A Self Adaptive Architecture for Hand-Tracked 3D Authoring Interface

Fabrizio Nunnari[1]
DFKI
Alexis Heloir[2]
DFKI and LAMIH

Abstract.

This paper presents a natural and intuitive interface that uses a consumer-range 3D hand capture device to interactively edit objects in 3D space. After assessing the potential benefit of 3D input interaction in a preliminary study, we propose a self-adaptive architecture that supports intuitive and efficient 3D manipulation while accounting for the user's skills and instantaneous performance. While running, the system monitors the user's behavior and performance to maintain a straightforward user model. This model then drives an online re-arrangement and re-parameterization of a rule-based system driving the interaction.

1 Introduction

Recent advances in consumer-range interaction devices like the Kinect[3] or the Leap Motion[4] has opened the door to an unpreceded range of new user interfaces, interaction modalities and metaphors where gesture and bodily interaction are the cornerstones. Taking inspiration from these trends, we propose a self adaptive architecture that has the potential to assist novice users in 3D authoring task while improving the productivity of skilled users.

In a preliminary experiment we evaluated the performances of two users performing 3D authoring tasks using a hand tracker and a classical keyboard. The experiment aimed at understanding how better the performance could be when using a hand tracker instead of a mouse. Additionally, we have been able to quantify the proportion of time spent using the keyboard (to manage the interaction) over the time spent accomplishing actual editing. We found cues indicating that this proportion depends on the user's experience. This experiment allowed us to define a set of requirements and guidelines that are addressed in a keyboard-less self-adaptive architecture presented in the second half of this paper.

2 Preliminary Experiment

The goal of this preliminary experiment is to assess the benefit introduced by using the Leap Motion controller in basic 3D authoring tasks: positioning objects in 3D space and posing humanoid characters. Using a mouse-based interface or a multitouch screen, users can control at most three to four degrees of freedom at the same time ([X, Y, scroll] or [X, Y, pinch, rotate]). In contrast, a 3D input device like the Leap Motion provides a direct mapping between the physical space of the user's hand and the edit space along six degrees of freedom (Rotation and Translation). In theory, users could simultaneously move and rotate objects in the 3D space, thereby perform edit tasks faster. We thus expect direct 3D manipulation to perform better than the mouse and keyboard, at least for 3D object positioning. For single target selection, Sears and Shneiderman [1] have shown that direct-touch outperforms the mouse.

2.1 Task and Experiment design

We compare the performance of a 3D positioning task across two input conditions: 1) Mouse and Keyboard (M&K) and 2) novel input system based on Hand-Tracker and Keyboard (HT&K). This comparison, however, can only be performed on subjects who already have experience with 3D software. Since we realized that novice subjects are not capable of performing the 3D manipulation task using keyboard and mouse, in our experiment the novice subject only used the HT&K mode. We could however compare the performance of the novice subject using the 3D input versus the performance of the experienced subject using the Keyboard and Mouse, as well as the differences for the same experienced user across the two modalities.

The evaluation has been carried out on two contexts. The first consisted in positioning a 3D brick (translation and orientation) in a 3D environment. The second consisted in posing a humanoid figure in the 3D environment using the handles and the inverse kinematics (IK) provided by its animation rig. In this configuration, the user is controlling the translation and orientation of the end effectors, while the joints configuration of the manipulated character are automatically inferred by the IK system.

The character's pose editing context has been split into two scenes: hands positioning (translation and rotation), and hands+elbows positioning. The subject had to accomplish five tasks per scene. For each task the goal was to fit the brick or the character's hands inside a semi-transparent box. Additionally, for the second scene of the pose editing, the elbows had to fit inside a semi-transparent sphere. Figure 1 shows examples of correct alignments for hand and elbow. The containers were by default coloured red, when the hand or the elbow was correctly centred and aligned the container colour turned into green. For elbows only position is taken into account. The size of the box was set to 22x16x10cm, distance tolerance to 3.5cm and rotation tolerance to 0.35 radians ($\sim$20 degrees). The radius of the spheres was set to 0.8, distance tolerance at 3.5cm.

