Perception Systems for Autonomous Mobile Robots: Selecting and Mitigating Limits

Konrad Cop [1, 2], Morteza Haghbeigi [1, 2], Marcin Gajewski [2], Tomasz Trzciński [1]
[1] Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warsaw, Poland
[2] United Robots Sp. z o.o., ul. Świeradowska 47, 02-622 Warszawa

Abstract: A well-designed perception system is crucial for the proper operation of an autonomous mobile robot however it is not trivial to engineer. For a comprehensive system not only the sensors' features must be considered but also the way their data are utilised for robotic operation. In this paper, we present a set of tightly coupled strategies that allow the creation of an end-to-end perception system utilising 3D representation in the form of point clouds. Our proposal is generic enough to be applied to various mobile robots as it is aware of the context of robotic operation and accounts for hardware constraints. As the first strategy, we introduce a formalised sensors selection process formulated as a multi-objective optimization problem with metrics relaxation which allows to choose an optimal set of sensors within budgetary limitations. Secondly, we validate various data filtration strategies and their combination in the context of the navigation system to find a trade-off between accuracy and computational effort. Finally, to mitigate the suboptimal field of view of combined sensors and augment the perception system we introduce a new concept for the occupancy grid layer which utilises motion information in the occupancy calculation. For these strategies, we conduct experimental verification and apply the results in an autonomous cleaning robot.

Keywords: Robotic Perception, Autonomous Robots, Robotic Sensors, Robotics, Navigation, Data Filtration

1. Introduction

One of the key enablers of robotic autonomy is the robot's ability to perceive its surroundings. Observation of the environment is realised by means of a multitude of onboard sensors combined into a perception system. In the context of mobile robots, what the robot perceives is directly propagated to the navigation system. Sensors inputs are utilised to estimate achievable space in the environment to plan actions (trajectories) of the robot in short- and long-time perspectives. Therefore, the ability to navigate efficiently highly depends on the sensors' choice and how they are deployed. In this paper, we formulate three key aspects that allow the creation of a successful end-to-end perception solution. Our proposal focuses solely on the 3D data in the format of point clouds which are commonly used in robotic perception systems due to their effectiveness in representing complex environments.

The first aspect which must be considered is that the choice of the sensors that would satisfy multiple, often contradicting requirements is not trivial. Depending on the mechanical design of the robot, characteristics of the robotic process, features of the environment, and budgetary limitations, various combinations of sensors can be selected. So far the selection process has usually been considered case-specific, for example, Shi et al. designed a perception system for a humanoid robot [1] while Deshpande et al. tackled a designated mobile platform [2] but no generic approach was proposed. Others proposed to compare sensors solely such as different LiDAR solutions [3] or RGBD cameras [4, 5] without deeply considering the application. Instead of going this path, we propose to formulate the selection process as an iterative procedure that takes into account multiple "nice-to-have" requirements of the application and optimizes the costs of the solution. The process is formally defined and was used already for selecting a sensor rig for an autonomous cleaning robot.

Secondly, each sensor is a piece of hardware that has some production irregularities and detection inaccuracies, which lead to more or less noisy data. Using such data would cause the robot to experience unstable operation. Therefore, proper filtration mechanisms are required to improve the behaviour of the robot. Additionally, depending on the sensor, the incoming data might be very dense or contain redundant information,

Autor korespondujący:
Konrad Cop, konrad.cop@unitedrobots.co

Artykuł recenzowany
nadesłany 08.10.2024 r., przyjęty do druku 16.01.2025 r.

which potentially leads to unnecessary utilisation of computational resources when using such information for navigation. Multiple works were conducted in this regard to propose filtration strategies for purposes of object reconstruction [6, 7] or 3D video streaming [8] and this is a well-explored field. However, in the context of mobile robots with limited onboard resources, it is not only important how well the filtration procedure performs but also how efficient it is. Therefore in this paper, we evaluate the influence of various filtration methods and their combination on the process efficiency.

Finally, as most robotic projects usually face budgetary limitations, purchasing enough sensors to satisfy all requirements might not be feasible. As a result, some criteria must be sacrificed. Based on practical experience, a common consequence of this limitation is the incomplete spatial coverage provided by the selected sensors, which fails to fully capture the robot's entire surroundings. This, in turn, makes navigation particularly challenging, as the reduced sensor coverage compromises the safety and reliability of the robot's movement through its environment. With inadequate hardware, one could utilise the software to fill in the gap. Few approaches have been proposed to address similar challenges, but they primarily focus on developing local planners or controllers that calculate collision-free trajectories based on limited sensing capabilities, such as maximum sensor range [9, 10]. These methods often assume a 2D field of view, where obstacles within this plane are guaranteed to be detected. However, this assumption does not hold in 3D environments, where detecting blind spots is far more complex. For instance, Phan et al. introduced a decision-making module for robots with limited sensing capabilities, which enables collision avoidance based on partial detections in 2D [11]. Despite these advancements, none of these approaches have fully explored the use of spatio-temporal information for generating navigation occupancy maps, which could significantly enhance the reliability of autonomous navigation. We therefore, propose to utilise the information from both the sensor's returns and the robot's motion to accumulate the relevant information by means of a Hit Layer.

