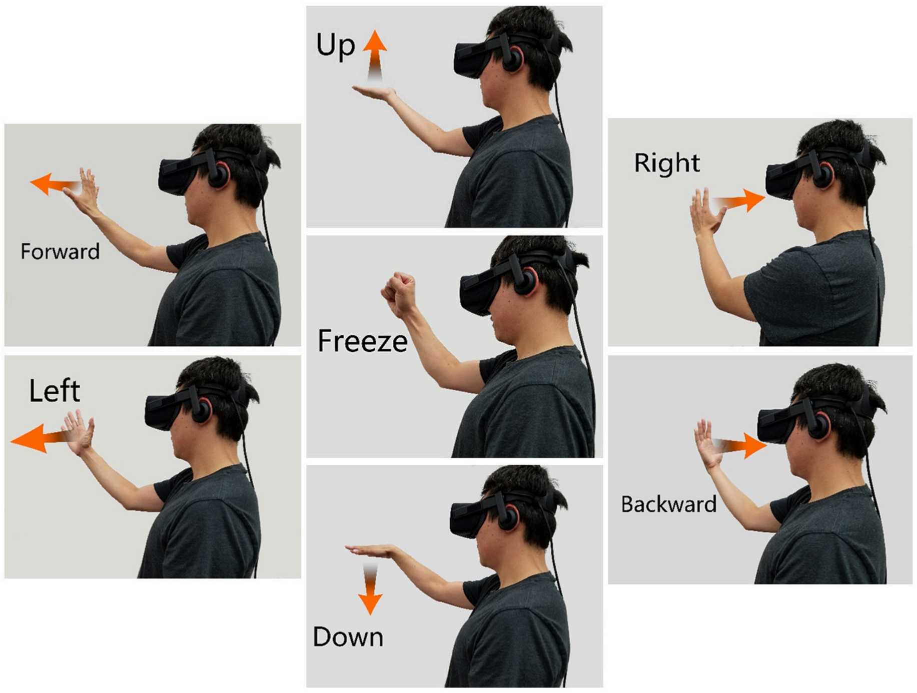

FIGURE 2 | The direction the palm is facing defines the translation direction of the manipulation. The "freeze" gesture stops the object from moving (The depicted individual is one of the authors of this paper and these images are published with his consent).

its position, the object would stop moving. This was considered undesirable and hard to control.

To improve the interaction experience, we drew upon the force-based metaphors described in section Metaphors and applied a constant amount of force to the object, driving its movement using a physics-based simulation every time the gesture was activated. This seemingly simple improvement was actually a fundamental change to the interaction design: the interface was changed from position-based direct mapping to acceleration-based indirect control.

Since this manipulation interface was driven by physics, inertia would keep the object moving for a while even when the gesture stopped. This made it hard to control since one could not stop the movement when the object reached the target location. Here we applied two improvements. The first one was adding resistance. In the Unity3D game engine, this was done by setting a positive value to the "drag" parameter of the rigid body. With resistance, the object would gradually slow down and stop its movement after the gesture stopped. We also implemented another static posture called "freeze" that would stop the object immediately from moving further: whenever the user closed his hand to make a fist (as shown in the center of **Figure 2**), the object would instantly stop. We found that the addition of the freeze gesture greatly enhanced the controllability since the user could stop the object at a desired location any time.

We let a number of users try out this prototype and collected informal feedback. No one had any difficulty in understanding the interface and control. Most people felt the interaction was fun and magical, as it gave them a "super power" that was similar to what they see in popular media. Some users also told us the physics-based smooth movement added to the user experience as it made the object feel more realistic. All this early informal feedback indicated that Force Push might provide desirable hedonic qualities compared to traditional direct control techniques. However, we observed that users still had trouble moving objects accurately to the desired location.

Expressive Gesture-to-Force Mapping

Previous research told us that acceleration-based indirect manipulation provides significantly worse controllability than position-based direct manipulation (Zhai, 1998), and Force Push was no exception. To make things worse, the technique could only apply a constant amount of force once activated. Without means to control how fast the object moved, one had to rely heavily on the freeze gesture to try to stop it at the right time. In order to provide a good balance between hedonic qualities and task performance, we needed a more effective method to define the quantity of translation.

Real-world object manipulation tasks may pose different requirements. The two main factors here are the initial distance between the object and the target location, and the required accuracy (i.e., how close does the object need to be to the target for the task to be considered completed). According to Fitts's law, the difficulty of an aimed movement task is determined by the amplitude (distance) and the target width (required accuracy). In certain difficult situations, even a direct, position-based manipulation can become cumbersome. If the target is far away, the task demands a small C/D ratio (i.e., a small movement of hand mapped to a large movement of the object). Otherwise, the user will have to repetitively clutch (grasp, move, and release) the object, which is time consuming and fatiguing. On the other hand, when the required accuracy is high, one needs a large C/D ratio, since hand instability makes it hard to align the object with the target accurately. In fact, these two factors pose requirements that contradict each other. Thus, it is difficult to design an efficient, direct control interface to align an object to a far-away target with very high accuracy. Realizing this, many researchers have designed techniques that attempt to sidestep the requirements of rapid movements across distance, precise fine-grained control, or both (e.g., Wingrave et al., 2002; Frees et al., 2007).