[1] e-mail:fabrizio.nunnari@dfki.de
[2] e-mail:alexis.heloir@dfki.de
[3] http://www.xbox.com/kinect (25 Feb 2014)
[4] http://www.leapmotion.com (25 Feb 2014)

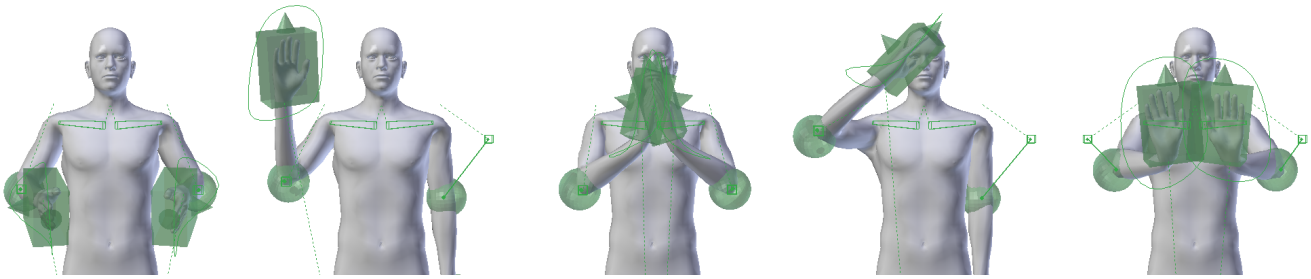

Figure 1. Screenshots of the 5 character arms positioning tasks: hands forward, hello, pray, sir yes sir!, and stop

2.2 Subjects and Apparatus

We conducted the preliminary study on two subjects. The first subject was an expert 3D modeller and animator who has been regularly using Blender for modelling and animation for the last 4 years. He accomplished the tasks with both traditional Mouse and Keyboard (M&K) input system and with the Hand-Tracker and Keyboard (HT&K). A second subject had no prior knowledge nor experience in 3D editing and manipulation. He conducted the study using only the HT&K only since he was not capable of accomplish the tasks using M&K.

The study has been conducted on a Mac Book Pro Laptop (2.4 GHz Intel Core i7 CPU, 16GB Ram, OS X 10.8.4) connected to a 22 inches monitor (resolution 1680x1050) at about 60 cm of distance from the eyes. An operator was sitting next to the subject, monitoring the advancement of the experiment, switching between tasks and (de)activation the logging system. The 3D editor was Blender version 2.66.1. We developed a set of Python add-ons to map the leap Motion input onto 3D objects position. We are publishing on-line[5] the sources that are necessary to build and reproduce the described experiment.

2.3 Preliminary Results

For each trial, we recorded the time spent by the subject while manipulating the interface (hitting one of the G,T,R or F key). We started the timer immediately after the subject touched the first edit key to begin a trail (task) and stopped the timer as soon as the task was done, fulfilled, performed. We distinguished between the time spend while moving objects in the scene (i.e. re-locating and/or rotating an object) from the time spent in switching between different editing modes.

Figure 2 shows the average time each participant took to accomplish each scene. We take as base reference the time needed to accomplish the tasks by the Expert user using traditional M&K devices (first column of each triplet). Results show that: i) the expert subject was able to accomplish the tasks faster using the HT&K (-40.6% average time needed), and ii) the novice subject using the HT&K needed +24.3% average time more. If we consider only scenes 6 and 7 (the longest and later ones) these percentages further converge in favour of the HT&K approach, with values of -45% and +5.7% respectively. In other words, using the HT&M approach the expert user is performing almost at double speed, and the novice user almost matches the speed of the expert with M&K. We found significant difference between the experienced user using M&K and the experienced user using Leap in scene 1 (Brick Translation) and 3 (Brick Rotation and Translation) ($p < 0.05$). We also found a tendency for scenes 4 (Hands) and 5 (Hands and Elbow) $p < 0.09$. Test was a two tailed t-test with a significance level (a) set at 0.05.

The more the target object was mis-aligned from the object to place, the better was the performance with the HT&K. A closer look to the results shows that the performance gains obtained by the HT&K compared to M&K increase with the task complexity. The great variance in the task duration by the expert subject using the HT&K can be explained by his tendency to quickly moving the object in space by performing large gestures, which were causing the Leap Device to occasionally loose track of the expert's hand, forcing him to redo some editing actions.

