Touch Feedback in a Head-Mounted Display Virtual Reality through a Kinesthetic Haptic Device

Andrew A. Stanley
Stanford University
Department of Mechanical Engineering
astan@stanford.edu

Alice X. Wu
Stanford University
Department of Mechanical Engineering
axwu@stanford.edu

ABSTRACT

This paper presents the integration of a three degree-of-freedom (3-DOF) kinesthetic haptic device with a head-mounted display (HMD), using a virtual piano keyboard as a motivating application. The inertial sensors in the HMD allow the user to view the full set of keys without zooming out simply by turning his or her head to look up and down the keyboard. The haptic device further increases immersion into the virtual reality, allowing the user to reach out and touch the piano keys, feeling the dynamics of each keystroke and hearing the individual notes played. The large visual workspace afforded by the HMD exceeds the physically-limited workspace of the haptic device, so a button clutch, workspace drift, and ballistic cursor control are each implemented as strategies to expand the reachable workspace without sacrificing tactile precision or haptic resolution.

Author Keywords

Virtual reality; head-mounted display; haptic rendering; system integration

ACM Classification Keywords

H.5.1. Information Interfaces and Presentation (e.g. HCI): Multimedia Information Systems - Artificial, augmented, and virtual realities;; H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces - Haptic I/O

General Terms

Human Factors; Design.

INTRODUCTION

With the advancements of technology for head-mounted displays (HMDs), virtual reality is rapidly gaining traction as a common form of user experience. However, these head-mounted virtual reality displays, by and large, lack haptic feedback, preventing the user from physically interacting with the environments they explore. As a result, users in the virtual reality of an HMD are often left grasping blindly or even swinging wildly at the real physical world in front of them that does not correlate to the scene that is rendered visually.

Figure 1. A user physically interacts with a piano keyboard in a virtual reality through a commercial haptic device while wearing a head-mounted display (HMD). The graphics shown on the screen of the HMD are mirrored onto the computer monitor.

Haptic devices incorporate the sense of touch into technology interfaces to provide additional information or feedback to a user or robot.

HEAD-MOUNTED DISPLAY AND HAPTICS INTEGRATION

The integration of an HMD and a kinesthetic haptic device requires real-time communication between the two sets of firmware. We selected CHAI3d [1] as the interface between the devices for its built in functionality to provide OpenGL capabilities to haptic environments on a wide range of commercial haptic devices. Furthermore, CHAI3d provides a convenient framework to split the haptics and graphics computations into separate threads to keep the haptics loop-rate around 1 kHz even with the complexities of rendering graphics to a HMD. While not the highest fidelity device, we chose the Novint Falcon as our 3-DOF kinesthetic haptic device due to its low cost and therefore greater accessibility to a wider range of user bases. The Oculus Rift Development Kit 1 served as our head-mounted display and its Software Development Kit (SDK) Version 0.2.5 C++ API provided access to the necessary components of its firmware.

Sensor fusion for head rotation tracking

The Oculus Rift HMD includes a built-in gyroscope, accelerometer, and magnetometer, so the sensor fusion of these inertial measurements provides very accurate estimates of the roll and pitch, relative to the gravity vector, and yaw, relative to starting orientation, of the HMD with minimal drift. To integrate the user's head motions into the virtual reality, we post-multiply the starting rotation matrix of the camera view by the individual rotation matrices for each of the roll, pitch, and yaw measurements about the fixed frame.As a result, the rotation matrix describing the orientation of the camera in the virtual reality matches the one describing the user's head relative to the real world at all times, creating a deeper sense of immersion into the CHAI3d graphics environment. While the head rotation allows the user to very intuitively reorient the camera to look around the virtual environment, Version 1 of the Oculus Rift Development Kit does not provide a means to move around the environment or adjust the zoom. To add these translational degrees of freedom we relied on the arrow keys on the computer keyboard. To keep these translations intuitive with the perspective changes caused by the user head rotations, we pre-multiply their transformation matrices by the camera rotation matrix such that

$$T_{\text{Camera}} = R_{\text{Camera}} \cdot T_{\text{KeyboardInputs}}. \quad (1)$$

Thus, when the user uses the up or left keys, for example, to zoom in or move left, the camera will zoom or move according to the direction he or she is currently looking rather than according to the base reference frame.

