ABSTRACT

Today, assistive robots are being introduced into human environments at an increasing rate. Human environments are highly cluttered and dynamic, making it difficult to foresee all necessary capabilities and pre-program all desirable future skills of the robot. One approach to increase robot performance is semi-autonomous operation, allowing users to intervene and guide the robot through difficult tasks. To this end, robots need intuitive Human-Machine Interfaces (HMIs) that support fine motion control without overwhelming the operator. In this study we evaluate the performance of several interfaces that balance autonomy and teleoperation of a mobile manipulator for accomplishing several household tasks.

Our proposed HMI framework includes teleoperation devices such as a tablet, as well as physical interfaces in the form of piezoresistive pressure sensor arrays. Mobile manipulation experiments were performed with a sensorized KUKA youBot, an omnidirectional platform with a 5 degrees of freedom (DOF) arm. The pick and place tasks involved navigation and manipulation of objects in household environments. Performance metrics included time for task completion and position accuracy.

Keywords: Assistive robotics, human-robot interface, mobile manipulation

1. INTRODUCTION

Robotics is currently in its 3rd phase of development. The late 1970s saw the rise of unintelligent, stationary industrial robots. During the 1990s, progress was made in mobility, intelligence, and cooperation to develop “personal” robots in areas such as research, education, and entertainment [1]. Today is the generation of “ubiquitous” robots that will support humans in everyday life. Unlike industrial robots, future co-robots will share their working space with humans and work in household environments. These assistive devices do not necessarily have to be fully autonomous. It has been shown that a human-robot pair can outperform either a human or robot working alone [2].

Therefore, intuitive human-robot interfaces will play a crucial role. Non-expert users must be able to interact and communicate with the robot, which may involve speech, gesture, haptic displays, etc. [3]. This interaction can also be physical. For example, a robot can learn a new task from a novice operator via kinesthetic teaching, where the user manually pushes and pulls the robot’s manipulator to complete a task [4].

Since physical contact may occur, human-robot interfaces will also play a key role in safety. Safety can be divided into a physical and behavior aspect [5]. When physical contact occurs, the control architecture can limit joint torques and velocities, and take advantage of passive compliance. There are adaptive schemes that can compensate for unknown parameters and disturbances while guaranteeing robust and stable control [6]. In our recent work, we validated a novel neuroadaptive framework that improved physical HRI [7, 8]. These methods use single F/T (force/torque) sensors to estimate the force applied by a human on the robot end-effector. A promising technology is robot skin, which equips the robot with touch and allows precise localization of multiple contact forces [9]. While whole-body, human-like skin has yet to be realized, pressure sensitive piezoresistive “taxel” arrays encapsulated in flexible silicone substrates already exist [10].
For behavior safety, the robot takes multi-modal cues such as facial expression and body pose to determine human intent. This allows the robot to align its goals with the operator, plan ahead, and adjust its behavior. For example, if a collision is anticipated or a child interacts with the robot, the control scheme could increase its compliance. Feedback from the operator could also allow the robot to behave more human-like, making the interface simpler to understand and more intuitive for the operator [11].

The intuitive and safe human-robot interfaces must be facilitated by multi-modal sensor data. The robot can perceive and understand its environment through camera images that are then processed by vision algorithms. For example, the OpenCV library implements several real-time computer vision functionalities, including people detection and face tracking [12]. Additional information can be gained from depth: RGB-D cameras such as the Microsoft Kinect or Asus Xtion have become the de-facto standard in robotics [13]. Laser scanners usually provide 2D depth information and are commonly used for mobile navigation. Sonar or ultrasonic sensors are less accurate but relatively inexpensive range-finders, and have been deployed on co-robots such as Baxter to detect human in the workspace and surrounding areas [14]. Another alternative is thermal sensors, which can detect radiated heat or far-infrared rays from nearby humans. In addition to low cost, processing infrared data is fast compared to image processing for human detection.

