Autonomous robotic stone stacking with online next best object target pose planning

Autonomous Robotic Stone Stacking with Online next Best Object Target Pose Planning

Fadri Furrer*,[1], Martin Wermelinger*,[2], Hironori Yoshida*
Fabio Gramazio[3], Matthias Kohler[3], Roland Siegwart[1], Marco Hutter[2]

***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 that detects randomly placed objects in a scene that are used by our next best stacking pose searching method employing gradient descent with a random initial orientation, 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.**

I. Introduction

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 [1]. However, there are still intensive manual labor works involved for on-site assembly of these components [2].

Digital fabrication has explored applications of autonomous robots in on-site operation scenarios [3], but are restricted to build with regular materials. Building structures with irregular shaped objects was presented in [4], however they apply glue to increase stability. To reduce the environmental impact we aim to use such material without additional adhesives, to build dry-stack compositions. Therefore, our work focuses on developing an automated fabrication process using irregular objects, which are not processed but found on-site.

As a case study, discrete rigid elements, such as stones or concrete rubble, are targeted as a building material. Our goal is to construct a balancing vertical tower with found objects, while maintaining the structure in static equilibrium using a robotic manipulator. To achieve this, we developed a holistic work-flow including precise object detection, motion control, and planning the next target pose (see Figure 1). As part of this work-flow, we describe an algorithm suggesting stable poses for stacking, validated by an implementation of this autonomous stacking work-flow in a real-world experiment. Due to the instability of vertical tower, it is natural to observe errors between a desired target pose and an actual stacked pose. Thus, the work-flow puts emphasis on the resultant pose evaluation and a dynamic re-planning of the target pose.

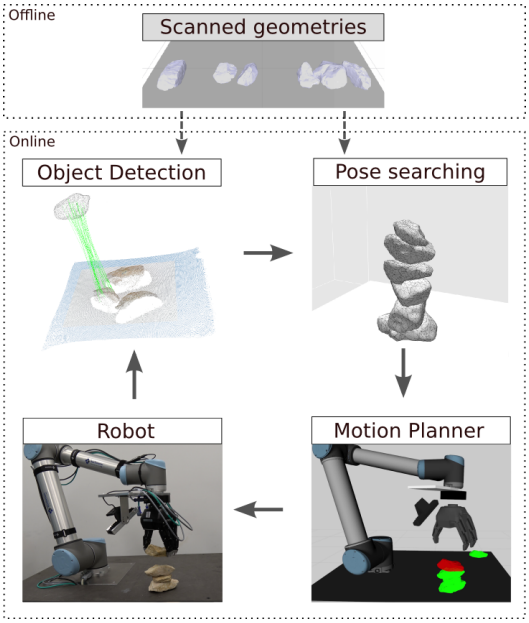

Fig. 1: In an offline step we scan a set of objects (top). These objects, or a subset of it, can be distributed arbitrarily on the work-space and get detected by our object detection pipeline (middle-left). From the detected objects the presented pose searching algorithm proposes the next stable stack (middle-right). A motion planner (bottom-right) is used to generate the trajectories to replicate the proposed stack with the robot arm (bottom-left). After placing the object, its pose is measured and used as base for the subsequent pose searching step.

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

II. Related Work

Research in architecture and digital fabrication investigates novel production techniques in which material behavior is linked to fabrication and assembly tasks [5]. Recently, it has

[1]Autonomous Systems Lab (ASL), ETH Zurich, Switzerland
[2]Robotic Systems Lab (RSL), ETH Zurich, Switzerland
[3]Gramazio Kohler Research (GKR), ETH Zurich, Switzerland
*The authors contributed equally to this work. F.F. was responsible for the object detection, M.W. for the manipulation tasks, H.Y. for the pose searching algorithm.

This work was supported in part by the Swiss National Science Foundation (SNF), the National Centre of Competence in Research Digital Fabrication.

been shown that robots bring new capabilities to construction sites, an uncontrolled environment full of uncertainties [3]. Whereas some works, like [6], show how to localize a mobile robot in such an environment, our work focuses on handling building materials of arbitrary shape, such as found irregularly shaped stones. 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; for example vertical tower stacking in our case, requiring recurring structural evaluation and target re-planning after each object placement.

