Decay-Based Error Correction in Collective Robotic Construction

Jiahe Chen[1] and Kirstin Petersen[1]

***Abstract*— Multi-robot systems have been shown to build large-scale, user-specified structures using distributed, environmentally-mediated coordination in simulation. Little attention, however, has been devoted to error propagation and mitigation. In this paper, we introduce a detailed simulation of TERMES, a prototypical construction system, in which robots have realistic error profiles. We use this simulator and 32 randomly generated 250-brick blueprints to show that action errors can have significant long-term effects. We study the spatio-temporal error distribution and introduce and characterize the efficacy of a simple decay-based error correction mechanism. Although inefficient, this type of error correction is promising because it can be performed by robots with the same limited sensory capabilities as those who place bricks. To limit the impact on the construction rate, we also examine decay mechanisms informed by spatial and temporal error distributions. The incorporation of decay in our building process increases the probability of successful completion by ∼4, at the expense of ∼1/4 decrease in construction rate.**

I. INTRODUCTION

In collective robotic construction (CRC), many robots work together to build structures at a scale far larger than the size of the individuals [1]. Such systems have shown promising potential for applications ranging from autonomous construction of human habitats [2], [3], [4], to assembly of access structures in inaccessible terrains [5] and retaining walls in flood zones [6], [7]. Borrowing inspiration from construction in natural super-organisms, swarms of these robots may leverage the physical construction process to prompt distributed spatio-temporal coordination eliminating the need for global sensing and communication [8].

The demands on CRC hardware, however, are significant and consequently reliability is challenging [1]. Robots must be able to maneuver over the structure and manipulate, carry, and reason about where to place material. Either designers rely on flying robots which cannot operate below the structure surface envelope, or on robots which are able to climb on the structure while carrying material. Because the latter are effectively building structures that shield themselves from off-board sensors and central coordinators, many systems instead utilize more affordable and robust robots that have only local sensing and communication. Although robot swarms are typically showcased as robust to errors [9], collective construction robots actively manipulate a shared environment and when errors happen, these can physically obstruct further progress. Even heavily engineered robots will make mistakes, especially given the lengthy sequence of actions required to successfully complete large structures. The problem is exacerbated in systems that target particular blueprints and rely purely on additive construction as is common in current CRC [2], [10], [11], [12].

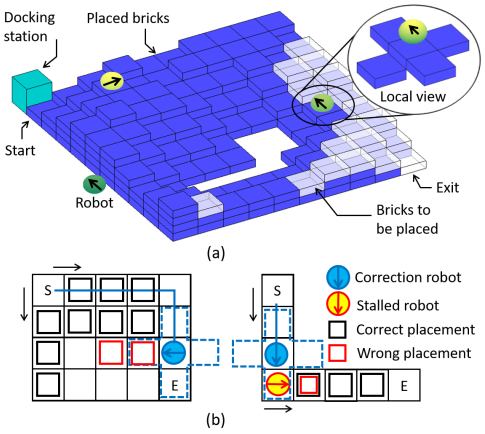

Fig. 1: (a) Sketch of the TERMES CRC system. (b) Example where knowledge beyond the local view (dashed line), is needed to identify a wrong placement or a stalled robot.

Few CRC research papers disclose error statistics [2], [13], [14], [4], although many describe feedback measures to instantaneously fix detected errors [15], [16], [14], [4]. Recent work also shows how to evaluate structure stability upon completion [17], and preventative techniques such as predictive local checks and planners that take error statistics into account [18]. However, we lack fundamental techniques that generalize across CRC system, to autonomously detect and correct errors accumulated in the structure. A typical colloquial solution is to introduce a separate class of "correction robots". However, such robots are subject to the same constraints as the assembly robots, may also make mistakes, and when limited to local sensing, will not be able to detect long-range errors (Fig. 1, Movie S1).

In this article, we focus on the impact of fatal errors stemming from wrongfully placed bricks or permanently stalled robots on the structure. While many factors may lead to these errors, e.g. hardware malfunction and environmental disturbance, we focus on the subset related to robot motion and perception, as we have found that these are the most common cause of failure to complete construction.

To address these errors, we explore an indirect correction mechanism inspired by *decay*, which plays an important role to nest construction in natural swarms [19]. This means that depositions are no longer permanent and that the construction environment is no longer static. We start with uninformed

Manuscript received March 1, 2022; accepted June 29, 2022. This letter was accepted as contributed paper by Program Chair Yasushi Nakauchi, Editor-in-Chief Hong Zhang, and General Chair Shugen Ma upon evaluation of the reviewers' comments. This work was funded by a Packard Fellowship for Science and Engineering, GETTYLABS, and NSF #2042411.

