2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Utilizing a Malfunctioning 3D Printer by Modeling Its Dynamics with Machine Learning

Renzo Caballero[1,*], Piotr Piękos[1,*], Eric Feron[1], and Jürgen Schmidhuber[1,2]

***Abstract*— To create a self-repairing 3D printer, it must continue operating even after experiencing corruption. This work focuses on developing a method to effectively utilize a malfunctioning printer for reliable printing. This method can be applied by the printer itself for self-repair and enhance the reliability of commercial 3D printers. We achieve this by modeling the dynamics of the corrupted printer using a machine learning model that by observing one trajectory infers the corrupted printer dynamics to improve its accuracy. Our method is evaluated on a digital twin of the 3D printer, demonstrating its capability to enable the printer to operate reliably, even when encountering new corruptions not encountered during training. The scripts are public on `https://github.com/piotrpiekos/adaptive-printer`.**

I. Introduction

Recent milestones in space exploration, including the discovery of water molecules on the Moon [1], the landing of the Perseverance Rover, the first drone flight on Mars [2], the potential for 3D printing in zero gravity [3], and substantial global investments in contemporary space research, have made the concept of self-sufficient space colonies, whether manned or unmanned, more prevalent than ever before.

The human body is uniquely suited to the conditions found solely on Earth. There is no single piece of land in the solar system outside of Earth with conditions suitable for human existence, and creating safe artificial environments in space where humans can coexist is expensive. Additionally, the extended travel and communication times make any complications potentially fatal. For these reasons, and supported by the fact that unmanned devices like the Voyager probes [4] and various rovers [5]–[7] have operated in space for extended periods, the authors believe that a society of self-maintaining robots is a more suitable solution if the goal is to colonize extraterrestrial lands [8].

The concept of self-repair, self-improvement, or self-replication at the software level is now a standard practice [9], [10]. However, kinematic self-repair machines—a term borrowed from [11]—are physical objects in a physical space, and they pose practical challenges as power requirements, material resources, fabrication, and assembly when aiming for automation [11]–[13]. Despite the limited research on kinematic self-repair machines [14], the utilization of 3D printing to fabricate replacements or even repair spare parts is becoming increasingly popular [15]–[17]. We focus on understanding the capacities and limitations of self-repair in a 3D printer, as it is a robot that, by default, can fabricate different parts for repair and replication purposes. We believe that in a system with many 3D-printable robots interacting, if a multipurpose 3D printer is part of the system and can self-repair, the entire system can achieve self-repair. Moreover, in essence, there are no restrictions preventing a 3D printer from producing its own components [18]–[20]. Therefore, we believe that studying the self-repair capabilities and developing strategies to maintain the functioning of a 3D printer are essential for self-sustaining robotic colonies.

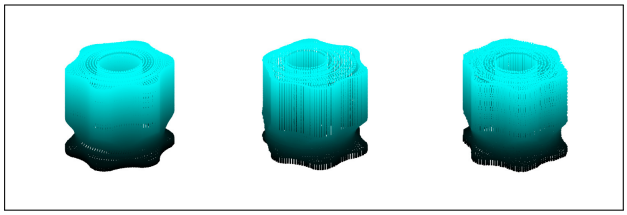

Fig. 1. Example of virtually printed bodies. We observe (left) an ideal printed body, (middle) the same 3D printed under corruption on the timing pulleys of the printer, and (right) the corrected body with our algorithm.

In our earlier study [21], we investigated a scenario in which a 3D printer, equipped with a faulty timing pulley, produces a series of increasingly superior timing pulleys until it attains the optimal one, effectively accomplishing self-repair. Additionally, we proved that, under some assumptions, the iterative method always converges to an ideal timing pulley no matter the initial damage on the timing pulley of the 3D printer [22]. Then, knowledge concerning the specific type of corruption on the pulley is not necessary to achieve self-repair.

A highly sophisticated machine, despite being more autonomous, can be more challenging to restore, while a too simple machine may have a limited capacity to adapt or detect various faults. We delve into the trade-off between complexity and self-repair in our previous work [23], where we modeled and analyzed a scenario involving a 3D printer that could not achieve self-repair without the addition of extra sensors—to provide feedback.

Feedback is the heart of control theory. The power of feedback is unquestionable since it gives devices a sense of their

[1]Renzo Caballero (`Renzo.CaballeroRosas@kaust.edu.sa`), Piotr Piękos (`Piotr.Piekos@kaust.edu.sa`), Prof. Eric Feron (`Eric.Feron@kaust.edu.sa`), and Prof. Jürgen Schmidhuber (`Juergen.Schmidhuber@kaust.edu.sa`) are with the Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) division, King Abdullah University of Science and Technology (KAUST), Thuwal, Kingdom of Saudi Arabia.
[2]Prof. Schmidhuber also holds the position of Scientific Director at the Dalle Molle Institute for Artificial Intelligence Research in Switzerland and serves as the Director of the Artificial Intelligence Initiative at KAUST.
*Ph.D. Students Renzo Caballero and Piotr Piękos contributed to this work equally.

own states and allows them to perceive their environment. In our previous work [23], we demonstrated that by assuming perfect information about the fault in the 3D printer, we could achieve self-repair, while the open-loop system was incapable of repairing itself. However, the addition of sensors complicates the self-repair and self-replication tasks. For instance, a camera cannot be 3D printed by a standard 3D printer.

