User Behavior Transformation through Dynamic Input Mappings

Dun-Yu Hsiao, Seth Cooper, Christy Ballweber and Zoran Popović
Center for Game Science
Department of Computer Science & Engineering, University of Washington
AC101 Paul G. Allen Center, Box 352350 Seattle WA 98195 USA
dyhsiao, scooper, christy, zoran@cs.washington.edu

ABSTRACT

Games increasingly use input devices – such as Kinect, Leap, and Move – that map the location of the player's body into a virtual space. Designers are often faced with two options: an intuitive, direct mapping that is initially easier to learn but not suited for all tasks, or an advanced mapping that is better suited to carrying out tasks but harder to learn. In this paper we propose a method for achieving the best of both worlds: user behavior transformation. Rather than a single static mapping, our method uses a dynamic input mapping that starts out with a direct mapping but changes over time to a more advanced mapping that is more suited to the types of tasks the player needs to perform. As the mapping changes, the player automatically transforms their behavior to take advantage of the new input space. In this way the mapping can allow for more advanced interactions the designer would like, while still being intuitive. We validate this approach with a Kinect based interface for a game with a complex task: protein folding in Foldit. The initial direct mapping from the player's hand position changes to a mapping that reduces occlusions caused by overlapping hands. In a user study, the dynamic mapping compares favorably to two static mappings with respect to task completion times.

Categories and Subject Descriptors

H5.2 [**Information interfaces and presentation**]: User Interfaces. - Graphical user interfaces.

General Terms

Design, Human Factors

Keywords

Foldit; Kinect; 3D object manipulation interface; detectability; usability; intuitiveness; occlusion; two-handed interaction; vision-based; single field-of-view; tracking system.

1. INTRODUCTION

With an increasing number of input devices that map the player's position from the real world into the virtual one – devices such as Microsoft's Kinect, Leap Motion's Leap, Playstation's Move or Nintendo's Wii Remotes – players have the possibility of using their bodies, hands, and fingers to more directly interact with games. Seeing the potential of these devices, much attention has been paid to creating novel interfaces and interactions that are more intuitive and effective to use. The current interface design trend is to simulate simple, high-level gesture based manipulations. For example, users can swipe to rotate objects, or pinch to zoom the camera. However, designers may desire players to access more complex, detailed manipulations of the virtual world; these types of interactions are still relatively rare.

We propose a method for creating mapped interfaces capable of supporting a wider range of tasks while maintaining the desirable intuitiveness of these types of interfaces: *user behavior transformation*. With this approach, the mapping from the player's inputs (location of the player's body) to the representation of those inputs in the virtual world (location of the player's avatar) is dynamic and changes over time. Initially, the mapping is direct so as to be as intuitive as possible. Over time, the mapping changes to a mapping that is less direct but better suited to the types of tasks the designer intends. As the mapping between the real world and the virtual one changes, the player changes their behavior to use the new mapping. We refer to this as "user behavior transformation" as it happens intuitively – no explicit training is necessary. Rather than explaining how the interface works through a tutorial or on-screen prompts, the player changes their inputs to compensate as the mapping changes.

In this paper, we explore the potential of user behavior transformation by evaluating a dynamically mapped two-handed Kinect interface for the protein folding game Foldit. One of the limitations of Kinect is robustness dealing with occlusions. As a single field-of-view visual tracking device, its tracking capability is heavily dependent on the visibility of its tracked subject. When tracking both hands simultaneusly, fidelity is reduced when the hands overlap or cross, as one of the hands is occluded and hidden from the tracking system. As shown in Figure 1, the interface functionality will be ruined by certain common gestures due to tracking failure. Therefore, it is more constrained when trying to track multiple objects simultaneously.

A direct mapping from hand locations in the real world to their locations in the virtual one will suffer from the adverse effect of occlusion if the player needs to cross their hands. An indirect mapping that avoids this occlusion may cause the player initial confusion as they do not see their virtual hands where they expect them.

With our new dynamic mapping scheme, we hypothesize that it will enable users to move both of their hands as desired, both phys-

ically and on screen, while avoiding occlusions, and that this new mapping will be as intuitive as using a direct mapping, but more effective. In the following section, we will describe our the challenges posed by using two-handed interaction for protein folding and how these informed the specific mapping we used. With respect to each challenge, we will then illustrate how we designed the systems and how we tested our hypotheses. In the results section, we will show that our scheme can improve the player performance in a variety of tasks regardless of any possible learning effects.

2. CHALLENGES

Designing an intuitive 3D two-handed interface using single field-of-view tracking device introduces many distinct challenges. We summarize those challenges into the following four categories.

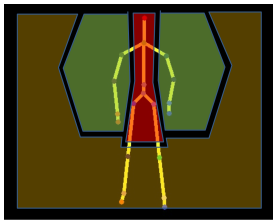


Figure 1: The skeleton tracked by Kinect. Colored areas indicate chance of occlusion occurrence. Red means high, green means low, and yellow means hard to reach intuitively by humans.