[1]School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, USA. jc34723@cornell.edu, kirstin@cornell.edu

decay patterns similar to evaporation, and then introduce two variations: a spatially biased decay pattern informed by particularly problematic parts of the structure, and a time-evolving decay pattern with a rate informed by the construction progress. Note that structure decay can be enforced by environmental forces, human intervention, or specialized robots. Importantly, such specialized robots would not target erroneous depositions explicitly and therefore would not require global knowledge, long-range sensing, or reliability beyond what the normal construction robots are capable of.

Specifically, we explore decay-based error correction in the context of a high-fidelity simulation of the TERMES CRC system [2], [16], where minimalistic, locally informed robots follow a stochastic plan to assemble user-specified, 3D structures (Fig. 1, Sec. II-III). We explore the detailed error characteristics of this system (Sec. IV) and use those to inform decay patterns for improved progress (Sec. V-Sec. VII). Although we use the TERMES system to ground our findings, the concept of decay-based error correction translates across platforms, and may prove key to secure long term autonomy in CRC systems.

II. TERMES OVERVIEW

For completeness, we briefly highlight the relevant features of the TERMES system as a case study next.

Physical Platform: TERMES consists of robots and patterned bricks which are wide enough for the robot to climb onto. Their actions include: move forward one brick, turn 90°, pick up and place bricks. The robots can climb at most one step up or down, and place bricks in front of themselves at the same level. They cannot place bricks in between two other bricks (gaps). The robots can perceive and avoid nearby robots, and can register progress and inclines as they move over the bricks, as well as the relative height of neighboring locations (i.e. the difference between flat structures, steps, unclimbable cliffs, and the ground).

Map Compiler: The construction process involves a pre-compilation step that takes a blueprint as an input, and compiles a "policy" given to deployed robots (Fig. 2 #1). The policy is a 2D map showing the number of bricks to be placed in every location, as well as uni-directional robot traffic directions between locations. Neighboring locations that lead to and from a location are termed the *parents* and *children* of it, respectively. The map further marks a start location. Robots enter the structure, pick up a brick, and then follow the traffic directions until they reach the end, after which they circumnavigate the structure perimeter until they relocate the start. If they find a valid deposition site while on the structure, they deposit their brick. The path robots follow through the structure is non-deterministic, therefore the structure may grow in many ways. In later versions of the TERMES compiler, it was shown that assigning biased transition probabilities between locations can increase the construction progress by orders of magnitude [20], and that policies which optimize the number of parallel paths enhance both construction time and probability of success when robots are prone to errors [18].

Placement Ruleset: To avoid obstructing progress by others, brick placements adhere to a ruleset which is designed around robots' inability to: 1) sense the state of locations beyond their immediate neighbors, 2) traverse cliffs, and 3) fill gaps. The ruleset [2] states that a brick can be placed iff: 1) The location height is less than the blueprint-specified height; 2) All parents have a height greater than the location, or have reached their desired height; 3) All children have a height equal to the location, or their desired heights differ from the desired height of the location by more than 1. When (non-erroneous) robots correctly follow the policy and ruleset, they can provably complete a compiled structure, despite the lack of global knowledge.

Predictive Local Checks (PLC): By leveraging PLC, robots use local knowledge to correct many navigation errors immediately[18]. To do this, the robot senses the height of neighboring locations before and after an action (Table I), comparing what is expected to what is sensed. These checks can have three outcomes: 1) If the local neighborhood has distinct features, robots may use these to infer its pose after a faulty action, correct its belief and carry on. 2) If there are duplicate features, robot can still register that an error happened, but cannot infer the new pose. 3) If there are no features (uniform height), errors cannot be detected.

Desired action	**Outcome**	**Probability [%]**
Move forward on level ground (FL)	by 1 brick	99.800
	by 0 brick ($FL_{e,0B}$)	0.100
	by 2 bricks ($FL_{e,2B}$)	0.100
Climb up (CU)	by 1 brick	99.900
	by 0 brick ($CU_{e,0B}$)	0.067
	by 2 bricks ($CU_{e,2B}$)	0.033
Climb down (CD)	by 1 brick	99.99
	by 0 brick ($CD_{e,0B}$)	0.001
	by 2 bricks ($CD_{e,2B}$)	0.009
Turn by 90° (T)	by 90°	98.600
	by 0° ($T_{e,0°}$)	0.700
	by 180° ($T_{e,180°}$)	0.700

TABLE I: Error statistics used in the simulator.

