Automated *In Situ* Placing of Metal Components Into 3-D Printed FFF Objects

Daniel Ahlers, Florens Wasserfall, Norman Hendrich, Arne Niklas Büngener, Jan-Tarek Butt, and Jianwei Zhang, *Member, IEEE*

***Abstract*—Screws are often used to connect 3-D printed parts to other objects. When screwing directly into printed plastic, the reliability is limited, and the connection can wear out over time. For more reliable connections, standard metal nuts are often inserted into slots designed into the object. This article presents an approach where nuts and other ferromagnetic components are integrated directly into the part while printing it. Our prototype machine is a modified Prusa-I3 fused filament fabrication printer with an electromagnetic pick and place tool. We introduce augmented slicing software, where the user can insert generic component models from a library and place them at arbitrary positions in the object. Cavities for the components and additional G-code commands for robotic placing are automatically generated and sent to the printer. A printed component tray is attached to the printbed, allowing different part configurations for each print. The pick and place unit is controlled by our OctoPrint plugin OctoPNP.**

***Index Terms*—3-D printing, fused filament fabrication (FFF), multimaterial printing, rapid prototyping, smart manufacturing.**

I. Introduction

BESIDE decorative objects and simple toys, functional 3-D printed objects are rarely used standalone but rather in combination with other parts. The main reasons are the use of nonprinted parts, such as electronics, mechatronic components, and material combinations not supported by the printer. Also, objects too large for the build volume of a given machine or objects with unprintable geometries or requirements for complicated support structures can often be assembled from several smaller parts, printed with proper orientation. A good example is the Prusa i3 3-D printer used in this work [1]. The design of this fused filament fabrication (FFF) printer combines a variety of printed parts with those components that cannot be built efficiently with current FFF technology (namely the metal frame and guide rails, hotend, motors, and the electronics). It is considered as an evolution of the replicating rapid prototyper concept [2], [3] of self-replicating machines [4], as the printer can manufacture parts of itself.

Due to printed plastics' material properties, only three of the common mechanical assembly methods of multiple parts [5] are realistic choices for the FFF process. The first option, *gluing*, forms a solid and usually permanent connection, as long as a suitable type of glue is used. However, there are material combinations where a strong glue simply does not exist. A second method is to *snap-fit* or slide components together. This connection requires tight tolerances, which are hard to achieve with FFF printing. Snap fitting is suitable for connections with little to no mechanical stress and is often used for lids and covers. These connections need to be disassembled very carefully because the snap connections get damaged easily. The third method is *screwing* together different parts, resulting in strong connections regardless of the material combinations. Screw connections can also be disassembled easily and are often used in combination with mechanical components and electronics.

From the geometry point of view, a standard screw connection needs a hole through the components, with enough room for a screw head in the first component and a connection point in the second component. In FFF printed components, this connection point can be done in different ways. The easiest way is to drive a metallic screw directly into a slightly undersized printed hole. Standard metric screws are not well-suited for this due to their fine and small threads, but there are special screws for *directly screwing* into plastic. These screws are made for permanent connections and are not suitable for frequent unscrewing due to wearing effects in the plastic. Direct screwing is also not suitable for moving threads like in a spindle. Another way is to use a *threaded insert* melted directly into a specifically sized hole in the plastic. Such threaded inserts are suitable for nonpermanent connections that are often unscrewed or used for adjustments. Threaded inserts do not add any pull-out strength to the connection, but they provide a better torque resistance [6]. The third and most common way to connect 3-D printed objects is by *securing the screw with a nut*. These connections can be used to tighten and release screws multiple times and are also suitable for permanently moving connections. The nut can be placed directly on the back surface of the second plastic part

Manuscript received December 30, 2020; revised March 26, 2021; accepted April 30, 2021. Date of publication May 7, 2021; date of current version August 13, 2021. Recommended by Technical Editor M. Grossard and Senior Editor X. Chen. This work was supported in part by the Federal Ministry for Economic Affairs and Energy in the ZIM-KamEl project and in part by the German Research Foundation DFG in projects Crossmodal-Learning, SFB TRR 169 and SFB TRR 169-2. *(Corresponding author: Daniel Ahlers.)*

The authors are with the Informatics Department, Universität Hamburg, 22527 Hamburg, Germany (e-mail: ahlers@informatik.uni-hamburg.de; wasserfall@informatik.uni-hamburg.de; hendrich@informatik.uni-hamburg.de; 4buengen@informatik.uni-hamburg.de; jan-tarek.butt@studium.uni-hamburg.de; zhang@informatik.uni-hamburg.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2021.3078409.

Digital Object Identifier 10.1109/TMECH.2021.3078409

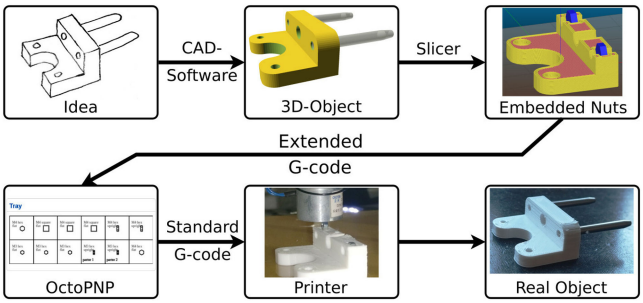

Fig. 1. Design flow supported by our toolchain. Left: Starting from a standard 3-D model. stl file, the user selects the magnetic parts to be added and places them using the interactive 3-D GUI. The object is then sliced, when cavities are automatically calculated for the selected objects, and G-Code for insertion and placement of nuts is generated. The final G-Code is then executed on the printer autonomously.

or in a nut-sized cavity, with the nut typically inserted from the object's back. Alternatively, if the hole cannot go through the whole object, the nut has to be slid in from the side. Design for assembly must be considered, as sliding-in nuts easily fall out during manual assembly, and other components sometimes block the slots.