Computational structural analysis methods have been explored with rigid discrete elements, such as simple brick-like geometries. Livesley [7], [8] set a basis of a numerical method for limit analysis of discrete rigid block structures. Block and colleagues employed graphical statics in interactive design tools with structural analysis feedback [9], and Whiting et al. [10] extended the limit analysis by Livesley to design a guidance system by adding infeasibility metrics. While these works analyzed a static equilibrium of given geometric configuration with obvious geometric contact surfaces (or support polygons), in our case with irregularly shaped elements, we need to start from contact detection, which is a core function of physics engines, and then acquire a contact surface.

From an architectural design motivation, simulation with physics engine has been explored by Nielsen et al. [11] but their construction input was to place the next stone at the lowest possible location. As for design with irregularly shaped objects, Lambert and Kennedy developed an application for guiding masonry construction with a geometric packing algorithm in two dimensional convex shapes, but it is limited to two dimensional tiling [12].

Humans synthesize the Center of Mass (CoM) position and the support polygon to infer the stability of objects [13]. On the other hand, physics engines, such as Open Dynamics Engine (ODE), have been used for evaluating structural stability of object compositions [14]. Similarly, our algorithm employs a physics engine to extract the CoM position and dynamic characteristics, as well as the contact points between the irregularly shaped objects.

Integrated autonomous systems dealing with object detection and picking these objects with robotic arms have been proposed for industrial applications [15], [16]. These works have successfully detected irregularly shaped stone-like objects for geometric packing, but are limited to place these objects in containers, and hence, did not need to consider structural stability.

III. 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 of the objects. In the scope of this work, we are only considering pre-scanned (the scanning method is described in Section V) models of the objects to be detected in the scene. Therefore, 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 *clustring* 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.

Keypoint Extraction and Description

From an RGB-D sensor we get a scene point cloud $\mathbf{P}_{\mathcal{C}}$, in camera frame $\mathcal{C}$. 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) [17], which can not only describe the keypoints, but also used as a keypoint detector. Besides the ISS descriptor we tested two additional descriptors, namely the Fast Point Feature Histogram (FPFH) descriptor [18] and the Rotational Projection Statistics (RoPS) descriptors [19]. Here, the RoPS descriptors were giving us the best results in the matching step, at a slightly higher computational cost.

Descriptor Matching and Clustering

We compare a keypoint $\mathbf{k}_{\mathcal{C},\text{scene}}$ of a scene point cloud with a keypoint of a point cloud of a pre-scanned object $\mathbf{k}_{\mathcal{O},\text{object}}$ in object frame $\mathcal{O}$. To find a pair of corresponding keypoints $\mathbf{k}_{\mathcal{C},\text{scene}}$ and $\mathbf{k}_{\mathcal{O},\text{object}}$, we set up a kd-tree in descriptor space to find the nearest neighbors. Then we use an approach, presented in [20], to verify that the matched keypoints are geometrical consistent. We select the b best transforms $T_{\mathcal{CO},j,\text{matching}}, \quad j \in \{1,\dots,b\}$ that give the most geometrical consistent matches. The transforms $T_{\mathcal{CO},j,\text{matching}}$ project the object point cloud $\mathbf{P}_{\mathcal{O},\text{object}}$ into the camera frame $\mathcal{C}$.

Transform Refinement and Verification

Using an ICP step we refine these transforms $T_{\mathcal{CO},j,\text{matching}}$, to get better alignments of the two complete filtered point clouds $\mathbf{P}_{\mathcal{C},\text{scene}}$ and $\mathbf{P}_{\mathcal{O},\text{object}}$. We denote these refined transforms by $T_{\mathcal{CO},j,\text{refined}}$. In a last step, we check for the inlier ratios of the transformed point clouds and we then select the one which has the highest ICP-score, given that we have an inlier ratio of the model points larger than a threshold value; in our application we set this threshold value to 20 % resulting in the final transform $T_{\mathcal{CO}}$.

