

Design of a 2D Joystick for Robot Control Based on a 6 DOF Haptic Device

Daniel J. Brooks, Michael Lunderville, and Holly A. Yanco

Department of Computer Science
University of Massachusetts Lowell
One University Avenue, Lowell MA

dbrooks@cs.uml.edu, michael_lunderville@student.uml.edu, holly@cs.uml.edu

Abstract—We introduce a simple modification which can be applied to a commercial haptic device to convert it into a 2D haptic joystick. We also introduce a simplified haptic API designed to allow rapid development of sophisticated haptic feedback behaviors on our 2D joystick. Finally, we present a case study in which we demonstrate the implementation of a haptic effect for controlling an autonomous robot using our set up. The design files, software, and instructions needed to duplicate our set up are being made publicly available.

I. INTRODUCTION

Autonomous mobile robots are often equipped with sophisticated sensors designed to provide the system with a model of its surrounding environment. This information can then be used for making task-related decisions and conveying information back to the operator. To date, autonomous systems tend to exceed at well defined tasks such as navigation, planning, and obstacle avoidance, usually in fairly structured environments. However, for many current mobile robotic systems, teleoperated control is still largely favored, in part due to a human operator’s sophisticated ability to reason about unstructured environments [1]. Introducing varying levels of autonomy into a teleoperated system allows for a human operator to make high level decisions while leaving other tasks to the autonomy [2]. With this technique, problems can arise when the human operator does not understand why a part of the system they do not have direct control over is behaving in a particular manner (see Fig. 1), usually due to poor situation awareness [3]. Attempts have been made to correct these issues by displaying additional sensor and system state information on the operator control unit (e.g. [4]).

A robot with any level of autonomy must perform as the user expects from the beginning; it is easy for users to lose trust in a system, which is hard to regain [5]. Research shows there are a number of factors which influence a user’s allocation of control to an autonomous robot system including familiarity [6], mental model [6]–[8], control [6], [8], reliability [6], [8], and trust [5]. Haptic feedback could be used to help prevent these issues by giving the user insight to the robot’s state or intentions. Additionally, haptic technology can be used to create new shared control paradigms that more closely match user’s mental models of how the robot is controlled (see Fig. 2).

Haptic interfaces have been used to control many robotic devices such as rotary cranes [9], wheelchairs [10], and surgical equipment [11]. Many different haptic control and feedback strategies have been offered up, and feasibility studies

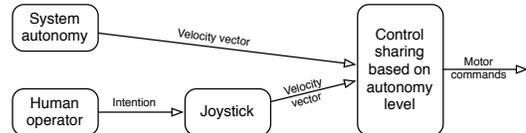


Fig. 1: Traditional Shared Autonomy

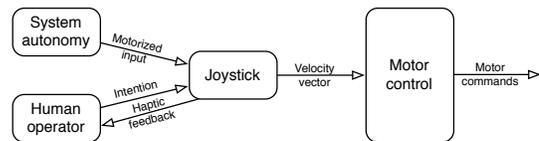


Fig. 2: Shared Autonomy With Haptic Feedback

for these methods have been conducted. Unfortunately, these studies often do not provide sufficient insight for determining the most appropriate strategy for any given task. Environmental feedback strategies try to let users “feel” their surroundings by having nearby objects emit force which is rendered haptically. This force could be a representation of the actual force the remote system is exerting on an object it is in contact with, or be a “force field” generated around sensed objects (also known as artificial force reflection [12]) to prevent contact. Alternatively, behavioral feedback strategies strive to use haptic force to represent the state or intentions of the remote system [13]. Position-position control is a control strategy in which the position of a master device is mapped directly (sometimes with scaling) to a remote slave device position. Position-velocity control is control strategy in which the position of a master device is mapped to velocity information for controlling the movement of a remote slave device [14]. All of these methods have been used for controlling mobile ground robots.

There has been a great deal of work in the field of haptic devices and interfaces, especially with respect to research dedicated to keeping haptic interface systems passive, or stable (e.g. [15]–[18]). Stability is a common problem found in bilateral control systems caused by time-delay occurring in network transmissions, which destabilizes the feedback loop between master and slave devices. Other haptic research has focused on building or implementing custom haptic devices (e.g. [9], [19], [20]). Custom design can be quite challenging and time consuming as it requires building hardware, electronics, and software systems. However, relatively little of this work has concentrated on how haptic interfaces impact end users’ interactions with the system, also called human-robot interaction (HRI). This is especially the case for mobile ground robots.

