Towards Automating Construction Tasks: Large-Scale Object Mapping, Segmentation and Manipulation

Author(s):
Mascaro, Ruben; Wermelinger, Martin; Hutter, Marco; Chli, Margarita

Publication date:
2020-08

Permanent link:
https://doi.org/10.3929/ethz-b-000457938

Originally published in:
Journal of Field Robotics 38(5), https://doi.org/10.1002/rob.22007

Towards Automating Construction Tasks: Large-Scale Object Mapping, Segmentation and Manipulation

Ruben Mascaro*
Vision for Robotics Lab (V4RL)
ETH Zurich
Zurich, Switzerland
rmascaro@ethz.ch

Martin Wermelinger*
Robotic Systems Lab (RSL)
ETH Zurich
Zurich, Switzerland
martiwer@ethz.ch

Marco Hutter
Robotic Systems Lab (RSL)
ETH Zurich
Zurich, Switzerland
mahutter@ethz.ch

Margarita Chli
Vision for Robotics Lab (V4RL)
ETH Zurich
Zurich, Switzerland
chlim@ethz.ch

Abstract

Automating building processes through robotic systems has the potential to address the need for safer, more efficient and sustainable construction operations. While ongoing research effort often targets the use of prefabricated materials in controlled environments, here we focus on utilizing objects found on-site, such as irregularly-shaped rocks and rubble, as a way of enabling novel types of construction in remote and extreme environments, where standard building materials might not be easily accessible. In this article, we present a perception and grasp pose planning pipeline for autonomous manipulation of objects of interest with a robotic walking excavator. The system incrementally builds a LiDAR-based map of the robot's surroundings and provides the ability to register externally reconstructed point clouds of the scene, e.g. from images captured by a drone-borne camera, which helps increasing map coverage. In addition, object-like instances, such as stones, are segmented out of this map. Based on this information, collision-free grasping poses for the robotic manipulator are planned to enable picking and placing of these objects, while keeping track of them during the manipulation. The approach is validated in a real setting on an architectural relevant scale by segmenting and manipulating boulders of several hundred kilograms, which is a first step towards the full automation of dry-stack wall building processes.

Video – https://youtu.be/4bc5n2-zj3Q

1 INTRODUCTION

In recent years, automating construction through robotics has seen increased popularity as it promises more efficient, more sustainable, and safer building operations (Petersen et al., 2019; Ardiny et al., 2015). Furthermore, it pushes forward the digitalization of construction processes, enabling the implementation of new design principles and the creation of novel types of architectural structures with unprecedented

*The authors contributed equally to this work. R.M. was responsible for the perception pipeline and M.W. for the manipulation tasks.

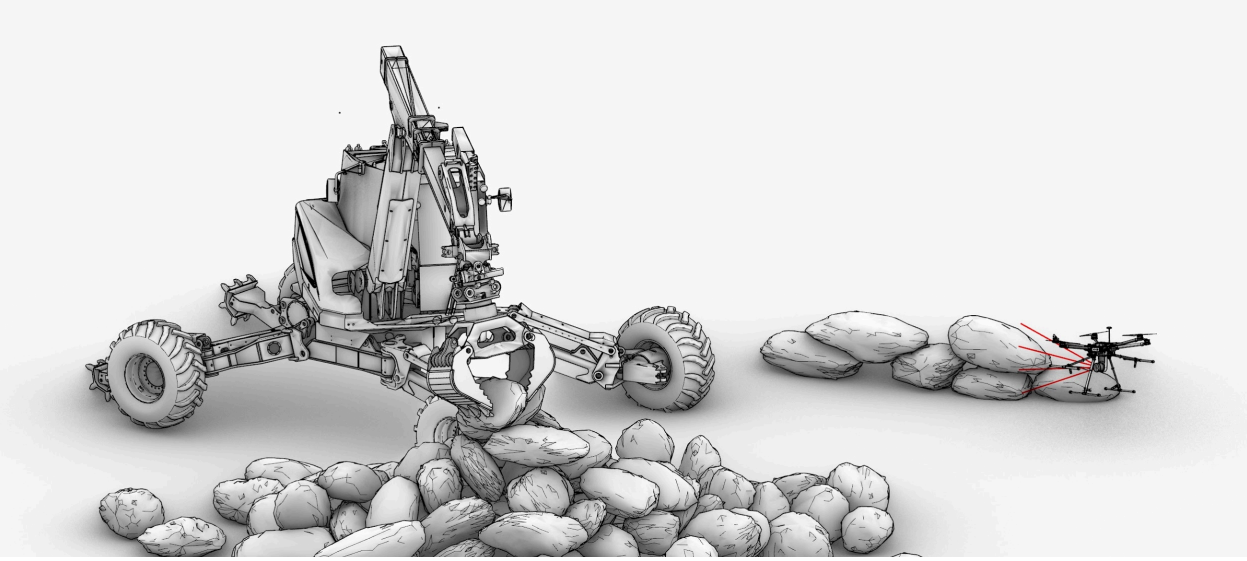


Figure 1: Potential application of the proposed system: robotic construction of utility structures with material found on-site. By fusing mapping data acquired from the excavator and a drone, the observed stones are segmented in the acquired map, while collision-free paths for the manipulator are computed and executed to move the stones of interest to the desired locations in the vicinity of the robot.

complexity and functional properties. Of special interest is the direct utilization of raw construction material found on-site without further processing, such as natural stones or rubble, since it can significantly increase flexibility of construction and enable building at potentially dangerous or difficult-to-access places, while helping to reduce material waste (Eversmann, 2018). This is particularly relevant for the task of building utility structures (e.g. noise protection walls, river bank reinforcements or protection barriers), which are usually constituted by great quantities of irregular elements and where the global approximate shape is more important than the exact composition.

Deploying robotic systems in such applications, however, poses several challenges. First, the fabrication of large-scale building structures typically requires the employment of mobile robots to overcome the constrained workspace limitations of stationary manipulators. These robots need to move during construction, while still being able to localize themselves with respect to the working environment and assemble structures accurately in space (Dörfler et al., 2016). Secondly, the use of arbitrarily-shaped materials found on-site requires efficient perception algorithms able to identify individual object instances on the fly, without having any previous knowledge about their geometry. Furthermore, to manipulate these irregular objects reliably, such systems must be capable of selecting good grasping poses among many possible configurations and have the ability of planning collision-free motions in potentially cluttered environments.

In this work, we address these challenges by demonstrating autonomous manipulation of large-scale stones with a robotic walking excavator. By extending state-of-the-art mapping and grasp planning approaches to real environments and irregular objects, we provide the robotic skills that are necessary to deploy autonomous construction machines for further construction applications (see Fig. 1). The proposed system initially maps the close vicinity of the robot by fusing the data acquired by its onboard LiDAR sensors and a camera placed on a drone, and segments single object-like instances in the resulting 3D point cloud. These segments are then added to a database that allows keeping track of their corresponding poses during manipulation. Finally, based on the map and the segment database, collision-free grasping poses are planned in order to move the objects of interest to the desired locations. All experiments are executed in the real-world setting and with the machine shown in Fig. 2, putting special emphasis on the evaluation of the robustness of the integrated perception and manipulation pipelines.

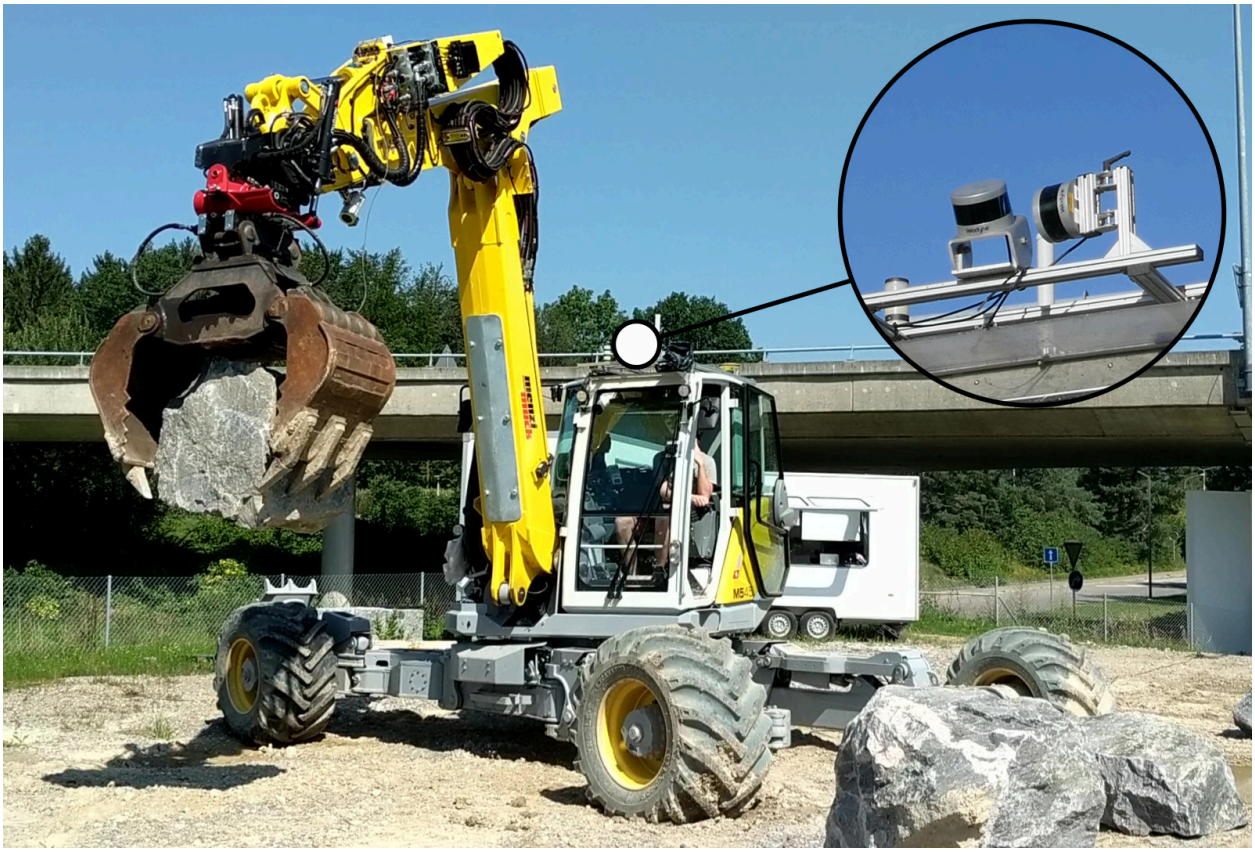


Figure 2: HEAP is an autonomous walking excavator based on a highly customized Menzi Muck M545. It is equipped with onboard sensors for state estimation (GNSS, IMUs, and encoders) and scanning the environment (3D LiDARs). As shown in the inset, the two LiDARs are mounted perpendicularly to each other such that a scanning motion is achieved not only while driving the robot around, but also while swinging the cabin.

