The Design and Evaluation of a Cooperative Handheld Robot

Austin Gregg-Smith and Walterio W. Mayol-Cuevas

Abstract—This paper concerns itself with a relatively un-explored type of personal robot that operates in the tool space. Handheld robots aim to cooperate with the user to solve tasks and improve what tools can offer enhanced by actuation, sensing, and importantly, task knowledge. To this end, we devised a new lightweight robotic platform that has 4 DoF and uses a cable driven continuum structure. Feedback from the robot to the user is provided in an intuitive, implicit manner by the robot end effector pointing towards the goal, avoiding pointing, and/or refusing to perform an action when it conflicts with the task specification. We evaluate two generic tasks involving aiming in space and picking/placing objects with a number of volunteers. Repeated measures ANOVA is used to analyse results to show in which conditions an increased level of automation in the handheld robot improves task performance or user perception of task load. The robot is offered as an open robotics platform[1] and the results indicate directions to improve on feedback and interaction mechanisms.

I. INTRODUCTION AND RELATED WORK

Historically, handheld tools used by humans have been inanimate, unintelligent instruments. Traditional tools are incapable of knowing the context they operate in, are fully directed by the user, and critically, lack any task knowledge. In this paper, we discuss work about the development of a relatively unexplored type of personal robotic device: cooperative handheld robots. In this case we assume these robots have a simulated level of cognition via pre-specified task knowledge. While a definition of cognitive robots is still elusive, here we focus on devices that are aware of their environment, know how to perform the task, and are aware of task’s progression. Figure 1 shows the basic components of such a system for a generic picking and placing task where the handheld robot has task knowledge, receives information from the environment and some directions from the user (e.g. tactical movements or trigger pressing) to drive the tool’s actions.

Handheld robots share physical proximity with their users but are neither fully independent nor part of their body. The aim is to capitalize on this proximity to aid in the sharing of cognitive and actuation abilities. By exploiting the intuitiveness of traditional handheld tools and adding embedded intelligence, a new range of capabilities opens up. However, endowing a tool with higher level competencies may also introduce challenges in the interaction between the user and robot depending on the level of autonomy. Apart from describing our design choices, this paper looks at key aspects of user-robot interaction for two generic handheld tasks via rigorous experimentation in a controlled measuring environment.

Department of Computer Science, University of Bristol, UK
ag7751@bris.ac.uk, wmayol@cs.bris.ac.uk

In Personal Robotics, where the person is in the loop [2], the embodiment of the robot is crucial. It determines both the extent and expectation of the interaction, as well as the kind of tasks that can be achieved. To date, personal robotics research has mainly been in two very different areas in terms of proximity to the user:

- External: an independent device with sensors and actuators directed toward the environment such as a remote surveying robot relaying and receiving information from users perhaps only occasionally.
- Wearable: a robot that is very close to the user and that helps to achieve tasks by receiving direct, and perhaps constant instructions from the user, such as a wearable exoskeleton or a robotic wheelchair.

There are many types of person-oriented robots that exist external to the person. These devices have to be carefully designed, not simply for their function and safety as with any other personal robot, but if intended for engaging with an individual, as in robotic mascots and robotic tour guides, their external appearance and shape has to be considered so that interaction is not impaired [3]. The other main type of person-oriented robot embodiment is wearable, where the device is intended to be always present, sometimes as an extension of the user’s body and may serve as a cognitive assistant [4], [5] or to provide mechanical advantage as in the case of wearable exoskeletons [6] and supernumerary robots [7]. Wearability brings a new series of critical design considerations such as social acceptability and safe load bearing [8]. Perhaps of more concern is that wearables are
by definition tailored to specific individuals. There is a bridging space between these two main scenarios which research into autonomous intelligent robots has largely neglected: this is the "tool space". The handheld tool space has been important through evolutionary history as a way to modify, adapt and shape the environment. Traditionally, it has enabled humans to increase their strength or accuracy beyond what limbs are capable of. Furthermore, tool use has been observed in several other species other than primates, for example, in birds [9] and rodents [10]. However, in humans tool use capacity is uniquely developed [11].

