From Rocks to Walls: a Model-free Reinforcement Learning Approach to Dry Stacking with Irregular Rocks

André Menezes Pedro Vicente Alexandre Bernardino Rodrigo Ventura
Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Portugal
andre.menezes@tecnico.ulisboa.pt, {pvicente, alex, rodrigo.ventura}@isr.tecnico.ulisboa.pt

Abstract

In-situ resource utilization (ISRU) is a key aspect for an efficient human exploration of extraterrestrial environments. A cost-effective method for the construction of preliminary structures is dry stacking with locally found unprocessed rocks. This work focus on learning this task from scratch. Former approaches rely on previously acquired models of rocks, which may be hard to obtain in the context of a mission. In alternative, we propose a model-free, data driven approach. We formulate the problem as the task of selecting the position to place each rock on top of the currently built structure. The rocks are presented to the robot in sequence. The goal is to assemble a wall that approximates a target volume, given the 3D perception of the currently built structure, the next object and the target volume. An agent is developed to learn this task using reinforcement learning. The deep Q-networks (DQN) algorithm is used, where the Q-network outputs a value map corresponding to the expected return of placing the object in each position of a top-view depth image. The learned policy outperforms engineered heuristics, both in terms of stability of the structure and similarity with the target volume. Despite the simplification of the task, the policy learned with this approach could be applied to a realistic setting as the high level planner in an autonomous construction pipeline.

1. Introduction

For a long term human exploration of extraterrestrial environments, such as the Moon and Mars, it is essential to use native materials as replacement to resources otherwise brought from Earth at great expense. This is commonly referred to as in-situ resource utilization (ISRU) and, amongst other applications, is useful for the construction of planetary infrastructures [13]. Initial settlement structures, such as roads, platforms and shade walls, may be built with unprocessed or minimally processed local rocks using the ancient method of dry stacking [16]. Although rudimentary, this technique has been proven to produce long lasting structures, while requiring very low pre-processing time and energy. Due to the increased risk and limitations imposed by human missions [1], it is important to have systems with the capability of autonomously setting up infrastructure.

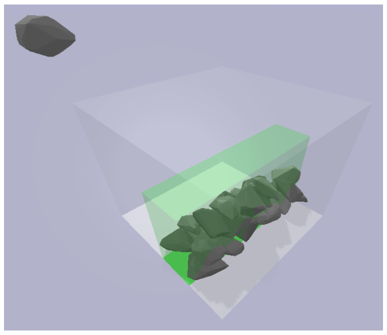


Figure 1. Visualization of the environment. The white box represents the region where the agent can observe the currently built structure, which corresponds to the region where placements are allowed. The green box represents the target volume. The new object is presented at the top left corner of the image. The boxes are represented for visualization and are not physically present in the environment.

However, assembling a structure from a set of irregularly shaped rocks is a difficult task, usually performed by experienced humans and requiring some amount of intuition. It is not clear how to translate this into an autonomous system, specially in a context where no models of the environment are available. Model-based approaches are not suitable within this context. For this reason, the problem of dry stacking in extra-terrestrial environments requires a learning from scratch and model-free approach. An increasingly successful approach to autonomously learning such difficult tasks is reinforcement learning (RL). Concretely, the field of deep RL has seen a number of breakthroughs in recent years, such as the deep Q-networks (DQN) algorithm [10]. We argue that optimization-based approaches are difficult to devise since it is not clear how the sequential nature of the problem should be translated.

With this work, we propose a formulation for the problem of autonomously building a stable dry stack wall with irregular 3D blocks that is compatible with a model-free RL approach. We aim to show that it is possible for an agent to learn this complex task from scratch.To accomplish this, a deep neural network architecture is developed to learn a mapping of the state representation to action values using DQN.

1.1. Related work

Some previous works exploit a physics engine to plan a stable structure from a set of pre-scanned irregular rocks. Lambert and Kennedy [6] developed an application that outputs an assembly plan for a set of prism-shaped rocks, a 2-D simplification of the problem, where the pose of each rock is obtained using simulated annealing to minimise a simple heuristic based on the vertical coordinate of the rock position. Nielsen and Dancu [11] propose a system for assisting real-time dry stone wall construction, where a good placement is found in simulation by dropping the 3D model of a stone at different positions and with different orientations. The reached poses are evaluated using the distribution of contact points and dot product between the contact normals and the vertical direction. Furrer *et al*. [4] present a method to select the best next object and pose by applying gradient descent from several random initial poses to optimize a cost function that takes into account: i) the area of the support polygon, ii) the deviation between the support polygon normal and the gravity direction, iii) the kinetic energy of the structure after the placement and iv) the distance between centers of mass of next and previous objects. This approach is validated in a real world setting with a pipeline capable of constructing vertical stacks of up to four irregular rocks with a robotic manipulator. Liu *et al*. [7] propose an approach that successively reduces a finite set of stable poses, generated by the simulator, by applying a hierarchical sequence of filters based on heuristics. Concretely, the area and normal of the support polygon, the height relative to neighboring objects, the top surface sloping and interlocking (number of objects in contact with the current one) are used. In an experimental setting, this method is shown to outperform the work of Furrer *et al*. [4] in building vertical stacks and to be able to construct walls of four courses. This was preceded by previous work of the same authors in a simplified 2-D setting [8, 17], including a RL approach [9] in which the DQN algorithm is used to learn the values of each computed stable pose, which are used instead of the sequence of heuristics.