However, a nut sliding pocket adds weakness to the connection since it is not fully supported by material on all sides. To get rid of this weakness and keep the nut in place, it can be inserted directly into the plastic during printing. Until now, this is a complex and lengthy manual task. First, the needed cavities have to be designed into the plastic object by hand. This includes the printer-specific tolerances, which are then hard-coded into the model and cannot be changed afterward. To place a nut, a pause command has to be added to the G-code manually. Whenever the printer stops, the nut must be placed by hand [7], and the print continues until the next nut.

To get rid of these manual steps, we have modified a Prusa-i3 printer with an electromagnetic pick and place tool and a camera to automatically place components into the objects while printing. This article describes the hardware modifications and our customized software.

The rest of this article is organized as follows. Section II first summarizes relevant previous work. Section III introduces the proposed design flow and explains the modified slicing software, where the user can interactively place custom components into the plastic object. The slicing software then generates all needed cavities and the extended G-code for plastic printing and object placement. Section IV describes how the magnetic pick and place tool is mounted to the printer, how the OctoPNP plugin handles the pick and place process, and how the printer is calibrated to ensure precise placement. Fig. 1 illustrates the design and placement process, as presented in this article. Some demo objects using different nut orientations are shown in Section V. Finally, Section VI concludes this article.

II. Related Work

The basic idea of picking-up components and placing them somewhere else is at the very heart of manufacturing automation. Many different techniques, including mechanical grippers, vacuum-based systems, or electromagnetic picking, have been studied in detail. Mechanical grippers use one or more actuators to grip the object on the outside, similar to a human hand. The gripping is either done by enclosing the object or by clipping the object from opposite sides [8]. These grippers are complex to build and finding a working grip needs precise information about an object's geometry. Vacuum picking works by creating a vacuum in a pickup device that fits the object's size to pick. Vacuum pick and place is widely used in printed circuit board manufacturing, especially to place surface mounting devices [9]. A flat surface near the center of mass is needed to pick up an object with vacuum. Electromagnetic grippers pick up objects with a switchable magnet [10]. For this to work, the objects need to be magnetic, whereas the surrounding area has to be not magnetic. Electromagnetic grippers are easy to build because they only consist of a magnetic coil, controlled by a switch, mounted to one or more motors to control rotation.

Kataria and Rosen [11] conducted early research on the integration of insets into SLA-fabricated parts, where the external structures are inserted during the printing process, but are allowed to protrude from the current build surface. Their main focus lies on SLA specific aspects: laser shadowing by the inset, support structures, and vat recoating.

Glasschroeder *et al.* [12] showed a technique to insert nuts into an object printed with a binder jetting printer. Due to the low material strength and the printer's low accuracy, these nuts were needed to get a strong connection between multiple parts. To place the nut, a cavity is designed into the part manually. Since the printing process fills everything with powder, they built an automatic vacuum powder remover into the printer to remove the material. The nut is then placed manually and the last layer is printed again to fill all holes in the powder bed that emerged during material removal. The same approach was also used to embed electronic components, which were automatically placed by a vacuum gripper and connected by conductive material, filled into channels generated by the powder removal tool [13].

Several groups have considered the design of 3-D printed electronics, often combining different printing technologies for the individual object generation steps. An overview of recent activities in this field is given by Espera *et al.* [14]. Often, direct-write techniques are used, where material dispensing or piezojetting are utilized to print conducting tracks to connect the electrical components. Positioning and integration of electronic components, usually in SMD technology, are important but not very well covered aspects of 3-D printed electronics. Positioning, cavity generation, and physical insertion are often entirely done manually [15], [16]. Baily *et al.* [17] proposed a combination of modified CAD and slicing software to integrate both wires and SMD components. They used SolidWorks for CAD modeling and Cura as slicing software. Both were extended with custom plugins to import electronic component models from Eagle schematic files, which are then cut-out from the plastic object. Carranza *et al.* [18] introduced a similar approach to import existing netlists and component geometries into a Blender plugin. None of the aforementioned uses robotic placing during or even after the printing process.

In our previous work, we have built an experimental hybrid manufacturing production cell that can print plastic filament as well as electronic wires using conductive paste, and which is also

equipped with a vacuum pick and place tool. With this machine, 3-D printed objects with integrated electronics can be printed fully automatically. A software toolchain for the design of those objects with integrated circuits was developed by extending an open-source slicing software [19], [20]. The user can position SMD components from imported Eagle schematics into the 3-D object and the software then generates cavities for the electronic components. An approach for topology-aware automatic wire-routing between all components without collisions, following the printed object's inner structure, was published recently [21]. The project also includes control software to automate visual inspection, documentation, and verification of the printed electronics [22].

Büngener extended this existing software toolchain in his thesis [23] to also integrate nuts and other magnetic objects into the printing process and place them automatically with a modified FFF printer. Some concepts and code from previous work were reused for this article to solve a related, but much more practical problem, with potential for immediate wide application. The integration of nuts or similar elements is fundamentally different from electronic components in a sense that it requires a force transmitting connection, firmly anchored to the surrounding material, instead of very smooth, uninterrupted layers below the electronic components. Compared to our previous work, the hardware costs were reduced significantly. The machine reported here can be assembled for less than 1000 $. The general development of additive manufacturing systems is at a transition state now, where plastic processes have become more or less stable and reliable enough and we are increasingly able to do advanced hybrid- and multimaterial manufacturing with very cheap hardware systems.

III. Design Process

Integrating external components into a 3-D printed model requires consideration of several aspects at different stages of the design and preparation process.

First, the position of nuts and bolt holes in the object model must be defined. This mostly depends on functional requirements and is typically done at the design stage in a CAD software. The position of screws in models specifically designed for 3-D printing is often influenced by process-specific properties, e.g., the build direction to avoid fractures when force is applied perpendicularly to the layer orientation.

Second, the slicing software should be aware of the metal components during G-code generation for several reasons. Small printed holes generally suffer from shrinking effects in the FFF process. If the slicer generates nut-cavities, the cavity margins can be adjusted for each specific printer in the slicing profile. This also allows the user to modify existing CAD object models by inserting components at slicing time (e.g., to change a designed plastic thread to a more robust embedded nut). During the G-code execution on the printer, pick and place operations must be triggered at the right moment. Therefore, they should be included in the G-code, along with the position and shape specification of the hardware components.