In brief, this article presents the following contributions:

- An online laser mapping system working with two LiDAR sensors on the same platform, enhanced with object segmentation capabilities and a segment-based registration scheme that allows fusing mapping data acquired from heterogeneous external sensors (e.g. a drone-borne camera) into the LiDAR-based map.
- An object database that allows keeping track of identified objects while they are being manipulated by the robot.
- A robust collision-aware grasp planning methodology for irregular objects in slightly cluttered and occluded scenes.
- Demonstration of autonomous mapping, segmentation and grasp planning in real-world experiments performed with a robotic excavator.

To the best of our knowledge, this is the first demonstration of such a level of autonomy in real experiments for a construction task on an architectural relevant scale.

2 RELATED WORK

In order to autonomously build structures with material found on-site, construction robots require the ability to perceive and map the complex, unstructured surrounding space, segment individual objects of interest in it and manipulate these objects safely. Despite the recent advances in the context of on-site digital fabrication, most applications only consider the use of regular materials, such as pre-fabricated bricks (Dörfler et al., 2016), and there still exist only a few robots integrating all the aforementioned capabilities. For example, in (Saboia et al., 2018) and (Fujisawa et al., 2015), completely autonomous systems are shown capable of constructing auxiliary structures in order to achieve and maintain navigability across previously untraversable terrain. However, they use customized compliant bags as construction material or apply polyurethane foam, respectively, and not naturally occurring building materials such as stones. Closely related to our work are the demonstrators presented in (Furrer et al., 2017) and (Liu et al., 2019), consisting of stationary robotic systems that detect randomly placed stones and construct balancing vertical stacks with them. Although the focus is also put on handling building materials of arbitrary shape, the aforementioned systems assume that the geometry of the objects of interest is known beforehand, i.e. they are initially pre-scanned in an offline step. Our approach, on the other hand, aims towards more generic use-cases and does not consider any previous knowledge about the objects present in the scene. Instead, it attempts to discover object-like instances in a map of the robot's surroundings that is built online and perform manipulation based on this information.

Estimating the robot's pose together with an internal representation of the observed scene, which is commonly referred to as Simultaneous Localization And Mapping (SLAM), has been an extensively studied problem by the robotics community in the last decades (Cadena et al., 2016). Particularly, LiDAR-based SLAM approaches (Mendes et al., 2016; Droeschel et al., 2017; Shan and Englot, 2018) have become quite popular due to its applicability to self-driving vehicles and other types of ground robots. However, these systems typically target autonomous navigation use-cases and mainly seek to achieve good pose estimates over very long trajectories, producing purely geometric maps of the traversed environment. Conversely, to aid the manipulation tasks, we aim towards a higher-level scene representation where the notion of object instances is available. While object-aware mapping systems have gained attention recently (Furrer et al., 2018; McCormac et al., 2018; Grinvald et al., 2019), most of the existing approaches use volumetric map representations and rely on RGB-D sensing, whose applicability is usually inhibited in large-scale outdoor environments due to the limited working range and the poor performance under sunlight of depth cameras. Our mapping approach, on the contrary, is more similar to (Dubé et al., 2017b), which uses a map representation based on geometric segments extracted from 3D LiDAR point clouds. The main difference is that, while in (Dubé et al., 2017b) these segments often correspond to partial observations of objects or structures and are mainly used for localization and loop-closure detection (Dubé et al., 2017a), here we aim at obtaining complete models of the individual objects in the scene in order to assist interaction planning.

Achieving an accurate and complete reconstruction of the scene becomes especially challenging when operating on construction sites, as the presence of bulky objects (e.g. building material) can limit the robot's mobility and occlude large regions of the map. To overcome this drawback, we take inspiration from works on aerial-ground registration (Forster et al., 2013; Gawel et al., 2017) and enable the augmentation of the LiDAR-based map with additional mapping data acquired by an external sensor that can observe the scene from different viewpoints, such as a camera placed on a drone. A general approach to solve the 3D global registration problem is to extract features from the input maps, match them and estimate the geometric transformation that best explains the set of found correspondences (Holz et al., 2015). Similarly to (Dubé et al., 2017a), our method aims at directly aligning sets of point-cloud segments extracted from the input maps, as they are typically the most salient elements in the observed scene. However, instead of treating these segments as single features, which might struggle when dealing with large differences in viewpoint (Gawel et al., 2017), we compute local descriptors on these segments and then use the segments' centroids to efficiently select a geometrically consistent set of descriptor correspondences.

Grasp planning is often divided into *analytical* methods (Prattichizzo and Trinkle, 2016) that evaluate the performance of a grasp according to physical properties such as the ability to endure external wrenches, and *empirical* (data-driven) methods (Bohg et al., 2014) that use physical trials or human labels to evaluate the performance. For analytical methods, the pose of the object to grasp is first estimated in the scene and subsequently it is reasoned about where and how to place the gripper into a grasp configuration to achieve the desired contact configuration. While it can be already difficult to accurately localize the pose of an object in a cluttered scene, given a noisy and only partial point cloud (Glover and Popovic, 2013), it implies that an accurate model of the object to manipulate already exists, which is not applicable to our use case. For empirical methods on the other hand, there are multiple grasp detection approaches generating grasp configurations in cluttered scenes directly without localizing single object instances first (ten Pas and Platt, 2015; Kappler et al., 2015; Herzog et al., 2014). These approaches generate first a large number of grasp candidates on the input point cloud, usually originating from an RGB-D camera. Then, they evaluate the probability of the candidates being a grasp, e.g. in terms of force closure. For this, a classifier or regression system is trained to detect the parts of a point cloud that can be grasped, given a large amount of labeled data. Since these methods detect grasps based on graspable regions and independently of object instances, they generalize well for new objects. In our outdoor case, depth images of the scene are not available directly. Therefore, we perform the grasp detection on the point cloud reconstructed by 3D LiDAR mapping. Additionally, in our approach we incorporate the available information of the segments in order to augment the point cloud used for grasping. The registered object segments are added to the grasp cloud as well as an estimate of the ground plane in order to compensate for missing information in the partially cluttered scene. We use a similar approach as presented in (ten Pas et al., 2017) to sample grasp hypotheses on the point cloud and classify the grasp candidates. However, from the sampled grasps we perform a further selection step where the remaining candidates are filtered for heuristic criteria derived from the specific task to obtain the final grasp candidate. The idea of filtering the best generated hypotheses is that we are not necessarily interested in the one optimal grasp (Borst et al., 2003). Many hypotheses might have a similar score, and we are rather interested in a sufficiently good grasp that fulfills other requirements like how easily the configuration can be reached.

3 SYSTEM DESCRIPTION

As robot platform we use HEAP (Hydraulic Excavator for an Autonomous Purpose), a highly customized Menzi Muck M545 excavator developed for autonomous use cases as well as advanced teleoperation. This machine, shown in Fig. 2, has customized, precisely force and position controllable, hydraulic actuators in the arm and the legs, making it adaptable to any kind of terrain. On the sensing side, HEAP is equipped with a Leica iCON iXE3 with two Global Navigation Satellite System (GNSS) antennas and a receiver that can be used for localization of the cabin. Real-time kinematic (RTK) corrections for the GNSS signals are received over the Internet from permanently installed base stations. In addition, SBG Ellipse2-A Inertial Measurement Units (IMUs) are installed both in the cabin and on the chassis, and Sick BCG05-C1QM0199 wire draw encoders are used to measure the piston position and velocity of the arm cylinders, whose force is estimated with pressure sensors integrated in the servo valve control modules. Finally, two Velodyne Puck VLP-16 LiDAR scanners placed at the front edge of the cabin's roof are used for mapping tasks. As shown in Fig. 2, the LiDARs are mounted orthogonally to each other such that a scanning motion is achieved not only while driving the robot around, but also while swinging the cabin. In the scope of this work, the sensor mounted perpendicularly to the ground plane will be referred to as the *vertical LiDAR*, whereas the other one will be named *horizontal LiDAR*. For a more detailed description of the system's sensors and actuators, we refer the reader to (Jud et al., 2019).

Besides HEAP, an AscTec Neo hexacopter equipped with a Visual-Inertial (VI-) Sensor (Nikolic et al., 2014) is used in the experiments presented in this work to provide additional mapping data of the environment. This data is registered into the excavator's LiDAR-based map leading to a reconstruction of the scene that is free of occlusions, as explained in Sec. 4.4.