2.4 Insights

We observed a number of phenomena suggesting that performance editing with the HT&M approach can further improve together with the reliability of tracking technologies. In the brick context the subject had a tendency to align the hand to more naturally grab the brick, as we do with handheld devices. In the arms editing context we noticed the tendency to start the editing action putting the hand in a mirrored position of the virtual hand. This tendencies were conflicting with the request for the subject to avoid extreme vertical or upside down palm positions, leading the Leap Motion into detection errors.

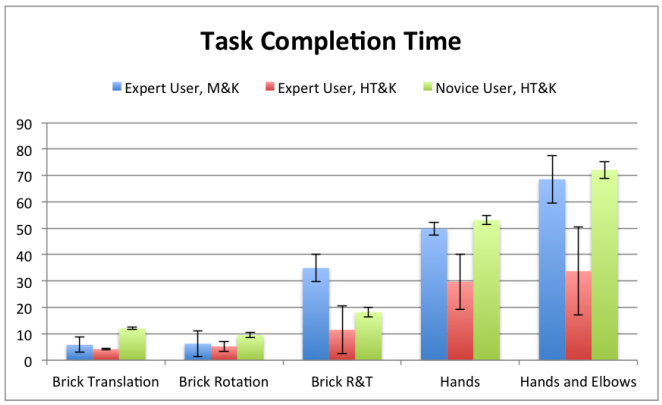

Figure 2. Average time (seconds) needed to accomplish the tasks.

[5] http://slsi.dfki.de/software-and-resources/

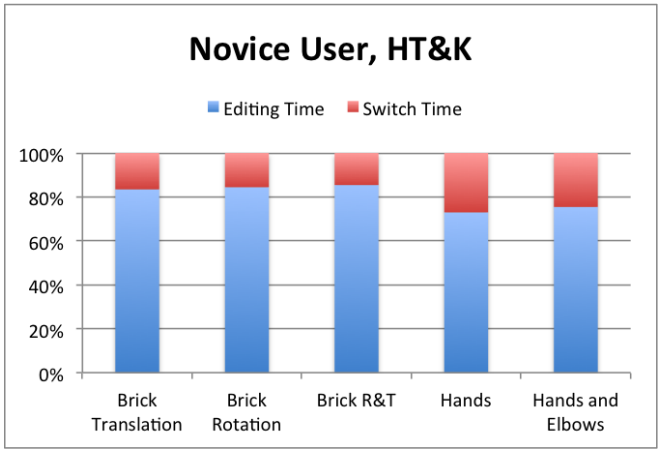

Figure 3. Proportions between editing time vs switch time.

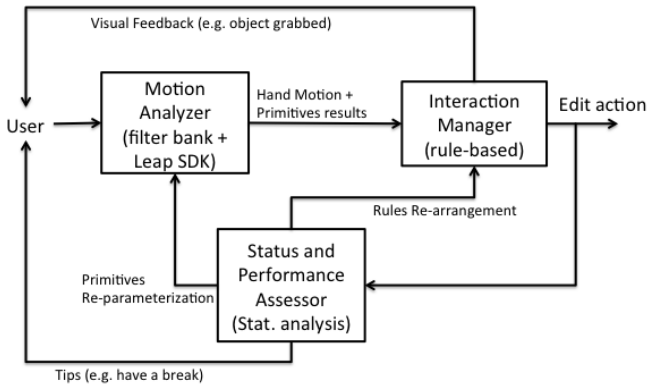

Figure 4. Our architecture is inspired by a feedback controller pattern

We also compared the time spend in editing with the time spent in switching editing operation. Figure 3 shows for, for the HT&K experiments, the proportion between the editing time vs the switching time. The more complex is the task, the more time (up to 20%) the subject spend in switching between actions rather than in editing.

This suggested us to design an interaction mode based solely on the hand tracking device, aiming at increasing even more the performance and eliminating the need to learn keyboard commands. Also, we believe that self-adaptation can be useful not only among different users, but also during a long working session of a single user, as consequence of his fatigue condition. We describe such design in the following section.

