Planning for Robotic Dry Stacking with Irregular Stones

Yifang Liu, Jiwon Choi and Nils Napp

Abstract Dry stacking with found, minimally processed rocks is a useful capability when it comes to autonomous construction. However, it is a difficult planning problem since both the state and action space are continuous, and structural stability is strongly affected by complex friction and contact constraints. We propose an algorithmic approach for autonomous construction from a collection of irregularly shaped objects. The structure planning is calculated in simulation by first considering geometric and physical constraints to find a small set of feasible actions and then refined by using a hierarchical filter based on heuristics. These plans are then executed open-loop with a robotic arm equipped with a wrist RGB-D camera. Experimental results show that the proposed planning algorithm can significantly improve the state of the art robotics dry-stacking techniques.

1 Introduction

As humans, we critically depend on modifying our environment by constructing shelter, infrastructure for transportation, water, energy, and waste management, as well as structures that regulate the natural environment such as dams, drainage, and avalanche protection. Labor and materials are the main cost drivers of the construction industry, which also produces approximately 500 million tons of demolition

Yifang Liu
Department of Computer Science and Engineering, University at Buffalo, NY.
e-mail: `yifangli@buffalo.edu`

Jiwon Choi
Department of Computer Science and Engineering, University at Buffalo, NY.
e-mail: `jiwoncho@buffalo.edu`

Nils Napp
Department of Computer Science and Engineering, University at Buffalo, NY.
e-mail: `nnapp@buffalo.edu`

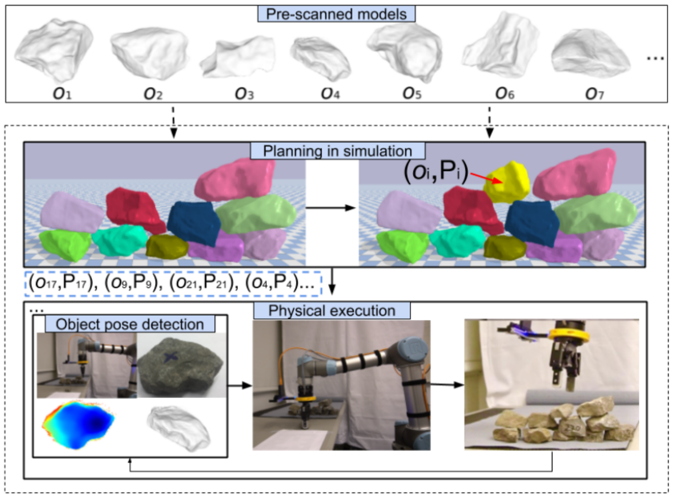

Fig. 1: System overview.

waste, mostly in the form of concrete [1]. Cement production accounts for (≈ %5) of global CO_2 emission [2]. Robotic construction with in-situ (found) materials simultaneously addresses primary cost drivers of construction while mitigating its environmental impact. This idea has been explored in specialized situations. Driven by the need for resource conservation in space, NASA has studied in-situ material use for extraterrestrial environments. Launching building materials into space is very costly, yet simple structures–such as berms, walls, and shelters–might be readily built from minimally processed but rearranged materials [3]. Such *utility structures*, i.e., structures that have a specific function but whose exact shape matters less, are also important on Earth, for example, erosion barriers for changing coastlines, temporary support structures in disaster sites, or containment structures made from contaminated materials from a nuclear or chemical leak. One particularly well-suited construction method for such types of utility structures is dry-stacking stones. This ancient method was practiced by humankind since 3500 B.C. [4], and some of the oldest man-made structures used this method. Theoretically, robots are ideally suited for this work, since robots make work safer and physically less demanding. However, a lack of understanding of how to pose and solve assembly planning problems of irregular natural material into in-situ functional structures is currently hindering robots from performing such useful construction tasks.