III. TERMES SIMULATION

To support the study of errors and decay-based error correction, we extended upon a TERMES simulator introduced in [18]. In every step of our simulation, each agent takes an error-prone action (Table I). To make simulation run-time practical, we lower the number of errors, and consequently construction times, by a factor of 10 compared to the real system statistics [2]. Simulation extensions particular to this paper include 1) upgrading from a single- to a multi-robot simulator which enables us to reproduce collisions and congestion phenomena observed in the real system; 2) more realistic robot actions failures and failure profiles; and 3) robots which can perceive (in accordance with the hardware [2]) whether they are on or off the structure, and whether a neighboring location holds a traversable or untraversable cliff to the structure perimeter.

The simulator includes four types of action failures. Forward motions, such as move forward on level ground (FL) and climbing up/down (CU/CD) can fail, either because the

robot moves too far (2 bricks) or stays in place (0 bricks). Turning motions (T) can fail such that the robot either turns too far ($180°$) or stays in place ($0°$). For each action failure, we describe the outcome in the subscript and use "e" to denote an error, e.g. a failure to climb up is noted as $CU_{e,0B}$.

We modified the original robot controllers to integrate PLC, such that when errors occur, robots take the following actions: 1) If, based on local features, robots can infer their new pose, they update their belief and carry on. 2) If the local features are symmetric, robots may recognize that an error occurred, but not which one. In this case, to prevent making lasting impacts on the structure, they will not place a brick, and if they find themselves by the perimeter at a height of 1, they will leave the structure.

Note that we focus explicitly on errors that directly affect the construction progress, we do not include: 1) Errors that happen during structure circumnavigation; 2) robots that fall off the side of the structure, since they do not effect others; or 3) docking errors – presumably these will immediately half the system because the entryway is blocked.

In the following, we leverage this simulator to reason about the effectiveness of decay-based error corrections. As such, we include the effect of decay, but not the actual robots that could inflict it. We do, however, ensure that such correction robots would require no sensing or communication abilities beyond those of the original construction robots. We focus our evaluation on 32 10×10 randomly generated 250-brick structures (Fig. 2). We use parallel policies with transition probabilities optimized according to [18]. For these structures, we found that 3 robots make the fastest progress without causing congestion, and use this number for all trials. Unless otherwise noted, all results are assimilated from 100 simulations of each structure. We terminate simulations that run longer than 20,000 actions/robot.

Finally, to help us study the impact of errors on the construction process, we introduce the following metrics:

Construction progress shows the state of construction. We compute this metric by the number of correctly-placed bricks divided by the total number of bricks in the blueprint; 100% marks successful completion.

Construction rate is the frequency of brick placement. The construction rate over a period of time is equal to the number of correctly-placed bricks divided by the number of robot actions. As progress increases, robots need to travel a longer distance to find valid depositions, and therefore the rate decreases as the structure near completion.

Probability of success is computed as the ratio of trials that achieve 100% construction progress.

IV. ERROR ANALYSIS

Action failures do not necessarily hinder construction [18], but may simply cause robots to leave the structure without finding a valid deposition location, or cause bricks to be assembled in the wrong order, but (by luck) avoid creating cliffs and gaps. We refer to errors that completely stall construction as *critical errors* and distinguish between two types. *Wrong placements* (WP) occur when a robot places

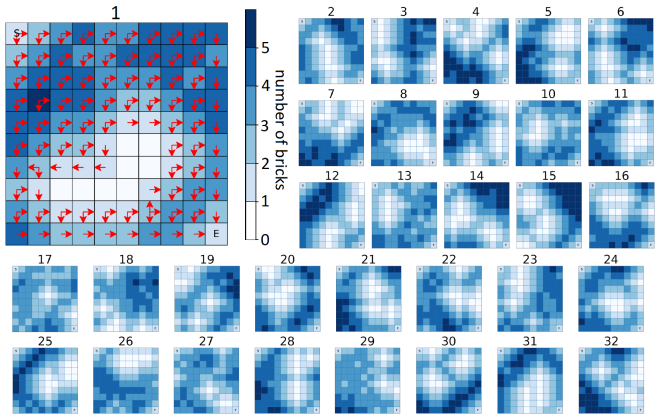

Fig. 2: 32 randomly generated blueprints with 10×10 locations and a total of 250 bricks. Blueprint #1 is enlarged to show an example of the policy traffic directions.

a brick at an incorrect location on the structure or when the placement violates the *ruleset*. *Robot stalling* (RS) errors occur when robots find themselves in a dead-end on the structure effectively blocking all future passage, either because of undetected action failures or WP resulting in gaps or cliffs. Fig. 3 shows an example of three robots attempting to construct blueprint #1. We can see that 6 action errors occur before one that leads to a critical WP which eventually stalls all progress. To emphasize the importance of error correction mechanisms, consider all the randomly generated blueprints shown in Fig. 2. With the somewhat optimistic error statistics reported in Table I, the probability of success is only 11±7%.