The novelty of our work lies in: (i) developing a setup in which it is possible to learn from scratch how to dry stack irregular rocks, without requiring previously acquired object models or internal physical simulation, including a new metric for the reward shaping of the dry stacking task, and (ii) the development of a two branched neural network, inspired by the Siamese architecture, that deals with two semantically different inputs (the perception of the rock to be placed and the currently built structure). To the best of our knowledge, there is no previous work that explores a completely model-free approach to this problem.

2. Background

In a RL setting, an agent interacts with an environment in order to learn a behaviour that maximizes a reward signal [15].

The environment can be represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p)$. At each time step t, the agent observes the environment's state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$ accordingly. This causes the environment to shift to state $s_{t+1} \in \mathcal{S}$, which comes with the reward $r_{t+1} \in \mathcal{R}$. The tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ represents the transition at time step t. The transition probability distribution $p(s_{t+1}, r_{t+1}|s_t, a_t)$ is a property of the environment, and can be factorized as $p(r_{t+1}|s_t, a_t, s_{t+1})p(s_{t+1}|a_t, s_t)$. While $p(s_{t+1}|a_t, s_t)$ is purely a function of the environment dynamics, $p(r_{t+1}|s_t, a_t, s_{t+1})$ depends on the underlying goal. A more general representation is obtained by explicitly including the dependency on the goal $g \in \mathcal{G}$, as $p(r_{t+1}|s_t, a_t, s_{t+1}, g)$.

The agent must learn a policy that maximizes the return $G_t = \sum_{t'=t+1}^{\infty} \gamma^{t'-t-1} r_{t'}$, where the discount factor γ defines the importance given to delayed rewards. This policy may be obtained implicitly from an estimate of the action-value function $q_{g,\pi}(s,a) = \mathbb{E}[G_t \mid s_t = s, a_t = a, \pi, g]$, by selecting the action that maximizes the expected return from a state s, given the policy π and the goal g. From the definition of G_t, it is possible to state the Bellman optimality equation $q_{g*}(s,a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} q_{g*}(s_{t+1}, a') \mid s_t = s, a_t = a, g]$, where q_{g*} is the action-value function for the optimal policy given the goal g [15].

2.1. Deep Q-networks

The DQN algorithm [10] uses a deep neural network to learn a function approximation Q of the optimal action-value function q_{g*}. This is done by minimizing the temporal difference error $\delta_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a|\mathbf{w}^-) - Q(s_t, a_t|\mathbf{w})$, derived from the Bellman optimality equation. $\mathbf{w}$ and $\mathbf{w}^-$ are the parameters of the current and target networks, used to estimate the action-values. The target network is updated with the parameters of the current network at a defined period and kept constant otherwise for stability reasons. At each interaction with the environment, the agent selects an action according to an exploratory policy (e.g. ε-greedy) based on the current action-value estimates and stores the transition in a replay memory. The network update is then performed by sampling a minibatch of transitions from the replay memory and performing a stochastic gradient descent step (or any improved optimization algorithm) to minimize $\sum_{t_i \in \text{batch}} \text{loss}(\delta_{t_i})$, with some defined loss function (e.g. quadratic).

The set of value functions for different goals may be unified using an universal value function approximator [14]. This can be achieved by using a deep neural network that takes the goal as an additional input, such that $Q(s, a, g|\mathbf{w})$ is an estimate of $q_{g*}(s, a)$. With this formulation, the function approximation allows to generalize not only over a potentially large state space, but also over a potentially large set of goals.

3. Methodology

The proposed approach to learn a policy using model-free RL is divided in two parts. First, a simplified version of the task is formulated as a RL environment with discrete action space. Then, an agent architecture is developed to learn a value function for the environment, using DQN.

3.1. Environment

The environment is set up in a physics engine to simulate the task of assembling a structure that approximates a target volume, using a sequence of irregular blocks. At each time step, the next object in the sequence is presented to the agent, as well as the currently built structure and the target volume. The agent chooses the new object position (i.e, takes an action) and a careful placement of the object on top of the current structure, with the same orientation as it was presented, is simulated in the environment. The simulation is then run until all objects stabilise. This process is repeated until all objects in the sequence are placed. Figure 1 shows a visualization of the environment.

