ROBOPuppet: Low-Cost, 3D Printed Miniatures for Teleoperating Full-Size Robots

Anna Eilering, Giulia Franchi and Kris Hauser

***Abstract*— ROBOPuppet is a method to create inexpensive, tabletop-sized robot models to provide teleoperation input to full-sized robots. It provides a direct physical correspondence from the device to the robot, which is appealing because users form an immediate "mental mapping" of the input-output behavior. We observe that untrained users can immediately exploit tactile and physical intuition when controlling the puppet to perform complex actions the target robot. The key contribution of this paper is a build procedure that embeds standardized encoder modules into scaled-down CAD models of the robot links, which are then 3D printed and assembled. This procedure is generalizable to variety of robots, and parts cost approximately seventeen dollars per link. We also present a simple software tool for fast calibration of the puppet-robot mapping, and a safety filtering procedure that sanitizes the noisy inputs so that the robot avoids collisions and satisfies dynamic constraints. A prototype ROBOPuppet is built for a 6DOF industrial manipulator and tested in simulation and on the physical robot.**

I. Introduction

Robots are teleoperated by humans in many applications, including nuclear plants, robotic surgery, explosive ordnance disposal, and search and rescue. But a common challenge for users is to learn a "mental map" of the correspondence between the input device and the target robot. Joint angle control, via buttons or joysticks, requires learning the nonlinear map from joint angles to workspace motion. Cartesian end-effector control is typically more intuitive, but the user must learn disparities between kinematic limits of the input device and those of the target robot. Moreover, end-effector motions may produce unexpected effects on other joints, which makes collision avoidance challenging. An alternate approach is often taken in programming from demonstration (PbD) [2] via *kinesthetic* teaching, in which the user directly pushes and pulls the robot. This is intuitive because the taught poses are in one-to-one correspondence with executed poses, and physical interaction immediately provides a sense of shape, dimension, and weight of the target robot. However, these systems require force sensing hardware and also cannot be used for remote teleoperation because they require physical co-presence.

This paper introduces ROBOPuppet, a new technique for building low-cost, kinesthetic robot control devices that duplicate the geometry and kinematics of a robot, but at smaller scale suitable for desktop use. The puppet is a 3D-printed miniature of the target robot with encoders embedded in the joints that translates the user's physical actions with the model directly to the robot's actions. The kinesthetic mode of operation is familiar to those who have played with action figures as a child, and we hypothesize that it lets users control complex motions in a more intuitive way than using joysticks and joint-level control.

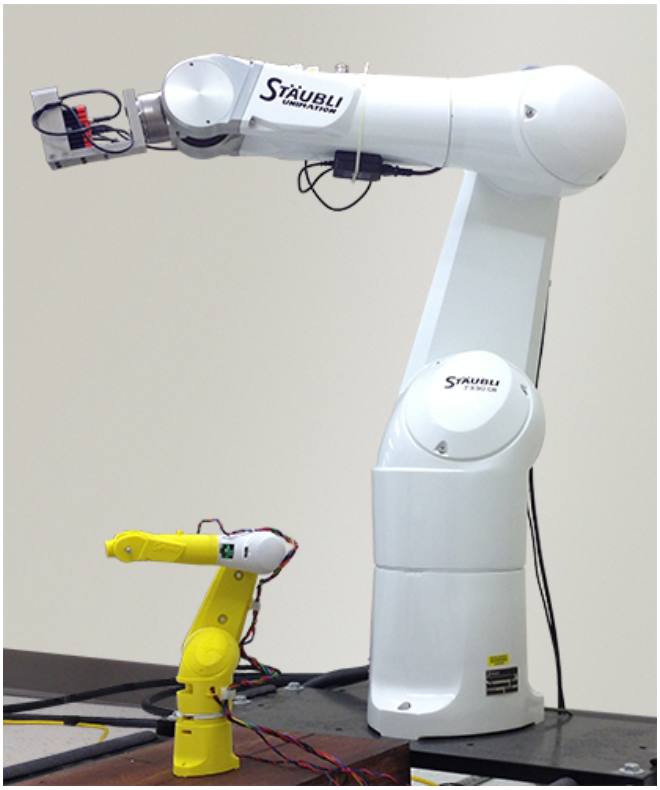

Fig. 1. ROBOPuppet next to target robot.

A key characteristic of the approach is its generality; it can be applied to a variety of different robots via the use of standardized hardware modules and geometry processing steps. The method is also highly accessible: we make use of inexpensive, off-the-shelf electronics and new rapid prototyping technologies, specifically the wide availability of 3D printers to non-specialized users [4] [16], to create models that accurately mimic the proportions, look, and feel of the target robot. Because the puppet is lower-fidelity, imprecise replica, we apply a sanitizing procedure to ensure the robot's safety. In summary, the contributions of ROBOPuppet are:

- A low-cost joint encoder assembly that is embedded

*This work is partially supported by a CRA-W Collaborate Research Experiences for Undergraduates (CREU) award and NSF CAREER award #1253553.