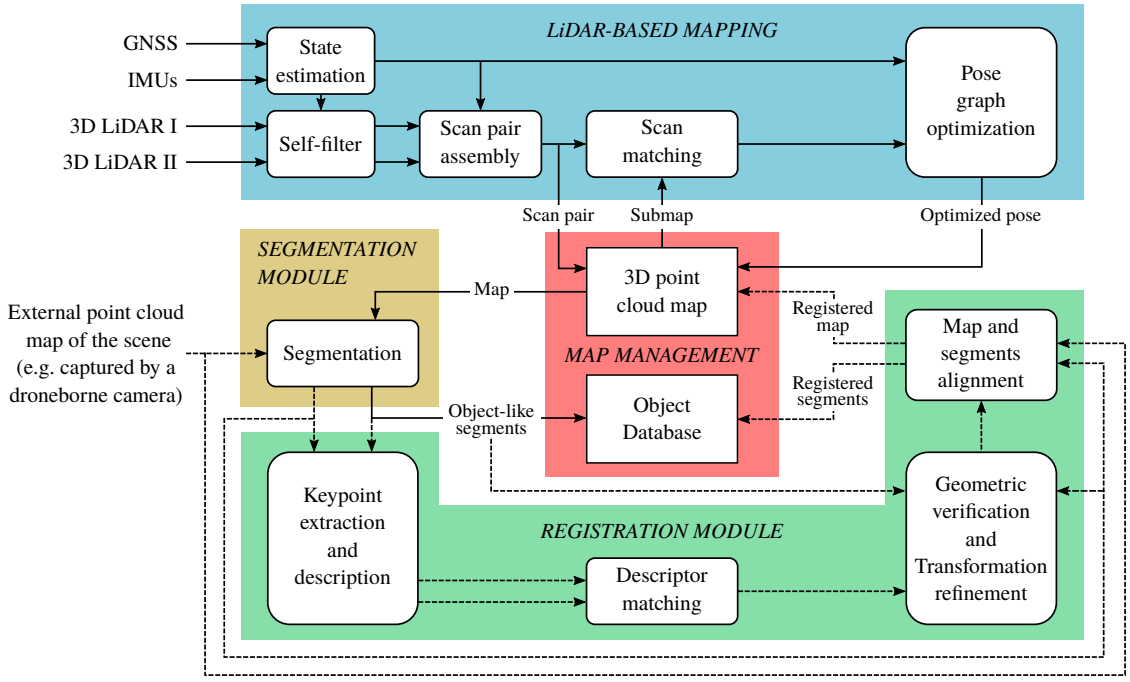


Figure 3: Overview of the perception pipeline developed for the excavator HEAP. The LiDAR, inertial and GNSS measurements are processed online to estimate the sensor-suite's pose and create a point-cloud map of the environment. When triggered, the segmentation routine extracts object-like segments from the current map and stores them as separate entities in the Object Database. Additionally, the registration module allows the fusion of externally built maps for improved scene reconstruction.

4 PERCEPTION PIPELINE

As the goal here is to autonomously manipulate objects of interest present in the scene, we are looking to achieve a consistent and up-to-date map of the robot's (i.e. excavator's) surroundings as well as accurate segmentation masks for the objects in the map. In this work, we assume no prior knowledge of the environment or of the objects to be grasped, for the sake of a generally applicable methodology. Therefore, we present a mapping pipeline that uses the excavator's onboard LiDAR sensors to incrementally build a 3D point-cloud map of the environment and is able to segment generic objects in it. Additionally, the system features a segment-based global registration module that enables fusing external mapping data, such as point-cloud maps generated from drone-borne vision sensors, into the LiDAR-based map to achieve a more complete reconstruction of the scene. The object-like instances segmented out from this map are stored in a database that is consulted to update the map following the movement of objects in the scene by the robot. An overview of the different modules constituting the perception pipeline is depicted in Fig. 3.

4.1 Vision-based Pre-mapping Using a Drone

With the aim of providing a complete map of the scene for grasp pose planning (i.e. free of occlusions), we intend to reconstruct a point cloud of the region of interest from multiple viewpoints and register it later on with the excavator's LiDAR-based map. To this end, we initially collect some visual data with a VI-Sensor mounted on the AscTec Neo hexacopter, which is manually piloted over the region of interest. The trajectory of the VI-Sensor's left camera is estimated online using the VINS-Mono SLAM system (Qin et al., 2018). Once this inspection task is done, we select a subset of about 200 images from the recorded

data and we perform an offline reconstruction of the scene based on Structure from Motion (SfM) using COLMAP (Schonberger and Frahm, 2016). The scale of the reconstructed model is finally recovered by computing the 3D similarity transformation between the camera locations provided by the SfM pipeline and the corresponding camera positions previously estimated with VINS-Mono. Although efficient vision-based online mapping algorithms exist, we decide to use a method that produces high quality point clouds (see Fig. 7 for an example), allowing for a better alignment with the LiDAR-based map. Note, however, that this method is used as a proof of concept and that other sensing modalities and mapping approaches able to reconstruct point-cloud maps of an observed scene (e.g. LiDAR-based) could also be employed here.

4.2 LiDAR-based Scene Mapping

The excavator's LiDAR-based mapping system exploits the well-established graph-based SLAM formulation (Grisetti et al., 2010), which typically models the underlying structure of the problem as a pose graph. In graph-based SLAM, every node in the graph corresponds to a robot pose, while edges connecting these nodes encode spatial constraints between robot poses (i.e. relative transformations) that result from observations or odometry measurements. Solving the SLAM problem then consists of determining the set of robot poses that best satisfies all the spatial constraints.

In mathematical terms, given a set of state variables $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$, where $\boldsymbol{x}_i$ describes the pose of node i, and a set of measurements, with $\boldsymbol{z}_{ij}$ and $\boldsymbol{\Omega}_{ij}$ being respectively the mean and the information matrix of a single measurement relating nodes i and j, the goal of this maximum likelihood approach is to find the configuration of the nodes $\mathcal{X}^*$ that minimizes the negative log-likelihood of all the observations. This is equivalent to solving the following least squares error minimization problem:

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_{\langle i,j \rangle \in \mathcal{C}} \boldsymbol{e}_{ij}^T \boldsymbol{\Omega}_{ij} \boldsymbol{e}_{ij}, \tag{1}$$

where $\mathcal{C}$ denotes the set of pairs of indices for which a measurement is available and $\boldsymbol{e}_{ij} = \boldsymbol{z}_{ij} - \hat{\boldsymbol{z}}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a vector error function that measures how well the constraint originated from measurement $\boldsymbol{z}_{ij}$ is satisfied by the predicted measurement $\hat{\boldsymbol{z}}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, given a configuration of the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

The nonlinear optimization problem in Eq. 1 can be effectively solved using the Gauss-Newton approach. Specifically, the proposed mapping system builds on top of the LaserSLAM (Dubé et al., 2017b) back-end, which performs incremental pose-graph update and optimization using the iSAM2 (Kaess et al., 2012) algorithm to estimate the robot's trajectory in real time. Measurements provided by the onboard sensors are processed by a customized front-end to generate odometry and scan-matching constraints between consecutive nodes of the pose graph. In this work, odometry constraints are generated from state estimation, which fuses GNSS measurements with the inertial data provided by the robot's IMUs. Scan-matching constraints, on the other hand, are computed by aligning consecutive LiDAR scans using a process that consists of the three following steps:

1) **Self-filtering:** To prevent the scene map from being corrupted by the robot's arm and legs, which move mostly inside the field of view of the LiDAR sensors, each incoming scan is initially processed by a self-filter that approximates the excavator's links with simple shapes, such as boxes or cylinders, and uses the robot's current state to discard the points lying on the robot itself (Jud et al., 2019).

2) **Scan Pair Assembly:** Once a new pair of filtered scans is available, the vertical scan S_k^V is merged with the corresponding horizontal scan S_k^H to form an "assembled scan pair" S_k, i.e. a point cloud composed of both S_k^V and S_k^H, expressed in the frame of the horizontal LiDAR. This requires S_k^V to be transformed into the frame of S_k^H, which is achieved by applying a fixed pre-calibrated transformation $\boldsymbol{T}_{\mathcal{HV}}$ from the vertical LiDAR frame $\mathcal{V}$ to the horizontal LiDAR frame $\mathcal{H}$. Since the sensors are not synchronized, we additionally use the state estimation measurements to correct for the potential motion of the cabin between the timestamps at which the two scans being merged were acquired.

3) **Scan Matching:** Constraints relating consecutive robot poses are obtained using the Iterative Closest Point (ICP) algorithm to register the current scan pair S_k against a *submap* composed of the m previous scan pairs, expressed in the frame of the previous horizontal LiDAR's pose. Compared to performing scan matching for each individual scan, our approach has two main advantages: first, given that scan pairs contain more points, the ICP registration against the *submap* becomes more robust; secondly, since nodes are added on a scan pair basis, the pose graph keeps the same size as if a single LiDAR sensor was used.

The LiDAR-based map is created by accumulating the 3D scan pairs once the corresponding nodes are optimized by the back-end. To avoid uninformative accumulation of data and the growth of the pose graph when the robot is not moving, a new node together with one odometry and one scan-matching constraint is added to the graph only if the horizontal LiDAR has travelled a minimum distance d_{poses}. Furthermore, since we are mostly interested in mapping the excavator's workspace, a local point cloud is extracted upon addition of new scans by defining a robot-centric cylindrical region, whose radius R_{map} is set to be slightly higher than the maximum arm's reach. The map data is finally filtered and downsampled using a voxel grid of resolution res_{voxel} with n_{min}, a minimum number of points per voxel to consider it as occupied. The result is a 3D point cloud map that is incrementally built around the current robot location without being corrupted by the platform's moving parts.

4.3 Map Segmentation

When the point density of the LiDAR-based map reaches a certain threshold during an initial scanning phase, a segmentation routine is automatically triggered in order to detect any distinct objects of interest present in the scene. Such objects are extracted from the current map as a set of 3D point clusters P_i (i.e. segments) using a geometric technique inspired by (Douillard et al., 2011). This method assumes that the ground operates as a separator between the segments and thus requires it to be previously removed from the input point cloud. In our implementation, this is achieved by running a RANSAC-based 3D plane fitting algorithm with a distance margin d_{plane} that accounts for potential terrain undulations. After most of the points belonging to the ground have been removed, Euclidean clustering is used to grow segments. The extracted clusters then go through a RANSAC-based planarity check that discards nearly planar segments, which usually correspond to regions of the terrain not having been filtered out previously. Finally, the remaining non-planar segments P_i and their centroids $\boldsymbol{c}_i$ (i.e. the average of all P_i's points), are added to the Object Database described in Sec. 4.5.