The emergence of the 6 degree of freedom (DOF) SensAble Phantom [21] and Novint Falcon [22] has allowed many researchers to conduct haptic related research by purchasing commercial devices. Researchers have investigated interface design and the effects of haptic feedback with respect to mobile robots using such unmodified commercial off the shelf (COTS) products (e.g., [23]–[26]). These products provide a very convenient alternative for researchers not wishing to develop sophisticated custom hardware devices, and also offer the advantage of support communities and software APIs [27], [28]. However, using a higher DOF device such as a 6 DOF stylus for controlling a robot only capable of 2D movement may not be as suitable as using a more traditional 2 DOF input device such as a joystick [29], [30].

We have created an easily reproducible and programmable haptic joystick modeled after a more traditional commercial 2 DOF joystick. This simple modification will allow HRI researchers to be able to focus on studying the effects of using different kinds of haptic feedback in robot control interfaces using the traditional interaction modality for the task.

II. DESIGN CONSIDERATIONS

Our goal was to create an easily reproducible and programmable haptic joystick. To facilitate the design process of our system, we considered the following. First, the joystick itself should be based on and have the same characteristics as a popular commercial joystick product. Ideally, a user should not be able to tell the difference between the haptic version (without special haptic effects) and the original product it is based on. This design requirement should allow researchers to establish a solid baseline interface against which they can test various effects. In addition, the system should be easily reproducible. Since constructing a haptic device from scratch is a non-trivial task, we decided a COTS product should be used as a base. All other components should be easily obtainable, with custom components being easily manufactured. Finally, the software should be easy to interface with. As a large portion of the robotics community moves to embrace Willow Garage’s Open Source Robot Operating System (ROS) [31], it seemed an appropriate choice for interfacing with our system.

III. IMPLEMENTATION

Our haptic joystick solution is comprised of 4 parts: a COTS haptic device, the joystick arm, a suspension mount, and software interface.

A. Model Joystick

Our design considerations specified the use of a commercial joystick product whose characteristics should be mimicked by our haptic system. Joysticks can be separated into hand grips (also known as flight sticks), finger sticks (arcade size), and thumb controls (such as those found on modern video game pads). Our calculations have shown that the Phantom Omni does not provide enough force for use with a hand grip, which we verified with a prototype joystick based on a CH Products Flight Stick [32]. Thumb controls are usually associated with handheld controllers, and incorporating one into a stationary fixture seemed unnatural. Thus, our final design has been based on the CH Products M11L061P (Fig.

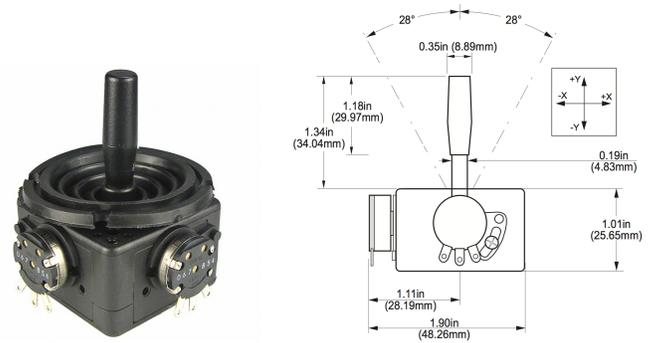


Fig. 3: CH Products M11L061P finger joystick [34] [35]



Fig. 4: SensAble Phantom Omni [36]

3) finger joystick [33]; a joystick commonly found on CCTV products but also used in many other applications including robot control, such as the Inuktun VGTV operator control unit. We refer to this product as our model joystick. It should be noted that our final joystick makes no use of this actual product; rather, we have created a haptically enabled imitation of it.

B. Haptic Engine

As previously mentioned, building a custom haptic device is difficult process that requires knowledge and skills in mechanical engineering and electrical design. Building an adapter for an existing product circumvents this complexity, and is far easier to replicate. We elected to use the SensAble Phantom Omni (Fig. 4) as the base from which to build our system. The Phantom Omni haptic device is a 6 degree of freedom (DOF) device already widely used in haptics research that comes with the OpenHaptics Toolkit API [27], which we discuss in Section III-E. The disadvantages of this device are that its many degrees of freedom and stylus grip make it unsuitable for use as a traditional 2 DOF joystick without modification, and its size is quite bulky compared to most traditional joysticks. Additionally, the Phantom’s API has been designed for use in a virtual workspace, which, if not handled appropriately, can cause unexpected problems in calculations done on points in the real world, which we further discuss in Section III-E.