School of Informatics and Computing, Indiana University, Bloomington, IN 47405 USA

{annaeile,gfranchi,hauserk}@indiana.edu

in the printed parts of the ROBOPuppet to provide encoder readings and adjustable friction for maintaining its configuration under gravity.
- A systematic, step-by-step process for building a custom ROBOPuppet for a new robot via geometry preprocessing, 3D printing, and assembly.
- A calibration tool for easily calibrating a mapping from encoder values from the puppet to desired joint angles of the robot.
- A real-time planning method for translating puppet movements into robot movements that respect the robot's dynamic limits and avoid collisions with known obstacles.

As a preliminary demonstration of the ROBOPuppet method, we built a prototype puppet of the Staübli TX90L 6DOF robot arm. A 30% scale device was created and used to control the arm both in real-time simulation as well as on the physical robot. The prototype puppet costs a total of $85. Instructions, requisite 3D models, and software for ROBOPuppet are available at http://robopuppet.org

II. Related Work

The Staübli TX90L robot is controlled manually using the SP1 control pendant (Fig. 2), which allows a user to control joint angles or Cartesian motion via twelve buttons (increase and decrease value on each axis) along with two buttons to increase/decrease speed. This is a typical controller for an industrial robot. Other robot control devices range from computer GUIs [5], haptic controllers [7], joysticks, 3D sensors, and game control pads [13], [18].

Input devices can be categorized between joint level vs cartesian control and kinesthetic vs non-kinesthetic approaches. [9] Cartesian control is often more intuitive than joint-level control, but requires careful design of the mapping to joint space to avoid singularities and kinematic limits of the robot and limits of the input device. Furthermore, it is challenging in Cartesian mode to control non-end effector points on the robot (for example, an elbow). Non-kinesthetic devices like control pendants, GUIs, and joysticks are often more ergonomically suited for long-term operation, e.g. holding a pose for a long time. Kinesthetic control devices like haptic devices translate bodily movements in workspace to robot movements in workspace, and are often more appealing for novice users to operate due to the ability to feel forces felt by the robot [3], [14]. In contrast, ROBOPuppet proposes a control device in one-to-one correspondence with the target robot, combining the benefits of direct joint control and kinesthetic feedback.

Direct kinesthetic control of joints is not a new idea. The most widely used technique is Programming by Demonstration (PbD) for teaching configurations to robots via direct physical manipulation [2]. This has been applied to several commercial industrial robots. However, physical manipulation of the robot is not possible in remote environments. The most closely related work to ours is the development of a 5-axis robotic motion controller to teleoperate an industrial robotic arm [15]. The application of this controller is similar to one considered here, but is more expensive, requires significant design and mechanical engineering expertise, and only matches the link lengths and joint angles of the target robot. By contrast, ROBOPuppet is a generally applicable process for building low-cost control devices, it requires very little experience to build and assemble, and the device matches the geometry of the robot.

III. Summary

The method consists of five major elements.

1) Standardized, inexpensive *joint encoder assemblies* that contain encoders, can be adjusted according to the joint orientation of the links of the robot, and enable the puppet to support its own weight against gravity using adjustable friction controls.
2) *Pocket geometries*, which are 3D models which are subtracted from the printed parts to provide structures for holding the base of the encoders in the joint surface and the encoder shafts in the opposing joint surface. They also provide holes for bolts used to control the friction between joint surfaces, and access ports for wiring and other innards of the puppet that are necessary for assembly.
3) A straightforward *build process* for building a custom ROBOPuppet for a new robot using a robot CAD model, the provided joint encoder assemblies, and pocket geometries in conjunction with 3D printing. Wiring is also completed in this step.
4) A *calibration tool* that allows the user to easily create a mapping from the input encoder values to appropriate joint angles on the robot. This tool asks the user to place the puppet into several poses and automatically calibrates the mapping.
5) An *input sanitizer* that ensures that requested motions are safe and feasible for the robot to perform. A real-time motion planner translates raw inputs to output trajectories that satisfy kinematic and dynamic limitations of the robot. This element can also avoid self-collisions and collisions with the environment, if an environment model is available.

We claim that **a viable ROBOPuppet can be constructed for a target robot** at any scale such that the scaled links fit within the 3D printer's workspace, and are sufficiently large to embed the joint encoder assemblies.