Object in Robot Arm Frame

To transform the point cloud of the localized object $\mathbf{P}_{\mathcal{O},\text{object}}$ into the robot arm frame $\mathcal{R}$, we apply the previously detected best transform $T_{\mathcal{CO}}$, a fixed pre-calibrated transform $T_{\mathcal{TC}}$ from the camera frame $\mathcal{C}$ to the robot arm tooltip frame

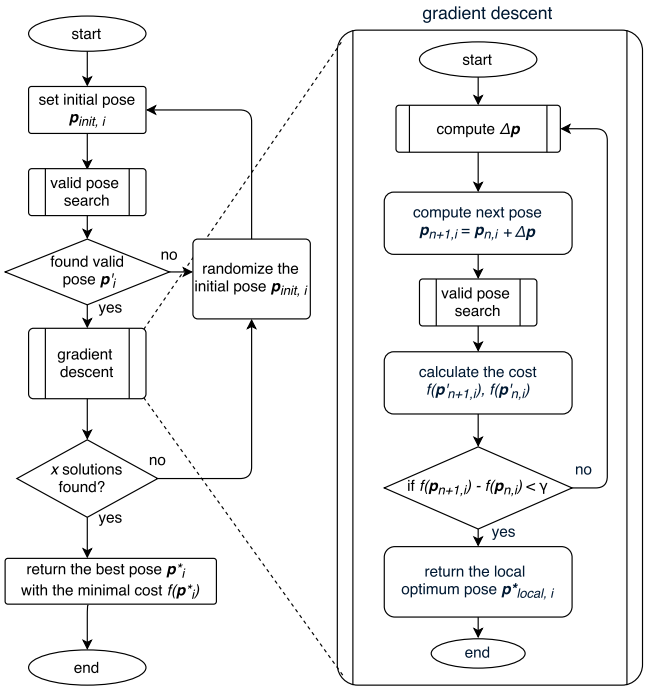

Fig. 2: Illustration of our pose searching algorithm to find the best pose for object o_i. The valid pose search sub-routine is described in Algorithm 1 and the cost calculation in Algorithm 2. The gradient descent sub-routine is further depicted on the right.

$\mathcal{T}$, and the transform given by the robot state $T_{\mathcal{RT}}$ between the robot arm frame $\mathcal{R}$ and the tooltip frame $\mathcal{T}$:

$$\mathbf{P}_{\mathcal{R},\text{object}} = T_{\mathcal{RT}} \cdot T_{\mathcal{TC}} \cdot T_{\mathcal{CO}} \cdot \mathbf{P}_{\mathcal{O},\text{object}}. \quad (1)$$

IV. Pose Searching

The global goal is to construct a vertical tower consisting of irregularly shaped objects from a subset $\mathcal{S}$ of available objects $o_i \in \mathcal{S} \subseteq \mathcal{O}$, where $\mathcal{O}$ denotes the complete set of given objects. Within the set $\mathcal{S}$, 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 $\mathbf{p}_i$, we introduce a cost function that maximizes the support polygon S_i's area A_i 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 $\mathcal{S}$, thus we find the best object o^* with the best pose $\mathbf{p}^*$.

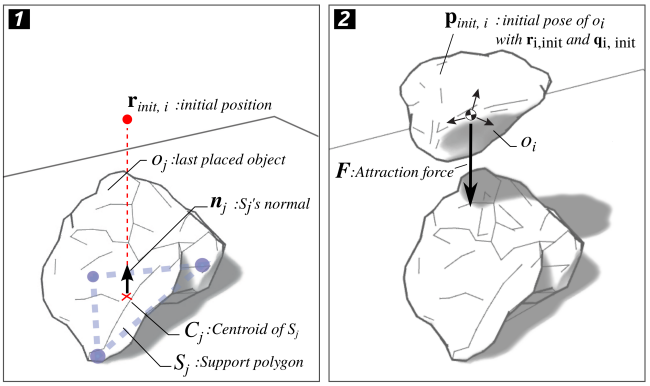

Fig. 3: (1) The initial position $\mathbf{r}_{\text{init},i}$ of object o_i is set along the normal direction n_j of S_j of the previously placed object o_j. (2) The initial pose of object o_i with attraction force $\mathbf{F}$.