To study the construction error distribution, we implemented a "restorative" correction mechanism, which logs and fixes critical errors as they happen. At each time step, this mechanism checks each robot and if any robot has been in the idle state for more than 36 steps consecutively, the robot will be removed from the structure and a RS error will be registered. After each brick placement, the mechanism will check if it meets the placement *ruleset*. If the *ruleset* is violated, the brick will be removed and a WP will be registered. This mechanism, only implemented as an analysis tool, provides insights on the spatio-temporal distribution of critical errors. From simulation, we found that critical errors are rare. On average we found 3.39±3.06 RS errors and 2.97±2.78 WP per 10,000 robot actions (it takes 5,029±2,086 actions/robot to complete a structure on average). Since WP can lead to RS errors, it makes sense that the latter occurs more frequently than the former.

Fig. 4 shows the spatial distribution of critical errors in blueprint #1 with the restorative correction mechanism. Interestingly, this plot indicates that WP and especially RS errors occur at higher rate in particular locations. Similar "error hot-spots" were observed across most structures; Fig. 5 shows the error frequency of the top 5 hot-spots in each blueprint. These amount to only 5% of all locations, but contain 68%±12% of all RS errors and 23±5% of all WP. A preliminary investigation did not reveal obvious spatial correlation between locations with higher frequency of RS errors and WP. Future work, however, may reveal that error

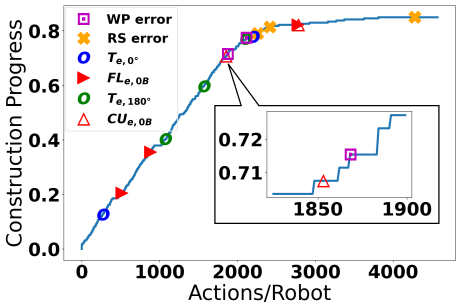

Fig. 3: A typical construction process without error correction mechanisms. The process stalls before it reaches 100%.

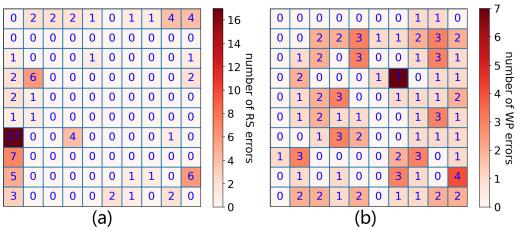

Fig. 4: Spatial distribution of critical errors in blueprint #1. (a) RS errors. (b) WP errors.

rates are higher in locations with 1) higher visiting rates due to the policy or the structure itself funneling robots through these locations, 2) locations with many parents or children, and/or 3) locations that lack unique structural features rendering PLC useless. If this is the case, future work may lower the number of critical errors simply by changing the generated policies.

Fig. 6 shows the temporal distribution of critical errors in all 32 blueprints. The number of critical RS errors increases, first linearly, and then steeply as the structure nears completion. The linear increase is likely due to the fact that robots have to travel further over the structure with every deposition, increasing the risk that errors occur. The steep increase when the structure is near completion is likely due to congestion near the exit, where errors can have a more significant effect. Conversely, the number of critical WP decreases, almost linearly, with increasing construction progress, presumably because placing a brick wrong near the end of the construction progress is less likely to block the path of future robots.

V. UNINFORMED DECAY

Inspired by addition and dissipation in natural processes, we next investigate the use of structure decay to deal with errors. Explained theoretically, our reasoning is as follows. The construction process can be modeled as a finite state Markov process, with a large (combinatorial) number of partial assembly states. The problem with errors is that they create deadlock and produce incorrect absorbing states from which the system cannot escape. Adding even a small active decay rate completely mitigates this problem since the absorbing error states are no longer absorbing and the system has some (small) probability of reaching the only absorbing state, which is the final, correct, assembly. There are two critical observations that affect this model: 1) How

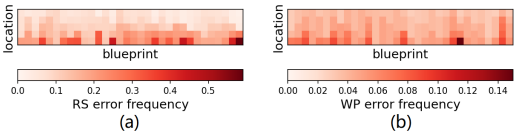

Fig. 5: Top 5 "error hot-spots" in all 32 blueprints. The error frequency is computed as the number of critical errors per location divided by the total number of critical errors. (a) RS errors. (b) WP errors.