We present an algorithmic approach for solving the planning problem of assembling stable structures from a collection of irregularly shaped rigid objects. The application is to enable dry-stacking with found, minimally processed rocks. We focus on the problem of high-level placement planning for rocks to build stable structures, and dry-stack structures with tens of rocks, which significantly improves the state of the art. These plans are executed open-loop without additional tactile sensing. Although, our results suggest that large-scale dry-stacking robots would benefit from better physical feedback during the construction process. The whole process is shown in Fig 1. To the best of our knowledge, this is the first work that can automatically dry-stack a wall with 4 courses using natural irregular rocks, which could significantly benefit the large-scale outdoor construction robots.

The rest of the paper is structured as follows: Section 2 provides a brief overview of related work; Section 3 describes the planning algorithm; results are presented in Section 4. Finally, Section 5 concludes the paper and discusses future work.

2 Related Work

The fundamental questions in autonomous construction are how to specify, plan, and execute the placement of building elements to achieve a final structure. Approaches [5, 6, 7, 8] differ in each of these aspects, and range from determining the assembly order of elements whose position is known in advance [5] to formulating building plans that need to pick the type, shape, and pose of elements to build approximate shapes, or to build structures that fulfill specific functions [9].

There has been work in multi-robot construction [10, 11, 12, 13]. There are approaches that multiple robots build structures that are large compared to themselves, such as decentralized approximation of large 2D structures [13], construction of 3D structures with robots that can navigate partial structures [11], and flying robots [10] use building elements that snap together. These works address the issue of decentralized coordination between building agents and use uniform building materials.

Using deformable and amorphous materials in a distributed setting has been explored in [14, 9, 15, 16, 17]. In contrast to the presented work, the complicated material model simplifies the planning, because the deformable nature of the building material can compensate for inaccuracies in placement and irregularity of the environment.

When building with rigid irregular objects, small surface features substantially affect the friction and stability. Compounding this difficulty, microfracture formation during execution can deform the surface. This makes planning and stably placing irregular objects fundamentally different from building with regular objects that have predictable contact geometry. In our previous work [18, 19, 8], we proposed an architecture for solving the dry-stacking problem, based on heuristics and deep Q-learning to build stable large-scale structures using physics simulation in 2D.

From instructional books for dry-stacked masonry, e.g. [20, 21, 22], there are guidelines for building a stable structure. For example, it is good to place large stones with inward-sloping top surfaces on the corners. Filler stones may be used to fill small voids created by the irregular shapes of the natural stones and can help to produce a more solid footing to the layers above. Such heuristics can provide a structured approach in making assembly decisions, but in their description, much is left to experience and human judgment.

The methods presented by Furrer et al. [6] propose a pose searching algorithm that considers structural stability using a physics simulator. In addition, they present an autonomous system, using a robot manipulator, for stacking balancing vertical towers with irregular stones. The pose searching cost function considers support polygon area, kinetic energy, the deviation between thrust line direction and the normal of the support polygon surface, and the length between the new object and the CoM of the previously stacked object.

3 Methods

In this section, we first describe the notations used in this paper; then we elaborate the planning algorithm for stacking irregular stones. Finally, we provide the object pose detection pipeline used in physical execution.

3.1 Notation

The *world frame* is a 3-D coordinate system where the gravity is in the negative z-direction, and the goal is to construct a target structure $T \subset \mathbb{R}^3$, i.e., a subset of the world space that should be filled by selecting and placing elements from a set of objects O. Each object is a connected subset of $\mathbb{R}^3$ with the origin at the center of mass (CoM).

An assembly $A = (a_1, a_2, ..., a_I)$ is a set of I assembled objects, where each element $a_i = (o_i, P_i)$ is a pair containing an object $o_i \in O$ and its pose $P_i = (p_i, R_i) \in SE(3)$. The position $p_i \in \mathbb{R}^3$ denotes the CoM position of object o_i in the world frame, and $R_i \in SO(3)$ is its orientation. Empty space set is a set $E \subset T$ s.t. every point $e \in E$ can be connected by a straight line from ∂T to ∂A without passing through any other a_i and ∂T, and ∂A denote the boundaries of T and the assembly, respectively. This definition excludes the complicated internal voids created by stacking irregular objects from counting as empty space. The top surface is given by $S = \partial E \cap \partial A$, i.e., the overlapping area of the empty set E and the assembly A, where ∂E denotes the boundaries of empty space E. We define the action space $X(o_i)$ of object o_i to be restricted to have the CoM in E:

$$X(o_i) = \{(p_i, R_i) | p \in E\}. \tag{1}$$

The world is initially assumed to be empty of objects, aside from a support surface at the bottom of T. We want to find an assembly strategy for autonomous agents, i.e., picking a sequence of elements o_i and actions from $X(o_i)$ to build an assembly that occupies the target structure T subject to physical contact, friction, and gravity constraints.