4.4 Segment-based Global Registration

The registration module gives the perception system the ability to align externally built point cloud maps with the LiDAR-based map that the robot uses for localization. This becomes especially valuable when the goal is to perform manipulation tasks, since augmenting the robot's map with additional data acquired by an external sensor (e.g. a camera placed on a drone) helps adding information in occluded regions of this map and allows for effective 3D reconstruction of the objects in it.

Our registration approach, whose goal is to find the transformation that best aligns the externally built (i.e. source) map with the LiDAR-based (i.e. target) map, leverages the segmentation module presented in Section 4.3 to perform registration on the basis of local geometric descriptors computed on segments. The resulting transformation is finally used to align the two original input maps and merge the overlapping segments from the source and target point clouds. By choosing a purely geometric method instead of relying on GPS-based co-localization strategies, our method is able to deal with mapping data acquired by robots that do not use GPS-based localization and is also suitable in GPS-denied environments. The proposed approach consists of the following steps:

1) **Segmentation of the Input Maps:** When the registration is triggered, the source map is initially filtered using a voxel grid of resolution res_{voxel} (i.e. the same that is used to downsample the LiDAR-based map), resulting in a point cloud which has a similar point density to the target map. Both the source and the target point clouds are segmented using the method described in Sec. 4.3. This step is especially useful to remove parts of the input maps whose geometry is not descriptive enough to allow for robust matches, e.g. planar or low point-density regions.

2) **Keypoint Extraction and Description:** From both segmented point clouds, we extract keypoints using the Intrinsic Shape Signatures (ISS) detector (Zhong, 2009) and describe them using the Unique Signatures of Histograms for Local Surface Description (SHOT) (Tombari et al., 2010). Besides SHOT, we tested two additional descriptors, namely the Fast Point Feature Histograms (FPFH) (Rusu et al., 2009) and the Rotational Projection Statistics (RoPS) (Guo et al., 2013), but they exhibited lower performance: the former were less robust in the matching step, whereas the latter, despite achieving comparable results to the SHOT descriptors, were considerably more expensive to compute, because the triangulation of both input point clouds was required in this case.

3) **Descriptor Matching:** The matching module solves the data association problem between keypoints extracted from both input maps by comparing their descriptors. In our implementation, we perform an efficient nearest-neighbour search in the descriptor space using a kd-tree.

4) **Geometric Verification:** From the set of 3D keypoint correspondences identified in the previous step, we extract clusters of geometrically consistent matches (i.e. matches that vote for the same geometric transformation) and select the b most voted for transformations. These transformations are then used to transform the source segment centroids from the source map frame $\mathcal{S}$ to the target map frame $\mathcal{M}$. For each set of transformed source centroids, we perform a nearest-neighbour search against the set of target centroids. Matches between closest centroids are accepted if the Euclidean distance between them lies below a threshold d_{match}. Finally, after all candidate transformations have been evaluated, the one that gives the highest number of inlier centroid matches gets selected. To discriminate between transformations leading to same inlier ratios, we choose the one that minimizes the highest distance between matched centroids.

5) **Transformation Refinement:** The transformation selected in the previous stage, which we denote as $\boldsymbol{T}_{\mathcal{MS}}^{\text{coarse}}$, is used as a prior in an ICP step that refines the alignment of the source and target point clouds, yielding an improved transformation $\boldsymbol{T}^{\text{icp}}$. To reject unsuccessful registrations, we apply a threshold e_{icp}^{*} to the Root Mean Square Error (RMSE) of ICP, which is computed using only the segments whose centroids have been matched in the geometric verification step. The final transformation between the source and the target maps, $\boldsymbol{T}_{\mathcal{MS}}$, is then computed as follows:

$$\boldsymbol{T}_{\mathcal{MS}} = \boldsymbol{T}^{\text{icp}} \cdot \boldsymbol{T}_{\mathcal{MS}}^{\text{coarse}} \tag{2}$$

6) **Map and Segments Alignment:** The transformation obtained as a result of the registration process, $\boldsymbol{T}_{\mathcal{MS}}$, is used in a last stage to align the original maps. In addition, overlapping segments from the source and target maps are combined forming single object models. This is done by merging each pair of previously matched segments in a common point cloud, which is then downsampled using a voxel grid filter. This way, the output of the registration pipeline is not only a map containing the registered input point clouds, but also the segmented, better reconstructed object models, which are fed into the Object Database described in Sec. 4.5 to aid the manipulation tasks.

4.5 Object Database and Dynamic Object Handling

To properly deal with potentially movable object instances in the map, we set up a database where, for each segmented object, we store its globally referenced point cloud P_i, its centroid position $\boldsymbol{c}_i$ and an identification number i. These object instances are generated as a result of the segment-based registration process, when an external map is available, or by simply segmenting the LiDAR map using the module described in Sec. 4.3.

The database is then used to aid the manipulation tasks by constantly updating the locations of the objects in the map, as described in the remaining of this section.

When an object is grasped, we associate the transformation between the map frame $\mathcal{M}$ and the gripper frame $\mathcal{G}$, $\boldsymbol{T}_{\mathcal{MG}}^{\text{grasp},i}$, given by the state estimation, with the corresponding object instance stored in the database. In addition, we remove all map points inside a sphere of radius R_{obj} centered around the grasped object's centroid in the LiDAR map. After lifting the robot's arm, the hole created in the map will be replaced gradually by newly detected points on the ground, while the self-filter described in Sec. 4.2 will prevent the object from being remapped as long as it lies inside the gripper.

As soon as the object is released in a new position, we again extract from state estimation the transformation representing the gripper pose with respect to the map frame, $\boldsymbol{T}_{\mathcal{MG}}^{\text{release},i}$, and use it together with the previously stored grasp pose to estimate the transformation $\boldsymbol{T}^i$ experienced by the corresponding object instance in the scene:

$$\boldsymbol{T}^i = \boldsymbol{T}_{\mathcal{MG}}^{\text{release},i} \cdot \left(\boldsymbol{T}_{\mathcal{MG}}^{\text{grasp},i}\right)^{-1} \tag{3}$$

This transformation is then applied to the object's point cloud and centroid. Note that, by representing the objects with their globally referenced point clouds, we implicitly keep track of their position and orientation in the map.

After releasing the object, a scanning motion is performed, which causes the object to be remapped by the LiDARs. At this point, an ICP step is triggered to realign the transformed object point cloud with the LiDAR map, correcting for potential inaccuracies in the predicted object's location. This way, the map and the Object Database are always kept consistent with the state of the environment and, at the same time, the identified objects can be individually tracked as they are being moved by the robot.

5 GRASP POSE PLANNING PIPELINE

The goal of the grasp pose planning pipeline (Fig. 4) is to find viable grasp configurations in order to pick the segmented objects instances. In the case of the autonomous excavator, a grasp configuration is defined as a 6 degree of freedom (DoF) pose of the gripper where a contact configuration with the object can be performed. A grasp configuration with contact wrenches that span the origin of the object is called a *force closure* grasp. The purpose of grasp pose detection is to find force closure grasps on the object of interest.

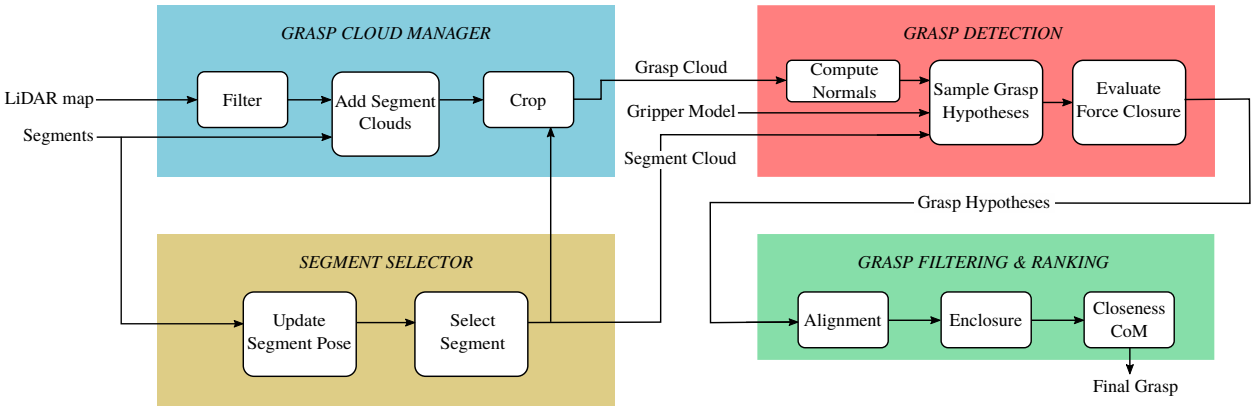


Figure 4: Overview of the grasp pose planning pipeline. The LiDAR map is augmented with the registered object segments and cropped to the region of interest around the selected segment. The generated grasp cloud is used for sampling grasp hypotheses which are filtered and ranked to obtain the final grasp.

The grasp pose planning consists of three steps: identification of the region of interest and generation of a planning point cloud; detection of grasp candidates in the region of interest; and subsequent filtering and ranking of the candidates to obtain a feasible grasping configuration. We will focus here on finding grasp poses for two-finger grippers, as the excavator is equipped with a two-jaw angular gripper. Both jaws are mechanically connected and are moved by the same hydraulic actuator, giving one DoF. The jaws are mounted on the base part of the gripper that we will further reference to as palm. For collision checking between gripper and planning point cloud, the gripper shape is approximated with convex polyhedra enclosing the jaws and the palm.