3 Self-Adaptive Architecture

The overall architecture of our authoring system is summarized in Figure 4. It follows a feedback-controlled loop model where the user input (hand motion) is filtered out and analyzed by a component called the Motion Analyzer. This component infers a set of mid-level motion primitives and sends them to the interaction manager: a reconfigurable rule based system in charge of triggering the right interaction mode according to its input motion primitives. The interaction manager delivers a flow of edit actions. This flow is continuously analyzed by the Status and performance assessor. All the components work at a high frequency (around 30 times per second), typically required by real-time interactive systems.

The system should also be capable of recognizing performance level deterioration and take measures against it. Again, by enforcing good practices or by suggesting pauses. Indeed gestural interaction, when used extensively, might induce fatigue, sometimes called *gorilla arm*[6]. To avoid this, the assessor maintains a vector of descriptors that could be viewed as a simplistic user model accounting for proficiency and fatigue level. When the fatigue level increases too much, the system suggests a short break to the user. Also, the Status and Performance Assessor is continuously tuning the rules of the interaction manager to improve the user's comfort and level of performance.

[6] http://en.wikipedia.org/wiki/Touchscreen#.22Gorilla_arm.22 (25 Feb 2014)

3.1 Motion Analyzer

The activity of the Motion Analyzer is based on the information stream received from the hand tracking device. The Leap Motion service provides data as a fast paced (30 Hz ca.) sequence of frames through a web socket local connection. Each frame contains, among others, information about the following elements: number of detected hands and a set of data-structure instances storing the position and the orientation of each palm and each detected fingertip.

The Motion Analyzer filters out and analyzes the flow of frames streamed from the Leap. It consists of a set of primitive functions performing the analysis on a time-sliding window buffer of the Leap frames received in the last 2 seconds. All these functions are predicates (they return true or false) and a selection of them can be viewed as anonymous functions that are to be curryfied according to the parameterization instructions delivered by the Status and Performance Assessor and used in the ruleset of the interaction manager.

3.2 Interaction Manager

The Interaction Manager is an online reconfigurable rule-based / production system in charge of switching to the right interaction state according to the values computed by the mid-level motion primitives introduced in the previous section. The system handles three interaction states, as depicted in the right side of Fig. 4:

- HOVER: this state is active when a hand has been detected by the tracking system, but no editing action is actually carried on,
- GRAB: active when the user has selected an object and is moving it (visual hint),
- IDLE: active when no hand is visible by the tracking device.

When defined, new motion analysis primitives are bound to a selection of dynamic variables. They mostly represent time and distance thresholds. The value of these variables is updated on the fly according to how the user performs.

The first rules govern the basic state transition triggered by the presence/absence of the hand. When a new hand is detected the system switches to state HOVER, regardless of its previous state. Similarly, when a hand tracking is lost the system switches to the IDLE state.

The rest of the interaction is based on the principle that when the user wants to start an editing action he needs to stabilize her hand into

the sensor action space. When the hand is stable for enough time, the selected 3D object will start following the hand. This rule requires the user hand to be somehow far from the position where an object was dropped (GRAB state exited) in order to avoid undesired re-grabbing. Stabilizing the hand again terminates the editing action.

Two special cases are handled in order to limit edit errors and keyboard interaction. The first deals with an excessive hand speed movement when editing the object. If the user moves her hand too fast, the object will be dropped in the position it was when the high speed was detected. This limits the possibility to accidentally "throw" the object in an undesired position – with consequent need to undo the operation (typically crtl-z) – when the user quickly removes the hand from the action space for whatever reason. The second special case helps the user in performing large repetitive movements of the object in the 3D space. When the user is moving an object towards one direction he might reach the limit of the action space or reach an uncomfortable position. Continuing the movement of the object in the same direction would require to stabilize the hand to drop the object before moving it back to a more comfortable position. A similar behavior is experienced when the user drag the mouse out of the mouse pad: he then needs to lift the mouse, bring it back to the center of the pad and continue editing. We call this behavior "carriage return" since we believe it evokes the carriage return operation that used to be performed when the printing head of the typewriter reached the end of a line. When the user is suddenly reverting the direction of movement to go back to a comfortable position the object is immediately dropped (switch to HOVER state) without the need to wait for hand stabilization.

