Special Issue: The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2020

Regular Article

# Towards new frontiers in mobile manipulation: Team CTU-UPenn-NYU at MBZIRC 2020

**Petr Štibinger[1,*], George Broughton[2,†], Filip Majer[2,†], Zdeněk Rozsypálek[1,*], Anthony Wang[3,‡], Kshitij Jindal[3,‡], Alex Zhou[4,§], Dinesh Thakur[4,§], Giuseppe Loianno[5,‡], Tomáš Krajník[2,†] and Martin Saska[1,*]**

[1]Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague
[2]Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague
[3]Department of MAE, Tandon School of Engineering, New York University
[4]GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA
[5]Department of ECE and MAE, Tandon School of Engineering, New York University

**Abstract:** In this paper we present an autonomous robotic system for picking, transporting, and precisely placing magnetically graspable objects. Such a system would be especially beneficial for construction tasks where human presence is not possible, e.g. due to chemical or radioactive pollution. The system comprises of two primary components – a wheeled, mobile platform and a manipulator arm. Both are interconnected through an onboard computer and utilize various onboard sensors for estimating the state of the robot and its surroundings. By using efficient processing algorithms, data from the onboard sensors can be used in a feedback loop during all critical operational sections, resulting in a robust system capable of operating on uneven terrain and in environments without access to satellite navigation. System functionality has been proven in Challenge II of the MBZIRC 2020 competition. The Challenge required a ground robot to build an L-shaped structure of colored bricks laid in a predefined pattern. Such a mission incorporates several demanding subchallenges, spanning multiple branches of computer science, cybernetics, and robotics. Moreover, all the subchallenges had to be performed flawlessly in rapid succession, in order to complete the Challenge successfully. The extreme difficulty of the task was highlighted in the MBZIRC 2020 finals, where our system was among the only two competitors (out of 32) that managed to complete the task in autonomous mode.

**Keywords:** mobile manipulation, navigation, wheeled robots, construction

[*]http://mrs.felk.cvut.cz
[†]https://chronorobotics.fel.cvut.cz/en
[‡]https://wp.nyu.edu/arpl
[§]https://www.grasp.upenn.edu

Received: 30 September 2020; revised: 17 February 2021; accepted: 26 April 2021; published: 14 March 2022.

**Correspondence:** Petr Štibinger, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Email: stibipet@fel.cvut.cz

## 1. Introduction

In recent years mobile robotics has dramatically risen in popularity. Compact, yet high-fidelity onboard sensors, complemented with powerful, embedded computers, have allowed machines to comprehend their operational environment in real-time. The advent of deep learning, supported by the available datasets, now enables deploying advanced machine perception and planning algorithms. All these factors, combined, have allowed robots to handle increasingly complex tasks and opened new possibilities for future applications and development. The combination of these advances in hardware and software is visible throughout the world of robotics, with notable, rapid advances over the last decade.

In the field of manufacturing and logistics, robots have seen rapid proliferation. Robotic manipulators are no longer built for specific tasks but have become more versatile and easy to reconfigure (Coppola et al., 2014). Similarly, the growth in e-commerce has led to vast investments in developing robots for warehouse environments, more specifically automated storage and retrieval of products on demand (Bogue, 2016). Moreover, robots are gradually leaving the structured environments of warehouses and assembly lines, and are now entering those of households and offices (Hawes et al., 2017; Triebel et al., 2016). Semi-autonomous cars and intelligent vacuum cleaners are now considered common.

However, such machines are still built for a single purpose only, and the vision of a versatile, multi-purpose robotic assistant remains unrealized. Such an assistant must be able to navigate, unaided, around cluttered and changing environments, and must be able to recognize, inspect and manipulate common household and office items. In addition, systems capable of mobile manipulation are inherently more complex, which affects their reliability and, subsequently, safety. Design and implementation of a safe, reliable, and fast mobile manipulator require a tight coupling of control between mobile base and manipulator since the navigation and manipulation capabilities are contrary to each other: Good manipulation ability requires a large robotic arm with multiple joints, which results in the need for a large mobile base. However, safe navigation and precise position control of a bulky mobile base are difficult and precisely positioning the lower base affects manipulation capabilities. Therefore, mobile manipulators deployed outside of the structured worlds of assembly lines and warehouses are still somewhat clumsy to operate without human supervision. This dilemma becomes even more prominent if the robots have to operate for extended periods of time (Hawes et al., 2017; Kunze et al., 2018). Finally, the need for human operation or supervision often makes the mobile manipulator economically unfeasible.

The issue of economical viability changes, if one imagines a supervised, multi-robot system, where the human can assist remotely in difficult tasks, or take control in hazardous situations. Such a system could be used for a variety of tasks where people might be exposed to unnecessary danger, e.g., search and rescue in underground environments (Rouček et al., 2019; Petrlík et al., 2020), clearing areas of toxic or radioactive debris (Amjadi et al., 2019; Nawaz et al., 2009), or tall building construction and maintenance (Moon et al., 2013). However, multi-robot systems capable of reliable mobile manipulation are even more complex and their design and implementation require solving numerous engineering challenges. Nevertheless, preliminary versions of such systems have been successfully deployed in field trials, such as during the Mohammed bin Zayed International Robotics Challenge (MBZIRC) 2017 (Spurný et al., 2019; Schwarz et al., 2019).

In this paper we present a mobile manipulator[1] operating within a heterogeneous, multi-robot system and capable of collecting building blocks of various sizes, delivering them to a specified location, and arranging them in a desired pattern. While the system comprises aerial as well as ground robots, all capable of object manipulation, our paper focuses solely on the ground robot that, unlike its flying partners, can carry several building bricks at once and arrange them in a specific pattern. Nevertheless, having a multi-robot system deployed in a shared arena provided an excellent opportunity for employing a cooperative approach. In the developed system, a special effort

[1] Multimedia material: http://mrs.felk.cvut.cz/fr2020ugv

was put into multi-vehicle cooperation in both air-air and air-ground robot pairs. The performance of the system was evaluated in a series of experimental trials, culminating with the "Brick challenge" of the MBZIRC 2020[2]. This event aims to push the current boundaries of mobile robotics in a set of demanding missions directly relevant to real-world applications, in this case, automating construction operations.

## 1.1. Related work

Mobile manipulators are becoming an increasingly discussed topic in industrial automation. The concept of a fourth industrial revolution, dubbed Industry 4.0, anticipates mass deployment of complex cyber-physical systems, machine learning algorithms, and human-machine interaction (Lu, 2017; Lasi et al., 2014). Since many contemporary assembly lines are designed to serve a singular purpose, customization of the production process is often costly and time-demanding. In the near future mobile manipulators will become a vital part of highly reconfigurable assembly lines, smart factories and workshops (Xu et al., 2018; Petrasch and Hentschke, 2016; Roblek et al., 2016; Davis et al., 2012).

Another domain for mobile manipulator deployment is precision agriculture. Unlike industrial applications, the outdoor environment is naturally less structured and the robots have to deal with variable lighting and weather conditions (Duckett et al., 2018). The authors of (Bac et al., 2014) reviewed over 50 mobile manipulators systems, which were intended to harvest high-value crops. They concluded that none of the systems were commercialized mainly due to the lack of reliability when deployed in real world conditions.

During the last 5 years a few authors have demonstrated that hardware and software designs that address the perceptual difficulties outdoors (Kusumam et al., 2017; Ponnambalam et al., 2020; Binch et al., 2020) can lead to systems capable of long-term deployments in agricultural scenarios (Pretto et al., 2020). A recent report provides an overview on fruit harvesting robots, which have come to the market over the last 5 years (Bogue, 2020), and concludes that the time of mobile manipulation in agriculture is near. However, the report also indicates that the manipulators are designed for specific fruits only and that the environment has to be tailored for the robots to operate reliably.

Using a mobile base dramatically expands the operation space of the manipulator, however, it also imposes much stricter safety requirements, as the manipulator may no longer be contained within a safety fence. To ensure a safe environment around the manipulator, additional sensors and control algorithms have to be used. Rather than having a human worker avoid the workspace of the robot, a *smart* robot will continuously adjust its workspace to avoid the human (Thoben et al., 2017).

Commercially available mobile manipulators include the KUKA KMR iiwa[3], KUKA KMR Quantec[4], Robotnik RB-1[5] or the Fetch Robotics Mobile Manipulator (Wise et al., 2016). However, at the time of writing this paper, industrial mobile manipulators remain predominantly a subject of academic research.

In (Bischoff et al., 2011), a 5 DOF manipulator on a mobile platform with omnidirectional motion capability is shown to navigate through a semi-structured environment, locate and pick up colored objects. The dexterity of a mobile manipulator is analyzed by (Chen et al., 2018; Dömel et al., 2017), who both consider the task of grasping objects from a shelf. Heavy payload transportation is addressed in (Ohashi et al., 2016). In (Pavlichenko et al., 2018), the KMR iiwa is tasked with fetching various equipment from a warehouse, while dynamically avoiding humans who occupy the same area.

Given the topicality of mobile manipulation missions, they were included in two of the challenges in MBZIRC 2017 and one in MBZIRC 2020. In particular, a ground robot scenario in MBZIRC

[2] http://mbzirc.com
[3] https://www.kuka.com/en-de/products/mobility/mobile-robots/kmr-iiwa
[4] https://www.kuka.com/en-de/products/mobility/mobile-robots/kmr-quantec
[5] https://robotnik.eu/products/mobile-manipulators/rb-1/

2017 required a mobile manipulator to autonomously locate and close a valve. After identifying the valve, the robot had to examine it and determine the size of a wrench tool needed to turn it, retrieve the right wrench from a toolbox, and finally, use it to turn the valve. A detailed description of the challenge is provided in (Schwarz et al., 2019).

The “Brick building” challenge can be imagined as a successor to the “Treasure hunt” challenge of the first MBZIRC contest, which took place in 2017 in Abu Dhabi. In the “Treasure hunt”, a group of Unmanned Aerial Vehicles (UAVs) had to locate and collect a set of circular disks and deliver those to a specific location (Spurný et al., 2019; Baca et al., 2019; Loianno et al., 2018). Several teams managed to score points by collecting and delivering at least one disk in 2017, therefore, the difficulty of a mobile manipulation task was increased for MBZIRC 2020. This time the challenge required simultaneous deployment of ground and aerial robots, and the objects had to be picked up and placed with considerable precision.

### 1.2. Contribution

Here we present a complete, ground-based system for autonomously localizing, grasping, transporting, and precisely placing magnetic objects. Although the primary objective of the system was to compete in the Brick challenge of MBZIRC 2020, our design is highly modular and may be easily adapted for other similar tasks.

The developed computer-vision algorithms provide fast and reliable feature detection using diverse types of input data. We show how to modify the fast segmentation procedure (Krajník et al., 2014) to reliably process both depth and RGB data to detect and localize both component bricks and target building pattern, respectively.

Since the competition runs from dawn till dusk, special effort was put into making the algorithms resilient to varying lighting conditions. We have built upon our previous experience from the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge (Rouček et al., 2019), where a team of robots autonomously explored an unknown area with reduced visibility. Through the use of multiple perception principles, our design can locate, load, and transport bricks even in complete darkness.

We have also tackled the entire task in a cooperative manner, with an approach utilizing the unique aspects of both robot types. We exploit UAV’s speed and ability to provide an elevated, multi-perspective view of the arena, and the Unmanned Ground Vehicle (UGV)’s high payload capacity. By the means of multi-robot collaboration, each vehicle can directly benefit from the others’ strengths.

The MBZIRC 2020 Brick challenge includes many subtasks, some of which are well-known and studied problems in robotics. However, chaining these in rapid succession and ensuring a robust performance outside laboratory conditions still poses a difficult challenge. To further emphasize the difficulty, we point out that out of 19 teams participating in the Challenge, only two managed to score points while operating their ground robots in autonomous mode. As the winners of the event, we believe that sharing our experience will make a valuable contribution to the community. Moreover, we decided to share the source codes of the system (Broughton et al., 2020) to facilitate the process of new teams becoming a part of the MBZIRC community before its next round in 2022–2023.

## 2. Task description

The Challenge goal is to employ a team of robots, consisting of up to three UAVs and one UGV, to build a multi-layer structure using colored bricks. While the available building material and construction sites are divided for the two types of robots, they can benefit from sharing information about the arena. This proved to be essential to achieve the contest goals within the specified time. Therefore, we can perceive the Challenge as two loosely coupled tasks, which could be performed independently in a shared arena. We will focus solely on the UGV task in the scope of this paper, while the UAV system is described in (Baca et al., 2020).

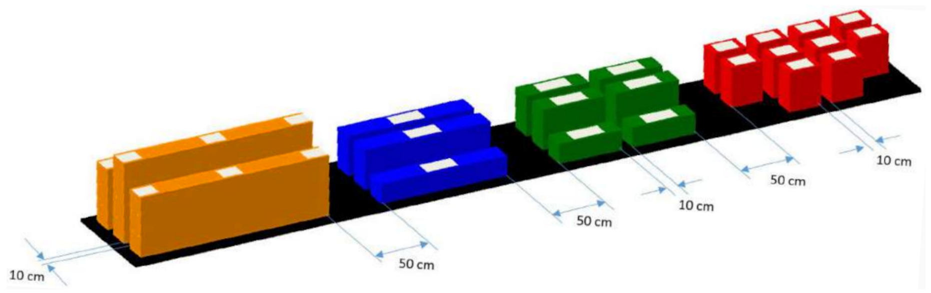
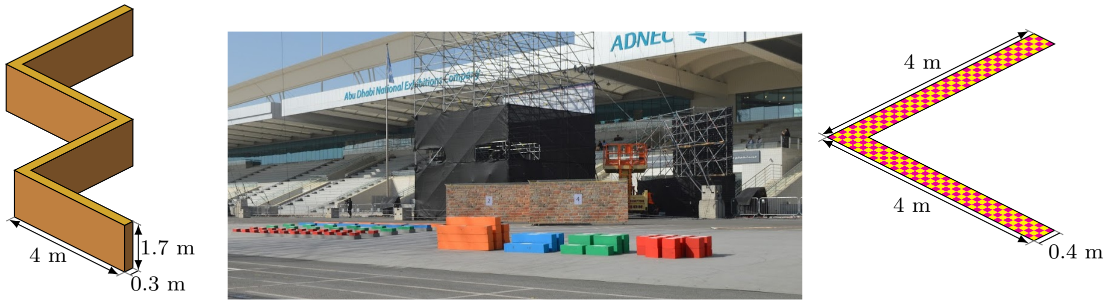


**Figure 1.** Layout of the brick pickup area for the UGV, as visualized in the competition rules.



**(a)** UAV construction site **(b)** Competition arena layout **(c)** UGV construction site

**Figure 2.** An overview of the competition arena showing the four objects of interest. The brick array for the UAVs is visible on the left, the UAV construction site in the background, the UGV brick array in the foreground, and the UGV construction site on the right side of the photo. One of the pillars, which supports the protective net, can be seen behind the UAV construction site.