We observed that there are some inherent qualities of acceleration-driven control that may help in these difficult scenarios. When the target is far away and coarse-grained movement is needed, inertia may enhance our ability to quickly move the object by a large distance with less user input. With the help of physics and inertia, one can "throw" the object toward the target by applying a short burst of force that results in fast, lasting movement without further user interference. When placing the object with a high accuracy requirement, mapping hand movement to acceleration instead of the positional change of the object serves as a low-pass filter—hand tremor does not directly result in shaky movement of the object (Zhai, 1998). These qualities of a force-based mapping could help us improve the performance of Force Push.

In order to meet both the requirements for coarse-grained movement and fine-grained alignment, the object's physics-based movement needed to be adaptive to these two ends of the spectrum. Moreover, the adaptation must happen based on the user's intention. Fortunately, we found that users tended to naturally express their intention into the dynamic gesture of Force Push itself. When they wanted to move the object by a large distance, they tended to extend the range of their arm/hand movement and increase the speed of the repetitive push gesture. When they hoped to manipulate the object precisely, they often resorted to small-ranged finger movement, with a slower pace of gestural input.

This observation told us that a gestural input carried much more information than the simple binary message of activating the action or not. Instead, it contained quantitative information that directly reflected the user's intention. Thus, we could alter the parameters of the interaction in order to meet the user's need, and this adaptation could happen seamlessly during the interaction. Specifically, we could alter the properties of the physics simulation to help the object move faster when we detected that the user preferred coarse-grained, ballistic movement, or we could make the object move slower and more accurately when we detected that the user needed precise control instead.

Conceptually, this is very similar to the PRISM technique, where the C/D ratio of the direct positional mapping is changed on the fly based on the user's hand movement speed (Frees et al., 2007). The difference here lies in the mapping between the input's feature and the output's property. In PRISM, the positional information of input is already fully dedicated to the direct control itself and cannot carry more information. To interpret the user's intention, the technique needs to search in the first derivative space, namely the speed. In the Force Push gesture, however, because we are using a gesture-to-force mapping instead of a position-to-position mapping, the positional information of the hand becomes available to carry the user's preference for the interaction instead of the control itself. Furthermore, since this dynamic gesture is a sequence of movements, its temporal characteristics expand the expressiveness of the feature. In a way, our gesture-to-force design facilitates a richer, more expressive mapping from the user's intention to the properties of interaction.

Assuming the mass of the object to be constant, there were two parameters we could change in the physics engine of Unity3D that would alter how the object moved: the amount of force applied and the physical resistance (drag) of the object. The general idea was simple:

- When the algorithm interpreted that the user wanted to move the object quickly (large range of hand movement with high travel speed), we would increase the amount of force and decrease the drag. In this way, a large amount of acceleration would drive the object to quickly travel a long distance without much resistance.
- When it appeared that the user wanted fine-grained control (small range of hand movement with low travel speed), we would decrease the amount of force and increase the drag. This way the object would move slowly with a small amount of acceleration and the large resistance would stop it quickly when the gesture stopped.

Because of the purely artificial nature of this gesture-to-force mapping, the actual mathematical function to dynamically change these parameters was not based on a real-world model. This implied that we had complete freedom to design it arbitrarily. We developed the equation by trying a variety of different functions and empirically choosing the best one.

We designed the following function to change the amount of force applied in real time when the "pushing" gesture has been recognized. The purpose of this design is to scale the force using both the amplitude of the hand gesture and the hand's travel speed.

$$F = \alpha \times R^2 / (N + 1) \tag{1}$$

F (in newtons) is the resulting amount of force; α is an empirically derived parameter to scale the value; R (in m) is the total distance that the hand has traveled along the axis of object translation (the axis along which the "pushing" gesture is performed) within a constant amount of time, as an indication of how much the hand has moved in that direction. To calculate this value, we sum up all five fingertips' forward and backward translation distances along that axis in each 0.1 s interval (1 frame in the gesture recognition algorithm, as mentioned in section Gesture Detection), and accumulate this value over the last 1 s (10 frames, same sliding time window in gesture recognition). N is how many times the fingertips have switched travel direction along the dimension of object translation during the same time period. To calculate this value, we look at the travel directions of each fingertip during the latest frame and the previous frame (again, each frame lasts for 0.1 s). If these directions are opposite to each other along that axis of object translation, the finger has switched travel direction once. We sum up how many times this kind of switch happens for all five fingertips in the last 1 s (10 frames) as the value of N. To avoid division by zero, we add one to N to serve as the divisor.