Overview of the algorithm

The pose searching algorithm iteratively evaluates poses with valid contacts (support polygon) between a newly placed object o_i and the existing stack. Once a pose with valid contacts is found, the algorithm refines the pose with gradient descent. 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 $\mathbf{p}_{\text{init},i}$ of a new object $o_i \in \mathcal{O}$ consists of its initial position $\mathbf{r}_{\text{init},i}$ and its initial orientation $\mathbf{q}_{\text{init},i}$. 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 $\mathbf{n}_j$ of S_j (see Figure 3). To obtain the initial orientation, the detected object orientation is rotated around a randomized axis with the random angle $\theta \in [-\theta_{\text{init}}, \theta_{\text{init}}]$.

Based on the initial pose $\mathbf{p}_{\text{init},i}$, the valid contact pose $\mathbf{p}_{\text{contact},i}$ is found by a sub-routine named *valid pose search* by applying an attraction force $\mathbf{F}$ parallel to a thrust line input (in our case along the gravitational axis). We then run gradient descent (see the right in Figure 2) to iteratively improve the contact pose $\mathbf{p}_{\text{contact},i}$ using the cost function presented below. After the local optimum pose $\mathbf{p}^*_{\text{local},i}$ is found, the orientation of the initial pose $\mathbf{p}_{\text{init},i}$ is randomized again to find the next valid contact pose $\mathbf{p}_{\text{contact},i}$. After computing certain number x of local minima, we find the pose $\mathbf{p}^*_i$ with the lowest cost as the solution pose of o_i. We iterate the process over the available subset $\mathcal{S}$ to find the best object o^*.

To avoid jittering effect in physics simulation, the whole algorithm performs physics engine update steps only when it is necessary. Objects in an existing stack are set to be immobile in the whole process, except when the cost is calculated.

Valid pose search

For evaluating physical stability of object o_i, it is a valid approach to analyze whether P'_i, the projection of the CoM position P_i onto the support polygon S_i, is inside the support polygon or not (see Figure 4). In order to find a valid support

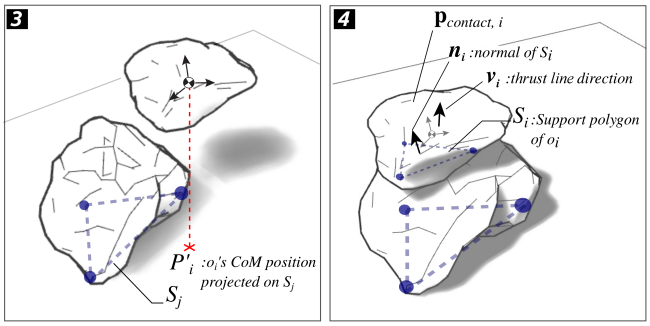

Fig. 4: (3): Projection P_i' of the CoM position P_i onto the support polygon S_j. (4): Contact pose $\mathbf{p}_{\text{contact},i}$ resulting from the valid pose search algorithm.

Algorithm 1 Valid pose search algorithm

function VALIDPOSESEARCH($o_i, \mathbf{p}_{\text{init},i}$)
  set o_i's pose to $\mathbf{p}_{\text{init},i}$
  do
    apply force $\mathbf{F}$ *to* o_i
    step physics simulation once
    if $P_i' \notin S_j$ **then return** false
  while $N_{\text{contact}}(\mathbf{p}_i) < 3$
  pause physics simulation
  $\mathbf{p}_{\text{contact},i} \leftarrow$ *current pose*
  if $E_{\text{kin}}(\mathbf{p}_{\text{contact},i}) < E_{\text{kin,stable}}$ **then return** true
  else return false

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 collision between two rigid body objects with parallel contact normals.