C. Joystick Arm

An important aspect of a joystick is its physical characteristics [37]. The topmost section of the joystick arm is the part the user will regard as “the grip.” Extending up through the surface of the user interface, it is capped with an ABS plastic handle and is stylized to be the same dimensions as our model joystick.

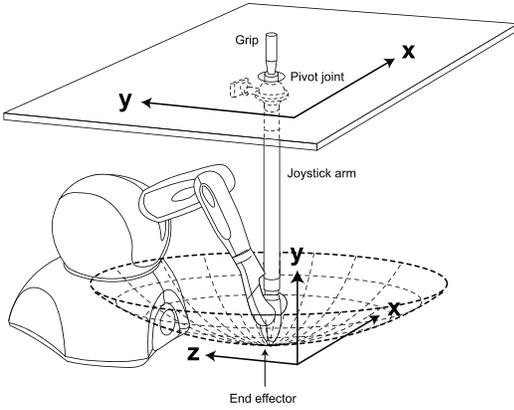


Fig. 5: The end effector moves about the surface of a sphere

Two other specific characteristics of particular but not obvious importance are 1) the physical boundaries of the device which restrict movement beyond certain points, and 2) the spring mechanism which serves as a centering force. In prior experiments, we have observed that users tend to use the physical boundaries of the joystick while driving, which seems to be easier than holding the joystick at an intermediate position. Just below the surface of the user interface, located midway along the arm, is a pivot joint. The pivot joint is located in the same position along the joystick that potentiometers are mounted on our model. Our pivot joint has a 65° circular range of motion which closely approximates the model joystick's 55° square range of motion, which is dampened by a circular rubber boot. Users also tend to push the joystick from one side, relying on the centering force to hold the stick in position. We have reproduced this effect using haptics, as discussed in Section III-E.

The bottom of the arm is an adapter connecting to the end effector of the Phantom Omni. This allows us to simultaneously measure the position of the joystick and generate haptic forces. The end effector's movement are restricted to rotating spherically below the pivot joint (see Fig. 5), constraining the Phantom's 6 DOF. We have calculated the length of the joystick arm to maximize the end effector's range of motion, thus also maximizing mechanical advantage over the user and providing the highest resolution for haptic effects.

The arm itself is constructed of grade 5 titanium and ABS plastic. Titanium was selected based on its light weight and strength. The titanium parts consist of one 4.75" long $3/16$ " round rod and one 3.25" long, $1/2$ " diameter tube. The pivot joint is a $1/2$ " PTFE lined ball joint rod end. The titanium and ball joint can be purchased from McMaster-Carr (part numbers #89055K321, #6960T61, and #6960T11), and the titanium can be cut to length. ABS plastic fittings connect the pieces of the arm together (see Fig. 7), and are printed using a 3D rapid prototyping machine. The arm is held together using 1mm pins inserted through holes in the fittings and across notches in the rod.

D. Suspension Mount

A special mount has been designed to hold the Phantom Omni suspended beneath the user interface. The mount po-

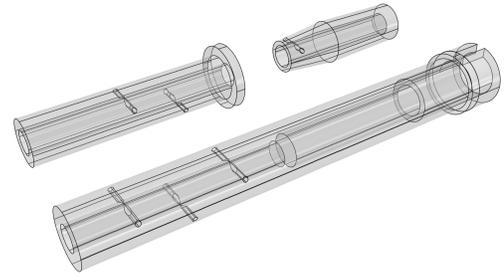


Fig. 6: ABS Plastic Parts. Counter-clockwise from the top right: Joystick Grip, Pivot Adapter, Omni End Effector Adapter

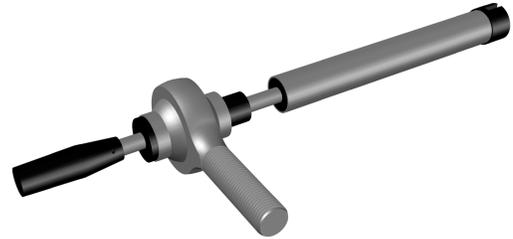


Fig. 7: Arm Assembly. Plastic parts from Fig. 6 are shown in black, metal parts are shown in grey.