The competition arena has a rectangular ground plan, approximately 50 m $\times$ 60 m in size. It is enclosed in a protective net, which is suspended by four pillars located in the corners, reaching a height of 20 m above the ground plane. The arena contains four distinct structures, which are shown in Figure 2b. These include two arrays of stacked bricks and two areas for brick placement, one for each robot type. In the context of the UGV task, the UAV-related structures are treated as obstacles. Nevertheless, the ground robot is still required to detect and identify these.

The bricks in the UGV pickup area are arranged in a predefined pattern shown in Figure 1. The layout of the pickup zone remains unchanged throughout the competition. The bricks consist of various types of extruded polystyrene foam. Each brick is equipped with a white ferromagnetic plate located on the top side, and all bricks have an identical cross-section of 0.2 m $\times$ 0.2 m. There are four different classes of bricks, which vary in their properties, and also in the score awarded for successful placement. The dimensions and scores for each brick class are summarized in Table 1. The pickup area is divided into four sections corresponding to the four brick classes. The bricks are predominantly stacked in two layers.

The placement area is a flat L-shaped surface covered in a yellow-magenta checkered pattern, as shown in Figure 2c. Both pickup and placement zones may be placed at an arbitrary position in the arena. The pickup zone for the UAVs consists of six narrow channels filled with one layer of bricks. Finally, the placement area for the UAVs is an elevated platform, consisting of four 4 m long segments arranged in a W-shaped pattern. The UAV placement area is shown in Figure 2a.

Before the Challenge starts, each team is given a uniquely generated sequence, which defines the color sequence of the wall to be constructed. One segment of the L-shape is dedicated to the largest, orange bricks. The remaining bricks are stacked on the other segment. Each layer of the wall consists

**Table 1.** A table of available construction material in the Challenge.

| Color | Length [m] | Weight [kg] | Score |
|---|---|---|---|
| Red | 0.3 | 1.0 | 1 |
| Green | 0.6 | 1.0 | 2 |
| Blue | 1.2 | 1.5 | 3 |
| Orange | 1.8 | 2.0 | 4 |

of exactly 2 orange, 1 blue, 2 green, and 4 red bricks. Finally, the time to complete the Challenge is limited to 30 minutes.

## 3. Platform description

The UGV platform selection was influenced by the Challenge rules, which limit the vehicle size to 1.7 m × 1.5 m × 2.0 m. The time constraints and arena dimensions also played a significant role in hardware design and strategy selection. It was in our interest to develop a robust system, which is not specifically tailored to one particular scenario. Instead, the developed system is highly modular and may be easily adapted to a variety of projects. For this reason, we opted for mostly off-the-shelf components, which have been used in our past projects (Petrlík et al., 2020; Rouček et al., 2019; Majer et al., 2019; Spurný et al., 2019; Loianno et al., 2018), or will be used in follow-up research in the near future.

Since the components used in the system are produced by various manufacturers, we rely heavily on the Robot Operating System (ROS) (Quigley et al., 2009). This open-source software platform provides hardware abstraction and serves as a middleware to interconnect individual components into a single large system.

### 3.1. Mobile base

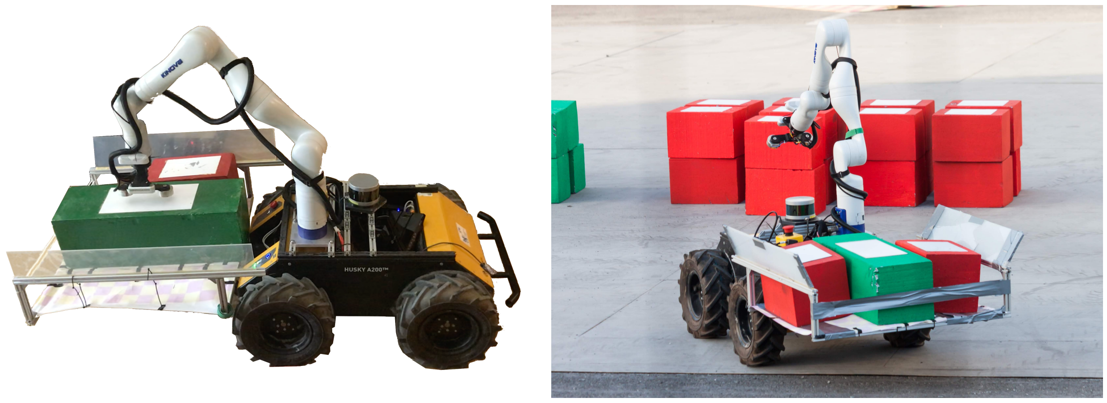
The Clearpath Robotics Husky A200[6] was selected as the mobile base for our system. The Husky is a rugged, wheeled robot, and a popular UGV platform for outdoor robotic applications and research. An open-source ROS interface (Clearpath, 2015) allows for easy interconnection of the base with other components, sensors, and modules. The Husky is well suited for cargo transportation, as it can carry up to 75 kg of additional payload. The compact form factor of the robot (0.99 m × 0.67 m × 0.39 m) provides enough margin to install additional components without exceeding the limits of the competition. A slight drawback of this platform is its limited top speed at approximately 1 m/s. On the other hand, the motors offer exceptional torque, and the rugged construction allows the robot to traverse difficult, uneven terrain.

To minimize the time spent by transitioning between the pickup and the placement area, the robot was equipped with a custom-designed cargo bay, which can fit up to 7 bricks of red, green, and blue color. This way, the UGV can load and transport an entire layer of the wall section in one trip. The cargo bay is attached to the rear section of the robot as shown in Figure 3.

### 3.2. Manipulator arm

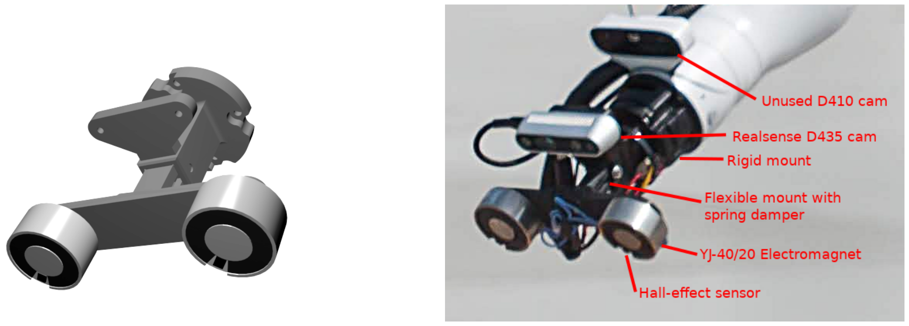
For the brick manipulation, we employ a manipulator arm by Kinova Robotics. A Gen3 Kinova[7] arm with 7 degrees of freedom (DOF) was chosen. The arm offers a working envelope, which is large enough to engulf all positions in the cargo bay, as well as all layers of bricks in the pickup zone. The added redundancy of the over-actuated manipulator also ensures exceptional dexterity of the arm. Thanks to these properties, the end effector can be oriented parallel to the ground at

[6] https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/
[7] https://www.kinovarobotics.com/en/products/gen3-robot



**(a)** Side view **(b)** Rear view

**Figure 3.** The complete mobile manipulator used during the competition, showing the Kinova robotic arm and the cargo bay attached to the Husky mobile base.



**(a)** CAD design **(b)** 3D printed and mounted

**Figure 4.** Detail of the custom-designed magnetic gripper for the end effector. The gripper is fitted with two electromagnets and Hall-effect sensors. A spring damper is installed inside the mounting bracket to provide force compliance during the grasping process. The gripper is controlled by an Arduino Nano board, and the rigid section also serves as a mounting point for the Realsense D435 camera.

all stages of the brick manipulation. This ensures equal distribution of load across the end–effector, and minimizes the risk of dropping the grasped object.

With a weight of 7.2 kg, the mounting point has to be placed near the center of mass of the Husky, to prevent tumbling when the arm is extended. Kinova also maintains an open-source Application Programming Interface (API) and ROS drivers, provided together in the Kinova Kortex[8] package. The Kortex API offers an interface for control in both joint space and Cartesian space, and internally prevents self-collisions. The arm is connected to the main onboard computer via Ethernet, and control commands may be issued up to 1000 times per second.

## 3.3. Magnetic gripper

The arm is fitted with a custom-built end-effector featuring a magnetic gripper. The gripper, shown in Figure 4, incorporates two YJ-40/20 solenoid electromagnets. We fitted each magnet with a Hall-effect sensor that serves as a proximity detector for the ferromagnetic plates. Each magnet is rated to operate at 12 V, and provides up to 25 kg of holding force. In our setup, the magnets are

[8] https://github.com/Kinovarobotics/kortex

overcharged by an input voltage of 24 V, which increases the holding force, but makes the gripper prone to overheating. The body of the gripper is 3D printed from PET-G, which becomes malleable at temperatures exceeding 100°C. With increasing temperature, the electrical resistance of the electromagnets also rises, and the holding force decreases as a result. Therefore, the gripper is only powered on for short periods of time during brick manipulation, to balance out the effects of heating.

The power switching and sensor readout are performed by an Arduino Nano board, which is connected to the main computer via USB. Based on our experience with the system, we have also incorporated a redundant software safety switch for the gripper. The mechanism cuts the power to the magnets if the Hall-effect sensors do not detect any nearby ferromagnetic object within 5 seconds of the magnets being powered on.

We have utilized our previous experiences with magnetic object grasping by aerial vehicles from the MBZIRC 2017 (Spurný et al., 2019; Loianno et al., 2018), and incorporated a spring-damper into the gripper body. The force-compliant design allows the gripper to retract by approximately 2 cm into the mounting socket. This feature mitigates the effect of communication delay within the system and prevents damage to the manipulator or the brick.

The proximity of a ferromagnetic object is detected as a rapid change in the magnetic field measured by the Hall-effect sensor. During the rehearsals, the threshold value is calibrated specifically for the competition bricks, as it depends on the material of the plate and its coating paint thickness.

We have gathered a lot of valuable insight on gripper design during our preparations, and also during the contest itself. The force-compliance allowed us to avoid damage to the equipment caused by communication delay between the gripper and the arm during the grasping phase. However, our design only allows one-axis deformation, which requires the ferromagnetic plate and the magnets to be aligned perfectly. Since the bricks could have been placed on an inclined surface, we have implemented a workaround by making the outer shaft of the gripper wider than the retracting element. This way, the inner core had enough margin for a lateral motion to compensate for the angular misalignment.

Robustness may also be improved by fitting a piezoelectric force sensor directly into the gripper shaft. The additional force feedback, to complement the torque sensors inside the arm, would significantly improve contact and collision detection. Further improvements could be made by measuring lateral forces acting upon the gripper, and by adding a temperature sensor to monitor the status of the magnets.

### 3.4. Sensory equipment

The competition took place in an outdoor environment, and the rules permitted the use of Global Navigation Satellite System (GNSS). The use of a more precise RTK/DGPS was also allowed but resulted in a score penalty. Conventional GNSS methods, however, tend to drift in the order of meters without corrections provided by an additional ground station. Moreover, the accuracy provided by a conventional GNSS system is not sufficient in the critical stages of the challenge. To ensure a robust, high-precision navigation, we have opted for a local positioning system based around the onboard sensors.

The UGV is equipped with a Velodyne Puck VLP-16 Light Detection And Ranging (LiDAR). The sensor consists of a rotating infrared laser rangefinder, which provides a 360 degrees horizontal field of view (FOV) and a resolution of up to 0.1°. The vertical FOV spans 30 degrees and is divided into 16 layers. The sensor has an advertised maximal range of 100 m. In practice, we have found the maximum reliable range to be just under half the advertised value, which was still sufficient for the competition arena. Measurements are streamed to the onboard computer via an Ethernet link at a rate of 10 Hz. The output data is represented as a 3D point cloud.

The Kinova Gen3 manipulator is equipped with an Intel Realsense D410 stereo camera integrated into the wrist. This camera is connected to the internal controller of the arm, which is connected to the main computer via an Ethernet link. During the development, we have experienced a fluctuating delay in the image output. The use of the wrist camera was intended in a direct feedback loop to align the end effector with the center of a brick. However, image delay often spiked into several seconds, which rendered the image stream unusable for such a task. Therefore, an additional Intel Realsense D435 stereo camera was attached to the gripper and connected directly to the main computer. The D435 provides an RGB image at a resolution of up to 1920 $\times$ 1080 pixels, and a depth image of up to 1280 $\times$ 720 pixels at distances ranging from 0.1 to 10 m. The depth sensor uses a global shutter, which prevents image distortions induced by the motion of the camera or the scene.

Prior to the competition, a high-resolution 3D scan of the arena and its surroundings was taken by a Leica BLK360 3D scanner. The point cloud produced by the 3D scanner is used to augment the scan matching of the LiDAR measurements, to improve localization of the robot within the arena.

## 4. Control architecture

In this section we summarize the control architecture of the mobile manipulation system. As mentioned before, the system is capable of completing the task described in Section 2 in autonomous mode. However, the extent of autonomy can vary widely between different systems. Let us first discuss the task complexity, and anchor the degree of autonomy required for successful wall placement.