The valid pose search method is detailed in Algorithm 1. 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 Figure 3). During this process, we continuously check whether the projection P_i' of the CoM position lies within the support polygon (see Figure 4). As soon as the number of contacts $N_{\text{contact}}(\mathbf{p}_i)$ between o_i and the existing stack is at least three (see Figure 4), 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. This approach for finding a valid contact pose $\mathbf{p}_{\text{contact},i}$ guarantees to satisfy the following constraints:

$$\begin{aligned} E_{\text{kin}}(\mathbf{p}_{\text{contact},i}) &\leq E_{\text{kin,stable}} \\ P_i' &\in S_j \\ N_{\text{contact}}(\mathbf{p}_{\text{contact},i}) &\geq 3. \end{aligned} \tag{2}$$

Cost calculation

We assign a cost to each valid contact pose $\mathbf{p}_{\text{contact},i}$ to compare its 'goodness' in terms of a robust object poses, which allows further stacking. Therefore, we maximize the area of the support polygon S_i as well as minimize other considerable parameters, such as kinetic energy E_{kin}, and surface normal deviation from the thrust line $\mathbf{n}_i$ for reducing sheer 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, in our case 10 steps, are collected and simplified [21]. After acquiring 3D point sets, Principal Component Analysis (PCA) is performed for dimension reduction from 3D to 2D point set. Processing the 2D point set with Delaunay triangulation [22], the polygon mesh S_i is created for calculating its area A_i and surface normal $\mathbf{n}_{c,i}$ (see Figure 4).

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 $\|\mathbf{r}_{P_jP_i}\|$ between P_i and the CoM of the previously stacked object P_j, we define the cost function as

$$\begin{aligned} f(\mathbf{p}_{\text{contact},i}) = w_1 A_i^{-1} + w_2 E_{\text{kin}}(\mathbf{p}_{\text{contact},i}) \\ + w_3 \|\mathbf{r}_{P_jP_i}\| + w_4 \|\mathbf{n}_i \cdot \mathbf{v}_i\|, \\ \text{s.t. } w_j \geq 0 \; \forall j \in 1, \ldots, 4 \end{aligned} \tag{3}$$

where w_j are manually selected weights of the individual energy function components. An overview of the cost calculation algorithm can be seen in Algorithm 2.

Algorithm 2 Cost calculation

function CALCULATECOST($o_i, \mathbf{p}_{\text{contact},i}$)
  set all objects in stack mobile
  contactsArray[] $\leftarrow 0$
  for $k \leftarrow 0,\ k < 10$ **do**
    step physics simulation once
    contactsArray[] $\leftarrow$ *current contacts set*
  contacts $\leftarrow$ *simplify*(contactsArray[])
  contactPlane $\leftarrow$ *PCA*(contacts)
  contacts $\leftarrow$ *projection*(contacts,contactPlane)
  $S_i \leftarrow$ *2D DelaunayTriangulation*(contacts)
  get $A_i(S_i),\ E_{\text{kin}}(\mathbf{p}_{\text{contact},i}),\ \|\mathbf{r}_{P_jP_i}\|,\ \|\mathbf{n}_i \cdot \mathbf{v}_i\|$
  reset poses of all objects in stack
  set all objects in stack immobile
  cost $\leftarrow f(\mathbf{p}_{\text{contact},i})$
return cost

After assigning the cost to the valid contact pose $\mathbf{p}_{\text{contact},i}$, gradient descent is performed for searching the local optimum pose $\mathbf{p}^*_{\text{local},i}$, as depicted in the right of Figure 2. In this process, we iteratively calculate a small pose step $\Delta\mathbf{p}$ consisting of $\delta\hat{\mathbf{r}}$ and $\delta\hat{\mathbf{q}}$, as given in Eq. (5) and Eq. (6), to obtain an updated initial pose for valid contact pose searching:

$$\begin{aligned} \mathbf{p}_{\text{local init},i}[n+1] = \mathbf{p}_{\text{local contact},i}[n] + \Delta\mathbf{p} \\ \text{s.t. } \Delta\mathbf{p} = (\delta\hat{\mathbf{r}}, \delta\hat{\mathbf{q}}) \end{aligned}, \tag{4}$$

where $\mathbf{p}_{\text{local contact},i}[0] = \mathbf{p}_{\text{contact},i}$. The translation $\delta\hat{\mathbf{r}}$ assigns a positive constant offset z_{const} in contact normal direction to avoid placing the object o_i at invalid poses overlapping

with the existing stack. Given ϵ_r as a small value, $\mathbf{T_i}$ as a homogeneous transformation matrix from world frame to the thrust line aligned frame at C_j (see Figure 3), and z_{const}, we obtain $\delta\hat{\mathbf{r}}$ as:

$$\begin{aligned} \delta\mathbf{r} &= \mathbf{T_i}\left(\frac{\partial f}{\partial r_x}^{-1}, \frac{\partial f}{\partial r_y}^{-1}, z_{\text{const}}\right), \\ \text{and } \delta\hat{\mathbf{r}} &= \epsilon_r \frac{\delta\mathbf{r}}{\|\delta\mathbf{r}\|}, \quad \text{s.t. } z_{\text{const}} \geq 0. \end{aligned} \tag{5}$$