The three aforementioned strategies are interrelated and essential when creating a perception system for a mobile robot in an end-to-end manner. The combined analysis constitutes the contribution of this paper which includes three aspects:

- We propose an iterative, generic process for selecting sensors for any mobile robot.
- We evaluate combinations of various filtration algorithms in terms of their computational efficiency.
- We introduce an approach for utilising perception and motion information into occupancy estimation by means of a newly proposed Hit Layer.

The following sections describe the contributions in details.

2. Selection of sensors

Selecting an appropriate sensor rig for mobile navigation is not a trivial task and it depends on multiple factors related to both robot's mechanical features (e.g. dimensions and inertia) and application specifics, which may lead to contradicting requirements. For example, one might require high resolution of data and simultaneously low hardware costs. Better-quality sensors are typically more expensive. In such a situation, a question arises of which parameters should be sacrificed. To facilitate the decision, we introduce a selection process that takes into account multiple aspects and formulate it as a multi-objective optimization problem with metrics relaxation. The overview of the process is depicted in Fig. 1. The base of our proposal is the definition of eight *Application Requirements* and six Sensor Features which are the starting point for the selection process. Careful analysis of these two groups of parameters allows us to formulate *Rig Selection Metrics* which are quantitative expressions of the system requirements. The metrics are split into *Satisficing Metrics* which aim to define boundary selection conditions for the sensors and the *Optimizing Metric* which allows finding the best constrained solution.

The selection process should be conducted as follows. At the start, one should define the set of most desirable parameters of the complete rig i.e. the most strict *Satisficing Metrics* for which the knowledge about the *Application Requirements* and the *Sensor Features* should be used. At this stage, multiple sensors can be considered. Details are described in Sections 2.1 and 2.2. Very likely, at the first step, multiple high-quality devices will be picked, but the choice implies the costs of both the sensors and the hardware to process the data arriving from them. This leads to the initial value of the solution costs, which we define as the *Optimizing metric*. If, after the first iteration, the result is within the budgetary limit of the project, the selection can be directly used for the robot's sensor rig. If not, one must repeat the process, by iteratively relaxing the *Satisficing Metrics* until the reasonable cost level is reached.

2.1. Application requirements definition

To be able to specify the needs of the application, we propose to characterise the intended system according to the following requirements (for the extended names and process scheme refer to Fig. 1):

R1 The physical dimensions of the robot and the required area that should be perceived constantly around it need to be defined.

R2 Kinematics and the intended direction of the robot motion should be considered. E.g. if the robot travels forward only, a sensor on the back is redundant.

R3 It is also important to match how quickly and how far the robot can see with the application it is intended to. For example, if the robot is supposed to detect dead-ends on the fly, while travelling across a complex combination of corridors, both the range and the frequency should allow correct decisions. This metric is coupled heavily to the environment characteristics and path planning algorithm used by the robot.

R4 Various objects in the environment can appear in the robot's surroundings rapidly or slowly. It is important to ensure that the frequency of observation (the frame rate) is fast enough to notice object dynamics, i.e. that the robot can notice the motion of the object quickly enough to react [12].

R5 Similarly, the frame rate should be adjusted to the robot's velocity. Distance travelled between consecutive observations should be small enough to allow the robot to replan the path and avoid obstacles on the way.

R6 The resolution of the sensor should be high enough to notice the tiniest objects that can endanger the robot. For example, if the environment is full of hanging, thin cables the sensor must be able to reflect them in the perceived point cloud [13].

R7 This requirement expresses how accurately the objects should be perceived by the robot to safely conduct its operation. Low-quality sensors might experience distortion that influences reconstruction accuracy, which, depending on the application, might become harmful. As an example, a sensor might not be considered flat the surfaces that actually are flat [14].

R8 The lighting in the environment has a direct impact on a sensor's performance. If the robot is supposed to work with limited light or in darkness, the sensors' visible light spectrum must be selected accordingly, potentially with active projection [15].