There are various related works, principally in the field of medical robotics, that have looked at handheld tools with sensors and actuators. Dario et al [12], [13] developed a 1 DoF cable driven mechatronic arthroscope designed to minimise contact with delicate tissue. The system automatically avoids collision with areas of tissue that have been identified in advance, using an optical tracking system. Brisson et al [14] developed a 1 DoF freehand medical sculptor to allow a surgeon to accurately cut bone to a shape the surgeon predefined using CAD software. Becker et al [15] developed a 3 DoF robotic scalpel, Micron, which used a handheld micro-manipulator to aid the cannulation of retinal blood vessels by actively stabilising the tool tip at 2 kHz.

Other examples include assisting eating with a stabilized spoon that aims to compensate for Parkinson’s disease tremors[16]. One of the few examples in the literature outside of the medical domain is the welding gun by Echtler et al [17] which is a handheld welder that has the function of activating the welding current if positioned in the correct spatial spot in a car assembly task.

From the above we note: i) the concentration of work in the medical domain, ii) the limited task knowledge in the tool, iii) the low number of degrees of freedom that makes most tools less generalizable and only able to perform single tasks, and iv) that in most cases the communication between the tool and the user is not explored fully.

We argue that cooperative handheld robots can have a much larger range of applications outside medical devices. For example, work tools that help in maintenance by knowing the where to act and the specific torque to remove or drill through objects. In agriculture a non-expert user would be able to roughly guide an intelligent weed sprayer or harvester and in general, for tasks that benefit from sharing actuation and cognitive load and where auditing and task skill is required.

II. SYSTEM OVERVIEW

There are many areas of interest to study in handheld robotics from the decision making process to the design of the ergonomics to the feedback mechanisms. In this paper, we mainly evaluate the interaction between the user and a handheld robot with a high number of degrees of freedom.

While it would be relatively easy to design both robot and task so that performance vastly surpasses any human capability (e.g. operating at micro-scales or ultra-high accelerations), we are instead interested in using a tool that operates at about the same scales and speeds as conventional tools used in standard field tasks. This allows us to concentrate on the effect of various degrees of autonomy and cooperation without being distracted by cases where the operational superiority of the tool swamps human capability.

Our design is new rather than based on existing tools as there were no off-the-shelf options that would have allowed us to evaluate the interaction between a lightweight, high DoF, handheld tool and different levels of automation. We depart from previous actuated handheld systems principally by increasing the scale and by extending the number of kinematic degrees of freedom, that allow us to explore tasks where spatial orientation is important.

A. Hardware Design and Optimisation

A cable driven, continuum robot design was chosen due to its speed, low mass and safety around inexperienced users. The mass of the arm section is low (161g), which allows for fast reactions. This is important because if the arm was not able to move faster than the operator, it would limit the ability to correct for mistakes in positioning. The total mass and centre of gravity of the arm is also important to consider so that it is not uncomfortable to hold while performing tasks. A cable driven design allows the driving motors to be placed near the user instead of at the joints, reducing the moments without requiring counterweights.

A four degree of freedom (4-DoF) tentacle-like robotic arm was designed and built around a grass strimmer base (Mod GL4525) with the robot arm attached to its end shaft. The existing strimmer trigger is used by the user in some modes to activate tool function.

The arm uses a carbon fibre central backbone similar to [18][19][20][21][22][23][24][25] but with modifications to the cable driving mechanism (figure 2). The arms mentioned above all have a dedicated motor to adjust the tension of each of the cables. This requires at least three motors for each 2-DoF section. The work in [26] improved upon this design by using one motor to control a pair of opposing wires reducing the motor count to one per DoF. Our arm uses eight tensioning wires controlled by four digital servo motors.
The carbon fibre rod is bent by changing the tension on the control wires. The naive approach of using a motor to control a pair of wires has the flaw that the length of wire pulled in by one side of the pulley is the same length that is let out on the other side.

Figure 3 shows the geometry of a flexible backbone cable driven arm. The central carbon fibre core (black) has five struts (blue) that act as wire guides for the purple and cyan control wires. On the left the arm is in the default straight state and on the right the pulley has been turned by angle theta to cause the cyan wire to bend the backbone by pulling on the blue wire guides.