The area of operation is assumed to be a semi-structured environment, consisting of a large, uneven ground plane and very sparsely distributed obstacles. Throughout the operation, the area is unpopulated by human workers. Besides the UGV, the only other moving entities are the UAVs, which operate outside the reach of the ground robot for most of the time. Nevertheless, the environment cannot be assumed entirely static, as the brick stacks and the build zones are relocated by human workers in between the trials. Since humans cannot achieve the same level of precision robots do, it is assumed that the layout of the stacked bricks will deviate from the expected layout.

To deal with the challenges of changing arena configuration, the robot is required to explore the arena and search for the objects of interest. However, even with the use of high-end sensors and advanced processing techniques, the object detection range of the robot is limited, and so is the top speed of the mobile base. During the 30 minute time window, a full exploration of the arena becomes nearly impossible, especially when the time required for brick manipulation is also factored in. On the other hand, navigational waypoints cannot be prepared in advance without knowing the exact arena configuration. The waypoints may be set manually before the task is started, however, this action has to be performed without entering the arena. This method alone lacks the accuracy required for the object grasping to be successful.

In this work, we aim to strike the balance between an exhaustive full exploration, and the following of a predefined path. The concept of "guided autonomy" (Gray et al., 2018; Dellin et al., 2016; Hebert et al., 2015) assumes the robot to be capable of autonomous operation and reaction to sparse inputs issued by an operator. This concept is suitable for robotic operations, where direct human presence is not possible and communication with the robot is costly or unreliable, but the operator still has a crude knowledge of the environment. Similar applications include the inspection of a damaged nuclear power plant (Bausys et al., 2019; Notheis et al., 2012), or space exploration by unmanned rovers (Francis et al., 2017; Starek et al., 2016).

The necessary computing power is provided by a primary onboard computer, the Intel NUCi7 running Ubuntu Linux 18.04 with ROS version Melodic. The computer handles all sensor data processing, state estimation, and a majority of the control logic.

As the overall task comprises multiple complex subtasks, we have implemented a hierarchical control strategy. The strategy divides the system into 4 layers. Each layer of control has direct access to the services provided by the lower layer, and the action results are propagated to the upper layer. The structure is illustrated in Figure 5.

At the bottom, we have the internal controller of the Kinova manipulator commanded via the Kortex API. The ROS drivers for the Realsense camera and the Velodyne LiDAR are also located at this layer. Finally, this layer also handles the sensor data preprocessing, such as image

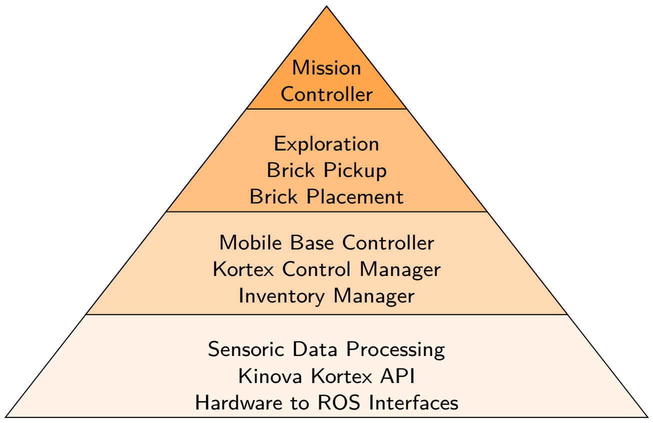


**Figure 5.** An illustration of the control hierarchy employed by the onboard computer of the UGV. Each subsystem is able to directly access the services provided by the lower layer subsystems and propagate results to the upper layer.

histogram normalization, noise filtering, and segmentation. The preprocessing is based on fast, efficient algorithms (Krajník et al., 2018; Krajník et al., 2014), which are application-independent and do not rely on the rest of the system.

The second-lowest layer comprises wrappers and abstractions for motion control. The Husky mobile base is controlled via the ROS move_base[9] navigation stack, and the Adaptive Monte Carlo Localization (AMCL) probabilistic localization system[10]. The original code has been adapted for operation in larger areas, such as the MBZIRC 2020 arena. This layer also includes a custom wrapper for the manipulator control interface, the Kortex Control Manager (KCM). The contents of the cargo bay are supervised by the Inventory manager, which also processes the construction blueprint provided at the beginning of the challenge.

The third layer consists of three state machines, which handle the long-duration phases of the mission. The arena exploration is focused on locating the stacked bricks and the placement area. During the exploration phase, the robot also builds a symbolic map of the arena and avoids obstacles. The brick pickup phase deals with moving close to the brick stacks, and fully loading the cargo bay with four red, two green, and one blue brick. The brick placement phase handles the alignment of the robot with the checkered pattern and deploying the bricks from the cargo bay in the order of the desired blueprint.

Finally, the Mission Controller ensures valid transitions in between the individual phases. The main loop of the Mission Controller is shown in Figure 6. The lower level controllers detect and attempt to resolve minor failures, such as an unsuccessful brick grasping attempt. Severe failures are propagated to the Mission Controller, and cannot be resolved by the integrated recovery behavior. In such a case, a manual reset is requested. Severe issues are typically of mechanical nature and require human intervention before continuing with the mission.

## 4.1. Arm manipulation

This section summarizes the operation of the Kortex Control Manager, a custom control layer built upon the Kinova Kortex API, which mediates communication between the arm and the main computer. In addition to self-collisions, which are avoided by the internal logic of the arm, the KCM also ensures collision avoidance with the structure of the Husky and the cargo bay. Payload

[9] http://wiki.ros.org/move_base
[10] http://wiki.ros.org/amcl

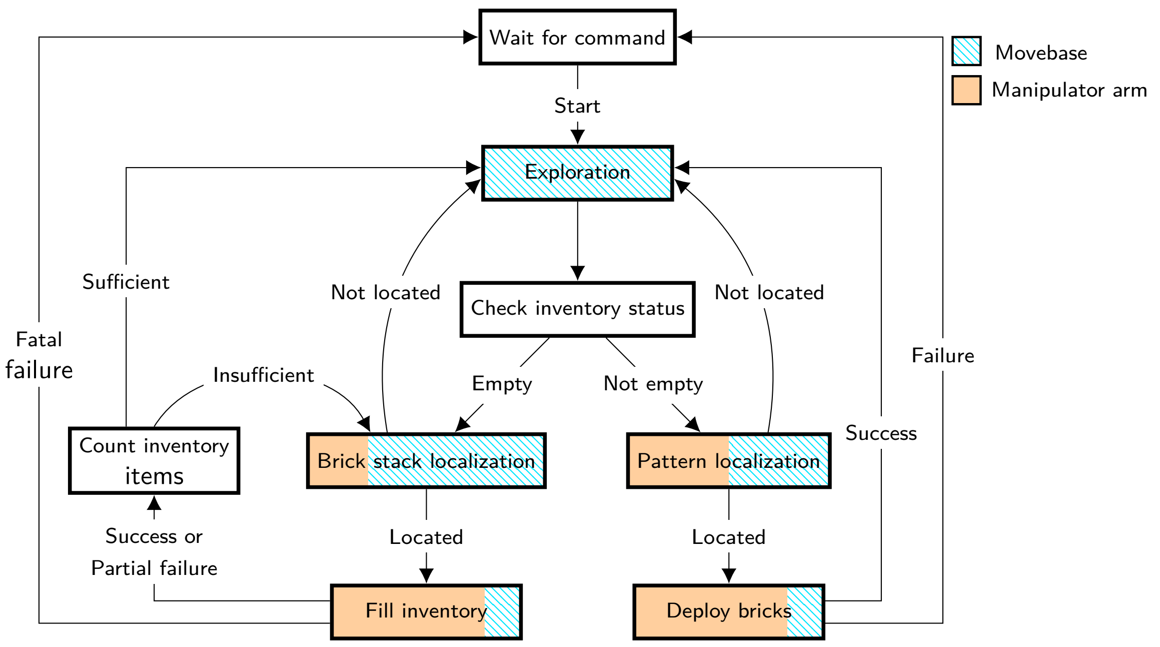


**Figure 6.** The core loop managed by the Mission Controller state machine. This controller represents the highest level of the system, and handles the transitions between the three main stages of the operation: Exploration, Inventory filling and Brick deployment. The tight cooperation of the movebase and the manipulator arm is highlighted by color coding. The colors represent, how much of the action is performed by the movebase and the manipulator arm.

awareness is also implemented, and additional avoidance maneuvers are performed if a brick is held by the gripper, or present in the cargo bay.

Two modes of operation are possible with the KCM – position control and velocity control. The KCM utilizes the output of the joint sensors, which provide the current joint position, velocity, and torque at a rate of 100 Hz. In both modes, the torque data is continuously analyzed, and the motion is terminated after the torque difference exceeds a given threshold. The torque feedback serves as the primary safety feature to prevent damage in case of unexpected collisions with obstacles.

The position control mode is used to move the arm into the desired position in joint space. In this mode, the trajectory from the current arm position to a goal position is interpolated by the MoveIt! planning interface (Chitta et al., 2012). The interpolated trajectory is sampled with 0.001 second time step, and the individual samples are sent as joint commands to the actuators via the Kortex API. The dense sampling allows the controller to terminate the motion abruptly, in case a collision is detected.

In the velocity control mode, the joint commands are used directly without interpolation. The commands are sent to the arm via the Kortex API at a rate of 100 Hz. Position feedback is used to constrain the motion inside the safety area. This prevents collisions with the Husky and the bricks in the cargo bay.

The core of KCM consists of a bank of short pre-programmed actions. Some of the actions are as simple as moving to a predefined position, while others utilize the Realsense camera or the magnetic gripper feedback to successfully align and connect with a brick. The actions are provided to the higher-level controllers as services. The internal state machine of the KCM ensures, that no more than one action is executed at the same time. The continuous monitoring of joint sensor data is used to detect anomalies, such as collisions with unexpected obstacles or missing bricks in the cargo bay. After an issue is detected, the KCM automatically performs a series of recovery actions to

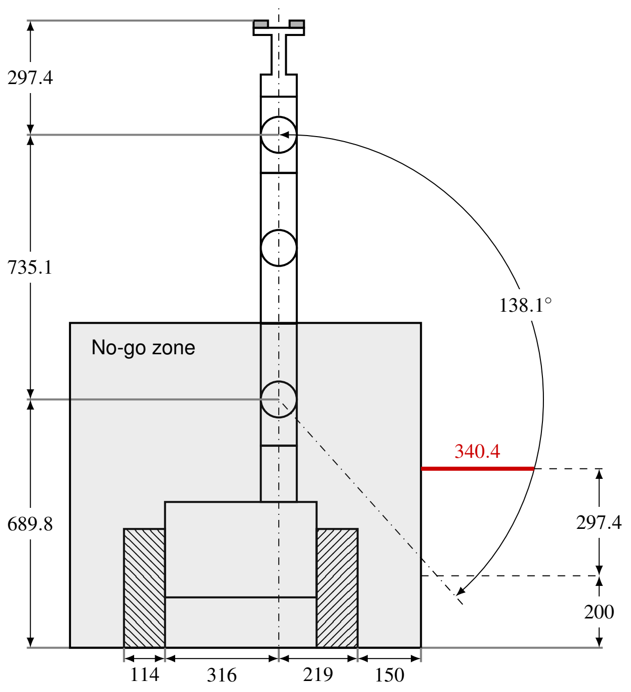


**Figure 7.** Reachability analysis for the lowest layer of bricks placed directly onto the ground plane. All dimensions are shown in millimeters unless stated otherwise. Since the gripper is required to be oriented perpendicular to the brick, only the reach of the wrist joint is considered. The area around the mobile base is declared a no-go zone for the wrist motion, to avoid collisions with the robot body. The no-go zone has to account for the brick size, since the grasping targets the brick center. The robot also requires a margin for turning during the navigation phase, which further limits the reachable area. The remaining part of the working envelope is highlighted in red. This highlighted area represents the reach of the wrist for picking up a brick off the ground plane. The limited reach generates a strict requirement on the precision of navigation for the mobile base.

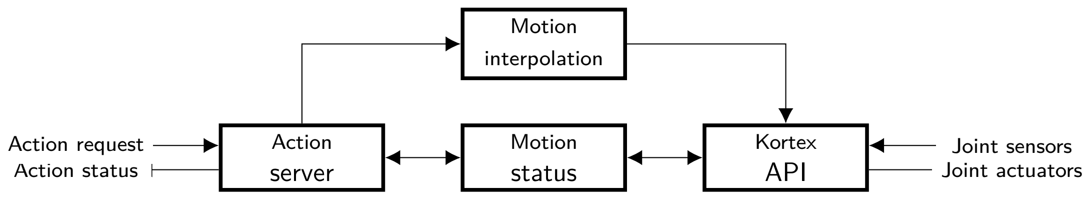


**Figure 8.** The internal structure of the Kortex Control Manager, a custom wrapper for the Kortex API. The Action server is provided with a bank of short, pre-programmed actions, which are specific to the brick grasping task. These actions range from assuming a predefined position, to launching a camera feedback control and aligning the end effector with a brick.

regain full control of the arm. If a brick is found missing or obstructed in the cargo bay, the issue is propagated to the higher levels of control, and a new strategy is selected. A block diagram of the KCM operation is shown in Figure 8.

Payload awareness is also implemented into the KCM. However, mounting a manipulator arm on top of a mobile robot introduces additional limitations. The problem of a reduced working envelope is illustrated in Figure 7. A significant portion of the area reachable by the manipulator is now obstructed by the frame of the robot and has to be declared a no-go zone. The zone must also be inflated by at least half a brick size, to account for bricks grasped by their center. Additional inflation is necessary to provide the mobile base clearance for safe turning without colliding with the bricks since the robot uses skid-steering. Finally, robotic arms typically struggle to reach positions below their base and lifting the base of the arm by the height of the mobile robot only makes the issue more pronounced.

