Greedy Stone Tower Creations with a Robotic Arm

Martin Wermelinger∗,1, Fadri Furrer∗,2, Hironori Yoshida∗,3
Fabio Gramazio4, Matthias Kohler4, Roland Siegwart2 and Marco Hutter1
1Robotic Systems Lab, ETH Zurich, Switzerland
2Autonomous Systems Lab, ETH Zurich, Switzerland
3Preferred Networks, Inc,
4Gramazio Kohler Research, ETH Zurich, Switzerland
martin.wermelinger@mavt.ethz.ch, fadri.furrer@mavt.ethz.ch, hyoshida@hy-ma.com

Abstract

Predominately, robotic construction is applied as prefabrication in structured indoor environments with standard building materials. Our work, on the other hand, focuses on utilizing irregular materials found on-site, such as rubble and rocks, for autonomous construction. We present a pipeline to detect arbitrarily placed objects in a scene and form a structure out of the detected objects. The next best stacking pose is selected using a searching method employing gradient descent with random initial orientations, exploiting a physics engine. This approach is validated in an experimental setup using a robotic manipulator by constructing balancing vertical stacks without mortars and adhesives. We show the results of eleven consecutive trials to form such towers autonomously using four arbitrarily in front of the robot placed rocks.

1 Introduction and Related Work

Over the last decade, robotics has been introduced to architectural construction not only for safer and more efficient construction, but also for exploring diverse forms [Kohler et al., 2014]. However, there are still intensive manual labor works involved for on-site assembly of these components [Knaack et al., 2012].

Digital fabrication has explored applications of autonomous robots in unstructured on-site operation scenarios [Dörfle et al., 2016; Sandy et al., 2016], but are restricted to build with regular materials. Building structures with irregularly shaped objects requires structural analysis, e.g., through numerical analysis [Livesley, 1978; Livesley, 1992]. There exist tools for obvious contact surfaces as shown by [Block et al., 2006] and [Whiting et al., 2009]. However with irregularly shaped elements, we need to start from contact detection and then acquire contact surfaces. This can be done in a physics engine, such as Open Dynamics Engine (ODE), for evaluating structural stability [Battaglia et al., 2013]. In our work we also use the simulation environment but with irregular, concave shapes.

Our work focuses on handling objects of arbitrary shape, such as found irregularly shaped stones, without using additional adhesives to build dry-stack compositions. As a case study, discrete rigid elements, such as stones or concrete rubble, are targeted as a building material. The goal is to construct a balancing vertical tower with found objects, while maintaining the structure in static equilibrium using a robotic manipulator. The use of such objects reveals the following challenges. Firstly, individual object instances need to be identified. Secondly, grasping and stacking poses are not obvious, requiring a novel algorithm to pick a ‘good’ next pose among infinitely many. Thirdly, the stacking task may be performed in unstable situations, requiring recurring structural evaluation and target re-planning after each object placement.

To achieve this, we developed an iterative work-flow including precise object detection, motion control, and planning the next target pose (see Fig. 1). As part of this autonomous stacking work-flow, we describe an algorithm suggesting stable poses for stacking, validated by an implementation in a real-world experiment.

This paper makes the following contributions regarding handling irregularly shaped objects:

– a pose searching algorithm considering structural stability using a physics engine
– an object detection pipeline
– an autonomous system for constructing balancing vertical towers using a manipulator

An extended and comprehensive overview of this work is given by [Furrer et al., 2017].

2 Object Detection

Before starting with the stacking algorithm, we need to find the objects in the scene. Additionally, during the course of the object stacking, we want to be able to track the locations

∗The authors contributed equally to this work. M.W. for the manipulation tasks, F.F. was responsible for the object detection, H.Y. for the pose searching algorithm.
of the objects. In the scope of this work, we are only considering pre-scanned models of the objects to be detected in the scene. We present an object detection pipeline that consists of the following steps. We start by extracting 3D keypoints from raw point clouds of an RGB-D sensor. These keypoints are then described using keypoint descriptors and matched to keypoints from a pre-scanned object in a descriptor matching step. Using these matches and a clustering step, we find an initial alignment of the scene and the pre-scanned object, which is then refined by applying an Iterative Closest Point (ICP) algorithm. As a final step of the object detection pipeline, we verify that we have enough inlier points by applying the identified pose transform of the object to the scene.

2.1 Keypoint Extraction and Description

From an RGB-D sensor we get a scene point cloud $P_C$, in camera frame $C$. To get keypoints, we used two methods, a simple voxel based subsampling, as well as the Point Cloud Library (PCL) implementation of Intrinsic Shape Signatures (ISS) [Zhong, 2009], which can not only describe the keypoints, but also be used as a keypoint detector. Beside the ISS descriptor, we use Rotational Projection Statistics (RoPS) descriptors [Guo et al., 2013], which were giving us better results in the matching step, at a slightly higher computational cost.