The wires forms a circular segment as the cyan wire is wrapped around the pulley and exactly the same length of purple wire is let out on the opposite side maintaining wire tension. As the blue struts are only able to approximate the arc profile as the cyan wire is pulled taut, the purple wire will have extra slack wire that causes it to fall out of the pulley groove and reduce rigidity of the arm.

In our approach, we minimise this problem by optimising non circular pulleys, where the slack in the wire was taken up by an increased radius at the appropriate point in the pulley. The shape of the pulley can be described as the vector \( \bar{r} \) which contains a list of radii. Gradient descent was used to find the optimal lengths for each “spoke” radius.

To make the problem easier to solve, the wire was broken up into sections. Each of these sections has a length dependent on the angle of the pulley as shown in figure 3

\[ t_r(\theta) \] is the length of wire on the left side of the guides and \( t_t(\theta) \) is the length of wire on the right side of the guides. The amount of wire wrapped around the pulley is dependent on both the angle and the pulley profile \( \bar{r} \). \( p_l(\theta, \bar{r}) \) is the length of wire on the wrapped around the left side of the pulley and \( p_r(\theta, \bar{r}) \) is the length of wire wrapped around the right side of the pulley. The amount of slack wire for an arbitrary angle \( \theta \) can be expressed as:

\[
\text{wireSlack}(\theta, \bar{r}) = (t_t(0) + p_r(0, \bar{r}) - t_t(\theta) - p_t(\theta, \bar{r}))^2 + (t_r(0) + p_t(0, \bar{r}) - t_r(\theta) - p_r(\theta, \bar{r}))^2
\]

(1)

Therefore we want a pulley that minimises the slack for all pulley angles:

\[
\text{fitness} = \int_0^{\pi/2} (\text{wireSlack}(\theta, \bar{r})) d\theta
\]

(2)

This finds the difference of squares between the length at the neutral position and each possible angle. The fitness will be zero when there is no slack in the pulley design. To find the optimal profile, the partial derivative of the slackness is found when modifying the length of one spoke. That spoke’s length is changed to minimise the slackness, and then the procedure is repeated for all the remaining spokes until an optimal profile is found (figure 4).

An important part of the algorithm is ensuring that when a spoke length is changed, it doesn’t cause any concavity in the pulley. When the wire is wrapped around a concave part of the pulley it will not follow the pulley profile as it will form a straight line between the two edges. Concavity in the pulley also makes the cost function non-continuous which causes problems with the gradient descent. Figure 3 shows the range of values that \( r_t \) can be modified to without causing any other segments to be concave.

A disadvantage of the concavity constraint is that it greatly limits the speed at which the gradient descent can converge as each spoke has a maximum change per iteration. The problem is only apparent as the resolution (number of spokes in the pulley) increases. When a large number of spokes is used, the angle between them is smaller and therefore reduces the range of acceptable values for \( \bar{r} \). A way to mitigate this problem is to start the gradient descent with a low resolution pulley and progressively increase it.

Algorithm 1 Pseudo code of pulley shape optimisation

\[
\begin{align*}
\text{radii} &\leftarrow \text{initialSample} \\
\text{for} \quad \text{minRes to maxRes} &\quad \text{do} \\
&\quad \text{resample the radii to a higher resolution} \\
&\quad \text{while score is improving do} \\
&\quad \quad \text{partial derivative of fitness(radii)} \\
&\quad \quad \text{modify radii according to the gradient} \\
&\quad \quad \text{Check concavity constraints are met} \\
&\quad \text{score} &\leftarrow \text{fitness(radii)} \\
&\quad \text{end while} \\
&\quad \text{end for}
\end{align*}
\]

This optimisation process allows for an actuator weight reduction of 50% compared with alternative designs requiring independent actuators for tensioning as it halves the number of required motors. The total mass of the manipulator (not including the strimmer base) is 509g of which 306g are from the actuators and the rest for the attachment rig and end effector.

The arm has two 2-DoF sections attached end-to-end. The first pair of servos control four wires attached to the midpoint of the tentacle while the second pair of servos controlling the four wires attached to the tip. At the tip of the arm, an electromagnet is fitted to enable picking up and dropping of light metallic objects. Figure 5 shows some examples of the four main shapes in the arm’s state space. It is important to require limits on the maximum range of bending on the carbon fibre as at extreme angles it can de-laminate. In this case we constrain normal operation for each servo to ±20°.