IV. Construction

The construction process can be further subdivided into geometry preprocessing, printing, and assembly phases. To build the ROBOPuppet for a new target robot, the robot's CAD model is scaled such that the smallest joint surface to be instrumented can successfully hold the joint encoder assemblies and, once scaled, the void primitives are subtracted out of each part. This section presents a step-by-step process for installation and assembly of the final puppet out of printed parts and joint encoder assemblies. The workflow for this process can be seen in Fig. 3.

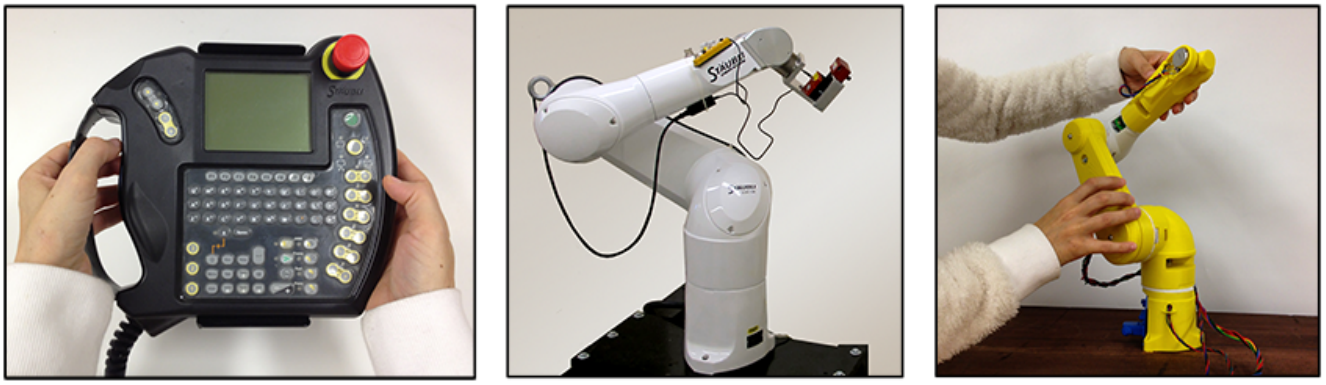

Fig. 2. SPI Control pendant(left) for the Staübli TX90L robot(center) compared with the ROBOPuppet controller(right)

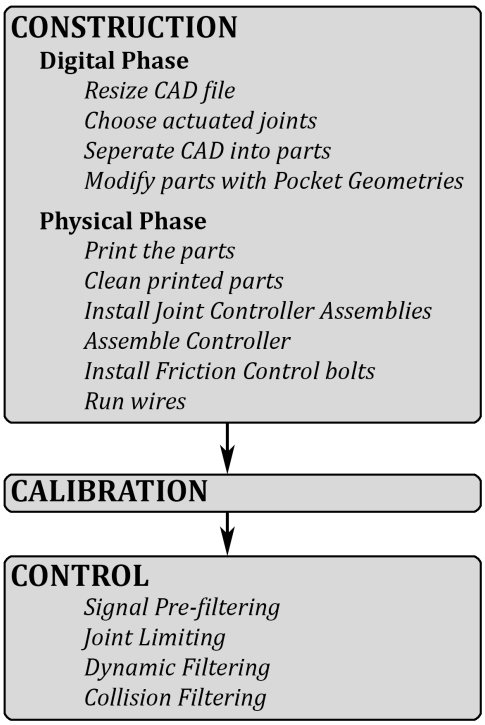

Fig. 3. The general workflow for using the ROBOPuppet method to create a ROBOPuppet controller for a new robot.

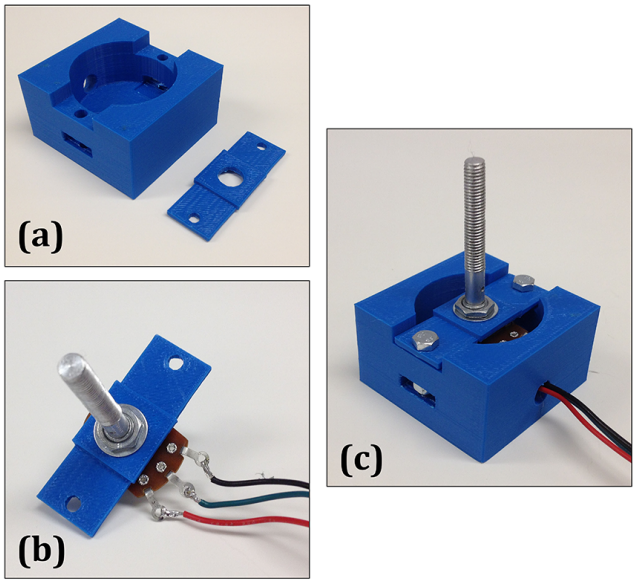

Fig. 4. (a) Printed part with modified joint surface and mounting bracket. (b) Joint encoder assembly is installed into bracket (c) Bracket with mounted joint encoder assembly is installed in joint surface using bolts through the bracket attached with nuts accessed in the installation access tunnel.