In this paper, a standard robot platform was modified to investigate the performance of several HMIs in a household environment, continuing the work described in [15]. The KUKA youBot, which is commonly used in academia, was transitioned from the “personal” to “ubiquitous” generation of robots by performing several hardware upgrades. The platform was sensorized by installing an RGB-D camera for object and human detection, a laser scanner for fully and semi-autonomous navigation as well as collision detection, robot skin patches consisting of piezoresistive pressure sensors for physical interaction, and thermal sensors to detect a human’s presence. A National Instruments roboRIO was added to provide additional input/output ports and software was developed to process incoming data. The network infrastructure was improved to allow remote control via a tablet interface and assisted joystick teleoperation. As such, the contribution of this paper is how to sensorize a standard robot platform (both hardware and software) for human-robot interfaces and their expected performance in a household environment.

In the following section, we describe the robot platform and how it was upgraded to meet the HMI requirements. In Section 3, we discuss the sensors used for the multi-modal interfaces and the software architecture in Section 4. The experimental setup is presented in Section 5 and the results in Section 6. Finally, Section 7 concludes the paper and postulates future work.

2. HARDWARE PLATFORM

In this section we describe the robot platform and the necessary hardware sensors and interfaces needed for teleoperation.

The automation company KUKA specifically developed the youBot as a research and application platform for mobile robotics [16]. It has a five degrees of freedoms (DOF) manipulator with a height of 655mm and a workspace of 0.513m³. The arm can lift a payload up to 0.5kg and has a position repeatability of 0.1mm. Grasping can be performed with a two-finger gripper that is being powered by 2 independent stepper motors and has a range of 70 mm.

The arm is mounted on an omnidirectional base of dimension 580mm (length) by 376mm (width). The four Mecanum wheels allow movement in any direction (x,y) at any orientation (θ). It can reach velocities up to 0.8m/s and carry a payload of 20kg. The mobile base houses a rechargeable battery which allows a runtime of approximately 90 minutes. An onboard mini PC (Intel Atom Dual Core CPU, 2GB RAM) runs the Linux-based operating system Ubuntu. The platform components rely on EtherCAT communication with a 1ms real-time cycle.

A remote workstation can be connected to the robot via Ethernet cable. This is suitable for running computationally demanding algorithms and heavy graphics processing that would overload the onboard PC. To create an untethered setup, our first upgrade consisted of mounting a wireless router (ASUS RT-N66U Dual-Band Wireless-N900 Gigabit Router) on top of the base. This router is connected to the onboard PC via Ethernet connection, and acts as a bridge to transmit data to a remote workstation (Fig. 1). This allows an operator to run and troubleshoot processes remotely, view and collect live data, and visualize the robot state in a graphics program. In addition, the youBot WiFi network can be used to connect interface devices such as tablets, phones, or laptops. The additional sensors for the HMI require Input/Output (I/O) ports for signal acquisition, conditioning, and networking. The National Instrument roboRIO is an advanced robotics controller that features several built-in ports for “I²C, SPI,
RS232, USB, Ethernet, PWM, and relays” [17]. Released in 2015, it is part of the reconfigurable I/O (RIO) family and use the Xilinx Zynq chipset. It has a reconfigurable FPGA and is powered by a 667 MHz dual-core Real-Time processor.

Figure 1. Communication diagram for sensor data acquisition and wireless setup between youBot and remote workstation.

3. SENSORS FOR HMI

Several sensors were mounted and integrated with the youBot platform to facilitate human-machine interaction (HMI), including a laser scanner, robot skin, and thermal sensors.

3.1 Laser scanner

A laser scanning rangefinder was installed at the front side of the youBot base to detect obstacles. The sensor data can be used for autonomous navigation and assistive teleoperation. A Hokuyo URG-04LX-UG-01 model was used, which has a 240° field of view and a measurement distance of 4m [18].

3.2 Robot Skin

For physical interaction, the youBot end-effector link was outfitted with four skin patches consisting of pressure sensors embedded in P10 RTV silicone rubber as shown in Fig. 2. The Tekscan Flexiforce thin-film sensors are ideal for measuring force between two surfaces. They are cost effective and durable with good sensor characteristics: linearity error within 3%, hysteresis less than 4.5%, drift less than 5%, and low temperature sensitivity (0.36% per °C). The maximum response time is 5 μsec and they handle up to 100lbs (445N). The active sensing area is 9.53mm in diameter.