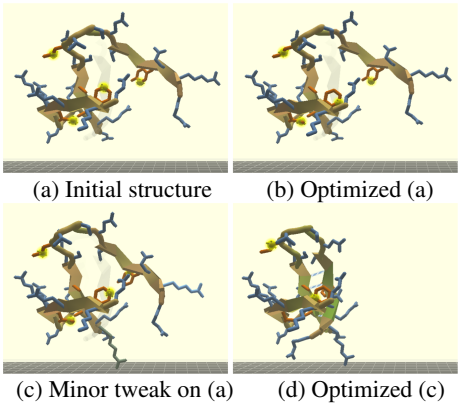
2.1 Occlusion and Estimation Noise from the Sensor

The occlusion noise is specific to field-of-view tracking devices. This type of noise results from the tracking estimation error caused by occlusion or near-occlusion. Occlusion may happen in different occasions. Take Kinect for example, occlusion happens between the joints of one player, or between multiple players. Because of its nature, this kind of noise is spatial-dependent. Its possibility of occurrence increases when two tracked parts become closer to each other. In Figure 1, we mark the high-probability-occlusion area in red, low-probability-occlusion area in green, and the area hard for humans to reach naturally is in yellow. This is coherent with the findings of Polacek et al. [7]. Other than the marked region, new high-probability-occlusion region can happen when two tracked objects are close such as when two hands are close to each other. The amplitude of this noise is peak-like when it occurs: it acts like a value interchange between the affected coordinates so players will find the controllability of the tracked parts are jumping or losing track.

This disruptive tracking can severely degrade the overall tracking performance and prohibit the player from folding the protein structure to desired 3D location. It interrupts and deteriorates any desired motion: the player has to recover from the erratic move and try again to make it right. The error may still affect the player's following attempts as long as he/she still introduces similar occlusion on the tracked parts. Take the normal folding gesture as an example, it is intuitive for humans to fold things right in front of their bodies. Unfortunately, as indicated in Figure 1, this is not an ideal region for detection. The occlusion noise can reoccur very easily because there are potentially eight tracked joints in that region that can occlude one another. This will greatly reduce efficiency and lead to fatigue quickly and possibly frustration.

2.2 Precise Positioning

As we mentioned earlier, the reason why people want to use both hands to solve a problem is because the problem is complicated and requires two hands to deal with. With both hands people expect to gain better controllability over the object. Here is a short explanation of why protein folding is a representative task for precise manipulation as shown in Figure 2.



(a) Initial structure (b) Optimized (a)

(c) Minor tweak on (a) (d) Optimized (c)

Figure 2: We use protein folding as a representative precise two-handed manipulation task. Precise manipulation is required in Foldit because optimization can lead to very different results with subtle differences in starting points.

A protein is a highly flexible long chain of amino acids. Each amino acid has its own degrees of freedom, thus the overall structure can possess extremely high inner degrees of freedom. A common protein can easily have hundreds of degrees of freedom.

As a result of this complexity, any minor difference in the structure might prohibit the structure from becoming the correct solution. In addition, the best compact conformation can vary subtly compared to other less ideal structures. Hence, to fold a protein correctly relies heavily on both 3D understanding and precise manipulation skill. Protein folders must choose the correct spot to work on and the exact spot to place the protein. To prove the effectiveness of our proposed scheme, we want to see if users can position their hands precisely on the protein structure.

2.3 User Understanding of 3D Two-handed Mapping

The mouse has been the major navigating interface between human and computer for decades, and people are very skilled in using a mouse and are very familiar with its 2D mapping to the screen. This can make new 3D interfaces difficult for people to learn because it is very different from the concept of a mouse. Some research also echoes that people often find it inherently difficult to understand 3D space mappings and to perform actions in free space [2].

In addition, two-handed interactions may make the situation more complicated. When users are busy learning the new mapping, they also need to pay attention to keep track with both of their hands simultaneously. If users cannot learn to see the correct connection from their hands to the respective on-screen counterparts, they will get confused. When they are confused, they are more likely to use the wrong hand, and make the wrong move. This results in more correction steps which reduces the overall effectiveness.

The challenge for our scheme is for users to easily adopt the new mapping and intuitively understand which cursor belongs to which hand so that when they are using the device, they forget about how their real hands are positioned. For example, people

nowadays don't look at their hands to move the mouse cursor to the right position, instead, they only keep track of the cursor on the screen and move their hands accordingly. We believe that our interface should have similar traits, so that people will only be staring at the screen while using it. However, how much training is sufficient for people to get to this point is unknown.

2.4 Intuitive Control with Less Instruction

The last challenge we face is how to make our interface easy to learn and intuitive to control. One side of the challenge is how we transfer the knowledge of using the new two-handed interface so that a novice can learn to use the interface like an expert in a short amount of time. The other side is how to teach people to use the interface as we want them to use it. The question that is challenging to answer is how much tutorial or instruction is required to achieve this and what exactly should the instruction address.

3. RELATED LITERATURE

While there are a plenty of articles on hand tracking research, we only include those that focus on the mapping scheme design.

Malik et al. [5] designed an asymmetric two-handed interaction study. However, their work put more focus on describing the interaction with less evaluation so we are not able to determine its effectiveness. Owen et al. [6] provided an analysis of the effectiveness between one-handed and two-handed scenarios. They tested three mappings to map either one or two mouse cursors onto the screen. However, the authors admitted that their two-handed task was not very complete, and they were unable to find significant difference between two of their proposed two-handed mappings. In contrast, our result shows great difference in effectiveness between different mappings.

With proliferation of single field-of-view 3D tracking systems like Microsoft Kinect, Sony Move & EyeToy, Nintendo Wii, or LeapMotion accessible to the public, some work is really concerned with how to create a two-handed scheme using such devices.

Polacek et al. [7] proposed a user interface that moves on the screen along with the user's motion. They also provide a few observations about the comfortable area for a humanâĂŹs hand to interact with Kinect. Our work brings more detailed analysis on two-handed interaction, and part of our design is consistent with their observations.