In this work, by estimating the dynamics of the corrupted printer, we reliably correct the printing process without the need for real-time feedback or multiple additional sensors—see Fig. 1 for an illustration of the consequences of corruption in the printing process. This is achieved by printing a test object that provides the necessary information for self-calibration. The main advantage is that we do not require various sensors—which would increase the complexity of the printer—to account for all possible different corruptions; instead, we aim to modify the integrated controls with the extra help of only a single test object before operation.

We demonstrate the performance of our model on a dataset of real 3D objects [24] and assess its performance on different corruptions, which are modeled after our previous works [21], [25]. As a result, we print more precisely than the uncontrolled corrupted printer—our best method reduces the Root Mean Square Error (RMSE) of printed objects by 90% with respect to the uncontrolled printer.

II. Related Work

John von Neumann was one of the pioneering scientists to formally introduce a theory for self-replicating systems [26]. Among various approaches, he posed the fundamental question: *How can dependable systems be constructed from unreliable components?* This inquiry holds significance in our research as a corrupted printer can be likened to a system with unreliable components.

A fundamental challenge, aligned with the question introduced by von Neumann, is the lack of precision or the degradation of precision after the self-repair or self-replicating process. Essentially, if a system repairs itself or replicates many times, it is not trivial to expect that each offspring will maintain the original functionality. Richard Feynman addressed this issue in his lecture titled *There's Plenty of Room at the Bottom* [27]. He proposed a method of iterative miniaturization for machines, wherein each machine constructs a smaller version of itself in successive steps. Nevertheless, Feynman emphasized the importance of enhancing the accuracy of the equipment at each stage to sustain this progression.

The idea of a robot adapting to changes in their dynamics is not new. For instance, a PID (Proportional, Integrator, and Differentiator) controller, which was first introduced in 1911 [28] and formally modeled in 1922 [29], is often robust to small changes in the dynamics. In *Learning to Drive and Simulate Autonomous Mobile Robots* [30], robots autonomously tune their own controls, specifically the parameters of PID controllers to compensate for damages in the physical components.

Some authors observe that a damaged robot that cannot follow a desired trajectory directly learns different control settings in real time to continue operating as if it were not damaged [31]. Other authors present a self-modeling robot that utilizes actuation-sensation [32], a technique where the robot actively interacts with its environment to gather information about its dynamics and morphology.

A controller that prints an object closely resembling the desired one using a corrupted printer can be formulated using reinforcement learning (RL) [33]. The reward is typically given at the end of the printing process, often based on a measure that is challenging to compute before completing the print, as demonstrated in other works [34], [35]. However, from an engineering point of view, it is crucial for the controller to efficiently produce reasonable prints after just one object used to infer dynamics.

Closely related to model the controller is the notion of world models [36], as in the reinforcement learning terms we are modeling the dynamics of the environment. A more general notion than world models [37] allow to have subroutines and function calls inside the world model.

In this work, we focus on the x-axis and y-axis of the printer. We treat each layer of the 3D body as an independent print and model them as a sequence of points in 2-dimensional space. As a model class that learns the dynamics of the system, we investigate a machine learning algorithm LightGBM [38].

III. Digital Twin of 3D Printer

The digital twin (DT) in the experiment represents an Ender-5 3D printer [39]—see Fig. 2. We utilize the DT for both creating the training data and evaluating our models. A typical desktop 3D printer consists of a single stepper motor responsible for material extrusion and three additional stepper motors dedicated to controlling the positioning along the x, y, and z axes. These stepper motors operate at a rate of 200 steps per revolution. The printer's emphasis in this study is on the x and y axes, which employ a belt-pulley transmission system to achieve linear motion and accurately position the printing nozzle.

