



**Faculty of Electrical Engineering**  
**Department of control engineering**

**Master's thesis**

# **Autonomous Exploration of Unknown Rough Terrain with Hexapod Walking Robot**

**Jan Bayer**

**May 2019**

**Supervisor: Doc. Ing. Jan Faigl, Ph.D.**

## I. Personal and study details

Student's name: **Bayer Jan** Personal ID number: **434910**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Control Engineering**  
Study program: **Cybernetics and Robotics**  
Branch of study: **Cybernetics and Robotics**

## II. Master's thesis details

Master's thesis title in English:

**Autonomous Exploration of Unknown Rough Terrain with Hexapod Walking Robot**

Master's thesis title in Czech:

**Autonomní explorační nerovného terénu šestinožným kráčečím robotem**

Guidelines:

1. Implement and experimentally verify precision and reliability of existing vision-based localization, e.g., [1], using hexapod walking robot of the Computational Robotics Laboratory [2].
2. Propose improvement of the localization using selected sensor fusion approach, e.g., consider inertial measurements and Kalman filter.
3. Implement a method for rough terrain mapping [3,4] and proposed exploration strategy with terrain-aware motion planning [5,6,7].
4. Verify the proposed solution in an experimental scenario with the real walking robot.

Bibliography / sources:

- [1] Raúl Mur-Artal and Juan D. Tardós, ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras, IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, 2017.
- [2] J. Mrva, J. Faigl: Tactile sensing with servo drives feedback only for blind hexapod walking robot, Robot Motion and Control (RoMoCo), 2015, pp. 240-245.
- [3] P. Fankhauser, M. Bloesch and M. Hutter: Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization, IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3019-3026, 2018.
- [4] P. Fankhauser, M. Bloesch and R. Siegwart: Collaborative navigation for flying and walking robots, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 2859-2866.
- [5] D. Belter, P. Labecki and P. Skrzypczyński: An exploration-based approach to terrain traversability assessment for a walking robot, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013, pp. 1-6.
- [6] S. Bartoszyk, P. Kasprzak and D. Belter: Terrain-aware motion planning for a walking robot, Robot Motion and Control (RoMoCo), 2017, pp. 29-34.
- [7] D. Belter, P. Labecki and P. Skrzypczyński: Adaptive Motion Planning for Autonomous Rough Terrain Traversal with a Walking Robot, Journal of Field Robotics, vol. 33, pp. 337-370, 2016.

Name and workplace of master's thesis supervisor:

**doc. Ing. Jan Faigl, Ph.D., Artificial Intelligence Center, FEE**

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **24.01.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

doc. Ing. Jan Faigl, Ph.D.  
Supervisor's signature

prof. Ing. Michael Šebek, DrSc.  
Head of department's signature

prof. Ing. Pavel Ripka, CSc.  
Dean's signature

### III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

\_\_\_\_\_  
Date of assignment receipt

\_\_\_\_\_  
Student's signature



## **Declaration**

I declare that the presented work was developed independently and that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 24, 2019

.....  
Jan Bayer



## **Acknowledgement**

I want to express thanks to my thesis supervisor, doc. Ing. Jan Faigl. He taught me a lot and provided me support for the development of such a challenging project.

## Abstrakt

Tato práce se zabývá úlohou autonomního robotického průzkumu neznámého prostředí šesti-  
nohým kráčečím robotem. Cílem průzkumu je nejen vytvořit prostorovou mapu prostředí, ale  
také určit průchodnost prozkoumaných částí prostředí. Robotický průzkum je založen na navi-  
gaci robotu k hranici mezi známým a neznámým prostředím. Robot na základě aktuálního mod-  
elu prostředí autonomně stanovuje další navigační cíl tak, aby efektivně vytvořil výškovou mapu  
prostředí.

Navržený systém pro autonomní robotický průzkum je zaměřen na malé kráčečící roboty,  
jež mohou nést pouze malé a lehké senzory. Maximální nosnost robotu omezuje možnost volby  
vhodného lokalizačního systému, který je nutný pro autonomní navigaci a stavbu výškové mapy  
prostředí. Z existujících lokalizačních systémů byla vybrána metoda ORB-SLAM2 jako výchozí  
vizuální lokalizace, která byla rozšířena o využití sensorické fúze s externím lokalizačním systé-  
mem a inerciální jednotkou. Kromě tohoto řešení je v práci využita kamera pro lokalizaci, jejíž  
součástí je jednotka kombinující výpočet vizuální lokalizace a sensorické fúze, která tak přímo  
poskytuje odhad polohy a natočení kamery (robotu).

V rámci experimentální části práce byly porovnány obě metody vizuální lokalizace a spolehlí-  
věji fungující metoda byla použita v experimentálním nasazení navrženého a implementovaného  
systému autonomního průzkumu v reálném prostředí. Vyvinutý systém byl nasazen na šestinohém  
kráčečím robotu uvnitř budovy i ve venkovním prostředí, kde systém úspěšně realizoval plně  
autonomní průzkum. Navržené řešení bylo dále nasazeno na pásovém robotu v několika dalších  
experimentálních scénářích zahrnujících vnitřní prostory a důlní chodby, které byly úspěšně  
mapovány v rámci DARPA Subterranean Challenge.

**Klíčová slova:** Explorace, SLAM, Šestinohý kráčečící robot, Elevační mapa, Reprezentace map  
stromem, ORB-SLAM2, Sensorická fúze

## Abstract

The autonomous robotic exploration of an unknown environment with a hexapod walking robot is addressed in this thesis. Since the multi-legged walking robot is capable of traversing rough terrains, the proposed exploration system creates a map of the unknown environment while simultaneously performing the traversability assessment of the explored environment to reach navigational waypoints. The concept of frontiers has been utilized to detect navigational waypoints at the border of unknown space and already explored space represented by an elevation map.

The proposed system is targeted to run onboard of a small robot. Therefore, the localization system required to build the elevation map incrementally uses vision-based localization deployed on a small and lightweight camera. The state-of-the-art visual localization ORB-SLAM2 has been chosen as the default localization system, for which we proposed improvements based on sensory fusion with the Global Navigation Satellite System and Inertial Measurement Unit (IMU). Although ORB-SLAM2 provides sufficiently precise localization, we have also studied an alternative embedded vision-based localization system which has an inbuilt processor capable of sensory fusion. The precision and reliability of both localization systems have been experimentally evaluated, and more suitable localization system has been used during the experimental deployments of the proposed exploration system.

Based on the presented results, we can conclude that both studied localization systems provide localization of the robot with sufficient precision for deployment in an autonomous exploration scenario, but the novel embedded localization system is more reliable than ORB-SLAM2. The proposed elevation mapping technique enhanced by efficient quadtree based data representation exhibited low computational requirements that supports onboard deployment, and it is capable of covering large areas which have been demonstrated during multiple deployments in the real-world scenario. The developed exploration system enabled the hexapod walking robot to explore indoor and outdoor environments fully autonomously. The selected parts of the developed framework have also been successfully deployed on a tracked robot platform in various scenarios, including mine environment within DARPA Subterranean Challenge.

**Keywords:** Exploration, SLAM, Hexapod walking robot, Elevation map, Quadtree, ORB-SLAM2, Sensory fusion

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related work</b>	<b>3</b>
2.1	Mapping . . . . .	3
2.2	Robot localization . . . . .	4
2.2.1	Vision-based localization . . . . .	5
2.2.2	Visual Simultaneous Localization and Mapping . . . . .	6
2.2.3	Sensory fusion for vision-based localization . . . . .	7
<b>3</b>	<b>Problem Statement</b>	<b>9</b>
3.1	Metrics of the localization precision . . . . .	9
<b>4</b>	<b>Proposed Method</b>	<b>11</b>
4.1	Exploration . . . . .	11
4.2	Vision based Localization . . . . .	12
4.2.1	ORB-SLAM2 . . . . .	12
4.2.2	Sensory fusion for the localization . . . . .	14
4.2.3	Tracking camera Intel RealSense T265 . . . . .	17
4.3	Mapping and Planning . . . . .	18
4.3.1	Map representation . . . . .	18
4.3.2	Detection of untraversable areas . . . . .	20
4.3.3	Cost map . . . . .	21
4.3.4	Frontier detection . . . . .	23
4.3.5	Planning over the elevation map . . . . .	24
4.3.6	Improvements to the exploration module . . . . .	25
4.4	Path following module . . . . .	27
<b>5</b>	<b>Results</b>	<b>29</b>
5.1	Simulations . . . . .	29
5.2	Evaluation of computational requirements . . . . .	31
5.3	Description of the hexapod walking robot . . . . .	31
5.3.1	Sensors . . . . .	32
5.4	Experimental evaluation of the localization systems . . . . .	36
5.5	Experimental results with hexapod walking robot . . . . .	38
5.5.1	Indoor exploration . . . . .	39
5.5.2	Outdoor exploration . . . . .	40
5.6	Experimental Deployment on Tracked Robot . . . . .	42
5.6.1	Description of the tracked robot . . . . .	42
5.6.2	Mapping mines during DARPA Subterranean Challenge . . . . .	43
5.6.3	Indoor exploration scenarios . . . . .	44
<b>6</b>	<b>Conclusion</b>	<b>45</b>
	<b>References</b>	<b>46</b>



## List of Figures

1	Robotic platforms used within this thesis. . . . .	2
2	Outdoor locations autonomously explored by the hexapod walking robot. . . . .	2
3	The elevation of an explored indoor corridor by the tracked robot. . . . .	2
4	An example of the elevation map . . . . .	4
5	Overview of a simple visual odometry. . . . .	5
6	Example of processed pixels by different SLAM methods. . . . .	7
7	Architecture of the proposed exploration framework. . . . .	11
8	ORB-SLAM2 architecture. . . . .	13
9	An example covisibility graph produced by ORB-SLAM2 . . . . .	14
10	Tracking camera Intel RealSense T265. . . . .	17
11	Elevation map representations . . . . .	18
12	Example map for comparison of a map representations . . . . .	19
13	QPTRmatrix for caching leaves of the quadtree . . . . .	20
14	Kernels based on distance metric $D_8$ . . . . .	21
15	Distance cost and grown untraversable cells . . . . .	21
16	Frontier cells detection . . . . .	23
17	The effect of distance cost algorithm on planning. . . . .	24
18	Paths smoothed by sliding average with different window sizes. . . . .	25
19	Path planning example . . . . .	25
20	Connection of two instances of the elevation map module. . . . .	26
21	Interactive interface . . . . .	27
22	The architecture of the path following module. . . . .	27
23	The sensory data provided by the simulator . . . . .	29
24	Map explored during the simulation. . . . .	30
25	Hexapod walking robot Phantom X Mark II . . . . .	32
26	Sensors considered for vision-based localization. . . . .	33
27	Hexapod walking robot with developed sensor rig . . . . .	34
28	View from the sensor rig mounted to the hexapod walking robot. . . . .	35
29	Schema of the evaluation process. . . . .	36
30	Experimental setup for the evaluation of the localization systems. . . . .	37
31	Indoor corridor . . . . .	39
32	Exploration of the chimney surroundings. . . . .	40
33	View from the Intel RealSense D435 during the experiment. . . . .	41
34	Exploration of the yard . . . . .	41
35	Tracked robot platform. . . . .	42
36	The entrance to the experimental mine . . . . .	43
37	The elevation map of a mine . . . . .	43
38	The basement exploration scenario . . . . .	44
39	The indoor corridor exploration scenario . . . . .	44



## List of Tables

1	Filter complexities . . . . .	7
2	Technical parameters of the Intel RealSense T265 . . . . .	17
3	Example of memory requirements of map representations . . . . .	19
4	Computational requirements of exploration processes . . . . .	31
5	Parameters of considered RGB-D sensors . . . . .	34
6	Precision of the Evaluated Localization Systems . . . . .	37
7	Performance indicators . . . . .	38

# Chapter 1

## Introduction

Robotic exploration is a problem to obtain a model of an unknown area by a mobile robot that autonomously decides where to go next. Many exploration approaches to create a spatial model of the environment, such as [1], [2], address the problem by extending the idea of the frontier-based exploration introduced in [3]. The frontiers are reachable parts of the environment at the border of the unknown and already explored parts of the environment, and they are used to generate possible waypoints towards which robots are navigated to acquire new information about not yet explored parts of the environment. The explored environment can be represented by a spatial map but also by more complex models that enable to drive the exploration by, e.g., entropy [4] or model variance if the purpose of the exploration is to measure additional physical quantity. This thesis focuses on spatial autonomous exploration, and the particular environments where the developed system has been deployed are shown in Figure 2 and Figure 3.

In addition to spatial mapping, traversability assessment [5] and planning of safe paths to the navigation waypoints [6] are essential tasks of autonomous exploration, especially for robots operating in unstructured and rough terrains environments. Such environments can be found outdoors, in mines, at search and rescue or extraterrestrial missions. Not just a terrain type or exploration approach, but also the particular robot platform might influence the exploration performance. Wheeled robots are a common choice for exploration of flat indoor terrains. However, uneven terrains require robots with high traversability capabilities that can be achieved by multi-legged robots or combined wheeled and legged robot as in [6]. Regarding the particular robot type, a multi-legged robotic platform is considered in this thesis.

Although multi-legged robots have relatively good terrain transmittance, small multi-legged robots, such as Phantom X Mark II (with the size of the robot body  $10 \times 18$  cm and paid load capacity approximately 1 kg including battery), have limited operational capacity, and therefore, the available sensors and more specifically computational resources are also limited. Fortunately, novel cameras provide several options for vision-based localization and spatial mapping that are the essential tasks for exploration. One of the options the needed localization is the state-of-the-art vision-based method ORB-SLAM2 [7], which can be further enhanced by the localization fusion [8]. In addition to such traditional vision-based localization method, a new embedded localization system the Intel RealSense T265 [9] has been recently introduced to the market. The Intel RealSense T265 uses the visual localization improved by the sensory fusion using measurements of the Inertial Measurement Unit (IMU) that are coupled onboard and the system directly provides an estimation of the robot pose. However, as far as we know, there is no experimental evaluation of the precision and reliability of this particular embedded localization system on multi-legged walking robots. Thus, the properties of both localization systems are provided in the thesis, together with the report on their comparison and experimental evaluation. Based on the evaluation results, the best performing system is deployed within the proposed and developed exploration framework that has been experimentally verified in several practical scenarios with completely autonomous decision-making and robot navigation.

The rest of the thesis is organized as follows. A brief overview of the related work is presented in Section 2 while the addressed problem is specified in Section 3. The developed system is described in Section 4 and achieved exploration and mapping results in the selected scenarios are reported in Section 5, together with the experimental evaluation of two localization systems. The concluding remarks and ideas for future work are summarized in Section 6.

## 1. Introduction



(a) Hexapod walking robot with developed sensor rig for autonomous exploration

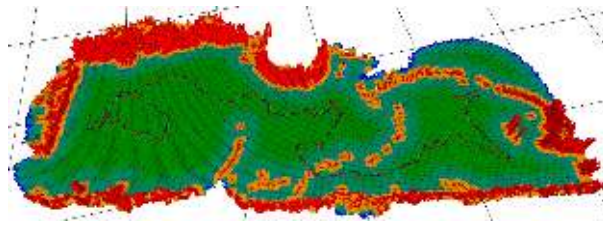


(b) Tracked robot used within DARPA Subterranean Challenge

Figure 1: Robotic platforms used within this thesis.



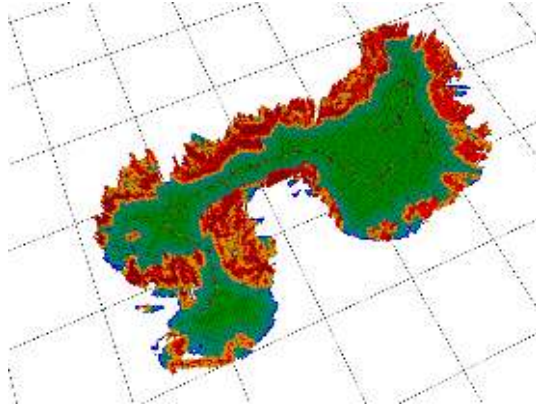
(a) Chimney location



(b) The Resulting elevation map of chimney location



(c) Yard location



(d) Resulting elevation map of chimney location

Figure 2: Outdoor locations autonomously explored by the hexapod walking robot.

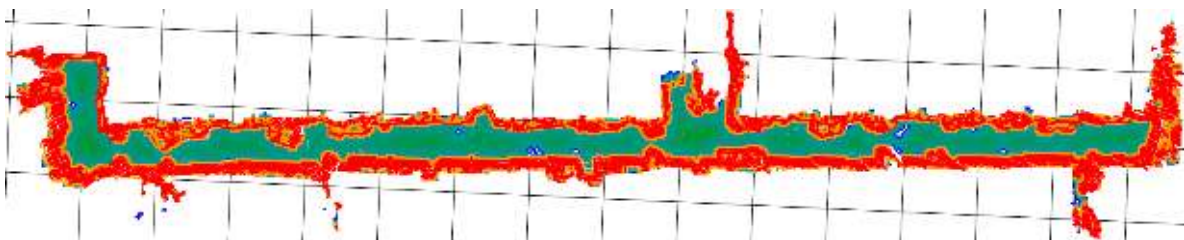


Figure 3: The elevation of an explored indoor corridor by the tracked robot.

## Chapter 2

# Related work

The fundamental approach for the autonomous building of a spatial map of the unknown area is frontier-based exploration [3], where frontiers are regions on the boundary of the already explored and unexplored parts of the environment. The frontiers are used as possible waypoints towards which the robot can be navigated to explore the environment. Numerous approaches extending the pioneering work [3] can be found in the literature, e.g., [1], [2], where the authors extend the method to multiple robots by a utility function.

The spatial exploration strongly relies on mapping and localization of the robot. Hence, it might be beneficial to explicitly consider the uncertainty of the robot pose because it can improve the accuracy of the build map as it is reported in [10], where the authors propose exploration strategy that combines the maximization of the map information and minimization of the uncertainty in the robot pose estimation. A similar idea has been exploited in [4], where entropy is employed to combine information about the robot pose and environment to select the most suitable navigational waypoint.

Although there are various exploration approaches, we base our exploration system directly on the idea of the frontier-based approach [3] because we expect further challenges in developing full autonomous navigation for a small hexapod walking robot. The frontier-based spatial exploration is principally composed of several parts from which the most important are the localization, mapping, and selection of the next navigational goal towards which the robot is navigated [11]. Therefore, the following parts of this section work are dedicated to an overview of the existing solutions to these parts.

### 2.1 Mapping

An important part of the spatial exploration is a suitable data representation of the environment model, the map. Several map representations can be found in the literature [12], but the traditional and commonly used approach is the occupancy grid [13] that divides the space into cells, and each cell represents the probability of being occupied. The pure occupancy grid is suitable for a combination of flat terrain and wheeled robot, but it is not suitable for capturing an uneven terrain traversable by a walking robot. On the other hand, the extension of the occupancy grid to full 3D representation, known as the OctoMap [14], is computationally more demanding than the elevation map proposed in [15]. Besides, further approaches exploiting properties of the elevation map have been proposed [16], [17], which steers for the selection of the elevation map as a suitable environment model representation.