A microcontroller interfaces servos and trigger button on the strimmer with the host computer. The ultimate aim of this
work is to have on-board sensors and less reliance on external infrastructure but in this evaluation, we opt for a motion tracking system to have an accurate spatial reference. We use an Optitrack system with positioning accuracy of \( \approx 0.2 \) mm at 100Hz and with 4ms latency. Our control software runs at around 350Hz including GUI and rendering.

### B. Robot Calibration

**Kinematic Modelling via Regression:** The modelling of a lightweight, cable-driven structure is not trivial as rigidity and remote actuation affect it. Conventional approaches based on forward kinematics for these structures are insufficient as they make assumptions about backbone rigidity and curvature for each 2-DoF as described by Tran et al[21]. The interaction between two segments is however not accounted for in this model and so in our evaluation did not provide an accurate representation of the arm’s state. Instead, we approach this problem with a data-driven strategy as the full dynamics of the continuous backbone proved too complex to model accurately analytically.

Our approach is based on a calibration phase where commands are sent to the servomotors and the resulting pose of the arm is recorded using the motion tracking equipment. In this case, the optical markers were placed on the base and tip of the arm so that pose of the tip can be calculated relative to the base. The desired motor angles were sent as a vector \( \vec{m} \) and the corresponding pose \( \vec{y} \) recorded where \( \vec{y} \) is comprised of the relative position \( \vec{p} \) and relative direction \( \vec{d} \).

\[
\vec{m} = (m_1, m_2, m_3, m_4); m_x \in \mathbb{R}[-1, 1] \tag{3}
\]

\[
\vec{y} = (\vec{p}, \vec{d}) \mid p, d \in \mathbb{R}^3 \tag{4}
\]

The entire state space of the arm is sampled at a chosen resolution and put in a data table. In this case, the table has \( 4^{11} = 14641 \) samples of \( \vec{m} \) with corresponding \( \vec{y} \) values.

As the table is discrete in nature, it cannot provide estimates for motor angles that were not sampled in the initial calibration. To alleviate this, kernel regression is used to interpolate between the calibration points to give a smooth estimate of pose for unrecorded \( \vec{m} \) inputs.

The Nadaraya-Watson kernel regression [27] is an estimator that operates on a set of independent examples \( [x_1, y_1] \ldots [x_n, y_n] \). The underlying function of these samples is approximated by:

\[
\hat{y}(\vec{x}) = \frac{\sum_{i=1}^{n} y_i K \left( \frac{x - x_i}{\alpha} \right)}{\sum_{i=1}^{n} K \left( \frac{x - x_i}{\alpha} \right)} \tag{5}
\]

where \( K \) is a kernel function with bandwidth \( \alpha \). An appropriate bandwidth was found by generating a second sample set that had no points that overlapped the calibration and minimising the estimate error. When calibrated, the estimates had an mean positional error of 3 mm.

A Gaussian kernel is used so that samples far away from the queried point will have a smaller influence on the interpolated value. This means that only values that are close to the query have a significant effect on the output of the regression. The naive kernel regression method sums the values of the kernel for the entire dataset, but because only close values make a significant contribution, most of the captured data can be ignored.

There is a trade-off between the accuracy of the calibration file and the time required to calculate the value of the kernel regression. A KD tree[28] allows accelerated spatial search over large (low dimensional) datasets. The kernel estimate process can be similarly accelerated by only summing samples within a specified volume. This preserves the function of the kernel regression without sacrificing real-time performance. We have thus paired the above kernel regression with a KD tree. Note that the above procedure generates a table for the forward kinematics of the arm.

\[
\text{Table I}
\]

<table>
<thead>
<tr>
<th>Table input</th>
<th>Table output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servo angles ( \vec{m} )</td>
<td>Tip position and orientation ( (\vec{p}, \vec{d}) )</td>
</tr>
<tr>
<td>Tip position ( \vec{p} )</td>
<td>Servo angles ( \vec{m} )</td>
</tr>
<tr>
<td>Tip orientation ( \vec{d} )</td>
<td>Servo angles ( \vec{m} )</td>
</tr>
</tbody>
</table>

Inverse kinematics are calculated using the same method to give a total of four KD tree tables (table I). This method
implicitly solves the problem of the coupling between the two arm sections because all the interactions are already captured in the table. Due to the coupling, when the tip moves in space the orientation also changes at the same time. The three inverse kinematic tables allow different methods of control for the tip. The input to the first inverse kinematics table is a position in space and the output is the motor angles that minimise the distance between the tip and the input position. The input of the second table is the desired orientation of the tip and the third table accepts both a position and tip orientation.