Since each motor has a total of 200 possible discrete positions per revolution, we model the system as a discrete-time system where the nozzle can only take discrete positions over the printing bed at each time-step. Ideally, if at the i-th time-step, $c_i = (\Delta x_i, \Delta y_i)$ represent a change in the angular position of the motors, and the dimensionless position of the nozzle is $s_i = (x_i, y_i) \in \{0, 1, \ldots, 1100\} \times \{0, 1, \ldots, 1100\} = \{0, 1, \ldots, 1100\}^2 = \mathcal{B}$, we model the transition to the $(i+1)$-th time-step as

$$(x_{i+1}, y_{i+1}) = (x_i + \Delta x_i, y_i + \Delta y_i). \tag{1}$$

Each experiment involves corrupting the pulley-belt system in both—x-axis and y-axis—directions, in a manner that renders (1) invalid. For different corruptions on the DT, we simulate the printing process of objects in the dataset described in Section V-A, and we store both the ideal output

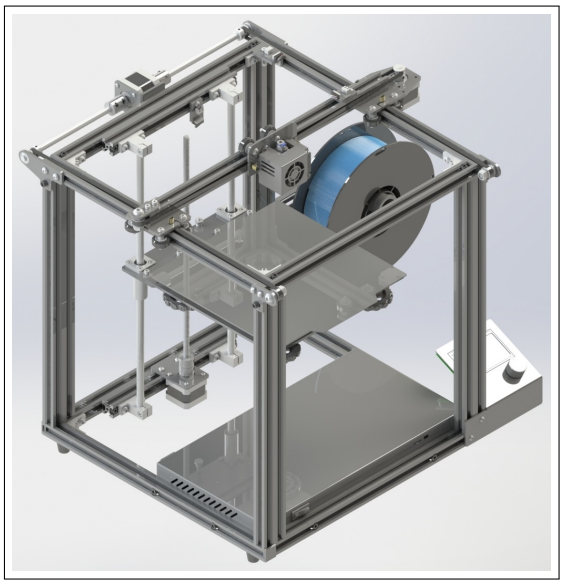

Fig. 2. Render of the simulated 3D printer utilized during the experiments. The render corresponds to an Ender-5 [40]. CAD model downloaded from [39].

that the DT would produce if it was undamaged and the actual corrupted output. Corrupted outputs will be trajectories of points from $[0, 1100] \times [0, 1100] = [0, 1100]^2 = \mathcal{B}_C \subset \mathbb{R}^2$.

The possible corruptions include (i) deformation on the timing pulleys and (ii) loose timing belts. The mathematical description of the corrupted dynamics of the DT for each corruption is detailed in Appendix X-A, and they are both based on our previous validated models [21], [25].

IV. Method

In this section, we offer a comprehensive description of the proposed method for modeling the corrupted dynamics of the DT.

A. Printing as an Optimization Problem

We view the printing process as the selection of a control sequence aimed at generating a trajectory as close as possible to the desired one. When the DT is ideal and (1) accurately models its dynamics, the optimization problem is straightforward since we have perfect information about the dynamics. Consequently, with an ideal DT, even in an open-loop setting, the trajectory of the nozzle is entirely predictable.

Formally, for each 2-dimensional print, there is an ideal trajectory $I = \{(x_{I_i}, y_{I_i})\}_{i=0}^{i=n-1} \subset \mathcal{B}$ for the nozzle of the DT, and an optimal trajectory $O = \{(x_i, y_i)\}_{i=0}^{i=n-1} \subset \mathcal{B}_C$ with $n \in \mathbb{N}$ being the number of points in both trajectories. Each point is connected to the next with a straight line, and together, they create a shape that ideally—when $I = O$—replicates the object we aim to print—see Fig. 3 for an example where the printer is corrupted.

During the printing process, we start from a known state $s_0 = (x_0, y_0) \in \mathcal{B}$ and attempt to print the trajectory. After the printing, the two trajectories are compared by some metric. We cannot access the true positions of the nozzle

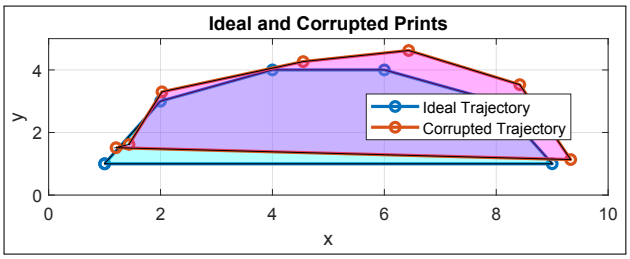

Fig. 3. Examples of trajectories with 7 points—ideally, the final point is equal to the starting point. We observe the ideal figure and the corrupted one.

s_i during the print. However, by (1), when the DT works ideally, the following assumption holds:

Assumption 1: For an ideal printer, we know the transition model exactly, $s_{i+1} = \mathcal{T}(s_i, c_i)$, where s_i and c_i represent the state and the selected control at the i-th time-step, respectively.

Corruptions in the printer break Assumption 1. With a corrupted printer, we have no information on the displacement in each—x-axis and y-axis—direction, so all non-initial states are unknown. This uncertainty about the current state of the nozzle results in the accumulation of errors that grow larger over the course of the printing procedure. We assume that after the printing procedure the printer is able to measure the coordinates of the actually printed points. This information will be used to learn the dynamics of the printer.