2.2 Descriptor Matching and Clustering

We compare a keypoint $k_C$ of a scene point cloud with a keypoint of a point cloud of a pre-scanned object $k_O$ in object frame $O$. To find a pair of corresponding keypoints $k_C$ and $k_O$, we set up a kd-tree in descriptor space to find the nearest neighbors. Then we use an approach, presented in [Chen and Bhanu, 2004], to verify that the matched keypoints are geometrical consistent. We select the $b$ best transforms $T_{CO,j,matching}$, $j \in \{1, \ldots, b\}$ that give the most geometrical consistent matches. The transforms $T_{CO,j,matching}$ project the object point cloud $P_O$ into the camera frame $C$. Using an ICP step we refine these transforms $T_{CO,j,matching}$, to get better alignments of the two complete filtered point clouds $P_C,scene$ and $P_O,object$. The final transform $T_{CO}$ is obtained after checking the inlier ratio of the transformed point clouds.

2.3 Object in Robot Arm Frame

To transform the point cloud of the localized object $P_O,object$ into the robot arm frame $R$, we apply the previously detected best transform $T_{CO}$, a fixed pre-calibrated transform $T_{TC}$ from the camera frame $C$ to the robot arm tooltip frame $T$, and the transform given by the robot state $T_{RT}$ between the tooltip frame $T$ and the robot arm frame $R$:

$$P_{R,object} = T_{RT} \cdot T_{TC} \cdot T_{CO} \cdot P_{O,object}. \quad (1)$$

3 Pose Searching

Our goal is to construct a vertical tower consisting of irregularly shaped objects from a subset $S$ of available objects $o_i \in S \subseteq O$, where $O$ denotes the complete set of given objects. Within the set $S$, we want to find the best object and its target pose. The search space is twofold: discrete object space and continuous pose space. To find a stable pose on a vertical stack, our pose searching method places each object $o_i$ on the top object of the existing stack in a dynamic simulation using a physics engine. For evaluating each object’s ‘goodness’ with a certain pose $p_i$, we introduce a cost function that maximizes the support polygon $S_j$’s area $A_j$ of the newly placed object $o_i$ and minimizes other considerable parameters, such as kinetic energy. Throughout this process, several initial poses are tested with fixed initial positions but randomized orientations. We are sampling our initial orientations randomly, to keep the problem viable in large problem sets, where a holistic pose sampling would become intractable. The returned cost value is interchangeable among available objects in $S$, thus we find the best object $o^*$ with the best pose $p^*$. 

3.1 Valid Contact Pose Search

To find a valid contact pose, we set the object to an initial pose in simulation that is close to the existing stack, but not yet touching it. The initial pose $p_{init,i}$ of a new object $o_i \in O$ consists of its initial position $r_{init,i}$ and its initial orientation $e_{init,i}$. The initial position is set with an offset from the centroid $C_j$ of the last placed object’s support polygon $S_j$ along the normal direction $n_j$ of $S_j$ (see Fig. 2). To obtain the initial
orientation, the detected object orientation is rotated around a randomized axis with the random angle \( \theta \in [-\theta_{\text{init}}, \theta_{\text{init}}] \).

For evaluating physical stability of object \( o_i \), it is a valid approach to analyze whether \( P_t^i \), the projection of the CoM position \( P_t \) onto the support polygon \( S_i \), is inside the support polygon or not (see Fig. 3). In order to find a valid support polygon \( S_i \) for an irregularly shaped object \( o_i \), we consider the contact points of the object to other objects. The assumed contact situations are simple; either on a flat surface for the first stack, or contact points between two rigid body objects with parallel contact normals.

To assure that \( \mathbf{p}_{\text{init},i} \) results in a valid contact pose \( \mathbf{p}_{\text{contact},i} \), we apply an attraction force \( \mathbf{F} \) along the thrust line direction vector \( \mathbf{v}_i \) to the object \( o_i \) (see Fig. 2). During this process, we continuously check whether the projection \( P_t^i \) of the CoM position lies within the support polygon (see Fig. 3). As soon as the number of contacts \( N_{\text{contact}}(\mathbf{p}_t) \) between \( o_i \) and the existing stack is at least three (see Fig. 3), the resulting pose \( \mathbf{p}_{\text{contact},i} \) is evaluated by \( o_i \)'s kinetic energy \( E_{\text{kin}}(\mathbf{p}_{\text{contact},i}) \) with a threshold value \( E_{\text{kin,stable}} \). By evaluating the kinetic energy, we limit the viable set of poses to the ones that cause minimal motion of the existing stack.