In our case, we have employed a dexterous over-actuated manipulator with 7 DOF. We have also mounted the arm as close to the side of the robot as possible, without disrupting the stability of the mobile base. Despite our efforts, the lowest layer of bricks placed directly on the ground remains very challenging to pick up, as the reachable area at this height only has 0.34 m in diameter. This limitation needs to be compensated by the mobile base, which is required to approach the target within ±0.17 m.

## 4.2. Navigation and localization

The navigation stack used on the Husky is based on the ROS navigation packages implemented by the STRANDS projects (Hawes et al., 2017). The primary source of information for navigation is the 3D LiDAR sensor, in addition to the estimated wheel odometry. The incoming LiDAR data is processed to calculate each point's height relative to the robot. This data is then split into two horizontal point cloud slices. The first of these slices, with points low to the ground, is used for obstacle avoidance, while the other, concentrating on objects further from the ground, is used for localization.

The system uses a pre-processed high-resolution 3D point cloud of the arena, as mentioned earlier. A horizontal slice from this point cloud, which corresponds to the upper slice of the LiDAR data, is used to generate a map. The limits for the LiDAR slice were determined during the rehearsals for the competition by searching for the optimum in terms of localization reliability. This height level was set to 1.5 m above the sensor by empirical evaluation.

The robot's pose in the map is estimated from the LiDAR slice and wheel odometry via the AMCL. We found out that the default ROS AMCL module is quite sensitive to odometry errors, even when factoring in the corresponding variables for noise. Therefore, some effort was taken to correctly calibrate the odometry information for the overall robot structure, taking into account the asymmetric center of gravity caused by the positioning of the arm. Additionally, the odometry was especially sensitive to the dynamically changing payload in the cargo bay, which further offset the center of gravity, and disrupted the localization. We have performed an exhaustive tuning of the parameters using data collected during the preliminary tests and the competition rehearsals. Eventually, the parameters were tuned so that the localization remained very stable, even when driving in the center of the arena, far from reliable navigational reference points.

Path planning for the Husky is handled by the move_base ROS package. The planning parameters were calibrated to allow driving past obstacles, without getting too close. Dynamic maps are used in conjunction with the lower slice of the LiDAR point cloud to avoid potential collisions occurring from sudden events, such as the UAVs accidentally dropping bricks, or performing an emergency landing in the UGV's path. This slice of the point cloud includes points from 0.1 m above the ground up to the height of the LiDAR sensor. In this range, the UGV could safely and reliably avoid bricks on the ground while navigating. Points below the 0.1 m threshold are assumed to be located on uneven ground. The threshold provides a balance between detecting bricks on the ground, and not planning paths around false-positive detections caused by the non-uniform ground plane in the arena, or by the sensor noise.

Minor adaptations were made to the code, so that the precision of positioning may be changed on the fly. This feature significantly reduces the mission time, as the robot is often able to reach its destination within a reasonable tolerance, and then expends much valuable time attempting to get slightly closer to its designated target. This is especially pronounced if the robot is displaced laterally from its destination. In such a case it must perform a costly reverse, turn, and drive in the style of parallel parking. Therefore, it is important to evaluate whether such a maneuver is really necessary.

During the exploration phase, assuming good localization, it is not necessary to reach every target with a high degree of precision, and the emphasis is given to covering large distances as

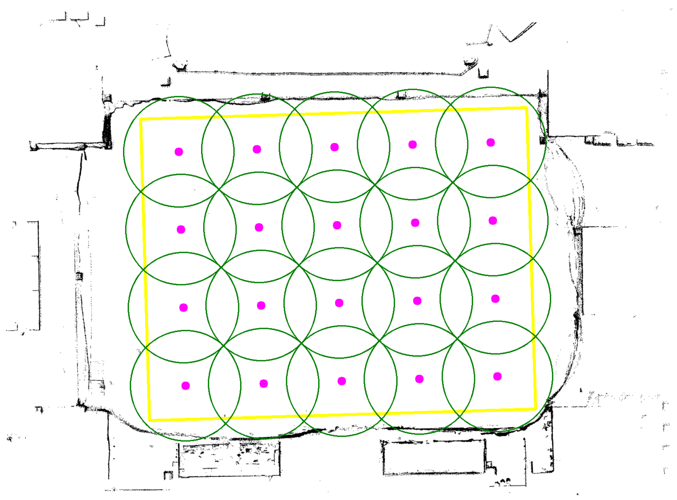


**Figure 9.** An exploration strategy generated for the MBZIRC 2020 competition arena. The black dots represent a pre-processed high-resolution 3D scan, which is used to augment the localization provided by the AMCL module. The yellow line marks a no-go zone the robot must not leave. Given the maximal range at which a checkered pattern can be reliably detected by the RGB camera (green circles), the system generated a set of $4 \times 5$ waypoints (purple points) that the robot must visit to ensure full area coverage. Note that for full coverage using circles, there is a great deal of overlap. This is somewhat for redundancy, however, in a time-critical challenge, great gains could have been made by switching to a rolling exploration strategy, rather than point-by-point.

quickly as possible. Therefore, the system detects the attempts at the "parallel parking" maneuver and prevents them by marking the current goal as reached. On the other hand, during the brick stack approach and pickup, the position tolerances are set very tightly to reach the target as accurately as possible. The motivation for the high degree of precision during the brick approach was explained in Section 4.1.

The exploration strategy is visualized in Figure 9. We input the width and height of the arena, as well as the maximum perception distance, and the exploration strategy is computed. A no-go zone is delineated around the perimeter, and a series of waypoints is generated. All the waypoints must be visited by the UGV to ensure the entire arena is fully explored. The exploration phase may be interrupted early if the objects of interest are found beforehand.

## 4.3. Inventory and Planning

While the robot's main control state machine is responsible for moving the robot from the pickup location to the deployment location and back again, we have implemented an additional module, which is responsible for the robot's inventory, planning, and strategy selection. The module tracks the bricks that the robot believes it is currently carrying, and also their place in the cargo bay. Other nodes can query the inventory module to see which bricks the robot currently has, which it needs to pick up, and which bricks can be accessed immediately.

When the robot reaches the stacked bricks, the inventory module determines which brick type should be picked up. The node processes the desired blueprint provided before each trial and constructs a dependency graph of the bricks. The module is aware that certain bricks need not be picked up if they cannot be placed at the building site. The dependency graph also takes into account the layout of the cargo bay. Doing so, it is able to determine that a certain brick may be picked up if and only if the robot first picks up another type of brick beforehand.

This module also allowed us to prepare multiple strategies for the competition. The competition rules heavily penalized teams for failing to place at least one brick. We opted for a strategy of initially going for a single brick and getting it placed as soon as possible, and then returning to pick up a full payload of bricks. Internally, this was done by simply restricting the inventory manager to only deem a single brick placeable at the building site initially.

Once the robot reaches the building site, it can poll the inventory specifically for which bricks are currently accessible by the arm, and at which position in the wall they should be placed at. The module then returns an array of instructions, for example, saying that a red brick stacked two levels high on the right side of the cargo bay is accessible, and can be placed on the second row of the wall at a certain position. The deployment position is determined from the blueprint as an offset from the right-hand side of the construction site.

The inventory manager allowed us to easily switch build strategies during the competition, but it was also very useful for debugging the system during the development. Furthermore, if the robot required a manual reset, this module would allow us to quickly feed in the current state of the world to the robot, e.g. a partially constructed wall. During the competition, most of the features of the module were not used due to time constraints. Nevertheless, the flexibility of the module contributed significantly to the development process and was useful for setting the robot into various test scenarios.

## 4.4. Brick stack detection (far)

The range of the Velodyne VLP-16 (up to 100 m) is utilized for the detection of the stacked bricks at a large distance. The LiDAR has a vertical FOV of 30° divided into 16 equally spaced linear scans.

Firstly, we estimate the maximal brick detection distance. To distinguish the brick from a ground plane or another obstacle, at least two scan layers are required to hit the brick. We assume an idealized scenario shown in Figure 10, with two rays hitting the brick side at the same angle. The intersection points and the sensor center form an isosceles triangle. This assumption simplifies the computation while still providing a fairly accurate approximation of the detection distance. From this assumption, the minimal number of rays hitting the brick can be computed as:

$$N = \frac{\arccos\left(1 - \frac{a^2}{2b^2}\right)}{\alpha}, \tag{1}$$

where $a$ is the height of the target object, $b$ is the ray length, $\alpha$ is the angular pitch of individual rays. The distance $d$ between the brick and the sensor is then computed as:

$$d = \frac{\frac{a}{4}}{\tan\left(\frac{\alpha}{2}\right)}. \tag{2}$$

For a single brick with a height of 0.2 m, at least two rays are guaranteed to hit the brick, if the UGV is within a 3.055 m radius. A priori knowledge about the layout of the brick stacks can be

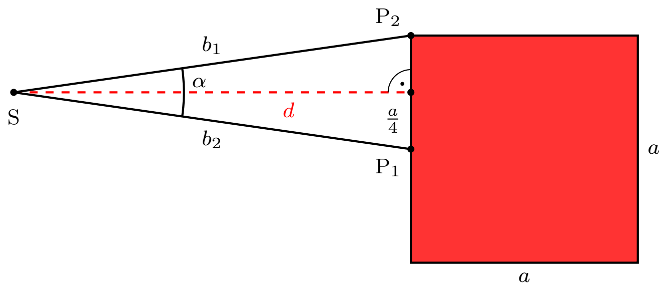


**Figure 10.** Approximation of maximal detection range is based on the vertical resolution of the LiDAR sensor. The brick cross-section is represented by a red square with a side length $a = 0.2$ m. The sensor is located at point S and $b_1 = b_2 = b$ represent two neighboring rays. The angle $\alpha = 1.875°$ is computed by dividing the horizontal FOV by the ray count. The distance $d$ serves as an approximation of the maximal detection range. At larger distances, the brick may be missed by both rays.

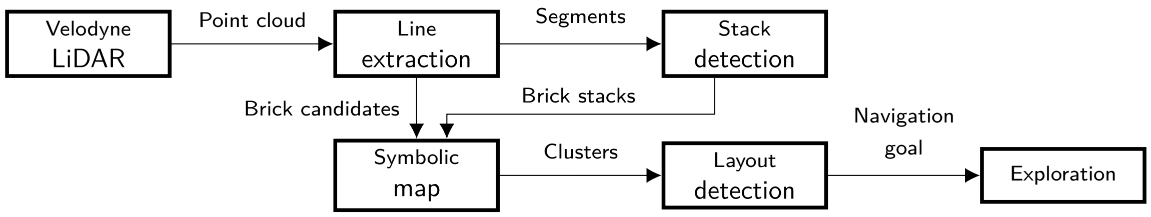


**Figure 11.** Pipeline for the brick stack detection from LiDAR data. The process outputs an estimated position of the brick pickup area in the map, which is passed into the exploration node as a navigation goal.

leveraged as well. All types of bricks are initially stacked in at least two layers. This increases the total height of the structure to 0.4 m and doubles the detection radius to 6.11 m. To achieve full coverage of the arena, a simple path over equally spaced waypoints would be generated, as shown before in Figure 9. However, for an arena of approximately 50 m × 60 m in size, such an endeavor would consume a significant portion of the limited time budget.

The idea of using an advanced classification method, such as a neural network, was scrapped during the development. Such classifiers, albeit very powerful, were deemed unnecessarily complicated for the detection of features, which appear as line segments in the LiDAR data. Instead, simpler methods based on line extraction are used. The brick stack detection pipeline is illustrated in Figure 11. The point cloud provided by the LiDAR is initially segmented using the Iterative End Point Fit (IEPF) algorithm (Ramer, 1972). Since the size of all brick classes is fixed and known in advance, only segments of a proper length are considered brick candidates.

This method alone would be sufficient and easily applicable under laboratory conditions. However, in the real world with an uneven ground plane, the algorithm tends to produce false-positive candidates. To address this issue, we have applied an unsupervised learning technique, the Expectation-maximization (EM) algorithm (Dempster et al., 1977), to filter out the erroneous classifications, and estimate the position and orientation of the brick pickup area.

We assume the density of brick candidates to be higher at the correct stack position. Moreover, the individual piles of colored bricks are arranged in a predefined order along a straight line, so searching for a sequence of multiple brick classes will dramatically improve the false-positive filtering. To leverage the spatial distribution of the bricks known a priori, the entire stack is represented as a Gaussian mixture model:

$$P(\vec{x}_m) \sim \mathcal{N}(\vec{x}_m,\, \vec{\mu} + k_m\vec{v},\, \mathbf{\Sigma}_{\boldsymbol{m}}), \forall m \in \{0, 1, 2, 3\}\ , \tag{3}$$

where $m$ is the brick class index, $\mathcal{N}$ is the Gaussian distribution, $\vec{\mu}$ is the mean of the model, $\mathbf{\Sigma}_{\boldsymbol{m}}$ is the covariance matrix of a brick class, $k_m$ is the scalar multiplier unique for each class, and $\vec{v} = (\cos\phi, \sin\phi)^T$ is a direction vector representing orientation.

The model provides the most likely estimate of parameters $\vec{\mu}$ and $\phi$, which represent the positions and orientations of the brick piles, respectively. Since the brick classes are clearly separated into individual piles, the model contains four distinct peaks, which correspond with the centers of the individual colored brick piles, as shown in Figure 12. The measurements $\vec{x}_m$ for the layout detection are clusters obtained from the symbolic map.

Since the brick candidates are already generated in the relative coordinate frame of the UGV, the position of the brick stack may be directly projected to the map as well. The mean of the model is found via the EM algorithm and directly represents the center of each colored brick pile. Additionally, the orientation of the stack is also estimated by the EM algorithm. This allows the robot to fully define the pose of the brick stack within the navigational map. The map is being continuously generated throughout the UGV's operation, and the model is continuously updated using new data.