3.2 Structure Planning

This section presents an assembly planning algorithm for irregular objects. Similar to [8], we design a greedy heuristic approach to find the next best pose from a set of feasible poses for a given object. For each object, we use a physics simulator to generate a finite set of feasible *stable* poses, strategically reduce this set, and choose the best available pose. By repeating this sequence, we incrementally build the structure in a systematic fashion.

Algorithm 1: Feasible Poses Generation

```
Data: $o_i$: object
Result: Feasible Poses Set
{(x_j, y_j, z_j)} ← discretize position;
for each (x,y,z) in {(x_j, y_j, z_j)} do
    for $N_{ori} \leftarrow 0$ to $N$ do
        $R \leftarrow$ random generate an orientation;
        reset $o_i$ pose to $(x,y,z,R)$;
        while $N_{contact}(o_i) < 3$ do
            step physics simulation once;
        end
        pause physics simulation;
        reset $o_i$ linear and angular velocity to 0;
        step physics simulation once;
        while $o_i$ is not stable do
            step physics simulation once;
        end
        if distance between current pose $P_i$ and $(x,y,z,R) <$ Threshold then
            add current pose $P_i$ to Feasible Poses Set $X_F$;
        else
            continue;
        end
    end
end
```

3.2.1 Feasible Poses Generation

Autonomously understanding the physical aspects of a system requires interaction with it in order to obtain feedback. Here, we simplify the problem of learning the system behavior by using a physics simulator to find *Physically Stable* configurations. We approximate the real-world state with simulation and provide a practical and efficient stability estimate of the system without actually having to physically interact with the external world. This helps us to acquire a good prior estimate for the system.

Since the action space (p and R) is continuous (Eq. 1), we first sample the action space in such a way that the position p is discretized, and each position p corresponds to a set of randomly sampled orientation R. We then make use of a rigid body simulator to find physically stable configurations. The simulator proceeds by, first selecting an initial placement (position and orientation) for a given object on the surface of the built structure, and then simulating the forces acting on the object until it settles into a stable pose, see Algorithm 1. Although the number of possible initial placement is large, a substantial amount of them settles down into a small subset of feasible poses. This set of feasible poses for an object o_i denotes $X_F(o_i)$. Even though all the poses in the feasible poses set are stable ones, many of these feasible poses are poor choices and result in low stability. In the next section, we will discuss how to refine this set by using heuristics gathered from instructional literature for masonry books [20, 21, 22].

3.2.2 Action Space Reduction

The refinement of action space is a hierarchical filtering approach, where each filter removes poses that do not meet the minimum requirement for a satisfactory placement according to a specific heuristic. The set of filters used in this work is presented below:

- Support polygon area: the area of an object's support polygon. A higher value of support polygon correlates to a stable footing for the object. Similar to method [6], in order to robustly find the support polygon from the sparse contacts, we update the simulator 10 steps and collect all the contacts. Then Principal Component Analysis (PCA) is used to reduce the 3D contacts to 2D points. Finally, the convex hull of these 2D points is calculated as support polygon area.
- Normal of support polygon: the normal direction of the support polygon. It measures how much the normal direction deviates from the thrust line direction vector [6].
- Neighbor height: the difference in heights of the object, after its placement, with its left and right neighboring objects. It helps maintain leveled surfaces in the structure. The height of an object is represented by CoM height.
- Stone top surface sloping: the top surface angle of the object at a given pose. In building a wall, we prefer inward sloping angles to prevent stones from the top layers to fall down from the structure ([22, Pg. 49]).
- Interlocking: the number of objects in the structure that are in contact with the current object at a given pose. The use of this feature allows for staggered layering and thus helps to prevent vertical stacking in the structure, see [21, Pg. 19].