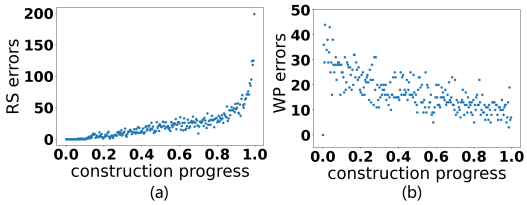

Fig. 6: Temporal distribution of critical errors in all 32 blueprints. (a) RS errors. (b) WP errors.

effective this approach is in practice depends on the details of transition rates and the Markov structure, so that tuning the decay rate relative to the construction progress is essential for practical performance. 2) Without global knowledge, we have no way to determine when the Markov process reaches the final, correct, assembly and when active decay should cease. At best, we can approximate this by the number of bricks taken from the docking station as explained later.

Next, we explore what happens when we apply uniform, low-probability brick and robot decay to every location at every time step. The outcome of attempted brick decay is determined by a decay ruleset which depends on local information only, and ensures that no additional critical errors are caused. As shown in Alg. 1, line 5 ensures that the decay does not cause a gap; line 6 ensures that the decay does not cause a cliff, but also allows decay when the cliff is part of the final blueprint; line 7 ensures that it is physically possible for the robots to add back the removed bricks. Because in some cases, the RS errors can only be corrected by removing the stalled robots, we also include robot decay. One can imagine such decay done by a robot leading others off the structure, or simply pushing them over the structure side.

We tried robot decay and brick decay separately and together with blueprint #1. The average maximum construction progresses without decay, with robot decay, with brick decay, and with decay of both were 66±28%, 69±29%, 78±24%, and 90±18% respectively. This shows that robot decay and brick decay mechanisms complement each other.

For a blueprint with m locations, the decay process can be modeled as m independent Bernoulli processes, in which there is a probability p_e of a decay attempt at each time step. The expected number of decay attempts at each time is then mp_e. However, since only certain locations satisfy the decay ruleset, the expected number of successful decay attempts is less than mp_e. We define the ratio of successful decay attempts to total attempts as the *decay success rate*. We also define the ratio of the number of efficacious attempts that remove critical errors to the total number of successful

attempts as *decay efficacy*. The most important factor in this error correction mechanism is p_e. High p_e enables more frequent decay attempts which increase the chance of correcting critical errors, but can slow down construction. Low p_e makes the decay process less destructive, but may result in impractically long correction times.

We tested the uninformed decay mechanism on the 32 10×10 blueprints. As the average number of critical WP and RS errors is similar, we used the same p_e for both brick and robot decay across all blueprints. For each blueprint, we set p_e to r_c/m, where $m = 100$ locations and r_c is the average construction rate extracted from simulations with the restorative correction mechanism. This construction rate represents an upper bound of the construction rate of a successful construction process when errors are present (since in reality it takes time for the system to correct errors). With this p_e, the expected number of decay attempts per time step is equal to the construction rate, but the actual decay is considerably less due to the decay ruleset.

The chosen p_e is not necessarily the optimal value. Due to the stochasticity of the construction processes and the large amount of possible ways to assemble the structure, analytically finding a decay rate that is globally optimal is difficult. Another option is to find a good range for p_e using parameter sweeps, however, this fine-tuning process can be very time-consuming and is not the focus of this paper.

Fig. 7 shows an example of blueprint #1 being built, with the decay-based correction mechanism. We can see that errors occur frequently, however, at ∼1,700 actions/robot a critical RS error causes the construction process to stall and progress starts decreasing due to decay. After ∼5,000 actions/robot the decay corrects all previous critical errors, effectively restoring the construction rate until the structure reaches ∼87% of its final size. As the construction progress nears 100%, it becomes harder for the robots to locate the last deposition sites and the construction rate drops. Fig. 7(b) shows the long term behavior of the system. As expected, after the construction progress peaks, the system enters a balanced state where the construction and decay rate are roughly equal. This indicates the necessity to lower the decay rate as the construction progress increases (Sec. VII).

Algorithm 1: Decay ruleset. h_i is the number of bricks in location i; j are neighbors of i; and k are children of i. l are the set of i neighbors which are opposite to each other (North-South or East-West).

```
for each time step t do
    for each location i do
        if Outcome of Bernoulli trial with decay
          probability p_e == decay then
            if there is no robot
            and for at least one location x ∈ l: h_i > h_x
            and for all j: |h_j − (h_i − 1)| < 2
            and for all k: |h_k − (h_i − 1)| = 0 then
                remove top brick: h_i = h_i − 1
            end
        end
    end
end
```

In Fig. 8 we compare the maximum construction progress achieved without decay and with uninformed decay for each blueprint. We can see that by introducing a simple uninformed decay mechanism, the construction progress generally improves. On average, the median increases from 59±16% to 95±10%. T-tests of all blueprints showed statistical significance (5% significance level). The probability of success increases from 11±7% to 49±19%. The expense of this simple decay-based error correction is a decrease in construction rate (Fig. 8), which occurs across all blueprints. The median of the construction rate (up to the point of maximum progress) decreases from 0.048±0.007 to 0.026±0.009.

