TELEPRESENCE: DESIGN, IMPLEMENTATION AND STUDY OF AN HMD-CONTROLLED AVATAR WITH A MECHATRONIC APPROACH

[1]DARREN MICHAEL CHAN, [2]JANE ZHANG

[1,2]Electrical Engineering Department, California Polytechnic State University,
San Luis Obispo, CA 93407, USA
E-mail: [1]dmchan@calpoly.edu, [2]jzhang@calpoly.edu

Abstract—This paper presents a solution for visual Telepresence, where an HMD is used to control the direction of a camera's viewpoint, such that the HMD's rotational coordinates are tracked by a mechanical avatar. The design and development of the avatar's hardware and software follows a mechatronic approach, where a real time operating system (RTOS) is used in conjunction with a microcontroller for mechanical actuator control. Experimental results show that the design provides adequate head-tracking capabilities as a proof of concept for use in Telepresence applications.

Keywords—avatar, Head-mounted Display, head-tracking, Oculus Rift, Telepresence

I. INTRODUCTION

Developments in electronic displays and vision systems continue to push the boundaries of communication, medicine, military, entertainment and consumer devices. Within the past decade, lesser charted areas of research, viz., Virtual Reality (VR), Augmented Reality (AR) and Telepresence have gained attention along-side the advancement of digital video and graphics systems. Recent innovations by Sony and Oculus VR include VR Head Mounted Displays (HMDs) available at consumer-level prices. With the advent of Virtual Reality technology in full development, it follows that closely related fields, viz. Augmented Reality and Telepresence also benefit.

Telepresence describes technologies that allow humans to gain the sensation of presence at an event without being physically present. In other words, it describes the means by which a user can experience the sensation of presence, remotely. Though Telepresence extends to visual, auditory and other experiences relating to the human senses, this paper introduces a novel implementation of a Telepresence device that consists of two parts: an HMD and a mechanical gimbal, or avatar.

II. RELATED WORK

There has been little research in the field of Telepresence due to limitations in the state of technology [1,2], but more current developments in faster, more powerful computing hardware have begun to erode such limitations. While there are many relevant topics in Telepresence that must be confronted, this paper primarily centers on head-tracking via mechanical 3-axis gimbal. Perhaps one of the most similar pieces of technology that is commonly employed, that relates to the work highlighted in his paper, is the integration of Helmet Mounted Displays in military aircraft control systems [3]. When used in fighter aircraft, the Helmet Mounted Display allows the user to aim his weapon using head movements. Although similar in principle, where actuators are controlled by head movements, this paper discusses the implementation of a device that is intended for Telepresence applications.

HMD-based head-tracking robots have been used for study, viz. Kratz et al. [4] and Martins et al. [5], where they have been purposed for teleoperation and navigation of mobile robots. However, this paper deviates from teleoperation for navigational applications and instead examines the fidelity of presence for head-tracking gimbals on stationary platforms. It should be noted that while Martins et al. and Kratz et al. developed Telepresence apparatuses that present similarities to the methods described in this paper, the device developed for this paper differs by employing a gimbal that provides head-tracking for 3 degrees of freedom in a stationary setting, where each axis is independently controlled.

III. CONCEPT

An avatar is a physical or nonphysical representation of a user that directly interacts with an environment that is not immediate to the user. For example, in Virtual Reality, an environment is simulated by a computer; the user is able to directly interact with the computer, but is unable to directly interact with the simulated environment– instead, an avatar represents the user in that simulated environment and is, by proxy, controlled by the user.

In Telepresence, the environment and avatar have physical forms where the avatar is remotely controlled by the user. An avatar within the context of Telepresence is also tasked with collecting information about its surroundings, in addition to being remotely controlled. The information is then delivered to the user in the form of haptic feedback, or information that pertains to the human senses. All of these responsibilities must be performed in real

time– and for a high performance system, in such a way that the user does not detect a delayed response.
The device described in this paper is designed to utilize an HMD, which delivers Yaw, Pitch and Roll (YPR) coordinates to the avatar as inputs, where the avatar behaves to mimic the HMD movements.
At the time of writing, no virtual reality HMD exists to be compatible with low-level devices (viz. microcontrollers), but instead, a PC that runs a native operating system is required to interface the two parts. An HMD transmits device data to the PC, wherein the PC decodes the data to be transferred to the avatar. The avatar responds to the received YPR coordinates by moving its actuators to track HMD's position.

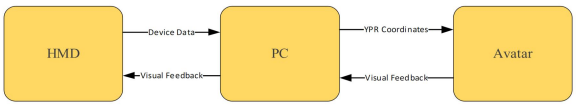


Fig. 1. High level system diagram showing connections of the HMD, PC and Avatar.