The particular sensors used are piezoresistive in nature; as force increases the resistance decreases from infinity to approximately 300kΩ. A voltage divider circuit with an emitter buffer was designed to measure human applied forces up to 50lbs (222N). The circuit was implemented on a custom data acquisition board (or MicroBoard), which conditions the pressure data from several sensors (or taxels). The results are read by the roboRIO using its analog input ports.

Figure 2. Robot skin patch placement on the KUKA youBot manipulator (left image modified from [16]).
3.3 Thermal sensors

Human presence can be detected by using cameras and computer vision algorithms. A simpler and more cost effective approach involves thermal sensors, which detect radiated heat or far-infrared rays of an object. Hence, they are not affected by different lighting conditions like conventional cameras and simple thresholding can be used for detection.

The Omron’s MEMS thermal sensor (D6T-44L) consists of a MEMS thermopile sensor chip covered with a silicon lens [18]. The chip measures an electromotive force and an embedded circuit converts the analog signals to digital temperature values. In contrast to conventional pyroelectric sensors, Omron’s sensor does not measure a change in signal and continually detects the far-infrared ray of an object. Hence, the sensor is able to catch a signal of a moving as well as stationary person.

A custom box with three thermal sensors was mounted on the end-effector just below the two finger-gripper. The sensor itself outputs a 4 by 4 pixel array (Fig. 3a) and has a viewing angle of 45° [18]. With three sensors, 4 by 12 pixels cover approximately 120° with some overlapping. The hardware including thermal sensors and wiring were encased in a 3D printed enclosure as shown in Fig. 3b and 3c. The measured values are transmitted through an I²C bus to the roboRIO.

![Figure 3. (a) The detection area of the Omron D6T-44L [18]. Three sensors were mounted in a 3D printed enclosure (b), including circuitry for robot skin and roboRIO interfacing shown from the top in (c).](image)

4. SOFTWARE ARCHITECTURE

The previous sections describe the youBot hardware upgrades. A software architecture was developed to process the sensor data and enable robot control with the various interfaces, consisting of ROS, an Android tablet application, and LabVIEW.

4.1 Robot Operation System

The KUKA youBot runs the linux-based operating system Ubuntu 12.04. The platform is controlled by the open-source Robot Operating System (ROS), a software framework originally developed by Stanford Artificial Intelligence Laboratory in 2007 [19]. ROS has become a de-facto standard in robotics and there is a large collection of software packages developed by the community. Since ROS is both programming language and hardware agnostic, these packages can easily be re-used on different platforms. Programs are executed as independent ROS nodes and data is transmitted via ROS messages. This distributed architecture allows several components to run simultaneously, for example sensor data processing, navigation, and perception.

In this spirit, we developed a multi-layered, ROS-based architecture to allow robot autonomy as well as user intervention via several HMIs such as tablet apps or pressure sensors (Fig. 4). Several stand-alone ROS packages were developed, with different functionalities for navigation and manipulation as described in Table 1. The program flow is determined by a cortex node which acts as layer between the interfaces and control modules. It contains a state machine which utilize ROS topics or services to call other ROS nodes that then execute the necessary code. If anything fails, the cortex node runs contingency plans and is able to restart faulty ROS nodes. This also simplifies data flow making it easier to debug and capture data.
Table 1. Overview of the developed ROS packages.

<table>
<thead>
<tr>
<th>ROS package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>youbot_cortex</td>
<td>Contains a state machine node which acts as a layer between the data received from the tablet app and the robot controllers. It sends service requests to enable/disable different control modes, joints, and sensors.</td>
</tr>
<tr>
<td>cortex_msgs</td>
<td>Defines custom ROS messages and services.</td>
</tr>
<tr>
<td>serial_io</td>
<td>Reads data from roboRIO via a serial communication channel and publishes it into ROS topics.</td>
</tr>
<tr>
<td>youbot_force_sensors</td>
<td>Subscribes to force data and computes the command velocities for the base movement. KDL (Kinematics and Dynamics library) is used for 3D frame and vector transformations.</td>
</tr>
<tr>
<td>youbot_ir_sensors</td>
<td>Processes thermal sensor data (filtering and thresholding) to control the gripper.</td>
</tr>
<tr>
<td>youbot_description</td>
<td>Contains the youBot robot model.</td>
</tr>
<tr>
<td>youbot_navigation</td>
<td>Utilities for mapping, localization, and path planning.</td>
</tr>
<tr>
<td>youbot_arm_controllers</td>
<td>Sends position commands to the youBot motors using different control algorithms (individual and combined joint control).</td>
</tr>
<tr>
<td>youbot_cartesian</td>
<td>Cartesian arm controller using the motion planning software MoveIt! [20].</td>
</tr>
</tbody>
</table>