B. Similarity Metric for the Printed Objects

Measuring the similarity between printed and desired shapes in $\mathbb{R}^2$ is challenging. If we minimize their non-intersecting area through rotations and translations, we are disregarding potential significant point-wise errors. Similarly, if we minimize the maximum point-wise distance for some selection of points, the intersecting area is not taken into account—see Fig. 4 and [41].

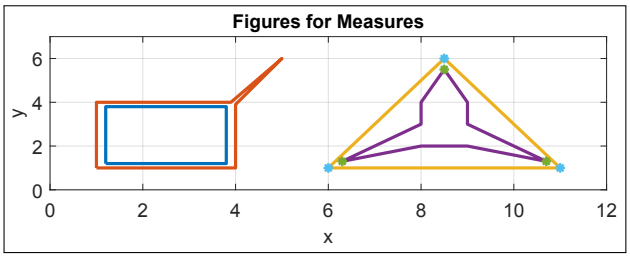

Fig. 4. On the left: two bodies with minimum non-intercepting area measure. On the right: two bodies with the minimum point-wise error when considering the three most proximate points.

Minimizing non-intercepting areas and point-wise metrics to create algorithms poses challenges, and they cannot be properly generalized to approximate all figures in $\mathbb{R}^2$, so we minimize the point-wise L^2 distance between trajectories.

C. Modeling the Printing Optimization Problem

In this subsection, we formally describe and motivate algorithms to find the optimal trajectories. For an unknown

corruption on the printer, a control that minimizes the L^2 distance between ideal and optimal trajectories in open-loop is a Partially Observable Markov Decision Process (POMDP) [42] . POMDP is a tuple $(S, A, \mathcal{T}, R, \Omega, P, D, \gamma)$, where $S = (X, \omega, C, O_t)$, is the state space - it represents current state of the printer, but it is not visible to the algorithm. It is composed of 4 parts: $(x_1, x_2) \in X$ is the current nozzle position, $(\omega_1, \omega_2) \in \omega$ is the angular position of the pulleys, $c \in C$ is the corruption of the printer and O_t is the optimal trajectory (description of the object we want to print).

$A = \mathbb{Z}^2$ is an action space, recall that each stepper motor can advance a discrete number of steps per time-step. $\mathcal{T}$ is a transition model, which in case of properly working, not corrupted printer is equal to (1). The transition model $\mathcal{T}$ depends on the corruption. $P : S \to \Omega$ is a function that, from the true state, produces a part that is visible for the model. R is the reward function, which here we treat as the RMSE of the optimal trajectory and printed trajectory. $\Omega = [0, 1100]^2$ is a set of observations - information that is available to the model from the environment - and in this case it is just the starting position of the nozzle —*but not the current position of the nozzle at every step* and the optimal points - description of the object to print. finally, D and γ are the distribution of initial states over S and the discount factor, respectively.

We assume that the model before the real printing can run one trajectory T to have information about current corruption in the printer. Therefore, relying on sparse rewards on precise metrics and training reinforcement learning such REINFORCE [43] or other classic algorithms [33] is hard from one trajectory. Instead we use the model of the system's dynamics.

Policy is an acting agent controls the printer given known information about the state. Finding the optimal policy with a full state information together with the transition model of corrupted printer would be easy. However, due to low information about the current state the problem is hard. Therefore we estimate remaining parts of the state and use them as if they are the true state to find the best next move.

We need to estimate the corruption c, angular position (ω_1, ω_2) and current nozzle position (x_1, x_2). We estimate angular position with s_o - sum of applied controls throughout the print and starting position. The model of dynamics $\mathcal{T}$ under corruption c is learned from the training trajectory and contains all necessary information about the corruption c. With the estimated transition model we will estimate the current state at every step.

More formally, We create our printing agent as a model-based algorithm [44]–[46]. We estimate the transition model $\mathcal{T}$ from our single observed training trajectory T [47]. Then, with the model, we greedily select the next state that is as close as possible to the next point in the optimal trajectory as imagined by the world model—see Algorithm 1. We choose the Euclidean norm as the L distance. We optimize greedily the point-by-point difference, which might lead to different optimal solutions than the formulation with sparse reward at the end of the episode. The benefit, however, is a significantly more efficient acting algorithm and, therefore, actually allows us to learn useful dynamic models with only 1 additional print.

Algorithm 1: Printing with a learned transition model

Data: $M_\theta : S \times C \to S$ - our estimated model of the transition model $\mathcal{T}$; O - optimal trajectory; L - distance on S space.
Result: $s \in S$ - printed trajectory
state $\leftarrow O_0$
$S \leftarrow$ [state]
for o_i *from* O *apart* O_0 **do**
  state $\leftarrow \text{argmin}_{c \in C} L(o_i, M_\theta(\text{state}, c))$
  Append $state$ to S
end

D. Learning a World Model - Problem formulation

To use Algorithm 1 effectively, we must learn some approximate model of transitions $M_\theta : S \times S \times C \to S$. We begin by formulating an optimization problem for a static environment and then extend it to a case involving in-context learning.