This function takes both the travel speed and the amplitude of the "pushing" gesture into consideration. For example, when the hand travels at the same speed but increases the range of movement, R will remain the same while N becomes smaller in the fixed amount of time, which leads to an increased amount of force. On the other hand, if the gesture maintains the same amplitude but the hand travels at a faster speed, R and N will increase at the same rate, which also leads to a stronger force since R is squared.

The implementation directly uses the tracking data given by the Leap Motion API without performing any filtering to account for hand tremor. Although filtering might be more ideal, we found that the lack of filtering does not affect performance much since hand tremor typically results in small travel distances, which will always result in small values using (1).

For the drag parameter, we applied a linear function that scaled the value based on the result of (1). When the amount of force increased, it would decrease the drag value; when the force was weak, it scaled up the drag. This function is shown in Equation (2), in which D is the value for the drag parameter; F is the real-time result of (1); and a and b are two empirically derived constants that linearly scale the amount of drag based on F.

$$D = a \times F + b \tag{2}$$

Both (1) and (2) are empirically designed functions to achieve our goals, and there are many other possible ways to define a mapping from gesture to force and to set the parameters of the physics simulation. Further research is needed here to gain a better understanding of this design space. Through empirical testing, we found that the mapping and parameters described here provide a good balance of expressiveness, control, comfort, and naturalness.

EXPERIMENT

Goal and Hypothesis

To gain an initial understanding of the user experience of the Force Push approach, we designed an experiment that compared it with conventional direct control manipulation using the simple virtual hand metaphor. The evaluation included both performance and subjective experience measures.

While we did not expect Force Push to outperform a zero-order mapping, we were interested in whether this method was usable for difficult placement tasks, and whether the desirable qualities of the dynamic force mapping might have performance benefits in some cases. We also wanted to gather some initial data on subjective aspects of user experience.

We proposed these hypotheses:

- As long as the direct control was equipped with an appropriate choice of C/D ratios, it would typically outperform the Force Push interface.
- With Force Push, we expected there to be a few cases where participants could not finish the task in a reasonable amount of time due to the difficulty of learning and controlling the acceleration-based mapping.
- In cases where even two C/D ratios were not enough (e.g., requiring high accuracy while the target was far-away), Force Push might provide performance superior to direct control.
- Beyond performance, we expected that users would perceive Force Push as more natural and fun than direct control.

In the rest of the paper, the phrase "direct control," "direct mapping," or "simple virtual hand" is used to describe an interaction technique with a position-to-position mapping between the movement of the hand and the object using one or more fixed C/D ratios. The name "Force Push" or "gesture-to-force mapping" is used to describe our novel technique that moves the object using force-driven physics simulation while adaptively changing its property using features extracted from the dynamic gesture.

Experimental Design

Tasks

Note that there are two factors that differentiate the control mechanisms of direct control and Force Push:

(1) Integrated degrees of freedom (DOF) vs. separated DOF: Direct control integrates all six degrees of freedom (DOF) together so that the user can translate/rotate the object in an arbitrary direction at any time. On the contrary, Force Push separates the six DOFs of manipulation, so that users can only translate or rotate the object along one of the limited, pre-defined directions at a time, as mentioned in section Gesture Detection.

(2) Two different mapping functions that determine the quantity of the object's movement: Direct control uses a zero-order mapping that maps the distance the hand travels (or the angle it rotates) to the distance the object moves (or the angle it rotates). Force Push uses the novel gesture-to-force mapping function that maps the feature of input hand gestures to the amount of force applied to the object, which in turn determines how much it moves at a given time.

As our first step in understanding the usability and user experience of Force Push, we intended to investigate only factor 2 while eliminating the influence from factor 1. In this study, therefore, we did not require the user to define the movement direction and instead focused only on evaluating the novel, gesture-to-force mapping algorithm when that direction is already set. Thus, we limited the task in the experiment to one-dimensional translation.

The user was required to move an object to a target location along a pre-defined dimension (in this case, vertical). Participants were asked to complete the task with a required accuracy as quickly as possible, and we evaluated performance by measuring completion time. A post-experiment questionnaire was used to gather preliminary data of subjective user experience.

In order to cover a wide range of difficulties, we created several conditions by altering the initial distance between the object and the target location (the amplitude in Fitts's Law). The required accuracy—how close the object needed to be to the target for the task to be completed (the target width in Fitts's law)—was kept constant. To test the robustness of the techniques, all task conditions were designed to be difficult with (relatively) large initial distances and a high accuracy requirement. **Table 1** shows the parameters for the four conditions.

To be fair in comparing the two techniques, they should be equipped with near-optimal parameters. Since we were covering a range of difficulties, the pre-defined parameters we chose wouldn't be perfect for each condition. But we ought to choose them to be as adaptive as possible for all the testing conditions.