Successful deployments of elevation map in exploration scenarios with walking robots are reported in [5], [6], and therefore, we have chosen the elevation map as the spatial model also because of low computational demands. An example of the build elevation map by the developed framework is shown in Figure 4. Moreover, we followed the idea of tree representation used in the OctoMap [14], and we propose to represent the elevation map by a quadtree to make its representation less memory demanding. The elevation map can also be used to assess the traversability of the explored terrain, which can be based on detecting the maximum slope as it is reported by the authors of [6] who further proposed to generate a cost map based on the distance from the obstacles to lowering the risk of driving the robot close to untraversable areas. In the herein proposed method, the traversability assessment follows [6], but the Distance Transform algorithm [18] has been employed to generate a cost map.

The navigation through the environment represented by the elevation map requires to plan a path for the robot. Multiple algorithms can be utilized and since the environment is represented by a grid-like map, graph search algorithms can be employed, such as the A\* algorithm [19] used by the authors

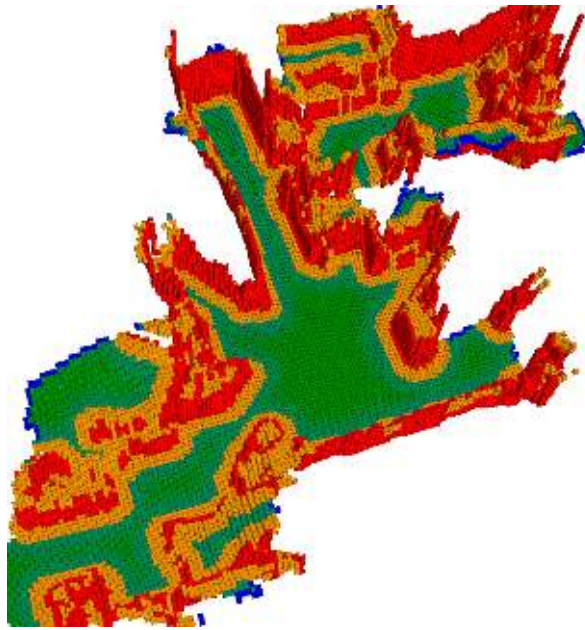


Figure 4: An example of the elevation map, where untraversable cells are shown in red, grown untraversable cells are in orange, traversable cells are in green, and frontier cells are in blue. The further visualizations of the elevation map use a similar color schema.

of [6] to plan paths to the waypoints generated based on the detected frontier cells. The detection of the frontiers can be based on [20], where two algorithms of frontier detection are described. The first method searches the whole map, while the second algorithm (called Fast Frontier Detector) detects the frontier cells only from new incoming measurements and then join them with the previously detected frontier cells. However, in very large scenarios, there can be a huge number of frontier cells; hence, clustering of the frontier cells and selection of the cluster representative has been proposed in [21] to lower the computational demands of determining possible waypoint locations. The particular next navigation goal is selected from the possible waypoint locations by a utility (or cost) function that, in the simplest case, selects the closest waypoint to the robot [3].

## ■ 2.2 Robot localization

Precise localization is necessary requirement for a mobile robot that incrementally builds a metric map of the environment. There are two main classes of the localization methods that are used to localize a mobile robot. The first class of methods is external localization methods that rely on equipment mounted in the environment, which is used to localize the robot. One of these methods is the Global Navigation Satellite System (GNSS), i.e., GPS, Galileo or GLONASS which provide the position of the robot outside of the buildings, where the satellites should ideally be at the line of sight. For indoor and GNSS-denied environments, where the satellites are not at the line of sight, a different localization method has to be used. Methods suitable for indoor and GNSS-denied localization can be based on different beacons or transmitters to localize the robot; these systems include localization estimated based on WiFi signal [22] or, e.g., ultrasound [23]. Other methods are based on optical sensors for robot tracking like total stations that measure the distance from a corner reflector attached to the robot or complex camera systems such as Vicon [24] can be used (for small scale indoor environments). Regarding cost-efficient camera-based systems, conventional cameras can be utilized to track distinctive pattern attached to the robot as in the existing WhyCon [25] or AprilTag [26] setups.

The second class of the methods represents localization methods that rely only on interoceptive

or exteroceptive sensors mounted on the robot. The interoceptive sensors include wheel encoders to localize the robot by counting the wheel turns, which is called wheel odometry, which at some extent can also be considered for multi-legged robots. In [27], the authors propose a relative localization for hexapod walking robot based on the robot model supported by measurements from IMU. The disadvantage of [27] is the necessity of the precise kinematic model, which is hard to obtain. The exteroceptive sensors can also be used for precise localization, see benchmark results reported in [28]. An example of a commonly used exteroceptive sensor for the localization is a laser range-finder (LIDAR) used, e.g., in [29]. Besides, various types of the cameras can be utilized as well [30], [31], [7]. Although LIDARs provide relatively precise localization [28] and distance measurements are more precise than measurements from the cameras, we limit ourselves to sensory equipment that can be carried by a small hexapod walking robot. Therefore, we mainly focus on vision-based localization using small cameras.

### 2.2.1 Vision-based localization

The well-known vision-based localization approach is the **Visual odometry** [32], where the initial position of the robot is known, and each next position of the robot is estimated by iteratively computing changes of the robot position based on the correspondences detected between the consecutive camera frames. The correspondences are detected as the regions in the images with a similar appearance; there are different types of regions used by localization approaches. A typical example of these regions is salient points in an image which appears at the places of significant changes of pixel intensities. The large class of methods that detect correspondences between images based on salient points is called feature-based localization. In this thesis, we focus mainly on the feature-based visual localization methods, because of their precision reported in [33], [28], or [34], and their relatively low computational demands.

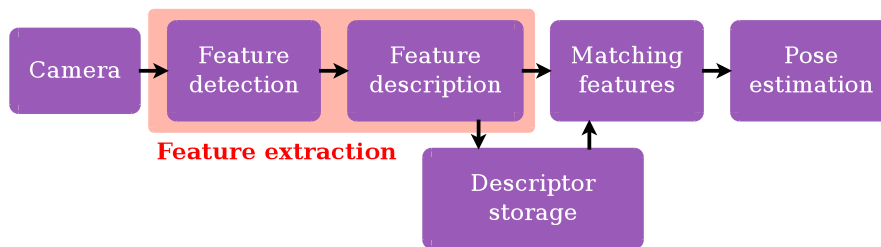


Figure 5: Overview of a simple visual odometry.

Processing blocks of the considered feature-based visual localization are depicted in Figure 5, and they operate as follows. At first, image features are extracted by a process that consists of two phases: feature detection and feature description. During the feature detection, a limited number of image features is detected, then a local representation of each detected feature, i.e., feature descriptor, is calculated. After that, descriptors of features detected in the current camera frame and feature descriptors obtained previously are compared to find the correspondences between the current and previous frames. The process of finding the correspondences based on the comparison of the feature descriptors is called feature matching. Matched features define the found correspondences between the frames that are used to compute a geometrical transformation between these frames. The resulting camera pose is computed from the initial pose and product of all following transformations calculated from the beginning frame to the current frame.

## 2.2 Visual Simultaneous Localization and Mapping

### Image features

The text above indicates that image features play a crucial role in feature-based visual localization methods. The properties of image features affect the computational requirements of the localization method and also its precision, which has been shown in [35]. In that paper, the authors show that Speeded Up Robust Features (SURF) [36] lead to more precise localization than faster to compute Features from Accelerated Segment Test (FAST) [37] combined with Binary Independent Elementary Features (BRIEF) [38], and Oriented FAST and Rotated BRIEF (ORB) [39]. However, it is worth noting that the authors of [35] processed the sensory data captured during the robotic experiment with the same framerates. However, during the online deployment, the onboard computational power of a robot is usually limited. Thus methods with lower computational requirements may run on higher framerates, which also improves the precision of the localization. Moreover, in [40], we have shown that if there is enough computational power to process all the frames incoming from the sensors, the trajectory estimate provided by visual localization can reach very high precision. We found out that in some cases, it is better to use features that are fast to detect like ORB, because it enables the localization system to provide pose of the robot with a higher frequency than in the case of more demanding features, and the localization system less likely loses the robot pose.

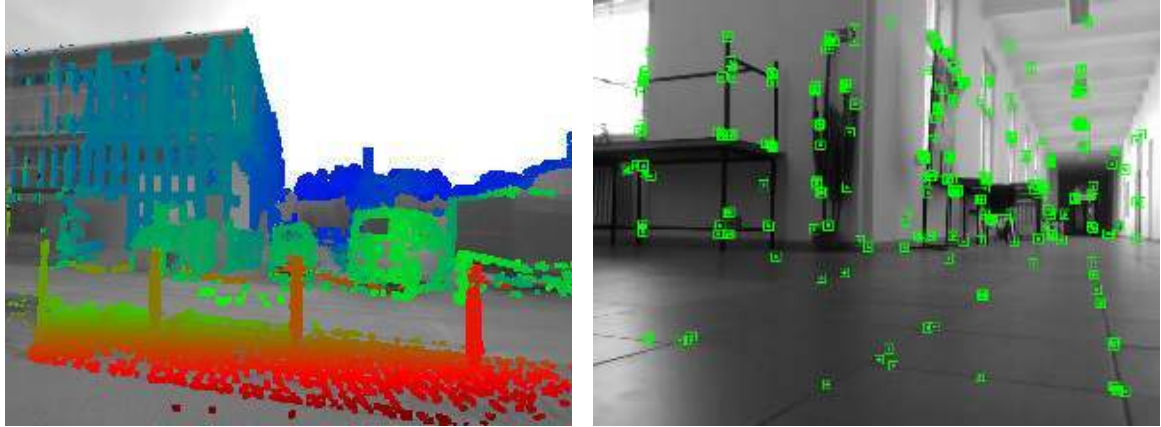
### ■ 2.2.2 Visual Simultaneous Localization and Mapping

Simultaneous Localization And Mapping (SLAM) [41] are methods that employ more complex techniques of searching correspondences between frames to allow more reliable and more accurate localization than simple visual odometry [32]. The fundamental technique of SLAM is a loop closure, which addresses the problem of the localization drift by observing a scene that was already seen. The localization drift is produced by the error accumulated from all the calculated transformations along the whole path of the robot. If the robot closes the loop, the error accumulated in the loop can be decreased. Therefore, SLAM is more suitable for long-term localization than the visual odometry, especially if the robot revisits some places.

Existing methods of visual SLAM can be divided based on several criteria: full/online, sparse/dense, and direct/indirect. Full SLAMs consider the whole path of the robot, which enables the SLAM system to recalculate the path when a loop closure is detected, and also improve the underlying map. Whereas the online SLAM seeks to estimate only the current camera pose, which is computationally less demanding. The amount of the information taken from an image can be used to further classify SLAM systems as sparse or dense. SLAM systems classified as dense use all or most of pixels in an image to estimate a position of the camera and detailed map of the environment. The advantage of dense SLAM is that the detailed underlying map can also be used for other purposes, e.g., for collision avoidance. However, the computational requirements of dense methods are high due to the high number of processed points. An example of a dense SLAM method is presented in [42], where the authors propose to use almost all available pixels of the RGB-D image to estimate a pose of the RGB-D camera. A method that uses a lower amount of points, and thus can be considered as the semi-dense method is, for example, the LSD-SLAM [43].

SLAM systems classified as sparse use a small number of image pixels to calculate a pose of the camera, and construct a coarse map of the environment. The comparison between the number of pixels used by semi-dense and sparse methods is shown in Figure 6. The examples of sparse SLAMs are ORB-SLAM2 [44], [7], RGBDSLAMv2 [45] or S-PTAM [46]. The way the image pixels are used divides the SLAM methods to direct and indirect methods. Direct methods use directly the intensity of pixels to estimate the camera position, e.g., from the depth of the scene, which is done in LSD-SLAM [43]. On the other hand, the indirect methods use a special representation of the interesting regions (e.g., image features) instead of operating directly with pixel intensities. Although SLAM methods can lower the position error if the robot closes the loop, the performance of the visual





(a) Pixels used for depth map construction by the LSD-SLAM (retrieved from [43])

(b) Image features detected by ORB-SLAM2

Figure 6: Example of processed pixels by different SLAM methods.

localization can be further enhanced by sensory fusion.

### 2.2.3 Sensory fusion for vision-based localization

Various approaches to sensory fusion can be identified in the literature. In this work, we consider mainly the approaches related to the robotic localization and sensory fusion for estimation of the orientation. The authors of [47] proposed to improve the Stereo Parallel Tracking and Mapping (S-PTAM) [48] by the Extended Kalman Filter (EKF) for a fusion of the sensory measurements from IMU to predict positions of the previously seen image features. Hence, the improved S-PTAM [47] is more reliable and can provide the position of the robot at a higher frequency, even when the robot moves unsmoothly. A different way of sensory fusion is to separate the sensory fusion algorithm from the localization module completely. In [8], we have proposed to use Kalman filter (KF) to fuse the output from ORB-SLAM2 and with the GNSS to lower the drift produced by the visual localization system. The approach [8] of sensory fusion has the advantage of being less dependent on the localization method. Thus, the localization method can be easily replaced.

Many other approaches use Kalman filter for sensory fusion as well, e.g., [49] to name some. The main functionality of Kalman filter is divided into two phases [41]. The first phase is the *prediction phase* during which the filter predict means of the states and their uncertainty solely based on the model of the system. During the second phase, the sensory measurements are used to correct the current estimate of the system state. Another approach, very similar to Kalman filter is Information filter utilized in [50]. From the usage perspective, the difference between Kalman filter and Information filter is the complexity of the *prediction phase*, and *correction phase*, which are listed Table 1.

Table 1: Complexities of Kalman filter and Information filter

Phase	Kalman filter	Information filter
<i>Prediction phase</i>	$O(n^2)$	$O(n^{2.4})$
<i>Correction phase</i>	$O(n^{2.4})$	$O(n^2)$

$n$  is the size of the state vector [41]

Besides the robot model, the sensory fusion for the estimation of the orientation has to take into account the representation of the orientation, which can be based on quaternion. The approach [51] proposes to use EKF for the sensory fusion of the orientation represented by quaternions with IMU

## *2.2 Sensory fusion for vision-based localization*

measurements. In addition to sensory fusion for visual localization described in the literature, there is a novel off-the-shelf commercially available sensor Intel RealSense T265 [9] with an inbuilt unit capable of vision-based localization enhanced by the sensory fusion with IMU.

## Chapter 3

# Problem Statement

The goal of this work is to propose a solution for autonomous spatial exploration and employ it on the hexapod walking robot in real-world scenarios. The complexity of the exploration task leads us to decompose the proposed solution into several parts: spatial mapping, exploration strategy, and algorithm for path following. The proposed framework is developed for real robots, especially for hexapod walking robots and robots with a differential drive. The use of the proposed framework requires a robot with a ROS [52] driver<sup>1</sup>.

The spatial mapping of the environment requires precise and reliable localization. Therefore, the localization fusion and two different state-of-the-art vision-based localization systems are detailed in the thesis. Both vision-based localization systems have been experimentally evaluated, and the more suitable localization system has been used during the deployment of the hexapod walking robot in autonomous exploration scenarios. The key criterion of the comparison of the localization systems is their precision. Thus, we describe metrics of the localization precision in the following text to make the thesis more self-contained.

### 3.1 Metrics of the localization precision

In principle, the localization performance metrics that can be found in the literature [53], [54], [34] assumes that the precision of the localization is measured from the comparison of a robot trajectories estimated by the localization system under the test and a reference localization system. The trajectories provided by localization systems used within this thesis are sequences of poses, where each pose is a time-stamped vector

$$\mathbf{p}_{vec} = \begin{bmatrix} \mathbf{q} \\ \mathbf{t} \end{bmatrix}, \quad (1)$$

where  $\mathbf{q}$  is quaternion, which represents the orientation of the robot, and  $\mathbf{t}$  is the position of the robot.

The localization system under the test generally provides the trajectory estimate with a different frequency than the reference localization, which is induced by, e.g., usage of different sensors. However, the comparison of two trajectory estimates proposed in [53] require to have both trajectories **time-synchronized**, i.e., all the poses of the trajectory estimate are compared with poses of the ground truth acquired at the same times. The authors of the evaluation method described in [53] propose to find for each pose of an estimated trajectory, the pose of the ground truth which has the closest timestamp. After the time-synchronization, we have two sequences of  $n$  poses ordered by timestamps in the format of (1), where for each pose of the estimated trajectory, there is a pose of the ground truth with the same timestamp.

The metrics described in [53] are defined for trajectories, where all poses are represented by SE(3) matrices. A robot pose in SE(3) is thus composed of

$$\mathbf{P}_{SE(3)} = \begin{pmatrix} \mathbf{R}_{3 \times 3} & \mathbf{t}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{pmatrix}, \quad (2)$$

where the vector  $\mathbf{t}$  is the robot position, and  $\mathbf{R}$  is SO(3) rotation matrix which represents an orientation

---

<sup>1</sup>ROS driver of a robot is not a part of the solution, and it supposed that the driver controls the robot motion by forward and angular velocities.

### 3.1 Metrics of the localization precision

of the robot <sup>2</sup>. Thus, the poses of the trajectory estimate and the ground truth are transformed from the format (1) to SE(3).

Let the trajectory estimate with poses in SE(3) be denoted as  $\mathbf{P}' = \{\mathbf{P}'_1, \mathbf{P}'_2 \dots, \mathbf{P}'_n\}$ , and the corresponding time-synchronized ground truth trajectory be denoted as  $\mathbf{Q} = \{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_n\}$ . Generally, the localization system under the test provides a trajectory estimate in a different coordinate frame than the ground truth. Therefore, we introduce a trajectory estimate  $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n\}$ , which is the trajectory estimate transformed into the coordinate frame of the ground truth reference trajectory. The first way to obtain the transformation  $\mathbf{S}$  from  $\mathbf{P}'$  to  $\mathbf{P}$  is to assume that the error of the first pose of the estimated trajectory is zero, then  $\mathbf{S}$  is computed from the first poses of the trajectory estimate and first pose of the ground truth as  $\mathbf{S} = \mathbf{Q}_1 \mathbf{P}'_1^{-1}$ . However, the absolute error of such transformation suffers from misalignment of the trajectory, e.g., if the orientation provided by the localization system under the test is not reliable even at the beginning of the trajectory. Therefore, we propose to use a second way to obtain  $\mathbf{S}$ . The second way to get the transformation  $\mathbf{S}$  is to seek a transformation that minimizes the root mean squared error (RMSE) between the estimated trajectory and the ground truth [53].