Each filter applies each one of the features aforementioned. The reductions are applied hierarchically for each object as follows:

- The original feasible poses set denotes X_F.
- At *Filter 1*, only select poses that have an inward sloping top surface angle. The remaining poses after applying this filter denotes X_{F1}.

- At *Filter 2*, discard poses with dot product $\|\mathrm{n}_i \cdot \mathrm{v}_i\|$ less than the mean of all poses from X_{F1}, where $\mathrm{n_i}$ represents the normal direction of the contact polygon, and $\mathrm{v_i}$ is the thrust line direction vector. The set of poses after applying this filter denotes X_{F2}.
- At *Filter 3*, remove poses with support polygon area less than the mean of the support polygon area value of all stable poses. The remaining poses after applying this filter denotes X_{F3}.
- At *Filter 4*, this is based on the current state of the structure. If it is not a corner placement, only choose poses whose centroid heights are lower than the average centroid heights of corner stones at the current course. The set of poses after applying this filter denotes X_{F4}.
- At *Filter 5*, remove the poses whose number of interlocking objects are smaller than the mean number of interlocking. The set of poses after applying this filter denotes X_{F5}.

This hierarchical reduction model is carefully designed such that a random pose at each level is more desirable than a random pose drawn at the earlier filtered levels. It is also designed such that no good possible stable poses are removed earlier before reaching the final selection filter. The relation between the various sets of poses is shown in Eq. 2.

$$X_F \supset X_{F1} \supset X_{F2} \supset X_{F3} \supseteq X_{F4} \supset X_{F5}, \tag{2}$$

where X_F is defined in Section 3.2.1.

Unlike the pose searching algorithm used in [6], which combines terms similar to Filters 2 and 3, as well as other heuristics into a single scalar cost function and finds poses by gradient descent, the planning algorithm proposed in this paper first considers geometric and physical constraints using a simulator to find a discrete set of feasible actions and further refines this set by using a hierarchical filter based on heuristics gathered from the instructional materials. This approach eliminates the need for tuning the relative weights in a scalar cost function. Without the need of cost-tuning the algorithm is more adaptable to different stones with various physical properties, such as size, density, friction, etc. The reason is that with a single scalar function the weights are coupled and the relative importance depends on the object sets physical properties. However, in the hierarchical filter each term is assessed in isolation and thus the method is less sensitive to the change of the object physical property for a given task.

3.2.3 Proposed Algorithm

Algorithm 2 describes how a structure is constructed. The inputs are the set of available objects along with the target structure to be built. During construction, it builds the structure course by course, and within courses, it first places the corner stones with the inward slope in the two course extrema, as shown in Algorithm 2 Line 2;

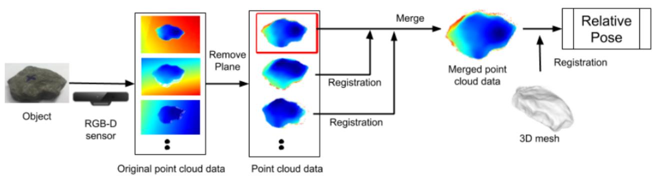

Fig. 2: Object pose detection pipeline.

then it builds the middle area within the course (Lines 3-5). The output is the set of assembly steps.

Algorithm 3 describes the steps to select an object and its pose for the placement. The inputs are the set of remaining objects and their type (corner stone or random stone), since different object types may require different hierarchical filters. The first step is to choose a random object (Line 3) and collect feasible poses (Section 3.2.1) of this object (Line 4). Then it applies Hierarchical Filter (Section 3.2.2) to reduce the action space at Line 5. If the reduced action space is not empty, we select one pose from it; otherwise, we try this procedure again for a different object until it reaches the maximum number of trials (Lines 6-11).