Song et al. [8] proposed a handle bar metaphor for 3D virtual object manipulation using Kinect. Their system may potentially introduce more fatigue by using both hands all the time. The tradeoff is gaining more controllability of the object. However, from their targeting task, it is hard to determine whether this kind of interaction can tackle realistic and complex manipulation similar to protein folding. Wang et al. [9] developed a vision-based system to track up to six degrees of freedom (DOF) of each hand. Song and Wang's work both use bimanual interaction to increase the controllability. In addition, the KinectFusion project [4] demonstrated its effectiveness of reconstructing the 3D scene by the point cloud from the depth map. This project also showed a way to do robust hand/object orientation estimation and tracking if the object is not too far from the Kinect camera and if the system can have some hardware acceleration support. However, the robustness of those systems are dominated by how visible the tracked hands are. With only direct coordinates mapping, those systems are vulnerable to occlusions. In the latter work of Wang et al. [1], they even used more than one Kinect camera in order to alleviate the issue.

Using Kinect, Hilliges et al.[3] made a holo-projection interactive system so people can interact directly with the device with less effort for users to imagine the 3D-3D mapping. However, the required extra hardware is not easily available to the public.

4. FRAMEWORK

In this section we describe the three mappings we evaluated: two static mapping schemes and our dynamic mapping scheme.

Absolute (ABS) - Static mapping scheme that maps a fixed space in the real world directly into the virtual one. Relative (REL) - Static mapping scheme that maps a relative space around each arm into the virtual world. Adaptive (ADA) - Dynamic mapping scheme that changes a per-hand mapping between real and virtual worlds over time.

4.1 Reduce Occlusion Noise by Coordinates Mapping

The first part of our proposed scheme focuses on solving the occlusion noise problem in the single field-of-view tracking scenario.

4.1.1 Absolute Mapping Scheme (ABS)

Absolute mapping, shown as in Figure 3(a), maps the coordinates in the real world to the screen directly. Absolute mapping is the most common and straightforward mapping scheme and is used in most tracking interfaces. This kind of mapping suffers from occlusion because its interaction space allows tracked objects to occlude one another, such as in Figure 1 and Figure 3(a). Depending on the mapping boundaries, this mapping can also be hard for the user to reach the screen corners. The user may even have to walk around in order to do so.

Data:
Tracked human joints of head, spline (body center), hands, shoulders, hip. Each joint has x, y, and z coordinates. We assume that the length of the outspread arms is equal to the height of a man and the maximum width of the shoulders is a quarter of the height of a man. Note that the unit distance D is correlated to user height, so the mapping function can be adaptive to people with different height.
H - height of the user in camera view
$D \simeq 1/2 * H$ - unit distance, obtained from $Head.y - Hip.y$
D_h - distance between both hands
B_W - bounding box width in camera view
B_H - bounding box height in camera view
(B_x, B_y) - bounding box center in camera view
S_W - screen width

Absolute Mapping Bounding Box:
$B_x = S_W/2$
$B_y = Head.y - 1/2 * B_H$
$B_W = S_W$
$B_H = 3/5 * D$

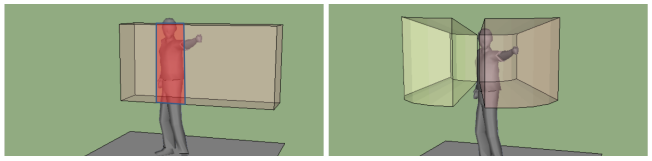
Algorithm 1: Algorithm 1: Absolute mapping function.

However, this mapping is intuitive to human because the hands' relative positions are identical in physical space and on the display. Thus, the user can establish the connection between the physical hand and the on-screen counterpart directly (as shown in Figure 4(a)). We treat this mapping as the baseline mapping scheme for our comparison.

4.1.2 Relative Mapping Scheme (REL)

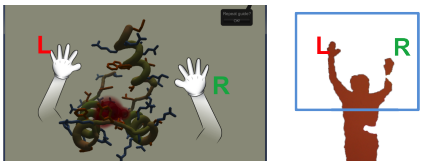
Bearing the idea of minimizing the shared space (high occlusion probability) region, it is reasonable to design a special mapping which is relative to each hand separately. We called this Relative

mapping scheme. In this mapping scheme, a hand's own bounding box that maps coordinates into the screen space will be created near that hand, shown as Figure 3(b). In Figure 3(b), the left hand owns the orange mapping bounding box and the right hand owns the green mapping bounding box.

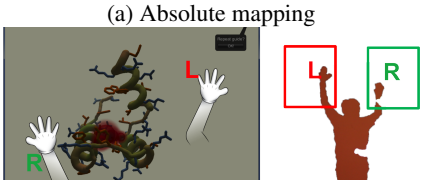


(a) Absolute mapping (b) Relative mapping

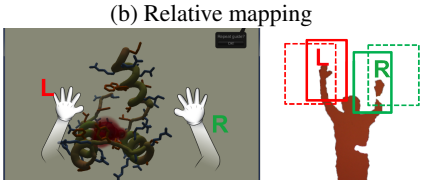
Figure 3: Different hand coordinate mapping schemes. (a) Absolute mapping can easily have occlusion near the red region due to shared space among tracked objects. (b) Relative mapping eliminates the shared space while retaining the freedom for hands to move.