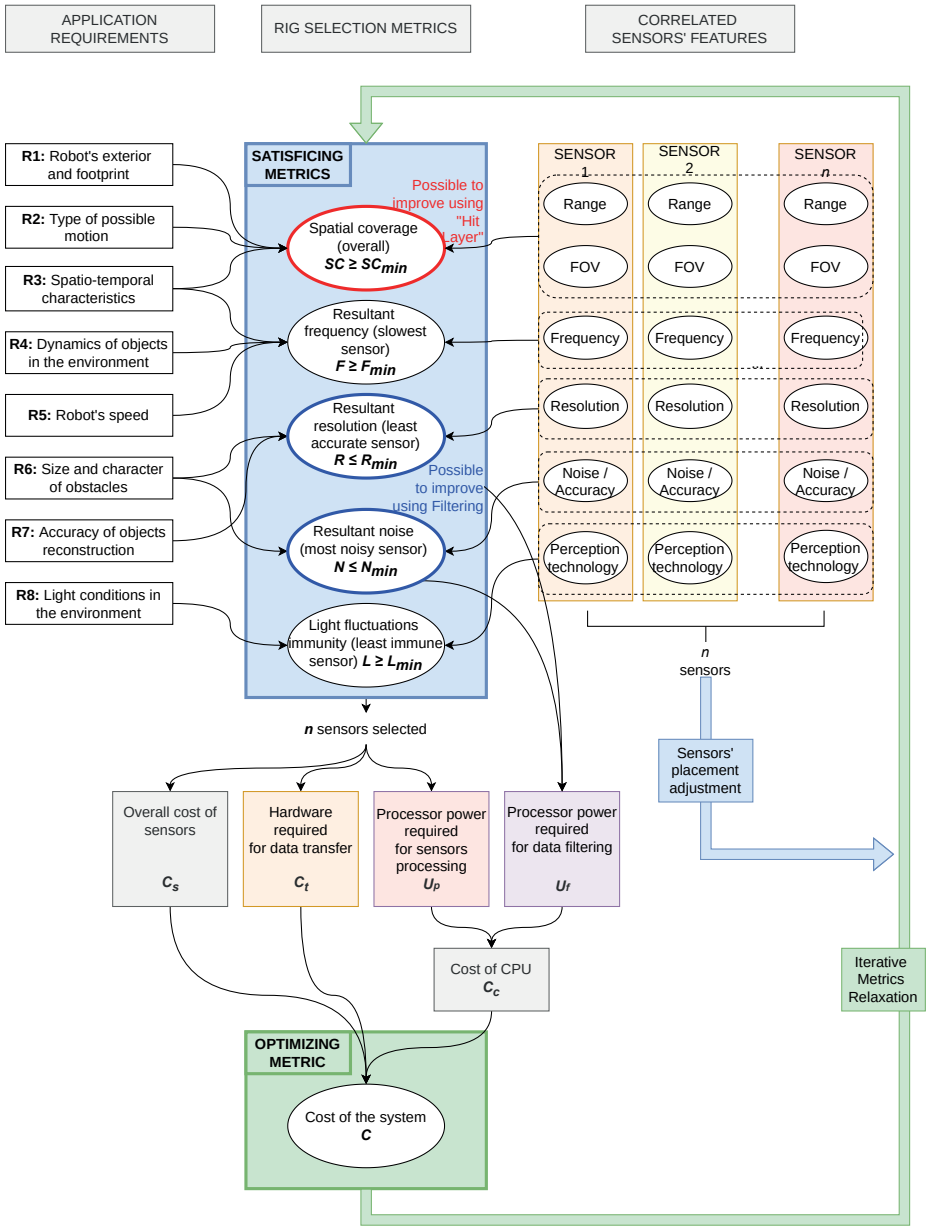

Fig. 1. Overview of the selection process in the form of Multi-Objective Optimization with Metrics Relaxation
Rys. 1. Schemat procesu selekcji w postaci optymalizacji wielokryterialnej z relaksacją metryk

2.2. Sensors features

Comparison of different sensors exceeds the scope of this paper, as the variety of available solutions is massive. There exists multiple publications on sensor types and technologies. Specifically detailed description and comparison can be found in [16, 17]. A comprehensive explanation of various characteristics of sensors in [18], Chapter 4. To be able to conduct the analysis process, the following features of potential sensors should be considered:

- **Range**: what is the distance at which the sensor can effectively notice an object
- **FOV**: Field Of View, defines the shape of the perception space. Usually a pyramid or cone.
- **Frequency**: (alternatively Frame Rate), a notion of how often the sensor can receive views of the environment.
- **Resolution**: how many discrete points are available per single scan and FOV cross-section. Usually, drops with the range.
- **Noise and accuracy**: expresses the noise level and ability to precisely reconstruct object shapes.
- **Perception Technology**: how is the point cloud generated e.g. Stereo Camera (passive), Structured Light (active), Time Of Flight (active) etc.