Figure 4. Software architecture showing the flow of data. The sensors are being read by the roboRIO, which then sends the data to ROS. The arm and gripper commands are then executed by the low-level hardware controllers in the youBot.
4.2 Tablet Interface

A youBot tablet application was developed using ROSJava [21] to allow users to control the platform with an easy-to-use graphical user interface (GUI). Figure 5. Tablet interface for controlling the youBot. The app is split up into 3 different classes: ViewController, NodePublisher and the VirtualDivet. The ViewController connects the GUI interface (buttons, toggle buttons, switches, text) to backend code. Using ROS messages, the NodePublisher communicates user intent to the cortex program which then initiates the robot movement. The VirtualDivet class implements the touch-screen joystick in the lower right hand corner. It allows the user to move around a divet inside a circular area, where the off-center distance produces velocity commands between 0 and 100%.

Figure 5 shows the layout on a Nexus 10 Android tablet. The top left corner displays the tablet orientation, i.e. the pitch and roll from the gyroscope and the yaw relative to earth from the magnetic compass. The active control mode can either be individual or combined joint control. Joint control mode allows the selection of individual joints which then can be moved by tilting the tablet. The joint velocity is proportional to the pitch of the tablet. In combined joint control, several joints are moved together to move the end-effector in an arc. Joint limits prevent the user from hitting the floor or robot. The Cartesian position of the end-effector can be controlled with buttons in the upper right corner. Vertical Cartesian control moves the end-effector along the y-axis of the base frame. Finally, there are switches for activating different interfaces such as the thermal and pressure sensors. The gripper switch opens or closes the two-finger pincher. The base joystick toggle enables the virtual joystick, which moves omnidirectional base of the robot platform.

![Tablet Interface for controlling the youBot.](image)

4.3 RoboRIO Software

The roboRIO was programmed using the National Instruments’ LabVIEW. The code is executed in a real-time loop, where pressure sensors are read every 1ms using the analog input ports. The infrared sensors generate data every 30ms and use the I²C bus interface. After being captured, the data is relayed via USB to the youBot using the RS232 communication protocol. Collecting sensor information and communicating with ROS are parallel tasks, made possible by the multicore processors on the roboRIO. This way the ROS nodes can access the sensor data at approximately the same rate: 1000Hz for pressure sensor and 33Hz for thermal data.

5. DESCRIPTION OF EXPERIMENTS

Experiments were conducted to test the functionality of the implemented hardware and performance of the proposed HMI schemes. The tasks involved object pick-and-place using the tablet interface and pressure sensors, as well as autonomous navigation. The primary objective was to measure the difference in completion time and accuracy of path trajectory for expert and novice users. Novice or non-expert users were defined as those who had no prior experience with the tablet application, the robot, and its functionalities. Both type of users were given a brief overview of the system and the tasks to be completed. Measurements included completion time and trajectory error, which was determined by localizing the youBot with the help of odometry and laser scanner data.
5.1 Pick and Place

In the pick and place task, a user was asked to move several objects between waypoints with the youBot. To create a rigorous method of testing, the waypoints were marked on the floor as shown in Fig. 6. The path involved straight and diagonal movements of various lengths: 3.05m from point $A$ to point $B$, 2.76m from $B$ to $C$, 3.68m from $C$ to $D$, 2.80m from $D$ to $E$, and 3.07m from point $E$ to point $F$.

Different interfaces were tested to compare the accuracy and completion time of physical guidance and tablet teleoperation. Both expert and novice users were asked to pick up 5 objects in 6 trials. This was repeated for two types of interfaces: teleoperation via tablet (tablet mode) and direct physical interaction (mannequin mode) via the skin patches.