For rotation, we describe the orientation with a quaternion $\mathbf{q}$ in axis-angle representation as (axis, angle). Given ϵ_q as small rotation angle, and $\mathbf{T_i}$, we obtain $\delta\hat{\mathbf{q}}$ as:

$$\begin{aligned} \mathbf{v}_{\text{axis}} &= \mathbf{T_i}\left(\frac{\partial f}{\partial q_x}^{-1}, \frac{\partial f}{\partial q_y}^{-1}, \frac{\partial f}{\partial q_z}^{-1}\right) \\ \text{and } \delta\hat{\mathbf{q}} &= (\mathbf{v}_{\text{axis}}, \epsilon_q) \end{aligned} \tag{6}$$

For minimization of the cost, we iterate the process described in Eq. (4) until the difference of returned cost becomes smaller than the threshold γ as $f(\mathbf{p}_{\text{local contact},i}[n]) - f(\mathbf{p}_{\text{local contact},i}[n+1]) < \gamma$. We write the optimization process as,

$$\begin{aligned} \mathbf{p}^*_{\text{local},i} &= \underset{\mathbf{p}_{\text{local contact},\,i}}{\text{argmin}}\ f(\mathbf{p}_{\text{local contact},i}) \\ \text{s.t. } A_i(S_i) &\geq A_{\text{min}}, \end{aligned} \tag{7}$$

where A_{min} is the minimal support polygon area.

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, as shown in the left of Figure 2. 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 $\mathcal{S}$, returning the best object o^* with the best pose $\mathbf{p}^*$.

V. Experimental Setup

To show the applicability and repeatability of the presented pose searching and object detection methods, we implemented the algorithms, using Robotic Operating System (ROS) [23], on a robotic platform 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.

A. Experimental Setup

We use a set of six natural lime stones (Figure 5) as objects because they show challenging properties for the stacking task like irregular shape and low friction coefficient. The point cloud and mesh model of the object's geometric shape were previously acquired with an ATOS Core high precision scanner. These models are used for detecting the objects in the scene, as well as for simulating in a physics engine used in the pose searching algorithm. To lower the computation cost, we reduced the mesh model for the pose searching algorithm to a homogeneously triangulated mesh with 500 faces. This showed to be a reasonable trade-off between reducing the computation cost and being able to generate a contact situation that correlates with the real world. 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 $\mu_{\text{stone}} = 0.1$. For manipulating the objects, we use a robotic arm equipped with a three-finger gripper as depicted in Figure 6. The object detection is performed with an RGB-D depth camera mounted on the robot arm. The size of the objects are selected to fit in the finger stroke of the gripper. We are using MoveIt! [24] to generate collision free motions of the robot. A force-torque sensor mounted at the attaching point of the gripper is used to detect impact during the placing 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.

Fig. 5: Lime stones with irregular shape are used to create vertical stacks.

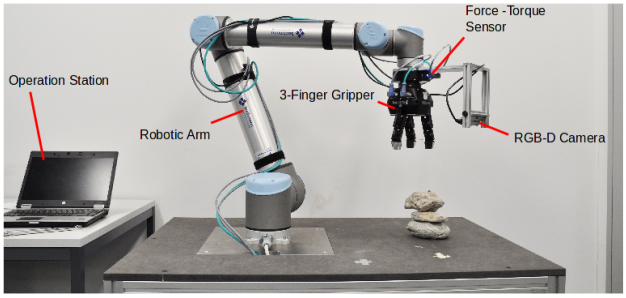

Fig. 6: An overview of the used hardware setup: a ROBOTIQ 3-finger gripper, a FT150 force torque sensor, and a Intel®RealSense™ SR300 RGB-D camera are attached to a UR10 arm.