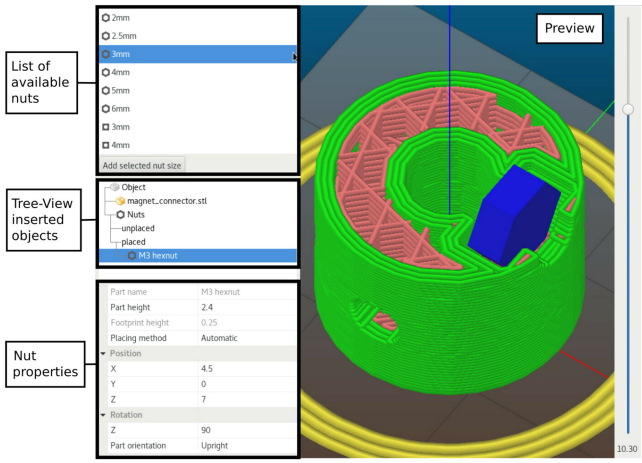

Fig. 2. User interface of the proposed system, with a 3-D preview of the model and placed nut.

Third, the printer itself requires a user interface to indicate the list of components required for a particular object and a control algorithm to execute the actual pick and place movements during the print.

Our implementation of the nut positioning and cavity generation is based on the open-source slicing software Slic3r [24]. We further augmented the modified version of Slic3r that already supported insertion of SMD components, originally intended for 3-D printed electronics. The general design of the Slic3r for electronics is described in our previous work [20].

To place a nut in the object's model, it is selected from a library of predefined standard sizes and added to the list of embedded objects. The position is set via drag and drop in a rendered preview of the resulting toolpath. The exact position, rotation, and orientation can then be refined in a properties grid. Fig. 2 shows a screenshot of the user interface, illustrating the workflow to integrate nuts into an existing model. The different types of embedded nuts are inherited from a general `EmbeddedPart` and contain a parametric mesh model of the body of each nut type.

The slicing process itself consists of a series of steps. A 3-D object is divided into a set of horizontal 2-D layers, each represented by a polygon that contains the surface of one layer. The nuts' cavities are generated by computing the difference between each layer outline and the convex hulls of all nuts intersecting with this layer. The cavities are slightly inflated by applying a margin offset to the convex hull polygon, providing tolerance against thermal shrinking and imprecise placing movements. Extrusion trajectories are then computed by first following the offset polygon contour several times to generate perimeters. The remaining area is filled with a regular infill pattern. Finally, all trajectories are ordered to minimize travel moves and exported line by line into a G-code file.

For nuts in an upright position or arbitrary rotation, as illustrated in Fig. 3, the cavities must match the surface of the bottom side, but leave enough space for insertion. For a given layer, this is achieved by generating a convex hull polygon from the rotated component geometry for this layer and computing the union of

Fig. 3. Cavities generated by our slicing software for horizontal alignment of embedded hexagonal nuts. The volume cross-hatched in red must remain free of plastic during printing to afford vertical insertion of the nut. Note that the volume could be filled after the nut has been placed, but that is not yet supported by our software.

```
M361 P1
...
;<object name="magnet_connector.stl">
;<part id="1" name="M3 hexnut">
;  <type identifier="hexnut" thread_size="3"/>
;  <position box="1"/>
;  <size height="2.4"/>
;  <shape>
;    <point x="2.60241" y="-1.5025"/>
;    <point x="2.60241" y="1.5025"/>
;    <point x="0" y="3.005"/>
;    <point x="-2.60241" y="1.5025"/>
;    <point x="-2.60241" y="-1.5025"/>
;    <point x="0" y="-3.005"/>
;  </shape>
;  <destination x="94.72" y="88.84" z="8.05"/>
;  <orientation orientation="Flat"/>
;  <rotation z="0"/>
;</part>
;</object>
```

Fig. 4. XML G-code extension for in-line description of nuts in a common G-code file. The M361 command executes the placing operation for the object during the printing process. The embedded XML-lines start with a ";" marking them as comments in regular G-code syntax to avoid interference with the printing process.

all component hull polygons from the lower layers. The upper half of the cavity is not entirely filled with material in such cases.

Shape and position information of all embedded nuts are eventually exported with XML-formatting at the end of the resulting G-code file. The example in Fig. 4 illustrates a single, lying hex-nut. A custom G-code command triggers the placing operation: `M361 Px`, where `x` is the part id; see Section IV-B for further details. Placing commands are inserted into the G-code at the end of each layer if required.

IV. Object Placement

A. Experiment Setup

A standard Prusa I3 open-source printer was modified to facilitate automatic handling of metal components during the print process. The overall setup is illustrated in Fig. 5.

To place the objects, an electromagnetic gripper was attached to the back of the x-carriage. The magnet is mounted on a hollow-shaft NEMA 8 stepper motor, as illustrated in Fig. 5. The wires are routed through the motor shaft and decoupled by a slip ring, allowing for unlimited gripper rotation. The motor is connected to a spare driver on the printer's mainboard and configured as a second extruder. The amount of extrusion is directly mapped to the angle of rotation; the G-code command `G1 E90 F1000` causes a rotation of 90°. The magnet is driven by a MOSFET and controlled by an extension pin via G-code.

During the print process, nuts are supplied by a tray with slots, matching the shapes of different types of nuts. Each slot is sized to exactly fit one nut shape, so all nuts are in a well-defined position. Currently supported are hex- and square-nuts in flat and upright positions. The tray is generated by a parametric script and can be adjusted for individual configurations. It is attached to the printbed at the same height, providing a collision-free surface.

The Prusa I3 gantry system's dimensions are designed to cover the entire surface of the printbed with the extruder nozzle. Both magnetic gripper and tray require additional space along the y-axis. Therefore, the y-axis was extended using longer rods from 35 to 60 cm to preserve the full original print volume. The original 8-mm linear bearings and rods were replaced by 12-mm diameter parts to maintain sufficient stability.