In **tablet mode**, the user was required to accomplish the following tasks:

1. Pick up object at current location
2. Transport object to next waypoint using the virtual joystick
3. Place object 2-3 inches within waypoint marker
4. Repeat until final waypoint has been reached (point $F$)
5. Place final item into a bin

The base was operated using the virtual joystick and the robot arm was controlled in joint or Cartesian mode depending on user preference (see [15] for further details). The gripper was opened and closed by pressing the corresponding button inside the tablet interface.

In **mannequin mode** the tasks were as follows:

1. Pick up object at current location
2. Guide robot to next waypoint using the robot skin
3. Place object 2-3 inches within waypoint marker
4. Repeat until final waypoint has been reached (point $F$)
5. Place final item into the bin

The user guided the youBot along the outlined path by pushing on the robot skin mounted on the manipulator. Once a waypoint was reached, the user moved the arm into position and operated the gripper with a simple hand-wave in front of the thermal sensors.

5.2 Autonomous Navigation

In addition to pick and place experiments, in which the user actively guided the robot, experiments were conducted to evaluate the youBot’s autonomous navigation and obstacle avoidance capabilities. To navigate, the youBot received goal positions via point-and-click mouse commands in the ROS Visualization (RViz) interface shown in Fig. 7. The software
provides a graphical user interface (GUI) showing a 3D model of the robot located in a 2D map outline in light grey color. Potential obstacles such as walls and furniture are marked in black and the real-time laser scan data is represented by white pixels. The youBot navigation program utilizes robot odometry and laser scan data to localize the robot. After a user provides a goal location in RViz, the global planner generates a collision free path using algorithms such as A* search. If an unexpected obstacle appears during motion, for example a human stepping in front of the robot, the local planner takes over and creates a modified path around the obstacle.

Figure 7. RViz interface showing a top-view of the youBot inside a mapped area (light grey region). The black pixels represent obstacles and white pixels the real-time laser scanner data.

6. RESULTS

6.1 Pick and place

The completion time for the pick and place experiments are listed in Table 2. Six trials were conducted with two expert and two novice users. The mean completion time for an experienced user was approximately 2 minutes and 45 seconds. As expected, the non-experts performed slower and needed more than double the time to complete an identical task. However, the time difference between the two interfaces for each user was minimal as shown in Fig. 8. This seems to indicate that both the tablet and mannequin mode provided the same level of control. However, this was not the case as shown further down by the position data. It should also be noted that the physical interface required almost no training, while it took a few minutes for the novice users to get used to the tablet interface. In addition, when the youBot was facing the operator, there was some confusion since the left and right tablet commands would be flipped from the operator’s point of view. The disparity in time between the users may be attributed to different individual skill levels.
Table 2. Task completion time (in seconds) for expert and novice users for the tablet and mannequin mode. The mean and standard deviation (STD) is computed for 6 trials.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Expert user 1</th>
<th>Expert user 2</th>
<th>Novice user 1</th>
<th>Novice user 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tablet</td>
<td>175</td>
<td>188</td>
<td>395</td>
<td>368</td>
</tr>
<tr>
<td></td>
<td>158</td>
<td>174</td>
<td>388</td>
<td>343</td>
</tr>
<tr>
<td></td>
<td>162</td>
<td>159</td>
<td>382</td>
<td>340</td>
</tr>
<tr>
<td></td>
<td>153</td>
<td>141</td>
<td>370</td>
<td>348</td>
</tr>
<tr>
<td></td>
<td>148</td>
<td>139</td>
<td>372</td>
<td>330</td>
</tr>
<tr>
<td></td>
<td>178</td>
<td>181</td>
<td>409</td>
<td>398</td>
</tr>
<tr>
<td>Mean</td>
<td>162</td>
<td>164</td>
<td>386</td>
<td>355</td>
</tr>
<tr>
<td>STD</td>
<td>12.0</td>
<td>20.7</td>
<td>14.7</td>
<td>24.7</td>
</tr>
<tr>
<td>Mannequin</td>
<td>169</td>
<td>176</td>
<td>401</td>
<td>382</td>
</tr>
<tr>
<td></td>
<td>171</td>
<td>159</td>
<td>402</td>
<td>389</td>
</tr>
<tr>
<td></td>
<td>173</td>
<td>170</td>
<td>393</td>
<td>381</td>
</tr>
<tr>
<td></td>
<td>166</td>
<td>162</td>
<td>399</td>
<td>395</td>
</tr>
<tr>
<td></td>
<td>159</td>
<td>160</td>
<td>383</td>
<td>346</td>
</tr>
<tr>
<td></td>
<td>171</td>
<td>169</td>
<td>401</td>
<td>389</td>
</tr>
<tr>
<td>Mean</td>
<td>168</td>
<td>166</td>
<td>397</td>
<td>380</td>
</tr>
<tr>
<td>STD</td>
<td>5.1</td>
<td>6.7</td>
<td>7.4</td>
<td>17.6</td>
</tr>
</tbody>
</table>