View, projection, and distortion

Working with an HMD like the Oculus Rift, where the screen is located in such close to proximity to the user, presents some complexities in rendering the graphics. The screen uses stereo rendering such that each eye sees a separate image on one half of the screen, but these images must differ to account for the difference in location between the two eyes. While the Oculus SDK API provides commands for calculating the projection matrix from the half-screen aspect ratio and vertical field of view, the viewport adjustment from the interpupillary distance, and the distortion correction for the lenses, we needed to create a custom stereo setting in CHAI3ds cCamera.cpp class to account for these adjustments in the actual OpenGL rendering of the scene. The effects of this custom stereo rendering are apparent in the background of Figure 1 where the Oculus display is mirrored onto the computer monitor while the user interacts with the haptic device in the virtual reality.

APPLICATION: HAPTIC VIRTUAL PIANO

To motivate our work integrating the HMD with CHAI3d and to demonstrate the effects we developed a virtual piano haptic environment. This environment included the full 88 keys of a standard piano keyboard. While one would have to zoom out very far to be able to view a full keyboard on a computer monitor, the HMD allows the user to look up and down full keyboard by simply turning his or her head while maintaining the resolution provided by showing only a subset of the keys on the screen at any given time, as shown in Figure 2. To render the surface interaction forces for each key on the haptic device, we implement virtual boxes with the god object algorithms described in [9] to prevent pop-through and improve the illusion of rigid objects.

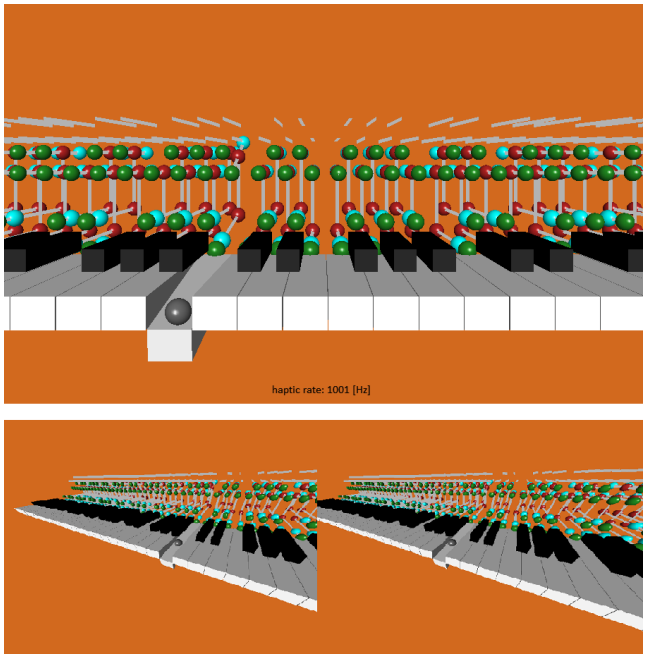

Figure 2. A full piano keyboard built in CHAI3d for the user to physically interact with through a haptic device. Normally, the user can only see a subset of the keys (top). With the HMD, the user can turn his or her head to see the highest and lowest keys (bottom), rendered in stereo vision with projection and distortion to account for the HMD lenses and interpupillary distance.