Components must be placed with high precision to prevent tilting and wedging during insertion into the cavity. This requires good calibration of the offset between the plastic extruder, the magnetic gripper, and the tray's position in relation to both tools. To assist these calibration steps, an upwards facing Raspberry Pi Camera V2.1 was integrated into the tray. It is connected directly to the Raspberry Pi computer, which controls the printer. The camera has a resolution of 3280 × 2464 pixels and a focal ratio of 2.0. The focus was manually set to a very close distance, only a few centimeters above the camera, making it possible to take high-resolution close-up pictures of the tools with a field of view of 20 × 27 mm. Since the camera is embedded into the tray, the tray position with respect to the extruder can be measured implicitly.

B. Control Software

OctoPrint [25] was used as a print server to control the execution of G-code files on the printer, as it offers a very extensive and well-supported plugin mechanism. We developed *OctoPNP* as a plugin to execute placing operations during a print and support the camera-based calibration process. Since tool configuration, the position of tray and camera as well as all calibration aspects are covered by OctoPNP, the G-code itself is very generic and can be executed on different types of printer hardware.

Upon opening a G-code file in OctoPrint, OctoPNP searches for the XML-encoded information about embedded nuts, as depicted in Fig. 4. All components are internally assigned to free slots on the tray. The tray is then rendered graphically for the operator, who manually inserts all requested components into their slots before the print is started. Fig. 5 illustrates how the tray is integrated into the printer. Different trays are used depending on the respective count and type of nuts. For applications that exceed the tray capacity of our prototype, fixed component feeders could, of course, be used.

Placing a particular component during the print is triggered by the `M361 P<part_id>` commands in the G-code, which were inserted by the slicing software described in Section III.

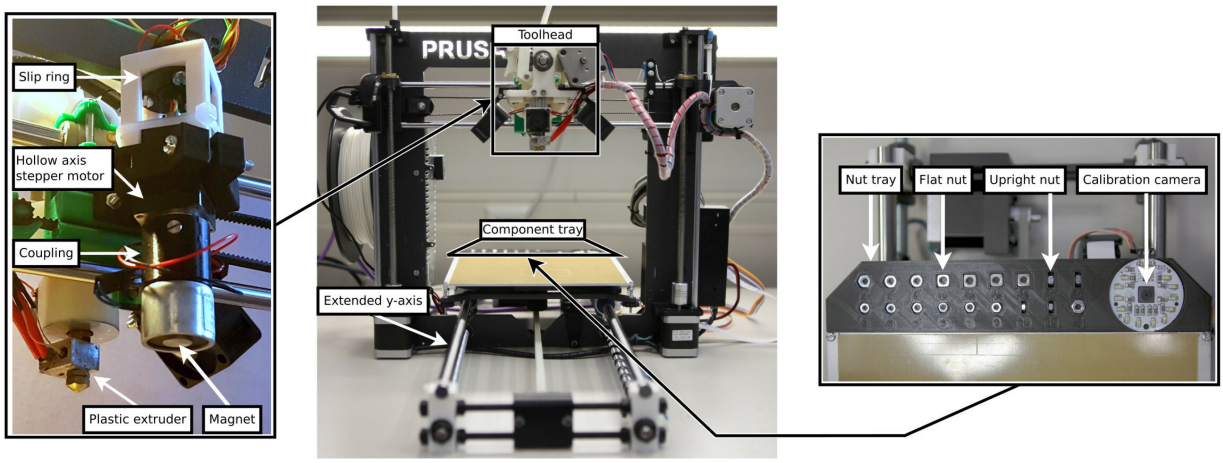

Fig. 5. Modified Prusa i3 FFF printer used for the experiments. Left: Back view with magnetic pick and place tool. Middle: Front view of the printer with the plastic extruder and the extended *y*-axis. Right: The component tray with the integrated calibration camera. Note that the stepper motor holder and the coupling both use embedded nuts that can be printed on the machine itself.

```
1  T1                  ; select tool 1 (magnet)
2  G1 X111.5 Y213.2    ; Move to nut position on tray
3  G1 Z0               ; lower gripper to tray level
4  M400                ; synchronize command queue
5  M42 P48 S255        ; enable solenoid
6  G1 Z13.9            ; lift gripper and nut
7  G92 E0              ; reset rotation
8  G1 E30              ; rotate component by 30°
9  G1 X97.293 Y95.276  ; move magnet to placement position
10 G1 Z3.9             ; insert component at correct height
11 M400                ; synchronize command queue
12 M42 P48 S0          ; disable solenoid
13 G1 Z13.9            ; lift gripper
14 T0                  ; select tool 0 (extruder)
```

Fig. 6. G-code generated by OctoPNP during a print job to place a single nut.

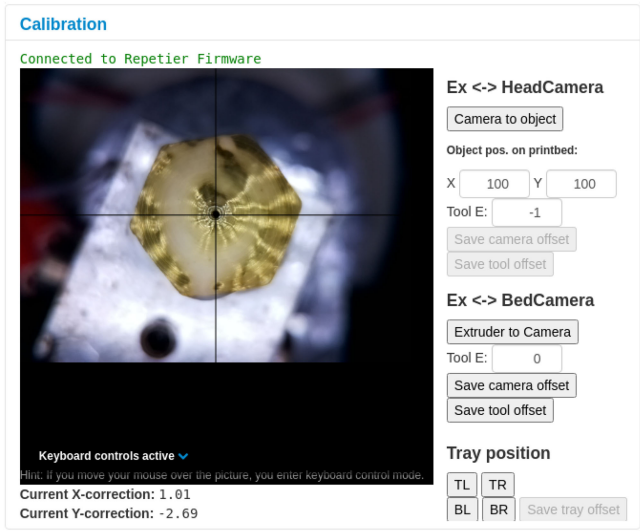

Fig. 7. Camera-based calibration tool, integrated into our OctoPNP plugin. Each tool (extruder and magnetic gripper) is roughly positioned at the camera and is then manually fine-adjusted using the cross-hairs. The resulting offset is directly stored in the printer's firmware. Note that the process can also be used for cameras facing either upwards or downwards, to calibrate the camera position itself, the tool offsets, and the precise tray position.