As the arm cannot self-intersect it is safe to linearly interpolate between the current and desired motor angles.

III. TASK SCENARIOS

In order to evaluate the interaction of the handheld robot with users we selected two generic spatial tasks, one for aiming in space (simulated painting) and one to pick and place objects. These tasks were used to investigate the effects of three levels of autonomy ranging from manual to semi-autonomous to fully autonomous operation.

In manual mode, the tool only performs actions that have been explicitly communicated by the user. The user has complete control over where the arm is pointing and when an action is activated by pressing the trigger. The end effector remains in its default (straight) state, and trigger presses from the user are executed without any autonomous intervention. In semi-autonomous mode, the tool has control over the pose of the end effector in space, but does not have control of the trigger. In this case, the user still decides when task actions occur, but does not have direct control of the position of the end of the arm. This mode is the first to incorporate task knowledge as the aiming of the tool is directed by this knowledge. In autonomous mode, the tool uses task knowledge to control both the pose of the arm and to override the trigger commands from the user. This is summarised in table II.

<table>
<thead>
<tr>
<th>Trigger state</th>
<th>Manual</th>
<th>Semi-autonomous</th>
<th>Autonomous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip pose</td>
<td>user</td>
<td>tool</td>
<td>tool</td>
</tr>
</tbody>
</table>

A. Painting Task

This task is an aiming-in-space task where the tool has to be positioned in 3D space to be able to use “virtual paint” on cells on a display screen. The aim is to completely paint a pattern on a wall as quickly and accurately as possible. A 27” display is used as the surface of the virtual wall and the user paints it by pointing the tip of the arm at the screen while holding down the trigger. Throughout the task, a template pattern is displayed on the screen in green on a black background. Aiming the tool at the screen and pressing the trigger will change the targeted pixel colour to red. The user paints patterns on the wall by selectively changing the appropriate pixels as indicated by the green template. A dot

is displayed on the screen which represents the calculated projection of the tool’s tip vector with the screen. Figure 9 shows the three different patterns used.

In manual mode, the user can aim the tool at any pixel they want and the tool will only paint the pixel if the trigger is pressed. The tip of the tool is not controlled autonomously and stays in its default state.

In semi-autonomous mode, the tool can control the state of the arm. The arm can change where the tip points without consulting the user using task knowledge of what needs to be painted. The tool aims to target the nearest contiguous unpainted template pixel while minimising the deviation from a straight state. For example, if the green template pattern shows two unpainted regions separated by a gap, and the user points the tool at one of the regions, the tool will autonomously paint all the pixels in that region as long as the trigger is held down. When the region is fully painted, the tool will not attempt to move to the second region on its own. This means that the user now has a tool that needs tactical positioning over regions which are then taken care of locally using the tool’s task information. Since the user fully controls the trigger, paint can land on out-of-pattern regions if the tool is aiming towards them.

In autonomous mode, the tool can override the user’s trigger input as well as control the direction the tip is pointing. The tool targets the pixel that is closest to both its current position and the pixel the base of the tool is pointing towards. When the user presses the trigger, it signals that the user wants to paint but task knowledge will not allow the tool to paint a pixel unless it is part of the template pattern. The tool will check the pixel that is currently targeted and override the trigger to an off state if the pixel should not be painted. Using the two regions example, the tool would paint the closest pixel at all times regardless of the gap between the two regions. When the tool tip moves over regions that are not supposed to be painted, the tool autonomously disables the painting function.