When we obtain trajectories  $\mathbf{P}$  and  $\mathbf{Q}$ , both metric of the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) can be employed to evaluate the precision of the trajectory estimate. ATE is defined by the equation [53]

$$\mathbf{F}_i = \mathbf{Q}_i^{-1} \mathbf{P}_i, \quad (3)$$

ATE represented by the SE(3) matrices  $\mathbf{F}_i$  can be divided into the translational part of ATE  $\mathbf{F}_{i,t}$  and the rotational part of ATE  $\mathbf{F}_{i,\phi}$ , as it was used in [35]. The translational part of ATE measures distances between the estimated poses and poses of the ground truth. Similarly, the rotational part of ATE measures an error of the robot orientation.

Statistical measures of the mean, root mean squared (RMS), standard deviation and maximum and minimum values are used in different benchmarks for a comparison of the trajectories. In this work, we use the mean of the translational part of ATE [53] given as

$$\overline{\text{ATE}}_t = \frac{1}{n} \sum_{i=1}^n \|\mathbf{F}_{i,t}\|, \quad (4)$$

$$(5)$$

RPE is calculated using [53]

$$\mathbf{E}_i = (\mathbf{Q}_i^{-1} \mathbf{Q}_{i+\Delta})^{-1} (\mathbf{P}_i^{-1} \mathbf{P}_{i+\Delta}), \quad (6)$$

where  $\Delta$  is a pre-defined interval. A straightforward selection used in this thesis is  $\Delta = 1$  for which the RPE measures the error between the consecutive frames. There are also other possible selections of  $\Delta$ , e.g., adaptive selection of  $\Delta$  used in [28] for which the distance traveled between the compared poses is constant. Besides, the authors of [53] propose to select  $\Delta$  as a set of values over all possible time intervals.

The statistical measure of the mean is used for RPE as for ATE, i.e.,

$$\overline{\text{RPE}}_t = \frac{1}{k} \sum_{\Delta} \left( \frac{1}{m} \sum_{i=1}^m \|\mathbf{E}_{i,t}\| \right), \quad (7)$$

$$(8)$$

where  $m = n - \Delta$  and  $k$  is the number of  $\Delta$  time intervals,  $\mathbf{E}_{i,t}$  is the translational part of RPE.

The reason to use RPE in addition to ATE is to evaluate a local drift of the localization, which, however, captures only the local consistency of the map.

<sup>2</sup>Representation of poses by the SE(3) matrix can be expressed as a transformation of the standard basis of the space  $\mathbb{R}^3$  from the beginning of the coordinates.

## Chapter 4

# Proposed Method

The exploration task is a complex problem, and therefore, we divided the proposed robotic exploration framework into three main modules: *localization*, *exploration*, and *path following*. The connections between the modules are visualized in Figure 7. The sensory measurements are processed by

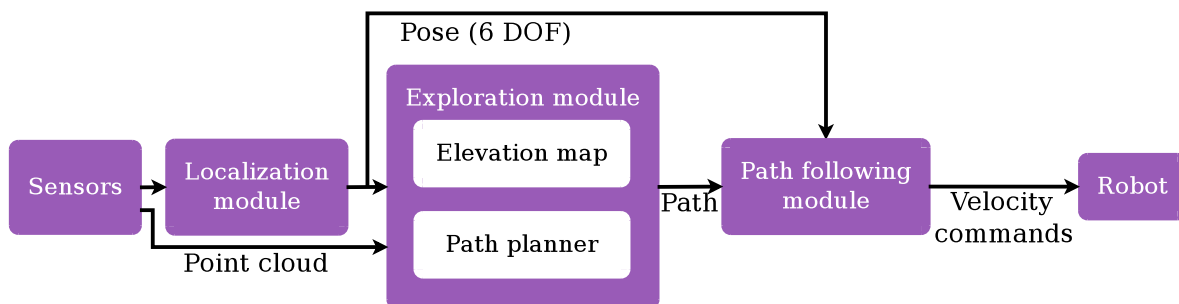


Figure 7: Architecture of the proposed exploration framework.

the *localization module* that provides the robot pose for the *exploration* and *path following* modules. The sensors also provide point clouds that are together with the robot pose used to build an elevation map of the robot surroundings. Next, paths to the determined waypoints that lead to the exploration of the unknown terrain are selected within the *exploration module*. Finally, the *path following module* ensures that the robot moves according to the path planned by the *exploration module*.

The Robot Operating System (ROS) [52] has been selected as middleware to mediate interfaces between the modules. The ROS connects the modules using a system of messages. Although custom messages can be created, only the standard ROS messages have been used for the communication between the modules to increase the reusability of the implemented software components. Moreover, the standard ROS messages enable the usage of various tools, e.g., 3D visualization tool RViz, and existing simulators.

### 4.1 Exploration

The exploration module is responsible for repeated determination of the path that is used by path following module to explore unknown parts of the environment. Thus, the exploration can be expressed as an iterative procedure that based on the current map of the environment being explored, determines a set of possible waypoints, selects the next most suitable navigational location to explore the environment, and plan a path along which the robot is navigated. Since the robot velocity is limited, it is not necessary to update the plan frequently, and therefore, navigational goal might updated at the frequency of 5 Hz, which is in accordance of the mapping module that is detailed in Section 4.3. In particular, for each new update of the elevation map, it is examined if the actual path of the robot is feasible. If the path is no longer feasible, new navigational goal is determined, otherwise the plan is updated every 8 s.

The procedure for determining new waypoints follows the frontier-based approach [3] and the next goal location is selected as the closest waypoint to the robot. The metric used to determine the distance between a possible waypoint and the robot is the cost of the shortest path to the waypoint produced by

## 4.2 Vision based Localization

the planner; but we further employ traversability assessment and safety among the obstacles. However, Euclidean distance is utilized to presort the possible waypoints to speed-up the waypoint selection, and the cost of a possible path is computed for the five closest reachable waypoints at maximum. Finally, the exploration terminates if there are no more possible waypoints to follow.

### ■ 4.2 Vision based Localization

The feature-based visual localization ORB-SLAM2 [7] has been chosen from the existing localization methods overviewed in Section 2.2 as the localization method suitable for the hexapod walking robot. Although ORB-SLAM2 is reported to be relatively accurate localization system [28], we propose several improvements including the localization fusion. Moreover, the new embedded localization system the Intel RealSense T265 [9] (further referred to as T265 for short), has been considered as an alternative to ORB-SLAM2. Because there are (as far as we know) no experimental evaluations of T265 in a scenario with the hexapod walking robot, we provide an experimental evaluation of the localization precision and reliability provided by T265 and also a comparison with ORB-SLAM2 in Section 5.4. Based on the results of the experimental evaluation, the more suitable approach has been used during the experimental deployment of the proposed exploration framework in real-world experimental scenarios. Both localization systems are briefly described in the rest of this section.

#### ■ 4.2.1 ORB-SLAM2

The ORB-SLAM2 has been selected as a suitable localization method mainly because of (i) it is publicly available<sup>3</sup>; (ii) its precision [28]; (iii) and its ability to use very small sensors that are suitable to be mounted on our small walking robot. In the rest of this section, we briefly describe ORB-SLAM2 to make the text more self-contained.

The architecture of ORB-SLAM2 can be divided into three parts (see Figure 8)

- tracking,
- local mapping,
- loop closing;

that run in parallel.

The tracking provides the camera pose for each new frame, and it is responsible for a keyframe selection. More specifically, the localization of the camera is based on the visual odometry enhanced by an optimization using the Bundle adjustment [55], and it works as follows. At first, features in a new frame are detected. The features are then tried to be matched with features from the previous frame. Feature matching is accelerated by tracking of features, which means that each feature is matched with features in a close neighborhood of the presumed position of the feature in the next frame. Thus, the feature does not have to be matched with all newly detected features. The reobservability of the previously seen features is supported by searching the covisibility graph, which is shown in Figure 9. If the matching phase fails, then ORB-SLAM2 employs a module for the place recognition to relocalize the camera. Tracking is also in charge of a keyframe selection which happens when the frame contains a sufficient quantity of new features. Thus, the keyframe is the frame which stores a sufficient amount of new information about the environment.

The keyframes are processed by the local mapping using Bundle adjustment to optimize the local map. Local mapping also extracts the most important features from the keyframes and put them into the global feature map, which is used when tracking is lost.

---

<sup>3</sup>ORB-SLAM2 is available at <http://webdiis.unizar.es/~raulmur/orbslam/>.

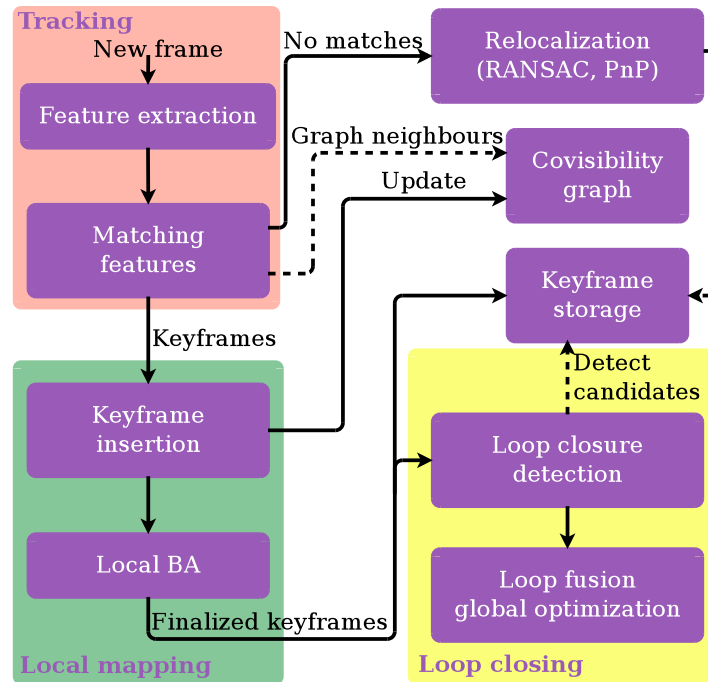


Figure 8: ORB-SLAM2 architecture.

A loop closure detection is based on detecting sets of features that were already observed, which is used to improve the precision of the localization [41]. ORB-SLAM2 detects loop closures for each keyframe, and when a loop closure is detected, the algorithm calculates error accumulated in the loop. Then, the covisibility graph is corrected and duplicated features in the feature map are merged.

The system uses ORB features [39] that are classified as close or far based on a distance from the camera center. Distances of the close features can be reliably estimated to be used for the visual odometry, especially for the calculation of the camera translation. On the other hand, a distance of far features cannot be reliably estimated, but these features are usually suitable for the loop closure.

Based on our experience with the experimental deployment of ORB-SLAM2 [8, 33], we improved its ROS interface as follows. First, we enable to process compressed RGB images and publish localization for each processed frame. Besides, we propose a method of sensory fusion for the localization.

## 4.2 Sensory fusion for the localization

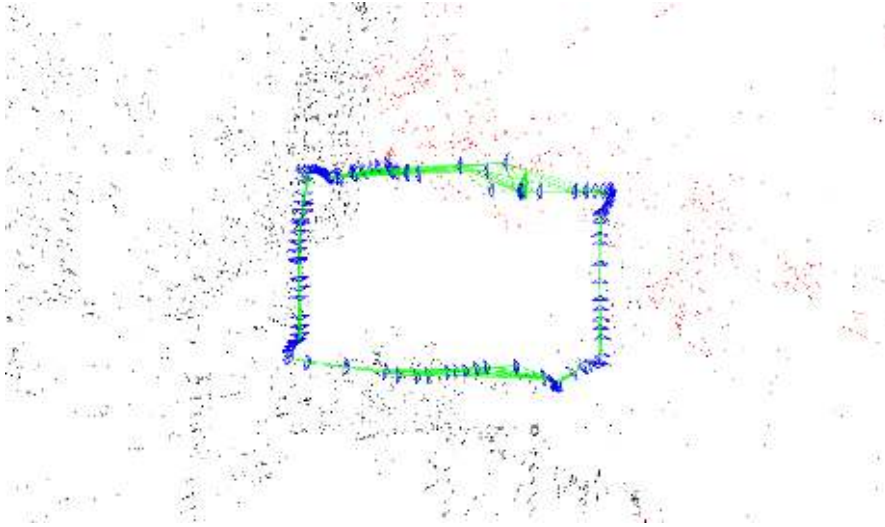


Figure 9: An example covisibility graph produced by ORB-SLAM2. Features are shown as small dots; green connections indicate reobserved features in the specific keyframes (blue elements).

### ■ 4.2.2 Sensory fusion for the localization

Unsmooth motion, temporal lack of image features, or localization drift affect the precision of the vision-based localization, and in the worst case, the localization system can fail to localize the robot. The embodiment of these issues can be partially overcome by careful parametrization of the selected visual localization method [33]. In this thesis, we propose to use the sensory fusion to get more reliable and even more precise localization. The proposed sensory fusion for the localization is based on approaches described in Section 2.2.3, especially on our recent work [8]. We propose to divide the fusion for the localization into two parts; the first part is responsible for fusion of the robot position, and the second part is in charge of the orientation fusion. Both parts are described in the following parts of this section.

#### Sensory fusion for the position estimation

The proposed sensory fusion is designed for the robot position estimated by the incremental localization that is further assisted by measurements from external localization system, e.g., GNSS, but it can be easily extended to use a measured acceleration of the robot or any other source of the robot position estimation. The incremental localization is supposed because the proposed exploration technique does not explicitly take into account the uncertainty of the robot position. Thus, the SLAM system might not close the loops<sup>4</sup>, since it is not explicitly driven to do so.

The core of the position fusion is **Kalman filter**. In particular, we consider a linear and discrete-time model of the robot that can be described by the state equation

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k. \quad (9)$$

---

<sup>4</sup>If the SLAM does not close the loops it drifts similar to incremental localization.



Kalman filter itself can be defined by the following equations [41]

$$\bar{\boldsymbol{\mu}}_k = \mathbf{A}_k \boldsymbol{\mu}_{k-1} + \mathbf{B}_k \mathbf{u}_k \quad (10)$$

$$\bar{\boldsymbol{\Sigma}}_k = \mathbf{A}_k \boldsymbol{\Sigma}_{k-1} \mathbf{A}_k^T + \mathbf{R}_k \quad (11)$$

$$\mathbf{K}_k = \bar{\boldsymbol{\Sigma}}_k \mathbf{H}_k^T (\mathbf{H}_k \bar{\boldsymbol{\Sigma}}_k \mathbf{H}_k^T + \mathbf{Q}_k)^{-1} \quad (12)$$

$$\boldsymbol{\mu}_k = \bar{\boldsymbol{\mu}}_k + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \bar{\boldsymbol{\mu}}_k) \quad (13)$$

$$\boldsymbol{\Sigma}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \bar{\boldsymbol{\Sigma}}_k. \quad (14)$$

For our description,  $\bar{\boldsymbol{\mu}}_k$  and  $\bar{\boldsymbol{\Sigma}}_k$  are predictions of mean and covariance of the robot state vector  $\mathbf{x}_k$  based on the last estimated mean  $\boldsymbol{\mu}_k$  and covariance  $\boldsymbol{\Sigma}_k$  of the robot state vector. The predictions can be made asynchronously with the correction phase represented by (12), (13), and (14), e.g., in the case the robot position is needed in an arbitrary time. The matrix  $\mathbf{R}_k$  represents the uncertainty of the robot model, and the vector  $\mathbf{z}_k$  is the measurement obtained according to

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \boldsymbol{\delta}_k, \quad (15)$$

where  $\boldsymbol{\delta}_k$  is the Gaussian noise with the zero mean and covariance  $\mathbf{Q}_k$ .

We propose to use a simplified model of the robot with the robot body motion that can be described as

$$\begin{bmatrix} \mathbf{p}_{k+1} \\ \mathbf{v}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & t_s \cdot \mathbf{I}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix} + \begin{bmatrix} \frac{t_s^2}{2m} \cdot \mathbf{I}_{3 \times 3} \\ \frac{t_s}{m} \cdot \mathbf{I}_{3 \times 3} \end{bmatrix} \mathbf{f}_k, \quad (16)$$

where  $t_s$  is the sampling time of the discretization,  $m$  is the robot weight,  $\mathbf{f}_k$  is the force that moves the robot,  $\mathbf{p}_k$  and  $\mathbf{v}_k$  are the position and velocity of the robot, respectively.

The position provided by incremental localization drifts, and therefore, its uncertainty grows to infinity. On the other hand, estimates of the transformations between the consecutive robot positions are relatively accurate, and uncertainty of these transformations is bounded because it is not affected by the localization drift. Hence, we propose to fuse differences between the consecutive robot positions instead of using directly the robot position provided by the incremental localization. The straightforward way to incorporate differences of the consecutive positions is to substitute the robot velocity using

$$\mathbf{v}_k = \frac{\mathbf{r}_k - \mathbf{r}_{k-1}}{t_s} = \frac{\Delta \mathbf{r}_k}{t_s}, \quad (17)$$

where  $\mathbf{r}_k$  is the 3 DoF position of the robot provided by the incremental localization system. Then, the model of the robot (16) changes to

$$\begin{bmatrix} \mathbf{p}_{k+1} \\ \Delta \mathbf{r}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{I}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \Delta \mathbf{r}_k \end{bmatrix} + \begin{bmatrix} \frac{t_s^2}{2m} \cdot \mathbf{I}_{3 \times 3} \\ \frac{t_s}{m} \cdot \mathbf{I}_{3 \times 3} \end{bmatrix} \mathbf{f}_k. \quad (18)$$

The force  $\mathbf{f}_k$ , which moves the robot, can be estimated from the robot acceleration measured, e.g., by IMU or from the control signals to the motors of the robot. If  $\mathbf{f}_k$  is not available, it is possible to follow the tracking scenario like in [56], which considers  $\mathbf{f}_k = 0$ . In that case, the model of the robot estimates the robot position by summing the robot position with the last known position difference; thus, it is a variant of the constant velocity motion model.

If the position measurements provided by the external localization  $\mathbf{z}_{p,k}$  and position differences  $\mathbf{z}_{\Delta r,k}$  computed from the incremental localization are provided synchronously, i.e.,  $\mathbf{z}_k = [\mathbf{z}_{p,k} \ \mathbf{z}_{\Delta r,k}]^T$ , we can incorporate the measurements using (15). The uncertainty of the measurements  $\mathbf{z}_k$  is represented by the covariance matrix  $\mathbf{Q}_k$  that is together with the matrix  $\mathbf{H}_k$  can be composed as

$$\mathbf{H} = \mathbf{H}_{k=1,2,\dots} = \mathbf{I}_{6 \times 6}, \quad \mathbf{Q}_k = \begin{bmatrix} \mathbf{Q}_{p,k} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{Q}_{\Delta q,k} \end{bmatrix}.$$

## 4.2 Sensory fusion for the localization

The uncertainty of the external localization and the uncertainty of the incremental localization are represented by  $\mathbf{Q}_{p,k}$  and  $\mathbf{Q}_{\Delta r,k}$ , respectively. In most cases, the external localization and incremental localization provide the measurements asynchronously. Therefore, the correction phase of Kalman filter is done separately for  $\mathbf{z}_{p,k}$ , and  $\mathbf{z}_{\Delta r,k}$  measurements.