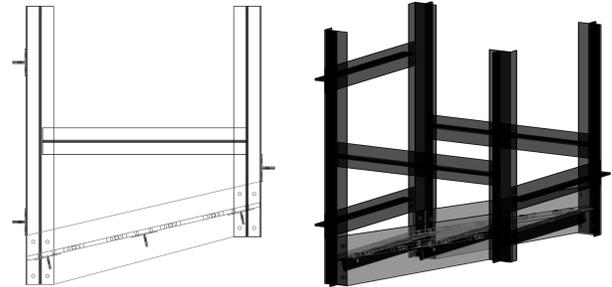


Fig. 8: Suspension Mount. *Left: Side View, Right: 3D View*

sitions the device a carefully measured distance below the pivot joint, ensuring that the end effector's movements are restricted by the range of motion of the pivot and not the physical constraints of the Omni. Additionally, we found that the haptic device has the smoothest range of motion when mounted at a slight incline, approximately 13 degrees.

The mount is sufficiently rigid so that energy from the haptic devices is not absorbed by the movement of the mount. The final design consists of an inclined plastic plane, suspended by four aluminum "T" shaped columns spaced wide enough apart to not cause interference with the movement of the end effector. The structure is stiffened with plastic and aluminum bracing on all four sides to reduce movement. The entire mount is constructed from $1/4$ " laser cut plastic, $3/4$ " aluminum angle, and $3/16$ " diameter blind rivets.

E. Software

The Phantom Omni comes with the powerful OpenHaptics Toolkit [27], consisting of Phantom Device Drivers, a Haptic Device API, and Haptic Library API. The Haptic Library API has been designed to compliment OpenGL, allowing scenes

rendered in OpenGL to be given physical properties which can be explored through the haptic device. This API is not well adapted for writing software to control a joystick, as this was not the original intention for the device. The Haptic Device API (HDAPI) is a low-level API which can be used to read raw information from the device and control motors. This is a powerful API for controlling the device; however, it was also written for use with a virtual environment and is cumbersome for controlling our joystick set up.

Traditionally, the mobile ground robot driving task has been performed using 2 DOF joysticks, controlling the linear (forward) and angular (turning) velocities along two axes lying on a plane. Using SensAble's HDAPI, we have created a new API which not only exposes HDAPI's low-level control as a ROS node (haptic driver), but allows us to read and control the joystick grip position in 2D (Haptic Joystick).

For the `haptic_driver` node, we found it necessary to calculate the end effector's position rather than use the coordinates reported by HDAPI, because the reported coordinates lie along curved axes. To get the end effector's position in a Cartesian coordinate system suitable for measuring position in the real world, we used the angles of each joint that are output by the device to calculate our own three dimensional coordinates. This was done simply by using trigonometry and knowing the lengths of each part of the arm. The angle measurements output by the device – turret, thigh, and shin (see Fig. 4) – are based on their starting position. When the device is first plugged in, the end effector must be in its dock. We shift these measurements so that all angles are based on a horizontal plane. By measuring the lengths of the shin L_s and thigh L_t (where $L_t = L_s = 133.35$), the values of x , y , and z coordinates are computed as

$$\begin{aligned} x &= L_t \cdot \cos(\text{thigh_ang}) \cdot \sin(\text{turret_ang}) \\ &\quad + L_s \cdot \cos(\text{shin_ang}) \cdot \sin(\text{turret_ang}) \\ y &= L_t \cdot \sin(\text{thigh_ang}) + L_s \cdot \sin(\text{shin_ang}) \\ z &= L_t \cdot \cos(\text{thigh_ang}) \cdot \cos(\text{turret_ang}) \\ &\quad + L_s \cdot \cos(\text{shin_ang}) \cdot \cos(\text{turret_ang}) \end{aligned}$$

The `haptic_driver` ROS node publishes information about the device's status over the `/haptic/status` topic in an `OmniStatus` message. The `OmniStatus` message includes the device end effector's 3D pose in Cartesian coordinates, the current velocity of the end effector, and information provided by the HDAPI such as the `turret_angle`, `thigh_angle`, `shin_angle`, etc. Forces can be sent to the device on topic `/haptic/feedback` as type `geometry_msgs::Vector3`, specifying the x , y , and z force components. Finally, the `haptic_driver` has been designed to robustly idle in the absence of a ROS master, allowing it to optionally be started independently and left running while the rest of a ROS system is brought up and down.