Due to the high difficulty of these tasks, the direct mapping interface was equipped with two C/D ratios for the user to switch between, with a smaller C/D ratio for coarse, ballistic movement and a larger C/D ratio for fine-grained accurate alignment to reach the target. The values we chose are shown in **Table 2**. In choosing these values, we took into consideration both the human arm's limited range and hand tremor, so an average person could complete the tasks with a reasonable amount of clutches while hand instability would not keep them from achieving the required accuracy.

An alternative approach here would be to use the PRISM technique to adaptively change the C/D ratio based on the hand's movement speed (Frees et al., 2007). However, the tasks in this experiment were extremely difficult, with requirements for both high accuracy and long travel distance. This means the discrepancy between the required largest C/D ratio and smallest C/D ratio was huge. This considerable gap is reflected by the chosen two C/D ratios in **Table 2**, which are the values that we found could cover both ends of the requirement while keeping this gap as small as possible. If PRISM were to be used, it would map the variation in hand movement speed (which provides a quite limited range) to this huge change of C/D ratio. We doubted that the change in hand speed alone could provide enough expressiveness to precisely reach an intended C/D ratio between the two extremes, and we surmised that it might feel similar to switching between the two extremes since the C/D ratio would change very quickly with even a subtle change of hand movement speed. Thus, we chose to provide the two C/D ratios instead. We will consider a comparison between Force Push and an adaptive technique like PRISM as future work.

For Force Push, we fixed the object's mass at 1.0 units in Unity3D. Equations (1, 2) in section Expressive Gesture-to-Force Mapping explained the algorithm to determine the amount of force and the value of the drag parameter. By trying a range of options for the empirical constants in the equations, we found the values shown in (3) to be a reasonable choice, with which the gesture appeared to be expressive enough to achieve both ballistic movements and precise control.

$$\alpha = 1.0,\ a = -0.17,\ b = 87.40 \tag{3}$$

Environment

To visualize the task, we designed the simple virtual environment shown in **Figure 3**. The green cube marked the target location and it remained static throughout the experiment, while the blue cube symbolized the object being manipulated. Since we were only examining 1-D translation on the vertical dimension, the blue cube would only move vertically. The task was to move the blue cube up and down until it had exactly the same height as the green one.

Because some of the difficult conditions would place the object too far-away (e.g., 10,000 m), the displayed distance between the green cube and blue cube was scaled down by a factor of 1000. This meant the blue cube would appear at a location that was 10 m away from the green one when their (conceptual) distance was meant to be 10,000 m according to the task. To maintain accurate information of the task, the original distance was displayed as a number using a unit of 1 mm. The sign and color of the number indicated whether the object was above or below the target, as seen in **Figure 3**. The user could always refer to it to know exactly how far away the object was from the target and which direction to move further. When the magnitude of this number was smaller than 1.0 and the object was released from interaction, the task was accomplished and the timer would stop (**Figure 3B**).

To help the user with accurate alignment when the object got very close to the target, the white board behind the object showed

TABLE 1 | Task parameters (meters).

Initial distances	10	100	1,000	10,000
Required accuracy	0.001 (i.e., 1 mm)			

TABLE 2 | Selected C/D ratios for the direct control interface.

Smaller C/D ratio	1:500
Larger C/D ratio	1:1

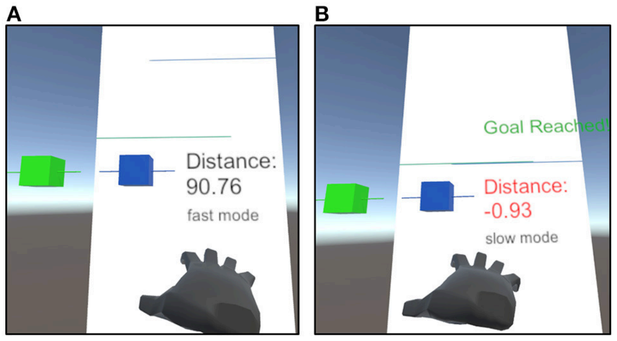

FIGURE 3 | The virtual environment. **(A)** The object is above and close to the target. **(B)** The task is complete.

a zoomed in visualization of the two lines attached to the two cubes. The distance between the two lines on the whiteboard was a scaled up version from the two lines on the two cubes. One simply needed to make the two lines overlap on the white board in order to complete the task. **Figure 3B** shows the display when the task was completed.

The text "fast mode" and "slow mode" seen in **Figure 3** indicated which C/D ratio was currently being used, and was only displayed in the direct control condition. When using Force Push, this text read "using gesture." As soon as the task was accomplished, "Goal Reached!" would be printed on the screen (**Figure 3B**). A virtual hand was always rendered on the screen to indicate whether the hand was being tracked at this moment.

During the experiment, we spent time explaining this visualization as part of the training session. We found that no one had difficulty in understanding it and could all finish the tasks in a reasonable amount of time by relying on them.

Questionnaire

The questionnaire asked about the participant's subjective impressions of the two techniques. The questions were designed to gather preliminary information about seven aspects of user experience:

- Accuracy: Which technique did you feel was more accurate?
- Ease of use: Which technique was easier to use?
- Speed: Which technique did you feel was faster in completing the task?
- Controllability: Which technique did you feel more in control?
- Naturalness: Which technique felt more natural?
- Fun: Which one was more fun to use?