A state of the environment is represented as a pair of elevation maps, containing: (i) an overhead view of the current structure (see Figure 2(b)), and (ii) a bottom view of the new object (i.e. the side of the object that will be in contact with the structure, see Figure 2(c)). An elevation map is a 2D array whose elements represent the elevation at a given discretised (horizontal) position. The elements are always greater or equal than zero, with zero corresponding to the minimum elevation. An action is represented by a pair of indexes (i, j), corresponding to the coordinates of the overhead elevation map (see Figure 2(e), where the object placing position is shown in yellow). The goal is represented with an elevation map with the same dimensions as the overhead view, but representing the target volume (Figure 2(a)).

3.2. Reward shaping

Reward shaping is a key aspect of RL. The learned behaviours are critically affected by the reward signal, so it must clearly capture the intended goal [15]. At the same time, it should be distributed in a way that makes learning feasible.

A simple way to translate the goal of approximating a target volume into a reward value is by using the intersection over union (IoU) between the volumes of the current structure and the target, which can be computed using the corresponding elevation maps. However, this metric presents two flaws for this context. Firstly, it considers empty spaces under the objects as part of the volume of the structure. This may lead the agent to place the objects with large spaces between them in order to increase the volume of the intersection. This can undermine the stability of the structure. Secondly, it ignores undesirable object displacements. For instance, an object placed on an unstable position that happens to fall inside the target volume will still contribute to the positive reward. However, object movement and unstable placements are always not desirable since they can affect the long term stability of the structure.

The first flaw can be tackled with a metric equivalent to the IoU, but instead of volumes, we exploit the number of objects to define the metric. We can represent (i) the intersection between the target and the current structure as the number of objects currently inside the target volume; (ii) the target as T, which is the ideal case where all the objects are inside the goal; and (iii) the current structure as the number of objects already placed, t. The union is obtained using the property $|\mathcal{X} \cup \mathcal{Y}| = |\mathcal{X}| + |\mathcal{Y}| - |\mathcal{X} \cap \mathcal{Y}|$, where the elements of the right hand side are obtained as described in (i), (ii) and (iii). With this definition, it is possible to address the second flaw by directly applying a discount factor to the contribution of each object, based on the distance between its current pose and the placing pose. The new metric is then given by:

$$\text{DIoU}_t = \frac{\sum_{i=0}^{t-1} b_t^{[i]} \cdot \mu_t^{[i]}}{T + t - \sum_{i=0}^{t-1} b_t^{[i]}}, \tag{1}$$

where $b_t^{[i]}$ is 1 if the object i is inside the target at time step t and 0 otherwise, and $\mu_t^{[i]}$ is the applied discount. This metric lays in the range $[0, 1]$, with 1 corresponding to the ideal case of a terminal state in which all objects are inside the target volume with no displacements from the original poses. The value of $b_t^{[i]}$ can be determined by retrieving the object position (center of mass) from the simulator and checking if it is inside the target volume. The discount can be defined as:

$$\mu_t^{[i]} = \begin{cases} 0 & \text{if } |\Delta \mathbf{x}_t^{[i]}| > \Delta \mathbf{x}_{\max} \vee |\Delta \theta_t^{[i]}| > \Delta \theta_{\max} \\ \left(1 - \left(\frac{|\Delta \mathbf{x}_t^{[i]}|}{\Delta \mathbf{x}_{\max}}\right)^{c_{\mathbf{x}}}\right) \cdot \left(1 - \left(\frac{|\Delta \theta_t^{[i]}|}{\Delta \theta_{\max}}\right)^{c_\theta}\right) & \text{o/w,} \end{cases} \tag{2}$$

where $\Delta \mathbf{x}_t^{[i]}$ and $\Delta \theta_t^{[i]}$ are the 3D translation and rotation distances between the original pose of object i and its pose