We have evaluated the proposed method of sensory fusion for the robot position in [8], where the incremental localization has been provided by ORB-SLAM2 for RGB-D data from a small aerial vehicle, and the external localization was provided by simulated unreliable GPS signal. Note, Kalman filter enables to predict the position of the robot independently on the incoming measurements. It is specifically useful if the visual localization provides position at a low frequency, and the synchronization with measurements from a range sensor is required. Since the proposed sensory fusion is only for the position estimation, we also propose the sensory fusion for the orientation to increase the accuracy and reliability of the whole 6 DoF pose estimation of the robot.

### Sensory fusion for the robot orientation

The main motivation of the sensory fusion for the orientation is to overcome a failure of the visual localization at least for a short period until the visual localization is reset or some image features become visible. There are several ways to do sensory fusion based on the available sensory equipment and orientation representations.

The ROS framework used within the exploration framework use quaternions to represent the robot orientation, thus we propose to use Extended Kalman filter (EKF) for sensory fusion based on [51]. The principle of filtering is similar to the previously shown position fusion; the main difference is in the state equation. Thus, we show how to derive the state equation in what follows.

Let we have an angular velocity  $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$  of the robot provided by IMU. Relation between the angular velocity and orientation represented by quaternion  $\mathbf{q} = [q_x, q_y, q_z, q_w]$  can be described by the differential equation [51]

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q}, \quad \boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y & \omega_x \\ \omega_z & 0 & -\omega_x & \omega_y \\ -\omega_y & \omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}. \quad (19)$$

Next, we can discretize (19) to

$$\mathbf{q}_{k+1} = \exp\left(\frac{1}{2}\boldsymbol{\Omega}_k(\boldsymbol{\omega}_k)t_s\right)\mathbf{q}_k, \quad (20)$$

where  $t_s$  is the sample time and  $\exp()$  denotes the matrix exponential. For a short sample time, the matrix exponential can be approximated using its first-order and second-order items of Taylor series [57] and we can get

$$\mathbf{q}_{k+1} = \left(\mathbf{I}_{4 \times 4} + \frac{1}{4}\boldsymbol{\Omega}_k(\boldsymbol{\omega}_k)t_s\right)\mathbf{q}_k. \quad (21)$$

Hence, (21) is the state equation, and the rest is just an application of EKF approach.

The alternative way to the depicted fusion with IMU is to directly use Attitude Heading Reference System (AHRS), which provides orientation as a result of the internal sensory fusion based on the gyroscope, accelerometers, and magnetometer sensors. Since the AHRS provides an orientation, it can be directly used instead of the orientation provided by a visual localization. Then, before the position fusion by Kalman filter, the position differences provided by the visual localization have to be projected into the coordinate frame defined by the AHRS and fused position. The advantage of fusion with AHRS is the removal of the localization drift produced by the visual localization.

### ■ 4.2.3 Tracking camera Intel RealSense T265

The proposed alternative localization system to the ORB-SLAM2 is the tracking camera Intel RealSense T265 shown in Figure 10. The visual localization provided by the T265 is assisted by the sensory fusion with IMU. Moreover, the tracking camera has fisheye lenses that ensure a high overlap of consecutive images to improve the reliability of the visual SLAM. The sensory fusion enables the camera to estimate the robot position even when the images from the camera temporary do not provide any useful information.



Figure 10: Tracking camera Intel RealSense T265.

Cameras of T265 have IR filters able to eliminate the influence of the IR light emitted by active RGB-D camera the Intel RealSense D435, and thus both cameras can be used at the same time. The advantage of the embedded localization system based on T265 is that all the computations related to the visual localization and sensory fusion are done onboard the camera, and thus a less computational power is needed to be carried by a robot, or it can be used for another demanding task such as path planning. Technical parameters of T265 are summarized in Table 2.

Table 2: Technical parameters of the Intel RealSense T265

Parameter	Value
Resolution [px]	$848 \times 800$
Max framerate [Hz]	30
Size [mm] (H/W/D)	$24.5 \times 108 \times 12.5$
Weight [g]	55
Power consumption [W]	1.5
Shutter type	Global

## ■ 4.3 Mapping and Planning

The elevation map is used as a spatial model of the robot surroundings within the proposed exploration framework. We build the elevation map incrementally from the preprocessed point clouds incoming from the sensors. The point cloud preprocessing includes removal of flawed noisy points (generated by reflections from the robot body and too distant measurements) and transformation of the point cloud to the global coordinate frame of the map using the robot pose provided by the localization module. Prior the points of a new point cloud are incorporated into the elevation map; we utilize the grid defined by the elevation map to select only the points that have the maximal height in the corresponding cell of the grid, which ensures the elevation map captures the highest obstacles on the ground. Then, the selected points are one-by-one incorporated into the elevation map as measurements of the terrain height.

Each cell  $(i, j)$  of the elevation map contains an estimate of the terrain height  $h_k$  and its variance  $\sigma_{h,k}^2$ . Thus, a newly measured height  $z_k$  and its variance  $\sigma_{z,k}^2$  for the particular cell  $(i, j)$  are fused with the current estimate of the height using Kalman filter based approach [16]

$$h_k = \frac{\sigma_{z,k}^2 h_{k-1} + \sigma_{h,k-1}^2 z_k}{\sigma_{z,k}^2 + \sigma_{h,k-1}^2} \quad (22)$$

$$\sigma_{h,k}^2 = \frac{\sigma_{z,k}^2 \sigma_{h,k-1}^2}{\sigma_{z,k}^2 + \sigma_{h,k-1}^2}, \quad (23)$$

where the variance of new measurement  $\sigma_{z,k}^2$  is calculated by the linear sensor model [16]. Cells of the elevation map, where no measurements were yet incorporated, are marked as *unknown*.

### ■ 4.3.1 Map representation

Because the exploration framework is supposed to be used in large scale scenarios, we put extra effort to find an efficient and effective map representation. There are several ways to represent elevation maps, and the straightforward approach is to represent the elevation map as a uniform grid. However, a more sophisticated way is to represent the map as a quadtree structure, see Figure 11, because it reduces the memory requirements. On the other hand, the main advantage of storing elevation maps

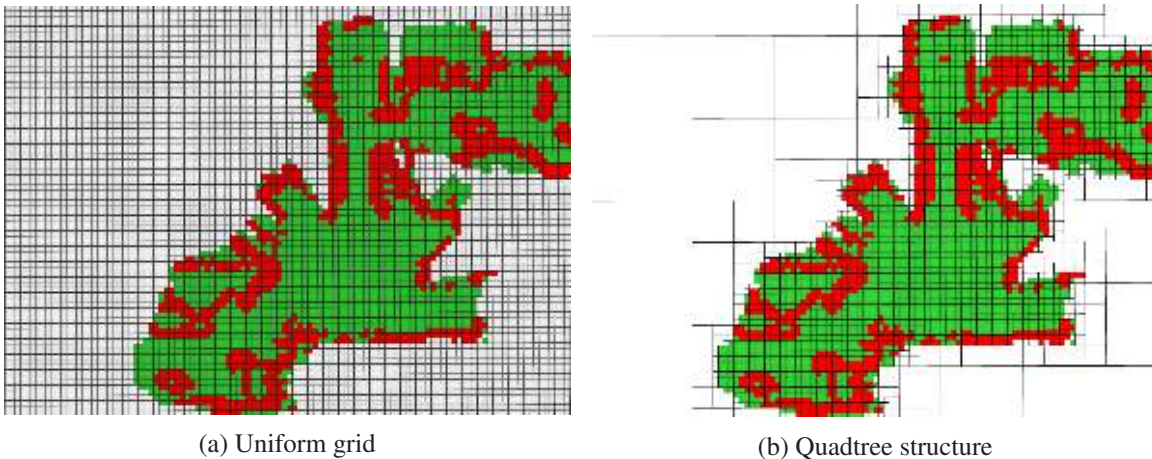


Figure 11: Elevation map representations (white color represents unknown space).

as uniform grids is the complexity of the access to individual cells, which does not depend on the size of the map and can be bounded by  $O(1)$ . However, an elevation map represented by a uniform grid uses memory inefficiently in the case of large environments with many obstacles and unreachable areas. The memory inefficiency is demonstrated in the following simple example.

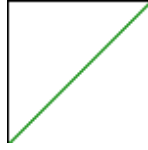


Figure 12: Example map with the size  $707 \times 707$  m and resolution 10 cm of unknown space (in white) crossed by a corridor with the size  $1000 \times 2$  m (in green).

Let the map represent a corridor 1 km long such as one that is visualized in Figure 12. The memory requirements of both map representations are depicted in Table 3. The quadtree structure requires to store links to the next nodes of the quadtree, and thus the size of a cell of the quadtree is more memory demanding than the cell representation of the uniform grid. However, if the quadtree covers the area crossed by the long thin corridor, it requires fewer cells in the total than the uniform grid representation. Since one of our motivation for the studied exploration and mapping arises from the DARPA Subterranean Challenge with such corridors and mine tunnels, we consider the quadtree representation as more suitable for the proposed exploration framework.

Table 3: Memory requirements of map representations for the scenario shown in Figure 12

Map representation:	Uniform grid	Quadtree
Cell content	Cell type, height, height variance	Cell type, height, height variance, links to next quadtree nodes
Cell size [B]	9	41
Size of known cells [MB]	2	8
Total map size [MB]	430	11

The main disadvantage of the elevation map stored as a quadtree structure is in noticeably increased complexity of accessing to individual cells, which can be bounded by  $O(\log n)$ , where  $n$  depends on the size of the environment represented by the map. It is because the quadtree has to be searched to find a particular leaf corresponding to the requested cell. The more complex access slows down map processing, especially when the elevation map is processed by multiple methods and a new point cloud regularly updates the map. We propose to overcome this issue by an additional data structure that works as a **cache**.

### Cache for the elevation map represented by the quadtree

The main assumption of the proposed cache technique is that new measurements do not affect all cells of the elevation map, but only cells that are close to the robot. Therefore, the proposed cache, which we call **QPTRmatrix**, stores the precomputed pointers to the leaves of the quadtree map in the neighborhood of the robot, see Figure 13. The minimal size of the neighborhood covered by the QPTRmatrix has to be larger than the range of the robot sensors plus an overlap, which ensures the consistency with the rest of the map. The size of the QPTRmatrix affects the computational requirements of the map update, and thus, we propose to use the QPTRmatrix as small as possible.

The quadtree structure and the QPTRmatrix enable us to represent unknown space efficiently, while still be able to run operations like convolution almost as fast as on the map stored in a uniform grid. The cost of QPTRmatrix maintaining is its computation, which is needed only when a new point cloud updates the map. In a particular deployment in the exploration, point clouds from sensors update the map at the frequency of 5 Hz. For computations performed on the Intel i5 processor with the map

### 4.3 Detection of untraversable areas

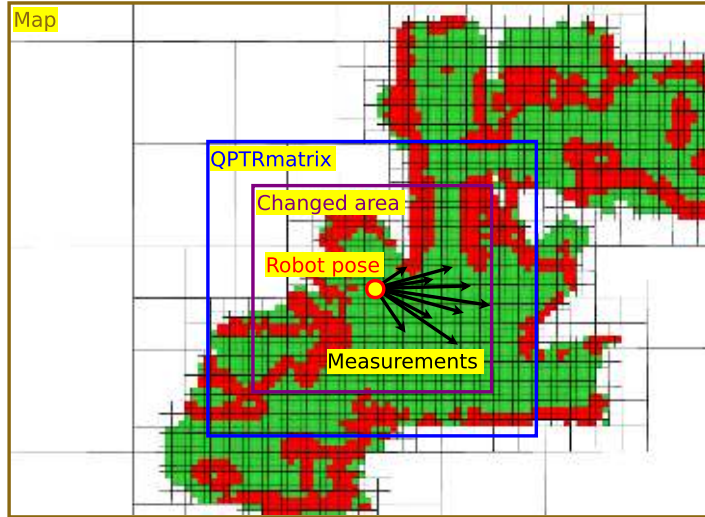


Figure 13: QPTRmatrix utilized for caching leaves of the quadtree representation of the elevation map.

resolution 5 cm and the cached area  $20 \times 20$  m large, the computation of QPTRmatrix takes approximately 10 ms. The QPTRmatrix is then used during the five operations with the map (measurement insertion, traversability assessment, growing untraversable areas, cost map computation, and frontier detection). Thus, overall, the QPTRmatrix saves approximately 50 ms as single access without the cache corresponds to the creation of the QPTRmatrix.

#### 4.3.2 Detection of untraversable areas

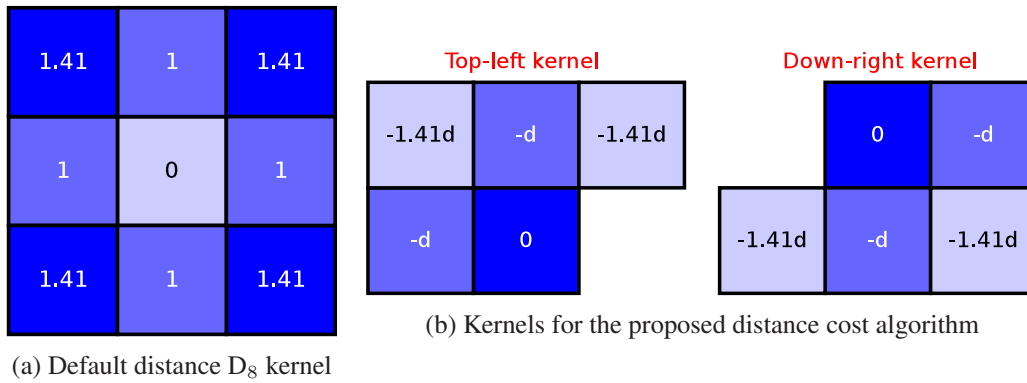
The proposed elevation map is used to detect untraversable places that should be avoided by the robot. The straightforward way to detect untraversable places is to use a model of the robot and test if there exists a configuration of the robot so that the robot legs can be placed on certain positions in the map. The disadvantage of such an approach is in the high computational requirements for searching the configuration space of the robot. Thus, we have used simplified approach based on [6], where untraversable cells  $(i, j)$  are detected by the maximal local height difference  $g_h(i, j)$  computed as

$$g_h(i, j) = \max(\{|h(i, j) - h(i - 1, j)|, |h(i, j) - h(i + 1, j)|, |h(i, j) - h(i, j - 1)|, |h(i, j) - h(i, j + 1)|\}), \quad (24)$$

where  $h(i, j)$  is the height of the cell  $(i, j)$ . A particular cell  $(i, j)$  is considered traversable if  $g_h(i, j)$  does not exceed a predefined threshold  $g_{max}$  that is estimated from the particular robot kinematics.

#### Growing untraversable areas

Since the planning algorithm plans a path for the center of the robot body, we grow untraversable cells by the radius of the robot shape circumference to consider the physical dimensions of the robot. An example of the grown untraversable areas is shown in Figure 15b.

Figure 14: Kernels based on distance metric  $D_8$ .

### 4.3.3 Cost map

The ability to detect untraversable places enables the robot to be navigated through the environment so that the robot should not reach the untraversable cells. However, the precision of the path following depends on the precision of the locomotion control and also localization. If the path following might be imprecise, it is desirable to navigate the robot to places that are as far from the untraversable areas as possible so that the chance of visiting these places is reduced. The authors of [6] proposed to use a cost map to penalize cells near the untraversable places. In our work, we propose to alter the Distance transform [18] to achieve similar behavior. Moreover, we also penalize cells near the unknown places, which are potentially dangerous, e.g., there might be a deep hole in the terrain, which is not captured by the sensors. There are more versions of Distance transform, e.g., Distance transform for quadtrees [58], but we have already employed the concept of QPTRmatrix which maps quadtree structure to the uniform grid. Hence, we can deploy methods specifically designed for simple grids.

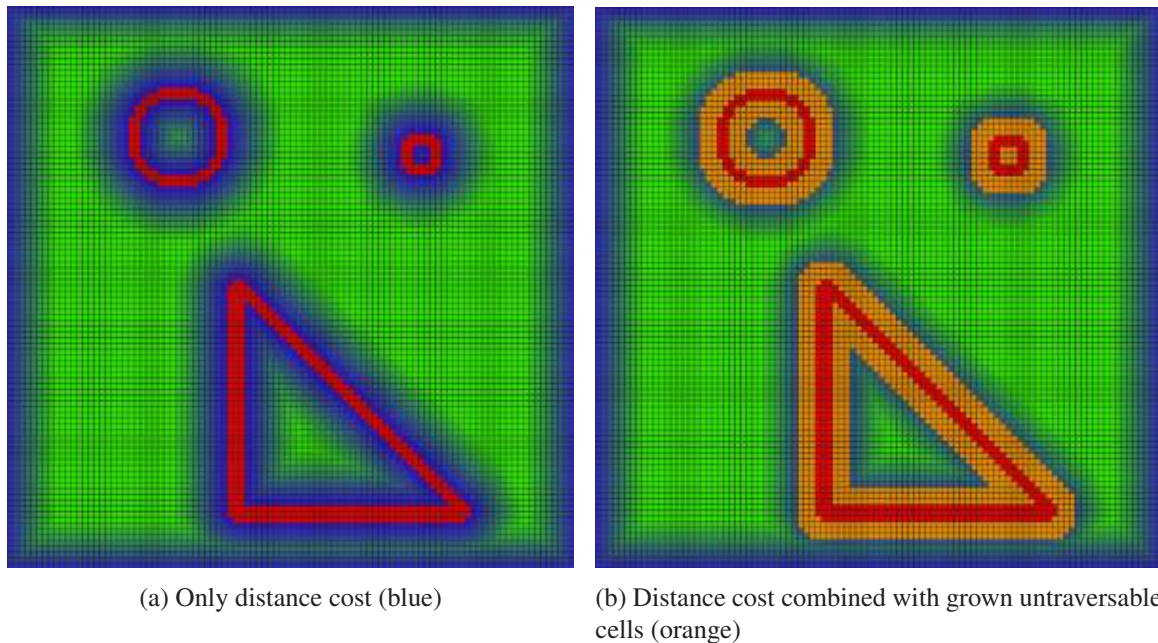


Figure 15: Distance cost ( $d = 0.07$ ,  $d_{cost} = 0.7$ ) and untraversable cells grown by 40 cm generated from red untraversable cells for map with resolution 10 cm. Note, the traversable area (green) is surrounded by an unknown space, which increases the distance cost as well.