The answers to these questions were collected after the experiment was finished. Participants were also welcomed to provide other comments about the techniques they used.

Apparatus

An Oculus Rift CV1 was used for display and a Leap Motion was applied for hand tracking. The virtual environment was developed in Unity 5.5 and the native physics engine of Unity was used to drive the physics-based simulation of the Force Push interface.

With the direct manipulation technique, the user needed to be able to activate the "grab" and "release" actions and switch between two C/D ratios. A naïve implementation would use the grabbing and releasing postures of the bare hand to activate and deactivate control, but we found that Leap Motion would experience significant tracking loss when the hand is closed (in the grabbing posture). Hence, we let the user hold an Oculus Touch controller in her non-dominant hand and use its trigger and one button to realize these functions, while the translation of the object was mapped to the open, dominant hand motion tracked by the Leap Motion. This separation of function might seem counter-intuitive, but it maintains the same tracking accuracy between the two techniques—the actual movement of the object was always driven by the dominant hand with an open posture tracked by the Leap Motion device.

Activating direct control was achieved through holding down the index finger trigger on the Oculus Touch controller. Once activated, the relative positional change of the hand on the vertical dimension was directly mapped to the translation of the object, using the chosen C/D ratio. Toggling between the two C/D ratios is easily done by pressing the "X" button on the Oculus Touch using the thumb if it's the left hand controller (or the "A" button at the same position if it's a right hand controller).

Participants and Procedure

A total of 20 people were recruited on campus as participants. We required them to have normal range of hand and arm movement. Three of them served as pilot participants in order to test and optimize the procedure. The data from the other 17 participants were collected as the experimental data. Among these 17 participants, 10 were male and seven were female. Two participants were above 30 years of age, while the others were all between 20 and 30. Only one participant was left-handed.

Using a within-subject design, each participant tested both techniques. There were four difficulties (initial distance between object and target) for each technique and each difficulty had two trials. Hence, each participant would perform 16 trials (2 techniques $\times$ 4 levels of difficulty $\times$ 2 trials for each difficulty).

The experiment always started with one technique, completed all its trials and then switched to the other one. The choice of which technique to start with was counterbalanced—nine participants began with direct control and the eight with Force Push. To avoid bias induced by the order of the eight trials, we generated two random orderings of eight trials beforehand. The first random ordering was always used for the technique presented first, and the second random ordering was used for the second technique. The participants were asked to go through a training session with five trials for each technique before the experiment started. Everyone expressed confidence in understanding the technique and controlling the object after the training session.

Results

Task Performance

Figure 4 shows the box plot of the completion time for each technique with the four difficulties.

We ran a two-way repeated-measures ANOVA for the task completion time metric. A significant interaction between the two independent variables was found [$F_{(3,\,14)} = 29.72$, $p < 0.0001$]. This interaction can also be easily observed from the box plot.

We conducted *post-hoc* Bonferroni-corrected *t*-tests to compare the two techniques at each level of difficulty, and found that the direct control technique was significantly faster than Force Push at the three lower levels of difficulty ($p < 0.001$). However, we found no significant differences between the two techniques at the highest level of difficulty, even though Force Push had better performance on average ($p = 0.32$).

Subjective User Experience

Figure 5 shows the distribution of answers to the questionnaire. Most participants perceived the direct control technique to be

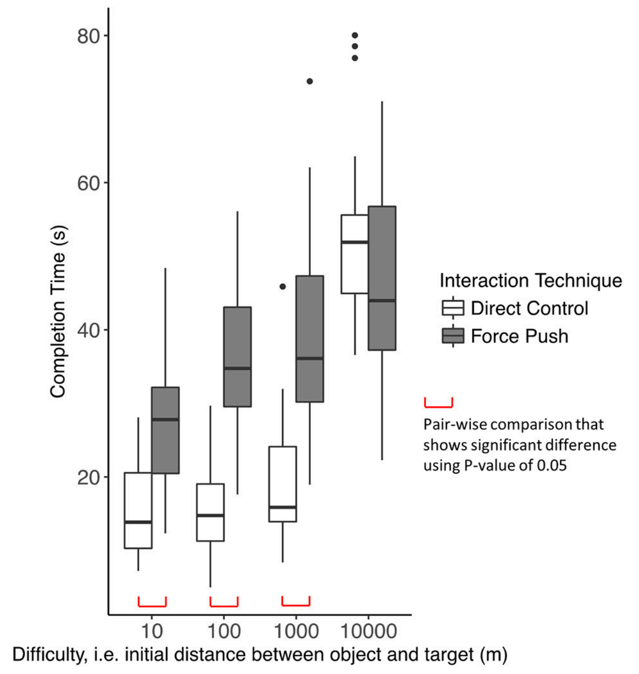

FIGURE 4 | Box plot of the completion time.