(a) Absolute mapping



(b) Relative mapping



(c) Adaptive mapping

Figure 4: Different hand coordinate mapping schemes and its effect. (a) Absolute mapping has no mapping confusion. (b) Relative mapping may have mapping confusion when hands enter the scene. (c) Adaptive mapping removes initial mapping confusion (solid boxes) and will adaptively transform back to Relative mapping (dotted boxes).

The bounding box is hand-centric, with position fixed with its owner's respective shoulder. Its size is determined as follows: height is from top of the user's head to chest, width is from the side of the ear extending a distance of the length between the shoulders, and depth is from chest extending a distance of the length between the shoulders. The depth mapping is curved and make the bounding box's shape look like a portion of a thick sphere shell. This is to accommodate the depth with respect to how much a human's hand can stretch.

Based on the bounding box design, the user hand can avoid the high occlusion region in Figure 1 and stay in the good region for tracking devices to detect. This mapping can hugely decrease the chance of self occlusion (occlusion between hand to hand or body/head to hand) and increase successful detections. Relative mapping scheme greatly eliminates the occurrence of occlusions.

Also based on the bounding box design, hands usually move near their respective shoulders, so they can move with less efforts. In addition, the coordinates inside each bounding box represents the coordinates of the whole screen, so the user can move each individual hand freely on screen without making both hands touched or crossed. This means the distance for hand to move is also reduced. These traits can reduce the fatigue during operation.

Relative Mapping Bounding Box: (one side)
$B_x = Spline.x + 1/3 * D + 1/2 * D$
$B_y = Head.y - 1/2 * B_H$
$B_W = D$
$B_H = 3/5 * D$

Algorithm 2: Algorithm 2: Relative mapping function.

However, we expect this special mapping will create new problems. Since Relative mapping gives each hand the freedom to navigate the whole space without touching or crossing the other, sometimes the hand mapping is not intuitive. For example, in Figure 4(b) and 10, the user's hands are reversed on screen but not physically. This problem is more severe when the users just start raising their hands to enter the scene (Figure 11(b)). If users first see that their hands appear to be reversed on screen, it's very hard for them to establish the correct link between their hands and the hands on-screen.

Data:
(B_Tx, B_Ty) - targeted bounding box center in camera view
S_W - screen width in camera view
H_d - hand displacement from last frame to the current frame
λ - bounding box morphing rate for Adaptive Mapping
Adaptive Mapping Bounding Box: (one side)
$B_Tx = Spline.x + 1/3 * D + 1/2 * D$
$B_Ty = Head.y - 1/2 * B_H$
Function **Initialization():**
$(B_x, B_y) = (Hand.x, Hand.y)$ when the user first put the hand into the camera view for one second.
$B_H = 3/5 * D$
Function **Update():**
while $(B_x, B_y) \neq (B_Tx, B_Ty)$ **do**
  if $(Hand - (B_Tx, B_Ty)) \cdot H_d > 0$ **then**
    $B_x = B_x + \lambda H_d.x$; $B_y = B_y + \lambda H_d.y$
  end
  if $D_h \leq D$ **then**
    $B_W = (D_h/2)/2/3$
  else
    $B_W = D$
  end
end

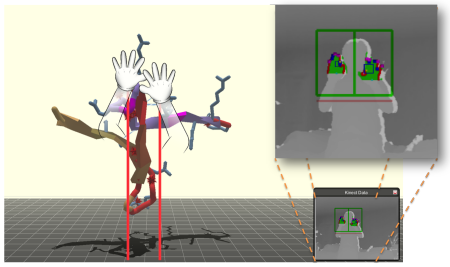
Algorithm 3: Algorithm 3: Adaptive mapping function.

4.2 Adaptive Mapping Scheme (ADA)

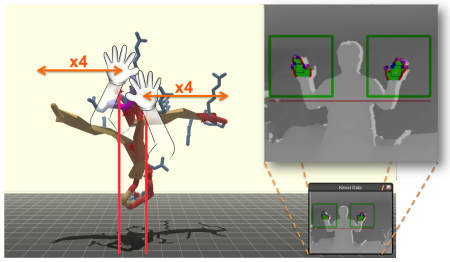
From Relative mapping, we saw the main problem is the confusion of the initial mapping. In order to reach the full potential of this mapping, we need to eliminate the initial mapping confusion.

To achieve this, we made a mapping scheme that makes sure that every time the userâĂŹs hands enter the scene, the relationship between the hands will be the same both physically and on the screen. Our approach is creating new bounding boxes every time the user raises his/her hands, shown as the solid boxes in Figure 4(c). Different from the bounding boxes of Relative mapping, the bounding boxes now are not fixed with respective shoulder position. Instead, they will appear with respect to where the hands are

first recognized by the tracking system. This approach can guarantee the initial mapping of hands will be in the correct relationship.



(a) Adaptive mapping initializes.



(b) Adaptive mapping after four horizontal moves.

Figure 5: Example of Adaptive mapping seamlessly moves user behavior in our desired space without any instruction or intervention. Note the mapping bounding boxes (green boxes) and the user's physical hand position are changed after a few horizontal moves in that extent but the on-screen hands retains the positions and relative motions.

Although this new bounding box design can solve the initial mapping problem, the bounding boxes have less optimal shapes and locations. Note that in Figure 4(c), the bounding boxes are narrower compared to the Relative ones shown in Figure 4(b), so the mapping will be more sensitive, resulting in a greater mapping ratio to the screen. The bounding boxes now may cover some high-probability-occlusion regions and may cause more tracking error.