For a static environment with a printer corrupted always in the same way, learning a world model is equivalent to calculating a good function approximation. Specifically, let $\mathcal{M} = \{M_\theta : \theta \in \Theta\}$ be a family of functions parameterized by θ. For example, $\mathcal{M}$ can be a class of neural networks with a given architecture or set of all possible lightgbm estimators with given hyperparameters. Let $M_\theta(s, s_o, c)$ be the estimated value of the next state s' with the selected control c at state s, with s_o being the nozzle position if the printer were not corrupted (used to estimate angular pulley position ω). We also have a set of transitions $D = \{(s_i, s_o, c_i, s'_i)\}$. We find the transition model estimator by minimizing a loss function

$$\theta_{opt} = \text{argmin}_{\theta \in \theta} \sum_{(s, s_o, c, s') \in D} L(M_\theta(s, s_o, c), s'),$$

where we use the simple Mean Squared Error as L.

In this notion, we train a LightGBM regression model. Results show that even a simple model can drastically improve over a standard baseline defined in Subsection VI-A. We select the layer of the 3D body with the largest number of transitions as an example trajectory used to infer the corruption by the model before printing that body. The printer prints it with corrupted control, and the information about the printed positions are used to train the LightGBM model to learn the dynamics of the printer. Since the average thickness of layers is $0.2\,\text{mm}$, from an engineering point of view, we are almost not increasing the printing time.

The model learns to infer from this one example a pattern that is expected to be seen in the following data. A more general formulation, meta-learning, or *learning to learn* would involve changing the parameters based on the inputs, and therefore, patterns learned can be used not only for

the next prediction but also for future learning, as well as learning how to learn [48], [49]. This approach is more powerful but also more challenging to train. We leave its investigation for future work.

V. Experiments

We utilize our previously designed and validated model of a 3D printer [21], [25] to build a DT for the printer and simulate the required data, accounting for potential faults. Additionally, we conduct the training and testing with a standard available 3D bodies dataset [24] representing many possible printable objects.

A. Standard Triangle Language (STL) Dataset

We utilize the 10,000 STL files from [24] since they represent a concise summary of real-world models used for 3D printing. We call b_i to a i-th 3D body on the set of 10,000 3D bodies $\mathcal{D} = \{b_i\}_{i=1}^{10,000}$, i.e., $b_i \in \mathcal{D}$.

B. Construction of Controls

G-code files are commands that sequentially control the motors on the printer, position the nozzle in a two-dimensional plane, adjust the altitude of the printing bed, and regulate the extrusion of material. Different 3D printing software programs convert STL files into G-code files before printing.

In this work, and to utilize the models constructed in [21], [25], we decompose the G-code files corresponding to the dataset described in Subsection V-A into a sequence of controls for the stepper motors, instead of coordinates in space.

We focus on faults on the x-axis and y-axis stepper-motors. Then, for the i-th 3D body b_i in $\mathcal{B}$, we consider the $m_i \in \mathbb{N}$ 3-dimensional slices of b_i with the same hight parallel to the printing bed with such that, if s_k^i is the k-th slice of i-th body, then

$$\cup_{k=1}^{m_i} s_k^i = b_i, \text{ and } s_k^i \cap s_k^j = \emptyset \text{ if } i \neq j \tag{2}$$

with $i, j \in \{1, 2, \ldots, m_i\}$.

Finally, for the k-th slice s_k^i of the i-th 3D body b_i, we construct a sequence of controls $\{(u_{i,k,j}^x, u_{i,k,j}^y)\}_{j=1}^{n_{i,k}}$ with $n_{i,k} \in \mathbb{N}$ the needed number of controls for the nozzle to print the entire slice.

C. Experimental Setup

We create a dataset of 1000 bodies to evaluate our method. To do that we preprocess part of the original STL dataset to 3D-printer ready format. To create the gcode files, we utilize the by-default printing setup for ABS of the Ender-5 3D printer. As some bodies do not fit well into printer with our setting the translation to 3D format yields corrupted results, for example consisting of couple points, where usually body consists of more than 100 000 points.

To filter erroneous translation we choose to keep only those files that have more than 1000 points in the 3d printing format. We base this on the observation, that 1000 points is not enough points to even draw a one dimensional circle of diameter size equal to 1cm would require more than 1000 points. On the other hand, we expect our method to perform better in general on larger object as it is based on statistical inference and larger objects, with longer layers give more data points for infering the printer corruption. Therefore, we maximize the number of real bodies included in the dataset, while having some erroneous bodies. They, however, decrease the performance of our algorithm, so presented results can be seen as a lower bound of the performance on real, 3d printable bodies.