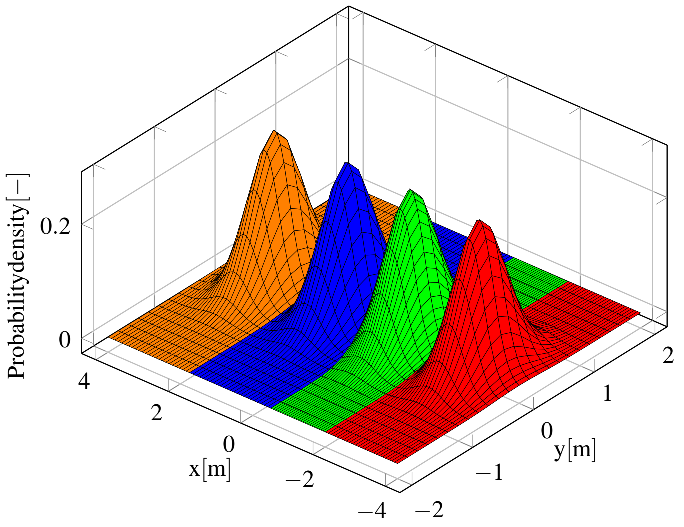


**Figure 12.** The probability density of the model used by the brick stack classifier. Exploiting the a priori knowledge of the brick stacks layout, false-positive classifications can be filtered out, which significantly improves detection quality and robustness.

## 4.5. Brick stack approach

The exploration phase ends with the robot roughly 4 meters away from the red brick pile. For the stack approach, much stricter precision parameters are uploaded to the navigation node. Due to the mechanical limits of the system, which were mentioned in Section 4.1, the maximum position error allowed for the move base is ±0.17 m. The brick pickup phase is controlled by a state machine, which is illustrated in Figure 13.

The approach is initiated by aligning the heading of the robot with the major axis of the brick stack. The axis of the brick stack is estimated from the LiDAR data. We once again utilize the a priori knowledge of the brick stack layout, and only take the slice of the point cloud, which fits the expected height of the bricks, and run a line-fitting algorithm based on the Random Sampling Consensus (RANSAC) method (Fischler and Bolles, 1981) on the point cloud slice.

Once the robot is aligned with the bricks, the arm is extended to the side in such a way, that the wrist camera points directly towards the ground plane. In the initial position, the distance between the camera and the ground plane is nearly 1 m. With a 86° FOV depth sensor, even a blue brick can fully fit into the image frame, if the arm is positioned above the brick's center.

Using the depth image stream, the mobile base then attempts to park next to one specific brick, so that the brick is located in the reachable zone of the arm. The target brick type is determined by the inventory manager based on the current world state and the construction blueprint. If multiple bricks are visible by the camera at the same time, the robot will always target the rightmost visible brick, as long as its type matches the demand. This behavior is illustrated in Figure 14.

After the alignment is completed, the arm attempts to grasp the brick and place it into the cargo bay. The state machine integrates multiple levels of fallback behavior, should any of the pickup stages fail. The pickup process is repeated until the inventory manager reports a sufficient number of bricks in the cargo bay. Afterward, the arm is used to secure the loaded bricks inside the cargo hold, and the robot returns to the exploration phase.

## 4.6. Brick detection, grasping and loading

The brick detection method is based on a library for fast segmentation presented in (Krajník et al., 2014), which was also successfully used in the "Treasure hunt" scenario of the MBZIRC

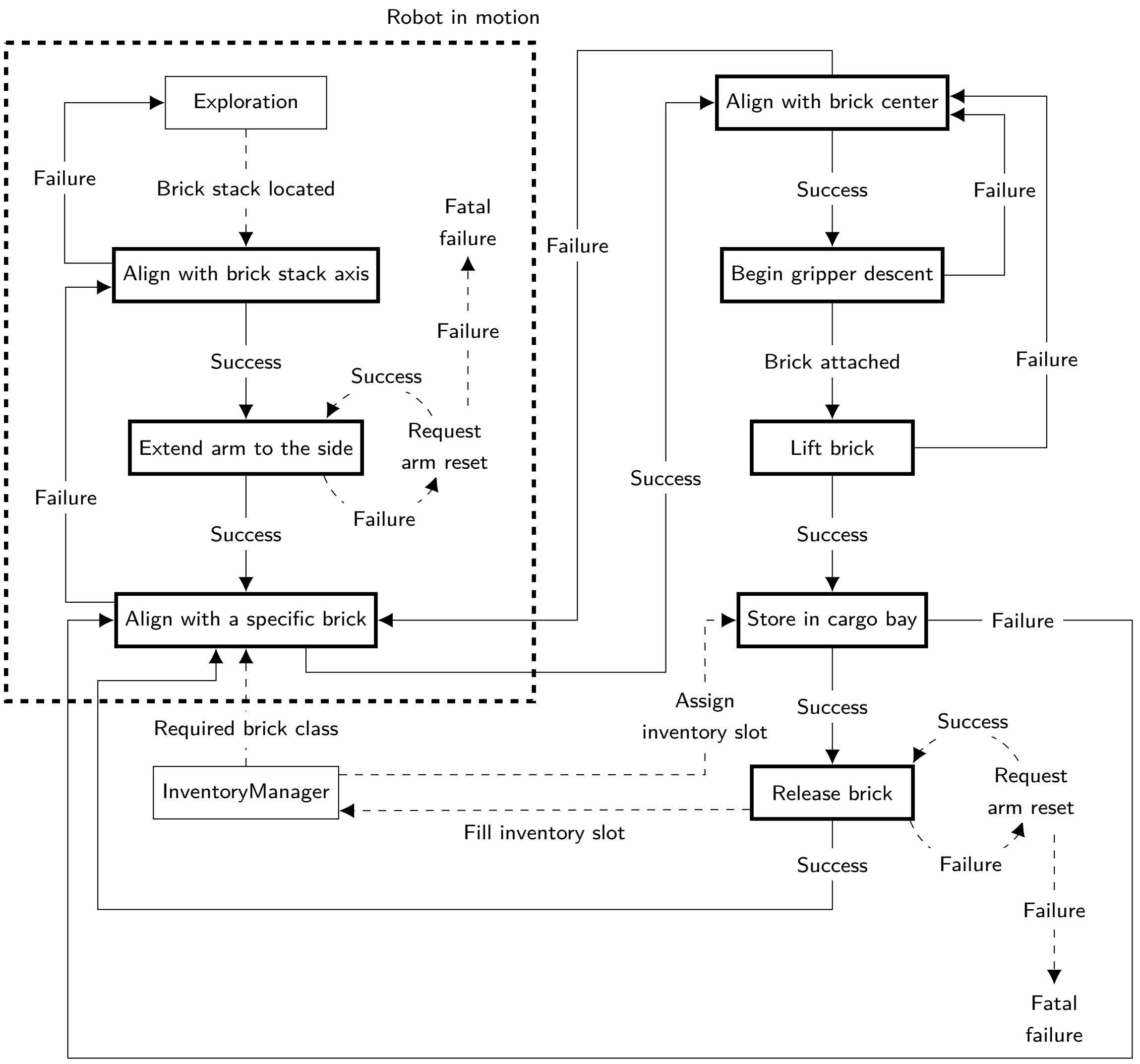


**Figure 13.** Diagram of the brick pickup procedure. The process is initiated after the exploration phase had finished, and the brick stack was located. The operation is controlled by a state machine running on the onboard computer. The entire system is in motion for the highlighted group of processes. The rest of the operations only uses the Kinova arm. Operations, which are handled by separate dedicated systems, and have been simplified for this illustration, are represented by dashed lines.

2017 challenge. However, the brick detection is easier compared to MBZIRC 2017, because the images do not suffer from vibrations induced by propellers, and the approximate position of the camera above ground was available all the time. Moreover, the robot was equipped with an RGBD camera, which provided stable depth information in direct sunlight as well as at night. Therefore, we decided that unlike in 2017, where the objects were detected based on their color, depth images will be used as the primary means of brick detection, as shown in Figure 14.

Using the known position of the end-effector, which is pointed to the ground during the brick pickup, we calculate the distance from the camera to the ground plane. Then, we set a threshold that rejects all data lower than the center of the brick to be grasped, i.e. remove all depth pixels closer than 0.1 m to the assumed ground level. After that, we initiate a flood-fill algorithm, described in (Krajník et al., 2014) and quickly obtain information about the detected segments sizes, centers, eccentricities, and distances from the camera.

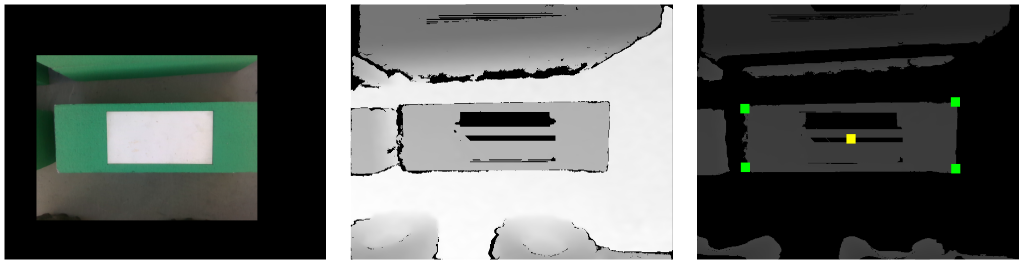

(a) RGB image (for comparison only) (b) Raw depth image (c) Segmentation and feature detection

**Figure 14.** Visual servoing by the RGBD camera mounted to the end effector. The RGB image is only shown for comparison. The brick detection is performed in the depth domain, as it is more resilient to object illumination. The gaps between the stacked bricks are small enough for the depth camera to capture neighboring bricks as well. The alignment algorithm will prioritize the rightmost visible brick. The current alignment target is highlighted in yellow. The depth image offers a considerably wider field of view than the RGB sensor. Note the robot wheels at the bottom of the depth image.

The method also provides a set of pixels lying on the segment's edge. The case where none of these pixels lie at the edge of the image indicates that a full brick is visible by the camera. As mentioned before, the depth camera should be able to capture at least one full brick from the initial position, after the mobile base is finished with the alignment maneuvers. Once a brick is fully visible, we attempt to determine its type using the positions of the segment corners.

We search the brick boundary pixels for one with the highest distance from the segment center – this is one of the segment corners, which we denote as $c_0$. After that, we search for a pixel $c_1$ with the highest distance to $c_0$. The third corner $c_2$ is found as a pixel with maximal distance from both $c_0$ and $c_1$. The final corner is found as the pixel with the maximal sum of distances from $c_0$, $c_1$ and $c_2$.

Using the image coordinates of the segment corners, the algorithm then determines the width and height of the segment. We use the ratio between width and height of the segment as the main classification property. If the segment aspect ratio corresponds to one of the brick classes, its corners are transformed from pixel coordinates into 3D, and the actual size of the brick is determined. Segments, which do not satisfy the conditions for any of the brick classes, are discarded. Using only the geometric properties of the segments allows us to skip any color information processing, which would require color calibration for the current lighting conditions.

If the brick type matches the one required by the inventory manager, the position of the brick center is transformed to the coordinate system of the gripper. In the case that the brick center is out of reach of the manipulator, the mobile base itself starts to move forward and backward, while performing small turns to adjust the grasping distance. Once the brick center is directly below the gripper center (with $\pm 8$ cm tolerance), we stop the base alignment and start aligning the arm itself.

As soon as the gripper is directly above the brick, we start descending the arm, while positioning it according to the information provided by the brick detection method. Once the brick becomes so close that it does not fit the camera FOV, the magnetic gripper is powered on and the arm descends straight down. As soon as the gripper indicates that the brick is attached to its magnet, the arm starts to move upward, lifting the brick.

Once the brick is lifted, we query the inventory manager for a slot in the cargo bay, where the brick should be stored. The brick is then stored by a series of pre-defined movements and the magnetic gripper is powered off. The inventory manager is informed about the outcome of the storage action and it either requests to pick up another brick or depart for the building area.

## 4.7. Building pattern detection

The building pattern is a flat object laid on the ground plane, which makes it impossible to detect by the LiDAR scan. Therefore, a different approach has to be used. We opted to use the RGB stream of the Intel Realsense camera, which is mounted to the wrist of the manipulator. This way, the RGB images can be taken from a variety of positions, some of them providing high detection range and some of them ensuring high accuracy.

#### *4.7.1. Pattern segmentation*

The marker indicating the building area is a high-contrast violet-yellow checkered pattern, see Figure 2c. To detect it, we use a fast segmentation method inspired by (Krajník et al., 2014), which was successfully deployed during the MBZIRC 2017 contest (Štěpán et al., 2019). The segmentation algorithm was tailored to find only one color of checkerboard squares and treat the other squares as a background.

Our method uses a 3D RGB look-up grid to classify the image pixels as background, unknown, or object. The method is efficient and versatile because it can implement any pixel-wise classification algorithm in constant time. This is achieved by pre-computing the RGB look-up grid during a calibration step. In our case, the grid was initialized using a Gaussian mixture to model the searched color in the hue-saturation-value color space. The Gaussian mixture model is specified using a GUI, which allows the user to select areas of the image belonging to the desired object and observe the segmentation results on the fly.

The method uses the RGB grid to consecutively examine if the image pixels belong to the searched object. Whenever a pixel is classified positively, a flood-fill algorithm is applied to quickly extract the segment the pixel lies in. However, running a simple flood-fill algorithm would extract each of the checkerboard squares as a separate segment. That would require complicated and potentially faulty post-processing steps to merge the detected segments into one. To avoid such issues, we modified the flood-fill method to search through a larger neighborhood of pixels identified as corners.

We exploit the fact that the flood-fill segmentation, presented in (Štěpán et al., 2019) naturally obtains the information about the number of pixels of the segment color surrounding each pixel of the segment. Thus, once the flood-fill method reports that a complete segment was found, we select all pixels with less than two neighbors (these pixels are at the corners of the segment) and search for another segment pixels within 5-pixel distance. If such a pixel is found, the flood-fill segmentation continues from its position. This technique allows to efficiently find and count nearby segments of the same color even if they do not form a contiguous area. This way, we obtain the count, size, positions, and color of half of the squares on the checkerboard pattern. By comparing these values to the a priori known pattern, we can easily filter out false-positive detections. The segmentation approach is capable of detecting the pattern at varying distances and from various viewpoints, given that the RGB lookup table is properly initialized.