The plugin mechanism in OctoPrint provides a callback interface, implementing a hook scheme to replace specific G-code commands before they are sent to the printer. Fig. 6 illustrates how each `M361` is replaced by a series of commands with parameters for the position of the respective tray slot and with corrected offsets derived from the calibration. Note how the `M400` commands in lines 4 and 11 synchronize the execution queue on the printer before the magnet is actuated. All popular firmwares use a communication queue to buffer several G-code commands to avoid delays and precompute jerk values for the next movement. However, nonmovement commands typically bypass the execution queue, e.g., to get immediate responses to temperature requests. Therefore, the `M400` command causes the queue to run empty, ensuring that the printer has arrived at the correct position before the solenoid picks up or releases a component.

C. Calibration

Since the component placing is currently carried out in an open-loop control scheme, it requires thorough calibration. The offset between plastic extruder (`T0`), magnetic gripper (`T1`), and the exact tray position with respect to the magnet needs to be known. Both, a wrongly calibrated magnet gripper or tray lead to a wrong component position in the final print. Note that it is not possible to distinguish between both of these error sources after placing a nut, as the effects accumulate. Hence, easy calibration is a crucial requirement. It is always necessary after mounting a component tray with a matching set of slots for the needs of the current design, but also to compensate tolerances on cheap hardware platforms with low mechanical stability and endstop precision. Therefore, we integrated a camera-based interactive calibration tool into OctoPNP. As illustrated in Fig. 7, the camera video stream is fetched directly from the camera via HTTP-request in our web-based user interface. The live video is rendered into a JavaScript canvas and overlaid with crosshairs.

The tool (extruder or magnet) is automatically positioned above the camera and can be manually fine-adjusted using the

arrow keys of a keyboard. The offset is then calculated from the difference of current and adjusted position, and can be stored either as OctoPNP parameters or at the firmware level, which is generally preferred as it provides better integration. Unfortunately, while all modern printer firmwares support tool offsets, no standardized way exists to update the values. Therefore, we implemented automatic firmware detection in OctoPNP with individual update protocols for different firmware types. Currently, Repetier, RepRapFirmware, and Virtual Marlin (for testing purposes) are supported. Each step of the calibration is triggered manually by the user and is performed before printing. A full calibration usually requires following three steps:

1) Since the tip of the `T0` (extruder) is the already known, the camera position can be calculated by aligning `T0` with the optical axis of the camera.
2) With the known camera position, the offset between `T0` and `T1` (magnetic gripper) can be measured by moving `T1` over the camera.
3) A flat nut is picked up from the tray with the magnetic gripper and positioned over the camera. Since the offset between the camera and the nut is known, the deviation of the expected nut center and the visible nut center is used to calibrate the exact position of the tray.

V. Evaluation

We printed multiple objects to validate our approach, demonstrating both the augmented slicing software and the automatic placement. All examples shown in this article are objects actually used in our group, which were previously mounted with nuts inserted after printing.

The basic top–down insertion and complete embedding of a single nut into a small object is demonstrated by the robot foot studs shown in Fig. 8(a). These studs are mounted on the feet of our University's RoboCup soccer robot [26], and they need to be exchangeable quickly (different sizes, soft or hard material) to adapt to different playing grounds. The previous design used slide-in nut slots, but nuts of spare studs were prone to fall out during transport, and dirt could catch in the nut slots. Both of these problems have been fixed with the new design.

The second object is the connector between the magnet and the stepper motor for the pick and place tool in this work [see Fig. 8(b)]. Due to the connector's small diameter, the material between the shell and the standing nut is very thin (only two lines of filament). Fully embedding the nut removes the need for a slide-in slot, which simplifies assembly and increases the object's strength.

The third example design is a modified dock for the E3D ToolChanger printer [see Fig. 8(c)]. This machine supports up to four different extruders or tools, only one of them is active at any time. The slide-on docks are used to park unused printing tools on the rear side of the printer. Our version has a wider distance between the pins than the original design to attach modified tools, with the nuts inserted upright since the alignment pins are screwed in from the side of the print. Note that the pin positions and angle needs to be very precise to align with the tool and