Algorithm 2: Proposed Assembly Approach

```
   Data: $O$: object dataset, $T$: target structure
   Result: Assembly steps
1  while target area $T$ still has room left to build do
2      place Corner stones with inward slope in the two course extrema;
3      while current course still has room left to build do
4          place stone in the current course ;
5      end
6  end
```

3.3 Object Pose Detection

In physical execution, we need first to detect the pose of the object in the scene. We start by capturing a set of point cloud data of an object from different views via an RGB-D camera; second step is to filter out the points that do not belong to the current object by removing the plane points from point cloud data using Point Cloud Library (PCL) [23]; third, the set of remaining point cloud data are merged together. We apply global registration to provide an initial transformation and Iterative Closest Point (ICP) algorithm to further refine the transformation using Open3D library[24]; finally, we run registration on merged point cloud data and pre-scanned 3D mesh of the object to get the relative pose between them. Similar to the third step above, the registration also contains global registration and ICP. The whole pose detection pipeline is shown in Fig. 2.

Once the relative pose between the current object and the 3D mesh is detected, we use the manipulator to pick up the stone and apply the same transformation to the end-effector of the manipulator to place the stone as the planned pose.

Algorithm 3: Place Stone

```
Data: $B$: set of available objects ($B \subseteq O$), object type
Result: Placed object pose
1  $n \leftarrow 0$;
2  while $n \leq$ Maximum Number of Trials do
3      $b \leftarrow$ randomly choose one object from $B$;
4      $X_F \leftarrow$ feasible poses set;
5      $X_{final} \leftarrow$ apply Hierarchical Filter to $X_F$ ;
6      if $X_{final} \neq \emptyset$ then
7          place one of the $X_{final}$ poses;
8          return;
9      else
10         $n \leftarrow n + 1$;
11     end
12 end
```

4 Experiments

In this section, we first describe the experimental setup, then we show the stone towers and stone walls using the proposed planning, as well as the comparison between the pose searching algorithm proposed in [6] and the proposed method.

4.1 Experimental Setup

As shown in Fig. 4, a UR5 (Universal Robots) manipulator equipped with a ROBOTIQ 2-Finger gripper is used in the manipulation task. An Intel®RealSense^{TM} SR300 RGB-D camera is attached to the UR5 arm for point cloud data acquisition. MoveIt! [25] package is employed for motion planning. We collect 23 shale stones as irregular objects, which are specifically selected in order to fit the size of the gripper. The average weight of selected stones is 193g with a standard deviation of 90g. The outer bounding box size of the stones are 0.0791±0.0144m, 0.0585±0.0086m, and 0.04±0.0088m. During physical execution, objects are manually fed into pickup area. The gripper grasping position varies depending on the detected object position, but the gripper orientation remains the same. The stone 3D model is acquired with a Matter and Form 3D Desktop Scanner. Fig. 3 shows some samples of the stones and their corresponding 3D mesh models. The object pose detection and manipulation parts are implemented using Robot Operating System.

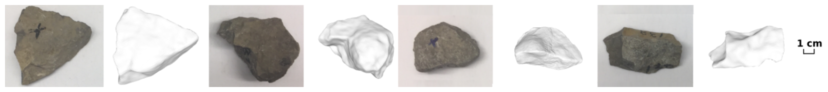

Fig. 3: Irregular shale stones and their corresponding 3D meshes.

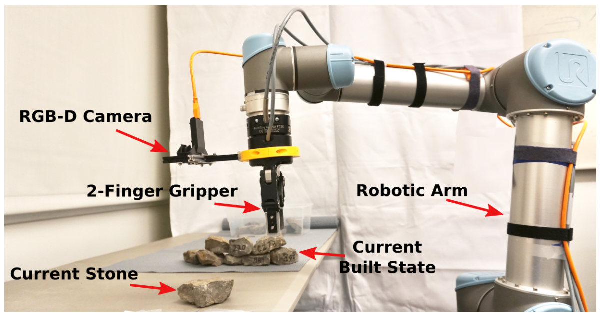

Fig. 4: An overview of the experimental setup.

4.2 Results

The autonomous building system is shown in Fig. 1. In this section, we first compare the proposed algorithm with other work in simulation and physical execution; then we will show the physical execution results of building the walls.

4.2.1 Stone Tower

The goal is to build a vertical stone tower using the pre-scanned 3D mesh models as a test structure to evaluate planning under stability constraints. We compare the proposed method with the pose searching algorithm proposed in [6]. The comparisons are conducted in Pybullet physics simulation [26].