#### *4.7.2. Building pattern search*

After storing the bricks in the cargo bay, the robot resumes exploration of the field. First, it generates a set of waypoints to visit, see Figure 9. Then, it solves a traditional Traveling Salesman Problem (TSP) to plan its route over these points. Upon reaching each point, the robot would extend its arm as high as possible and then sweeps horizontally with the camera in an attempt to detect the building pattern, see Figure 15a. This method works reliably up to 10 m from the robot.

Once a pattern is detected, the robot folds its arm to look 0.5–2.5 m in front of it, as shown in Figure 15b. The robot then starts to move towards the pattern. After reaching the pattern, the arm is extended to the side, and the robot uses visual servoing to align itself with the pattern. This procedure is very similar to the alignment with the brick stack. The robot performs a series of “parallel parking” maneuvers to align its heading with the major axis of the pattern. Pointing the camera to the ground, the robot then drives along the checkers, until an edge of the pattern appears in the camera view.

## 4.8. Brick deployment

The brick deployment is controlled by a state machine and is very similar to the brick pickup procedure introduced in Section 4.5. The robot initially aligns itself with one section of the checkered

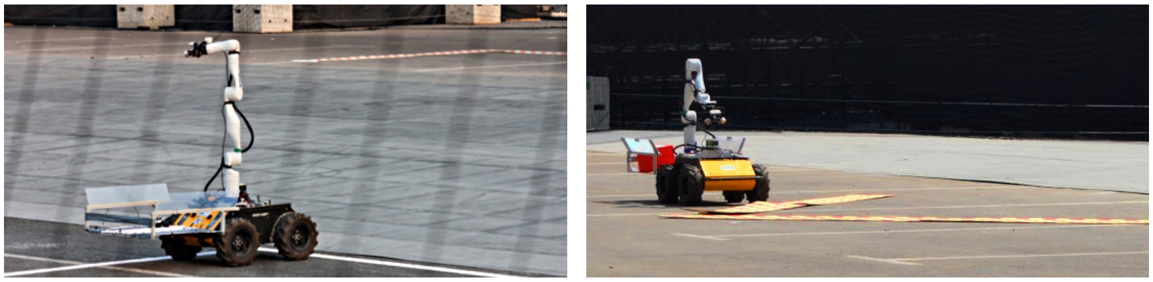

**(a)** Searching for the build pattern by fully extending the arm. **(b)** Arm is folded and camera pointed forward while driving.

**Figure 15.** During the exploration phase, the robot extends the arm and rotates the wrist joint to scan the surroundings with an RGB camera. With the arm fully extended, the system also has a higher center of gravity, rendering it unstable. Therefore, the robot has to stop moving before each scanning action. While driving, the arm is folded so that the camera points to the ground in front of the robot.

pattern and moves to the edge of the pattern. Afterward, the inventory manager is polled for the next brick to be placed down. The inventory manager provides the robot with a placement coordinate relative to the pattern edge. Moreover, it informs the arm about the coordinates of the brick to be placed in the robot's cargo bay. The robot drives to the provided coordinates while picking up the brick from its cargo bay. It then unloads the brick by following a series of pre-defined loading actions in reverse. After releasing the brick onto the pattern, the inventory manager is queried for the next location and the next brick to be placed.

A gap of 0.1 m is left in between individual bricks to prevent damage in case of minor misalignment. Even with this offset, all bricks will still be fully placed onto the pattern.

### 4.9. Speeding up exploration

The full exploration of the arena by the onboard RGB camera alone is a time-consuming activity. Since the UAVs (Baca et al., 2020) were deployed in the arena simultaneously with the UGV, the option of multi-robot cooperation was also explored. In particular, mutual communication between the robots was enabled via a 5 GHz Wi-Fi network, which was provided by the contest organizers. We have employed the *nimbro_network* (Schwarz et al., 2017; Schwarz, 2020), a system allowing message passing between multiple independent ROS systems within the same network.

One of the UAVs was initially tasked with quickly flying over the entire arena at a height of 4.5 m in a sweeping pattern, and creating a map of all objects of interest using its onboard sensors. One of the sensors is a down-facing Intel Realsense D435, which matched the RGBD camera used by the UGV. Since the drones could run the pattern detection described in the previous section on their onboard computers, they could provide the robot with an approximate position of the searched pattern. This position would then be used in the TSP method as the initial location of the exploration.

## 5. Experimental evaluation

### 5.1. Early development

Early experimental evaluation of the system took place at the tennis courts of the Abu Dhabi Campus of New York University, shown in Figure 16. Here, the system was able to repeatedly locate and load the bricks, move to the approximate position of the building pattern, align to it and place the bricks in the specified order. The accuracy of the brick picking and delivery was around $\pm3$ cm, and the success rate of loading seven bricks, i.e. the full stack needed to build one level of the wall, was around 75%. In the remaining 25% of cases, the robot failed to load some of the bricks, which

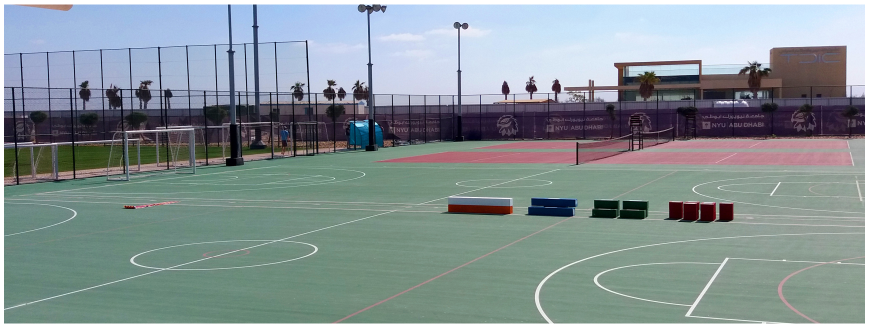


**Figure 16.** The tennis courts at the Abu Dhabi Campus of New York University were used for experimental evaluation of the system. In here, we were able to repeatedly pick up and place an entire layer of the brick wall, and perform autonomous exploration of an area comparable in size with the competition venue.

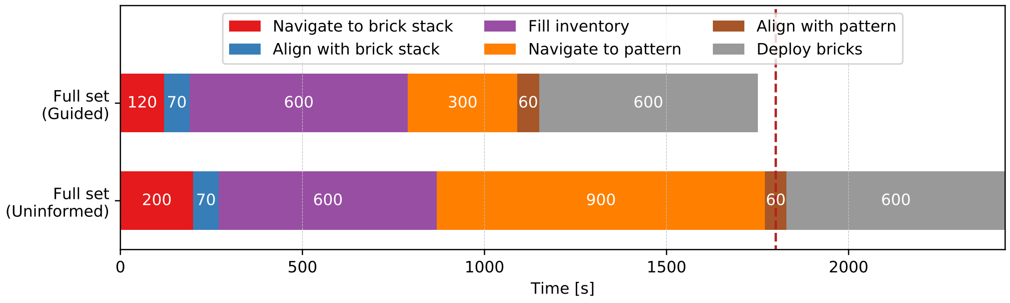


**Figure 17.** Approximate duration of the individual sub-processes as observed during the early experiments at the NYUAD campus. The 30-minute time limit of the competition is marked by a dashed red line. Using the guided autonomy approach, the system was capable of finishing the task with a full set of 7 bricks (4 red, 2 green, 1 blue) within the time limit.

lead to interrupting the loading procedure and heading for the building area, where it deployed the bricks it had in the cargo bay.

We also tested a fully uninformed scenario, in which the robot could not detect the objects of interest from its starting position and the exploration subsystem was given no initializing condition. This variant proved to be extremely time-demanding and significantly exceeded the competition time limit. As seen in Figure 17, the uninformed search for the building pattern alone required approximately 15 minutes – half of the entire time budget. This was caused by multiple contributing factors.

Firstly, the perception range for pattern detection is fairly limited. Since the building pattern is laid out flat on the ground, the robot cannot exploit its long-range LiDAR, which is otherwise used to detect the objects of interest. Instead, we have to rely on the onboard RGB camera and the image-segmentation techniques described earlier. To clearly distinguish the checkered pattern, the observation has to be made from above. With increasing distance between the robot and the pattern, the observation angle becomes very shallow, and the characteristic checkered properties are lost to aliasing. This effect can also be observed in Figure 15.

Additionally, extending the arm upwards significantly shifts the robot's center of gravity. Driving with the arm in an upright position turned out to be impossible, as the entire UGV became unstable and tended to tumble over. Therefore, the arm has to be folded while driving, and the mobile base has to stop before the arm is raised. The camera is kept stationary while the segmentation is taking place and 10 consecutive images are used in the process. In the upright position, the arm then sweeps horizontally in 60° steps, with a brief pause for the image acquisition in between. After performing a full 360-degree scan of the surrounding area, the arm is retracted and the mobile base resumes driving. When combined with the mobile base braking, arm extending, scanning the surroundings, and retracting, one such a scan takes approximately 90 seconds.

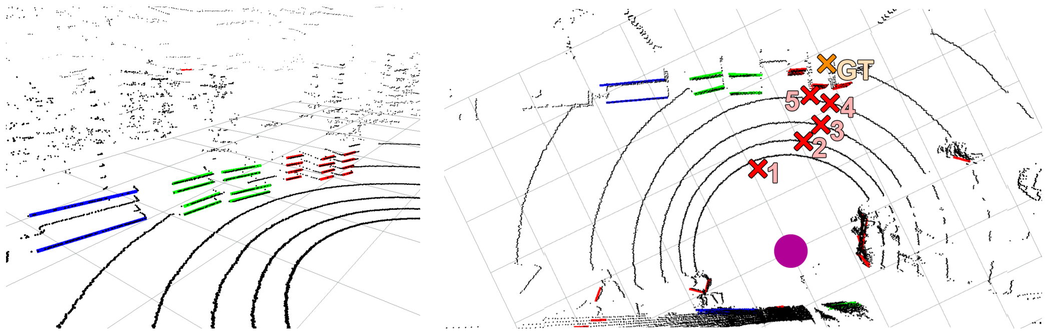
To improve exploration efficiency, we introduced the concept of "guided autonomy". In this mode, the robot performed the entire task autonomously, however, the exploration could be started from a predefined position. The initial position may be set in advance, during the preparation phase, or provided on the fly by a cooperating UAV. Using this approach, we were able to shorten the exploration phase, so that a full section of the wall (7 bricks) could be assembled within the 30-minute limit. The values shown in Figure 17 represent a conservative time estimate based on a scenario wherein the objects of interest are located in the opposite corners of the arena and the robot has to traverse a large distance.

Despite a significant speedup, the time budget was still very tight with almost no room for error. Due to the need for precise alignment of the robot base with the bricks before picking up each of them, the loading procedure itself took almost 10 minutes. The brick manipulation had to be performed rather slowly, as faster motion or sudden acceleration might cause the brick to be dropped by the magnetic gripper. The unloading procedure was similar to brick loading and also required nearly 10 minutes, due to the slow and careful brick handling, as well as the continuous alignment of the mobile base with the building pattern.

Before the competition trials, we have also prepared an emergency protocol, where the robot would only load one brick and deliver it to the building area immediately. In the remaining time, the robot would attempt to pick up and deploy additional bricks in a second run. This procedure ensured, that the UGV would not hinder the faster and more agile UAVs, as the rules of competition required both robot types to score at least once in the same trial.

## 5.2. Brick stack detection

Detection of stacked bricks was crucial during the exploration phase. We have developed detection techniques based on the segmentation of the point cloud provided by the Velodyne VLP-16 LiDAR. The processing pipeline is discussed in greater detail in Section 4.4. In Figure 18a, we show the line segments extracted from the point cloud. The segments are classified based on their length and



**(a)** Line segments extracted by the IEPF algortihm.

**(b)** Red brick pile localization by the EM algorithm. The red crosses represent the estimated position of the brick pile center in 5 consecutive iterations of the EM. The ground-truth is highlighted in orange, the UGV position is marked by a purple dot. Several false-positive detections are visible in the lower half of the image.

**Figure 18.** Estimation of the brick stack position and orientation using the Velodyne VLP-16 LiDAR. The black dots represent the raw point cloud returned by the sensor. The line segments extracted from the data are assigned a color based on the most likely brick class. The processing pipeline also utilizes a priori knowledge of the pickup zone layout to filter out the false-positive candidates.

**Table 2.** Accuracy of brick candidates generation by the IEPF algorithm. The evaluation was performed offline using data collected during one of the rehearsals.

| Brick color | Correct | Incorrect | Success rate |
|---|---|---|---|
| Red | 174 | 70 | 0.713 |
| Green | 57 | 19 | 0.75 |
| Blue | 24 | 63 | 0.275 |
| Orange | 32 | 11 | 0.615 |

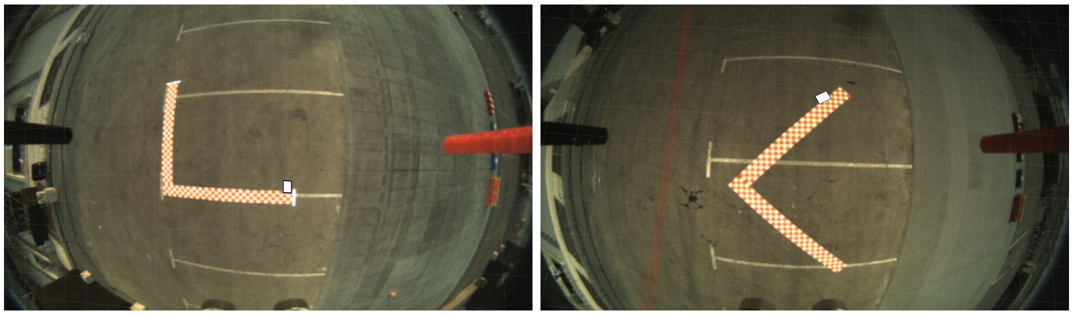


**Figure 19.** Approximate detections of the UGV building pattern, as acquired by the wide-angle camera onboard the UAV during the Brick Challenge trials. The black and white rectangle represents the estimated position of the pattern edge. Note that the detections are transformed into 3D using the known UAV position within its local frame. The 3D position is used as an overlay for the fish-eye image for visualization purposes. The overlay does not apply any transformation to account for the lens distortion, which results in a misalignment of the marker.

position in the brick stack. The most likely brick class is assigned to each segment and the segment is colored accordingly.