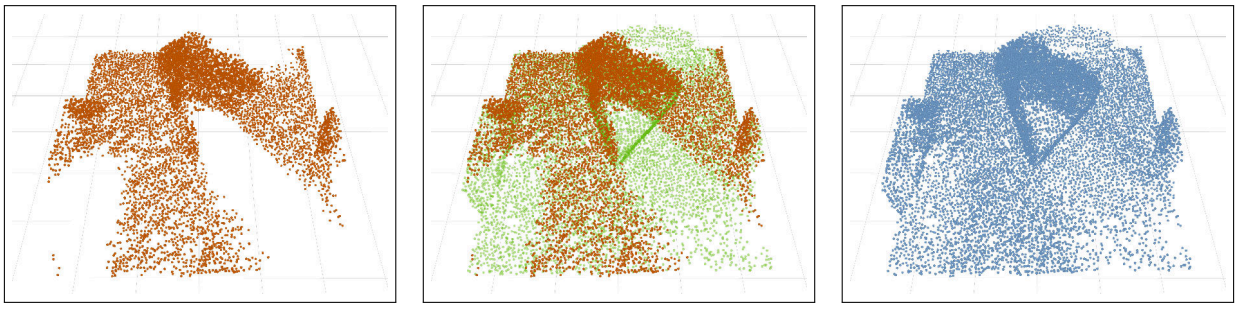
5.1 Segment Identification

The grasp pose detection tries to find feasible grasp configurations directly on a point cloud representing the proximity of segmented objects, called planning point cloud. This allows not only to evaluate the quality of grasps, but also to consider the collision with surrounding objects and the ground. In a typical application the decision which object to grasp would come from a higher level planning instance, and depends on the structure to be built. In our case we decide to grasp the largest segment with the most points from all the available segments P_i (Sec. 4.3). Starting with the largest segment is motivated by using them first for building a construction.

5.2 Grasp Planning Point Cloud

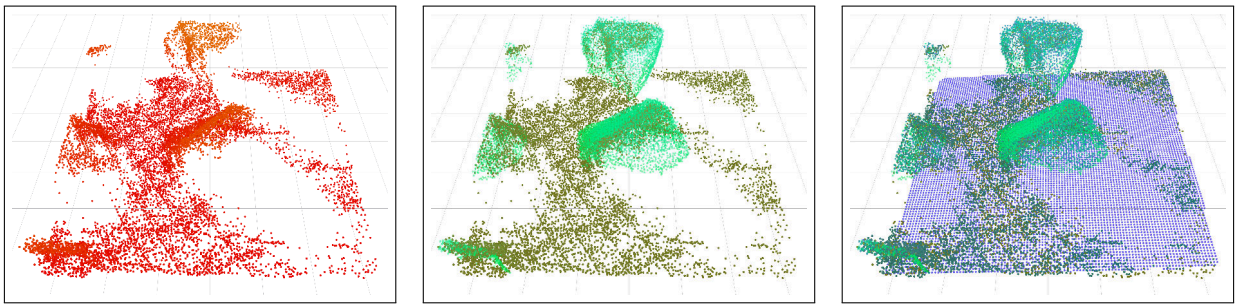
The grasp planning point cloud around an object of interest is generated as preparation for subsequent manipulation. In order to obtain the planning point cloud, we crop the LiDAR-based map (Sec. 4.2) around the location of the selected segment's centroid $\boldsymbol{c}_i$ with a margin corresponding to the full gripper width (Fig. 5a) and combine it with the registered external point cloud (Fig. 5b). The merged maps are down-sampled to obtain a uniform point density on the grasp planning point cloud (Fig. 5c).

If no external point cloud is available, e.g. after relocating the segments from their initial position, we could only rely on the LiDAR map for grasp planning (Fig. 6a). However, the LiDAR-based map may be partially cluttered and occluded by the objects itself due to the view point, leading to holes in the map where no data points are available. To provide a complete map for grasp pose planning without external map, the aligned object models from the Object Database are merged to the planning point cloud, assuring that we also have information on the opposite side to the LiDAR direction (Fig. 6b). There might still be holes in the ground plane due to the occlusion by the objects, leading to grasps that would penetrate the ground. To fill these holes, a RANSAC plane estimation is performed to find the ground plane and to artificially augment over the complete planning point cloud (Fig. 6c).



(a) (b) (c)

Figure 5: Given an externally built scene point cloud, the grasp planning cloud is obtained by cropping the LiDAR map in a region of interest around the desired Segment (a) and merge it with the registered external point cloud (b). The merged map is down-sampled to get a uniform distribution (c).



(a) (b) (c)

Figure 6: Without external point cloud, the grasp planning cloud is obtained by cropping the LiDAR map in a region of interest around the desired Segment (a). The registered segments are merged to the planning cloud (b) and a ground plane is fitted to fill holes in the map due to occlusion (c).

5.3 Grasp Detection

The intention of grasp detection is to generate a large number of grasp hypotheses on the planning point cloud that do not collide with the environment. In a subsequent step, the hypotheses can be filtered and ranked according to their applicability and success chance. To generate the grasp hypotheses, N points are sampled on the desired segment P_i in the planning point cloud. Each sample p is assigned a local reference frame $\mathcal{F}$ by evaluating the Eigenvectors of the matrix

$$\boldsymbol{M}(p) = \sum_{q \in \boldsymbol{B}_r(p)} \boldsymbol{n}(q)\, \boldsymbol{n}(q)^\top \tag{4}$$

where $\boldsymbol{n}(q)$ is the outwards pointing unit surface normal at point q, and $\boldsymbol{B}_r(p)$ is the r-ball around point p. The local reference frame is composed of $\mathcal{F} = [\boldsymbol{\nu}_3(p), \boldsymbol{\nu}_2(p), \boldsymbol{\nu}_1(p)]$, where $\boldsymbol{\nu}_1(p)$ corresponds to the largest Eigenvalue and $\boldsymbol{\nu}_3(p)$ to the smallest one. This assures that the x-axis of the reference frame is pointing away from the object and the z-axis is pointing along the axis of minimal curvature. For each sample point p, we generate multiple orientations by rotating around the y- and z-axis of the local reference frame in discrete intervals. The resulting x-axis of the rotated frame is denoted as approaching direction. The polyhedral gripper model with open jaws is moved along the approaching direction until palm or jaws are in contact with the planning point cloud. We add a sampled pose to the list of grasp hypotheses if the closing region of the jaws is not empty.

Note that the grasp detection does not require a perfect segmentation of the object as the planning point cloud represents the closer vicinity of an object. However, faulty segmentation, like merging close objects to one segment, may cause that grasp contact points to be placed on several different objects, which is undesired because it reduces the grasp success rate and we are generally interested in manipulating one object at the time.

5.4 Grasp Filtering and Ranking

The grasp detection can generate hundreds of grasps for a single object that have to be filtered and scored in order to obtain the desired grasping configuration. In a first step, the grasp hypotheses are evaluated for force closure. In (ten Pas et al., 2017) they present a grasp classifier using a four-layer Convolutional Neural Network (CNN) that predicts if a grasp is a force closure based on the planning point cloud and a 2-finger gripper model. We apply this classifier to the detected grasp hypotheses. This predicts whether the grasps are force closure, but still many candidates may be valid. To select the final grasp candidate, we sequentially rank and filter the remaining grasps by task specific criteria that showed to provide reliable grasps. First, the grasps are ranked according to their approaching direction. Given the upwards directed normal of the

ground plane $\boldsymbol{n}_{\text{ground}}$ and the approaching direction of a grasp hypothesis, which is the normal pointing away from the palm $\boldsymbol{n}_{\text{approach}}$, the alignment cost c_{align} is given as the angle between the two vectors

$$c_{\text{align}} = \arccos\left(-\boldsymbol{n}_{\text{ground}}^{\top}\boldsymbol{n}_{\text{approach}}\right). \tag{5}$$

We prefer approaching directions that are normal to the ground plane because they correspond to a nominal configuration of the excavator arm. Furthermore, they help assuring that the arm is not colliding with surrounding objects. After selecting the best 40 grasps based on the approach normal by minimizing c_{align}, they are ranked on how much the object is enclosed by the gripper and the 20 best grasps are selected. An enclosing (or power) grasps is preferable to a pinching (or precision) grasp as the object can be gripped more stably. The enclosing cost c_{enc} is represented by the distance between the palm and the centroid in approaching direction

$$c_{\text{enc}} = \boldsymbol{r}_{\text{CoM}}\boldsymbol{n}_{\text{approach}}, \tag{6}$$

where $\boldsymbol{r}_{\text{CoM}}$ is the position vector from the palm center to the object centroid $\boldsymbol{c}_i$. Finally, the grasps are ranked by closeness to the centroid of the segment. The closeness of a grasp to the centroid is given by the distance

$$c_{\text{dist}} = \frac{\|\boldsymbol{r}_{\text{CoM}} \times \boldsymbol{n}_{\text{approach}}\|}{\|\boldsymbol{n}_{\text{approach}}\|} \tag{7}$$

between the centroid of a line going through the Tool Center Point (TCP) along the approaching direction. This criterion is motivated by the need to avoid torsional moment on the grasped object during motion that might lead to rotational shift of the object in the gripper. The best ranked grasp is selected to be executed.

6 EXPERIMENTS

To show the applicability and repeatability of the presented system, we implemented the different modules using the Robotic Operating System (ROS) (Quigley et al., 2009) and integrated them on the robotic excavator HEAP to perform autonomous manipulation of large objects. The goal is to autonomously map, segment and grasp a set of randomly placed, irregularly-shaped stones and move them to user-predefined positions, while constantly updating the map in the process, as shown in the complementary video footage[1].

6.1 Experimental Setup

We use a set of seven gneiss rocks which show variety in properties like shape and size, ranging from $0.5\,\text{m}^3$ to $1\,\text{m}^3$, approximately. The geometric models of the stones are not available beforehand and therefore need to be discovered and segmented on the fly.

We perform two experiments, each of them starting with the stones randomly placed on rough terrain within the reach of the excavator's arm. In a first step, a point cloud of the initial stone arrangement is reconstructed from the data acquired by the drone-borne VI-Sensor using the method explained in Sec. 4.1 (see Fig. 7 for an example). Upon creation of the vision point cloud, we start building the LiDAR map online by swinging the excavator's cabin. When the region of interest has been scanned by the LiDAR sensors, the vision point cloud is registered into the LiDAR map using the approach described in Sec. 4.4 and the segmented object instances are fed into the Object Database. Then, the robot starts planning viable grasp configurations and paths in order to pick the detected objects and place them at different user-predefined positions, while updating the map accordingly. In the scope of these experiments, the stones are moved to equally-spaced fixed locations on the ground. To achieve actual architectural construction, additional planning is necessary to decide the exact location of the objects in the target structure. This is an involved process as not only geometric fit, but also structural stability and functionality of the target structure have to be considered (Johns et al., 2020), and therefore will not be addressed in detail in this work.