For a more interactive haptic piano, the user must not only be able to feel the surface interaction forces of each key but also the dynamics of individual key strikes. To add these features to our application, we built upon the simplified dynamic models of piano keys described by Gillespie et al. [4]. We selected the two-body model, described and illustrated in greater detail in [4], that includes a key and a hammer rotating about fixed fulcrum points, shown in green in Figure 3 along with the constants describing the model layout. The center of mass, shown in blue, for the hammer is approximated at the end of its lever and for the key is located just beyond the fulcrum behind the key box that the user interacts with. A spring-damper, whose endpoints are shown in red, connects the key lever to the hammer lever but acts only in compression so that the hammer can strike the string and fall back in free flight. The user applies an interaction force to the key lever with the gray cursor. The equations of motion describing this dynamic system are

$$\ddot{q} = \frac{kl^3}{I_h}(l_2 s + l_3 q) - \frac{m_h l_4 g}{I_h} \quad (2)$$

as the variables are defined in Figure 3. To maintain an acceptable haptic loop-rate despite the inclusion of so many dynamic objects in a single virtual environment, we only update the dynamics of keys with which the user is currently

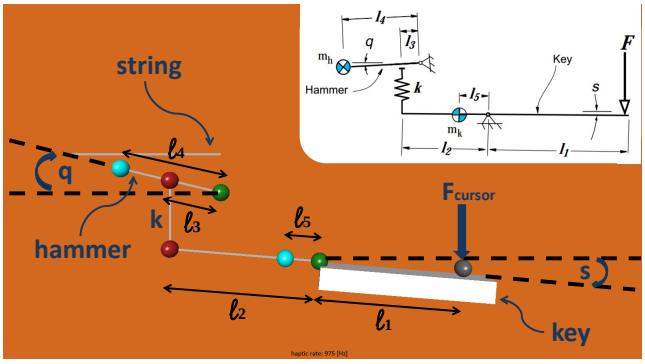

Figure 3. Side view of a single key as rendered in CHAI3d with notations labeled. Green spheres represent fixed fulcrums, blue spheres represent centers of masses, and red spheres represent ends of a compression spring. The dynamic model from Gillespie et al. is inset in the upper right.

interacting with or which are not currently in static equilibrium in each loop.

For an additional sensory modality beyond vision and touch to even further increase the level of immersion in the virtual reality, we used a customized MIDI library to add auditory feedback to the application. When any hammer strikes its respective string, a piano recording for that specific note is played at a volume corresponding to the force with which the user struck the key.

WORKSPACE EXPANSION

While the HMD affords a greatly expanded visual workspace, it also increases the disparity between the physical workspace of the kinesthetic haptic interface and the graphically rendered environment. Scaling the motion of the device can expand the physically reachable workspace but, as with zooming a camera out in the visual rendering, this reduces the resolution and also detriments tactile dexterity. To work around this challenge, we implemented a set of set of device workspace expansion strategies and integrated them into the virtual piano application to test with the HMD.

Button Clutch

The first workspace expansion method we tested with the HMD was the button clutch, which allows the user to freeze the position of the cursor by holding a button while repositioning the device to a different part of the physical workspace. A typical computer mouse provides a clear corollary here, where the clutching is achieved not by holding a button, but rather by picking the mouse up off the table to reposition it. To achieve clutching on a commercial haptic interface, we simply sum all motions when the button is held down and apply that as an offset, O_{clutch}, to the cursor position from the device position, where

$$O_{\text{clutch}} = \sum_i B_{\text{held}} * P_{\text{device}}(i) - P_{\text{device}}(i-1), \qquad (3)$$

and B_{held} is a boolean describing whether or not the user is holding the clutch during timestep i. Figure 4 shows a plot of a user interacting with the full virtual piano keyboard while