In order to eliminate the initial mapping confusion and simultaneously have the effectiveness from Relative mapping, we designed Adaptive Mapping to retain all the advantages. In Adaptive mapping, the initial mapping is respective to the user hand's initial position. However, when the user moves his/her hands, the mapping bounding box will adaptively and seamlessly change its shape and location. The change happens silently and keeps the user's on-screen hand motion relatively the same as his/her physical hand motion. This mapping tricks the user by morphing the bounding box with a comparatively slower speed when the user's hand is moving toward the correct mapping space. The morphing speed is set as a portion of the physical hand speed to keep the relative motion on screen. In a few accumulative motions (usually within 5 seconds) the mapping will become identical to Relative mapping. This is shown in Figure 5. Note that the Adaptive mapping is not a transition or combination of Absolute mapping and Relative mapping. The figure demonstrates this: if the user moves both of his hands back and forth along the arrows. After repeating four times of that motion and back to the original position on screen, the bounding boxes (shown in green solid boxes) move from high-probability-occlusion regions to occlusion-free, and good-for-tracking regions.

5. EXPERIMENT

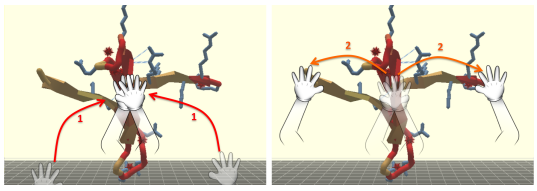
The following experiment was designed to investigate whether our proposed scheme is intuitive and effective. The chosen tasks are the ones in which occlusion easily occurs and which make single field-of-view tracking systems hard to use.

	Split-> Dock-> Swap	Split-> Swap-> Dock	Dock-> Split-> Swap	Dock-> Swap-> Split	Swap-> Split-> Dock	Swap-> Dock-> Split	
ABS->REL->ADA->ABS							3
ABS->ADA->REL->ABS							3
REL->ABS->ADA->REL							3
REL->ADA->ABS->REL							3
ADA->REL->ABS->ADA							3
ADA->ABS->REL->ADA							3
	3	3	3	3	3	3	

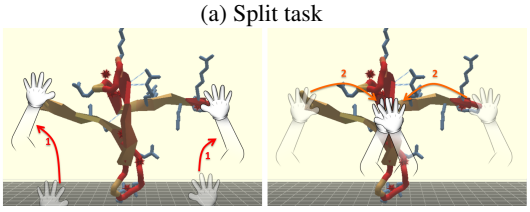
Figure 6: Distribution of subjects randomly assigned to the 36 different task-condition combinations possible. Cells used for experiment is in orange.

5.1 Experimental Design

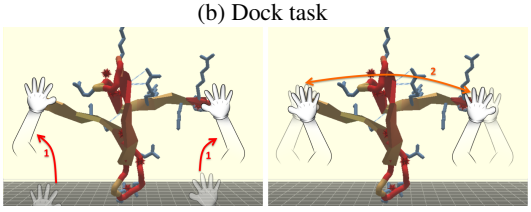
Our design was a 3 x 3 within subjects design. Each subject performed 3 tasks using each of the 3 mapping schemes.



(a) Split task



(b) Dock task



(c) Swap task

Figure 7: (a) Split (b) Dock (c) Swap task for each condition iteration. The steps of each task is marked on the images. Users can complete each step symmetrically or asymmetrically.

Each subject tried all three mapping conditions–absolute, relative, and adaptive–in random order, counterbalancing to reduce the chances of possible ordering or learning effect. In each condition, players do all three tasks, also in random order. In addition, in order to observe whether learning effects influence the performance of each mapping scheme, we intentionally added a fourth iteration of the first mapping scheme condition each player experienced to let the player revisit the first mapping condition again. Every player experienced four sessions in total using the three different mapping schemes.

Here are the three tasks we used in our experiment, these tasks are Split, Dock, and Swap as shown in Figure 7. Each of the task simulates a common object manipulation in the real world:

- Split: spreads the object into two from a center

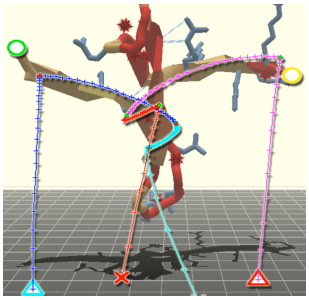


Figure 8: Example user hand trace under Dock task. The two triangles are the starting hand positions of the task and the crosses are the end positions. Green/yellow circles show the ends of the structure. Blue/cyan line is the trace of left hand and the magenta/red line is the right hand trace. The color difference is to show the trace in the first/last half duration. High curvature turning points of the traces are shown in green/red dots.

- Dock: puts two parts closely together.
- Swap: exchanges the position between two parts.

Subjects were asked to perform each task before we actually record the time, so that we can make sure all subjects understood the task goals and were able to perform correctly. Within each task we timed three separate intervals. The first interval started when the player raised his/her hands and ended when the player's hands reached step one. The second interval started when the player's hands reached step one until they completed step two. Users can complete each step with symmetrical or asymmetrical hand movements. We also examined the total time and let the player try four times in each task, taking the best result in each step.

Since there are six permutations of tasks ordering, and six permutations of condition ordering, there are 36 different task-condition combinations. Each subject was randomly assigned one of the 36 combinations. Our goal was to have an even number of subjects distributed over each condition (row) and task (column) combination, shown in Figure 6. There were 3 subjects per condition each with different task ordering. There were 6 subjects who experienced a certain condition sequence (e.g. 6 people take condition 1 as the first condition).