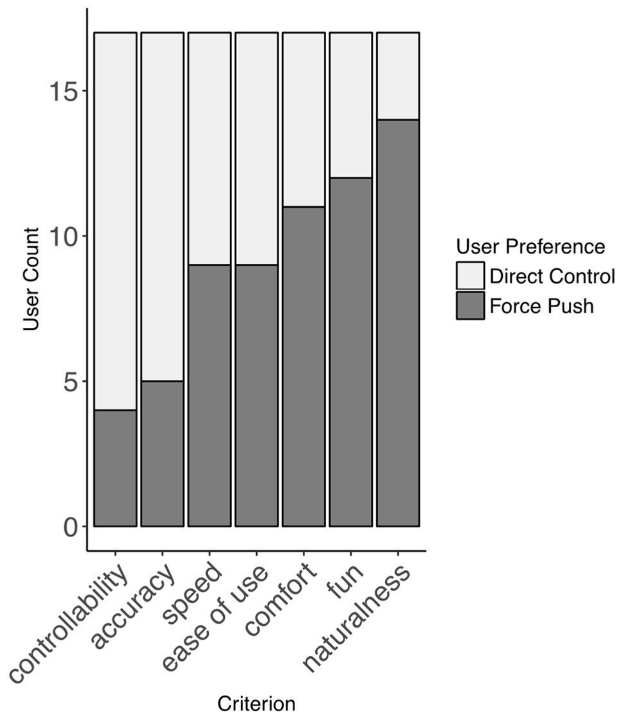

FIGURE 5 | Bar chart for the distribution of user preference based on seven criteria.

more controllable and accurate, but perceived the Force Push technique to be more fun and natural. Slightly more participants perceived Force Push to be more comfortable than direct control. Preference for the two techniques was approximately equal for the criteria of speed and ease of use.

DISCUSSION

A direct observation from the performance result was that all participants could complete all the tasks in a reasonable amount of time. This was not surprising for the direct control interface since it was equipped with two C/D ratios optimized for long-distance and precise movements, respectively. However, this was very encouraging in the case of Force Push, as it used a second-order mapping, which was inherently more difficult to control, and some of the tasks were extremely difficult. Even though Force Push did not outperform direct control overall, it provided reasonable usability in most cases. Observing this result, we tentatively answer research question 1: by dynamically mapping expressive features of gestures to properties of the physics simulation, we created a controllable gesture-to-force mapping for remote object manipulation that was usable even in extremely difficult cases. However, as this experiment was limited to a 1D manipulation task, future work is needed to verify that the Force Push technique remains controllable in the full 3D case.

The fact that direct control had much shorter completion time in most conditions was consistent with the general consensus that a zero-order mapping is easier to control than an acceleration-based mapping. What was more interesting was how each technique's performance changed with increasing difficulty. The time used in Force Push seemed to grow approximately linearly with the log of the initial distance, while the time for direct control seemed to grow at an exponential rate. Granted, the number of difficulty settings and the number of trials was too small here to model these relationships precisely, but we have some preliminary evidence that the gesture-to-force mapping was more robust against the increasing difficulty. One way to interpret this is that the gesture-to-force mapping seems to be more adaptable to extremely difficult situations where the object was placed far away and direct mapping struggled to cover the two ends of the spectrum even with two distinct C/D ratios. We speculate that if the task becomes even harder than the conditions presented in this study, the performance advantage of gesture-to-force mapping will become more apparent.

The participants' subjective evaluation also showed some interesting results. Even though Force Push was less efficient than direct control in most cases, the criteria of speed and ease of use showed equal preference. More surprisingly, the answers to "Which one feels more comfortable to use?" actually favored the gesture-to-force method. We speculate that this result is due to the significant muscle tension required in the direct control interface during the fine tuning stage of the manipulation task. With Force Push, although the hand is held in mid-air, the position of the hand does not have to be controlled precisely.

The comparison on the criteria of fun and naturalness supported our hypothesis, as the majority of the participants favored the Force Push interface over traditional direct manipulation. We surmise there are three reasons behind this result. First, the hyper-natural property of Force Push may have made it feel magical, and this kind of superpower frequently appears in popular movies like *Star Wars*. It is not realistic, yet it feels familiar, which makes the technique fun and natural. Second, we suggest that the force-driven physics-based simulation may have created a richer perception of the virtual world's plausibility (Slater, 2009). For example, the user could have a much stronger sense of the weight of the object using Force Push as compared to the direct control interface, in which the object instantly became "weightless" when it was attached to hand movement. Virtual environments with consistent and explainable physics models may be more plausible and relatable than those without, leading to a deeper sense of presence (Skarbez et al., 2017). Third, the interaction of Force Push might appear to be more novel, as the majority of users have already used position-to-position mappings in human-computer interaction on a daily basis (e.g., mouse).