VI. INFORMED DECAY

As discussed in Sec. IV, most critical errors occur in a few locations. Next, we explore the effect of biasing the decay rate for both bricks and robots according to the spatial distribution of critical errors. Each location i now has specific decay probability $p_{e,i}$ equal to the critical error frequency of location i times the construction rate r_c. Notice that the bias does not change the expected number of decay attempts, but can change the resulted decay success rate.

Compared with the uninformed decay, the informed decay achieves similar increase in maximum construction progress and probability of success (Fig. 8)). The medians of the maximum construction progress and the probability of success are 96±5% and 44±9%, respectively. T-tests between informed decay and no decay showed that all blueprints have statistical significance (5% significance level). Noticeably, the median of the construction rate (up to the point of maximum progress) is improved from 0.026±0.009 to 0.037±0.010. T-tests between informed decay and uninformed decay showed that 87.5% of blueprints have statistically significant difference (5% significance level). However, the medians of the decay efficacy for WP and RS errors dropped from 0.036±0.016 and 0.382±0.119 to 0.012±0.009 and 0.154±0.055. We hypothesize, as before, that this is due to the fact that error hot-spots are biased towards locations that experience high flux. Therefore, even though our decay is less effective, it works on locations that are refilled more often. In other words, this method

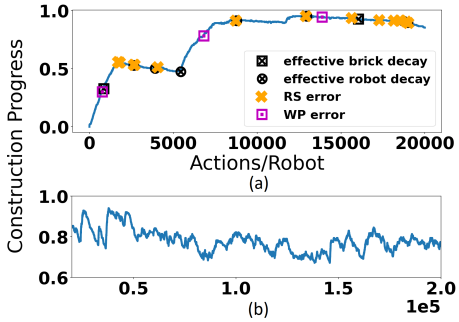

Fig. 7: A typical construction process with decay-based error correction mechanism. (a) Construction before 20,000 actions/robot. (b) Construction after 20,000 actions/robot.

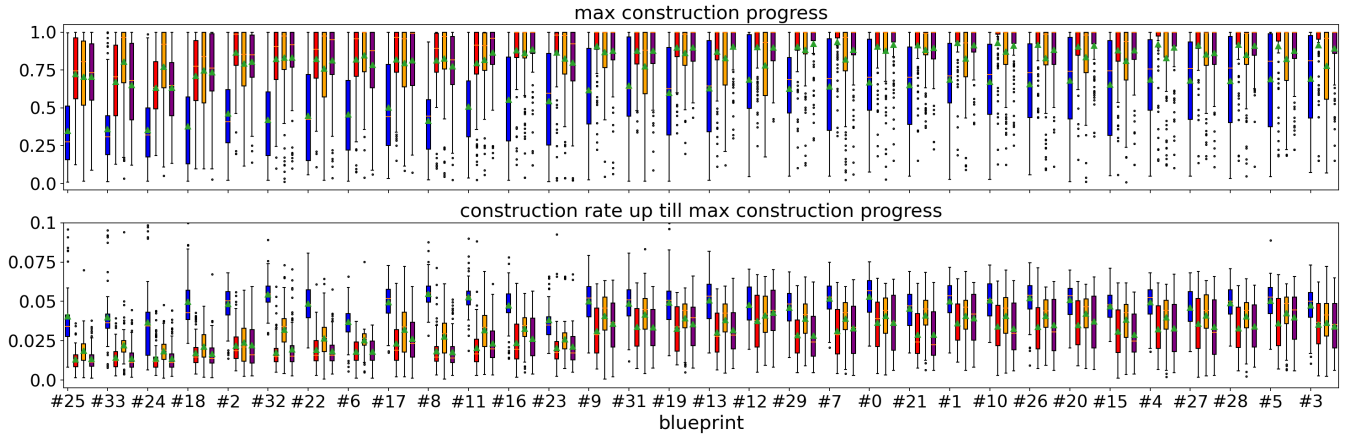

Fig. 8: Average construction progress (top) and construction rate up till the maximum progress (bottom) without decay (blue) and with uninformed decay (red), informed decay (yellow), and time-evolving decay (purple) in all 32 blueprints. Blueprints are sorted by the median of the maximum progress achieved without decay.