A. Joint Encoder Assemblies

Our joint angle encoders use 5kΩ linear taper potentiometers. They are carved with threading down the length of the encoder shaft for the installation of the friction control bolt and a small hole is drilled to contain the roll pin, a small pin used to translate motion of the part into motion in the encoder shaft. The potentiometer joint encoder assembly has a base dimension of 2.5 cm x 3.25 cm x 1.75 cm with a shaft of diameter of 0.75 cm and a length of 5.25 cm and is installed in a printed bracket that is sized to easily fit in the modified parts. A mounted joint encoder assembly, ready to be installed in the printed parts, is shown in Fig. 4(b).

B. Digital Phase

During the digital phase the target robot's CAD model is downloaded and the robot is scaled to the desired size. The minimum scale factor for a ROBOPuppet can be determined by identifying smallest joint surfaces that are to be controlled and ensuring that these surfaces can hold the joint encoder assemblies. The user chooses the joints to instrument and the CAD file is separated along these joints into individual meshes suitable for printing. Each mesh is modified using pocket geometries to provide space to access and install the joint encoder assemblies. (Fig. 5)

Pocket Geometries Our build process uses three *pocket geometries* that have been designed to ease the modification of all model links to fit the joint encoder assemblies.

1) Joint Encoder Pockets - complex geometries that create a void in which the base of the encoder resides, spaces

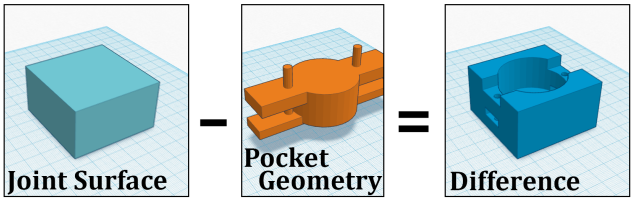

Fig. 5. Pocket geometries are subtracted from the joint surface to create the voids, pockets and tunnels neccesary to embed and install hardware components.

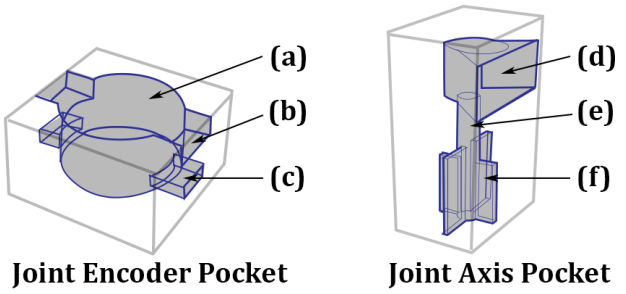

Fig. 6. (a) Encoder void, (b) Encoder bracket mount point, (c) Installation access tunnel, (d) Friction bolt void, (e) Encoder shaft pocket, (f) Roll pin pocket

to mount a bracket to affix the encoder to the joint surface and access tunnels to assemble the controller.

2) Joint Axis Pockets - geometries that create a pocket to hold the encoder shaft on opposing joint surfaces with voids for the roll pin and tunnels and voids for the friction control bolt.
3) Convenience voids - simple geometries used to create access tunnels and voids in printed parts to give access during installation and maintenance.

The pocket geometries were created in such a way that, during mesh modification, they are to be centered about the axis of rotation; this makes them simple to use and ensures that the encoder's shaft is centered properly and the part moves correctly when the controller is assembled. Examples of pocket geometries embedded within joint surfaces can be seen in Fig. 5 and Fig. 6.

C. Physical Phase

The physical phase of the construction stage involves printing the parts, installing joint encoder assemblies into the joint surfaces, and assembling the parts into the finished controller.

Printing. The puppet is printed on a 3D printer. Post processing must be done in order to clean any extraneous material from the printing process.