The brightness of each pixel in Figure 6 indicates the relative score for deciding the next pixel to paint. The yellow line shows the direction the base of the tool is pointing and the cyan shows the tip direction. The yellow line is assumed to be the intent of the user, so the tip will always try to point as close to it as possible.
B. Tiling Task

This task involves the picking up and dropping of real objects, a generic task as involved in e.g. sorting or assembly activities. The aim is to pick up red and black circular tiles sourced from two hoppers and place them in a predefined chequerboard pattern as quickly and accurately as possible. There are five black tiles in one hopper and five red tiles in a second hopper and the chequerboard pattern has room for nine tiles, five red and four black. When doing the task, users can see a picture of the correct pattern which is placed next to the chequerboard for reference. This is done so that the results are not confused by user’s different memory abilities. The tool is equipped with an electromagnetic tip that is activated by pressing the trigger. The tiles have small square steel plates so they can be picked up and dropped. The task is completed by picking up tiles from their respective bins and placing them on the board in the specified pattern.

In manual mode, the arm stays in its default straight state and the electromagnet only activates when the user presses the trigger.

When in semi-autonomous mode, the tool controls the position of the tool tip and autonomously seeks the next way-point in the task. For example, if the user is at a point in the task where they could pick up either a red or black tile, the tool will move towards the closest tile bin. If all the black tiles have been picked up already, it will only try to move towards the red tiles. When the tool is holding a tile, it will try to move towards the nearest corresponding empty space on the chequerboard and avoid incorrect drop-off points. In this mode, the electromagnet is still fully controlled by the user so the point at which the tile is picked up and dropped off is still under user control.

In autonomous mode, the tool has control over the tip position as above but now also automatically turns the electromagnet on and off. The tool is operated by moving the tool towards a tile bin until the tile has been picked up. The tool will then already start aiming towards an empty region on the target tile. The user then moves the tool towards the chequerboard where the tool will keep searching the nearest drop off point and turn the electromagnet off when the tile is sufficiently close to the drop off zone.

In both semi-autonomous and autonomous modes, feedback from the tool to the user is provided in an intuitive, implicit manner by the robot pointing towards, avoiding pointing, and/or refusing to perform an action when it conflicts with the task specification.

Figure 7 shows a storyboard close-up of the fully automated process of tile picking and dropping. The accompanying video shows the autonomous mode for this task in action.

IV. EXPERIMENTAL PROCEDURE

To evaluate the interaction between the user and the robot with the various modes of autonomy, user trials were conducted for the painting and tiling tasks. We recruited 16 volunteers between the ages of 25 and 40. Only one of the volunteers had prior experience with the robot. No reward was advertised to participants. Each volunteer was tested on two separate days with one task per day. The average session lasted around 15 minutes with rests between trials and modes. The experimental procedure was the same for each task:

1) The user is shown the robot and the experimenter demonstrates how it works.
2) The user is allowed to use the robot until they are satisfied they understand how to use it. Each participant must practice each mode at least twice.
3) An autonomy level mode is randomly chosen for the user to perform the task in.
4) The user fills out a NASA-Task Load Index (TLX)[29] survey after each mode.
5) Steps (3) and (4) are repeated until each mode has been tested three times.

A. Data Processing

The trial times were recorded automatically by the testing software. The NASA-TLX and amount of virtual paint used were also recorded automatically by the testing software. The accuracy of each tile placement was measured optically.

V. RESULTS

The painting and tiling trials were conducted on 16 participants each. The tiling task was performed twice for each of the three modes to fill a total of 96 boards (96×9 = 864 individual tile placements). Three patterns were available to be painted in each mode and both patterns and modes of operation were presented randomly for each individual. An average of 4 to 5 images were produced per participant that resulted in a total of 78 painting records.

To evaluate the results, we used repeated measures ANOVA with autonomy level as the independent variable.
and completion time as the dependent variable. The null hypothesis was that there was no difference in means between the three autonomy levels. If null hypothesis was rejected, the Bonferroni method was used to calculate the pairwise comparisons between the levels. We also did the same analysis for the other dependent variables: amount of paint used, average placement error, and combined TLX. This section presents the results of the null hypothesis test, and table III contains the pairwise comparisons for each mode. Figure 8 shows box and whisker plots of the recorded data.

**Completion Time**: The time taken to complete the painting and tiling task rejected the null hypothesis \((F(2, 14) = 88.8, p < 2.5 \times 10^{-15})\) and \((F(2, 14) = 6.32, p < 0.03)\) respectively.