TABLE I: Parameters

Parameter	Value	Parameter	Value
w_1	0.179	A_{min}	$1e^{-5}\,\text{m}^2$
w_2	0.472	F	100 N
w_3	0.094	$E_{\text{kin,stable}}$	20 J
w_4	0.255	θ_{init}	$\frac{\pi}{4}$ rad
x	5		

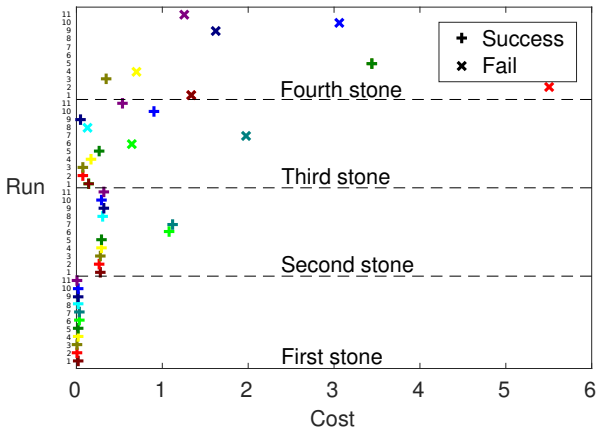

Fig. 7: The cost of the selected target pose of an individual stone for all eleven runs at each level of the stack. A higher cost indicates a less preferable target pose. '+' denotes a successful stacking, failed attempts are represented with a 'x'. Each color corresponds to an individual run.

First, the objects are detected and localized in the scene, resulting in a set of available stones $\mathcal{S}$. For each trial we used an alternating subset of four stones from the complete set of the six lime stones. From this set $\mathcal{S}$, the presented pose searching algorithm proposes the next stable stack. To replicate the proposed stack, a collision free grasping configuration from a predefined set of feasible configurations is chosen and a motion planner generates executable trajectories. After placing the stone at the proposed pose, we detect the updated pose and validate if the stacking was successful. The updated stone pose is used as a foundation for the next pose searching step. If the robot could successfully execute the proposed stack, the next proposed stable stack is computed from the remaining set of stones. The stacking task is terminated once the pose searching does no longer find a feasible solution from the available objects or the stacking was not successful.

B. 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. For the pose searching algorithm presented in Section IV, we used the parameters given in Table I. The average cost of the last pose the system was able to successfully stack was 0.7425. Whereas, in the case where the robot failed to place the object the average cost was 1.8018, which 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 7).

[1]Watch the accompanying video: `https://www.youtube.com/watch?v=bXz52KMGUng`

TABLE II: Mean execution times.

Task	Mean time (s)	σ time (s)	Fraction (%)
Pose search	66.2	7.0	24.4
Object detection	17.0	4.4	6.3
Manipulation	166.7	17.2	61.5
Other tasks	21.1	4.1	8.8

Table II shows the mean execution times of the different parts of the presented approach, although computational and execution speed was not a focus of this paper. On average, a trial to construct a vertical stack lasted 271.0 s. The main fraction of this time is spent for the manipulation task that includes path planning, arm and gripper motion. This is mainly due to the fact that the robot, for safety reasons, is operated with reduced speed. The remaining time is spent for the pose searching algorithm itself, object detection and other tasks, such as visualization and stopping the motion before capturing depth images. The stacking work-flow is not yet optimized in terms of construction time and could be greatly improved by parallelizing manipulation with pose searching and object detection, and by increasing the operation speed.

VI. Conclusion

In this paper, we introduced an autonomous robotic system that constructs a balancing vertical tower out of irregularly shaped stones without using mortars or 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 to generate stable stacks. The proposed algorithms were implemented on a robotic system and tested in an experimental setup (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, contributing to a preliminary setup for detecting irregularly shaped objects and also validating the proposed pose searching algorithm.

As a next step to improve the stability of the planned stack in this setup, the pose searching could consider the future cost by simulating several steps ahead. 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.

Furthermore, we aim at creating more complex target shapes, such as arches or walls. These shapes can create structures where each object is in contact with more than two objects. In such a situations, not just a vertical or single thrust line, but a more complex thrust line network needs to be analyzed.