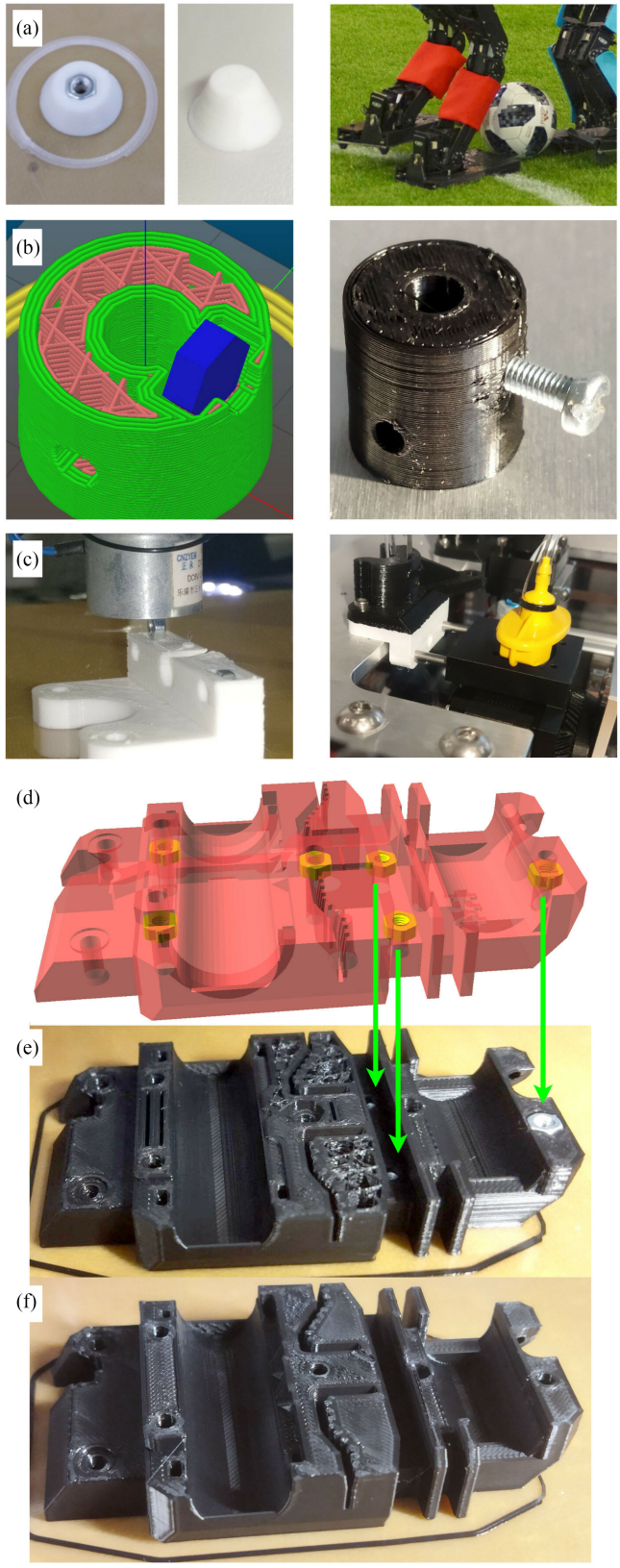

Fig. 8. Example multimaterial objects with embedded nuts. (a) Robot foot studs with embedded nuts. (b) Magnet connector with upright nut. (c) Tool dock for E3D ToolChanger. (d) X-carriage model from Prusa i3 with embedded nuts. (e) Right after nut placement. (f) Finished x-carriage with embedded nuts.

TABLE I
PLACEMENT EXPERIMENT FOR MULTIPLE (HEX-) NUTS WITH (x, y) POSITION OFFSETS

Clearance [mm]	X [mm]	Y [mm]	Placement M3 success	Placement M4 success
0.4	0	0	16 / 16	16 / 16
0.4	-0.2	0	15 / 16	9 / 12
0.4	+0.2	0	16 / 16	12 / 12
0.4	0	-0.2	16 / 16	12 / 12
0.4	0	+0.2	13 / 16	12 / 12

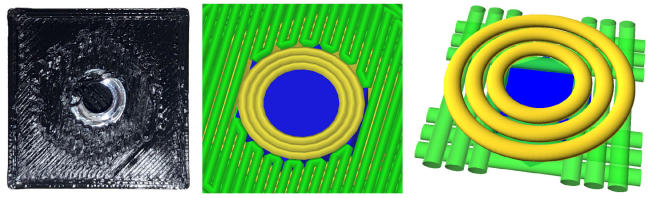

Fig. 9. Perimeter is not printed correctly and sticks to the side of the infill due to low adhesion of extrusions at the first layer covering a metal component (left), expected perimeter layout (middle), and proposed *crosshatch-bridging* strategy to ensure extrusions are connected to the plastic surface at both ends of each line (right). (Yellow = perimeter, green = infill, and blue = nut).

fit without jamming. The alignment with inserted nuts is better than with directly screwed pins, and the dock works as intended.

As a typical example of a more complex 3-D printed functional object, we present the x-carriage from the Prusa I3 printer, which is the machine we used in this work. The object demonstrates that the nut placement also works on redesigned existing CAD models with multiple nuts placed on different levels. Fig. 8(d) shows a 3-D rendering of the nuts inside the carriage, a snapshot of the carriage right after inserting a nut [see Fig. 8(e)], and the finished part with the nuts hidden in it [see Fig. 8(f)]. Compared to the original design, the modified part with embedded nuts is easier to assemble since the nuts cannot fall out while putting everything together.

As each pick and place operation only requires a few seconds, the total production times for our example objects were completely dominated by the FFF plastic printing (several minutes to a few hours).

To evaluate the reliability and precision of our setup, we conducted a study, where a total of 160 nuts were inserted into printed cavities. A simple cube with four cavities was sliced for two nut sizes, printed, and the placing operation was executed four times for each such cube, resulting in 16 individual placing operations for each parameter combination. Due to a typo in the G-code with additional displacement, one M4 nut was not placed in each execution, resulting in only 12 repetitions. The nut types tested are M3 and M4 hex shape, with nominal across flat sizes of 5.5 and 7.0 mm, actual measured values are 5.40 and 6.86 mm. The cavities were widened in the slicing software by a clearance of 0.4 mm, the actual printed cavities came out with 5.49 and 6.96 mm due to significant shrinking and backlash effects, leaving virtually no remaining tolerances around the inserted nuts. Each nut type was placed exactly at the position given by the G-code (row 1) and intentionally placed with small offsets of 0.2 mm to its actual destination position, to test the robustness against minor inaccuracies (rows 2–5).

The outcome of this experiment is listed in Table I, showing that automatic placing of nuts is reliable. Even with artificial perturbations, the success rate remains high, indicating that the process is robust against small deviations induced by cheap hardware or poor calibration. It seems that the chamfered nuts still slide into their slots when placed with a slight offset.

Although our software is still missing some comfort features, such as automatically aligning nuts with existing holes, the modified Slic3r works as intended. The pick and place process is reliable but requires some initial fine-tuning of tolerances to get the right-sized cavities (depending on nozzle, printer settings, and filament). As the insertion force generated by the magnetic gripper is very small, a press-fit (as sometimes used with manually placed nuts) is not possible, and nuts do not fit if cavities are too small. If the cavities are too big, small nuts (especially M2 and M2.5) may be placed off-center or start corotating with the screw in the hole as soon as torque is applied. This problem can largely be avoided by using square (instead of hexagonal) nuts.