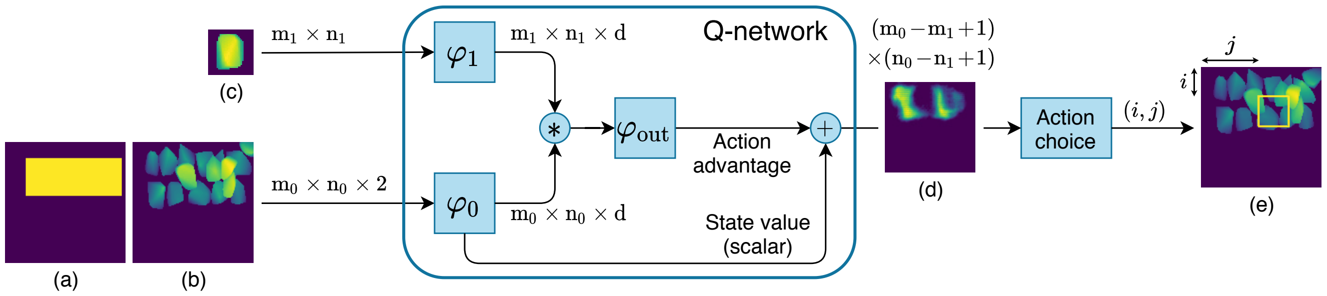


Figure 2. Diagram of the proposed approach. The input of the Q-network consists of the $m_0 \times n_0$ elevation maps of (a) the target volume and (b) the current structure, and the $m_1 \times n_1$ elevation map of (c) the next object. The output is a value map (d) with the values estimated for each position, from which an action (i, j) is selected, as visualised in (e). The modules φ_0 and φ_1 perform a pixelwise feature extraction from the inputs, returning d channel images.

observed at time step t. The parameters Δx_{max} and $\Delta\theta_{max}$ represent the maximum allowable distances. If the displacement exceeds one (or both) of these distances, its contribution to the reward is zero. The exponents c_x and c_θ control how μ decreases with the distances. Higher values ($c_x, c_\theta > 1$) make it less sensitive to small displacements.

The value of the metric in the end of the episode DIoU_T provides an evaluation of the obtained structure and could be sent as a reward in the terminal state, which perfectly captures the goal of reaching the ideal case described. However, it is hard to learn the value of each action from it, since the actions influence the final result in complex and unclear ways. In order to make the problem tractable, each contribution to the final value may be rewarded immediately, as $r_t = \text{DIoU}_t - \text{DIoU}_{t-1}$. This way, an action that increases the metric (e.g. successfully places an object inside the target) is immediately rewarded, while an action that makes it decrease (e.g. causes part of the existing structure to collapse) is penalised.

3.3. Model generation

In order to allow the agent to learn a generalisation, the experience provided by the simulated environment must be diverse, which includes the object models used. As it is hard to find a large number of models of real irregular objects (e.g. natural rocks), these models are synthetically generated. This allows the generation of a large set of models from which the sequences used in each episode are sampled.

The process used to generate 3D models is inspired on the method presented by [17] to generate datasets of irregular 2D shapes. Our method takes a rectangular prism as a starting point and then applies a sequence of random vertex displacements and mesh subdivisions. The displacements are sampled from a truncated normal distribution scaled by an irregularity parameter ς. After each subdivision, the scale of the random displacements of the new vertices is halved. The irregular object is obtained as the convex hull of the displaced vertices. This is done for two reasons: firstly, the usage of convex shapes makes the collision computations more efficient in the physics engine; secondly, the convex hulls actually resemble more natural rocks (often subject to erosion) than the obtained irregular shapes. From initial experiments in this setup without using convex hulls, we observed that the algorithm performs similarly for both cases. Thus, this assumption seems not to affect significantly the solution, while allowing faster and more stable simulations.

The meshes of the models are generated using Trimesh[1]. Each mesh is positioned and oriented in the model's frame so that its oriented bounding box is centered at the origin and aligned with the axes, with the largest extent aligned with the first axis and the smallest with the third (vertical). A value for the material density is uniformly sampled from an interval $[\rho_{min}, \rho_{max}]$, which simulates diverse materials or materials with variable composition or porosity. The robot does not use this information whatsoever, but induces robustness in the learning process. Some examples of models generated with different values of ς are presented in Figure 3.

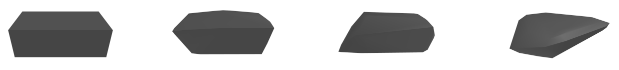

(a) $\varsigma = 0.$ (b) $\varsigma = 0.1.$ (c) $\varsigma = 0.25.$ (d) $\varsigma = 0.5.$

Figure 3. Models generated with different values of the irregularity parameter ς (3 subdivisions performed).

3.4. Agent

The agent takes as input the elevation maps of the target volume, the current structure and the next object, as depicted in Figure 2. The first two are stacked to form a two channel input, which makes the spatial relation between the current structure and target volume clear to the

[1] M. Dawson-Haggerty *et al.* Trimesh. https://trimsh.org/, version 3.8.1.

agent. A mapping of these inputs to the values of all possible actions (i.e. corresponding to all possible offsets between the inputs) is obtained with a cross-correlation like operation that slides the object elevation map through the overhead elevation maps. This is translated into the neural network architecture depicted in Figure 2, which is inspired on the fully convolutional Siamese network presented in [3]. Differently from the Siamese architecture, where the inputs have the same meaning and must be matched, we use two different fully convolutional modules φ_0 and φ_1 as the branches, with no shared weights. Both modules perform a dense (pixelwise) feature extraction and return outputs with the same number of channels, which are then cross-correlated. Each branch uses the U-net architecture [12], which consists of a contracting path that gradually extracts higher level, lower resolution features, and a symmetric expanding path that gradually increases the resolution. As the object elevation map is smaller and contains less semantic information, a shallower network is used for its branch. The network sizes used are 5 levels in φ_0 and 3 in φ_1, and both branches use convolutional layers with 16 channels at the first level (see [12]).