The pose searching method used in [6] places each object on the top object of the existing stacking using a physics engine. A cost function is introduced to evaluate the “goodness” of each pose, which considers 4 elements: contacts area C_i, kinetic energy E_{kin}, the length between the newly placed object pose P_j and the previously placed object pose P_i (denoted as $\mathrm{r}_{P_jP_i}$), and the dot product between normal of the contact polygon and the trust line direction vector $\|\mathrm{n}_i \cdot \mathrm{v}_i\|$. The cost function is defined as:

$$f(P_i) = w_1 C_i^{-1} + w_2 E_{kin}(P_i) + w_3 \|\mathrm{r}_{P_jP_i}\| + w_4 \|\mathrm{n}_i \cdot \mathrm{v}_i\| \ , \tag{3}$$

where w_j are tuned for the object set. After assigning the cost to the valid contact pose, gradient descent is used to search the local optimal pose P_i^*.

Since we use a different type of stone from that of [6], and the size of the stones are also different given that we use different arms and grippers, the w_j given by [6]

Table 1: Average stone number comparison

Random	Area	Kinetic Energy	Distance	Deviation(m)	Deviation(d)	Weighted Cost	Proposed
3.4565	4.3944	4.0845	3.9362	3.2	4.0207	4.75	5.4118

Table 2: Physical execution (✗ represents the stone drops down, ✓ represents the stone can be successfully placed to the desired pose, ∅ means no more planned object left.)

Stone Number	Tower 1	Tower 2	Tower 3	Tower 4	Tower 5	Tower 6	Tower 7	Tower 8	Tower 9
4th	✓	✓	✗	✓	✓	✓	✓	✓	✓
5th	✓	✓		✗	✗	✓	✓	✓	✓
6th	✓	✓				✓	✗	✓	✗
7th	✗	✗				∅		✗	

are no longer optimal in this application. Furthermore, we modify the last component of Eq. 3 as $w_4\|\mathrm{n}_i \cdot \mathrm{v}_i\|^{-1}$ given the fact that the larger the dot product is, the smaller the cost should be. At last, we apply Bayesian optimization for Gaussian process modeling called GPyOpt [27] to optimize the weights.

Fig. 5 depicts the results of using different cost functions and the proposed hierarchical filter based algorithm (Algorithm 1). Since the heuristics used in building a vertical tower is different from that of a wall, we modify the filters to fit the task. The filter contains contact polygon area C, distance $\mathrm{r}_{P_jP_i}$, and top surface slope. We also evaluate each cost component used in Eq. 3 separately. We can see that the proposed hierarchical filter algorithm has more chances to build a vertical towers that have more than 5 stones than all other compared methods. Table 1 gives the average number of stones each algorithm can build. It also shows that the proposed method can build more stones than other methods. We randomly select 9 towers planned by the proposed method from all of the towers that have a height of at least 6 stones for physical execution. Table 2 shows the building process. Since the first three stones can always be placed successfully, we start the table from the 4th stone. We can see that $4/9$ can be built up to 6 stones, and only $1/9$ drops at the 4th stone. Compared to the towers built in work [6], which only builds up to 4 stones with a chance of $2/11$, our method can build higher stone towers both in simulation and physical execution. The reasons for the failures in our execution could be: object pose detection error, pickup error, opening gripper moves already placed objects, the difference between object 3D mesh and the real object, error in contacts modeling in simulation, etc. Fig. 6 (a) and (b) illustrates one tower example of the proposed algorithm in the simulation environment and physical execution.