### 4.3 Cost map

---

**Algorithm 1:** Distance cost algorithm for a grid of size  $L \times M$

---

```

1 procedure Reset ()
2   foreach  $l \in L$  do
3     foreach  $m \in M$  do
4       if  $cell(l, m) = untraversable$  or  $cell(l, m) = unknown$  then
5          $d_{cost}(l, m) \leftarrow d_{obs}$ 
6       else
7          $d_{cost}(l, m) \leftarrow 0$ 

8 procedure Top-left pass ()
9   for  $l = 1$  to  $L - 2$  do
10    for  $m = 1$  to  $M - 2$  do
11       $d_{cost}(l, m) \leftarrow \max( d_{cost}(l, m),$ 
12         $d_{cost}(l, m - 1) - d, d_{cost}(l - 1, m) - d,$ 
13         $d_{cost}(l - 1, m - 1) - 1.41d, d_{cost}(l - 1, m + 1) - 1.41d )$ 

14 procedure Down-right pass ()
15   for  $l = L - 2$  to  $1$  do
16    for  $m = M - 2$  to  $1$  do
17       $d_{cost}(l, m) \leftarrow \max( d_{cost}(l, m),$ 
18         $d_{cost}(l + 1, m) - d, d_{cost}(l, m + 1) - d,$ 
19         $d_{cost}(l + 1, m - 1) - 1.41d, d_{cost}(l + 1, m + 1) - 1.41d )$ 

```

---

The Distance transform computes the distance from the closest untraversable or unknown cell. We propose to use Distance transform algorithms [18] utilizing distance metric  $D_8$  so that it directly computes the cost which penalizes planning over the cells near the untraversable or unknown cells. The developed **distance cost algorithm** is depicted in Algorithm 1 and it is three-pass algorithm that computes distance costs using decomposition of the  $D_8$  distance kernel, see Figure 14. The two main parameters of the distance cost algorithm are (i) the cost of the untraversable and unknown cells  $d_{obs}$ , and (ii) a decrement  $d$  of the distance cost with the distance  $D_8$  from the untraversable or unknown cells. The negative costs are prevented by the initial *Reset* procedure of Algorithm 1.

An example of the resulting penalization provided by the distance cost algorithm is visualized in Figure 15a. Note, that at some distance from the obstacles, the distance cost completely vanishes, which enables to recompute distance cost on a particular area based on the QPTRmatrix and still keep the cost map consistent for the whole map. More specifically, the consistency is achieved by doing the reset phase of the distance cost algorithm at an area that is smaller than the area used in the rest of the phases. The distance cost can also be used to the speed-up growing of untraversable areas, but since the distance cost does not use the Euclidean distance metric, it provides just an approximation of the obstacle growing by a circular shape.



#### 4.3.4 Frontier detection

Possible waypoint locations for the exploration are generated from frontier cells that are detected by employing ideas of [20]. The frontier detection method called the Wavefront Frontier Detector utilizes the Breadth First Search algorithm to search almost the entire map for frontiers, which is however computationally intensive, and thus not suitable for the proposed framework. On the other hand, the second algorithm presented in [20] and called the Fast Frontier Detector detects frontiers from new incoming measurements and then join them with the previously detected frontiers, which does not depend on the size of the map, but on the range of the utilized sensors.

We have built our frontier detection method on the principle of the Fast Frontier Detector [20], and thus we detect the new frontiers from the minimal amount of data. More specifically, the proposed method detects new frontier cells only in the robot surroundings defined by the QPTRmatrix. Then, the new frontier cells are clustered [21], and the clusters are merged with the clusters in the rest of the map. The frontier cells are identified as cells that exhibit the following properties on their four-cell neighborhood.

- The cell is traversable, and at least one of its neighbors is traversable as well.
- At least one neighboring cells is unknown, i.e., the cell has unknown height.
- No neighboring cell is untraversable.
- The robot center has not yet been close to the cell, which prevents the robot from being navigated to very close waypoint locations. It also ensures that the robot does not fall off an edge between the traversable and unknown space separated by a high slope.

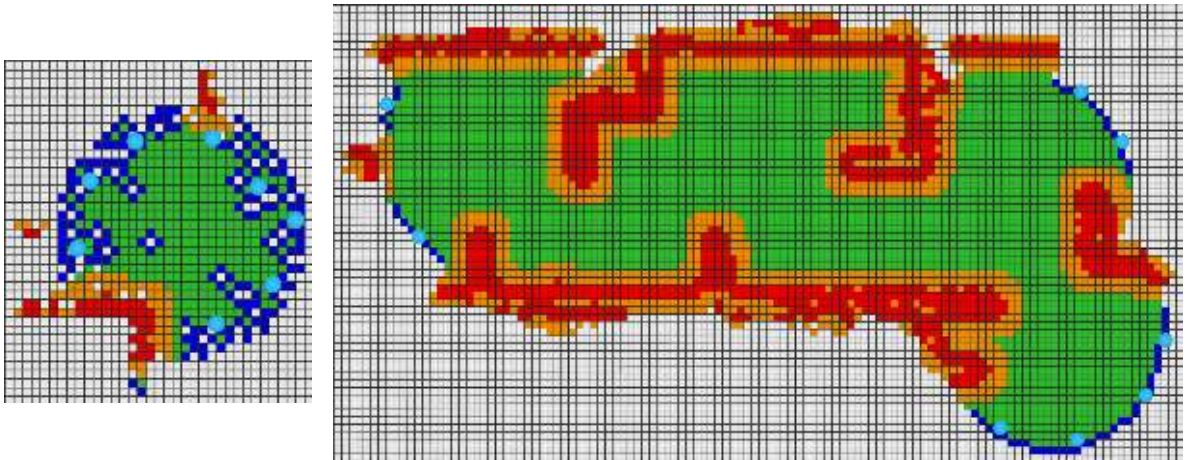


Figure 16: Frontier cells (blue) detected in an example map and clustered to close groups represented by possible waypoints (light blue). Untraversable cells are visualized as a red cells, grown untraversable cells are orange and unknown space has white color.

Based on [21,59], the frontier cell clustering is proposed to lower the computational requirements of the goal selection for the navigation. The clustering of new frontier cells is performed by Algorithm 2. The algorithm consists of two phases; the first phase divides frontiers into clusters, based on maximum distance  $c_{radius}$ , which ensures that frontier cells of one cluster are not farther than  $2 \cdot c_{radius}$ . During the second phase, the clusters with less than  $c_{min}$  frontiers are removed.

The possible waypoint location is from each cluster extracted as a position of the frontier cell closest to the average position of all frontier cells in the cluster, which ensures that valid possible waypoints are located at known cells. After the newly detected frontiers are clustered by Algorithm 2,

### 4.3 Planning over the elevation map

---

#### Algorithm 2: Proposed frontier cells clustering

---

**Input** :  $(F, c_{radius}, c_{min})$  -  $F$  set of frontier cells,  
 $c_{radius}$  - the maximal radius of each cluster,  
 $c_{min}$  - desired minimum number of cells in each cluster

**Output**:  $K = \{C_i\}$  - set of clusters  $C_i$ , where each  $C_i$  is a set of frontier cells

---

```

▷ Init clusters
1  $K \leftarrow \emptyset$ 
2 foreach  $f \in F$  do
3   if is_close_to_any_first_cell_of_clusters( $f, K, c_{radius}$ ) then
4     add_to_corresponding_cluster( $f, K, c_{radius}$ )
5   else
6      $C \leftarrow \{f\}$            // Create a new cluster with the first cell  $f$ .
7      $K \leftarrow K \cup C$ 
▷ Remove too small clusters
8 foreach  $C_i \in K$  do
9   if  $|C_i| < c_{min}$  then
10     $K \leftarrow K \setminus C_i$ 
11 return  $K$ 

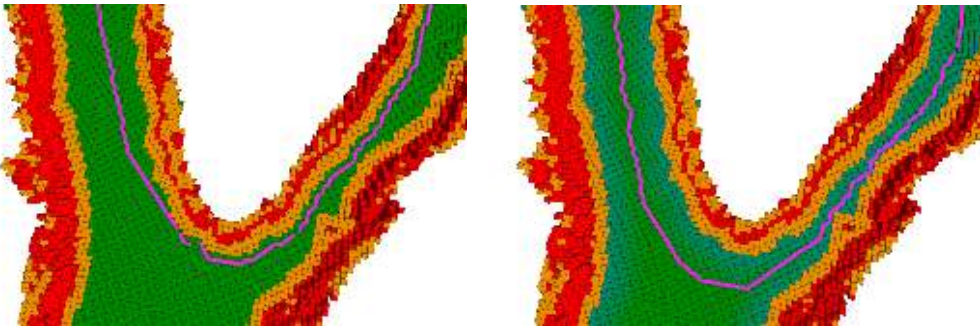
```

---

the means of the clusters generate new possible waypoints that are joined with all current possible waypoints in the whole map. The frontier detection and clustering are employed during each update of possible waypoints, which is triggered by each update of the elevation map whenever new measurements are collected. Example of the detected frontier cells and determined clusters are visualized in Figure 16.

#### 4.3.5 Planning over the elevation map

Similar to [6], the A\* planner [19] has been used for planning over the map. The A\* planner is guided by the heuristic function computed as Euclidean distance to the waypoint, and the travel cost between two neighboring nodes  $n$  and  $n - 1$  is calculated using the eight-neighborhood that is increased by the distance cost  $d_{cost}(n)$  to penalize paths close to the untraversable and unknown cells. The effect of distance cost is shown in Figure 17. If the number of A\* expansions is higher than a certain value  $a_{max}$  (e.g.,  $a_{max} = 20000$  in our case), the waypoint is considered unreachable. We avoid planning over the unknown space because it is possibly dangerous; the only exception is the case when the robot is already located in the unknown space, e.g., at the beginning of the experiment.



(a) Planning without distance cost

(b) Planning with distance cost

Figure 17: The effect of distance cost algorithm on planning.

### Path smoothing

The path generated by the planner is a sequence of coordinates of the centers of the corresponding cells. In Figure 18a, it can be seen that due to the rough resolution of the map, the path has sharp edges, which might produce discontinuities in motion actions. Such a behavior has been overcome using a sliding average filter, which effect can be observed in Figure 18. The validity of the path after the filtration procedure has to be checked, especially when the requested sliding window is wide. If the path is not valid after the smoothing, the easiest solution is to reduce the size of the sliding window. An example of the smoothed path planned by the A\* algorithm on a map obtained during an experimental deployment is shown in Figure 19. It can be seen that the planned path tends to be in the middle of the corridor and it also avoids unknown space in the map (white cells).

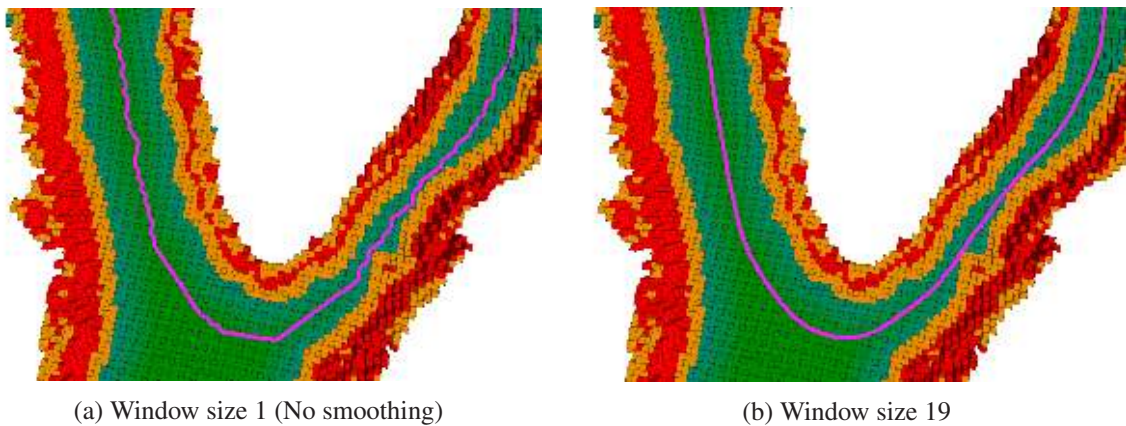


Figure 18: Paths smoothed by sliding average with different window sizes.

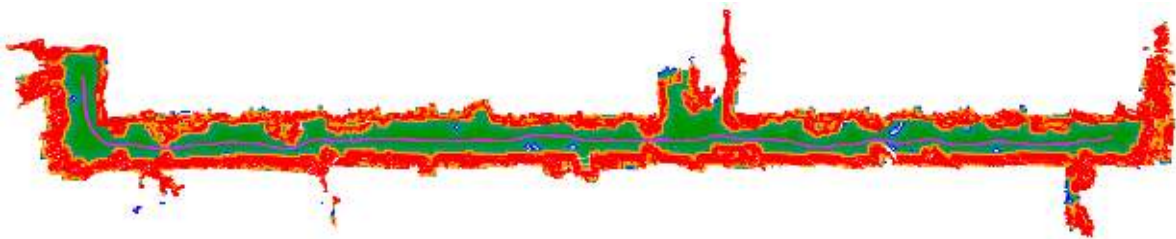


Figure 19: Path planning example (Path is the pink line, red are untraversable objects, green is traversable.)

### 4.3.6 Improvements to the exploration module

Even though the proposed exploration framework is sound, it has been further enhanced by the following practical improvements induced by the real-world deployments.

- **Automatic expansion of the map size**

In the beginning, the map saved in the quadtree structure is initialized to cover an area of a certain size. If the robot gets close to the edge of the elevation map, the map is automatically expanded by creating a new root for the quadtree map. The former root of the map is connected to the new root as one of its child nodes.

### 4.3 Improvements to the exploration module

- **Map sharing through the network**

The exploration module can produce point cloud generated from the robot surroundings that represents the resulting local elevation map around the robot. The input to the elevation map is a point cloud which is inserted according to the rules described in Section 4.3. The source of these point clouds is usually RGB-D camera or LIDAR, but it can also be a local map produced by a different exploration module. Such a setup is visualized in Figure 20, and it is beneficial if the robot is supervised through the network. When we have deployed the mapping framework

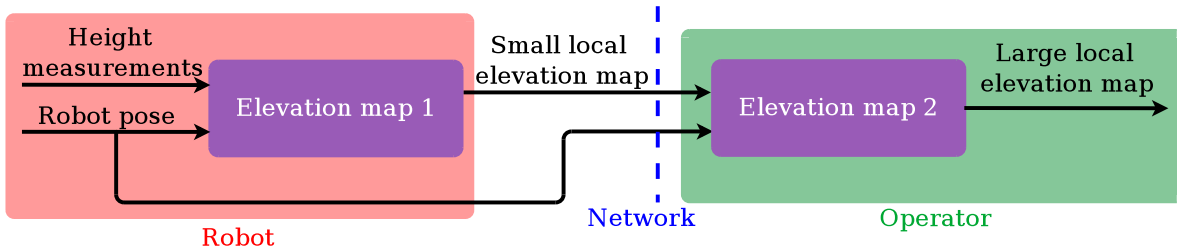


Figure 20: Connection of two instances of the elevation map module.

on a tracked robot within the setup shown in Figure 20, the required network bandwidth for sending sparse parts of the elevation map was less than 100 Kb/s for the size of the local map  $18 \times 18$  m and the map resolution 15 cm. As our future work, we propose to apply a similar principle to share the elevation maps between multiple robots.

- **Moving the robot from untraversable area**

The proposed concept of planning the collision-free path combined with the distance cost steers the robot from untraversable cells and keeps the robot in most cases at some distance from these cells. However, imperfections in localization or imprecise path following may result in situations, where the robot center is located at the place in the map that represents an untraversable cell. In such a case, the untraversable cells under the robot are temporarily cleared, so that it is possible to plan the path outside the untraversable area.

- **Interactive interface**

The elevation map module can further support planning a collision-free path to the specified goal, save the current elevation map to a file, clear the faulty map, etc. These functionalities can be activated through dedicated ROS topic. Although in most cases, the elevation map is used within the exploration framework, which works fully autonomously, we have also proposed an interactive interface which can be used by an operator to select the required functionality. The interface has been implemented using ROS interactive markers, and an example is shown in Figure 21. Because the functionalities are controlled through dedicated ROS topics, the proposed interactive module is entirely independent, and it can be used only if the operator requires some interaction. The interactive interface has been primarily deployed in experimental mines within DARPA STIX event which is detailed in Section 5.6.2.

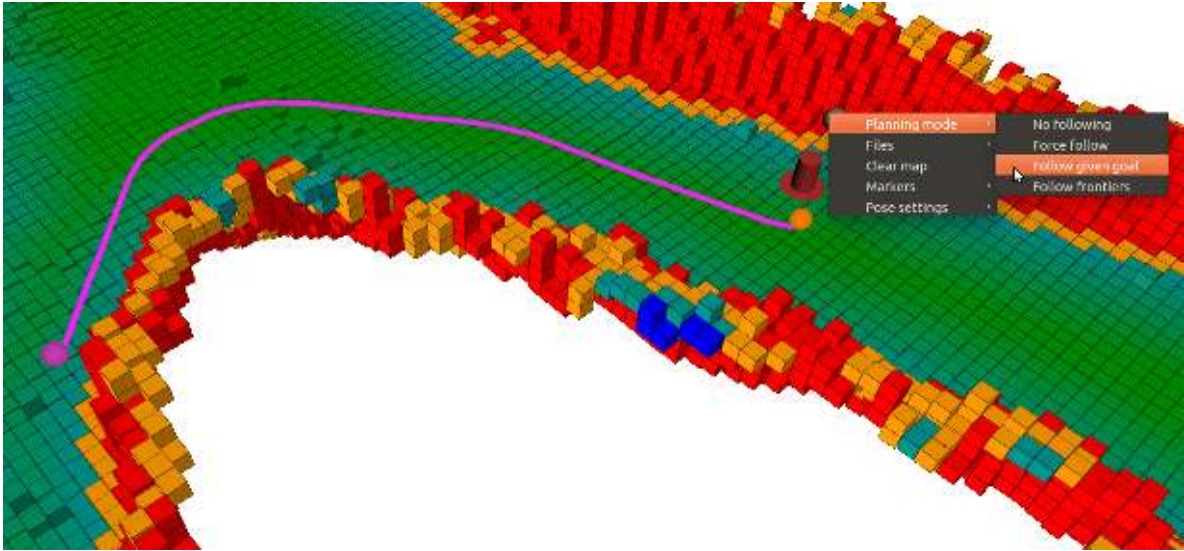


Figure 21: Example of the interactive interface, where the robot is forced to follow a given goal location.

## 4.4 Path following module

We have decoupled the path generation and path execution to increase the modularity of the system. Hence, the exploration module generates a collision-free path for the robot, and the responsibility for the path execution lies on the path following module. More specifically, the task of the current module is to generate control commands for the robot, based on the given path and the robot position.