[1] `https://youtu.be/4bc5n2-zj3Q`

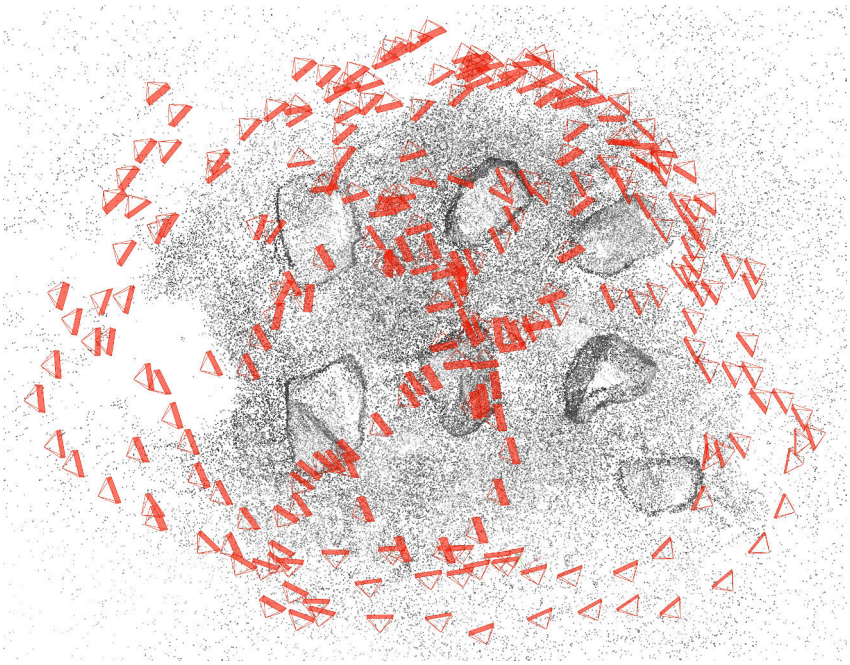


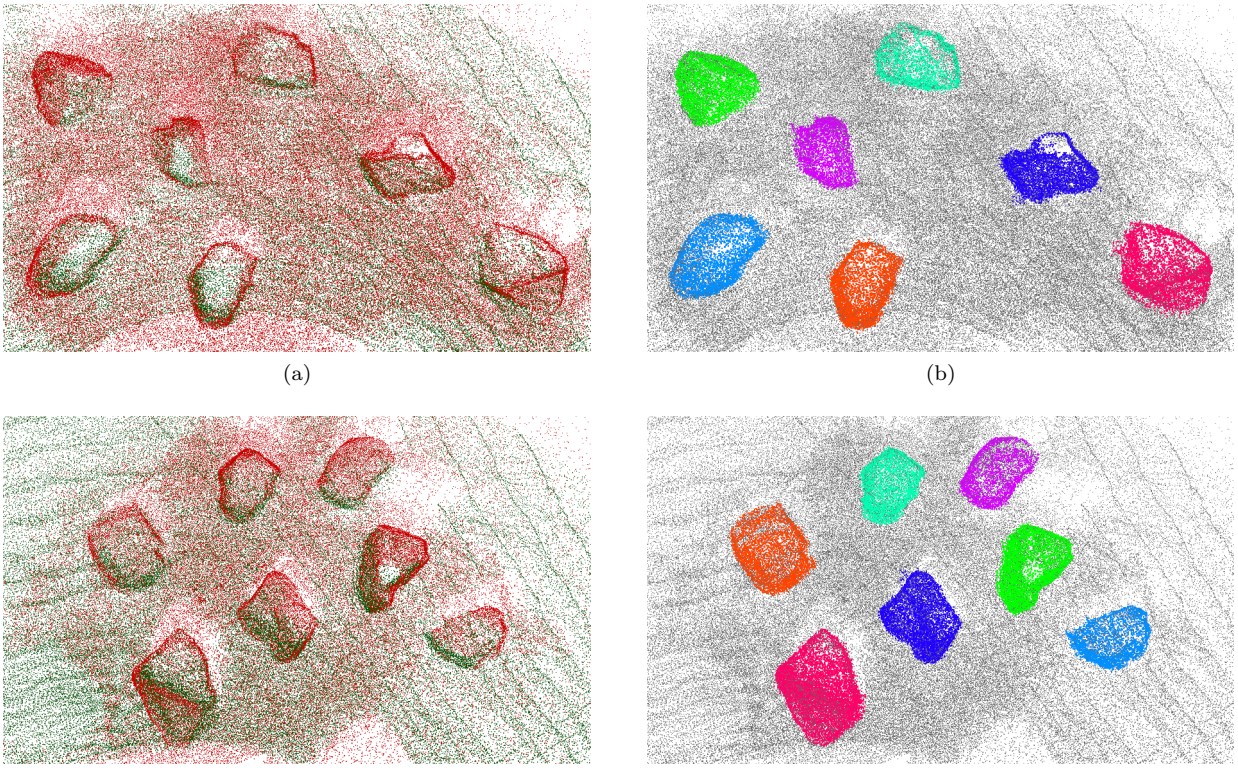
Figure 7: Top view of the reconstructed point cloud of the second experiment's initial stone arrangement from drone images. The red frusta represent the camera poses from which the registered images were taken.

All the movements of the platform are planned and executed autonomously except for the closing and opening of the gripper, and the operator is placed in the cabin for supervision and safety only. For motion planning, we use a whole-body trajectory planning framework that is suitable for both, wheeled (legs) and non-wheeled (arm) limbs (Jelavic and Hutter, 2019). However, since the focus of this work is the manipulation of the stones and not the traversing of the terrain, we keep the excavator base always at the same desired location during manipulation. For the arm motion itself, a simple heuristic-based planning approach suffices. The arm trajectory is composed of waypoints that respect collision constraints with the excavator's cabin and legs, and approach the start and goal poses perpendicularly to the ground. These waypoints are then interpolated with Hermite splines to put together a trajectory. The arm controller used to track the trajectory relies on a hierarchical-optimization based inverse kinematics approach that computes joint velocities and enforces kinematic limits (Bellicoso et al., 2016).

In the first experiment, the stones are grasped and moved once, whereas in the second one, the process of grasping and relocating all the stones is repeated four consecutive times without resetting the map. With this, we demonstrate that the perception pipeline effectively handles objects being moved by the robot and the map can therefore be used for planning grasp poses and safe motions during the course of a continuous application.

6.2 Segmentation and Global Registration

Fig. 8 illustrates the maps obtained as a result of the initial mapping phase in the two experiments, before moving any stones. These maps are created by swinging the excavator's cabin for about 20 seconds before triggering the registration routine to align the previously reconstructed vision point cloud with the LiDAR map, which is being built online. For each stone arrangement, we show the registered vision and LiDAR-based point clouds, as well as the segmented object instances, which are obtained after merging corresponding segments extracted from both input point clouds. These qualitative results evidence that the registration module is especially useful for adding information in occluded regions of the LiDAR map and improving the completeness of the segmented object models.



(a) (b) (c) (d)

Figure 8: Example results of the map registration and segmentation process during the first (a, b) and the second (c, d) experiments. The left pictures show the aligned LiDAR (green) and vision (red) maps, whereas the right pictures illustrate the merged map (grey) overlaid with the segmented object instances (shown in different colors).

To evaluate the robustness of the segmentation and registration modules against a variable number of accumulated LiDAR scans, we use additional LiDAR data that was collected while swinging the excavator's cabin after setting up each of the experiments mentioned above, as well as the reconstructed vision point clouds of the two initial stone arrangements. The recorded data is played back and, while the LiDAR map is being built, the registration routine is triggered every five seconds for a total duration of one minute. Each trial is defined as successful if the RMSE of the final ICP step, as defined in Sec. 4.4, lies below e^*_{ICP} and the seven stones get detected as single objects in the map.

Results show that approximately 15 seconds need to be spent in building a LiDAR map of the initial region of interest, which in our experiments is of about $50\,\mathrm{m}^2$, before achieving the first successful registration. The subsequent registration attempts (here 20 attempts in total) are executed with an overall success rate of 90 %, proving our segmentation and registration methods to be able to handle different point densities in the LiDAR map. The observed failure cases are caused by the fact that many of the extracted 3D features are wrongly matched and the algorithm is not able to find a geometrically consistent transformation that is close enough to the optimal one. In the real-world application, however, these cases are detected by monitoring the RMSE of the final ICP step, and the system continues accumulating LiDAR scans until a successful registration is achieved.

For the sake of completeness, in Table 1 we report the computational times of the individual steps in the registration pipeline, including the initial segmentation of the input point clouds, when executed on an Intel Xeon E-2176M CPU. As it can be observed, the complete segment-based registration routine is executed in approximately 1 second in both of our experiments, which makes our approach suitable for online operation. The sizes of the maps used to perform the experiments detailed in this section are reported in Table 2. Additionally, the most relevant parameters of the perception pipeline are summarized in Table 3.

Table 1: Mean computation times and standard deviations (in ms) of each step involved in the registration process, as computed on an Intel Xeon E-2176M CPU.

Submodule	**Experiment 1**		**Experiment 2**	
	LiDAR	**Vision**	**LiDAR**	**Vision**
Segmentation	52 ± 8	58 ± 1	61 ± 17	58 ± 1
Keypoint extraction	57 ± 7	47 ± 2	48 ± 14	53 ± 2
Keypoint description	17 ± 2	18 ± 1	20 ± 12	21 ± 1
Descriptor matching	279 ± 23		284 ± 64	
Geometric verification	8 ± 1		7 ± 2	
Transform. refinement	470 ± 226		420 ± 275	
Total	998		974	

Table 2: Sizes of the LiDAR-based and vision maps (mean and standard deviation given in number of points) in the registration experiments. Note that the size of the LiDAR maps varies as the number of accumulated scans increases, whereas the vision point clouds are pre-computed and therefore their size is fixed.