### 3.2 Contact Pose Refinement

We assign a cost to each valid contact pose \( \mathbf{p}_{\text{contact},i} \) to compare its ‘goodness’ in terms of a robust object pose, which allows further stacking. Therefore, we maximize the area of the support polygon \( S_i \) as well as minimize the kinetic energy \( E_{\text{kin}} \), and surface normal deviation from the thrust line \( \mathbf{n}_i \) for reducing shear forces. To robustly find the support polygon \( S_i \) from the sparse contacts between \( o_i \) and the existing stack, contacts over several simulation update steps are collected and simplified [Alliez et al., 2016], [Karavelas, 2016]. Given the area \( A_i \) of the support polygon \( S_i \), the kinetic energy \( E_{\text{kin}}(\mathbf{p}_{\text{contact},i}) \), the dot product \( \|\mathbf{n}_i \cdot \mathbf{v}_i\| \), where \( \mathbf{v}_i \) is the thrust line direction vector, the length \( \|\mathbf{r}_{P_t,P_j}\| \) between \( P_t \) and the CoM of the previously stacked object \( P_j \), we define the cost function as

\[
f(\mathbf{p}_{\text{contact},i}) = w_1 A_i^{-1} + w_2 E_{\text{kin}}(\mathbf{p}_{\text{contact},i}) + w_3 \|\mathbf{r}_{P_t,P_j}\| + w_4 \|\mathbf{n}_i \cdot \mathbf{v}_i\|
\]

s.t. \( w_j \geq 0 \quad \forall j = 1, \ldots, 4 \)

where \( w_j \) are manually selected weights of the individual cost function components.

After assigning the cost to the valid contact pose \( \mathbf{p}_{\text{contact},i} \), gradient descent is performed for searching the local optimum pose \( \mathbf{p}_{\text{local},i} \) until the cost converges. To compute the gradient, the contact pose \( \mathbf{p}_{\text{contact},i} \) is perturbed with a small pose step \( \mathbf{\delta p} \).

After finding a local optimum pose \( \mathbf{p}_{\text{local},i} \), a new randomized rotation is assigned to the initial pose \( \mathbf{p}_{\text{init},i} \) and the process is repeated until \( x \) local solutions are found. The pose with minimum cost is selected as a solution \( \mathbf{p}_i^* \) for object \( o_i \). We iterate the entire process over all objects of the available subset \( S \), returning the best object \( o^* \) with the best pose \( \mathbf{p}_i^* \).

### 4 Experimental Setup

To show the applicability and repeatability of the presented pose searching and object detection methods, we implemented the algorithms, using a robot arm to perform autonomous dry-stacking as in Figure 1. The goal is to create a vertical tower out of randomly placed irregularly shaped objects whose mechanical and geometric properties are known.

#### 4.1 Experimental Setup

We use a set of six natural lime stones as objects because they have challenging properties for the stacking task, they are of irregular shape and have low friction coefficients. The objects’ geometric shape were previously scanned and the point cloud and mesh model were scaled to a reasonable resolution of 500 mesh faces. The weight, CoM position, and moment of inertia of each stone were measured and added to the geometric model description. The friction coefficient was estimated with a low value of \( \mu_{\text{stone}} = 0.1 \) and uniformly applied to all stones. For manipulating the objects, we use a robotic arm equipped with a three-finger gripper as depicted in Figure 4. The object detection is performed with an RGB-D depth camera mounted on the robot arm. A force-torque sensor mounted at the attaching point of the gripper is used to detect impact during the placement of the object.
The work-flow of autonomously creating a vertical stack of arbitrarily placed stones is shown in Figure 1. This task is performed by continuously executing a loop consisting of object detection, pose searching and object manipulation. The stacking task is terminated once the pose searching no longer finds a feasible solution from the available objects or the stacking was not successful.

### 4.2 Results

The robotic system performed the vertical stacking task in eleven consecutive runs with an alternating set of four stones\(^1\). In two of these runs, the system succeeded to construct a stack out of all four available stones. In six cases the system was able to vertically stack three stones, but failed to place the fourth stone, and in three cases the system did not succeed to stack the third stone. The average cost where the robot failed to place the object was 2.5 times higher than the average cost of the last pose the system was able to successfully stack. This shows that these poses were already identified as less favourable compared to the ones that were successfully stacked. If the cost at a previous step is high, the probability increases that the following stone placement will fail. See for example the high cost of the second stone in run 6 and 7 (see Figure 5).

On average, a trial to construct a vertical stack lasted 271.0 s, where the main fraction of this time (61%) is spent for the manipulation task that includes path planning, arm and gripper motion.

### 5 Conclusion

In this paper, we introduced an autonomous robotic system that constructs a balancing vertical tower out of irregularly shaped stones without using extra materials. Its work-flow consists of a continuous loop with object detection, target pose search, physical manipulation, and evaluation. We presented an object detection pipeline suited to localize irregularly shaped objects in a scene and a target pose searching algorithm, based on a physics engine, to generate stable stacks. The proposed algorithms were implemented and tested on a robotic system (a fixed platform in a controlled environment with a flat terrain, and pre-scanned objects). The system showed to be able to perform stacking tasks autonomously and thus validating the proposed pose searching algorithm.

Aiming at more practical situations, we want to focus on construction with unseen objects. This involves the segmentation of unknown objects in a scene and their handling with incomplete information. Solving this can either be achieved by creating object models in the construction process or by applying techniques where we do not require a specific representation of the object, as shown in [Breyer et al., 2018] for grasping objects. We plan to extend this work to place objects including force-torque sensor readings. Furthermore, we want to create more complex target shapes, such as arches or walls.

### Acknowledgements

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab).

### References


---

\(^1\)Watch the accompanying video: https://www.youtube.com/watch?v=bXz52KMGrUng