Hardware installation. At this point polystyrene foam sheeting is installed on joint surfaces to aid in friction control and creating a smooth action. Finally, joint encoders assemblies should be installed using the bracket mounts created in each joint surface. (Fig. 4)

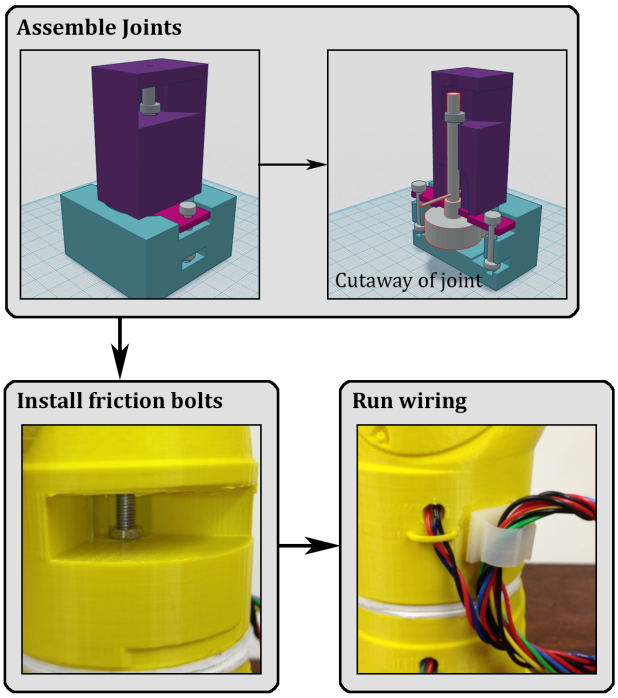

Fig. 7. Installing the joint assemblies and friction control bolts in the printed model.

ROBOPuppet Assembly. Joints are assembled by first installing a roll pin into a hole pre-drilled into the encoder shaft and fitting the encoder shaft into the prepared joint axis pocket. Friction bolts are installed during this process to both fix the joint bodies together and adjust friction and action between the joint surfaces. Wiring was run on the outside of the model's surface and affixed using hot glue and purchased brackets with care taken to ensure there was enough slack between joint allow a full range of motion (Fig. 7). Once the puppet is assembled, the joint encoders are aligned to ensure that the encoder limitations roughly match to the joint limitations on the target robot.

V. Mapping Puppet Motions to Robot Motions

Once RoboPuppet is built, its joint angle values must be mapped to matching joint angles of the robot and transmitted to the robot controller. Free software solutions are used to keep the barrier of entry low. Reading the joint encoder values was done using the Arduino environment [1] and all calibration and control tasks were completed using the Klamp't software library [11]. We must overcome two challenges: first, the mapping from encoder values to the robot's joint angles must be calibrated; second, direct transmission of commanded joint angles leads to unsafe behavior. This section describes our approach to these issues.

A. Calibration

Since the puppet uses linear potentiometers the mapping from encoder values to joint angles is also linear. To calibrate it we use a tool that asks the user to pose the puppet

in a number of target robot configurations (two or more) displayed in a 3D GUI. The correspondence between puppet pose and robot configuration is then estimated using linear regression.

Because posing the puppet by hand with visual comparison is not precise, we consider several techniques to reduce errors. The first option is to simply use many configurations and hoping errors average out. This is somewhat tedious for the user, and and users can make systematic errors. A better option is to choose target configurations that *use physical correspondences* to reduce errors. Examples include configurations that lie in local minima / maxima of the gravity potential, configurations that make contact with natural landmarks, such as the puppet's table plane or self-contact, or configurations that line up with natural features on the robot's geometry. Due to the absence of strong lines on the Staübli TX90L we use the gravity- and contact-based approaches, with configurations that are vertically outstretched and configurations that touch the table with the elbow and end effector tip.

We also observed that asking users to simultaneously pose many joints is harder than just posing one or two joints. A total of 5 configurations were used in our calibration program, with at most 2 joints moved between configurations (Fig. 8).

B. Safety Filtering

Instead of direct transmission of the puppets motion, we introduce filtering methods for overcoming the following safety issues:

1) The encoders are relatively low-precision and suffer from jitter,
2) Commanded values may violate the robot's joint limits due to mismatches in the encoders' joint stops,
3) The puppet may move too quickly for the robot to catch up,
4) The puppet may cause the robot to self collide or collide with objects in the remote environment.

These are addressed in sequence by the following steps.

Signal Pre-filtering. Because the puppet's analog encoders are digitized to 10-bit values, the digitized values tend to oscillate between two adjacent values even when the puppet is not moving. To smooth these oscillations a deadband filter of width 1 is applied to the signal, as well as an exponential filter with smoothing factor 0.5, before applying the calibrated mapping.

Joint Limiting. The smoothed joint angle command is capped to lie within the robot's joint limits.

Dynamic Filtering. The puppet often moves or accelerates faster than the robot can, so we limit the joint angle command by the robot's velocity and acceleration limits. Simple acceleration and velocity limiting introduces oscillatory effects and can even cause the robot to overshoot its joint limits due to insufficient stopping room. So, we use an online trajectory optimizer [10] to produce time-optimal velocity- and acceleration-bounded trajectories that start at the robot's current configuration and velocity, and end in the target configuration. This optimization has an analytical piecewise-quadratic solution and hence can be done quickly enough to be performed at every time step. The effect is similar to the trajectory generation approach of [12].