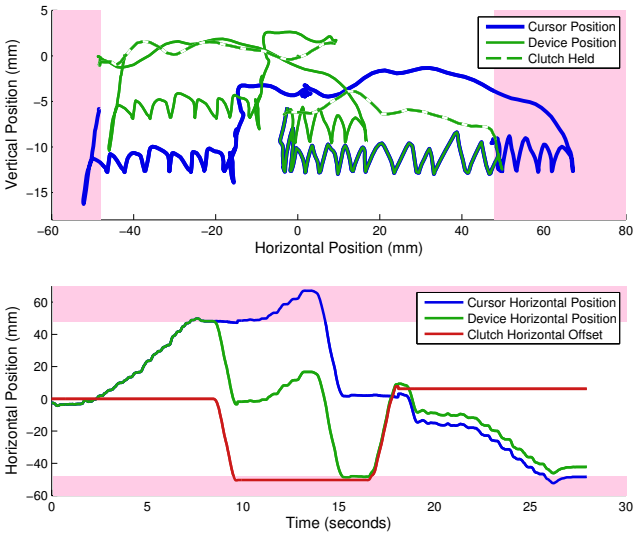

Figure 4. Data collected while a user interacts with the full keyboard using the clutch button to expand the workspace. Pink regions are outside of the physical device workspace so the user must hold the clutch while moving the device back to the center before continuing to explore past these limits.

using the clutch to reposition the haptic device whenever the physical workspace limits are reached.

Workspace Drift

Although the button clutch achieved the desired goal of expanding the physical workspace without sacrificing haptic resolution or tactile dexterity, the conscious effort required by the user to press the button and reposition the device detracted from the immersion into the virtual reality of the HMD. Thus, we tested a technique know as workspace drift [2] to expand the workspace without requiring conscious actions by the user. This relies on the lack of precision in the human's sense of proprioception [5] to use the visual cues to create the illusion that the user is moving his or her hand further than it is actually moving [6]. As explained in greater detail in [2], this algorithm applies an offset between the visual cursor display and the physical device location that slowly accumulates over time so that the user does not notice it drifting. The velocity of the drift

$$\overrightarrow{v_{\text{drift}}} = \frac{k_{\text{d}}}{R} \cdot \|\overrightarrow{v_{\text{d}}}\| \cdot (\overrightarrow{r_{\text{d}}} - \overrightarrow{r_{\text{d0}}}) \qquad (4)$$

is proportional to the velocity of the device, the distance from the physical center of the workspace, the inverse of the radius of the device workspace, and a scaling factor that we set to 0.3, as recommended in [2], so that the user unconsciously corrects the drift by slowly repositioning the device toward the center of the physical workspace while exploring objects near the workspace boundaries. We calculate the offset from the drift by numerically integrating the velocity of the drift. In the virtual piano application, as shown in Figure 5, the user's horizontal steps between keys become smaller near the edge of the keyboard because the visual cursor is slowly drifting further out beyond the workspace boundary and the user unconsciously accommodates for this.

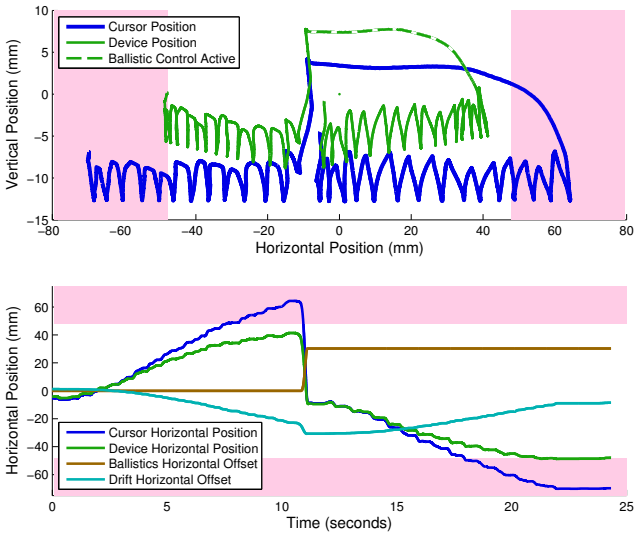

Figure 5. Data from a user interaction with ballistic cursor control enabled above a threshold velocity and workspace drift. The combination of these offsets allows interaction with the full keyboard without requiring any conscious effort from the user.