The 2D haptic joystick API allows for more abstract control of the joystick without loss of functionality, allowing researchers to focus on the nuances of creating haptic joystick behaviors. This is accomplished through the use of a calibration routine and 3D to 2D coordinate transformation. The end effector's movement is restricted to the surface of an imaginary sphere centered about the pivot joint (See Fig. 5). Calibration

is required to establish joystick positioning and orientation in our coordinate system, and it consists of moving the joystick to centered, forward, backward, left, and right positions, then moving the joystick about randomly.

This routine allows us to define two line segments between opposing maximum points and collect a set of points lying along the spherical range of motion of the end effector. The points collected are passed through a voxel filter, yielding an even distribution. To estimate the center and radius of the sphere, which corresponds to the location of the pivot joint and length of our joystick arm, we minimize the difference between the radius and each point's distance from the sphere's center using least squares. Letting $R = x_c^2 + y_c^2 + z_c^2 - r^2$ where x_c , y_c , and z_c are the coordinates of the sphere's center and R is its radius, the equation of the sphere becomes

$$-2x \cdot x_c - 2y \cdot y_c - 2z \cdot z_c + R = -x^2 - y^2 - z^2$$

Solving this for n points gives the matrix equation

$$\begin{pmatrix} -2x_1 & -2y_1 & -2z_1 & 1 \\ -2x_2 & -2y_2 & -2z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -2x_n & -2y_n & -2z_n & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ R \end{pmatrix} = \begin{pmatrix} -x_1^2 - y_1^2 - z_1^2 \\ -x_2^2 - y_2^2 - z_2^2 \\ \vdots \\ -x_n^2 - y_n^2 - z_n^2 \end{pmatrix}$$

A linear algebra approach to least squares gives us estimated values of

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ R \end{pmatrix} = \left[\begin{pmatrix} -2x_1 & -2y_1 & -2z_1 & 1 \\ -2x_2 & -2y_2 & -2z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -2x_n & -2y_n & -2z_n & 1 \end{pmatrix}^T \begin{pmatrix} -2x_1 & -2y_1 & -2z_1 & 1 \\ -2x_2 & -2y_2 & -2z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -2x_n & -2y_n & -2z_n & 1 \end{pmatrix} \right]^{-1} \begin{pmatrix} -x_1^2 - y_1^2 - z_1^2 \\ -x_2^2 - y_2^2 - z_2^2 \\ \vdots \\ -x_n^2 - y_n^2 - z_n^2 \end{pmatrix}$$

Note that this equation is always solvable unless all n points are coplanar. The orientation is calculated by transforming the forward-backward and left-right line segments into the base of an upside-down regular square pyramid inscribed in the calculated sphere. This is done by shifting the maximum points defining these line segments so the segments intersect with each other and the radius touching the sphere's bottom (defined by the point collected when the joystick is centered). The segments are shifted again so that each maximum point is equidistant from the intersection point. This is followed by another shift forcing the maximum points to create a plane that is normal to the radius touching the sphere's bottom (defined by the joystick's centered position) and so the segments are at a right angle to each other. Finally, the points are projected onto the sphere's surface using a ray through each point from the sphere's center.

With this calibration, we can then calculate the 2D joystick position corresponding to any 3D end effector position and vice versa. Converting 3D to 2D, the point is first projected onto the plane made by the sphere's center and the forward-backward axis. The planar angle is then calculated as the angle between the ray made by the sphere's center and bottom and the ray made by the sphere's center and the projected point. The forward-backward output coordinate is measured as the percentage of this angle to the maximum angle possible, giving a range of -1 to 1. A similar calculation is done to find the left-right output coordinate. An algorithm performing the inverse of this operation is used to convert in the opposite direction.

The joystick can be controlled haptically by specifying the (X, Y) coordinate to which the joystick should travel and the amount of force to use in getting there. This is done by using a “gravity well” to pull the joystick toward a certain point. This effect is based on Hooke’s Law and made by finding the vector from the current actual position to the target position. A force is applied in that direction with a magnitude proportional to the length of that vector and scaled by the input magnitude of that target. The actual force vector calculation is

$$\mathbf{F} = \frac{\min(|\mathbf{v}|, d_{thresh})}{d_{thresh}} \cdot ((F_{max} - F_{spring}) \cdot m + F_{spring}) \cdot \frac{\mathbf{v}}{|\mathbf{v}|}$$

where \mathbf{v} is the vector from the current position to the current target, F_{max} is the maximum force allowed, F_{spring} is the minimum force allowed that creates a spring like effect, m is the input magnitude, and d_{thresh} is the distance after which $|\mathbf{v}|$ is no longer used to scale the force. m ranges from 0 to 1 where 0 yields just a spring force, and 1 yields the strongest force allowed. By setting a weak amount of force centered at the origin, we can create the spring centering effect mentioned in Section III-C.