The network architecture is extended with one additional fully convolutional module φ_{out}, placed at the output of the cross-correlation. This is motivated by the fact that the network must learn a specific value function, estimated from the collected rewards. With the additional module φ_{out}, the output of the cross-correlation is mapped to the intended values. This gives the branches more freedom to express the features, as the output of the cross-correlation is not forced to fit the value function. This module consists of two 3×3 convolutional layers with 16 channels and ReLU activation followed by a 1×1 single channel convolution, with no activation, that maps the 16 channels to the output value map.

Additionally, the network is adapted to match the dueling architecture [21]. The intrinsic value of a state is greatly influenced by the current structure, as it defines the availability of potentially good positions for a new object. Additionally, it reflects the stage of the episode (i.e. an advanced episode, indicated by a structure with many objects, means lower expected return because the terminal state is closer). Although the shape of the object by itself may also be indicative of the overall expected quality of the placement, this influence is not significant when compared to the overhead input. This observation is translated into the architecture by extracting the state value from the branch φ_0, which is then combined with the advantage function returned by φ_{out} as in [21]. This value is extracted by applying a global average pooling layer to the higher level, lower resolution feature maps in φ_0 (i.e. the end of the contracting path of the U-net), and then applying one fully connected layer with 256 units and ReLU activation and a single fully connected unit (no activation) to map the feature vector to a scalar.

3.5. Baselines

Three levels of baseline performance are considered for comparison with the learned policies. The first corresponds to the policy that randomly samples actions from the complete action space. The second is given by the policy that randomly samples actions inside the target volume, which corresponds to capturing the notion of the goal of the environment. For the third level, a position choice criterion is introduced in the form of a heuristic function that must be optimized. This level corresponds to capturing a basic understanding of the environment dynamics. The heuristic is given by the cross-correlation between the elevation maps of the current structure and object. As the values in the elevation maps are always greater or equal to zero, the cross-correlation effectively results in a blurring operation on the overhead elevation map with a filter given by the object shape. The local minima of the cross-correlation output are likely to correspond to concavities in the structure where the object fits, because sharper local minima in the elevation map would have been blurred out. Additionally, the global minimum is also the lowest point in the blurred surface, which is consistent with the height criterion used in related work (e.g. [7]). The action is selected as the minimum cross-correlation value inside the target region, thus, this baseline is a optimization-based approach to the dry stacking problem.

Although several methods on dry stacking were presented on the related work section, none of the methods were considered as a baseline since they require that the agent has an internal model of the environment to compute the set of possible positions. This is not available in this setup, which is consistent with the ISRU paradigm.

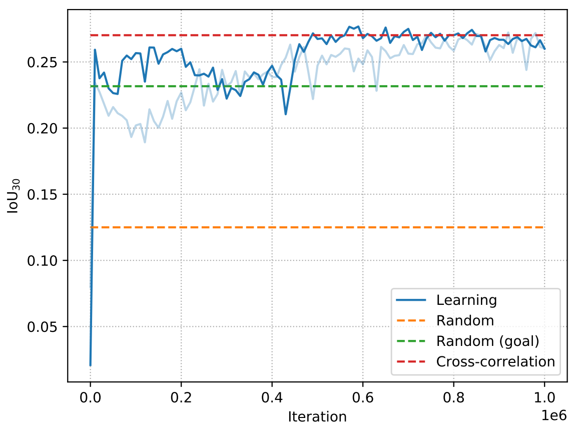
4. Results

A set of experiments[2] is performed on the described environment, simulated using PyBullet[3]. The dataset of rock models used is composed of 500 models generated with each value of $\varsigma \in \{0.5, 0.55, \dots, 1\}$, resulting in a complete dataset of 5500 models. The size of the overhead and object elevation maps are 128×128 and 32×32, respectively. The episode length T is set to 30 objects. Two metrics are used to evaluate the obtained policies. The first is the IoU, which is a measurement of the similarity between the built and target volumes. The second is the average of the discounts defined in (2), given by

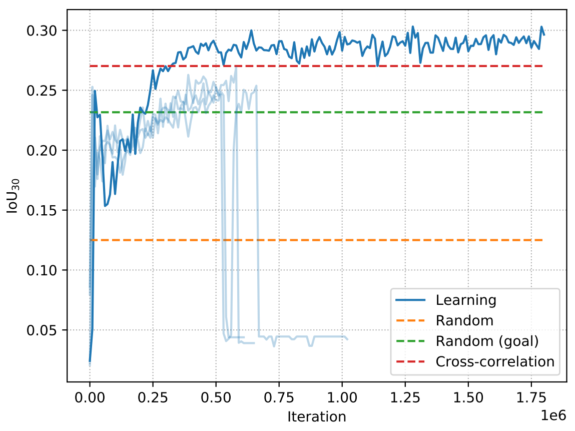
$$\mathrm{AD}_t = \frac{1}{t} \sum_{i=0}^{t-1} \mu_t^{[i]}, \tag{3}$$

[2]Implementation available at `https://github.com/menezesandre/stackrl`.