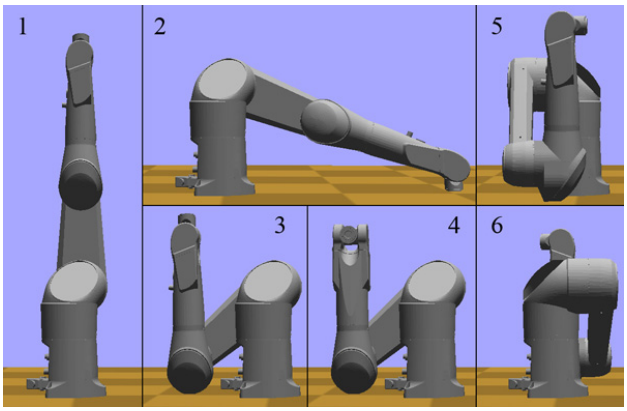

Fig. 8. To calibrate the puppet, the user is asked to pose it in a sequence of configurations.

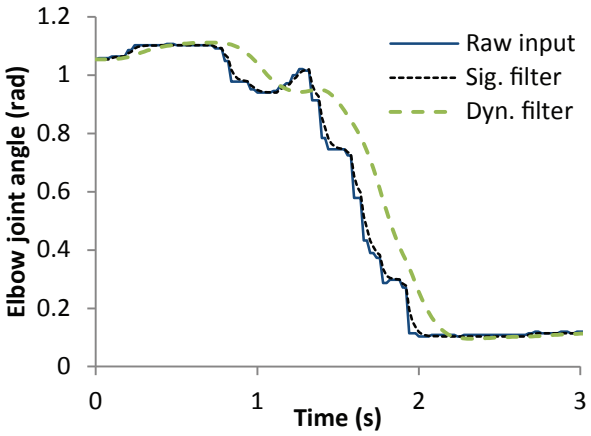

Fig. 9. Raw, noisy joint angle commands (Raw input) are passed through a signal pre-filter (Sig. filter) to eliminate jerks, and then through a dynamic filter (Dyn. filter) to generate smooth trajectories that respect the robot's velocity and acceleration limits.

The effects of signal filtering and dynamic filtering are depicted in Fig. 9.

Collision Filtering. If the robot has a model of obstacles in its environment, then it can decide whether to accept the motion to the target configuration via collision detection (Fig. 11). Specifically, our system first generates a candidate optimal trajectory to the target configuration, then runs collision detection on the trajectory (details in [10]). If the trajectory is collision free, it is accepted and the robot begins moving along it. If not, it is rejected and the robot continues along its current trajectory until the next input from the puppet is received.

VI. Prototype Puppet for an Industrial Robot

Our prototype is a 30% scale controller for the Staübli TX90L robotic arm. The arm has six degrees of freedom, of which the puppet includes five. (Fig. 12) At this scale, the

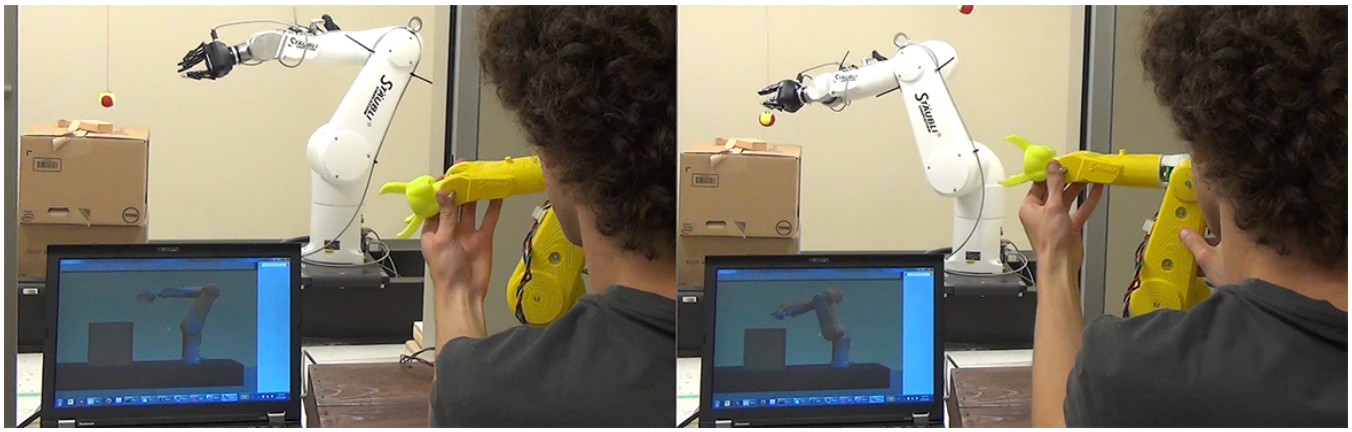

Fig. 10. A first time user controls the Staübli TX90L robotic arm with the ROBOPuppet controller. The user was asked to interact with targets in the environment (hanging tennis balls and blocks on the table).