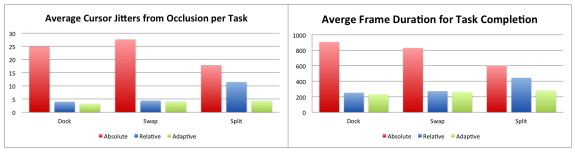


Figure 9: Result comparison for different mappings. The result is the average hand trajectory of the test subjects. In terms of low occlusion, Adaptive mapping is comparative to Relative mapping to have the lowest. Furthermore, the Adaptive mapping also has the advantage of eliminating the hand swapping issue in the Relative mapping and gains some average time reduction to perform a task.

In this experiment, we measured the minimum time for each step per task per condition. For our analysis we used the best times of all conditions, including the repeated condition (note that the first condition repeated after the completion of the other two conditions) to further investigate the learning effect to see how much it will impact the performance among all three mapping schemes. For example, if a user did the condition in the following sequence: Relative, Adaptive, Absolute, and the Relative, the best time in each step of Relative mapping is chosen from the best one in those two iterations.

Table 1: **Mapping Condition by Task: Three-way Comparisons of Minimum Time to Complete Task in Seconds (Friedman Test)**

Red bold face ones are the minimum time spent per task among three conditions

Task	Step	Condition	Mean	Stdev.	Min	Max	Mean Rank
Swap	Step 1	ABS	2.42	0.865	1.3	4.4	1.78
	Chi-Square = 18.353	REL	2.81	0.970	1.2	5.3	2.78
	$p < .001$	ADA	**1.87**	**0.451**	**1.1**	**2.5**	**1.44**
	Step 2	ABS	31.88	38.924	2.7	99.0	3.00
	Chi-Square = 28.800	REL	3.09	0.860	**1.9**	4.5	1.67
	$p < .001$	ADA	**2.78**	**0.497**	2.0	**3.6**	**1.33**
	Total	ABS	34.45	38.715	4.1	101.3	3.00
	Chi-Square = 28.778	REL	5.97	1.691	**3.2**	9.8	1.72
	$p < .001$	ADA	**4.67**	**0.733**	3.4	**6.0**	**1.28**
Dock	Step 1	ABS	2.38	0.674	1.4	3.7	2.19
	Chi-Square = 18.464	REL	2.68	0.856	1.7	5.0	2.58
	$p < .001$	ADA	**1.64**	**0.345**	**1.1**	**2.5**	**1.22**
	Step 2	ABS	11.86	21.981	2.8	99.0	3.00
	Chi-Square = 29.662	REL	2.49	0.560	**1.5**	3.5	1.75
	$p < .001$	ADA	**2.02**	**0.367**	**1.5**	**2.9**	**1.25**
	Total	ABS	14.49	22.008	4.2	101.8	3.00
	Chi-Square = 36.000	REL	5.18	1.224	3.3	7.5	2.00
	$p < .001$	ADA	**3.68**	**0.500**	**3.0**	**4.9**	**1.00**
Split	Step 1	ABS	7.05	8.461	2.6	37.6	2.56
	Chi-Square = 21.778	REL	3.77	1.309	2.3	6.7	2.33
	$p < .001$	ADA	**2.46**	**0.703**	**1.2**	**4.0**	**1.11**
	Step 2	ABS	3.16	0.643	2.0	4.9	2.64
	Chi-Square = 19.391	REL	2.68	**0.526**	1.5	3.6	2.14
	$p < .001$	ADA	**2.24**	0.542	**1.2**	**3.5**	**1.22**
	Total	ABS	10.41	8.540	5.7	41.5	2.72
	Chi-Square = 26.290	REL	6.48	1.665	3.8	10.1	2.19
	$p < .001$	ADA	**4.71**	**0.955**	**2.4**	**6.4**	**1.08**

6. RESULTS

We collected player data from 18 subjects, most of which were novices at using Kinect, and of playing Foldit. An evaluation of the Shapiro-Wilkes Test of Normality indicated that our data were not normally distributed, thus, we used non-parametric statistical methods to analyze our data. The Friedman Test, a non-parametric equivalent to the repeated measures analysis of variance was used to analyze our data, along with subsequent Wilcoxon Rank Sum Tests to identify statistically significant 2-way comparisons. Our analysis of the overall total time for completing our three tasks shows evidence to statistically support that our proposed two-handed mapping scheme out-performed the other schemes, with all but 3 of our 2-way comparisons also showing statistical significance even when including the subject's best time on the repeated condition and a small sample size.

6.1 Interpreting the Results

Here is a list of our hypotheses of the three different mapping schemes:

H1. Occlusion-reduction: Absolute mapping should have no differences in time as compared to the other mapping scheme when there is no occlusion or near-occlusion. So the step one of the Swap and the Dock task, and the step two of the Split task should work normally. However, we should see it will take participants longer to complete a task in the rest.

H2. Intuitiveness: Adaptive mapping removes the initial mapping confusion that Relative mapping has, so there should be less time spent in step one of each task to correct the on-screen hand initialization.

H3. Impact of learning: Because Adaptive mapping becomes Relative mapping after users moves their hands a few times, if

users are learning over time, the latter condition should have the advantage in step two time results.

H4. Intuitiveness: Adaptive mapping will have the best times in all tasks because it can shift the user's behavior to a place better for detection and require less effort to play.