[3]E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, version 2.9.6.

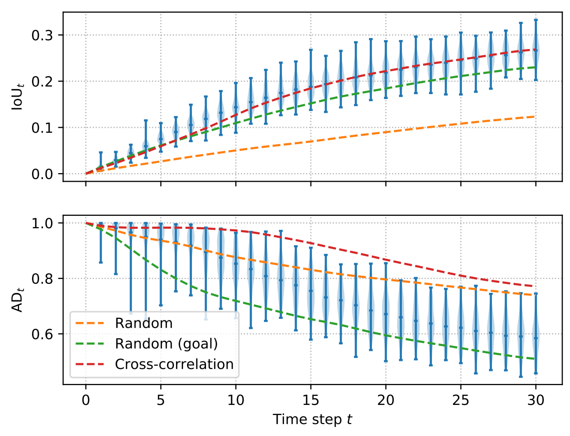


(a) Rewards generated by the IoU.

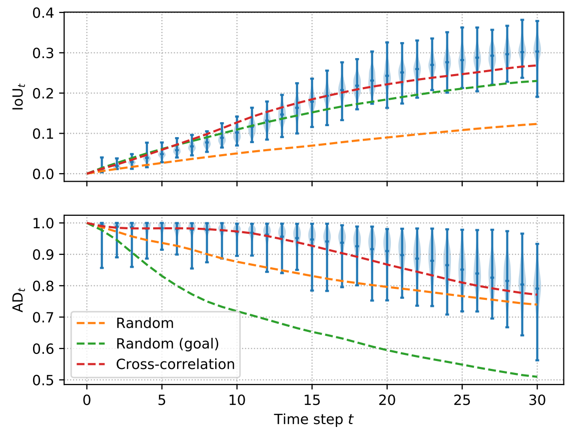


(b) Rewards generated by the DIoU.

Figure 4. Evolution of the rewards during training. In blue are several runs of our proposed model-free RL approach trained using (a) IoU rewards and (b) DIoU rewards. Solid blue corresponds to the best run. The orange, green and red horizontal lines correspond to the average results obtained with each baseline policy over 100 episodes.



(a) Rewards generated by the IoU.



(b) Rewards generated by the DIoU.

Figure 5. Evolution of the intersection over union (IoU) and average discount (AD) metrics for the learned policies during an episode. Values obtained over 100 episodes. In blue are the distributions obtained with the learned policies. In orange, green and red are the average values obtained with each baseline policy.

which is a metric of the similarity between the planed structure (given by the poses were the objects were placed) and the actual structure (given by the current poses of the objects). The AD$_t$ evaluates the effectiveness and stability of the placements.

The training sessions are performed using the DQN algorithm with a replay memory of size 400000, minibatch size of 32, and 2 transitions collected between network updates. The network optimization uses Adam [5] with learning rate $6.25 \cdot 10^{-5}$ and exponential decay rate 0.95 for the first and second moment estimates.

One training iteration of our approach takes on average 0.15s of continuous process time (CPU for the simulation and GPU for inference and backpropagation) on the used hardware (an Intel(R) Core(TM) i7-7820X, and a GeForce GTX 1080 Ti, respectively). Thus, a training session with 10^6 iterations lasts around 42 hours.

Two training sessions are run with rewards generated with the IoU and four with rewards generated by the DIoU (1). At intervals of 10000 iterations, the current greedy policy is evaluated with 100 episodes. Figure 4 shows the evolution of the results during training, compared with the results of the three baseline levels. The curves highlighted with the stronger colour are considered the best run for each reward shape, and correspond to the same network initialisation for both.

From the learning curves in Figure 4, it is observable that the notion of the goal is easily (and quickly) learned. This corresponds to reaching the second level of baseline performance (green line), given by the policy that samples random