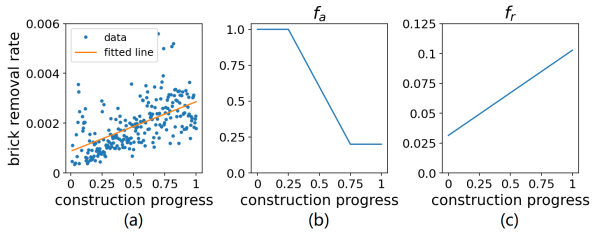

Fig. 9: (a) Average brick removal rate vs construction progress with uninformed decay in blueprint #32. (b) Function f_a and (c) f_r used for adjusting the decay rate and for estimating the construction progress in the time-evolving decay for blueprint #1.

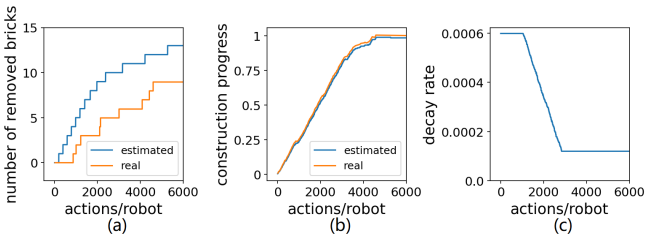

Fig. 10: A typical construction process with time-evolving decay in blueprint #1. (a) Bricks decayed. (b) Construction progress. (c) Decay rate over time.

improves the probability of success and would require more 'decay robots', but also enable faster construction progress as compared to uninformed decay. Note that tuning the decay rate may affect this outcome.

VII. TIME-EVOLVING DECAY

In this section, we explore the effect of adjusting the decay rate throughout the construction process to account for the fact that the number of WP decrease and the construction rate drops as the structure near completion. One way to do this without enforcing global sensing is to estimate the construction progress based on the number of bricks taken from the docking station. To perform the time-evolving decay, we use the unbiased decay pattern seen in the uninformed decay. The decay rate for both bricks and robots start from the same decay rate used in uninformed decay. However, for robots the decay rate will stay constant and for bricks the decay rate will change over time.

The time-evolving decay algorithm is shown in Alg. 2. There are two key tasks: 1) estimating the construction progress (line 3) and setting the decay rate (line 4). The construction progress is estimated as the number of bricks in the structure divided by the number of bricks in the blueprint. The former is simply the number of added bricks minus the number of removed bricks. The number of added bricks can be retrieved from the docking station. The number of removed bricks is estimated by integrating the brick removal rate over time. In discrete time, it is computed as the sum of past brick removal rates. The brick removal rate is a function of the construction progress and the decay rate. We use historical data from the uninformed decay to predict the brick removal rate. This is done by fitting a linear model for the brick removal rate vs construction progress (Fig. 9 (a)). We then scale this linear model by $1/p_e$, where p_e is the decay rate used in the uninformed decay, to obtain the function f_r (Fig. 9 (c)). We can then use f_r to predict the brick removal rate given the current construction progress and decay rate (line 5).

Algorithm 2: Update of decay rate in time-evolving decay. N is the number of bricks taken from the docking station. C is the estimated construction progress. R_d is the set decay rate. R_r is the estimated brick removal rate. f_a and f_r are functions of the construction progress. n is the total number of bricks in the blueprint. β is the initial decay rate.

1 **for** *each time step* t **do**
2 Retrieve $N^{(t)}$ from the docking station.
3 $C^{(t)} = (N^{(t)} - \lfloor \sum_{i=0}^{t-1} R_r^{(i)} \rfloor)/n$
4 $R_d^{(t)} = \beta f_a(C^{(t)})$
5 $R_r^{(t)} = R_d^{(t)} f_r(C^{(t)})$
6 **end**

Given the estimated construction progress, we use function f_a to adjust the decay rate for all blueprints (Fig. 9 (b)). The reasoning behind applying f_a is as follows: 1) For the early stage of the construction (construction progress below 20%), the relatively high initial decay rate is used because critical errors in early stage often have higher impact. 2) For the intermediate stage (progress between 20% and 75%), the decay rate is slowly decreased since number of WP is decreasing. 3) For the final stage (progress above 75%), decay rate remains low to maintain the correction function at a very low cost. An example of how the system adjusts the decay rate over time is given in Fig. 10.

Compared with other decay mechanisms, the time-evolving decay achieves similar increase in maximum construction progress and the probability of success (Fig. 8). The medians of the maximum construction progress and the probability of success are 94$\pm$10% and 48$\pm$18%, respectively. T-tests between time-evolving decay and no decay showed that all blueprints have statistical significance (5% significance level). Compared with the uninformed decay, the time-evolving decay results in a similar decrease in construction rate. The median of the construction rate (up to the point of maximum progress) is 0.026$\pm$0.010. However, the median of the decay efficacy improves by 55%, from 0.036$\pm$0.016 to 0.056$\pm$0.020. The median of the number of bricks removed also decreases by 58%, from 48$\pm$13 bricks to 20$\pm$3 bricks. In other words, the time-evolving decay improves the overall probability of success and would require fewer 'decay robots' at the cost of a slower construction progress, because the cumulative decay does not keep up with the system error frequency. Again, parameters such as β and f_a could be tuned to improve this performance. Future work may also reveal whether combining informed decay and time-evolving decay would further improve the system.