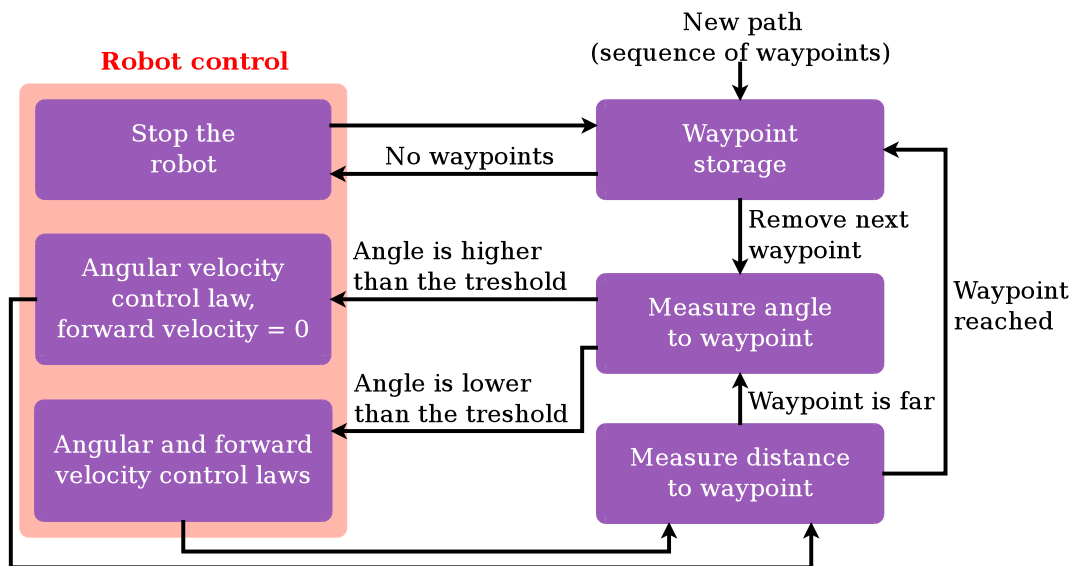


Figure 22: The architecture of the path following module.

The input to the path following module is the path in the form of a sequence of waypoints. The path following module pops the first waypoint from the sequence and commands the robot so that the robot moves towards the waypoint. If the robot reaches the waypoint, the tracker takes the next waypoint from the sequence, and the robot follows the new waypoint again. The procedure is repeated until there are some waypoints in the sequence. If there are no waypoints in the sequence, the robot stops. The execution of the path following module is captured in Figure 22.

#### 4.4 Path following module

The motion of the robot towards a waypoint is ensured by two proportional control laws; the first control law sets the angular velocity of the robot based on the angular displacement. If the angular displacement is higher than the threshold  $\alpha_t$  ( $\alpha_t = 15$  deg in our case), the forward velocity of the robot is set to zero. Otherwise the second control law sets the forward velocity based on position displacement of the robot.

In a real-world deployment, the localization might be noisy, or a localization system may provide pose of the robot with low frequency. Thus, we consider the waypoint as reached if the distance between the waypoint and the robot is less than the threshold  $\beta_{reach}$ . ( $\beta_{reach} = 2$  cm was used for hexapod walking robot.) Moreover, if the pose of the robot is provided with very low frequency, the robot might skip a waypoint without reaching it, which results in undesired behavior, because the robot will turn and try to reach the skipped waypoint. Therefore, we always remove waypoints that get behind the robot on its path.

## Chapter 5

# Results

The proposed exploration framework has been initially run without the localization system in simulations to test the exploration strategy and data handling within the framework. Beside the simulation results, we have also analyzed computational requirements of the proposed framework. Next, we experimentally evaluated the selected state-of-the-art localization technique ORB-SLAM2 and new embedded visual localization system the Intel RealSense T265 (T265 for short). Based on the evaluation results, the particular localization approach has been selected for experimental deployment of the framework in real-world indoor and outdoor autonomous exploration scenarios with the hexapod walking robot. Beside of the deployments on the hexapod walking robot, the proposed robotic framework has also been deployed on the tracked robot to show modularity of the proposed exploration framework. In particular, we reported on the deployments during the Subterranean Integration Exercise (STIX) of the DARPA Subterranean Challenge and indoor exploration.

### 5.1 Simulations

The main simulation tool used for the initial testing of the developed software implementations was the STDR Simulator [60]. Therefore, we provide a brief description of the simulator and simulation results for one of the exploration scenarios. Beside the 2D STDR Simulator, we have also used the Gazebo simulator [61] with the setup for a virtual track of the DARPA Subterranean Challenge to get more realistic 3D simulations of point cloud data incoming from the robot sensors, and expected behavior of the proposed solution in tunnels.



(a) 2D laser scan from the simulator.

(b) 3D point cloud based on the obtained 2D scan.

Figure 23: Change to the sensory data provided by the STDR Simulator. Note that we consider a certain height of obstacles (1 m) and ground plane at  $z = 0$ .

The STDR Simulator has been selected due to its simplicity and also because it is publicly available<sup>5</sup>. The robot motion in the simulator is taking place in the XY plane, while the localization of the robot is provided in 6 DoF, thus the localization provided by the simulator has replaced the localization module of the proposed robotic framework. We have simulated single robot equipped with 2D

<sup>5</sup> <http://stdr-simulator-ros-pkg.github.io/>

## 5.1 Simulations

360° LIDAR with range 4 m, which provided the laser scans at frequency of 5 Hz. Since the proposed mapping technique is designed for 3D point clouds, we had to alter the scans incoming from the simulated LIDAR so that they cover ground, not only obstacles. The change to the 2D data is shown in Figure 23.

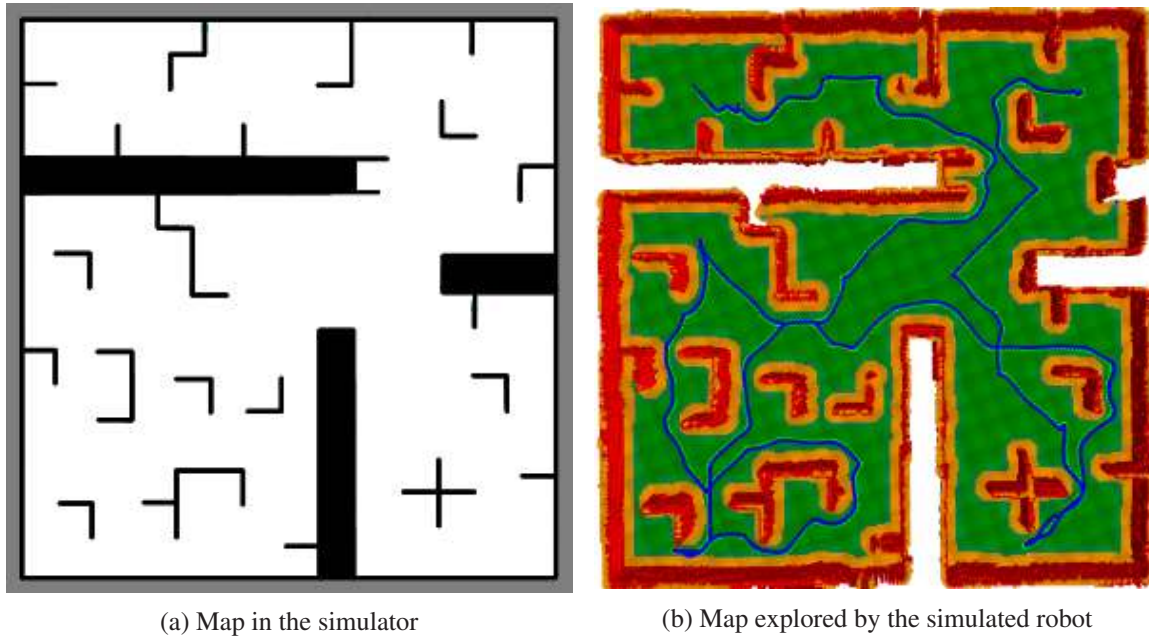


Figure 24: Map explored during the simulation run of the proposed exploration framework. The path traveled by the robot is visualized by the blue line.

The utilized environment map in the simulation and the resulting map built by the proposed autonomous exploration are depicted in Figure 24. The presented results indicate that the proposed framework is capable of the exploration, under the assumptions that the range data are not very noisy, the localization system is precise and reliable, and the robot moves smoothly. Most of these preconditions usually not held in the real-world deployment of the walking robot. Nevertheless, we deployed the framework on the real hexapod walking robot that is described in Section 5.3 and experimentally validated if similar results obtained in the simulation can be achieved also in real indoor and outdoor scenarios. However, we firstly examine the computational requirements of the individual exploration processes and precision of the considered localization methods that are reported in the following sections.



## ■ 5.2 Evaluation of computational requirements

The computational requirements of the individual processes (modules) of the developed exploration framework have been evaluated to verify feasibility of the proposed solution and possible deployment of the framework onboard of the small hexapod walking robot. The computational requirements are expressed by the CPU utilization of the used Intel i5 3320M processor clocked at 2.6 GHz accompanied with 8 GB RAM that corresponds to the computational power that can be onboard of the robot. The same computational environment has been used for the exploration deployments described in Section 5.5. Note that the computational requirements depend on many parameters, especially on the size of an input point cloud, the map resolution, and size of the cached area. Therefore, the average utilization of the used processor reported in Table 4 provides only an overview of the real computational requirements in a particular exploration setup.

Table 4: Computational requirements of exploration processes

Exploration Process (Module)	CPU usage	Update rate
Localization by ORB-SLAM2 [7]	208.0%	11 Hz
Elevation map building	56.0%	5 Hz
Exploration – Determination of navigational goal and path planning	16.8%	8 s
Path Following and Locomotion control	2%	–

The maximal CPU usage is 400% because of two cores with Hyper-threading

The results in Table 4 are for the elevation map with the resolution 75 mm and cached area  $20 \times 20$  m. The input sensor was the Intel RealSense D435, which usually provides 70000-280000 points per a point cloud. For a point cloud with more than 150000 points, the update rate is usually lower than 10 Hz. But for smaller input point clouds with thousands of points and cache of the size  $100 \times 100$  cells, it is possible to build the elevation map including the traversability assessment, growing untraversable places, frontier detection, and cost map generation with the update rate almost 100 Hz. Notice that in the case of T265 as the localization system, the update rate of the localization is 200 Hz, and ORB-SLAM2 is not needed.

## ■ 5.3 Description of the hexapod walking robot

In this section, we describe the utilized hexapod walking robot, to provide a complete description of the setup used during the real-world exploration scenarios. The mechanics of the robot is based on the off-the-shelf hexapod robot Phantom X Mark II, see Figure 25. The robot platform consists of the robot body, and six legs. Each leg of the robot is composed of three Dynamixel servo motors AX-12A, which enables to set a position of the endpoint of the leg at three degrees of freedom (DoF). The robot weights approximately 2.5 kg, and its actuators enable the robot to carry about 1 kg of additional load. During the experiments, the total weight of the robot, including the developed sensor rig, was approximately 2.75 kg. The robot is powered by a Li-pol battery pack (3s, 4 Ah), which provides enough energy to power the robot for 90 minutes. The distance that the robot is capable to traverse depends on the terrain type.

The hexapod walking robot can be controlled from a small onboard computer Odroid-XU4 or by an external computer. During the experimental evaluation of the localization systems, we have controlled the robot via the embedded platform, since it is handy for teleoperation. On the other hand, we have used the connection to the external computer during the exploration experiments, due to the poor compatibility of camera drivers (for T265 and D435) with the utilized Odroid-XU4 platform. The already existing robot interface implemented as a ROS module has been used to control the locomotion of the robot based on the velocity commands provided by the developed path following module.



Figure 25: Hexapod walking robot Phantom X Mark II utilized in the exploration scenarios.

### Motion gait

There are principally two main ways how control the locomotion of the hexapod walking. The first way is to use a pattern of leg motion (motion gait), and the second way is to use a precise motion planning for all the leg endpoints. The motion gait uses different motion generation as a motion pattern and one of the option is to generate a motion pattern by defining curves for the endpoints of the robot legs and then compute the inverse kinematics to set the positions of the individual actuators [62].

If there is no sensory feedback, the motion gait blindly steers the robot according to the velocity commands. Even without any sensory feedback, the robot is able to traverse uneven, but not very rough terrains. On the other hand, a motion gait supported by the sensory feedback allows the walking robot to traverse much rougher terrains. In case of the used walking robot, the basic sensory feedback is provided by utilized Dynamixel servo motors, which have in-built sensors of the angular rotation that can be utilized to a minimalistic approach for detecting the contact of the leg with the ground based on the angular error of the servo motor [63]. Also, a noteworthy fact about the mechanical construction of the hexapod walking robot is that the servo motors consume the most significant amount of energy to lift the robot body. Thus, the amount of energy consumed when the robot stands with the raised body and walks is nearly similar. Hence, a faster locomotion of the hexapod robot yields to lower amount of the energy consumed per traveled distance.

If the terrain is extremely rough with very few possible footholds usable by the robot, it is necessary to precisely plan the motion of the robot body and all the endpoints of the robot legs. Since, the configuration space of the robot has 24 DoF (6 DoF for robot body and 18 DoF for the robot legs), such a planning is computationally very demanding and precise information of the robot surroundings is necessary [64], [65]. On the other hand, the particular terrains considered for the experimental deployment are traversable by the robot [63]. Thus, we used the motion gait locomotion control because it is less computationally intensive than precise planning [64].

#### ■ 5.3.1 Sensors

Mapping, and also visual localization can use measurements from various sensors, and thus we discuss possible off-the-shelf sensors in this section. The vision-based localization ORB-SLAM2 supports three different types of sensors: monocular camera, stereo camera, and RGB-D camera. The usage of monocular cameras is problematic due to the initialization of the map scale, which has to be estimated based on known proportions of some object in the scene, or it needs to be initialized based on observed objects at different distances. Moreover, the scale of the scene in a new camera frame is based on distances between the previously seen features and translation motion of the robot towards the features; hence, the scale drifts. The highest drift of the scale is observed if the robot turns with a radius close to the zero, which the hexapod walking robot is capable of and the path following module is designed



Figure 26: Sensors considered for vision-based localization.

to use this feature to quickly clear out unexplored areas that are not in camera field of view. Here, it is worth of noting that a high drift induced by a near zero turn radius is one of the main issues of deploying vision-based localization methods designed for cars on hexapod walking robots [34]. On the other hand, stereo and RGB-D cameras can estimate the depth of the scene from a single frame, and therefore, they are more suitable for ORB-SLAM2 deployed on the hexapod walking robot.

### Stereo camera

The drift of the scale induced by a single camera can be avoided by using stereo camera, which is composed of two cameras with the parallel optical axes at the defined distance called the baseline. The distance of a feature detected by both cameras is then calculated using triangulation. It is obvious that the triangulation is more accurate if the image is taken from both cameras at the same time, i.e., the shutters of both cameras are synchronized. The estimate of the depth of the scene does not depend only on the synchronization of the shutters, but also on a type of the camera shutters. The most suitable shutter type for mobile robotics is the global shutter that enables the camera to capture the whole scene at the same time. An example stereo camera which has synchronized sensors MT9V024 with the global shutter is E-con systems Tara [66] that is shown in Figure 26a. Tara is lightweight, runs at 60 FPS, and has built-in IMU, but its resolution is only  $752 \times 480$  pixels.

### RGB-D camera

The RGB-D camera provides standard an RGB color image and depth image with the distances from the camera center to points corresponding to the pixels of the color image. The advantage of the RGB-D camera against the ordinary stereo camera is that a depth of the scene is directly provided by the camera and it is not necessary to triangulate each image feature from a pair of images, which is slower to compute. Therefore, the ORB-SLAM2 can process data from RGB-D camera faster than data from an ordinary stereo camera, which makes the RGB-D cameras more suitable sensors for ORB-SLAM2. Different RGB-D cameras are available and a brief description of three of them (see Table 5) follows.

One of the RGB-D cameras used in mobile robotics is Microsoft Kinect V2 [67]. Although the resolution of the Microsoft Kinect V2 is relatively high; it is not suitable sensor for the used hexapod walking robot because of the weight and its dimensions. The Asus Xtion Pro has already been suc-

### 5.3 Intel RealSense D435

Table 5: Parameters of considered RGB-D sensors

RGB-D camera	Asus Xtion Pro	Microsoft Kinect V2	Intel RealSense D435
Max resolution [px]	640 × 480	1920 × 1080	1280 × 720
Framerate [Hz]	30	30	30
Max distance of use [m]	3.5	4.5	10
Size [mm] (H/W/D)	50 × 180 × 35	66 × 249 × 67	25 × 90 × 25
Weight [g]	170	1400	72

cessfully used for visual localization of the hexapod walking robot [35], but its resolution is lower than the resolution of the Intel RealSense D435 [68] (further denoted D435). Moreover, D435 is able to provide not just a color and depth image for the visual localization system, but it also provides a point cloud of the scene, which makes the preprocessing of the mapping faster. Therefore D435 has been chosen for the visual localization and mapping within the proposed robotic framework. Parameters of discussed RGB-D cameras are summarized in Table 5.

**Intel RealSense D435** D435 consists of two IR cameras (both with the global shutter), IR projector, and one RGB camera with the rolling shutter. The absence of the global shutter on the RGB camera induces a geometrical distortion of the image, especially when the robot turns fast, which motivate to use D435 as an IR stereo camera instead of the RGB-D. Under good light conditions, D435 can provide depth image computed from the IR stereo pair while having the IR projector switched off. However, if the light conditions are not perfect, e.g., indoors, the IR projector has to be used to emit a special pattern of the infrared light. The pattern is reflected from objects in front of the camera, and depth of the scene can be estimated even in the environment with low sensing conditions. The IR cameras capture the reflected pattern and distances are computed from the comparison of the reflected pattern and the reference pattern<sup>6</sup>. D435 further supports synchronization with more D435 cameras on a hardware level using special synchronization cable.



Figure 27: Hexapod walking robot with developed sensor rig designed for the exploration.

<sup>6</sup>If the IR projector is switched on, D435 cannot be used as a stereo camera, because the IR pattern is visible in the images, and it would affect the localization system.

### Sensor rig

Since the cameras of T265 have IR filters able to eliminate the influence of IR light emitted by D435, it is possible to use both cameras at the same time. Thus, we have designed a special sensor rig for the hexapod walking robot to hold both cameras together with additional LED light (with the power of 18 W). The sensor rig itself is shown in Figure 27, and a view from the sensor rig is in Figure 28.



(a) T265: left image



(b) T265: right image



(c) D435: RGB image



(d) D435: depth image

Figure 28: View from the sensor rig mounted to the hexapod walking robot.

## 5.4 Experimental evaluation of the localization systems

Prior the deployment of the proposed exploration framework in real-world scenarios, the accuracy and reliability of the localization system based on the ORB-SLAM2 with Intel RealSense D435, and the precision of the localization provided by the Intel RealSense T265 were experimentally evaluated. Since the exploration framework is designed for rough terrains, we have also considered rough terrains during the examination of the precision of the evaluated localization systems. The experimental evaluation took place in an indoor laboratory environment that is shown in Figure 30b. In the experimental deployment, the robot walked 20 closed trails on a flat terrain, partially covered by a rough artificial terrain build from the irregularly shaped wooden blocks with different heights, see Figure 30b. During the experiments, the robot has been teleoperated and walked the trails ten times (trial 1), five times (trial 2), and five times (trial 3).