Despite these preliminary findings about potential user experience benefits of the Force Push approach, more work is necessary to validate our hypotheses regarding the plausibility, fun, and naturalness of our technique as compared to traditional virtual hand techniques in realistic object manipulation tasks.

The results can be used to make preliminary design recommendations for certain applications. For example, if the target application is entertainment (e.g., VR games), it may be beneficial to choose a gesture-to-force mapping over a traditional position-to-position mapping for object manipulation, even though it may sacrifice a certain degree of user performance. On the other hand, if the target application is more engineering-oriented and requires both speed and accuracy (e.g., VR modeling), a direct control interface may still be more appropriate in most cases.

It is important to realize that our work to date with gesture-to-force mappings is only a small step in exploring the design space of mapping gesture features to interaction properties. One purpose of this work is to inspire more effort in thinking deeply about how to leverage the expressive information that may be contained in an individual gesture instead of only searching for a wider range of different gestures. An important take-away from this effort is that the rich interaction realized by creatively using one gesture is at least as interesting as a number of simple actions accomplished by many gestures.

CONCLUSIONS AND FUTURE WORK

Inspired by magic interaction in popular culture and relatable metaphors, this paper proposes Force Push, a novel method of mapping hand gestures to physics-driven movement for remote object manipulation. We hypothesized that this metaphor would result in superior hedonic qualities for user experience, and that even though force-based mappings are inherently hard to control, the novel algorithm of dynamically mapping rich features of input gesture to properties of physics-based simulation would make the interface controllable in most cases.

In an object translation task, the performance and user experience of this novel method was compared against a traditional direct manipulation interface with two control-display ratios. The experiment showed that:

- Force Push appeared to be controllable even in extremely difficult tasks.
- Direct control provided better task performance in most cases.
- However, in extremely difficult tasks where even two C/D ratios struggled to cover the intended range of control, there was some evidence for potential advantages of the gesture-to-force mapping.
- Gesture-to-force appeared to be more adaptive and robust to the change of task difficulty.
- Preliminary results on subjective user experience show that users preferred Force Push for its hedonic qualities, such as being natural and fun to use.

There is a large amount of potential future work. There is still much to learn about object translation via gesture, such as how to find the most effective gesture-to-force mapping in this one case (mapping functions, parameters, gesture features, etc.). We plan to continue searching for improved transfer functions from the gesture features to the physics simulation. Further evaluation of Force Push will focus on more ecologically valid scenarios involving full 3D manipulation. We are also interested in a deeper analysis of subjective user experience than what we were able to study in this first experiment. Finally, we hope to compare Force Push to other state-of-the-art 3D manipulation techniques, such as the PRISM technique (Frees et al., 2007).

At the same time, the design space of mapping gesture features to interaction is huge; we have barely scratched its surface with this one type of gesture (Force Push) for one task (object translation). We hope to test this approach with a wider range of gestures and tasks. For example, a gesture-to-interaction mapping of full-body gestures for effective travel in VR may be an interesting case to examine.

ETHICS STATEMENT

This study was carried out in accordance with the recommendations of guidelines from Institutional Review Board for projects involving human subjects at Virginia Tech. The protocol was approved by the Institutional Review Board for projects involving human subjects at Virginia Tech. All subjects gave written informed consent in accordance with the Declaration of Helsinki.

AUTHOR CONTRIBUTIONS

RY designed the interaction technique, implemented the algorithm and system, completed the experiment, and drafted this manuscript. DB supervised the design, implementation, and experiment of this research work and provided advices throughout the process. He also helped improve the manuscript writing.

REFERENCES

Aigner, R., Wigdor, D., Benko, H., Haller, M., Lindbauer, D., Ion, A., et al. (2012). *Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI.* Microsoft Research TechReport MSR-TR-2012-111 2.

Aliakseyeu, D., Irani, P., Lucero, A., and Subramanian, S. (2008). "Multi-flick: an evaluation of flick-based scrolling techniques for pen interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence: ACM), 1689–1698.

Benko, H., and Wilson, A. D. (2008). "DepthTouch: using depthsensing camera to enable freehand interactions on and above the interactive surface," in *Proceedings of the IEEE Workshop on Tabletops and Interactive Surfaces* (Amsterdam: Citeseer).

Bowman, D., Kruijff, E., LaViola, J. J. Jr., and Poupyrev, I. (2004). *3D User Interfaces*: *Theory and Practice.* Boston, MA: Addison-Wesley.

Bowman, D. A., and Hodges, L. F. (1997). "An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments," in *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Providence, RI: ACM).

Card, S. K., Mackinlay, J. D., and Robertson, G. G. (1990). "The design space of input devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, WA: ACM), 117–124.

Collomb, M., Hascoët, M., Baudisch, P., and Lee, B. (2005). "Improving drag-and-drop on wall-size displays," in *Proceedings of Graphics Interface 2005* (Victoria, BC; British Columbia: Canadian Human-Computer Communications Society), 25–32.