Ballistic Cursor Control

Workspace drift causes issues if the user wants to quickly move to the other side of the workspace after extended periods near one edge. Even though the drift scales with velocity, the limits imposed to avoid having the user notice its effects prevent the cursor from reacting strongly enough to these attempts to quickly transverse the workspace. To work around this limitation, we implemented another strategy described in [2] that is also used in many common computer mice called ballistic cursor control. Above a certain velocity threshold, we square the velocity of the physical device before integrating it as the position of the visual cursor so that the user can quickly cover large distances by moving the device quickly without necessarily moving it far. This is demonstrated in Figure 5 where the user quickly moved from one side of the keyboard to the other without running into the device workspace limits on either side.

CONCLUSIONS AND FUTURE WORK

This paper presented the integration of haptic feedback into virtual reality for head-mounted displays using a 3-DOF commercial force-feedback haptic device. The implementation allows user to move through a virtual reality graphic rendering while simultaneously interacting with it physically. When the user touches the virtual piano application developed in this work, this interaction is bidirectional in that it both affects the forces that the user feels as well as the state of the virtual environment. The HMD allows both more intuitive interaction as well as the exploration of much larger workspaces without loss of resolution. As this increase in visual workspace is not matched by the device workspace, we implemented a number of control strategies to allow a larger physical interaction workspace without the loss of dexterity from direct motion scaling. The combination of workspace drift and ballistic cursor control provided an intuitive interface that did not require conscious effort actions on the part of the user.

Despite these control strategies for workspace expansion, the physical device workspace remains somewhat limited. As such, future work could implement a button click or other option to recenter the device cursor in the visual workspace, for example when the user has turned his or her head very far in one direction and wants to immediately explore that section of the virtual environment. Furthermore, multiple kinesthetic haptic devices would be necessary to provide more than a single-point tactile interaction. Encountered-type haptic displays [7] [3] may ultimately provide a more intuitive user interface for head-mounted displays by allowing multitouch contact without grounding the user's hand to a device end effector. A controllable stiffness, deformable geometry tactile display integrated with a 6-DOF cable-driven parallel robot with a screen for 3D visualization has been implemented for medical simulation [8], and adding a sensing modality to this tactile interface to integrate it with an HMD could provide a simpler, more immersive simulation.

ACKNOWLEDGEMENTS

The authors thank Francois Conti and Sonny Chan for their help working with CHAI3d and instruction and insight in 3-DOF haptic rendering.

REFERENCES

1. F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, E. Vileshin, J. Warren, O. Khatib, and K. Salisbury. The CHAI libraries. In *Eurohaptics*, pages 193–205, 2003.
2. F. Conti and O. Khatib. Spanning large workspaces using small haptic devices. In *IEEE World Haptics Conference*, pages 183–188, 2005.
3. S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii. inform: Dynamic physical affordances and constraints through shape and object actuation. In *UIST*, pages 417–426, 2013.
4. B. Gillespie. The virtual piano action: Design and implementation. In *International Computer Music Conference*. Citeseer.
5. M. J. Morgan. *Molyneux's question: Vision, touch and the philosophy of perception.* Cambridge U Press, 1977.
6. M. A. Srinivasan, G. L. Beauregard, and D. L. Brock. The impact of visual information on the haptic perception of stiffness in virtual environments. In *ASME Winter Annual Meeting*, volume 58, pages 555–559, 1996.
7. A. A. Stanley, J. C. Gwilliam, and A. M. Okamura. Haptic jamming: A deformable geometry, variable stiffness tactile display using pneumatics and particle jamming. In *IEEE World Haptics Conference*, pages 25–30, 2013.
8. A. A. Stanley, D. Mayhew, R. Irwin, and A. M. Okamura. Integration of a particle jamming tactile display with a cable-driven parallel robot. In *EuroHaptics Conference*, 2014.

9. C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *IEEE/RSJ Intelligent Robots and Systems*, volume 3, pages 146–151, 1995.