6.1.1 *Time Analysis*

Table 2: **Mapping Condition by Task: Two-way Comparisons of Minimum Time to Complete Task in Seconds (Wilcoxon Rank Sums Test)**

NS (Not Significant), * ($p < .05$), ** ($p < .01$)

	Swap Step 1	Swap Step 2	Swap Total	Dock Step 1	Dock Step 2	Dock Total	Split Step 1	Split Step 2	Split Total
ABS Vs. REL	*	**	**	NS	**	**	NS	*	**
REL Vs. ADA	*	NS	**	**	**	**	**	*	**
ABS Vs. ADA	*	**	**	**	**	**	**	*	**

In Table 1, we showed the minimum time to complete each task under different conditions, and the minimum time in each category is marked in red boldface. The data supports our hypotheses that the proposed Adaptive mapping scheme outperforms other two schemes, and the trend of the data is also consistent with the features we predicted. We see the difference between Relative mapping and Adaptive mapping is smaller in step two comparing to step one, and the Absolute mapping is not an effective mapping scheme in high-probability occlusion situations. The data supports our hypotheses H1 and H2.

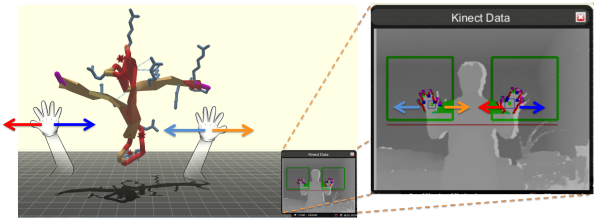


Figure 10: Relative mapping can be counter-intuitive. Hands are reversed on screen but not physically. If the user puts his physical hands closer together, this will result in the on-screen hands spreading further apart. To correct the initial mapping confusion is counter-intuitive.

Since we included the best time result from the additional repeated condition test to get more robust results over all conditions, the data overwhelmingly supports the successfulness of the Adaptive mapping because even when users are given the opportunity to learn the other mappings with an additional session, the performance is still not on par with the Adaptive mapping. This also implies that our proposed Adaptive mapping is easily learned in one short session. This finding rejects our hypothesis H3, but shows that the learning effect will not dominate the effectiveness of the Adaptive mapping scheme.

Table 2 demonstrates the statistical significance of our subsequent analysis of our two-way comparisons. Each of our 3-way comparison were statistically significant as shown in Table 1, and most of our 2-way comparisons (see Table 2) also show statistical significance. The second step of the Swap task between Relative mapping and Adaptive mapping is not statistically significant, but this is actually inline with our prediction as they are designed to be very similar to each other after users take a few moves in the first step. However, interestingly, in the other cases Adaptive mapping all outperform the Relative mapping with statistically significant differences. This suggests that the users migrated their hands into the better working zone and achieved higher effectiveness. This supports our hypothesis H4.

The other non-statistically significant differences are step one of the Dock and Split task between Absolute mapping and Relative mapping conditions. This might be a result of our small sample size, but the data overwhelmingly shows that the subjects were most effective at completing the tasks in the Adaptive condition.

6.1.2 *Hand Trajectory Analysis*

We also examined the hand trajectories in order to support our claim. We first want to see if the new mapping scheme helps bring down the jittery trajectory by reducing occlusion error. An example trajectory of a given task is shown in Figure 8. Since where occlusion occurs will cause the trajectory to have undesired jittery movements, the more random jitters along the trajectory implies more occurrence of occlusion. To identify the random jitters, we use Ramer-Douglas-Peucker algorithm to simplify the trajectory with $\epsilon = 10$ and detect short and high curvature turns. We marked identified locations in red or green dots as shown in Figure 8. In Figure 11, we can see that the jitters are successfully identified and those identified jitters are much more under Absolute mapping. Average jitter rate is summarized in Figure 9.

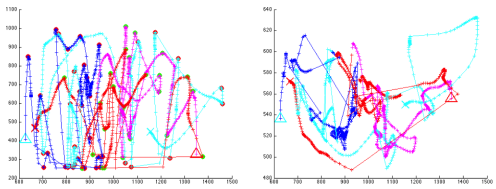
6.2 Observations

When users are using the Relative mapping scheme, most users tend to move their physical hands closer in order to bring the on-screen hands together, like the red-arrow example in Figure 10. However, this results in their hands getting out of the screen and they have no idea how to get their hands back again. The ways to recover from this mapping oddity are by spreading the hands further (blue arrows) or raising the hand/enter the scene with a much wider hand spread, but these are very counter intuitive to users.

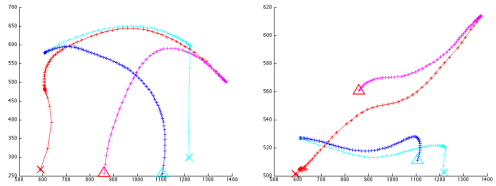
Such confusion may also make the user panic, resulting in more erratic movements. For example, some users attempted to stretch their hands toward the Kinect camera in order to get their hands back into the screen because this is what they usually do with the Kinect camera. This movement fails because it takes their hands out of the effective detection area and moves their virtual hands from the screen. If incorrect detection introduces any faulty moves, users have to take more moves to correct the error. This hugely decreases the effectiveness of the interface and increases user fatigue and frustration. Note that in Figure 11 the X-Z views that the user under Absolute mapping moves more in the Z direction when trying to regain control over jittery movements whereas the Adaptive mapping guides the user to work in the desired area and less Z direction movement.