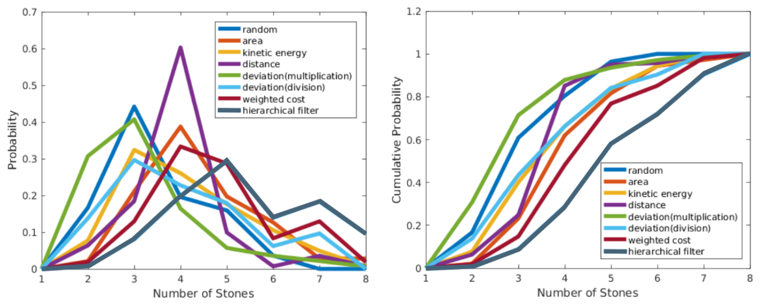

Fig. 5: Vertical tower building results. For each method, we build 150 different vertical stone towers in simulation. The x-axis shows the number of stones each tower has, and on the left figure the y-axis shows the percentage of each height; on the right figure, the y-axis represents cumulative percentage of each height. “random” means that we randomly pick a pose from feasible poses set; “area” represents that the cost function only contains contact polygon area (C) one element, so as “kinetic energy” (E_{kin}), “distance” ($\mathrm{r}_{P_jP_i}$) and “deviation” ($\|\mathrm{n}_i \cdot \mathrm{v}_i\|$). For “deviation”, we test both multiplication $\|\mathrm{n}_i \cdot \mathrm{v}_i\|$ and division $\|\mathrm{n}_i \cdot \mathrm{v}_i\|^{-1}$. The “weighted cost” uses the optimized cost function. The proposed “hierarchical filter” significantly outperforms the other methods.

Table 3: Stone wall execution failure rate

	Course 1	Course 2	Course 3	Course 4
Poor placement	0.14	0.18	0.29	0.48
Structure collapse	0	0.04	0.07	0.05

4.2.2 Stone Wall

In this experiment, the goal is to build a stone wall. The structure is planned in simulation using Algorithm 2, and then the UR5 manipulator places the stones to the planned pose. The planned wall has 4 courses, and each course has 3 to 5 stones. The execution order is manually calculated but complies with the assembly order in the simulation if there is no collision during assembly due to the gripper, i.e., the simulated assembly order does not take into account clearances for the fingers or grippers, but if problems exist reordering stones within a course often fixes potential collisions. As mentioned in the previous section, several reasons may yield failures during the execution process. We categorize the failure cases into two classes: poor placement and structure collapse. Poor placement contains bad grasping, wrong object pose detection, and the drop of current stone after placement. Structure collapse is the case that after placing the current stone, more than 1 stone falls down. In this experiment, 7 out of 13 walls can be successfully built without collapsing. Table 3 shows the failure rate during execution. We separate the execution process based on different courses. We can see that as the course increases, the poor placement rate also increases. All the previous minor errors building up to larger errors lead to more failure cases. Fig. 6 (c) and (d) shows an example of a planned wall in the simulation and the wall built in the real world using the robotic arm without collapsing.

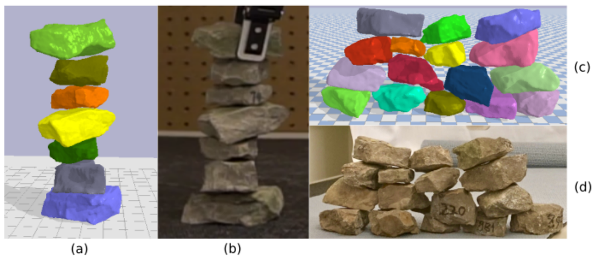

Fig. 6: Vertical tower and wall in simulation and corresponding physical execution results.

5 Conclusion and Future work

The proposed method is able to plan placements for a set of irregularly shaped rocks and build stable dry-stacked structures. Similar to the previous work, we use a rigid body simulation engine in order to find stable poses for rocks. We introduce two primary innovations, first is that we only use the physics engine to create a finite set of feasible poses, second is a layered refinement architecture that significantly improves performance compared to optimized scalar cost functions in evaluating the quality of feasible poses. We also introduce new filtering terms, which are specific for building walls with interlocking layers, compared to vertical stacks.

We focus on high-level placement planning as it is a central issue in dry-stacking. The overall system could be significantly improved with a more specialized and robust execution system, specifically reactively re-planing in the face of errors and unmodeled action outcomes, and in incorporating tactile feedback during placement and pose evaluation. The instructional literature suggests this approach for human builders as well: candidate stones are placed, wiggled, and then either removed or stabilized by wedging small rocks into crevices until the newly placed rock is stable. In any of these situations, having better high-level placement plans that can be executed in an open-loop fashion will be beneficial and this paper represents significant progress in that direction.