Experiment 1		**Experiment 2**	
LiDAR	**Vision**	**LiDAR**	**Vision**
92301 ± 9443	73547	123564 ± 29622	98190

Table 3: Main parameters of the perception pipeline.

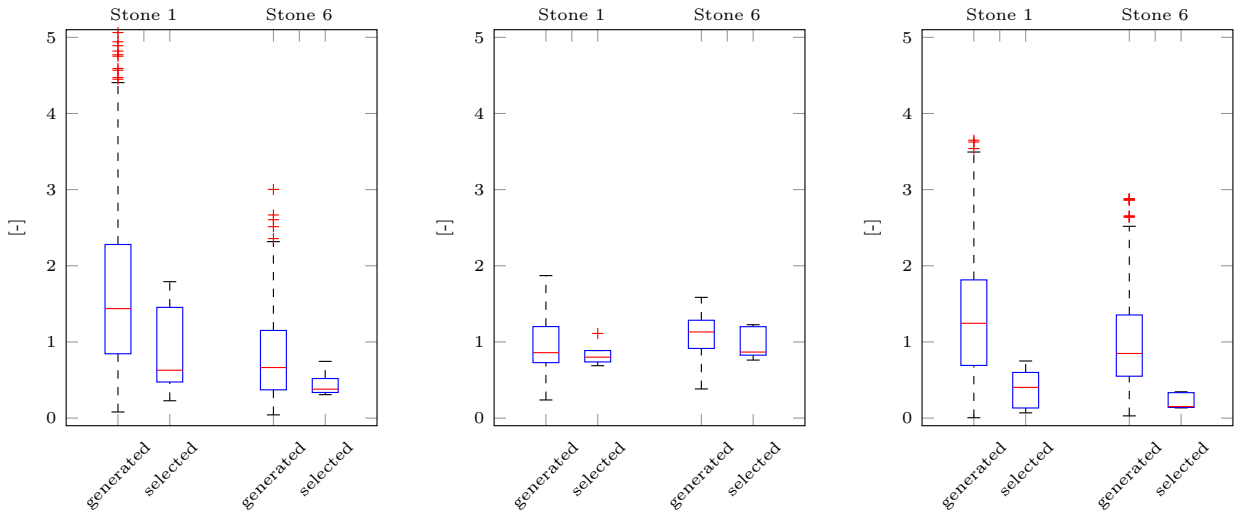
LiDAR-based Mapping	
Min. distance between poses, d_{poses}	1 cm
Num. of scan pairs per submap, m	10
Local map radius, R_{map}	10 m
Voxel grid resolution, res_{voxel}	3 cm
Min. point count per voxel, n_{min}	1
Radius for object removal, R_{obj}	1 m
Segmentation and Registration	
Plane fitting distance threshold, d_{plane}	10 cm
ISS detector salient radius, R_{iss}	6 cm
SHOT descriptor radius search, R_{shot}	10 cm
Max. num. of candidate transformations, b	10
Max. dist. btw. matched centroids, d_{match}	1 m
ICP RMSE threshold, e^*_{icp}	3 cm

6.3 Grasp Pose Planning

The stones were grasped and relocated five times each, resulting in a total of 35 grasp attempts. The best ranked grasp from the grasp pose planning pipeline was selected and could be executed with a success rate of 88.6 %. A grasp is deemed successful if it is collision-free, force closure and allows moving the object without dropping it. Grasp attempts that are not successful are detected by applying a closing force while lifting the stone. In case of slippage or dropping, the gripper jaws further close during the lift process, which is detected by the wire draw encoder of the gripper piston. A sudden drop of the object after lifting can be recognized by monitoring the arm cylinder forces. Note that this notion of grasp success does not account explicitly for any safety margin, i.e. whether the grasped object can withstand an external disturbance force, but it implies robustness to orientation changes and shaking during placing trajectory execution. If the best ranked grasp could not be executed successfully, the grasp pose was manually adjusted in order to relocate the object.

The grasp filtering and ranking is designed to successively filter grasps with high costs for a certain criteria and finally select a grasp that achieves the lowest cost for the distance between the TCP and the stone centroid. Fig. 9 compares the costs of the grasp filtering and ranking by the three criteria consisting of the alignment of the approaching direction to the ground plane (a), the enclosing of the grasp as distance between the centroid and the palm (b), and the closeness of the grasp to the centroid (c) for all generated grasps predicted being force closure and the finally selected grasps. For better comparison, each grasp cost is normalized by its median value of all generated grasps. The evaluation is shown for two different stones labeled with 1 and 6 (see Fig. 10). Stone 1 is more roundish, whereas stone 6 is an example of a flat stone. The evaluation for the other stones shows similar results and is left out for the sake of brevity.

We can see that for the roundish stone 1, the generated grasp hypotheses have a larger variation in the cost representing alignment to the ground plane (see Fig. 9a) as for the flattish stone 6, meaning that the generated grasps are approaching the object from directions all around it. Whereas stone 6 is flat and has to be pinched by the gripper, stone 1 is better suited for an enclosing grasp as it is larger and more roundish, resulting in a better enclosing cost (see Fig. 9b). Because the distance between the TCP and the centroid is



(a) Alignment (b) Enclosing (c) Distance CoM - TCP

Figure 9: Grasp costs of the hierarchical filtering steps in terms of the alignment of the grasp approaching direction to the ground plane normal in (a), the enclosing, measured as distance from the palm to the centroid in approaching direction in (b), and the closeness of the tool center point to the centroid of the stone in (c). Shown are the cost distribution of all generated grasps predicted being force closure and the selected grasps of stone 1 and 6. For comparison, each grasp cost is normalized by its median value of all generated grasps.

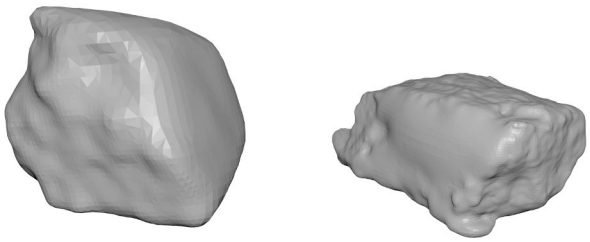
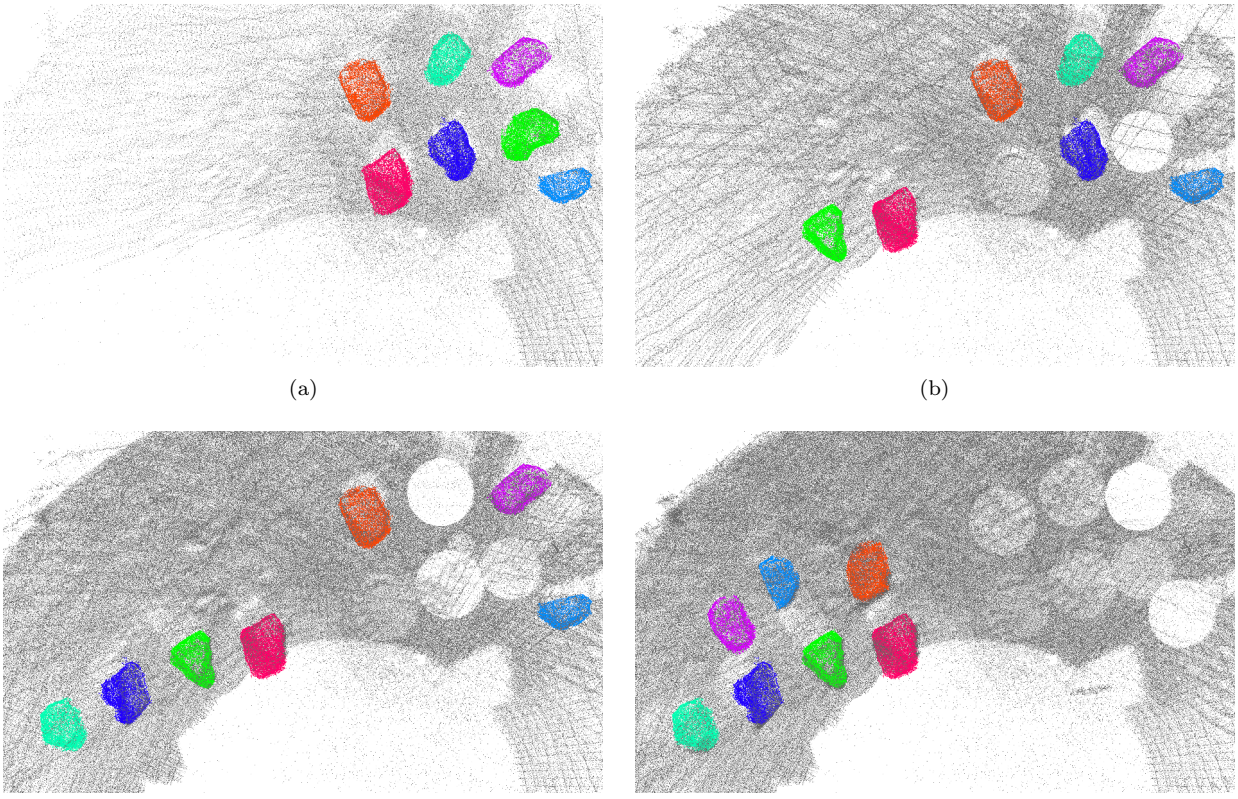


Figure 10: Mesh reconstruction from point cloud data of stone 1 (left) and stone 6 (right). Stone 1 has a rather roundish shape, whereas stone 6 is flatish.

the final filtering criteria, we can observe the largest impact on the cost for the selected grasps (see Fig. 9c). We see that the sequential filtering improves all cost criteria while maintaining a balance between them, leading to a success-promising grasp selection.

Four grasps out of 35 had to be adjusted manually because the selected grasping pose could not be executed successfully (slippage of the stone). Especially for flat but relatively thin objects, the grasp detection provided only a small number of grasp hypotheses (around 10) that are labeled force closure. This is due to the noise in the point cloud and the fact that the grasp has to be placed close to the ground. Thus, the subsequent filtering and ranking selects the grasp based only on where the TCP is closest to the centroid of the object without considering the other criteria.