D. Error Evaluation

We compare our models by average trajectory RMSE between the printed and optimal trajectory in each layer of the 3D body. Let P_i be a printed trajectory for i-th trajectory in the dataset, and O_i be the desired i-th trajectory from the dataset. Then, we define the Average Body RMSE as

$$ABRMSE := \frac{1}{|D|} \sum_{i=1}^{|D|} RMSE(P_i, O_i),$$

where RMSE is the standard Root Mean Squared Error. The choice of ABRMSE attempts to capture the total error in the printing process of the entire 3D body, and not a single layer.

VI. Results

A. Baselines

Apart from the performance of the model, we also evaluate two simple baselines for comparison. Specifically, *Naive* printer assumes that the printer is not corrupted at all and prints in such a way. This is the result we would have in a standard printing method of a corrupted printer. We also calculate *Oracle* model that has access to the true transition model of the corrupted printer. This baseline is used as an upper bound on the performance (lower bound on the loss) that we can possibly achieve.

B. Analysed Models

We analyse two models used as a world model in our method. Both are based on the LightGBM algorithm. In main results (LightGBM row in the Table I) we use information about both dimensions for the prediction of both dimensions. Therefore, we do not utilize the fact that our corruptions are dimension-independent, meaning that the corruption in one dimension depends only on this dimension. We do that to demonstrate that our method is applicable to any corruption, not only those that corrupt dimensions independently.

The second method, which we call Independent Light-GBM (Ind. LightGBM) we utilize the assumption of independence in our generated corruption dataset and treat each dimension independently, meaning that the predictions of it are based only on this dimension. This is significantly easier version because one of the main bottlenecks is the size of the "training" data, which in our case is just one layer trajectory for each body. Assuming independence gives strong inductive bias and therefore reduce sample requirements. We do that to demonstrate the power of the algorithm if the corruptions can be limited to a certain subclass.

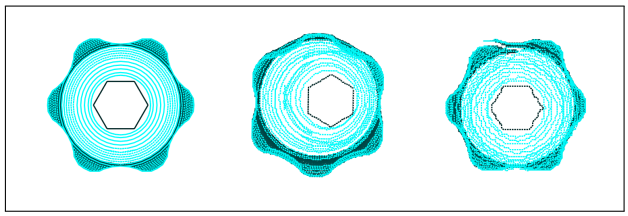

Fig. 5. Top view of (left) ideal printed body, (middle) the same 3D printed under corruption on the timing pulleys, and (right) the corrected body with our algorithm.

TABLE I

AVERAGE TRAJECTORY RMSE (ABRMSE) ON TEST DATASETS FOR DIFFERENT CLASSES OF TRANSITION FUNCTION ESTIMATORS.

Model	ABRMSE	In mm
Naive	27.6	5.6
Oracle	0.6	0.1
LightGBM	11.9	2.4
Ind. LightGBM (VI-B)	4.0	0.8

The main results are shown in Table I. the table shows ABRMSE and the equivalent error in millimeters. We observe that LightGBM reduces the error with respect to Naive significantly. Also, Independent LightGBM (Ind. LightGBM) with dimensions modeled independently is even much better, which is limited to certain type of corruptions, but demonstrates the capabilities of the model and presents promising direction for improving general results as a future work.

Fig. 6 shows the distribution of bodies as a function of their ABRMSE for the LightGBM algorithm. Fig. 5 shows a graphical representation of (left) an ideal print, (middle) a corrupted one, and (right) our correction. Since we are choosing a single training trajectory T, which informs about the corruption and helps in the printing process, and that trajectory is also a layer of the body itself, the more similar the layers of the 3D body are to each other, the more informative T will be. In other words, more cylindrical bodies benefit more from the correction.

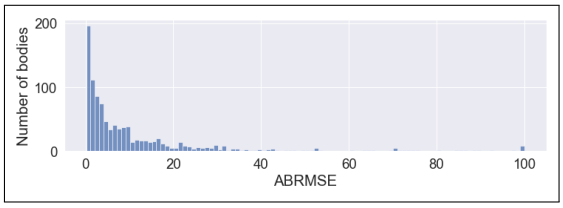

Fig. 6. Histogram of the number of bodies as a function of their ABRMSE for 1000 test bodies. ABRMSE was clipped to 100 for values greater than 100 for better visibility.

VII. CONCLUSION

This work demonstrates that modeling a state transition mechanism in a corrupted printer can significantly improve its accuracy, consequently enhancing the quality of the printed objects. It also shows that we can significantly increase the self-repair capabilities of a standard desktop 3D printer without the need for additional hardware, thus bounding the complexity of the printer while simultaneously increasing its autonomy. We conclude that artificial intelligence can alleviate the complexity-autonomy trade-off. The advantages of training a neural network with offline parameters and deploying it on low processing-power hardware are indisputable.