In any case, the embedded nuts are more resistant to wear than direct screwing into the plastic. On the downside, a single misplaced nut often implies a failed print because either the printer loses steps on one axis, the printing nozzle crashes into the nut, or the nut is missing and cannot be inserted afterwards. Since the nuts are not removable once printed in, these prints also have limited options to recycle plastic and metal separately. Another general problem is caused by the low adherence of the plastic material used in common filaments (PLA, ABS, PETG) to metallic objects. In most cases, the nut is followed by a hollow cylindrical structure for the screw. Fig. 9 (left) illustrates how the first layer covering the nut is printed. The perimeter is laid entirely on metal and the infill extrusions end on the nut, producing unreliable and poor results. The effect is currently mitigated by printing multiple plastic layers on top of the embedded nuts. Fig. 9 (right) also illustrates our proposed solution, where two interface layers serve as bridges, connected to the previous plastic layer at both ends of each extrusion line. This requires support from the slicer and cannot be modeled in the CAD software. Another possible approach would be to change the printing order so that the perimeter is printed after the surrounding infill. But both of these solutions need further inspection. It should be noted that this is a general problem that also occurs when printing slots to slide in nuts or other structures where hollow cylinders are not properly supported.

VI. CONCLUSION

We presented an integrated approach to embed metal nuts directly into 3-D printed plastic objects. This direct embedding results in objects both more robust and easier to assemble when compared to current practice, where plastic threads are likely to wear out, slide-in nut channels usually weaken the object structure and press-in nuts require significant and risky manual

assembly effort. Our hardware prototype is based on a modified Prusa I3 FFF printer, equipped with an extra rotating magnetic gripper for object pick and place, a customizable tray for nuts and similar components, and an integrated camera for quick and reliable tool calibration. Several successfully printed prototype objects demonstrated the key aspects of the proposed method.

Our software is fully based on popular open-source tools, combining a modified version of Slic3r for interactive placement of nuts and metal parts with the OctoPNP plugin for component tray management and machine calibration. It should be noted that the software can also be used together with standard FFF printers, generating cavities for the components as designed, pausing the print process, and alerting the user for manual insertion of nuts.

Planned future work concerns both hardware limitations and software features. Based on the three-axis Cartesian configuration of common FFF printers, our rotating pick and place tool covers the most common use cases, but it can still only insert nuts from above. We are working on tilting tools and corresponding nonplanar slicing software to remove this restriction, which would also remove the need for weakening cavities around vertically mounted hex nuts (compare Fig. 3). Another way to eliminate these empty volumes is to fill them with plastic after placing the nut. This should hold the hex nuts tighter and could improve stability, especially for delicate geometries. As already mentioned, oozing artifacts from the plastic extruder can obstruct cavities, which sometimes results in failed prints. This can be solved by cavity-aware slicing algorithms, preventing unretractions and travel moves touching the cavities.

It should be obvious that other magnetic components with smooth surfaces (e.g., washers) could also be handled by our approach. For objects prone to excentric pickup (e.g., ball-bearings) suitable conical mechanical alignment structures clipped onto the gripper can help to center the parts.

Finally, integration with at least one common CAD software for one-click import and re-export of full "assemblies," including designed screws and nuts, would be an important step on the path to a fully integrated toolchain.

More information, including the software source code, is available on the project website: https://tams.informatik.uni-hamburg.de/research/3d-printing

References

[1] J. Prusa, "Prusa iteration 3." Accessed: Dec. 2020. [Online]. Available: https://github.com/josefprusa/Prusa3

[2] R. Jones *et al.*, "RepRap–The replicating rapid prototyper," *Robotica*, vol. 29, no. 1, pp. 177–191, 2011.

[3] A. Bowyer *et al.*, "RepRap project," 2006. Accessed: Dec. 2020. [Online]. Available: https://reprap.org/wiki/RepRap

[4] J. V. Neumann *et al.*, "Theory of self-reproducing automata," Univ. Illinois Press, Champaign, IL, USA, Tech. Rep. TR-08226-11-T, 1966.

[5] M. P. Groover, *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems*, *7th ed*. Hoboken, NJ, USA: Wiley, 2019.

[6] S. Hermann, "Helicoils, Threaded insets and embedded nuts in 3D prints—Strength & strength assessment." Accessed: Dec. 2020. [Online]. Available: https://www.cnckitchen.com/blog/helicoils-threaded-insets-and-embedded-nuts-in-3d-prints-strength-amp-strength-assessment

[7] *Embedding Nuts in 3D Printed Parts for Hidden Fastener Strength*. Accessed: Dec. 2020. [Online]. Available: https://markforged.com/resources/blog/embedding-nuts-3d-printing

[8] G. J. Monkman, S. Hesse, R. Steinmann, and H. Schunk, *Robot Grippers*. Hoboken, NJ, USA: Wiley, 2007.

[9] H.-H. Chen, "Surface mounting device pick-and-place head," *US Patent* 4 860 438, Aug. 29, 1989.

[10] J. P. C. Charlebois, "Electromagnetic and vacuum lifter," *US Patent* 1 181 112, May 2, 1916.

[11] A. Kataria and D. W. Rosen, "Building around inserts: Methods for fabricating complex devices in stereolithography," *Rapid Prototyping J.*, vol. 7, pp. 253–261, 2001.

[12] J. Glasschroeder, E. Prager, and M. F. Zaeh, "Powder-bed based 3D-printing of function integrated parts," in *Proc. Int. Solid Freeform Fabr. Symp.*, 2014, pp. 775–792.

[13] J. Hoerber, J. Glasschroeder, M. Pfeffer, J. Schilp, M. Zaeh, and J. Franke, "Approaches for additive manufacturing of 3D electronic applications," *Procedia CIRP*, vol. 17, pp. 806–811, 2014.

[14] A. H. Espera, J. R. C. Dizon, Q. Chen, and R. C. Advincula, "3D-printing and advanced manufacturing for electronics," *Prog. Additive Manuf.*, vol. 4, pp. 245–267, 2019.

[15] D. Espalin, D. W. Muse, E. MacDonald, and R. B. Wicker, "3D printing multifunctionality: Structures with electronics," *Int. J. Adv. Manuf. Technol.*, vol. 72, pp. 963–978, 2014.