Given the limitations in current technology, viz., long range communication, the prototype is designed with several constraints:

1. The device is designed for short range study, such that the device is physically tethered to a PC.
2. The gimbal is mounted on a stationary platform, rather than on a mobile robot.
3. The HMD and avatar tracks head rotation, rather than head position.
4. We acknowledge that some problems exist with the first revision of the prototype, where a second iteration is improved upon the first. However, aspects of the first prototype contribute to the state of the art by providing substantial results to be carried to future designs.

IV. PROTOTYPE - HARDWARE

A 3-axis camera-stabilization gimbal was purchased off the shelf to use as the robotic platform, which consists of the avatar's mechanical parts. The gimbal is driven by three 3-phase brushless DC motors, each of which control one of three gimbal axes (Yaw, Pitch and Roll). 3-phase brushless DC motors were selected due to their ability to produce high torque for their compact size and have the ability to provide smooth rotation, which are desirable qualities for camera stabilization.
Grossman et al. [6] determined that maximum rotational head velocities for humans, while vigorously shaking their heads, are 780deg/s and 380deg/s for the horizontal and vertical directions, respectively. It is assumed that the device operator will not be shaking his head vigorously for extended periods within the duration of its use. Therefore, Grossman et al.'s findings serve as conservative estimates for the required motor speeds, which equate to motor specs of 130rpm for the Yaw motor, and 63.3rpm for the Pitch and Roll motors.
An inertial measurement unit (IMU) and encoder provide positional feedback to a controller, which correct for position error, or the difference between the motors' goal positions and their measured positions. The controller is designed using an Atmel ATmega2560 microcontroller, which was selected due to its large number of timer/counter pins– each motor requires three PWM signals to be driven. The board features three Half H-bridge drivers (SN754410) that source necessary power to drive the DC motors, and a UART (FTDI FT232RL) that allows the HMD (via USB-PC) to communicate with the avatar.
The Oculus Rift DK2 (Development Kit 2) was selected as the HMD to be interfaced to the prototype due to its low cost, high performance and reliability. The Oculus Rift DK2 features high resolution imaging at 1080p, is designed for rotational tracking with an update rate of 1000Hz, and has a considerably lower price tag (at time of writing) compared to alternative virtual reality headsets.

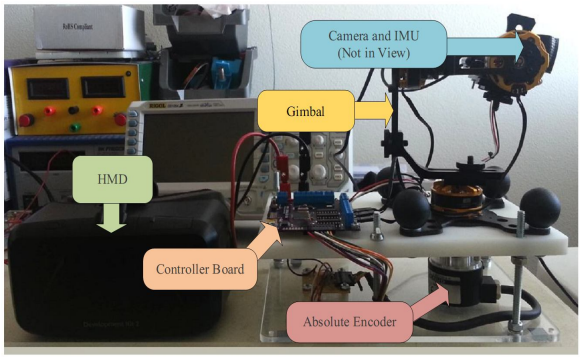


Fig. 2. The Avatar and all of its critical components.

V. PROTOTYPE – SOFTWARE

The microcontroller integrates a Real Time Operating System (RTOS) to schedule events that control the avatar's behavior. These events are known as tasks, to which an RTOS behaves as an instrument to handle multitasking, or the running of multiple tasks at the same time– in actuality, these tasks are handled by a single processor, such that the tasks run in quick succession to appear as though they are running simultaneously. Task timing refers to how often the processor should switch to that task, and priority indicates which task is ordered to run according to the scheduler. Three tasks are created to form the controller:

1. Read HMD– This task reads and parses the coordinates provided by the HMD via UART (connected to the PC).
2. Read IMU– This task reads and parses the coordinates provided by the IMU module.

3. Motor Controller – This task uses the positional data gathered by the Read HMD and Read IMU tasks and actuates the gimbal motors to correct for error.

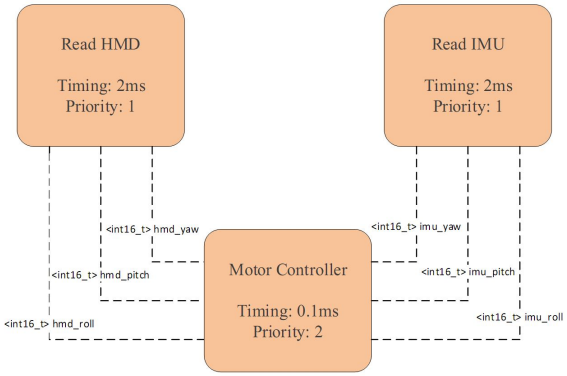


Fig. 3. Task diagram showing data that is communicated between tasks. Higher priority value indicates higher scheduling priority.