However, the Adaptive mapping scheme totally eliminates such artifacts in the Relative mapping scheme. This is because Adaptive mapping retains the relative hand motion when morphing the mapping bounding box, we observed that users feel this happens intuitively. However, for most of the time, users do not even notice the change. Similar to Figure 5 demonstrates, we observed that users move their hands from high-probability-occlusion regions to occlusion-free, good-for-tracking, and hand-relaxed regions. It is even more amazing that all of these are achieved without any instruction: users of Adaptive mapping seamlessly shift their hands to the exact location we want them to work. Adaptive mapping transforms user behavior elegantly.

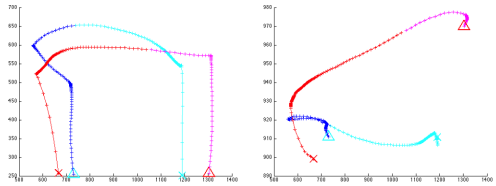
In short, our proposed Adaptive mapping achieves the following



(a) Absolute mapping trace X-Y view (left) and X-Z view (right)



(b) Relative mapping trace X-Y view (left) and X-Z view (right)



(c) Adaptive mapping trace X-Y view (left) and X-Z view (right)

Figure 11: Example user hand traces on screen for the Swap task under different mappings. Absolute mapping: the hand trace shows that tremendous trajectory jitter is caused from occlusion. Relative mapping: the hand trace is smooth because of the elimination of occlusion. However, the starting points of both hands are swapped so that the user has to learn how to recover from swapped hands. Adaptive mapping helps the user to perform a very direct and short traversing path.

goals:

- Our proposed scheme is effective in reducing occlusion artifacts and increases effectiveness of detection.
- Our proposed scheme is intuitive to use and easy to learn, and leads users to move their hands to the most effective area effortlessly.

7. CONCLUSION & FUTURE WORK

In this paper we have presented an approach to user behavior transformation based on a two-handed Kinect interface with a dynamic input mapping. The dynamic mapping improved task completion times as compared to two static mappings in a user study. This approach produced an interface that was both intuitive and allowed the types of interactions the designer desired by avoiding occlusions. The player's behaviors were shifted to the area we wanted elegantly without a single explanation or instruction; it can be an alternative to tutorials or prompts that may be needed to explain how a static mapping works.

The Kinect system presented can be applied to other precise two-handed interactions tracked from a single field-of-view. In addition, we believe that the concept of user behavior transformation can have broad applications for interfaces. We are interested in examining how this approach can be applied to other interfaces, such as full-body Kinect or finger-based Leap interfaces.

Since the dynamic mapping scheme can guide people to work in a certain space, we might be able to generalize our scheme to guide multiple players to work in the same workspace at proper locations effortlessly. We also want to explore the possibility to apply this dynamic mapping in other aspects, such as reducing fatigue. The system could also adapt to specific users. It could automatically recognize input configurations of the player's body that cause trouble (such as occlusions) and adjusting the mapping to avoid those configurations. Or, it could recognize once a player has learned the less direct mapping (after using the system for some time) and transform to it right away.

As virtual reality systems become more pervasive, new possibilities for dynamic mappings may emerge: if the player cannot see their body directly, new transformations may be possible solely based on the user's sense of proprioception.

8. ACKNOWLEDGEMENT

This work was supported by the Office of Naval Research grant N00014-12-C-0158, the Bill and Melinda Gates Foundation grant OPP1031488, the Hewlett Foundation grant 2012-8161, Adobe, and Microsoft.

9. REFERENCES

[1] 3GearSystems. `http://www.threegear.com`.

[2] Herndon, K. P., van Dam, A., and Gleicher, M. The challenges of 3d interaction: a chi '94 workshop. *SIGCHI Bull. 26*, 4 (Oct. 1994), 36–43.

[3] Hilliges, O., Kim, D., Izadi, S., Weiss, M., and Wilson, A. Holodesk: direct 3d interactions with a situated see-through display. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM (2012), 2421–2430.

[4] Izadi, S., Newcombe, R. A., Kim, D., Hilliges, O., Molyneaux, D., Hodges, S., Kohli, P., Shotton, J., Davison, A. J., and Fitzgibbon, A. Kinectfusion: real-time dynamic 3d surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks*, ACM (2011), 23:1–23:1.

[5] Malik, S., Ranjan, A., and Balakrishnan, R. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, ACM (New York, NY, USA, 2005), 43–52.

[6] Owen, R., Kurtenbach, G., Fitzmaurice, G., Baudel, T., and Buxton, B. When it gets more difficult, use both hands: exploring bimanual curve manipulation. In *Proceedings of Graphics Interface 2005*, GI '05, Canadian Human-Computer Communications Society (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005), 17–24.

[7] Polacek, O., Klima, M., Sporka, A. J., Zak, P., Hradis, M., Zemcik, P., and Prochazka, V. A comparative study on distant free-hand pointing. In *Proceedings of the 10th European conference on Interactive tv and video*, EuroiTV '12, ACM (New York, NY, USA, 2012), 139–142.

[8] Song, P., Goh, W. B., Hutama, W., Fu, C.-W., and Liu, X. A handle bar metaphor for virtual object manipulation with mid-air interaction. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM (2012), 1297–1306.

[9] Wang, R., Paris, S., and Popović, J. 6d hands: markerless hand-tracking for computer aided design. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 549–558.