The `haptic_joystick` node provides the 2D control interface used to program haptic behaviors for the joystick. The 2D joystick position can be read by subscribing to the topic `/joy_pos` which is encoded using the standard ROS convention of `Twist` messages. The forward-backward values are stored in `linear.x`, and the left-right value in `angular.z`; values range from -1.0 (back or left, respectively) to 1.0 (forward or right). Commanding the 2D joystick can be done by publishing a message to the topic `/joyfeedback`. The `JoystickInput` message type specifies the joystick’s target linear and angular positions (-1.0 to 1.0), and the amount of force to use as a magnitude (0 to 1). `take_control` (boolean) is used to toggle the joystick between haptic and non-haptic modes.

IV. CASE STUDY

We are currently in the process of performing user studies to explore the effects of haptic feedback on situation awareness [3] using our hardware and software implementation described in Section III. In the study, participants are asked to navigate a remotely located semi-autonomous robot down a hallway with eight obstacles evenly spaced down the center. Each obstacle has an arrow on it indicating which side the robot should pass. The robot is capable of autonomously navigating the course through the use of preset way-poses which allows it to help the participant with the task of driving. The participant is encouraged to rely on this feature not only due to the difficulty of depth perception when driving by camera, but also due to needing to perform a secondary targeting task which forces them to move the camera. The joystick has absolute control over the wheels, and both the participant and the onboard autonomy must control the robot using the joystick. We have programmed the steering algorithm to be compliant; that is, the robot will exert increasing amounts of force on the joystick to get the participant to change directions, but can also revise its route to match the participant’s intended course if they remain insistent. The robot has been programmed to occasionally make a wrong turn in order to cause this scenario to occur, but performs correctly close to 90% of the time to encourage trust [5].

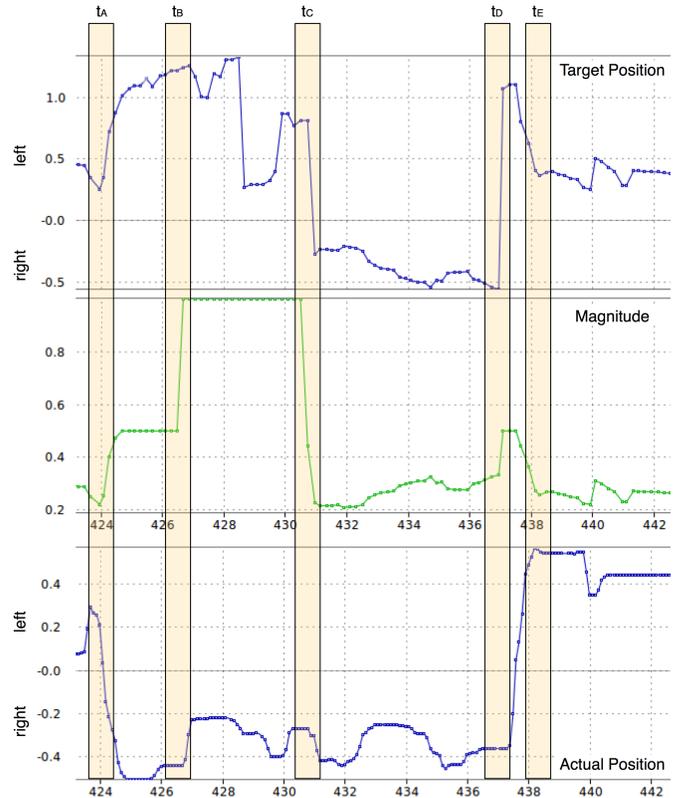


Fig. 9: Haptic forces. *Top*: Target joystick angular position, *Middle*: Magnitude of force (m), *Bottom*: Actual joystick angular position.