Acknowledgements We would like to thank Hironori Yoshida and Dr. Marco Hutter for their valuable input regarding the comparison algorithm, and Jackie Chan for scanning the stones. This work was partially supported by NSF Grant #1846340 and the SMART CoE at UB.

References

1. USEPA. Advancing sustainable materials management: 2014 fact sheet. *United States Environmental Protection Agency, Office of Land and Emergency Management, Washington, DC 20460*, (November):22, 2016.
2. Green in Practice 102 - Concrete, Cement, and CO2.

3. Gerald B Sanders and William E Larson. Progress Made in Lunar In Situ Resource Utilization under NASA's Exploration Technology and Development Program. *Journal of Aerospace Engineering*, 26(1):5–17, 2013.
4. Wikipedia contributors. History of construction — Wikipedia, the free encyclopedia, 2019.
5. I-Ming Chen and Joel W Burdick. Determining task optimal modular robot assembly configurations. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 132–137. IEEE, 1995.
6. Fadri Furrer, Martin Wermelinger, Hironori Yoshida, Fabio Gramazio, Matthias Kohler, Roland Siegwart, and Marco Hutter. Autonomous robotic stone stacking with online next best object target pose planning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2350–2356. IEEE, 2017.
7. Volker Helm, Selen Ercan, Fabio Gramazio, and Matthias Kohler. Mobile robotic fabrication on construction sites: Dimrob. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4335–4341. IEEE, 2012.
8. Vivek Thangavelu, Yifang Liu, Maira Saboia, and Nils Napp. Dry stacking for automated construction with irregular objects. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.
9. Nils Napp and Radhika Nagpal. Distributed amorphous ramp construction in unstructured environments. In *Distributed Autonomous Robotic Systems*, pages 105–119. Springer, 2014.
10. Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction with quadrotor teams. *Autonomous Robots*, 33(3):323–336, 2012.
11. Kirstin Petersen, Radhika Nagpal, and Justin Werfel. Termes: An autonomous robotic system for three-dimensional collective construction. June 2011.
12. Kirstin H Petersen, Nils Napp, Robert Stuart-Smith, Daniela Rus, and Mirko Kovac. A review of collective robotic construction. *Science Robotics*, 4(28):eaau8479, 2019.
13. Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.
14. Kathrin Doerfler, Sebastian Ernst, Luka Piškorec, Jan Willmann, Volker Helm, Fabio Gramazio, and Matthias Kohler. Remote material deposition. In *International Conference, COAC, ETSAB, ETSAV*, pages 101–107, 2014.
15. Maira Saboia, Vivek Thangavelu, Walker Gosrich, and Nils Napp. Autonomous adaptive modification of unstructured environments. *Proc. Robotics: Science & Systems XIV (RSS 2018)*, 2018.
16. Touraj Soleymani, Vito Trianni, Michael Bonani, Francesco Mondada, and Marco Dorigo. Autonomous construction with compliant building material. In *Intelligent Autonomous Systems 13*, pages 1371–1388. Springer, 2016.
17. Maira Saboia, Vivek Thangavelu, and Nils Napp. Autonomous multi-material construction with a heterogeneous robot team. In *Distributed Autonomous Robotic Systems*. Springer, 2018.
18. Yifang Liu, Maira Saboia, Vivek Thangavelu, and Nils Napp. Approximate stability analysis for drystacked structures. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.
19. Yifang Liu, Seyed Mahdi Shamsi, Le Fang, Changyou Chen, and Nils Napp. Deep q-learning for dry stacking irregular objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1569–1576. IEEE, 2018.
20. Kevin Gardner. *Stone Building*. The Countryman Press, 2017.
21. Charles McRaven. *Building stone walls*, volume 217. Storey Publishing, 1999.
22. John Vivian. *Building Stone Walls*. Storey Publishing, 1976.
23. Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.
24. Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.
25. Ioan A Sucan and Sachin Chitta. Moveit! *Online at http://moveit. ros. org*, 2013.
26. E Coumans, Y Bai, and J Hsu. Pybullet physics engine, 2018.
27. GPyOpt. Gpyopt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.