2.3. Satisficing metrics

Having analysed Application Requirements and Sensor Features one can conclude five metrics that are intended to define the "No-go" limits of the selection. In other words, Satisficing metrics are the system's minimum requirements at the given selection process iteration. The list includes (brackets explain the corresponding units):

- Spatial Coverage **SC** [% of coverage], described in details in the following section.
- Resultant Frequency **F** [Hz], encompasses the publication frequency of all sensors (i.e. how often new data is observed) and expresses the combined metric as the frequency of the slowest sensor.
- Resultant Resolution **R** [mm], expressed as the resolution of the lowest-resolution sensor at the distance of interest.
- Resultant Noise **N** [mm], expressed as the noise level of least accurate sensor.
- Light Fluctuations Immunity **L** [binary], we propose as binary metric whether the sensor can handle given lighting conditions.

It is important that *Satisficing metrics* are not modified directly. Instead, *Application Requirements* and *Sensor Feature*s should be adjusted (relaxed) to alter *Satisficing metrics*.

Spatial Coverage SC For the purpose of this paper, we introduce a new metric that encapsulates the notion of how well the space around the robot is covered. Where and how the sensors are mounted on the robot results in a complex shape of the combined FOV, as shown in the example from Fig. 2a). To qualitatively express the coverage, we propose to compute intersections of the combined Field of View at various angles around the vertical axis of the robot and calculate areas covered by sensors in each intersection from ground to the robot's height $\boldsymbol{h}$. As the robot has a non-zero diameter, we propose to take into account only the area from the robot's exterior $\mathbf{r}$ to the given range $\boldsymbol{d}$. Examples of such intersections for $r = 450$ mm and $d = 3500$ mm can be seen in Fig. 2b, c). As it is visible, depending on the angle, different ratio of space is covered by the combination of the sensors. We propose to combine the coverage ratios into the **Spatial Coverage** metric expressed as two graphs of coverage plotted against rotation angle, as depicted in Fig. 2 d, e). The first one shows the complete coverages $h \times [r,\ d]$. We also introduce the second metric which corresponds to floor coverage within the range $[r,\ d]$. To justify the purpose of the second metric, one must remember that most obstacles "stand on the ground" therefore even if an object is not completely visible, noticing just the bottom part might be sufficient for efficient navigation. As this is very application-specific, it is a designer who needs to decide to which part of the complete space (as defined by the first graph) the system can be limited. If there are hanging or "sticking-out-of-wall" objects in the environment, they may impose danger to the robot as depicted in Fig. 7. This however can also be mitigated software-wise as described in section 4.

2.4. Optimizing metric and iterative relaxation

The final optimization parameter of the process is the purchase cost of the complete system. It consists of multiple elements mentioned in the following section:

Overall cost of sensors This cost component is expressed as the sum of prices P_{si} of all sensors:

$$C_s = P_{s1} + P_{s2} + \ldots + P_{sn}. \tag{1}$$

Overall cost of data transfer hardware This cost component incorporates the purchase price of devices like routers, cables, and network cards. To select appropriate components the amount of data per second generated by each sensor must be computed. More specifically, each sensor occupies a specific bandwidth B_i, which is a product of the weight of single data shot w_i and i-th sensor's frequency f_i, expressed in Bps (bits per second):

$$B_i = \omega_i \cdot f_i. \tag{2}$$

Where w_i results from the Field of View of the sensor, resolution, precision of data representation etc., and is a characteristic of each device usually expressed in bytes. The bandwidth occupied by all sensors is the sum of each sensor's bandwidth:

$$B = B_1 + B_2 + \ldots + B_n. \tag{3}$$

Each component in the transfer system (cables switches, routers etc.) should be selected to have a bandwidth capacity higher than all sensors' bandwidth $B_{tj} > B$. The cost of transfer components is the sum of prices P_{tj} of all m devices and components in the system that fulfill the aforementioned criterion:

$$C_t = P_{t1}(B_{t1}) + P_{t2}(B_{t2}) + \ldots + P_{tm}(B_{tm}). \tag{4}$$

Cost of CPU This cost component describes the cost of a processing unit used for performing operations on the data arriving from sensors. To define processing power requirements of such a CPU, two factors need to be taken into accout:

- **Processing power for sensor operation**. Each sensor is operated by means of a software driver which performs data reception, conversion into the appropriate format, and communication with higher-level software. The exact processing power requirement for i-th sensor U_{pi} is difficult to analytically compute and requires experimental measurement for each sensor.
- **Processing power for data filtering**. If a sensor needs specific preprocessing e.g. filtering, additional CPU power is needed U_{fi}. Further analysis of this topic is described in Section 3.