The graphs in Fig. 9 were taken from data collected during our study and show the angular position of the joystick as well as the haptic target and force magnitude being exerted over time. In this scenario, the robot is initially steered towards the left (incorrect) side of an approaching obstacle, with the robot’s autonomy intentionally encouraging the behavior. The participant, identifying the mistake, begins to correct this by pushing the joystick to the right (t_A). The robot’s autonomy, still believing the robot should be heading left, responds by increasing the force pushing the joystick left. As the participant deviates further from the autonomy’s desired direction, it increases the force pushing left again (t_B) in an attempt to change the participant’s actions. However, as the participant persists in contradicting the autonomy, the robot modifies its route to match that of the participant (t_C). As the robot crosses the front of the obstacle, it becomes necessary to turn left to continue down the hallway. The robot again exerts a leftward force (t_D), to which the participant complies. As the participant and autonomy both become satisfied with the rate of turn, the autonomy applies a weak centering force (t_E) to simulate the physical springs of the model joystick.

V. CONCLUSIONS AND FUTURE WORK

We have presented a haptic joystick which has been designed for easy replication and a simplified API for joystick control. We have used this setup in the implementation of a user study, which we described in Section IV. Information about duplicating the system is available on our website¹.

¹<http://www.robotics.cs.uml.edu/haptics>

We will continue using our set up to investigate how various styles of haptic interfaces impact user interactions with mobile ground robots. One topic of particular interest which could be studied through the use of haptic interfaces is perceived agency. Prior to an initial interaction with an intelligent system, users may have preconceptions about how the system should behave, which can lead to frustration in future ones. Some users may believe the robot is a tool that should perform tasks on command. Others might see it as an intelligent agent or “being” they can work collaboratively with to achieve a common goal [38]. Still others may see it as an agent, but one whose primary function is to act as a tool (e.g., a service animal). Haptic interfaces could be interpreted as affording mental state, a trait that Luck and d’Inverno note as the minimal requirement for a system to be considered an agent [39]. By better understanding what users believe or think haptic feedback is and what it means, we will be able to design more intuitive haptic effects and possibly even change the way a user thinks about the system they are using.

VI. ACKNOWLEDGMENTS

This work was funded by an Army Research Office MURI (W911NF-07-1-0216). PHANToM Omni Haptic Device provided courtesy of SensAble Technologies Inc. Thank you to Kate Tsui for her assistance in this work.