VII. Acknowledgement

The authors would like to thank Mirjan Ammar for his feedback throughout the project, Jemin Hwangbo for discussions on the optimization methods, and Luca Forni and Yves Zimmermann for their great work in setting up the demonstrator.

REFERENCES

[1] M. Kohler, F. Gramazio, and J. Willmann, *The Robotic Touch: How Robots Change Architecture*. Park Books, 2014.

[2] U. Knaack, S. Chung-Klatte, and R. Hasselbach, *Prefabricated systems: Principles of construction*. Walter de Gruyter, 2012.

[3] K. Dörfler, T. Sandy, M. Giftthaler, F. Gramazio, M. Kohler, and J. Buchli, *Mobile Robotic Brickwork*. Cham: Springer International Publishing, 2016, pp. 204–217.

[4] N. Napp and R. Nagpal, "Distributed amorphous ramp construction in unstructured environments," *Robotica*, vol. 32, no. 02, pp. 279–290, 2014.

[5] C. Feng, Y. Xiao, A. Willette, W. McGee, and V. Kamat, "Towards autonomous robotic in-situ assembly on unstructured construction sites using monocular vision," in *Proceedings of the 31th International Symposium on Automation and Robotics in Construction*, 2014, pp. 163–170.

[6] T. Sandy, M. Giftthaler, K. Dörfler, M. Kohler, and J. Buchli, "Autonomous repositioning and localization of an in situ fabricator," 2016.

[7] R. K. Livesley, "Limit analysis of structures formed from rigid blocks," *International Journal for Numerical Methods in Engineering*, vol. 12, no. 12, pp. 1853–1871, 1978.

[8] R. Livesley, "A computational model for the limit analysis of three-dimensional masonry structures," *Meccanica*, vol. 27, no. 3, pp. 161–172, 1992.

[9] P. Block, T. Ciblac, and J. Ochsendorf, "Real-time limit analysis of vaulted masonry buildings," *Computers & structures*, vol. 84, no. 29, pp. 1841–1852, 2006.

[10] E. Whiting, J. Ochsendorf, and F. Durand, "Procedural modeling of structurally-sound masonry buildings," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 112, 2009.

[11] S. A. Nielsen and A. Dancu, "Fusing design and construction as speculative articulations for the built environment," 2015.

[12] M. Lambert and P. Kennedy, "Using artificial intelligence to build with unprocessed rock," in *Key Engineering Materials*, vol. 517. Trans Tech Publ, 2012, pp. 939–945.

[13] S. A. Cholewiak, R. W. Fleming, and M. Singh, "Perception of physical stability and center of mass of 3-d objects," *Journal of vision*, vol. 15, no. 2, pp. 13–13, 2015.

[14] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013.

[15] M.-C. Ko, "Algorithms and automated material handling systems design for stacking 3d irregular stone pieces," Ph.D. dissertation, 2011.

[16] V. Sujan, S. Dubowsky, Y. Ohkami *et al.*, "Design and implementation of a robot assisted crucible charging system," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 1969–1975.

[17] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *ICCV Workshops*, 2009.

[18] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *ICRA*, 2009.

[19] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3D local surface description and object recognition," *International Journal of Computer Vision*, vol. 105, no. 1, pp. 63–86, 2013.

[20] H. Chen and B. Bhanu, "3D free-form object recognition in range images using local surface patches," *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, pp. 136–139 Vol.3, 2004.

[21] P. Alliez, C. Jamin, Q. Mérigot, J. Meyron, L. Saboret, N. Salman, and S. Wu, "Point set processing," in *CGAL User and Reference Manual*, 4.8.1 ed. CGAL Editorial Board, 2016. [Online]. Available: http://doc.cgal.org/4.8.1/Manual/packages.html#PkgPointSetProcessingSummary

[22] M. Karavelas, "2D segment delaunay graphs," in *CGAL User and Reference Manual*, 4.8.1 ed. CGAL Editorial Board, 2016. [Online]. Available: http://doc.cgal.org/4.8.1/Manual/packages.html#PkgSegmentDelaunayGraph2Summary

[23] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[24] I. A. Sucan and S. Chitta, "Moveit!" *Online Available: http://moveit.ros.org*, 2013.