The CPU should be selected in a way that fulfills the following condition:

$$\begin{aligned} U_p &= U_{p1} + U_{p2} + \ldots + U_{pn} \\ U_f &= U_{f1} + U_{f2} + \ldots + U_{fn} \\ U_{CPU} &> U_p + U_f. \end{aligned} \tag{5}$$

And the cost of the CPU is the purchase price of a device:

$$C_c = P(U_{CPU}). \tag{6}$$

Finally, the overall cost of the system is expressed simply as the sum of all costs mentioned above:

$$C = C_s + C_t + C_c. \tag{7}$$

If the resultant cost exceeds the intended budget, one should return to the initial step and relax the *Application Requirements* and desired *Sensor Features*. At this step, if it is mechanically possible, one might consider modifying the mounting position of the sensors. When wisely considered, this can even reduce the number of sensors. Finally, the process of relaxation should be repeated until a satisfactory cost is achieved.

2.5. Applicability

UR Cleaner use case The described process was used for the selection of sensors for the autonomous cleaning robot The resultant sensor rig consists of five depth sensors. Kinect Azure [26] as the front sensor, three PMD pico flexx [27] for side and back and the Velodyne VLP-16 Puck [28] for the top. The Satisficing metrics for which the acceptable cost level (which cannot be disclosed due to business sensitivity) was reached are described in Table 1. The **SC** metric for this robot is depicted in Fig. 2 d, e).

Tab. 1. Satisficing metrics for industrial use case
Tab. 1. Metryki satysfakcjonujące w przypadku przemysłowego robota sprzątającego

F [Hz]	**R** [mm]	**N** [mm]	**L** [binary]
15	122	30	True

Potential of generalization The proposed approach for the selection of sensors was designed to generalize for various robot types. Specifically one can consider a situation when the robot does not move on a flat surface but instead moves on uneven terrain, in the air, or under water. In such scenarios, the robot's orientation varies in a 3D space. Importantly, all *Application Requirements* remain valid but require careful