REFERENCES

- [1] T. B. Sheridan, *Telerobotics, automation, and human supervisory control*. MIT Press, 1992.
- [2] T. B. Sheridan, “Human and Computer Control of Undersea Teleoperators,” Jul. 1978.
- [3] M. Endsley, “Automation and situation awareness,” *Automation and human performance: Theory and applications*, 1996.
- [4] H. Yanco, B. Keyes, J. Drury, C. Nielsen, D. Few, and D. Bruemmer, “Evolving interface design for robot search tasks,” *Journal of Field Robotics*, vol. 24, 2007.
- [5] M. Desai, “Modeling trust to improve human-robot interaction,” Ph.D. dissertation, University of Massachusetts Lowell, December 2012.
- [6] A. Steinfeld, “Slightly subversive methods for promoting use of autonomy in robots,” in *RSS Workshop On Human-Robot Interaction: Perspectives And Contributions To Robotics From The Human Sciences*, 2011.
- [7] T. Carlson and Y. Demiris, “Using visual attention to evaluate collaborative control architectures for human robot interaction,” in *New Frontiers in Human Robot Interaction, a symposium at AISB*, 2009.
- [8] D. Norman, “How might people interact with agents,” *Communications of the ACM*, vol. 37, no. 7, 1994.
- [9] A. Takemoto, K. Yano, T. Miyoshi, and K. Terashima, “Operation assist control system of rotary crane using proposed haptic joystick as man-machine interface,” in *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, 2004.
- [10] R. C. Luo, C.-Y. Hu, T. M. Chen, and M.-H. Lin, “Force reflective feedback control for intelligent wheelchairs,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1999.
- [11] A. M. Okamura, “Methods for haptic feedback in teleoperated robot-assisted surgery,” *Industrial Robot: An International Journal*, vol. 31, no. 6, 2004.
- [12] S.-G. Hong, J.-J. Lee, and S. Kim, “Generating artificial force for feedback control of teleoperated mobile robots,” in *IROS*, 1999.
- [13] D. Barnes and M. Counsell, “Haptic communication for mobile robot operations,” *Industrial Robot: An International Journal*, vol. 30, no. 6, 2003.
- [14] I. Farkhatdinov, J. Ryu, and J. Poduraev, “Control Strategies and Feedback Information in Mobile Robot Teleoperation,” *17th World Congress of the International Federation of Automatic Control*, vol. 1, 2008.
- [15] R. J. Anderson and M. W. Spong, “Bilateral control of teleoperators with time delay,” in *IEEE Transactions on Automatic Control*, May 1989.
- [16] I. Elhajj, N. Xi, and Y.-H. Liu, “Real-time control of Internet based teleoperation with force reflection,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.
- [17] B. Hannaford and J.-H. Ryu, “Time-domain passivity control of haptic interfaces,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, Feb. 2002.
- [18] G. Niemeyer and J.-J. E. Slotine, “Stable Adaptive Teleoperation,” *IEEE Journal of Oceanic Engineering*, vol. 16, no. 1, Jan. 1991.
- [19] S. K. Cho, H. Z. Jin, J. M. Lee, and B. Yao, “Teleoperation of a Mobile Robot Using a Force-Reflection Joystick With Sensing Mechanism of Rotating Magnetic Field,” in *IEEE/ASME Transactions on Mechatronics*, 2010.
- [20] S. Han and J. Lee, “Tele-operation of a Mobile Robot Using a Force Reflection Joystick with a Single Hall Sensor,” *RO-MAN*, 2007.
- [21] SensAble Technologies Inc., “PHANToM Omni,” 2011, <http://www.sensable.com/haptic-phantom-omni.htm>, accessed Nov. 30, 2011.
- [22] Novint Technologies Inc., “Novint Falcon,” 2013, www.novint.com/index.php/novintfalcon, accessed Jan. 14, 2013.
- [23] F. Schill, R. Mahony, P. Corke, and L. Cole, “Virtual force feedback teleoperation of the InsectBot using optical flow,” in *Australasian Conference on Robotics and Automation*, J. Kim and R. Mahony, Eds., 2008.
- [24] I. Farkhatdinov and J. Ryu, “Improving Mobile Robot Bilateral Teleoperation by Introducing Variable Force Feedback Gain,” *IROS*, 2010.
- [25] N. Diolaiti and C. Melchiorri, “Haptic tele-operation of a mobile robot,” in *Robot control 2003: 7th IFAC Symposium (SYROCO’03)*, Sep. 2003.
- [26] N. C. Mitsou, S. V. Velanas, and C. S. Tzafestas, “Visuo-Haptic Interface for Teleoperation of Mobile Robot Exploration Tasks,” in *RO-MAN*, Sep. 2006.
- [27] SensAble Technologies Inc, “OpenHaptics Toolkit,” 2011, <http://www.sensable.com/products-openhaptics-toolkit.htm>, accessed Nov. 30, 2011.
- [28] Kyle Machulis, “libnifalcon,” 2010, qdot.github.com/libnifalcon, accessed Jan. 14, 2013.
- [29] D. A. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa, and Stuerzlinger, “3D User Interfaces: New Directions and Perspectives,” *Computer Graphics and Applications, IEEE*, vol. 28, no. 6, 2008.
- [30] J. F. Lapointe, P. Savard, and N. G. Vinson, “A comparative study of four input devices for desktop virtual walkthroughs,” *Computers in Human Behavior*, vol. 27, no. 6, Nov. 2011.
- [31] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System,” *ICRA Workshop on Open Source Software*, 2009, <http://www.ros.org>.
- [32] D. J. Brooks and H. A. Yanco, “Design of a Haptic Joystick for Shared Robot Control,” in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2012.
- [33] CH Products, “M Series Miniature Resistive Joysticks,” www.chproducts.com/files/chproducts/brochures/M_CH_low-res.pdf, accessed Jan. 14, 2013.
- [34] Digikey, <http://media.digikey.com/photos/APEM%20Comp%20Photos/M11L061P.jpg>, accessed Jan. 14, 2011.
- [35] Digikey, <http://media.digikey.com/PDF/Data%20Sheets/APEM%20Components%20PDFs/M11L061P.pdf>, accessed Jan. 14, 2011.
- [36] SensAble Technologies Inc., <http://www.sensable.com/documents/images/LargePHANTOMOmniImage.jpg>, accessed Jan 14, 2011.
- [37] B. D. Kechavarzi and S. Sabanovic, “Evaluation of control factors affecting the operator’s immersion and performance in robotic teleoperation,” *RO-MAN*, 2012.
- [38] T. Fong, C. Thorpe, and C. Baur, “Collaboration, dialogue, human-robot interaction,” *Robotics Research*, 2003.
- [39] M. Luck and M. d’Inverno, “A formal framework for agency and autonomy,” in *International Conference on Multi-Agent Systems*, 1995.