Data gathered by the Read HMD and IMU tasks are passed in a thread-safe manner to the Motor Controller task. The RTOS is configured for preemptive mode at 10KHz tick rate (0.1ms sampling).
The following subsections discuss each of the Tasks shown in Figure 3.

1. Read HMD Task
The Read HMD Task reads data received from the serial port; the rotational coordinates of the HMD are preserved to two decimal places and converted to ASCII for serial transmission. Delimiters to the data stream must be added between coordinates, so that the task can correctly attribute the data to its corresponding axis. This, the ASCII string is formatted in the form:

Y#P#R#E (1)

When the task is called to read the data stream, it is unknown where it will begin. Hence, the Y, P and R characters are inserted between the encoded data (#), where the data following that delimiter corresponds to that axis (Y for Yaw, etc.). The "E" character indicates the end of the data stream.

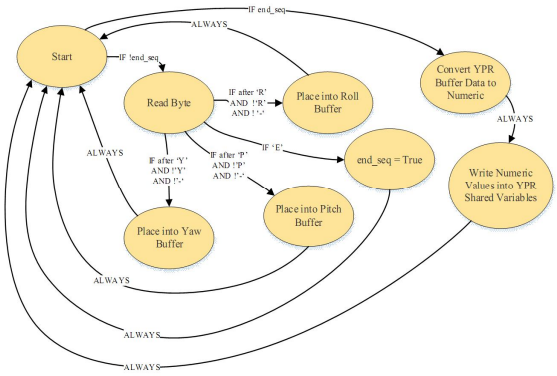


Fig. 4. State diagram for Read HMD task.

Each character is read from the UART, one at a time, and placed into its corresponding coordinate buffer, which is parsed using the flag that is set to trigger from the last detected delimiter. Only when all of the data has been collected, does the state reset– this ensures that all the data has been collected from a single measurement via the HMD.

2. Read IMU Task
The Read IMU Task behaves similarly to that of the Read HMD Task, in that the IMU data is output in the same format as (1). However, one minor difference is the inclusion of a process that reads the measured value from the encoder; this is necessary, because we discovered that the magnetic fields produced by the motors interferes with the magnetometer readings, rendering the YAW measurements from the IMU unusable.

3. Motor Controller Task
The Motor Controller Task responds to the coordinate values collected by the Read HMD and Read IMU tasks by controlling the behavior of the gimbal motors – the IMU is mounted on the avatar in such a way that motors move to eliminate error, or the difference between the HMD (goal position) and IMU coordinates (current position).
Unlike traditional DC motor drivers that require a single PWM signal for motor velocity control, the prototype utilizes three PWM signals per motor (one per phase). In the ideal case, 3-phase sinusoidal signals are delivered to the windings of the motor. However, since the power must be sourced via the Half H-bridge drivers and is driven by a microcontroller, the 3-phase sinusoids must be broken into discrete (time and voltage) steps– this behaves similar to micro-stepping technology that is commonly deployed in stepper motor controllers. Figure 5 depicts a three sinusoidal PWM signals output by the microcontroller timers.

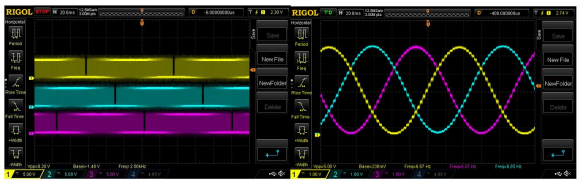


Fig. 5. 3-phase sinusoidal signals generated by PMW (left) and its analog equivalent, filtered by a low pass filter (right).

Unlike most typical motor drivers, the motor velocity is proportional to the frequency of the 3-phase signals, rather than PWM duty cycle. Thus, the Motor Controller task timing significantly contributes to the way that the motor is controller. An internal counter is integrated to this task and is set or reset to control the motor speed; tighter task timing allows for greater speed resolution. As with typical motor controllers, some gain value is calculated based on the motor's currently measured position to some goal position, to dynamically change the motor's velocity.

The prototype utilizes a Proportional-Integral-Derivative (PID) controller, where each component is broken into the following equations:

$$K_p = \rho \times err(t) \quad (2)$$

$$K_I = I \times \int_0^{\tau} err(t)\, dt \quad (3)$$

$$K_d = \delta \times \frac{d}{dt} err(t) \quad (4)$$

Summing the three equations, the PID controller is constructed:

$$G = K_p + K_I + K_d \quad (5)$$

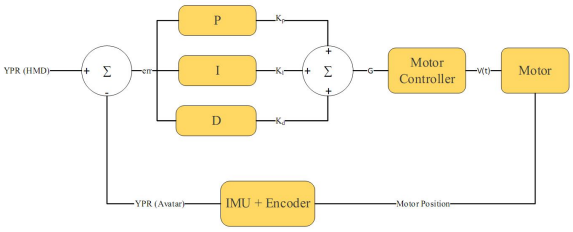


Fig. 6. Complete control diagram for the PID controller as it is integrated into the Motor Controller task.