Figure 18b demonstrates the first few iterations of the EM algorithm. This algorithm is used to estimate the position and orientation of the brick piles in the pickup area. The extracted line segments usually include a large number of false-positive classifications, as seen in Figure 18b. However, the processing pipeline is capable of filtering out the false-positive data by leveraging the a priori knowledge of the pickup area layout. Only a coarse estimate of the pickup area position and orientation is required. Finer alignment is performed afterward, using the wrist-mounted RGBD camera and visual servoing.

The accuracy of the brick candidates extraction from the LiDAR data by the IEPF algorithm is shown in Table 2. The evaluation was performed offline by processing one of the datasets obtained during the competition rehearsals. Without any additional filtering or sensor fusion, the classifier success rate is strongly correlated with the total number of bricks deployed in the pickup area. The lowest success rate was achieved for the blue bricks, which also represent the least numerous brick class. As a result, the classifier is more susceptible to false-positive detections. The success rate can be significantly improved by fusing the candidate positions with the current position of the UGV provided by the localization pipeline. Further improvements are made by the EM algorithm, which utilizes that the shape and layout of the pickup area are known in advance.

## 5.3. UAV-assisted pattern detection

The UAVs were equipped with two down-facing cameras: the Intel Realsense D435 and the mvBlueFox-200w – an RGB camera with a fish-eye lens. Throughout the development of the aerial system, it turned out that disabling the RGB stream from the D435 improved the reliability of the depth stream, leaving the mvBlueFox as the only option for pattern detection. Figure 19 shows the detections made by one of the UAVs in the MBZIRC arena. An approximate position of the building

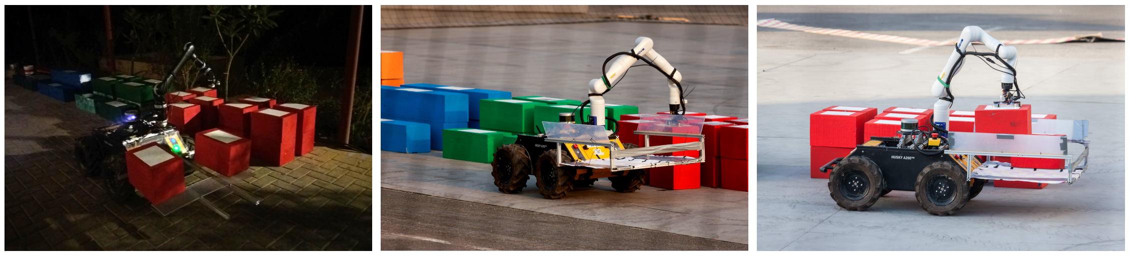

**(a)** Poor illumination at night **(b)** Strong lateral illumination at sundown **(c)** Broad daylight

**Figure 20.** The visual servoing and brick grasping procedures have been tested under a variety of lighting conditions, to ensure a robust performance throughout the entire competition day.

pattern was repeatably obtained by the UAV in each rehearsal, as well as in the competition trials. The data, however, was not actively shared with the UGV, due to coordinate-frame incompatibility.

Each UAV was using a local coordinate frame, and the mutual synchronization relied on GNSS providing a common origin. The UGV was localized using the onboard LiDAR and wheel odometry without an external positioning system. Synchronizing the two frames of reference by simply adding a GNSS receiver to the ground robot was not possible. The onboard equipment, notably the devices using an Ethernet connection (the Velodyne LiDAR and the Kinova arm), induced significant electromagnetic interference, making the receiver unable to get a fix.

Unfortunately, we did not manage to implement a reliable coordinate-frame synchronization system in time before the competition, and therefore the pattern position detected by the UAV could not be used to assist the UGV directly. Due to the fear of clogging the communication channel and disrupting the mutual UAV coordination, the UAV-UGV communication link was not even initialized in the competition trials. Instead, we estimated the position of the pattern from images taken during the previous contest rounds, and the communication was only emulated.

## 5.4. Competition trials

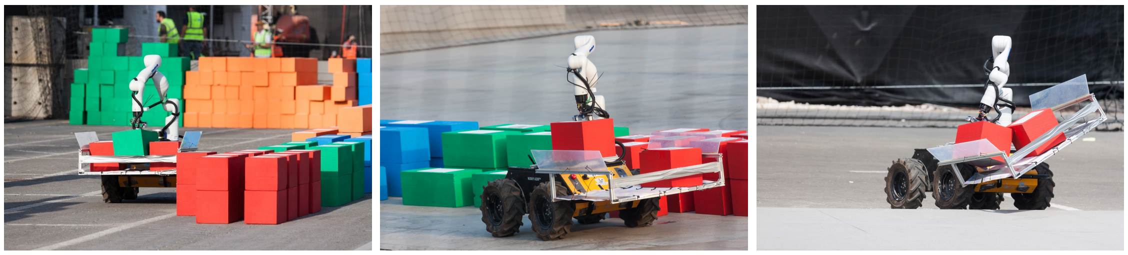
Since the competition was held throughout the whole day, we have extensively tested the visual servoing and brick grasping under various lighting conditions, as shown in Figure 20. These conditions ranged from near-complete darkness[11] to broad daylight. Having the perception invariant to external illumination allowed the robot to grasp and load the bricks at any time of day in a reliable and repeatable manner.

The final experimental evaluation took place at the MBZIRC 2020 Challenge rehearsals and competition trials. The main problem encountered was that the ground of the contest area was not flat as expected, but its central part formed a long non-flat slope. The bricks were placed at the edge of the slope, which not only complicated their detection by the LiDAR, it also meant that the robot was not positioned vertically during the brick pickup and loading. The sloped pickup area was also one of the reasons, why several teams were not able to pick up a single brick in the competition.

Since the gripper was partially compliant, the bricks were not held straight, and their angle relative to the robot was not the same as if the robot were positioned on level horizontal surface. This had a negative impact on the ability to store bricks in the cargo bay, as shown in Figure 21.

During the first competition trial, the placement of the brick stack on the edge of the slope triggered an erroneous output of our obstacle detection system, which perceived the bricks as if they were inside an obstacle. For this reason, the UGV was unable to reach the brick stack on the first day of the contest. After the setback, we processed the LiDAR data recorded during the trial

[11] Video from the experiment: http://mrs.felk.cvut.cz/fr2020ugv



**(a)** Loading on level ground **(b)** Loading on the inclined surface **(c)** Result of the misaligned loading

**Figure 21.** Loading bricks into the cargo bay using the manipulator arm. On level ground, the robot was able to fill the cargo bay as intended. However, during the competition, the bricks were positioned on an inclined ramp, which ran through the center of the arena. In such a case, the bricks were not being lowered into the cargo bay along a vertical line. As a result of the misalignment, the bricks could not have been stored with the expected level of precision.

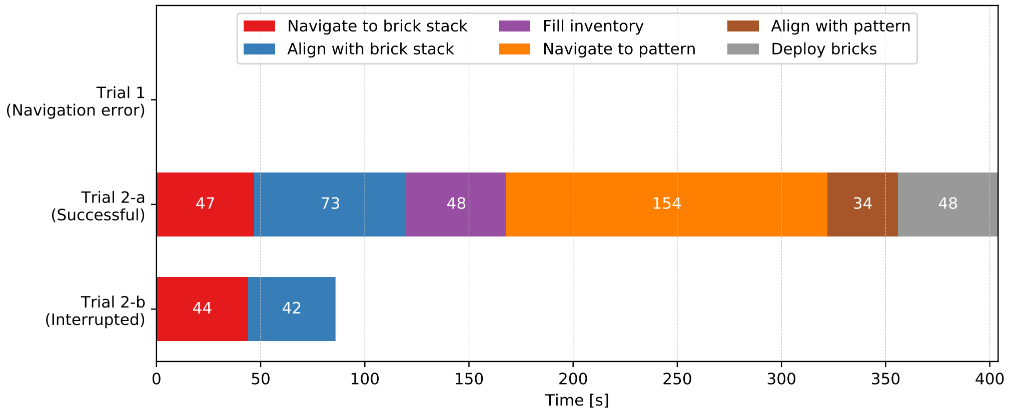
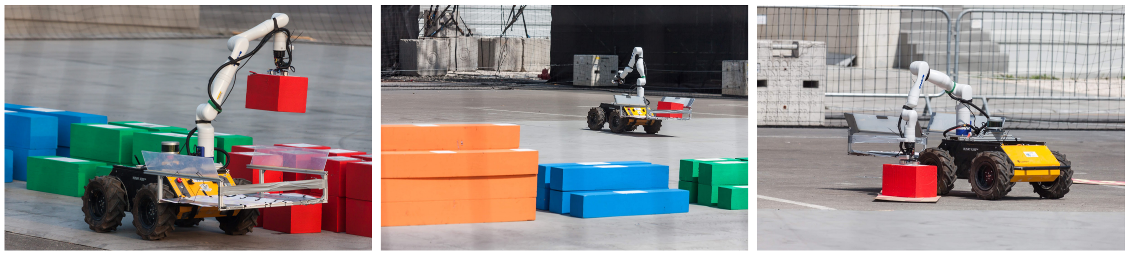


**Figure 22.** The duration of individual sub-processes in the final trials of the MBZIRC 2020 competition. On the first day, the robot never reached the brick stack due to a navigation error. On the second day, the initial inventory requirement was set to 1 red brick, which the robot successfully loaded and deployed to the building pattern. Afterward, a manual reset was requested to install fresh batteries to all the robots. In the remaining time, the robot attempted to place 3 additional bricks. However, the operation was interrupted while filling up the inventory, after one of the UAVs had performed an emergency landing directly in front of the robot.

and the rehearsals and updated the obstacle detection pipeline to work as intended regardless of the arena configuration.

For the second trial, we also enabled the emergency protocol to place at least one brick as quickly as possible. The first run of the second trial was successful, and the robot was able to autonomously pick up, deliver and place one red brick in under 7 minutes, as shown in Figure 22. After the first run, a manual reset was requested to install fresh batteries on all the robots. Following the reset, the inventory requirement was set to 1 green and 2 red bricks. However, the onboard safety system aborted the execution while filling up the inventory, after one of the UAVs had performed an emergency landing directly in front of the UGV. After the second reset, it was no longer possible to deliver additional bricks within the remaining time.

As mentioned before, our system was among the only two UGVs that managed to place at least one brick in autonomous mode during the competition trials. Figure 23 shows the system as it performed the entire task successfully during the final round of competition. In cooperation with the UAV system (Baca et al., 2020), our team (CTU-UPenn-NYU) managed to win the Brick challenge in a landslide, scoring 8.24 points. Team NimbRo (Lenz et al., 2020) from the University of Bonn



**(a)** Loading a red brick into the cargo bay **(b)** Searching for the checkered pattern **(c)** Successful placement of the red brick

**Figure 23.** Snapshots from the successful competition trial. For this trial, we opted to employ a reliable and well-tested strategy, which ensured placing a single red brick onto the building pattern.

ranked second with a score of 1.33 points and the Technical University of Denmark ranked third with a score of 0.89 points.

## 6. Conclusion

We have presented a complex, mobile manipulation system for localizing, grasping, transporting, and precisely placing magnetic objects. Our system can operate autonomously in a semi-structured environment and even handle unexpected encounters with dynamic obstacles, such as bricks dropped by other robots.

Our design relies solely on onboard sensors for localization and uses the Adaptive Monte Carlo Localization approach. We have introduced an extension of the localization approach to improve reliability in environments with relatively sparse feature-density and an uneven ground plane. The implementation also corrects for drift in wheel odometry caused by dynamic shifts in the robot's center-of-mass as bricks are added into the cargo bay and the manipulator arm moves.

We have conducted a thorough experimental evaluation in a broad variety of conditions. The brick detection and loading subsystems have been proven to work even at night, without the need for an external light source. As a result, such a mobile manipulator may also be deployed in a poorly illuminated indoor environment, or perform the outdoor-indoor transition without the need for modification. Furthermore, we developed a series of efficient computer-vision techniques for quickly and reliably detecting both component bricks and the target construction pattern. We were able to run the necessary image-segmentation and feature-extraction methods on the onboard computer in real-time. This performance allowed us to use visual servoing to precisely align both the mobile base and the manipulator with the target object. We also used tightly-coupled motion control for the mobile base and the manipulator arm. In combination with the aforementioned computer-vision techniques, our system was able to achieve centimeter-level precision in grasping and placing objects.

The entire system is built using commercially available off-the-shelf components except for the magnetic gripper. After the competition, the control software was released as open-source (Broughton et al., 2020).

### 6.1. Lessons learned

Throughout the MBZIRC 2020 competition, we have seen many other teams successfully place at least one brick in a controlled environment. However, nearly all teams struggled to perform the same task in the competition arena for various reasons. The most notable obstacle was the inclined ramp, upon which the brick pickup area was placed. This obstacle also thwarted our primary intention of loading multiple bricks at the same time.

Despite the setbacks, we have greatly benefited from the complex inventory-management system that we had developed primarily for debugging purposes. The inventory manager allowed us to practice different scenarios and start the task with an arbitrary cargo bay load. Most importantly, however, it allowed us to change the strategy on the fly. After encountering the loading issues on the first competition day, we were able to turn on the emergency protocol by simply changing the number of bricks required by the inventory manager to one. After successfully fulfilling the minimum requirements, the manager would automatically adjust the requirements for the next run. This scenario, among others, was extensively tested during our experimental sessions conducted at the Campus of the New York University in Abu Dhabi, see Figure 16. For this reason, the well-tested backup solution was activated for the final round of competition.

The sequential nature of the task essentially requires all subsystems to perform flawlessly to successfully place a brick onto the building pattern. The result of each action directly feeds into the next action and essentially creates a serial execution pipeline, where a failure of one node results in a failure of the overall system. We aimed to develop a system capable of operating without human supervision for extended periods of time. For this reason, special effort was put into failure detection and recovery mechanisms, which were embedded into all levels of the control hierarchy, and ensured that the robot carried on with the task even after one of the actions had failed.