3.3 Self-adaptivity – Status and Performance Assessor

In the previous section, we saw how the state-transition was governed by a set of rules in the Interaction Manager. These rules are can be modified by reparameterizing the primitive functions composing the rules and their arrangement. Such changes have an influence on the interaction dynamics and the goal of the Status and Performance Assessor is to guarantee that the current rule arrangement and parameterization maximize the user's comfort and efficiency.

We monitor aspects of the user interaction, such as her ability to keep her hand still, to control the velocity profile of the movement, as well as the reaction time to visual cues. This architecture is user-agnostic, which means that it does not build and track a model of the user, it rather tracks the short-term evolution of the user interaction and apply adaptation strategies in order to either increase the user's efficiency or to limit the decrease of performance level.

We now present how the Status and Performance Assessor monitors the user's performance level. Table 1 lists the variables we use to measure user performances. The values of the variables are calculated through the observation of the last $N_o = 25$ recognized actions. In the following we describe each assessment variable and its influence on the interaction manager's ruleset parameterization.

The value e is the exponential moving average of the last 25 edit actions.

The evolution of the resulting e value gives us a hint about the user's ability to perform faster or slower edit actions. If e decreases, we assume that the user needs a more responsive system. Practically, e is used as an adjustment for a feedback gain that multiplies GRAB_START/STOP_STABILITY_TIME by $(1 + e_i)$ at each iteration.

m is computed exactly the same way as e on the cumulative distance that is traveled by the hand during an action. Since an experienced user is capable of accomplishing an edit with only a few large actions (GRABS) of the hand in space, A decreasing value of m indicates the user tendency to perform longer movements, including large positioning (hand is fast) and fine positioning (hand is slow) of the manipulated object. This suggests that the user is gaining in efficiency and that the interactions rules must be accommodated.

Table 1. List of variables used to assess user performance

Name	meaning
e	action edit time
m	action linear hand movement
c	cancelled actions
l	lost hand tracking
f	fast hand movement detected

The value of c depends on how many of the last N_o operations have been canceled. An operation is canceled when the user presses the ESC key in order to restore the 3D object in its initial position. For each operation with $1 < i < N_o$, we consider $c_i = 0$ if the operation has not been canceled and $c_i = 1$ id it did. We calculate the tendency by interpolating a line among the sampled results. The tendency c' is calculated as the tangent of the interpolated line. We consider a positive tendency as the fact that that too many operations started when the user didn't really mean to do.

The value of l is calculated, similarly to c, by counting how many times the hand tracking has been lost while performing the last N_o operations. A positive tendency tells us that the user hand is exiting too often from the editing space. This means that the user hardly feel uncomfortable in extreme hand positions. We use this as hint that we can increase the sensitivity of the overall system, i.e., increase the ratio between the quantity of motion performed by the dragged 3D object with respect to the same quantity of motion performed by the hand in real world.

The value of f is calculated similarly to c and l, by counting how many times the GRAB state has exited because a fast hand movement has been detected.

If this occurs too frequently, the system may suggest the user to have a short break so that she could recover from the fatigue that might be induced by the gorilla arm effect. We are conducting further tests involving long edit session to correctly adjust this variable.

4 Conclusion and Future Work

We first showed in a preliminary user study that a straightforward combination of hand tracking and keyboard interaction has the potential, on the one hand, to enable novice users to accomplish non-trivial 3D authoring tasks and on the other hand, to increase the productivity of experienced users. Capitalizing on this preliminary result, we propose a self-adaptive system that considerably limits keyboard input while maintaining the highest reactivity level. Such behavior plasticity is enforced by tuning the rules driving the interaction according to the parameters of a basic user model that are inferred at runtime from the user's behavior. This system will be tested on a dozens of skilled student from a renowned design institute. Results of the evaluation campaign will be available and presented during the workshop.

REFERENCES

[1] Andrew Sears and Ben Shneiderman, 'High precision touchscreens: design strategies and comparisons with a mouse', *International Journal of Man-Machine Studies*, **34**(4), 593–613, (April 1991).