This controller is based off the work of Wescott [7], such that the controller can be trivially tuned for each motor by modifying variables δ, ρ and I.

Relating PID gain, G, to the task's native counter, the threshold variable that is compared to by the counter (reset on match) is set by:

$$Counter\ Threshold = round\left(\frac{REF}{G}\right) \quad (6)$$

The REF variable inverts the calculated PID gain to control the 3-phase periodicity; this variable also behaves to set the sensitivity of the Gain, which must be tuned accordingly. As it would seem, the selection of the REF value is mostly arbitrary with a few considerations– some extreme cases are presented to highlight the significance of this variable:

- If REF is set to a value of 1, the motor can only be set with one of two possible counter threshold values: 0 or 1. In this case, the motor is only allowed two speeds.
- IF REF is set to too large of a value, the motor speed becomes less sensitive to the PID gain, G.

One REF is divided by the gain, the threshold value is rounded because the counter must match this value to reset, and consistent of positive integer values.

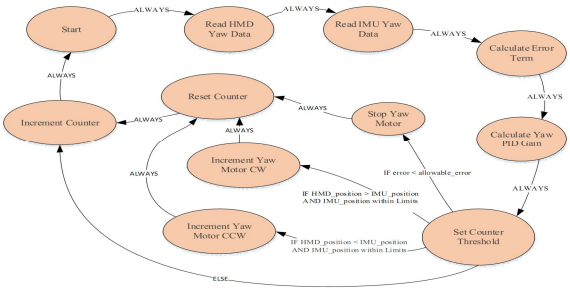


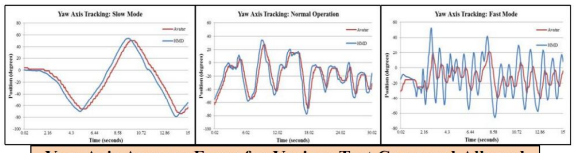
Fig. 7. State diagram for the Motor Controller task. Note that this corresponds to a single axis (Yaw). The actual algorithm accounts for all three axes.

VI. RESULTS AND CONCLUSIONS

The design is tested by measuring the current and goal coordinate positions at the same time to determine average error; the device is tested under three conditions, since the error directly relates to how the HMD is being used; error will inevitably be higher when the HMD is rotated more quickly because it will be harder for the avatar to track its movements.

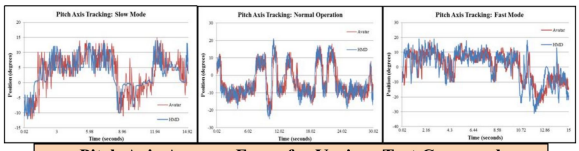
1. Slow Mode – the HMD is slowly rotated about the axis of interest.
2. Normal Operation – the HMD is placed on a user's head and is asked to look around. This represents the device under normal operating conditions.
3. Fast Mode – the HMD is shaken vigorously.

Each axis is measured independently, since each axis is controlled independently.



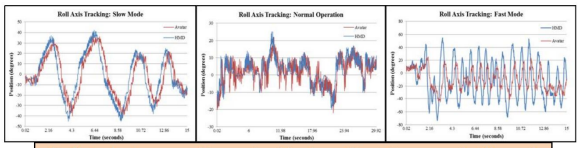
Yaw Axis Average Error for Various Test Cases and Allowed Error		
Mode	**Average Error (degrees)**	**Allowed Error (degrees)**
Slow	5.21	±4
Normal	6.24	±4
Fast	15.78	±4

Fig. 8. Typical results for Yaw-axis tracking for the slow mode (left), normal operation (center) and fast mode (right) tests.



Pitch Axis Average Error for Various Test Cases and Allowed Error		
Mode	**Average Error (degrees)**	**Allowed Error (degrees)**
Slow	2.43	±1
Normal	3.56	±1
Fast	5.44	±1

Fig. 9. Typical results for Pitch-axis tracking for the slow mode (left), normal operation (center) and fast mode (right) tests.