[16] G. L. Goh *et al.*, "Additively manufactured multi-material free-form structure with printed electronics," *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 1–4, pp. 1309–1316, 2018.

[17] C. Bailey *et al.*, "Augmenting computer-aided design software with multifunctional capabilities to automate multi-process additive manufacturing," *IEEE Access*, vol. 6, pp. 1985–1994, 2017.

[18] G. T. Carranza, U. Robles, C. L. Valle, J. J. Gutierrez, and R. C. Rumpf, "Design and hybrid additive manufacturing of 3-D/volumetric electrical circuits," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, no. 6, pp. 1176–1183, Jun. 2019.

[19] F. Wasserfall, "Embedding of SMD populated circuits into FDM printed objects," in *Proc. 26th Int. Solid Freeform Fabr. Symp.*, 2015, pp. 180–189.

[20] F. Wasserfall, D. Ahlers, N. Hendrich, and J. Zhang, "3D-printable electronics– Integration of SMD placement and wiring into the slicing process for FDM fabrication," in *Proc. 27th Int. Solid Freeform Fabr. Symp.*, 2016, pp. 1826–1837.

[21] F. Wasserfall, N. Hendrich, D. Ahlers, and J. Zhang, "Topology-aware routing of 3D-printed circuits," *Additive Manuf.*, vol. 36, 2020, Art. no. 101523.

[22] F. Wasserfall, D. Ahlers, and N. Hendrich, "Optical in-situ verification of 3D-printed electronic circuits," in *Proc. IEEE 15th Int. Conf. Automat. Sci. Eng.*, 2019, pp. 1302–1307.

[23] A. N. Büngener, "Integration von mechanischen Objekten in das 3D-Druckverfahren," Bachelor thesis, Univ. Hamburg, Hamburg, Germany, 2019. [Online]. Available: https://tams.informatik.uni-hamburg.de/publications/2019/BSc_Arne_Buengener.pdf

[24] A. Ranellucci, "Slic3r website." Accessed: Dec. 2020. [Online]. Available: https://slic3r.org

[25] G. Häußge, *Octoprint project website*. Accessed: Dec. 2020. [Online]. Available: https://octoprint.org

[26] M. Bestman *et al.*, *Hamburg Bit-Bots and WF Wolves Team Description for RoboCup* 2018. Accessed: Dec. 2020. [Online]. Available: https://dx.doi.org/10.13140/RG.2.2.27864.85763

Daniel Ahlers received the B.Sc. and M.Sc. degrees in computer science from the University of Hamburg, Hamburg, Germany, in 2016 and 2018, respectively. He is currently working toward the Ph.D. degree in computer science focusing on verifying 3-D printed electronics in the University of Hamburg.

He is currently a Research Associate with the Department of Informatics, University of Hamburg. His research interests include slicing algorithms for printed electronics and nonplanar printing, multimaterial printing, and image processing for 3-D printing.

Florens Wasserfall received the B.Sc. degree in computer science (with a minor in oceanography) in 2009, started working with 3-D printers from the University of Hamburg, in 2011, the M.Sc. degree in adaptive slicing algorithms for FFF printing from the University of Hamburg, in 2014, and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 2020, presenting design software for the integration of electronics and sensors into 3-D printed parts.

He is currently a Postdoc Research Associate with the Department of Informatics, University of Hamburg. He teaches courses in computer architecture and applied robotics. His research interest is focused on slicing and design algorithms for hybrid additive manufacturing and their application in the field of robotics.

Norman Hendrich received the B.Sc., M.Sc. degrees in physics, and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 1986, 1991, and 1996, respectively.

He is currently a Senior Lecturer with the Department of Informatics, University of Hamburg. He participated as a Principal Investigator in several collaborative European research projects and currently acts as a Project Manager of the joint Sino-German project Transregio-SFB TRR169 "Crossmodal Learning." His research interests include computer simulation and machine learning, with a focus on applications in service robotics and dexterous manipulation.

Arne Niklas Büngener received the B.Sc. degree in computer science, focusing on integrating mechanical objects into the 3-D printing process, in 2020, from the University of Hamburg, Hamburg, Germany, where he is currently a student with the Department of Informatics.

Jan-Tarek Butt received the B.Sc. degree in computer science from the University of Applied Sciences Emden/Leer, with an emphasis on computer engineering, in 2019. He is currently working toward the master's degree in computer science with the Department of Informatics, University of Hamburg, Hamburg, Germany.

He is currently a Research Assistant with the Department of Informatics, University of Hamburg. He joined Google's "summer of code" scholarship program in 2016 as a student. Since 2018, he has been active as a mentor. His primary research interests include embedded systems and 3-D printing.

Jianwei Zhang (Member, IEEE) received the bachelor of engineering (with distinction) and the master of engineering degrees in computer science from the Department of Computer Science, Tsinghua University, Beijing, China, in 1986 and 1989, respectively, the Ph.D. degree in computer science from the Department of Computer Science, Institute of Real-Time Computer Systems and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany, in 1994, and Habilitation from the Faculty of Technology, University of Bielefeld, Bielefeld, Germany, in 2000.

He is currently a Professor and the Director of group TAMS, Department of Informatics, University of Hamburg, Hamburg, Germany. He has authored or coauthored about 400 journals, conference papers, technical reports, and four books in the areas of his research interests, which include sensor fusion, intelligent robotics, multimodal machine learning, etc.

Dr. Zhang was the recipient of the multiple Best Paper Awards. He is the Coordinator of the DFG/MSFC Transregional Collaborative Research Centre SFB/TRR169 "Crossmodal Learning" and several EU robotics projects. He is a life-long Academician with the Academy of Sciences in Hamburg and was a Member of the IEEE Robotics and Automation Society AdCom from 2013 to 2015. He is the General Chair of IEEE MFI 2012, IEEE/RSJ IROS 2015, and the International Symposium of Human-Centered Robotics and Systems 2018.