VIII. CONCLUSION

In natural construction processes, the structure state is never static, but rather an emergent outcome of additive and dissipative processes. This paper introduced a similar decay mechanism to correct errors that have high impact on the construction progress and can be commonly seen in most CRC systems, with the distinct benefit that the robots executing the error correction do not need global knowledge or long-range sensing. We found that although decay will slow down the construction rate, we can use both spatially and temporally informed decay mechanisms to lower this effect. Critically, instead of methods relying on accurate detection and correction of errors which can be computationally expensive and subject to errors by itself, the work in this paper demonstrates an alternative method that relies on randomness and local state information. The effectiveness of our proposed methods, shown through extensive simulations, shows that for CRC systems that primarily rely on additive manufacturing by real (i.e. error-prone) locally-aware agents, adding a low-frequency decay process will not hinder the construction progress in the long run, but rather increase the chance of successfully completing the structure. We also found exciting directions for future work related both to the optimal decay rate and the spatial correlation of errors to locations on the structure.

REFERENCES

[1] K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac, "A review of collective robotic construction," *Science Robotics*, vol. 4, no. 28, 2019.

[2] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[3] S. Jokic, P. Novikov, S. Maggs, D. Sadan, S. Jin, and C. Nan, "Robotic positioning device for three-dimensional printing," *arXiv preprint arXiv:1406.3400*, 2014.

[4] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with quadrotor teams," *Autonomous Robots*, vol. 33, no. 3, pp. 323–336, 2012.

[5] M. Saboia, V. Thangavelu, W. Gosrich, and N. Napp, "Autonomous adaptive modification of unstructured environments." in *Robotics: Science and Systems*, 2018.

[6] N. Melenbrink, K. Rinderspacher, A. Menges, and J. Werfel, "Autonomous anchoring for robotic construction," *Automation in Construction*, vol. 120, p. 103391, 2020.

[7] R. L. Johns, M. Wermelinger, R. Mascaro, D. Jud, F. Gramazio, M. Kohler, M. Chli, and M. Hutter, "Autonomous dry stone: On-site planning and assembly of stone walls with a robotic excavator," *Construction Robotics*, 2020.

[8] K. Petersen and R. Nagpal, "Complex design by simple robots: A collective embodied intelligence approach to construction," *Architectural Design*, vol. 87, no. 4, pp. 44–49, 2017.

[9] N. Nedjah and L. S. Junior, "Review of methodologies and tasks in swarm robotics towards standardization," *Swarm and Evolutionary Computation*, vol. 50, p. 100565, 2019.

[10] M. Allwright, N. Bhalla, H. El-faham, A. Antoun, C. Pinciroli, and M. Dorigo, "Srocs: Leveraging stigmergy on a multi-robot construction platform for unknown environments," in *International conference on swarm intelligence*. Springer, 2014, pp. 158–169.

[11] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," *Proc. Robotics: Science & Systems VII*, 2011.

[12] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *2013 IEEE International conference on robotics and automation*. IEEE, 2013, pp. 855–862.

[13] Y. Liu, S. M. Shamsi, L. Fang, C. Chen, and N. Napp, "Deep q-learning for dry stacking irregular objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1569–1576.

[14] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014.

[15] B. Jenett, A. Abdel-Rahman, K. Cheung, and N. Gershenfeld, "Material–robot system for assembly of discrete cellular structures," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4019–4026, 2019.

[16] K. H. Petersen, R. Nagpal, and J. K. Werfel, "Termes: An autonomous robotic system for three-dimensional collective construction," *Robotics: science and systems VII*, 2011.

[17] Y. Liu, M. Saboia, V. Thangavelu, and N. Napp, "Approximate stability analysis for drystacked structures," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8819–8824.

[18] J. Chen, Y. Liu, A. Pacheck, H. Kress-Gazit, N. Napp, and K. Petersen, "Errors in collective robotic construction," in *International Symposium Distributed Autonomous Robotic Systems*. Springer, 2021, pp. 269–281.

[19] S. J. Turner, "Extended physiology of an insect-built structure," *American Entomologist*, vol. 51, no. 1, pp. 36–38, 2005.

[20] Y. Deng, Y. Hua, N. Napp, and K. Petersen, "A Compiler for Scalable Construction by the TERMES Robot Collective," *Robotics and Autonomous Systems*, vol. 121, 2019.