Redundant perception sources also played a crucial role in our success, as they allowed the system to detect faulty readings and trigger automated recovery actions. It is worth noting, that nearly all aspects of perception within the system utilize two or more independent sensor inputs and that the fallback process after an unsuccessful attempt usually switches to a different perception or motion system.

The brick pickup phase was by far the most complex part of the challenge, and some sort of recovery behavior was triggered almost routinely during this phase. However, the alternating use of both motion systems (mobile base, manipulator), paired with the powerful computer-vision methods, allowed the robot to achieve centimeter-level grasping accuracy in a reliable and repeatable manner.

The built-in failure detection and recovery system significantly improved the robustness of the overall system and allowed the UGV to better cope with the actual conditions inside the competition arena. The redundant design is also in accordance with the latest demands from the industry, where safety and reliability are the most desirable attributes of a mobile manipulator, especially if the workspace is shared with other mobile robots or humans.

### 6.2. Future work

There were also certain aspects of the system that would benefit considerably from further development. During the contest, brick manipulation was by far the slowest component of the system. We did not employ any advanced control or motion planning for the gripper trajectory. Instead, the manipulator followed a sequence of pre-defined waypoints in the joint space. To prevent the loss of a grasped object, the maximum allowed acceleration of the arm had to be very limited. Using a desired torque or acceleration as a reference would allow us to make faster maneuvers, even with a grasped brick, and reduce the overall manipulation time. We would also like to finish the coordinate-frame synchronization work to enable proper real-time cooperation in future heterogeneous robotic systems.

## Acknowledgments

The authors acknowledge the hard work of all members of the CTU-UPenn-NYU team (captured in Figure 24), who cooperated and contributed towards the MBZIRC 2020 competition. We also thank our colleagues from Abu Dhabi campus of NYU, who arranged the testing grounds for our robot near the competition venue. The phenomenal results would not have been possible without them. The presented work was sponsored by the Czech Ministry of Education by OP VVV funded project CZ.02.1.01/0.0/0.0/16 019/0000765 “Research Center for Informatics”, the Czech Science Foundation (GAČR) under research project No. 20-27034J, the Czech Technical University under

**Figure 24.** Members of the CTU-UPenn-NYU team being awarded for the 1st place in the Brick challenge of the MBZIRC 2020.

Grant No. SGS20/174/OHK3/3T/13, the EU H2020 project AERIAL CORE, No. 871479 and by the Khalifa University under the MBZIRC 2020 sponsorship for 15 selected teams.

## ORCID

Petr Štibinger[1,*] https://orcid.org/0000-0002-7662-9230
George Broughton[2,†] https://orcid.org/0000-0003-0071-5834
Filip Majer[2,†] https://orcid.org/0000-0002-4921-3360
Dinesh Thakur[4,§] https://orcid.org/0000-0001-5046-8160
Giuseppe Loianno[5,‡] https://orcid.org/0000-0002-3263-5401
Tomáš Krajník[2,†] https://orcid.org/0000-0002-4408-7916
Martin Saska[1,*] https://orcid.org/0000-0001-7106-3816

## References

Amjadi, A. S., Raoufi, M., Turgut, A. E., Broughton, G., Krajník, T., and Arvin, F. (2019). Cooperative pollution source localization and cleanup with a bio-inspired swarm robot aggregation. *arXiv preprint arXiv:1907.09585*.

Bac, C. W., van Henten, E. J., Hemming, J., and Edan, Y. (2014). Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911.

Baca, T., Penicka, R., Stepan, P., Petrlik, M., Spurny, V., Hert, D., and Saska, M. (2020). Autonomous cooperative wall building by a team of unmanned aerial vehicles in the mbzirc 2020 competition. *arXiv: 2012.05946*. Submitted to: Robotics and Autonomous Systems.

Baca, T., Stepan, P., Spurny, V., Hert, D., Penicka, R., Saska, M., Thomas, J., Loianno, G., and Kumar, V. (2019). Autonomous landing on a moving vehicle with an unmanned aerial vehicle. *Journal of Field Robotics*, 36(5):874–891.

Bausys, R., Cavallaro, F., and Semenas, R. (2019). Application of sustainability principles for harsh environment exploration by autonomous robot. *Sustainability*, 11(9):2518.

Binch, A., Das, G. P., Fentanes, J. P., and Hanheide, M. (2020). Context dependant iterative parameter optimisation for robust robot navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3937–3943. IEEE.

Bischoff, R., Huggenberger, U., and Prassler, E. (2011). Kuka youbot-a mobile manipulator for research and education. In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE.

Bogue, R. (2016). Growth in e-commerce boosts innovation in the warehouse robot market. *Industrial Robot: An International Journal*, 43:583–587.

Bogue, R. (2020). Fruit picking robots: has their time come? *Industrial Robot: the international journal of robotics research and application*, 47:141–145.

Broughton, G., Štibinger, P., Krajník, T., Majer, F., and Dinesh, T. (2020). Ugv code repository of the mbzirc challenge ii. citied on 2020-09-11.

Chen, F., Selvaggio, M., and Caldwell, D. G. (2018). Dexterous grasping by manipulability selection for mobile manipulator with visual guidance. *IEEE Transactions on Industrial Informatics*, 15(2):1202–1210.

Chitta, S., Sucan, I., and Cousins, S. (2012). Moveit! [ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19.

Clearpath, R. (2015). Robots/husky-ros wiki. cited on 2020-08-27.

Coppola, G., Zhang, D., and Liu, K. (2014). A 6-dof reconfigurable hybrid parallel manipulator. *Robotics and Computer-Integrated Manufacturing*, 30(2):99–106.

Davis, J., Edgar, T., Porter, J., Bernaden, J., and Sarli, M. (2012). Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers & Chemical Engineering*, 47:145–156.

Dellin, C. M., Strabala, K., Haynes, G. C., Stager, D., and Srinivasa, S. S. (2016). Guided manipulation planning at the darpa robotics challenge trials. In *Experimental Robotics*, pages 149–163. Springer.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Dömel, A., Kriegel, S., Kaßecker, M., Brucker, M., Bodenmüller, T., and Suppa, M. (2017). Toward fully autonomous mobile manipulation for industrial environments. *International Journal of Advanced Robotic Systems*, 14(4):1–19.

Duckett, T., Pearson, S., Blackmore, S., Grieve, B., Chen, W.-H., Cielniak, G., Cleaversmith, J., Dai, J., Davis, S., Fox, C., et al. (2018). Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762*.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Francis, R., Estlin, T., Doran, G., Johnstone, S., Gaines, D., Verma, V., Burl, M., Frydenvang, J., Montaño, S., Wiens, R., et al. (2017). Aegis autonomous targeting for chemcam on mars science laboratory: Deployment and results of initial science team use. *Science Robotics*, 2(7):1–12.

Gray, S., Chevalier, R., Kotfis, D., Caimano, B., Chaney, K., Rubin, A., Fregene, K., and Danko, T. (2018). An architecture for human-guided autonomy: Team trooper at the darpa robotics challenge finals. In *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, pages 549–582. Springer.

Hawes, N. et al. (2017). The strands project: Long-term autonomy in everyday environments. *IEEE Robotics Automation Magazine*, 24(3):146–156.

Hebert, P., Ma, J., Borders, J., Aydemir, A., Bajracharya, M., Hudson, N., Shankar, K., Karumanchi, S., Douillard, B., and Burdick, J. (2015). Supervised remote robot with guided autonomy and teleoperation (surrogate): a framework for whole-body manipulation. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 5509–5516. IEEE.

Krajník, T., Majer, F., Halodová, L., and Vintr, T. (2018). Navigation without localisation: reliable teach and repeat based on the convergence theorem. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1657–1664. IEEE.

Krajník, T., Nitsche, M., Faigl, J., Vaněk, P., Saska, M., Přeučil, L., Duckett, T., and Mejail, M. (2014). A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562.

Kunze, L., Hawes, N., Duckett, T., Hanheide, M., and Krajník, T. (2018). Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters*, 3(4):4023–4030.

Kusumam, K., Krajník, T., Pearson, S., Duckett, T., and Cielniak, G. (2017). 3d-vision based detection, localization, and sizing of broccoli heads in the field. *Journal of Field Robotics*, 34(8):1505–1518.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, 6(4):239–242.

Lenz, C., Schwarz, M., Rochow, A., Razlaw, J., Periyasamy, A. S., Schreiber, M., and Behnke, S. (2020). Autonomous wall building with a ugv-uav team at mbzirc 2020. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 189–196. IEEE.

Loianno, G., Spurny, V., Thomas, J., Baca, T., Thakur, D., Hert, D., Penicka, R., Krajnik, T., Zhou, A., Cho, A., et al. (2018). Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert-like environments. *IEEE Robotics and Automation Letters*, 3(3):1576–1583.

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of industrial information integration*, 6:1–10.

Majer, F., Yan, Z., Broughton, G., Ruichek, Y., and Krajník, T. (2019). Learning to see through haze: radar-based human detection for adverse weather conditions. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE.

Moon, S.-M., Huh, J., Lee, S., Kang, S., Han, C.-S., and Hong, D. (2013). A survey on robot systems for high-rise building wall maintenance. *Journal of the Korean Society for precision Engineering*, 30(4):359–367.

Nawaz, S., Hussain, M., Watson, S., Trigoni, N., and Green, P. N. (2009). An underwater robotic network for monitoring nuclear waste storage pools. In *International Conference on Sensor Systems and Software*, pages 236–255. Springer.

Notheis, S., Kern, P., Mende, M., Hein, B., Wörn, H., and Gentes, S. (2012). Towards an autonomous manipulator system for decontamination and release measurement. In *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, pages 263–268. IEEE.

Ohashi, F., Kaminishi, K., Heredia, J. D. F., Kato, H., Ogata, T., Hara, T., and Ota, J. (2016). Realization of heavy object transportation by mobile robots using handcarts and outrigger. *Robomech journal*, 3(1):27.

Pavlichenko, D., García, G. M., Koo, S., and Behnke, S. (2018). Kittingbot: A mobile manipulation robot for collaborative kitting in automotive logistics. In *International conference on intelligent autonomous systems*, pages 849–864. Springer.

Petrasch, R. and Hentschke, R. (2016). Process modeling for industry 4.0 applications: Towards an industry 4.0 process modeling language and method. In *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–5. IEEE.

Petrlík, M., Báča, T., Heřt, D., Vrba, M., Krajník, T., and Saska, M. (2020). A robust uav system for operations in a constrained environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176.

Ponnambalam, V. R., Fentanes, J. P., Das, G., Cielniak, G., Gjevestad, J. G. O., and From, P. J. (2020). Agri-cost-maps-integration of environmental constraints into navigation systems for agricultural robots. In *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 214–220. IEEE.

Pretto, A., Aravecchia, S., Burgard, W., Chebrolu, N., Dornhege, C., Falck, T., Fleckenstein, F. V., Fontenla, A., Imperoli, M., Khanna, R., et al. (2020). Building an aerial-ground robotics system for precision farming: An adaptable solution. *IEEE Robotics & Automation Magazine.* Manuscript accepted for publication, doi:10.1109/MRA.2020.3012492.

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*, volume 3, page 5. Kobe, Japan.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256.

Roblek, V., Meško, M., and Krapež, A. (2016). A complex view of industry 4.0. *Sage Open*, 6(2):1–11.

Rouček, T., Pecka, M., Čížek, P., Petříček, T., Bayer, J., Šalanský, V., Heřt, D., Petrlík, M., Báča, T., Spurný, V., et al. (2019). Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 274–290. Springer.

Schwarz, M. (2020). nimbro_network - ros transport for high-latency, low-quality networks. cited on 2020-09-11.

Schwarz, M., Droeschel, D., Lenz, C., Periyasamy, A. S., Puang, E. Y., Razlaw, J., Rodriguez, D., Schüller, S., Schreiber, M., and Behnke, S. (2019). Team nimbro at mbzirc 2017: Autonomous valve stem turning using a wrench. *Journal of Field Robotics*, 36(1):170–182.

Schwarz, M., Rodehutskors, T., and Droeschel, D. (2017). Nimbro rescue: Solving disaster-response tasks through mobile manipulation robot momaro [j]. *Journal of Field Robotics*, 34(2):400–425.

Spurný, V., Báča, T., Saska, M., Pěnička, R., Krajník, T., Thomas, J., Thakur, D., Loianno, G., and Kumar, V. (2019). Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles. *Journal of Field Robotics*, 36(1):125–148.

Starek, J. A., Açıkmeşe, B., Nesnas, I. A., and Pavone, M. (2016). Spacecraft autonomy challenges for next-generation space missions. In *Advances in Control System Technology for Aerospace Applications*, pages 1–48. Springer.

Štěpán, P., Krajník, T., Petrlík, M., and Saska, M. (2019). Vision techniques for on-board detection, following, and mapping of moving targets. *Journal of Field Robotics*, 36(1):252–269.

Thoben, K.-D., Wiesner, S., and Wuest, T. (2017). “industrie 4.0” and smart manufacturing-a review of research issues and application examples. *International journal of automation technology*, 11(1):4–16.

Triebel, R., Arras, K., Alami, R., Beyer, L., Breuers, S., Chatila, R., Chetouani, M., Cremers, D., Evers, V., Fiore, M., et al. (2016). Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and service robotics*, pages 607–622. Springer.

Wise, M., Ferguson, M., King, D., Diehr, E., and Dymesich, D. (2016). Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*.

Xu, L. D., Xu, E. L., and Li, L. (2018). Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962.

**How to cite this article:** Štibinger, P., Broughton, G., Majer, F., Rozsypálek, Z., Wang, A., Jindal, K., Zhou, A., Thakur, D., Loianno, G., Krajník, T., & Saska, M. (2022). Towards new frontiers in mobile manipulation: Team CTU-UPenn-NYU at MBZIRC 2020. *Field Robotics, 2*, 75–106.