Frees, S., Kessler, G. D., and Kay, E. (2007). PRISM interaction for enhancing control in immersive virtual environments. *ACM Trans. Comp. Human Interact* (TOCHI). 14:2. doi: 10.1145/1229855.1229857

Geißler, J. (1998). "Shuffle, throw or take it! working efficiently with an interactive wall," in *CHI Conference Summary* (Los Angeles, CA), 265–266.

Hascoët, M. (2008). "Throwing models for large displays," in *HCI'03: 11th International Conference on Human Computer Interaction* (British HCI Group), 73–77.

Hinckley, K., Jacob, R. J., Ware, C., Wobbrock, J. O., and Wigdor, D. (2004). "Chapter 21: Input/output devices and interaction techniques," in *Computing Handbook, Third Edition: Computer Science and Software Engineering*, eds T. Gonzalez and J. Díaz-Herrera (Boca Raton, FL: CRC Press), 21-1-21-54.

Kopper, R. A. P. (2011). *Understanding and Improving Distal Pointing Interaction*. Doctoral dissertation, Virginia Tech.

LaViola, J. J. Jr., Kruijff, E., McMahan, R. P., Bowman, D., and Poupyrev, I. P. (2017). *3D User Interfaces: Theory and Practice.* Boston, MA: Addison-Wesley Professional.

Lévesque, J.-C., Laurendeau, D., and Mokhtari, M. (2011). "Bimanual gestural interface for virtual environments," *IEEE Virtual Reality Conference*. (Singapore: IEEE), 223–224.

Moyle, M., and Cockburn, A. (2002). "Analysing mouse and pen flick gestures," in *Proceedings of the SIGCHI-NZ Symposium on Computer-Human Interaction* (Hamilton: ACM), 19–24.

Norton, J., Wingrave, C. A., and LaViola, J. J. Jr. (2010). "Exploring strategies and guidelines for developing full body video game interfaces," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games* (Monterey, CA: ACM), 155–162.

O'Hagan, R., and Zelinsky, A. (2000). "Visual gesture interfaces for virtual environments," in *Proceedings of the First Australasian User Interface Conference (AUIC 2000)* (Canberra, ACT: IEEE), 73–80.

Ortega, F. R., Galvan, A., Tarre, K., Barreto, A., Rishe, N., Bernal, J., et al. (2017). "Gesture elicitation for 3D travel via multi-touch and mid-Air systems for procedurally generated pseudo-universe," in *2017 IEEE Symposium on 3D User Interfaces (3DUI 2017)* (Los Angeles, CA: IEEE), 144–153.

Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (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, WA: ACM), 79–80. doi: 10.1145/237091.237102

Poupyrev, I., Weghorst, S., and Fels, S. (2000). "Non-isomorphic 3D rotational techniques," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (The Hague: ACM), 540–547. doi: 10.1145/332040.332497

Reetz, A., Gutwin, C., Stach, T., Nacenta, M., and Subramanian, S. (2006). "Superflick: a natural and efficient technique for long-distance object placement on digital tables," in *Proceedings of Graphics Interface 2006* (Quebec, QC: Canadian Information Processing Society), 163–170.

Sato, Y., Saito, M., and Koike, H. (2001). "Real-time input of 3D pose and gestures of a user's hand and its applications for HCI," in *IEEE Virtual Reality Conference* (Yokohama: IEEE), 79–86.

Segen, J., and Kumar, S. (1998). "Gesture VR: vision-based 3D hand interace for spatial interaction," in *Proceedings of the sixth ACM International Conference on Multimedia* (Bristol, UK: ACM), 455–464.

Skarbez, R., Brooks, F. P., and Whitton, M. C. (2017). "Immersion and coherence in a visual cliff environment," in *EEE Virtual Reality Conference* (Los Angeles, CA: IEEE), 397–398.

Slater, M. (2009). Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philos. Trans. R. Soc. B Biol. Sci.* 364, 3549–3557. doi: 10.1098/rstb.2009.0138

Song, P., Goh, W. B., Hutama, W., Fu, C.-W., and Liu, X. (2012). "A handle bar metaphor for virtual object manipulation with mid-air interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, TX: ACM), 1297–1306.

van Beurden, M. H., Ijsselsteijn, W. A., and de Kort, Y. A. (2011). "User experience of gesture based interfaces: a comparison with traditional interaction methods on pragmatic and hedonic qualities," in *International Gesture Workshop* (Athens: Springer), 36–47.

Wigdor, D., and Wixon, D. (2011). *Brave NUI World*: *Designing Natural User Interfaces for Touch and Gesture*. Burlington, MA: Elsevier.

Wingrave, C. A., Bowman, D. A., and Ramakrishnan, N. (2002). "Towards preferences in virtual environment interfaces," in *Eighth Eurographics Workshop on Virtual Environments* (Barcelona), 63–72.

Zhai, S. (1998). User performance in relation to 3D input device design. *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques* (Orlando, FL), 32, 50–54.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.