Roll Axis Average Error for Various Test Cases and Allowed Error		
Mode	**Average Error (degrees)**	**Allowed Error (degrees)**
Slow	8.56	±1
Normal	3.56	±1
Fast	12.32	±1

Fig. 10. Typical results for Roll-axis tracking for the slow mode (left), normal operation (center) and fast mode (right) tests.

An interesting observation from the YPR trends is that the Yaw data is much less noisy than the data for the other axes. This is caused by the inclusion of the

encoder (8-bit resolution) that has less precision than that of the IMU, which makes measurement of the device's position less sensitive to small movements. Additionally, the encoder is much less susceptible to environmental noise, whereas the other axes rely on the IMU's accelerometer and gyroscope. However, this noise is more or less undetectable by qualitative measure, since error allowance is added to each of the axes controllers which effectively behave as low pass filters.

Error allowance values are set until the best possible performance is achieved for each axis. We acknowledge that the Yaw and Roll motor cannot provide the necessary torque at the required velocities, and so compromises were made to the performance. Using error allowance, the average error (shown in Figures 8–10) is computed with the following equations:

$$e(t) = |position(t)_{hmd} - position(t)_{imu}| - EA \quad (7)$$

$$F(t) = \begin{cases} e(t), & e(t) > 0 \\ 0, & e(t) \leq 0 \end{cases} \quad (8)$$

$$Avg_{err} = \frac{\sum_{t=0}^{T} F(t)}{T} \quad (9)$$

In (7), EA corresponds to the allowed error value, and T in (9) refers to the total number of samples collected. For each test, a total of 5000 points (per axis) were selected for 0.2ms between intervals.

From the data, the Pitch-axis proves to have the greatest potential for study; in addition to having the least average error per test while having the tightest error allowance, users who participated in the testing of the device remarked that the Pitch-axis has near-perfect response– that is, there is little to no detectable delay between head movements and the avatar's Pitch-axis tracking. The results show that when the motors are selected more carefully, the performance can be substantially improved and has the potential for achieving presence.

A strange phenomenon for the Roll axis data set is that the average error measured for the slow mode test is substantially greater than that of the device tested in normal operation. In the Roll-axis slow mode test, the HMD was rotated from side to side in a greater range of motion than that of the device tested under normal operation; hence, the calculated error is higher. The data retrieved under the devices normal operating mode (and based on user experience) show that humans do not typically rotate their head about the roll axis as much as they do about the yaw and pitch axes when looking around.

From this study, we derive that the quantitative results provide some correlation to the qualitative performance of the avatar's ability to track the HMD and that an average error of less than 3 degrees per axis is a desirable goal for achieving presence.

VII. FINAL REMARKS

In summary, this paper presents a control solution for visual Telepresence, whereby an electro-mechanical gimbal is designed and implemented to gain a further understanding for some quantitative requirements needed to achieve presence for Telepresence applications. We acknowledge that an improved method relating latency between the HMD and avatar can greatly improve future designs and is the direction of our future research. Additionally, our research can greatly benefit from a second iteration prototype that builds upon the basis of our discoveries presented in this paper.

ACKNOWLEDGEMENT

The authors thank John R. Ridgely for the C++ code libraries that supplement the implementation of the design highlighted in this paper, and Bryan Mealy for his suggestions and advice.

REFERENCES

[1] A. Liu, G. Tharp, L. French, S. Lai, and L. Stark, "Some of What One Needs to Know About Using Head-Mounted Displays to Improve Teleoperator Performance," IEEE Transactions on Robots and Automation, vol. 9, pp. 638–648, October 1993.

[2] A. Kristoffersson, S. Coradeschi, and A. Loutifi, "A Review of Mobile Robotic Telepresence", Advances in Human-Computer Interaction, Vol. 1, 2013

[3] Cameron, A.A. "Visor Projected Helmet Mounted Displays Technology and Applications." Microprocessors and Microsystems 22.8 (1999): 465-75.

[4] Kratz, Sven, Jim Vaughan, Ryota Mizutani, and Don Kimber. "Evaluating Stereoscopic Video with Head Tracking for Immersive Teleoperation of Mobile Telepresence Robots." Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts - HRI'15 Extended Abstracts (2015).

[5] Martins, Henrique, Ian Oakley, and Rodrigo Ventura. "Design and Evaluation of a Head-mounted Display for Immersive 3D Teleoperation of Field Robots." Robotica (2014): 1-20.

[6] Grossman, G.e., R.j. Leigh, L.a. Abel, D.j. Lanska, and S.e. Thurston. "Frequency and Velocity of Rotational Head Perturbations during Locomotion." Exp Brain Res Experimental Brain Research 70.3 (1988).

[7] Wescott, Tim. "PID Without a PhD." Embedded Systems Programming, October, 2000.

★★★