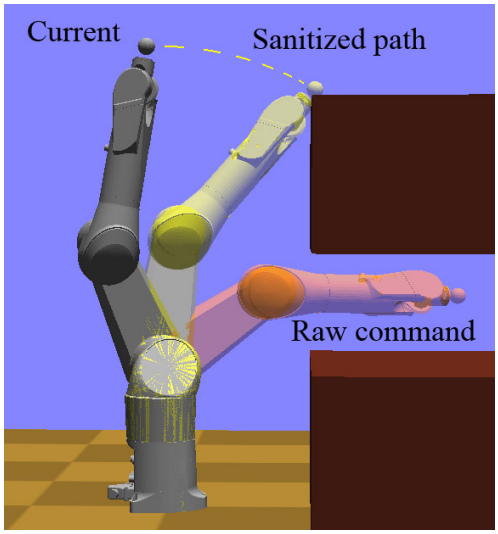

Fig. 11. The input sanitizer includes a collision filtering step that prevents the robot from colliding into known obstacles.

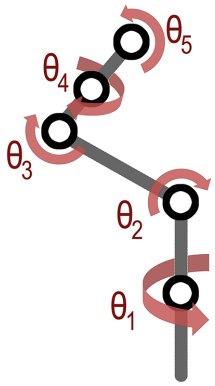

Fig. 12. Five of the TX90L joints were instrumented, and the joint surfaces of each was selected in such a way to ensure the final controller mimicked the target robot's range of motion.

sixth joint is too small to contain the joint encoder assemblies and, thus, was not included.

CAD Preprocessing. To keep the entry requirements for creating the controller low, we chose to use simple and free software (Freecad [6] and TinkerCAD [17]) to modify the robot's CAD files. A user with no previous 3D modeling or CAD experience was able to prepare each part for printing in approximately 15 minutes. This can be compared to 45 minutes or longer when not using premade pocket geometries.

Printing and assembly. The device was printed on a Makerbot Replicator 2X using ABS 1.75 mm filament. 3D printing took 29 hours (Table I) and required 0.483 kg of filament. Once the parts are printed, the assembly and joint encoder alignment can be completed in approximately 1.5 hours. Encoders and additional hardware were chosen to be affordable and readily available. All parts required for the model can be sourced either from local stores or the internet and are chosen to be simple to use and wire.

TABLE I

TIME TAKEN FOR MODIFYING THE CAD MESH USING POCKET GEOMETRIES AND PRINTING. (HOURS:MINUTES.SECONDS)

Link	Digital Preprocessing	Printing
L1	00:15	05:10
L2	00:20	06:37
L3 - top	00:10	02:57
L3 - bottom	00:10	04:14
L4	00:25	04:03
L5	00:15	04:07
L6	00:10	00:44
TOTAL	01:45	27:52

Printing is by far the most limiting step in our method. "Prosumer"-grade 3D printers can require near-constant supervision, but rarely require intervention. As a result, most of the CAD processing and hardware installation tasks can be completed while waiting for the model to complete printing.

Cost. Table II shows the costs of individual components, with a total puppet cost of approximately $85. For further savings, the most expensive component, the Arduino UNO,

can be replaced with a less expensive microcontroller.

TABLE II
COST FOR ALL PARTS AND MATERIALS.)

Equipment	Cost	Amount	Total Cost
5k-Ohm Linear Taper Potentiometer	3.49 each	5	$17.45
Stranded wire	10.00 kit	0.5	$5.00
Solder	6.49 spool	0.1	$0.65
Solderless Breadboard	8.90 each	1	$8.90
Arduino Uno R3	29.95 each	1	$29.95
ABS filament	0.048 per g	483	$23.18
TOTAL			$85.13

Preliminary testing. The device and controller was implemented in a physics simulator [11] as well as on the target robot, as shown in Fig. 10 and in the supplemental video. Undergraduate students, seniors, and children as young as 6 years old were able to perform a sequence of trial reaching tasks without instruction. Our observations are that large and medium scale movements are easily controllable, and collision filtering is necessary due to accidental movements, like dropping the puppet. We also observed that fine-grained positioning (within millimeters) is not yet possible due to the low fidelity of the encoders, but we imagine higher grade encoders could be used, or the puppet could be used in conjunction with alternative control methods.