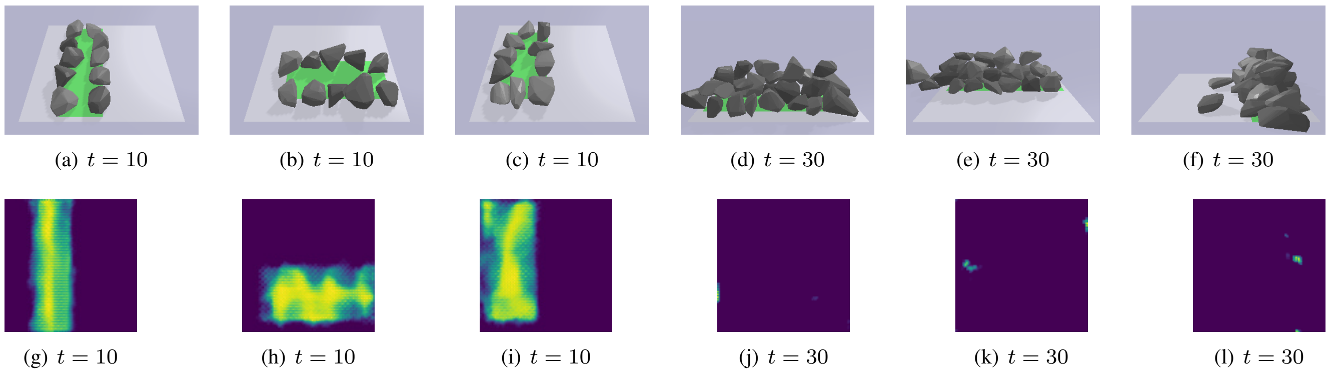


Figure 6. Examples of structures assembled with the learned policy are presented on the top row [(a) to (f)] and the corresponding value maps estimated by the trained Q-network on the bottom row [(g) to (l)]. The examples show initial ($t = 10$) and advanced ($t = 30$) stages of the episode. The initial stages illustrate the initial strategy, while the advanced stages depict the obtained structures. The value maps at $t = 30$ reflect the scarcity of stable positions at that stage.

actions from the target region. Afterwards, the dynamics of the environment are gradually captured by the value functions. The best run with DIoU rewards (solid blue line in Figure 4(b)) is able to learn a placing criterion that outperforms the third baseline level, given by the cross-correlation heuristic. However, it can be verified that the rewards generated by the DIoU result in returns that are difficult to predict. Indeed, the frequent breakdowns of the training sessions are one of the reasons for this hypothesis. These breakdowns correspond to one of the layers in the network stopping to produce activations for any input, which results in the prediction of equal values for all actions and the consequent performance breakdown. The layer that does not produce activations blocks backpropagation, which stops the learning process. This outcome could be prevented with network architecture modifications (e.g. residual connections). On the other hand, the policies learned with the IoU are more consistent, but do not significantly outperform the baseline. To further analyse the policies obtained from the best run with each metric, a set of 100 episodes is used to access the evolution of the evaluation metrics, IoU and AD, during an episode. The results are shown in Figure 5 comparing them to the average evolution obtained with the baselines. Table 1 reports the average and standard deviation of the final values of the metrics obtained with each policy (learned and baselines).

Regarding the policy learned with the IoU, it is possible to observe that it produces generally unstable placements. The AD curve in Figure 5(a) shows average values of AD_t closer to the random policy. This ultimately prevents the policy from outperforming the cross-correlation baseline.

From Figure 5(b), the evolution of the IoU and AD throughout the episodes allows an interpretation of the behaviour improvement over the baselines. The IoU starts by increasing slower for the proposed method than for the cross-correlation heuristic. This is explained by the fact that this policy starts by packing a tight first course, which also results in keeping the value of AD closer to one for longer. This initial behaviour, along with a suitable understanding of the dynamics of the environment, end up enabling the assembly of structures more similar to the target and with less displacements from the original poses. Figure 4 shows examples of structures assembled with the learned policy and value maps estimated by the trained Q-network.

Table 1. Evaluation results. Values reported as the average and standard deviation (in parentheses) over 100 episodes.

	IoU_{30}	AD_{30}
Learned with IoU	0.266 (0.026)	0.584 (0.064)
Learned with DIoU	0.303 (0.029)	0.791 (0.058)
Random	0.125 (0.027)	0.738 (0.06)
Random (goal)	0.232 (0.025)	0.507 (0.064)
Cross-correlation	0.27 (0.024)	0.769 (0.046)

A remarkable observation is the strategy learned for the initial stage. The agent learns to start an episode by placing the first objects in the boundary of the target area. This allows the following objects to be supported by an inwards sloping surface, which prevents them from falling out and increases the overall stability of the structure. This behaviour, learned from scratch, is consistent with dry stacking theory [20].

5. Conclusion and Future Works

The major achievement of this work was to develop a setup in which a dry stacking policy can be learned without any previously acquired models of the environment. An agent is able to learn from scratch a policy that captures the goal of the environment and its dynamics. The emerged behaviour is, to some extent, consistent with dry stacking theory [20]. The agent learned the policy using a model-free approach in simulation, which can be directly used in extra-terrestrial environments since the agent is not based on pre-existent models. Our data-driven approach is more suitable for ISRU.