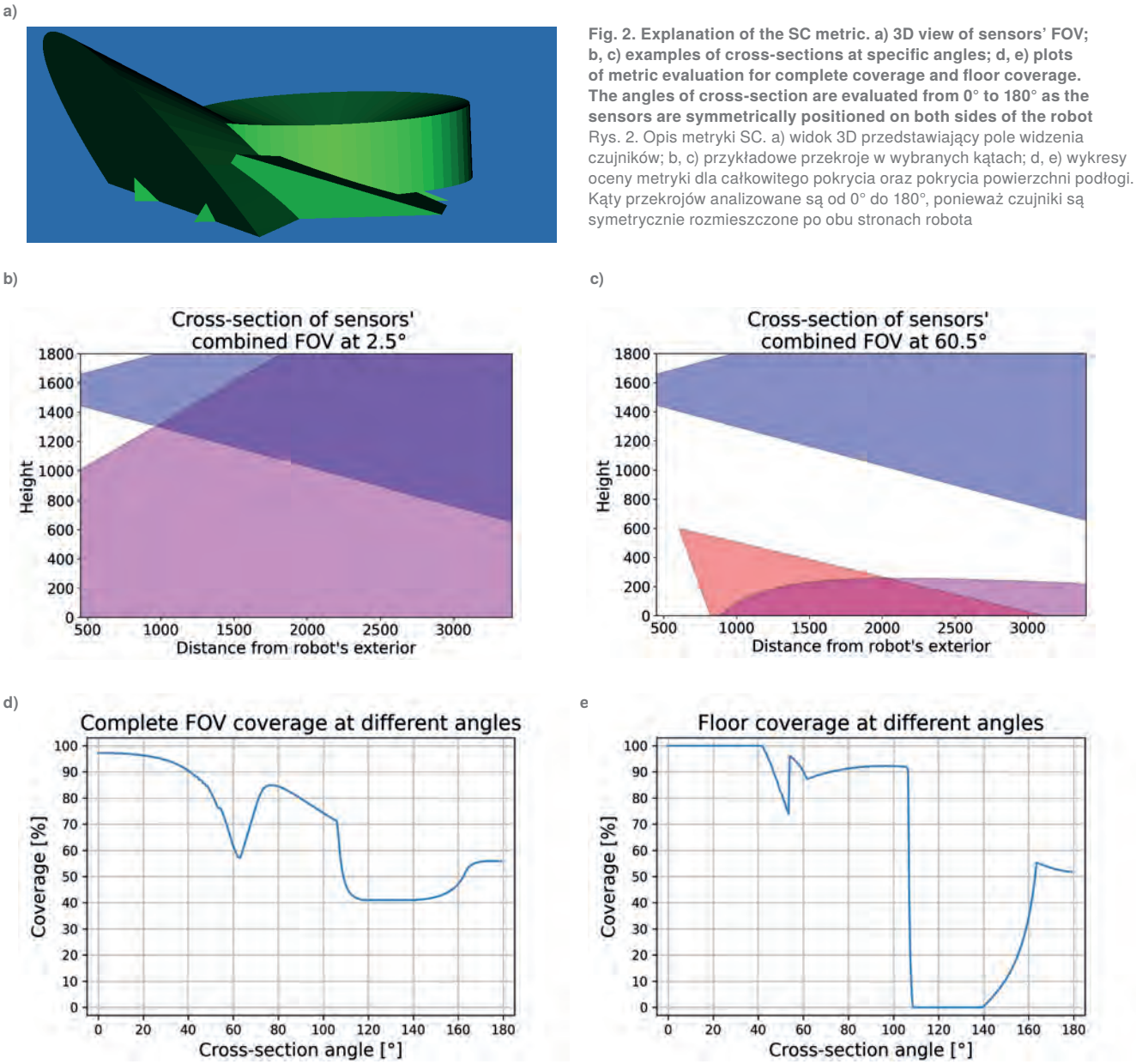

Fig. 2. Explanation of the SC metric. a) 3D view of sensors' FOV; b, c) examples of cross-sections at specific angles; d, e) plots of metric evaluation for complete coverage and floor coverage. The angles of cross-section are evaluated from 0° to 180° as the sensors are symmetrically positioned on both sides of the robot
Rys. 2. Opis metryki SC. a) widok 3D przedstawiający pole widzenia czujników; b, c) przykładowe przekroje w wybranych kątach; d, e) wykresy oceny metryki dla całkowitego pokrycia oraz pokrycia powierzchni podłogi. Kąty przekrojów analizowane są od 0° do 180°, ponieważ czujniki są symetrycznie rozmieszczone po obu stronach robota

analysis and potentially experimental verification. E.g. the same sensor working in an outdoor environment might experience different noise characteristics than in the indoor case. Similarly, the dynamics of the objects in the environment and the size of obstacles heavily depend on the application use case. Nevertheless, most of the *Satisficing Metrics*, namely **F**, **R**, **N**, **L** remain fully valid regardless of the working conditions of the robot whether it is an outdoor, indoor, structured environment, air or underwater robot. The remaining metric i.e. **SC** is defined within the space limited by the flat ground in the bottom and specific height at the top. Such a definition is especially useful for robots operating in structured environments and does not apply to robots whose orientation varies in 3D space. It can however be easily generalized by not reducing the lower and upper limits but using the complete sphere around the robot instead.

3. Data preprocessing

When the sensors are selected one has to consider whether using raw data outputs for multi-sensor systems is feasible. Sensors produce high-frequency data that can strain the robot's computational resources. Filtering serves to reduce the data volume, thereby facilitating more efficient real-time processing. Furthermore, it diminishes noise and filters out irrelevant information, enhancing the accuracy of environment mapping, object detection, and navigation. This specifically allows to use cost-effective sensors while ensuring data quality through effective filtering techniques. Additionally, filtering ensures that data processing is consistent with sensor data rates, preventing delays and enabling timely responses to dynamic environmental conditions.

Multiple filtration methods have been developed so far and can be used together but the overall performance is highly dependent on the combination of parameters and sequence of processing. This section introduces some wellknown filtration methods and evaluates the influence of the order of execution on computational efficiency. A comparative evaluation is presented based on experimental results.

3.1. Preprocessing methods

Out of multiple algorithms which have been developed so far, we selected four well-established methods that, in our assessment, are sufficient to construct effective filtering pipelines:

– Statistical Outlier Removal — a method that removes noise by analysing the distribution of distances between neighbouring points. Points with distances significantly deviat-

ing from the mean are classified as outliers and excluded, resulting in a cleaner and more reliable point cloud [7, 19]. The filter is parameterized by the number of neighbouring points **N** and the standard deviation multiplier **M**, which defines the threshold as a multiple of the standard deviation from the mean distance. Points falling outside this range are classified as outliers and subsequently removed.

- Voxel Grid Filtering — A method for downsampling point clouds by approximating points within a cubic region (voxel) with their centroid [7, 19]. The process is parameterized by a voxel grid size **V**.
- Pass-Through Filters — a method that selects points within a specified range along a given axis (e.g., x, y, or z). Points outside this defined interval are excluded, allowing for targeted segmentation of a region of interest [19]. Parametrized by coordinates of interval **<A, B>** along given axis.
- RANSAC (Random Sample Consensus) algorithm for plane segmentation — an algorithm used to identify and segment planes in point clouds by fitting models to random subsets of points and selecting the model with the highest number of inliers [19, 20]. Parametrized by max iterations **I** that specifies the maximum number of iterations to run, determining how many random samples are tested and distance threshold **T** that defines the maximum allowable distance from a point to the plane for it to be considered an inlier.