VIII. FUTURE WORK

This work is a first effort at improving the self-repair characteristics of a 3D printer without adding additional sensors. Many possible directions for future investigations arose during this work. A direct next step is validating the performance of the method on a real corrupted 3D printer.

Despite applying our algorithm to a 2-dimensional case, the extension to three dimensions is straightforward. The authors already worked on modeling corruptions on the z-axis [50], which is a direct next step to be implemented on the model.

Another promising direction of research is using more appropriate losses. For example, instead of using point-to-point using non-differentiable losses trained by a reinforcement learning algorithm.

Currently, the selected test trajectory T resembles the actual printout as much as possible. It would be worthwhile to investigate whether this is the optimal—most informative—choice. Even more ambitious is applying automated scientist and artificial curiosity [51], [52].

IX. ACKNOWLEDGMENT

This work was supported by the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI).

The authors would like to express their gratitude to Ph.D. students Arman Ibrayeva and Yasmine Marani for their time spent on discussions and valuable feedback.

X. APPENDIX

A. Mathematical Models for Corruptions

Non-ideal timing pulley: When the timing pulley on the stepper motor is not ideal, the expected relationship for the displacement Δx of the nozzle as the pulley experiences an angular displacement $\Delta\theta$ is given by

$$\Delta x = r\Delta\theta, \tag{3}$$

where r is equal to $6\,\mathrm{mm}$. A non-ideal pulley is defined as either a pulley with a non-circular convex hull or a timing pulley with a circular convex hull that has a radius different from the ideal value.

To model the dynamics of the printer with a non-ideal timing pulley C on the x-axis, we follow the model described in [21], where we essentially find the function $f(\cdot)$ such that

$$\Delta x = f(\theta_0, \Delta\theta; C), \tag{4}$$

which depends on the initial angular position θ_0 of the timing pulley as it starts rotating. The case for the y-axis is analogous.

REFERENCES

[1] C. Honniball, P. Lucey, S. Li, S. Shenoy, T. Orlando, C. Hibbitts, D. Hurley, and W. Farrell, "Molecular water detected on the sunlit moon by sofia," *Nature Astronomy*, vol. 5, no. 2, pp. 121–127, 2021.

[2] J. Balaram, M. Aung, and M. P. Golombek, "The ingenuity helicopter on the perseverance rover," *Space Science Reviews*, vol. 217, no. 4, p. 56, 2021.

[3] T. Prater, N. Werkheiser, F. Ledbetter, D. Timucin, K. Wheeler, and M. Snyder, "3d printing in zero g technology demonstration mission: complete experimental results and summary of related material modeling efforts," *The International Journal of Advanced Manufacturing Technology*, vol. 101, pp. 391–417, 2019.

[4] S. Borovikov and N. Pogorelov, "Voyager 1 near the heliopause," *The Astrophysical Journal Letters*, vol. 783, no. 1, p. L16, 2014.

[5] B. K. Muirhead, "Mars rovers, past and future," in *2004 IEEE aerospace conference proceedings (IEEE Cat. No. 04TH8720)*, vol. 1. IEEE, 2004.

[6] M. H. Carr, *The surface of Mars*. Cambridge University Press, 2007, vol. 6.

[7] K. A. Farley, K. H. Williford, K. M. Stack, R. Bhartia, A. Chen, M. de la Torre, K. Hand, Y. Goreva, C. D. Herd, R. Hueso *et al.*, "Mars 2020 mission overview," *Space Science Reviews*, vol. 216, pp. 1–41, 2020.

[8] V. Graziano, T. Glasmachers, T. Schaul, L. Pape, G. Cuccu, J. Leitner, and J. Schmidhuber, "Artificial curiosity for autonomous space exploration," *Acta Futura*, vol. 4, pp. 41–51, 2011.

[9] M. Mitchell *et al.*, "Computation in cellular automata: A selected review." *Non-standard computation*, pp. 95–140, 2005.

[10] J. Kari, "Theory of cellular automata: A survey," *Theoretical computer science*, vol. 334, no. 1-3, pp. 3–33, 2005.

[11] R. C. Merkle and R. A. Freitas Jr, *Kinematic self-replicating machines*. Landes Bioscience, 2004.

[12] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[13] S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, and S. Kokaji, "Self-repairing mechanical systems," *Autonomous Robots*, vol. 10, pp. 7–21, 2001.

[14] S. Brooks and R. Roy, "An overview of self-engineering systems," *Journal of Engineering Design*, vol. 32, no. 8, pp. 397–447, 2021.

[15] H. Kim, M. Cha, B. C. Kim, I. Lee, and D. Mun, "Maintenance framework for repairing partially damaged parts using 3d printing," *International Journal of Precision Engineering and Manufacturing*, vol. 20, pp. 1451–1464, 2019.