VII. CONCLUSION AND FUTURE WORK

This paper presented ROBOPuppet, a technique for building miniature robot control devices using 3D printing and low-cost electronics. It consists of a standardized encoder module, 3D printing procedure, and assembly process that is generalizable between a wide variety of robots. Novel methods for calibration and real-time input sanitization for collision prevention are also presented. A prototype puppet is presented for a 6DOF industrial robot with a materials cost of $85. In the near future we will produce another prototype for a more complex robot, such as the Rethink Robotics Baxter. We also intend to share parts and modifications [8], with the goal of cultivating an online community of researchers, citizen scientists, and robot enthusiasts.

The current iteration of ROBOPuppet has a few limitations that we intend to address in future work.

- We used encoders with a range of 300°, which may not be sufficient to capture the joint range of a given robot. Multi-turn or continuous encoders could be used instead.
- Only hinge type joints or joints that can be represented using rotary motion can be replicated using the current method. We are exploring solutions for alternative joint types.
- Robot joint limits could be implemented as physical stops in the puppet itself, providing immediate feedback.
- The current joint encoder assemblies may not fit into small links, placing a lower bound on the size of the model. We are currently working on a smaller joint encoder assembly.
- Mobile robot navigation control may be implemented with new encoder modules, such as the trackballs of computer mice.
- The current implementation does not provide haptic feedback. Standardized motor/encoder assemblies could provide this functionality.
- Direct control may be inappropriate for robots that must walk and/or maintain balance (e.g., bipeds). Separate balance control filtering may need to be implemented.

We also wish to perform user studies to test the hypothesis that ROBOPuppet makes robots easier to control than alternative input modalities, such as joysticks, mice, and off-the-shelf haptic devices.

REFERENCES

[1] Arduino: an open-source electronic platform for protyping. `http://www.arduino.cc/` (accessed Feb. 6, 2014).

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Springer Handbook of Robotics*, chapter Robot Programming by Demonstration. Springer-Verlag, 2008.

[3] B. Dariush, M. Gieger, A. Arumbakkam, Y. Zhu, B. Jian, K. Fujimura, and C. Goerick. Online transfer of human motion to humanoids. *International Journal of Humanoid Robotics*, 6(2):265–289, 2009.

[4] I. Ebert-Uphoff, C. M. Gosselin, D. W. Rosen, and T. Laliberte. *Cutting Edge Robotics*, chapter Rapid Prototyping for Robotics. Pro Literatur Verlag, Germany, 2005.

[5] T. Fong and C. Thorpe. Vehicle teleoperation interfaces. *Autonomous Robots*, 11(1):9–18, 2001.

[6] Freecad:an open source parametric 3d cad modeler. `http://www.freecadweb.org/` (accessed Feb. 6, 2014).

[7] Geomagic haptic devices. `http://geomagic.com/en/products-landing-pages/haptic` (accessed Feb. 6, 2014).

[8] Thingiverse. `http://www.thingiverse.com/` (accessed Feb. 6, 2014).

[9] C. Guo. *New Paradigms for Human-Robot Interaction Using Tangible User Interfaces*. PhD thesis, University of Calgary, 2008.

[10] K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *Intl. Conf. Rob. Automation*, 2010.

[11] Klamp't. `http://klampt.org/` (accessed Feb. 6, 2014).

[12] T. Kroger and F. M. Wahl. Online trajectory generation: basic concepts for instantaneous reactions to unforeseen events. *IEEE T. Robotics*, 26(1):94–111, 2010.

[13] S. Lee, G. Sukhatme, J. Kim, and C.-M. Park. Haptic control of a mobile robot: a user study. In *IEEE / RSJ International Conference on Intelligent Robots and Systems(IROS)*, Lausanne, Switzerland, 2002.

[14] A. Okamura. Methods for haptic feedback in teleoperated robot-assisted surgery. *Industrial Robot: An International Journal*, 31(6):499–508, 2004.

[15] A. Payne. A five-axis robotic motion controller for designers. In *ACADIA Conference: Integration Through Computation*, Banff, Canada, October 13 - 16 2011.

[16] 1 G. Reshko, M. Mason, and I. Nourbakhsh. Rapid prototyping of small robots. *Tech. Report CMU-RI-TR-02-11, Robotics Institute, Carnegie Mellon University*, 2002.

[17] Tinkercad. `https://tinkercad.com/` (accessed Feb. 6, 2014). Member of Autodesk, 123D family of products.

[18] B. Yamauchi. Packbot: a versatile platform for military robotics. *Unmanned Ground Vehicle Technology VI*, 2004.