Despite their simplicity, these techniques can be combined to create efficient filtration pipelines that address the constraints of time and resources. We will present example pipelines that integrate these filters to achieve both resource efficiency and effective operation.

3.2. Proposed filtration pipelines

In this study, we propose four distinct data filtration pipelines for point cloud processing, which essentially consist of various combinations of the four basic filtration algorithms described before. We aimed to develop a filtration pipeline that allows getting rid of information that is irrelevant for the navigation and would cause unnecessary load when other algorithms of a robot, such as navigation, utilise the data. This can be achieved not only by reducing the noise level and downsampling the point clouds but also by making additional assumptions. More specifically, in our case, the sensors are firmly located in the robot's reference frame and the robot is always traveling on the ground, therefore the surface of contact (floor) can be removed. This further reduces the weight of the point cloud. With this in mind, we propose four pipelines consisting of noise removal, size reduction, and floor removal in various combinations as depicted in Fig. 3.

For the first three pipelines, a two-step process for floor removal, referred to as "Floor Removal 1", is implemented in the following manner:

1. The region of interest, where the floor plane is assumed to be located, is identified using a pass-through filter. The point cloud is divided parallel to the estimated floor plane, creating a slice within which the plane is searched.
2. The floor plane is then segmented using the RANSAC algorithm and corresponding points are removed.

In the fourth pipeline, the floor removal process, referred to as "Floor removal 2", is enhanced by introducing an additional voxelization step, applied to a copy of the original point cloud before running the RANSAC algorithm. By using a voxel filter with larger voxel dimensions, the floor plane equation can be determined more efficiently, allowing points to be segmented based on their distance from the plane (same value as RANSAC threshold).

3.3. Empirical Experiment Results

Experiments on proposed pipelines were conducted using 100 example point clouds gathered from an interior of UR offices employing Microsoft Azure Kinect ToF camera. They consist of vast open spaces (typical for industrial cleaning robots), as well as office objects like tables, chairs, etc. An example point cloud that consists of 144 545 points, is presented in Fig. 4, top. We assume that the point cloud is already in the robot's navigational coordination frame that is placed on the floor plane.