[16] L. Li, F. Yu, J. Shi, S. Shen, H. Teng, J. Yang, X. Wang, and Q. Jiang, "In situ repair of bone and cartilage defects using 3d scanning and 3d printing," *Scientific reports*, vol. 7, no. 1, p. 9416, 2017.

[17] A. van Oudheusden, J. Bolaños Arriola, J. Faludi, B. Flipsen, and R. Balkenende, "3d printing for repair: An approach for enhancing repair," *Sustainability*, vol. 15, no. 6, p. 5168, 2023.

[18] M. Moses, H. Yamaguchi, and G. S. Chirikjian, "Towards cyclic fabrication systems for modular robotics and rapid manufacturing." in *Robotics: Science and Systems*. Citeseer, 2009.

[19] M. S. Moses and G. S. Chirikjian, "Design of an electromagnetic actuator suitable for production by rapid prototyping," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 54808, 2011, pp. 491–497.

[20] A. Ellery, "Are self-replicating machines feasible?" *Journal of spacecraft and rockets*, vol. 53, no. 2, pp. 317–327, 2016.

[21] R. Caballero and E. Feron, "Experiments in robotic self-repair: A 3D printer repairs its own timing pulley," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3702–3709.

[22] Caballero, Renzo and Feron, Eric, "Study of Fixed-Points in the Self-Repair Process of a 3D Printer," *IEEE Control Systems Letters*, 2022.

[23] R. Caballero, A. Coronado, and E. Feron, "Role of Feedback in the Asymptotic Self-Repair Behavior of a 3D Printer," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 2925–2932.

[24] Q. Zhou and A. Jacobson, "Thingi10k: A dataset of 10,000 3d-printing models," *arXiv preprint arXiv:1605.04797*, 2016.

[25] R. Caballero, A. Coronado, and E. Feron, "Computational Modeling in System with Non-Circular Timing Pulleys," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9714–9720.

[26] J. Von Neumann, A. W. Burks *et al.*, "Theory of self-reproducing automata," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1966.

[27] R. P. Feynman, "Plenty of room at the bottom," in *APS annual meeting*. Little Brown Boston, MA, USA, 1959, pp. 1–7.

[28] R. P. Borase, D. Maghade, S. Sondkar, and S. Pawar, "A review of PID control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, pp. 818–827, 2021.

[29] N. Minorsky, "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, 1922.

[30] D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, *RoboCup 2004: Robot Soccer World Cup VIII*. Springer, 2005, vol. 3276.

[31] J. Schmidhuber, "An on-line algorithm for dynamic reinforcement learning and planning in reactive environments," in *1990 IJCNN international joint conference on neural networks*. IEEE, 1990, pp. 253–258.

[32] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[34] A. Ross, "Procrustes analysis," *Course report, Department of Computer Science and Engineering, University of South Carolina*, vol. 26, pp. 1–8, 2004.

[35] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," *Advances in neural information processing systems*, vol. 13, 2000.

[36] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[37] J. Schmidhuber, "On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models," *arXiv preprint arXiv:1511.09249*, 2015.

[38] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[39] "Creality Ender 5 Pro CAD Model," https://grabcad.com/library/creality-ender-5-pro-1, accessed: 2021-08-22.

[40] "Ender-5 3D Printer," https://www.creality.com/products/ender-5-pro-3d-printer, accessed: 2023-09-08.

[41] N. J. Mitra, L. J. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3d geometry," *ACM Transactions on Graphics (ToG)*, vol. 25, no. 3, pp. 560–568, 2006.

[42] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[43] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[44] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 1, pp. 7–20, 1987.

[45] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine learning proceedings 1990*. Elsevier, 1990, pp. 216–224.

[46] K. Young, A. Ramesh, L. Kirsch, and J. Schmidhuber, "The benefits of model-based generalization in reinforcement learning," *arXiv preprint arXiv:2211.02222*, 2022.

[47] H. Abdelraouf, F. Albalawi, and E. Feron, "On linear time invariant systems analysis via a single trajectory: A linear programming approach," in *2022 IEEE Conference on Control Technology and Applications (CCTA)*, 2022, pp. 989–994.

[48] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook," Ph.D. dissertation, Technische Universität München, 1987.

[49] L. Kirsch, S. van Steenkiste, and J. Schmidhuber, "Improving generalization in meta reinforcement learning using learned objectives," *arXiv preprint arXiv:1910.04098*, 2019.

[50] Caballero, Renzo and Feron, Eric, "Sensitivity Analysis in Self-Assembly Self-Repair System," *IFAC*, 2023.

[51] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. of the international*

conference on simulation of adaptive behavior: From animals to animats, 1991, pp. 222–227.

[52] J. Schmidhuber, "Curious model-building control systems," in *Proc. international joint conference on neural networks*, 1991, pp. 1458–1463.