Figure 8. Error plot showing mean completion times (in seconds) for two expert and two novice users.

In addition to completion times, the youBot’s location in the map frame was recorded during the experiments. The position was estimated from internal odometry and laser scanner data. The two expert and two novice users were asked to follow the path marked on the floor as closely as possible. Fig. 9 and 10 depict the trajectories obtained via tablet and mannequin control, respectively.

Due to slippage of the Mecanum wheels on the hard floor, the odometry position estimate drifted over time. Therefore, the position measurements were not accurate enough to quantitatively compare the trajectory errors. In future work, the youBot position estimate could be recalibrated between trials or a motion capture system could localize the robot more accurately.
accurately. However, by general inspection of the graphs in Fig. 9 and 10, one can infer that the mannequin mode produced smoother and more accurate trajectories compared to the tablet mode. The mannequin trajectories in Fig. 10 show less spread and there are fewer path deviations. Hence, similar completion times does not equate to identical performance. For both interfaces there was a clear difference between expert and non-expert users. From a qualitative assessment, the expert users produced smoother paths and performed fewer corrections.

6.2 Autonomous Navigation

Two different setups were utilized to evaluate autonomous navigation using the RViz interface. First, the youBot had to avoid a circular obstacle to reach the same waypoint from three different initial positions. Moving right to left in Fig. 11(a), the youBot successfully avoided any collisions and chose the shortest trajectory around the obstacle.

In the second setup, the robot moved along two successive waypoints requiring a 90 degree turn. The results depicted in Fig. 11(b) show that the position accuracy decreased over time, which again was caused by slippage of the wheels. Qualitatively, the accuracy is similar to mannequin mode operation and better than tablet control.

Figure 9. The youBot map coordinates in meters during tablet mode operation for 6 trials with (a), (c) expert and (b), (d) non-expert users.
Figure 10. The youBot map coordinates in meters during mannequin mode operation for 6 trials with (a), (c) expert and (b), (d) non-expert users.

Figure 11. The youBot map coordinates in meters during autonomous navigation from right to left. The setup in (a) required avoiding a circular obstacle and (b) involved two waypoints resulting in a 90 degree turn.
7. CONCLUSION

In this paper the KUKA youBot mobile manipulator was sensorized for user operation via a tablet, physical interaction, and simple hand gestures. Sensor upgrades included a laser scanner, pressure sensitive robot skin, and thermal sensors. System integration was accomplished through the addition of a roboRIO controller and the software architecture was based on the Robot Operating System (ROS). After the HMI and sensor upgrades, the robot can be guided by users via teleoperation with a tablet, direct physical guidance, as well as through a point-and-click GUI. We demonstrated basic capabilities and acquired data from a few users, which will pave the way for experimental studies including a larger number of subjects in near future. Several pick and place tasks were completed by two users familiar with the system (experts), and two users not familiar with the system (non-experts). The results indicate that:

1. Physical interaction (or mannequin) guidance results in the most accurate robot trajectories for both expert and non-expert users. The interface is intuitive and might require less training time.
2. Tablet and mannequin control result in similar task completion times. Expert users can complete pick and place manipulation and mobility tasks significantly faster than non-expert users.
3. The autonomous point-and-click interface resulted in shortest path for traversing free and obstacle-filled environments, and had similar accuracy to mannequin guidance. There is minimal effort for the user, however the robot requires a map of the environment.

ACKNOWLEDGEMENTS

This research was supported by National Science Foundation Grants NRI-1208623 and BIC-1261005720, and Qinetiq-North America Corporation.

REFERENCES