The policies learned with the proposed approach could be applied to a realistic setting as the high level planner in an autonomous construction pipeline, where a lower level feedback controller would be responsible for reaching, grasping and carefully placing the objects [19], while adjusting the action as needed. In opposition to feedforward execution of the planned actions, this would allow the system to better approximate the way a human performs the task. Additionally, the gap between the simulated training and a real environment can be reduced by introducing a source of randomness into the synthetic observations [2, 18] (e.g. random noise in the depth images).

As future work, we plan to improve the network architecture with residual connections to avoid training breakdowns. The overall efficiency of the method can also be improved by giving the agent more freedom on building the structure. One solution is allowing the agent to select the rock orientation (right now, the orientation is fixed). Additionally, a pool of available rocks could be presented to the agent instead of the pre-defined (random) sequence. In this more challenging scenario, the agent should be able to choose (i) the next rock to place on the structure, (ii) its orientation and (iii) where to place it. These improvements can be accomplished as an extension of the current approach, by presenting a set of possible rocks and different orientations as a batch of inputs, and allowing the agent to select which rock and orientation provides a placement with greatest value.

Acknowledgments This work was supported by LARSyS - FCT Project UIDB/50009/2020, P2020 INFANTE 10/SI/2016 and FCT PhD grant PD/BD/135115/2017.

References

[1] Christopher S Allen, Rebeka Burnett, John Charles, Frank Cucinotta, Richard Fullerton, R Goodman, Anthony D Griffith, Joseph J Kosmo, Michele Perchonok, Jan Railsback, Sudhakar Rajulu, Don Stilwell, Gretchen Thomas, Terry Tri, Ray Wheeler, Alyssa Mueller, and Anne Simmons. Guidelines and Capabilities for Designing Human Missions. *Nasa/Tm–2003–210785*, 2003. 1

[2] Alexandre Almeida, Pedro Vicente, and Alexandre Bernardino. Where is my hand? Deep hand segmentation for visual self-recognition in humanoid robots. *arXiv preprint arXiv:2102.04750*, 2021. 8

[3] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H.S. Torr. Fully-convolutional siamese networks for object tracking. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9914 LNCS, pages 850–865. Springer Verlag, 6 2016. 5

[4] 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 *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2350–2356. Institute of Electrical and Electronics Engineers Inc., 7 2017. 2

[5] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2015. 6

[6] Malcolm Lambert and Paul Kennedy. Using artificial intelligence to build with unprocessed rock. In *Key Engineering Materials*, volume 517, pages 939–945, 2012. 2

[7] Yifang Liu, Jiwon Choi, and Nils Napp. Planning for Robotic Dry Stacking with Irregular Stones. *12th Conference on Field and Service Robotics (FSR19)*, 2019. 2, 5

[8] Yifang Liu, Maira Saboia, Vivek Thangavelu, and Nils Napp. Approximate stability analysis for drystacked structures. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2019-May, pages 8819–8824. Institute of Electrical and Electronics Engineers Inc., 5 2019. 2

[9] Yifang Liu, Seyed Mahdi Shamsi, Le Fang, Changyou Chen, and Nils Napp. Deep Q-Learning for Dry Stacking Irregular Objects. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1569–1576. Institute of Electrical and Electronics Engineers Inc., 12 2018. 2

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2 2015. 1, 2

[11] Stig Anton Nielsen and Alexandru Dancu. Fusing design and construction as speculative articulations for the built environment. In Ajla Aksamija, John Haymaker, and Abbas Aminmansour, editors, *Future of Architectural Research: Proceedings of the Architectural Research Centers Consortium Conference*, pages 65–73, Chicago, 2015. Perkins+Will. 2

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. Springer Verlag, 5 2015. 5

[13] Kurt R. Sacksteder and Gerald B. Sanders. In-Situ Resource Utilization for lunar and Mars exploration. In *Collection of Technical Papers - 45th AIAA Aerospace Sciences Meeting*, volume 6, pages 4232–4237, 2007. 1

[14] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In Francis Bach

and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 2015. PMLR. 3

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, second edition, 2018. 2, 3

[16] Madhu Thangavelu. Living on the Moon. In *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, Ltd, Chichester, UK, 12 2012. 1

[17] Vivek Thangavelu, Yifang Liu, Maira Saboia, and Nils Napp. Dry Stacking for Automated Construction with Irregular Objects. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4782–4789. Institute of Electrical and Electronics Engineers Inc., 9 2018. 2, 4

[18] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-September, pages 23–30. Institute of Electrical and Electronics Engineers Inc., 12 2017. 8

[19] Pedro Vicente, Lorenzo Jamone, and Alexandre Bernardino. Towards markerless visual servoing of grasping tasks for humanoid robots. In *IEEE International Conference on Robotics and Automation*, 2017. 8

[20] John Vivian. *Building Stone Walls*. Storey Publishing, second edition, 1976. 7, 8

[21] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando Freitas. Dueling Network Architectures for Deep Reinforcement Learning. In Maria Florina Balcan and Kilian Q Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 2016. PMLR. 5