**Accuracy**: The amount of paint used and tiling placement accuracy were not significantly different between the different levels of autonomy \((F(2, 14) = 1.34, p > 0.28)\) and \((F(2, 14) = 2.58, p > 0.093)\) respectively.

**Task Load Index Scores**: The difference in combined TLX means for the painting and tiling task was statistically significant using a Greenhouse-Geisser correction \((F(1.42, 14) = p < 4.3 \times 10^{-8})\) and \((F(1.44, 14) = 11.148, p < 0.01)\) respectively.

**Effort**: Figure 9 shows the recorded trajectories of the trial with the lowest frustration score on the TLX. The trajectory is recorded from the base of the tool, not the intersection point of the tip vector and the screen. In this task, the users did not move the tool towards or away from the screen so these plots simply show the x and y coordinates without significant loss of information.

**VI. DISCUSSION**

In general, increasing the level of autonomy reduces the completion time and perceived task load. This is most apparent in the painting task where there was a high level of significance between the difference in the means of the completion time for the different modes. For the other measures, there was no significant difference between manual and semi-autonomous mode. This suggests that the cooperation between the tool and user of the end effector placement did not have a detrimental effect on task performance, but that for significant gains to be made, giving full control to the tool was beneficial. During the experiment, users expressed frustration that, “The tool won’t go where I want it to” (paraphrase) and said that the tool often made mistakes when it chose a different pixel to paint from the one the user wanted. These effects were most apparent in semi-auto mode where the most cooperation was required. This could be improved by making the tool’s behaviour more predictable and by better modelling of the user’s intention.

The accuracy of the task did not correlate with differing levels of autonomy for either task. This is probably because the actuated end effector accuracy was similar to the human positional accuracy so the arm was not able to improve upon it.

Figure 9 shows a significant difference in trajectory behaviour for the three painting modes. The image illustrates the underlying user motion in the three different modes. All users chose large scale movement of the base over precise changes of orientation as the method for aiming at the pixels.
and this behaviour did not change across the different modes. In manual mode, the image structure is evident from the trajectory path as the user has to aim the tool precisely to paint the pixel pattern. In semi-autonomous mode, some of the fine image structure is retained in the tool trajectory but there are now long straight sections where the user moves the tool from one image region to another. In autonomous mode, most of the fine detail is lost as that is delegated to the tool so the majority of the path is composed of long smooth curves moving between regions of the image.

User opinions of semi-autonomous and autonomous mode were more split in tiling than in painting where everyone preferred automatic. In tiling, some users reported that they liked the extra control they had over tile placement while others preferred the ease of use. One user reported that when in autonomous mode they felt less responsible for the outcome of the task because it was “the robot’s fault” if it made a mistake. This statement is not entirely true as the user does have a large impact on the performance of the tool.

VII. CONCLUSIONS AND FUTURE WORK

This paper is about the design and evaluation of an example of a relatively unexplored type of personal robot. Handheld robots aim to share motion and cognitive abilities with users and work in cooperation. We develop a multi-DoF lightweight handheld robot around an optimised continuum robot structure. Calibration for forward and inverse kinematics is conducted via regression in order to capture the underlying complexities of such a structure. The proposed design is offered as an open source robot[1] that allows for evaluation of the interaction between users and high DoF sensor and actuation capable handhelds. In this paper, we concentrate on looking at how users interact and perform under different levels of automation. The robot gives feedback by pointing towards, avoiding pointing, and/or refusing to perform an action when it conflicts with the task specification. We carried out evaluations with a number of user volunteers in two generic tasks involving aiming in 3D space and picking/placing objects. Results were analysed using repeated measures ANOVA which showed that overall a high level of automation had a significant impact on some crucial aspects of task performance such as completion time and perceived workload. Future work will include the ways these tools can acquire task knowledge, the type of sensors to use, alternative feedback methods, modelling user intention, and further studies on how people adapt to carrying out tasks with handheld tools that have embedded cognitive abilities.

VIII. ACKNOWLEDGEMENT

We would like to thank the James Dyson Foundation for providing funding for this research and Laurie Bose and Rebecca Perez for technical advice.

REFERENCES