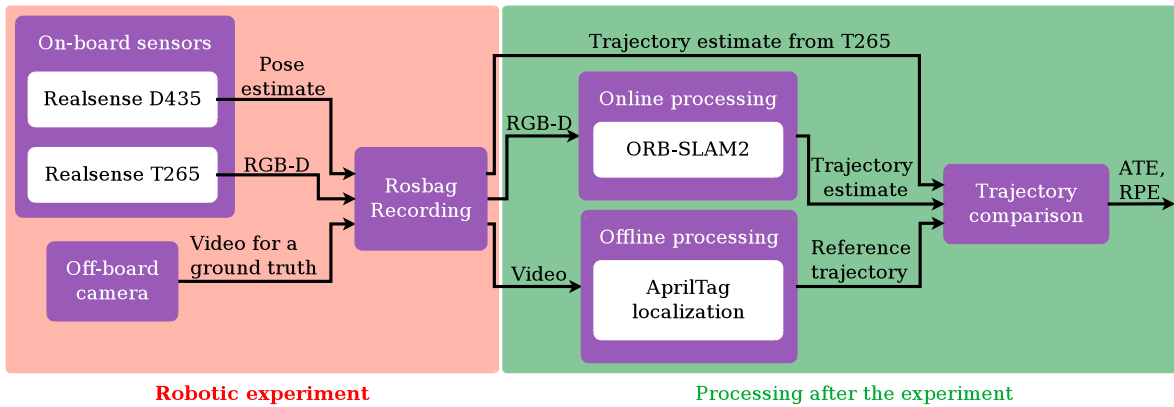


Figure 29: Schema of the evaluation process.

Both sensors (T265 and D435) were attached to the robot using the sensor rig shown in Figure 30a. Since ROS is already used as middleware in the proposed system, the rosbag tool has been utilized to collect all data from the onboard sensors synchronously together with the measurements for an off-board reference localization system [26]. The RGB-D images from D435 were captured at the frequency 15 Hz with the resolution  $1280 \times 720$  px. The stored images can be read by the rosbag frame-by-frame, or processed in real-time for the same frequency as they were captured. The online deployment was simulated by feeding the ORB-SLAM2 by RGB-D images at the same frequency as they were captured. After all the data were processed, the trajectory estimates provided by T265, by ORB-SLAM2, and the corresponding reference trajectories were evaluated using metrics described in Section 3.1, i.e., absolute trajectory error (ATE) and relative pose error (RPE). The data flow during the evaluation of both localization systems is visualized in Figure 29.

The **reference localization system** is based on AprilTag [26] that is an external localization system able to provide full 6 DoF position of the robot. AprilTag localizes a distinctive pattern attached to the robot with centimeter precision, which is shown in Figure 30a. The robot with the attached pattern has been observed from the top camera with the resolution  $1920 \times 1080$  and framerate 20 FPS. The images from the top camera obtained during the experiment were processed frame-by-frame to get as detailed ground truth as possible.

### Evaluation

The accuracy of the localization systems is evaluated using the standard metrics of ATE and RPE [53]. Besides the accuracy, the reliability of the localization systems was evaluated by the following principle. We have counted all the cases, when the localization system failed to track the robot for more than two seconds as a *track lost*, which not necessarily leads to complete failure of the localization,

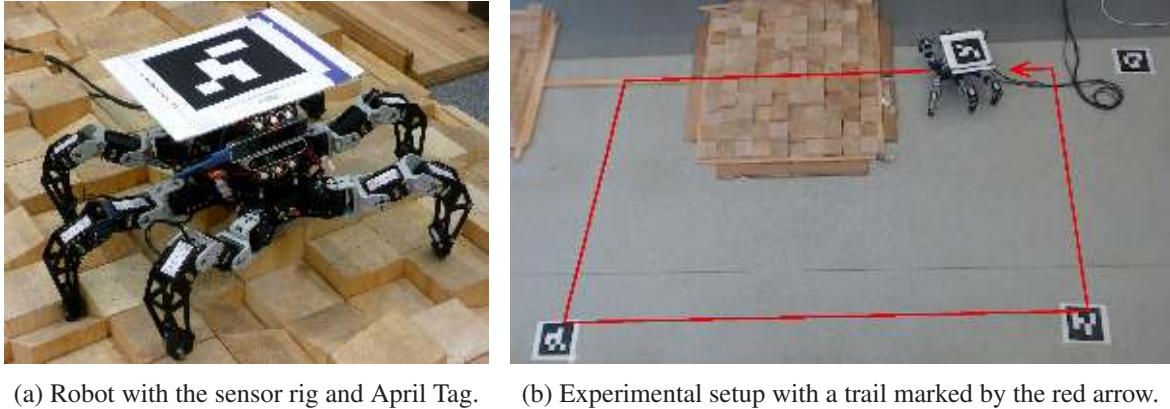


Figure 30: Experimental setup for the evaluation of the localization systems.

e.g., ORB-SLAM2 can deploy its module for relocalization. However, if the localization system was unable to cover more than 75% of the trail, we count it as *loc. failed*.

During each trial, several trails were performed sequentially without a reset of the localization systems. Thus the localization systems were enabled to use loop closures to improve the precision of the localization. After each trial, both localization systems under the test were reset. The results for all the trials of the experimental evaluation are listed in Table 6.

Table 6: Precision of the Evaluated Localization Systems

Localization system	ORB-SLAM2				Intel RealSense T265			
	$\overline{ATE}_t$ [cm]	$\overline{RPE}_t$ [cm]	<i>track lost</i> [-]	<i>loc. failed</i> [-]	$\overline{ATE}_t$ [cm]	$\overline{RPE}_t$ [cm]	<i>track lost</i> [-]	<i>loc. failed</i> [-]
Trial 1 (10 trails)	10.15	1.32	19	1	8.31	0.49	0	0
Trial 2 (5 trails)	10.91	1.49	8	2	10.27	0.55	0	0
Trial 3 (5 trails)	12.45	1.29	10	1	10.92	0.55	0	0

The results summarized in Table 6 show that both localization systems provide competitive precision and can be employed in the exploration. Note, that a lower RPE of T265 is probably induced by the fusion with IMU. The numbers of *track lost* and *loc. failed* indicate that ORB-SLAM2 is less reliable than the embedded localization system of T265, which always provide some estimate of the robot position. Most of the fails of ORB-SLAM2 have been produced on the rough terrain which induced fast changes of the robot gaze. The relatively quick changes of the robot gaze were also provided by turns of the robot at the corners of the robot path. Another fact which affects the performance of ORB-SLAM2 is a low processing frequency of the incoming frames, which rarely reached 11 Hz. The computer with Intel i5 3320M CPU clocked at 2.6 GHz, and 8 GB RAM has been used to run the ORB-SLAM2, which roughly correspond to the computational power that can be available onboard of the hexapod walking robot. Based on the presented experimental evaluation, we state that a more suitable localization system for the proposed exploration framework is T265.

## ■ 5.5 Experimental results with hexapod walking robot

Based on the evaluation of the localization systems, the Intel RealSense T265 has been selected as the localization system for the experimental deployment of the developed exploration framework. A description of the deployment of the proposed framework in fully autonomous exploration scenarios on the hexapod walking robot follows.

Table 7: Performance indicators for the exploration scenarios with hexapod walking robot

Exploration scenario	Indoor corridor	Chimney location	Yard location
Map resolution [mm]	75	75	75
Traversability threshold $g_{max}$ [mm]	80	80	120
Max sensor range [m]	2.5	3	3
Exploration time	28 min 15 s	40 min 9 s	44 min 16 s
Traversed distance [m]	49.093	55.507	45.958

The exploration framework has been deployed in fully autonomous exploration missions in three scenarios: an indoor corridor, outdoor terrain with a concrete surface, and rough grassy terrain; all located at the campus on Karlovo náměstí. Since the experimental deployment does not allow to utilize the reference localization system or spatial reference model of the terrain, the exploration performance is measured by the exploration time, traveled distance, and visual evaluation of the created map of the environment. The achieved performance indicators of the experimental scenarios are summarized in Table 7 and individual maps are shown in Figure 31, Figure 32, and Figure 34. For the comparison purposes; the average speed of the teleoperated hexapod walking robot in the setup similar to the indoor corridor shown in Figure 31a is approximately  $0.1 \text{ ms}^{-1}$ .



### 5.5.1 Indoor exploration

In the indoor scenario, we put the localization system on a stress by the shortened range of the RGB-D camera to 2.5 meters, so the sensor cannot capture both walls of the corridor from a single spot. Therefore, the robot frequently changed the gaze direction to see the walls on both sides of the corridor, see Figure 31d.

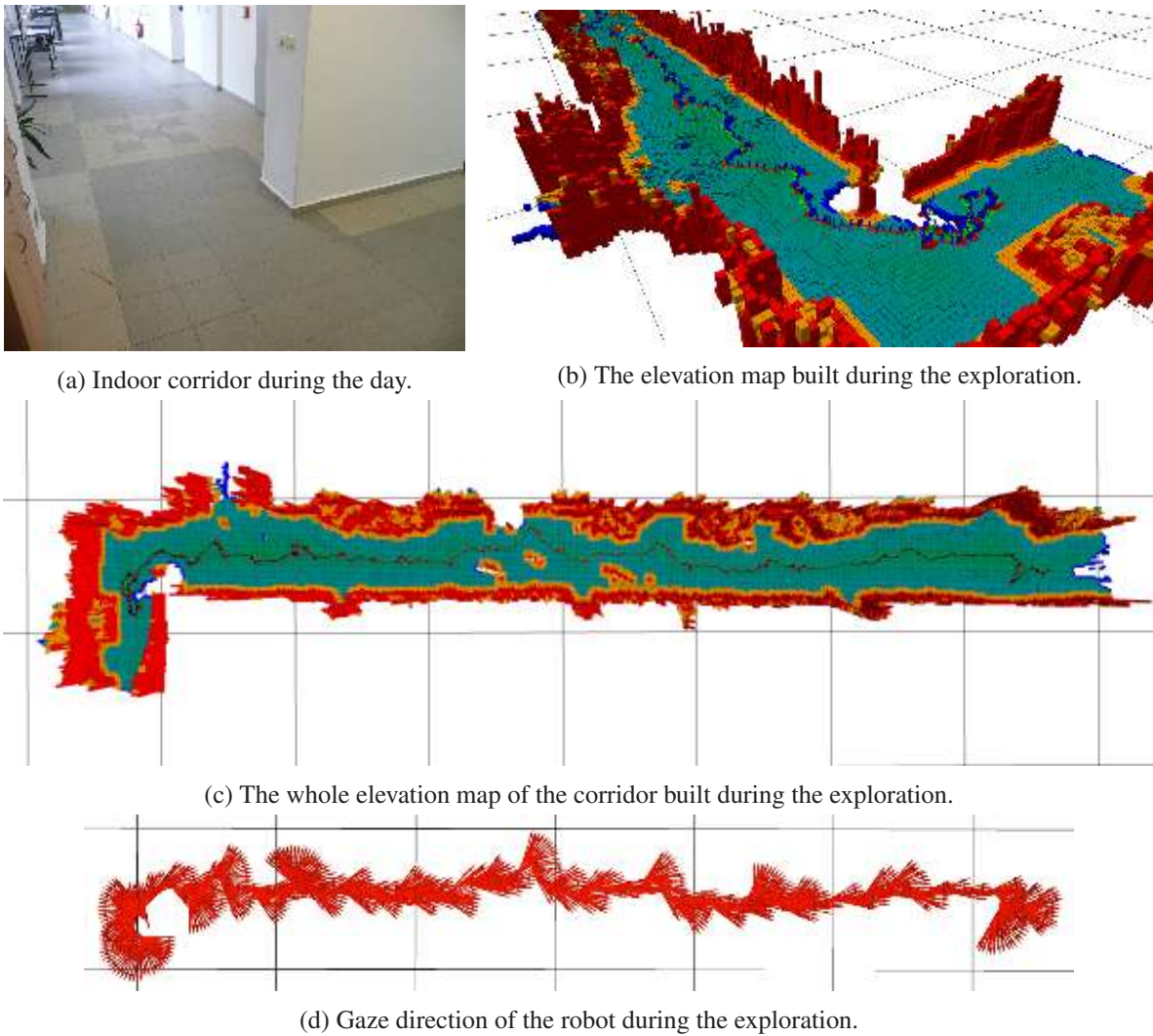


Figure 31: Indoor corridor, size of the grid is 5 m.

The small untraversable sets of cells can be seen near one side of the corridor in Figure 31c. These cells were affected by the measurements biased by the light incoming from the opposite side of the corridor. Thus, these untraversable cells do not correspond to any objects as it can be seen from Figure 31a.

## 5.5 Outdoor exploration

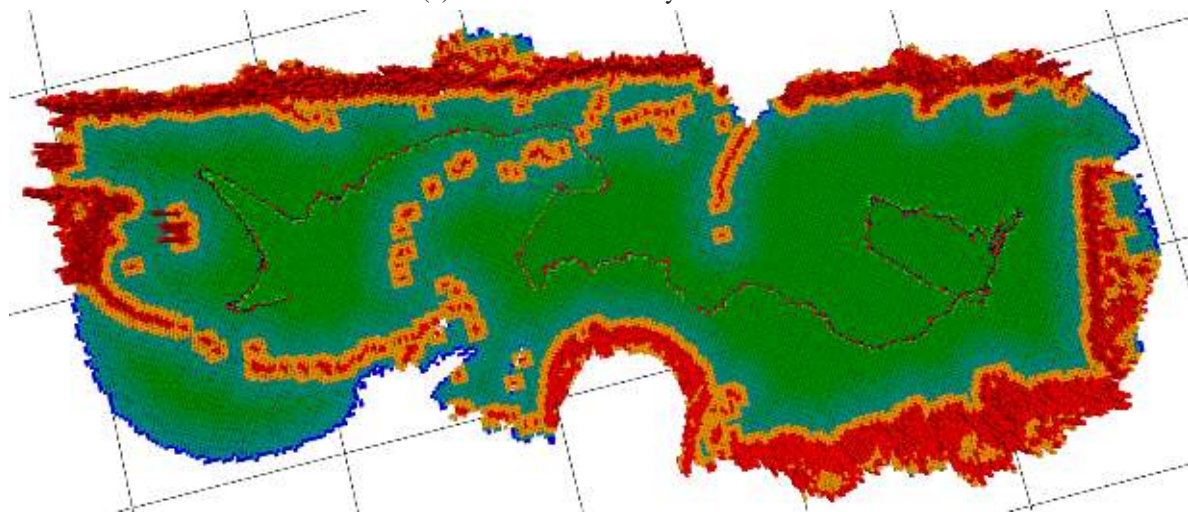
### 5.5.2 Outdoor exploration

#### Chimney location

The next experimental deployment of the exploration framework took place on the concrete panels near a chimney. The concrete panels are mostly flat, but there are also large cracks in the panels, which can also be seen in the created elevation map shown in Figure 32. Some of these cracks induced significant misalignments of the concrete panels, so that these places were detected as untraversable, and successfully avoided by the robot during the execution, see Figure 32.



(a) View of the chimney location.



(b) The elevation map built during the exploration, size of the grid is 5 m.

Figure 32: Exploration of the chimney surroundings.

### Yard location

The final experimental testing of the exploration framework took place in the yard, where a wooden structure is placed in the middle. The uneven grass surface, plants moving in the wind, and net made exploration of this location very hard, see the view from the onboard sensor in Figure 33. A picture

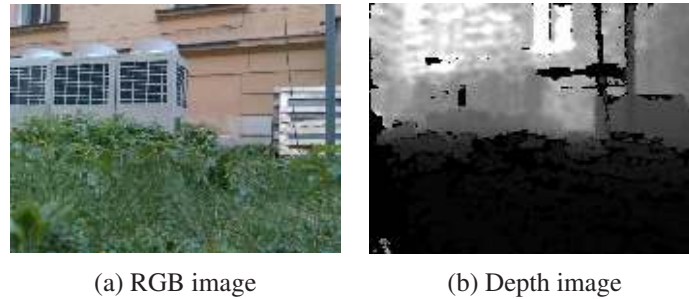


Figure 33: View from the Intel RealSense D435 during the experiment.

of the yard and maps obtained during the experiments are shown in Figure 34. The robot was able to explore more than half of the yard (see Figure 34a), but then the experiment had to be terminated due to weather conditions. In Figure 34c, we show the complete map of the yard that has been finished after the interruption of the autonomous exploration.

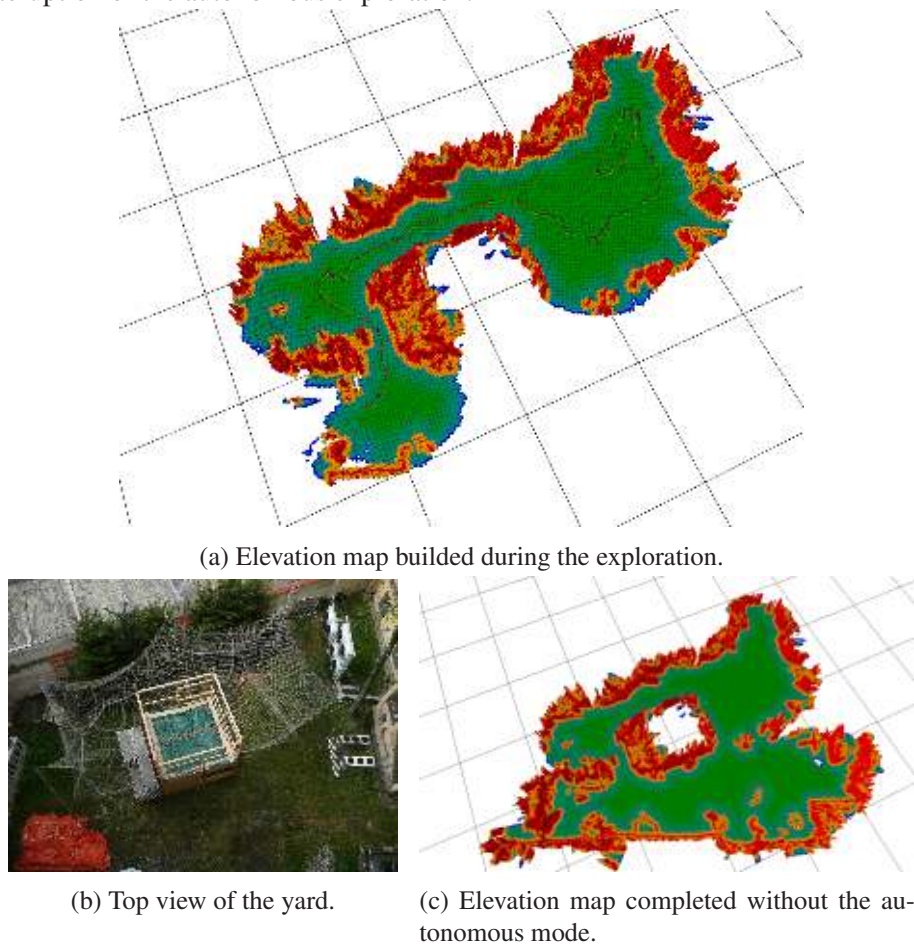


Figure 34: Exploration of the yard, size of the grid is 5 m.