(a) (b)

(c) (d)

Figure 11: State of the excavator's map at four different instants of the second experiment: (a) before moving any stones, (b) after moving the 2nd stone (c) after moving the 4th stone and (d) after moving the 7th stone. The object instances are effectively tracked (note that same color is always associated with same stone across all figures) and realigned with the map when moved by the robot.

6.4 Dynamic Object Handling

Fig. 11 depicts the state of the map overlaid with the segmented objects in four different instants during the course of the second experiment mentioned in Sec. 6.1. The illustrations qualitatively show that the object models are effectively tracked (the same color is always associated with the same stone across all four figures) and realigned with the LiDAR-based map after being moved. Although no ground truth data is available to quantify how precise the object models and their estimated locations in the map are, the high success rates achieved by the grasp pose planning module (see Sec. 6.3) indirectly indicate that the overall accuracy of the perception pipeline is sufficient for the task at hand.

7 CONCLUSION AND OUTLOOK

In this article, we have introduced an integrated perception and grasp pose planning system for autonomous manipulation of large-scale irregular objects with a robotic excavator. The core of the perception pipeline constitutes of a LiDAR-based mapping algorithm coupled with a segmentation approach that enables the detection of object-like instances in the observed scene. Furthermore, it is enhanced with multi-sensor data fusion capabilities, allowing for the registration of externally built maps of the regions of interest (e.g. 3D reconstructions from images captured by a drone-borne camera, as shown in the experiments), and with a database that helps keeping track of the identified objects' poses during manipulation. The grasp pose planning pipeline, on the other hand, is capable of sampling grasp hypotheses in a 3D scene given the information provided by the mapping system, i.e. a point-cloud map of the robot's surroundings and the segmented objects in it. Besides selecting the grasp hypotheses by predicted force closure, a custom filtering and ranking step is added in order to increase the grasp reliability. To the best of our knowledge, this is the

Figure 12: The presented mapping and manipulation tools in this paper were used for the application of dry stone wall construction with the autonomous hydraulic excavator HEAP. The in-progress wall is overlaid with a potential extension of the structure, generated by allowing the geometric planning solver to continue with additional digital stone models. Further details on the geometric planning of the desired stone poses can be found in (Johns et al., 2020).

first demonstration of such a level of autonomy in real experiments for a construction task on an architectural relevant scale.

The mapping, segmentation and grasping system presented in this work is a required step towards fully automated construction of utility structures such as noise protection walls, retaining walls, river bank reinforcements, or protection barriers. The described mapping and manipulation tools have already been used in an actual application to build a wall-like irregular stone-based assembly on an architectural scale (see Fig. 12). In this case, instead of simply placing stones in user-predefined positions, complete 3D reconstructions of the grasped objects are generated and the desired pose of each stone is computed immediately before placing. The reconstructed stone models are relocalized after being placed in the structure using the approach explained in Sec. 4.5, which allows to account for settling and unexpected deviations in the target structure. The geometric planning of the desired stone pose, which follows an iterative selection process that aligns the considered stone with the geometric constraints of the target shape and verifies stability in a physics engine, is beyond the scope of this work and described in more detail in (Johns et al., 2020). Note, however, that the mapping and manipulation tools presented in this work are task-agnostic and can be employed with different planners for autonomous construction and demolition applications.

A limitation of the current system is that the boulders need to be initially spread for accurate segmentation. While this is still a realistic scenario that can be achieved by careful unloading of the material on site, we aim at handling even more generic and practical situations in the future. Therefore, an important area of research will focus on identifying individual object instances in more challenging settings, such as piles of stones. For this, current work investigates the addition of a camera on the excavator's arm, which could potentially be used to map regions of the scene that are not visible by the onboard LiDARs and to introduce more advanced segmentation techniques leveraging both texture and geometry. In addition, tracking the object's pose in the gripper will be necessary to improve the execution accuracy of the computed plans.

On the manipulation side, our experiments showed that it is not realistic to achieve a perfect success rate on the first grasp attempt. Therefore, we plan to improve the autonomy of the grasp process with an automated strategy for adapting the grasp or performing complete re-grasping upon detection of stone slippage or dropping. This approach of attempting a grasp, observing if slippage appears, and adapt correspondingly is also inspired by how human operators grasp irregular objects.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research Digital Fabrication (NCCR DFAB).

References

Ardiny, H., Witwicki, S., and Mondada, F. (2015). Are autonomous mobile robots able to take over construction? A review. *International Journal of Robotics, Theory and Applications (IJR)*, 4(3):10–21.

Bellicoso, C. D., Gehring, C., Hwangbo, J., Fankhauser, P., and Hutter, M. (2016). Perception-less terrain adaptation through whole body control and hierarchical optimization. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*.

Bohg, J., Morales, A., Asfour, T., and Kragic, D. (2014). Data-driven grasp synthesis — A survey. *IEEE Transactions on Robotics (T-RO)*, 30(2):289–309.

Borst, C., Fischer, M., and Hirzinger, G. (2003). Grasping the dice by dicing the grasp. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics (T-RO)*, 32(6):1309–1332.

Dörfler, K., Sandy, T., Giftthaler, M., Gramazio, F., Kohler, M., and Buchli, J. (2016). Mobile robotic brickwork - automation of a discrete robotic fabrication process using an autonomous mobile robot. In *Robotic Fabrication in Architecture, Art and Design (ROB—ARCH)*.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the segmentation of 3D LIDAR point clouds. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Droeschel, D., Schwarz, M., and Behnke, S. (2017). Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner. *Robotics and Autonomous Systems*, 88:104–115.

Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., and Cadena, C. (2017a). SegMatch: Segment based place recognition in 3D point clouds. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Dubé, R., Gawel, A., Sommer, H., Nieto, J., Siegwart, R., and Cadena, C. (2017b). An online multi-robot SLAM system for 3D LiDARs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Eversmann, P. (2018). Robotic fabrication techniques for material of unknown geometry. In *Design Modelling Symposium*.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2013). Air-ground localization and map augmentation using monocular dense reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Fujisawa, R., Nagaya, N., Okazaki, S., Sato, R., Ikemoto, Y., and Dobata, S. (2015). Active modification of the environment by a robot with construction abilities. *ROBOMECH Journal*, 2(9).

Furrer, F., Novkovic, T., Fehr, M., Gawel, A., Grinvald, M., Sattler, T., Siegwart, R., and Nieto, J. (2018). Incremental object database: Building 3D models from multiple partial observations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Furrer, F., Wermelinger, M., Yoshida, H., Gramazio, F., Kohler, M., Siegwart, R., and Hutter, M. (2017). Autonomous robotic stone stacking with online next best object target pose planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Gawel, A., Dubé, R., Surmann, H., Nieto, J., Siegwart, R., and Cadena, C. (2017). 3D registration of aerial and ground robots for disaster response: An evaluation of features, descriptors, and transformation estimation. In *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*.

Glover, J. and Popovic, S. (2013). Bingham procrustean alignment for object detection in clutter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Grinvald, M., Furrer, F., Novkovic, T., Chung, J. J., Cadena, C., Siegwart, R., and Nieto, J. (2019). Volumetric instance-aware semantic mapping and 3D object discovery. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):3037–3044.

Grisetti, G., Kummerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43.

Guo, Y., Sohel, F., Bennamoun, M., Lu, M., and Wan, J. (2013). Rotational projection statistics for 3D local surface description and object recognition. *International Journal of Computer Vision (IJCV)*, 105(1):63–86.

Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Bohg, J., Asfour, T., and Schaal, S. (2014). Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2):51–65.

Holz, D., Ichim, A. E., Tombari, F., Rusu, R. B., and Behnke, S. (2015). Registration with the Point Cloud Library: A modular framework for aligning in 3-D. *IEEE Robotics and Automation Magazine (RAM)*, 22(4):110–124.

Jelavic, E. and Hutter, M. (2019). Whole-body motion planning for walking excavators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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

Jud, D., Leemann, P., Kerscher, S., and Hutter, M. (2019). Autonomous free-form trenching using a walking excavator. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):3208–3215.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research (IJRR)*, 31(2):216–235.

Kappler, D., Bohg, J., and Schaal, S. (2015). Leveraging big data for grasp planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Liu, Y., Choi, J., and Napp, N. (2019). Planning for robotic dry stacking with irregular stones. In *Conference on Field and Service Robotics (FSR)*.

McCormac, J., Clark, R., Bloesch, M., Davison, A. J., and Leutenegger, S. (2018). Fusion++: Volumetric object-level SLAM. In *International Conference on 3D Vision (3DV)*.

Mendes, E., Koch, P., and Lacroix, S. (2016). ICP-based pose-graph SLAM. In *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*.

Nikolic, J., Rehder, J., Burri, M., Gohl, P., Leutenegger, S., Furgale, P. T., and Siegwart, R. (2014). A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Petersen, K. H., Napp, N., Stuart-Smith, R., Rus, D., and Kovac, M. (2019). A review of collective robotic construction. *Science Robotics*, 4(28).

Prattichizzo, D. and Trinkle, J. C. (2016). Grasping. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, chapter 38, pages 955–988. Springer International Publishing.

Qin, T., Li, P., and Shen, S. (2018). VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics (T-RO)*, 34(4):1004–1020.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Saboia, M., Thangavelu, V., Gosrich, W., and Napp, N. (2018). Autonomous adaptive modification of unstructured environments. In *Robotics: Science and Systems (RSS)*.

Schonberger, J. L. and Frahm, J. M. (2016). Structure-from-Motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shan, T. and Englot, B. (2018). LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

ten Pas, A., Gualtieri, M., Saenko, K., and Platt, R. (2017). Grasp pose detection in point clouds. *International Journal of Robotics Research (IJRR)*, 36(13-14):1455–1473.

ten Pas, A. and Platt, R. (2015). Using geometry to detect grasp poses in 3D point clouds. In *International Symposium on Robotics Research (ISRR)*.

Tombari, F., Salti, S., and Stefano, L. D. (2010). Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)*.

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *IEEE International Conference on Computer Vision (ICCV) Workshops*.