### ■ 5.6 Experimental Deployment on Tracked Robot

In addition to the hexapod walking robot, we also considered the proposed framework and its exploration and mapping capabilities on a tracked robot to demonstrate modularity of the framework. The tracked robot has been deployed in challenging mine environment within DARPA Subterranean Challenge STIX event, where we have deployed mapping and planning techniques to provide an alternative to the teleoperation of the robots. Moreover, we have deployed the exploration and path following modules in multiple fully autonomous exploration scenarios with the same tracked robot. A brief description of the robot is provided in the next section that is followed by the report on the achieved results.

#### ■ 5.6.1 Description of the tracked robot

The considered tracked robot platform is shown in Figure 35. The robot has an onboard computer based on the Intel NUC with the i7 class processor, which has been used for all computations during the mapping and exploration experiments with the tracked robot. Various equipment has been mounted to the robot, but only LIDAR is directly utilized by the framework modules.



Figure 35: Tracked robot platform.

In particular, the tracked robot is equipped with the SICK LMS151-10100 laser range finder that provides 2D scans of the environment, but the servo motor periodically turns the device, so that the laser provides 3D scans of the robot surroundings. The disadvantage of such a setup is that the period of the LIDAR motion is approximately 2 s, which results in a very low frequency of the point clouds for elevation mapping. The localization of the tracked robot was also based on the point clouds from the LIDAR using the laser SLAM based on approach [69], which is not a part of this thesis. Hence, the localization module in the proposed exploration architecture was replaced by the SLAM using LIDAR scans, but the rest of the framework remained the same.

### ■ 5.6.2 Mapping mines during DARPA Subterranean Challenge

The proposed solution for the elevation mapping has been deployed during the Subterranean Integration Exercise (STIX) event of the DARPA Subterranean Challenge. In DARPA Subterranean Challenge, the robots search underground terrain to locate specified artifacts, e.g., survivors, backpacks, and fire extinguishers as quickly and as precise as possible. The assignment for the STIX event was to find the artifacts in the tunnels of the Edgar Mine at the Idaho Springs near Denver in Colorado. The classes of the found artifacts have to be reported with the absolute position of the artifact in the coordinate frame defined by the entrance gate to the mine, which is shown in Figure 36.



Figure 36: Tracked robot entering the experimental mine through the reference entry gate.

The rules of the STIX enables each team to have one human operator responsible for submitting detected objects to the DARPA server and controlling the robots. Thus, the operator can teleoperate the robots, but since the operator is also responsible for checking discovered artifacts, it was found impractical to rely solely on the operator. Moreover, the delay in the communication, which we have been experienced makes teleoperation very hard. Therefore the selected strategy to balance between pure teleoperation and fully autonomous run was to let the operator select goals for the robot via an interactive interface and let the exploration and path following modules of the developed exploration framework do the rest. An example of the created map from one of such trial is shown in Figure 37.

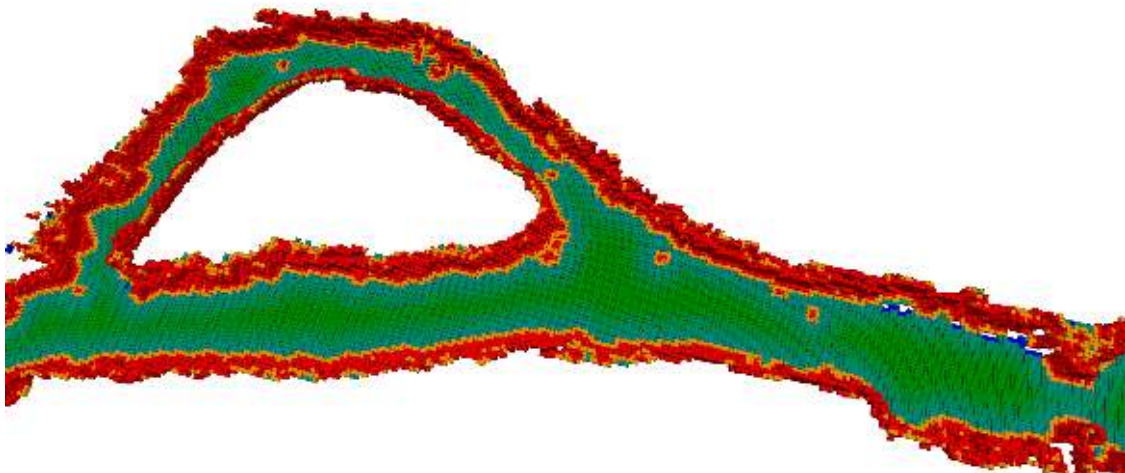


Figure 37: The elevation map of the entry part of the experimental mine (resolution of the map is 15 cm).

## 5.6 Indoor exploration scenarios

### 5.6.3 Indoor exploration scenarios

The developed exploration framework has been further deployed on the tracked robot in multiple indoor scenarios. The example of the created elevation map obtained in the fully autonomous exploration of a part of the campus basement is shown in Figure 38.

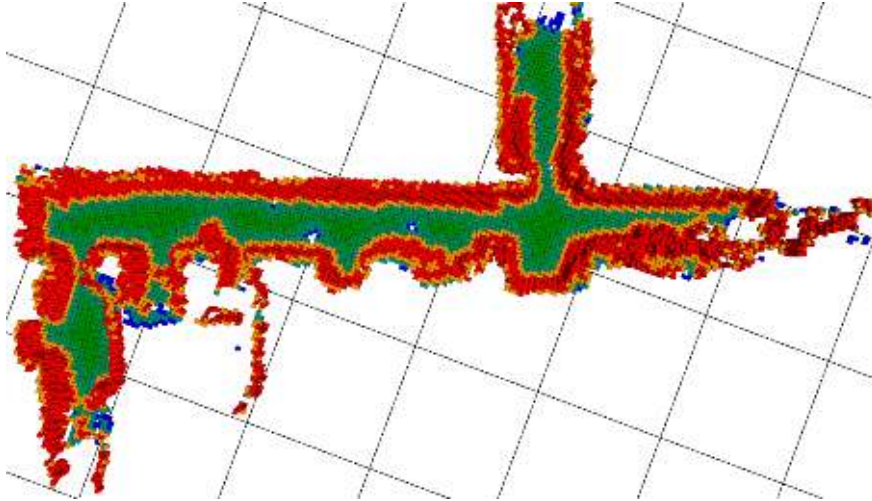


Figure 38: Created elevation map during the basement exploration scenario with the grid cell size 5 m.

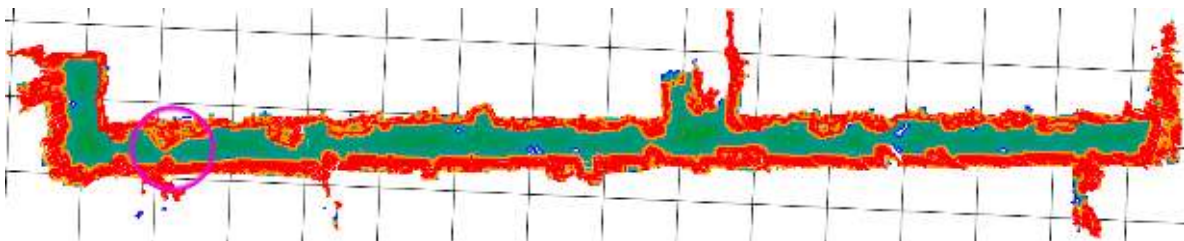



Figure 39: The created elevation map of the second floor obtained from exploration with marked narrowing by ping circle. The size of the underlying grid cell is 5 m.

An additional experiment took place in a corridor that is similar to one shown in Figure 31a, but this time, the corridor has been filled with different obstacles that have been successfully avoided by the robot during the exploration. In the left part of the map, there was narrowing, so that the passage was less than 90 cm wide; however the robot with the width about 60 cm was capable of successfully passing the narrowing without hitting the obstacles. The another issue of this particular exploration scenario was induced by the tiled floor which reduced the effective range of the LIDAR, so that the space farther than 3 m in front of the robot was not detected. The exploration of the corridor took approximately 15 min, and the explored map is shown in Figure 39.



## Chapter 6

# Conclusion

In this thesis, we have proposed a robotic framework for the spatial exploration of rough terrains. The proposed solution includes a careful selection of the localization system, where we have considered state-of-the-art vision-based localization system ORB-SLAM2, for which we have proposed sensory fusion with external localization system and IMU. The selection of the localization system has been completed by the experimental evaluation of the localization provided by ORB-SLAM2, and localization obtained from the novel embedded solution for the localization based on the Intel RealSense T265, which utilize sensory fusion and do all the necessary computations onboard. The experimental evaluation showed that both vision-based localization systems provides competitive accuracy, but the localization provided by the Intel RealSense T265 is more reliable. Therefore, we have used T265 for the experimental deployment of the developed framework in autonomous exploration with the hexapod walking robot.

The simulation results and the evaluation of computational requirements have shown the feasibility of the proposed exploration approach as well as deployments in real-world scenarios. The proposed exploration framework utilizing the proposed implementation of the elevation mapping has been deployed on the hexapod walking robot which was able to explore the indoor corridor and also outdoor scenarios, including rough grassy terrain. In addition, the developed framework has been further deployed on a tracked robot in mapping scenario of the mine environment within the DARPA Subterranean Challenge and in indoor corridors, which have been explored also fully autonomously. During the indoor exploration experiments with the tracked robot, all the necessary computations were run onboard.

Beside the reported results from various real-world deployments, the proposed framework has been further augmented by incremental learning framework for a spatial exploration combined with the traversal cost model learning that has been accepted for publication at the Robotics: Science and Systems (RSS) conference [70]. As the future work, we are going to extend the framework to build the elevation map from multiple sources of the measurements to support the multi-robot exploration.

## References

- [1] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006.
- [2] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.
- [3] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997.
- [4] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 487–494, 2015.
- [5] Dominik Belter, Przemyslaw Labecki, and Piotr Skrzypczynski. An exploration-based approach to terrain traversability assessment for a walking robot. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2013.
- [6] Max Schwarz, Marius Beul, David Droschel, Sebastian Schüller, Arul Selvam Periyasamy, Christian Lenz, Michael Schreiber, and Sven Behnke. Supervised Autonomy for Exploration and Mobile Manipulation in Rough Terrain with a Centaur-Like Robot. *Frontiers in Robotics and AI*, 3, 2016.
- [7] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [8] Jan Bayer and Jan Faigl. Localization fusion for aerial vehicles in partially gns denied environments. In *Modelling and Simulation for Autonomous Systems (MESAS)*, pages 251–262, 2019.
- [9] Collective of autors. Intel RealSense Depth Camera T265. <https://click.intel.com/order-intel-realsense-tracking-camera-t265.html>. accessed May 23, 2019.
- [10] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545 vol.1, Sep. 2002.
- [11] Jan Faigl and Miroslav Kulich. On benchmarking of frontier-based multi-robot exploration strategies. In *European Conference on Mobile Robots (ECMR)*, pages 1–8, 2015.
- [12] Sebastian Thrun. Robotic mapping: A survey. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, pages 1–35. Morgan Kaufmann Publishers Inc., 2003.
- [13] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.



- [14] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.
- [15] Anderson Souza and Luiz M. G. Gonçalves. Occupancy-elevation grid: an alternative approach for robotic mapping and navigation. *Robotica*, 34(11):2592—2609, 2016.
- [16] Patrick Pfaff, Rudolph Triebel, and Wolfram Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *The International Journal of Robotics Research*, 26:217–230, 02 2007.
- [17] P. Fankhauser, M. Bloesch, and M. Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.
- [18] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8:415–428, 2012.
- [19] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [20] M. Keidar and G. A. Kaminka. Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, 33(2):215–236, 2014.
- [21] Miroslav Kulich, Jan Faigl, and Libor Přeučil. On distance utility in the exploration task. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4455–4460, 2011.
- [22] H. M. Khoury and V. R. Kamat. Evaluation of position tracking technologies for user localization in indoor construction environments. *Automation in Construction*, 18(4):649–656, 2009.
- [23] Marvelmind robotics. Precise indoor gps. <https://marvelmind.com/>, accessed October 04, 2018.
- [24] Collective of authors. Vicon motion systems inc. <https://www.vicon.com/>, accessed 22 4, 2018.
- [25] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail. A Practical Multirobot Localization System. *Journal of Intelligent and Robotic Systems*, 76(3-4):539–562, 2014.
- [26] Edwin Olson. AprilTag: A Robust and Flexible Visual Fiducial System. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407, 2011.
- [27] Estelle Lubbe, Daniel Withey, and Kenneth R. Uren. State Estimation for a Hexapod Robot. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 6286–6291, 2015.
- [28] Moritz Menze and Andreas Geiger. Object Scene Flow for Autonomous Vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015.
- [29] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2d lidar slam. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.
- [30] Georg Klein and David Murray. Parallel Tracking and Mapping on a Camera Phone. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–86, 2009.

- [31] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-D Mapping with an RGB-D Camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.
- [32] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.
- [33] Jan Bayer. Autonomous localization of hexapod walking robot. Bachelor’s thesis, Czech Technical University in Prague, 2017.
- [34] Michał Nowicki, Dominik Belter, Aleksander Kostusiak, Petr Čížek, Jan Faigl, and Piotr Skrzypczynski. An Experimental Study on Feature-based SLAM for Multi-legged Robots with RGB-D sensors. *Industrial Robot: An International Journal*, 44(4):320–328, 2017.
- [35] Petr Čížek and Jan Faigl. On Localization and Mapping with RGB-D Sensor and Hexapod Walking Robot in Rough Terrains. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2273–2278, 2016.
- [36] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.
- [37] Edward Rosten and Tom Drummond. Machine Learning for High-speed Corner Detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443, 2006.
- [38] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision (ECCV)*, pages 778–792, 2010.
- [39] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.
- [40] Jan Bayer, Petr Čížek, and Jan Faigl. On Construction of a Reliable Ground Truth for Evaluation of Visual SLAM Algorithms. In *Acta Polytechnica CTU Proceedings*, volume 6, pages 1–5, 2016.
- [41] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.
- [42] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106, Nov 2013.
- [43] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision (ECCV)*, pages 834–849, 2014.
- [44] Raúl Mur-Artal, Jose Maria Martinez Montiel, and Juan D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.
- [45] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An Evaluation of the RGB-D SLAM System. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1691–1696, 2012.
- [46] Taihú Pire, Thomas Fischer, Javier Civera, Pablo De Cristóforis, and Julio Jacobo Berlles. Stereo Parallel Tracking and Mapping for Robot Localization. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, 2015.
- [47] Thomas Fischer, Taihú Pire, Petr Čížek, Pablo De Cristóforis, and Jan Faigl. Stereo Vision-based Localization for Hexapod Walking Robots Operating in Rough Terrains. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2492–2497, 2016.

- [48] Taihú Pire, Thomas Fischer, Javier Civera, Pablo De Cristóforis, and Julio Jacobo Berlles. Stereo parallel tracking and mapping for robot localization. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378, 2015.
- [49] J. Q. Cui, S. Lai, X. Dong, P. Liu, B. M. Chen, and T. H. Lee. Autonomous navigation of uav in forest. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 726–733, 2014.
- [50] Annett Stelzer, Heiko Hirschmüller, and Martin Görner. Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain. *The International Journal of Robotics Research*, 31(4):381–402, 2012.
- [51] A. M. Sabatini. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *IEEE Transactions on Biomedical Engineering*, 53(7):1346–1356, 2006.
- [52] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an Open-source Robot Operating System. In *IEEE International Conference on Robotics and Automation (ICRA): Workshop on Open Source Software*, 2009.
- [53] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.
- [54] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On Measuring the Accuracy of SLAM Algorithms. *Autonomous Robots*, 27(4):387–407, 2009.
- [55] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In *Lecture Notes in Computer Science*, volume 1883, pages 298–372, 2000.
- [56] Alireza Toloei and Saeid Niazi. State estimation for target tracking problems with nonlinear kalman filter algorithms. *International Journal of Computer Applications*, 98(17):30–36, 2014.
- [57] Kaiqiang Feng, Jie Li, Xiaoming Zhang, Chong Shen, Yu Bi, Tao Zheng, and Jun Liu. A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm. *Sensors*, 17(9), 2017.
- [58] H. Samet. Distance transform for images represented by quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(3):298–303, 1982.
- [59] J. Faigl and M. Kulich. On determination of goal candidates in frontier-based multi-robot exploration. In *European Conference on Mobile Robots (ECMR)*, pages 210–215, 2013.
- [60] Collective of authors. Stdr simulator. [http://wiki.ros.org/stdr\\_simulator](http://wiki.ros.org/stdr_simulator), accessed May 20, 2019.
- [61] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2149–2154, 2004.
- [62] J. Mrva and J. Faigl. Tactile sensing with servo drives feedback only for blind hexapod walking robot. In *10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 240–245, 2015.

- [63] Jan Faigl and Petr Čížek. Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only. *Robotics and Autonomous Systems*, 116:136–147, 2019.
- [64] P. Čížek, D. Masri, and J. Faigl. Foothold placement planning with a hexapod crawling robot. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 4096–4101, 2017.
- [65] Dominik Belter, Przemysław Łabecki, and Piotr Skrzypczyński. Adaptive Motion Planning for Autonomous Rough Terrain Traversal with a Walking Robot. *Journal of Field Robotics*, 33(3):337–370, 2015.
- [66] Collective of autors. Tara - USB 3.0 Stereo Vision Camera. <https://www.e-consystems.com/3D-USB-stereo-camera.asp>. accessed May 20, 2017.
- [67] Oliver Wasenmüller and Didier Stricker. Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In Chu-Song Chen, Jiwen Lu, and Kai-Kuang Ma, editors, *ACCV Workshop on Computer Vision*, pages 34–45, 2017.
- [68] Collective of autors. Intel RealSense Depth Camera D435. <https://click.intel.com/intelr-realsensetm-depth-camera-d435.html>. accessed August 4, 2018.
- [69] R. Dubé, A. Gawel, C. Cadena, R. Siegwart, L. Freda, and M. Gianni. 3d localization, mapping and path planning for search and rescue operations. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 272–273, 2016.
- [70] Miloš Prágr, Petr Čížek, Jan Bayer, and Jan Faigl. Online incremental learning of the terrain traversal cost in autonomous exploration. In *Robotics: Science and Systems (RSS)*, 2019. accepted, to appear.