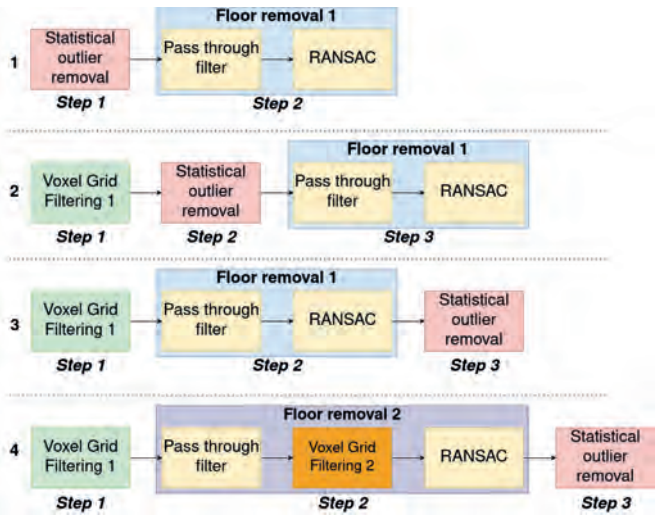

Fig. 3. Proposed filtration pipelines
Rys. 3. Proponowane sposoby filtracji

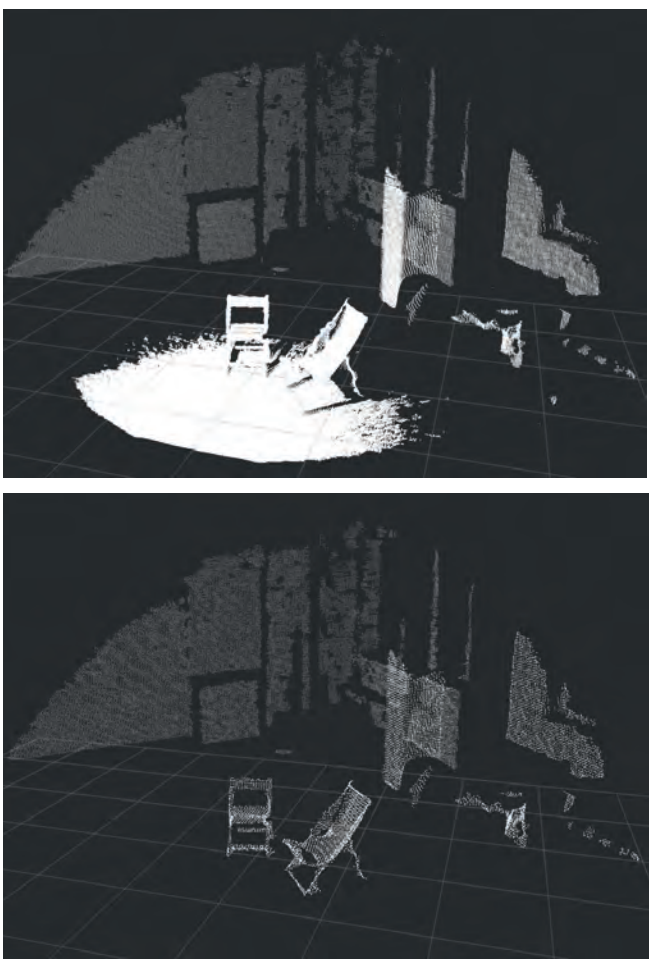

Fig. 4. Top: Example initial point cloud, Bottom: Corresponding result point cloud for Pipeline 4
Rys. 4. Góra: Przykładowa wejściowa chmura punktów; Dół: ta sama chmura odfiltrowana zgodnie z procesem Pipeline 4

Tab. 2. Filters parameters for analyzed pipelines. V1 is voxel grid size for "Voxel Grid Filtering 1" and V2 for "Voxel Grid Filtering 2"
Tab. 2. Wartości parametrów filtrów dla analizowanych sposobów filtracji; V1 – wielkość siatki woksela dla „Voxel Grid Filtering 1", V2 – dla „Voxel Grid Filtering 2"

N	M	A [cm]	B [cm]	T [cm]	I	V1 [cm]	V2 [cm]
10	1	-10	10	3	100	3	30

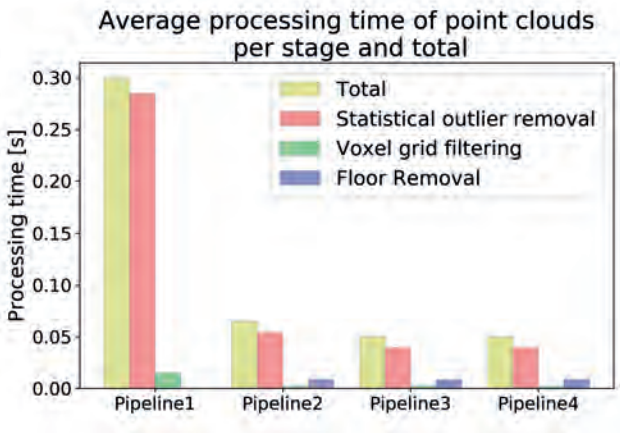

Fig. 5. Comparison of the four filtering pipelines by their average processing time total and in each stage
Rys. 5. Porównanie czterech proponowanych sposobów filtracji pod względem całkowitego średniego czasu przetwarzania oraz czasu przetwarzania na każdym z etapów

Thanks to that, the time required for transformation is ignored and the operation of floor plane removal is more intuitive.

To get a comparable result, the same filters used in different pipelines have the same parameters, as presented in Tab. 2. To select filtration parameters, prior to the experiments, an extensive review of scientific literature on point cloud filtering techniques was conducted. The review identified only a single study that implemented precise versions of both statistical outlier filtering and voxel grid filtering, along with a thorough parameter analysis [6]. To the best of our knowledge, no existing studies offer a comprehensive evaluation of filter parameters. Due to that, the parameter values for this research were established based on the authors' industrial experience, ensuring an optimal balance between detailed scene reconstruction and processing efficiency, while preserving crucial data necessary for navigation. The initial voxel grid filtering was configured with a grid size large enough to reduce the number of points, yet small enough to maintain an appropriate resolution for accurate object representation. Parameters for statistical outlier removal were selected to eliminate outliers without omitting small objects. The parameters for floor removal (using RANSAC and Pass-through filters) were optimized to ensure computational efficiency while retaining small floor-level objects. For the secondary voxel grid filtering, a larger grid size was chosen to accelerate processing without compromising critical data. Pipelines were compared in terms of processing time (total and per stage) as depicted in Fig. 5, and the number of points in the resultant point cloud, shown in Fig. 6.

An examination of Fig. 5 reveals that reducing data complexity through basic voxel grid filtering significantly decreases processing time. With each subsequent optimization, the processing time can be further reduced to below 0.05 seconds, corresponding to a frequency exceeding 20 Hz, which is higher than the operating frequency of any sensor listed in Table 1 for before mentioned UR case. Furthermore, as illustrated in Fig. 4, the example output point cloud produced by Pipeline 4 (the most

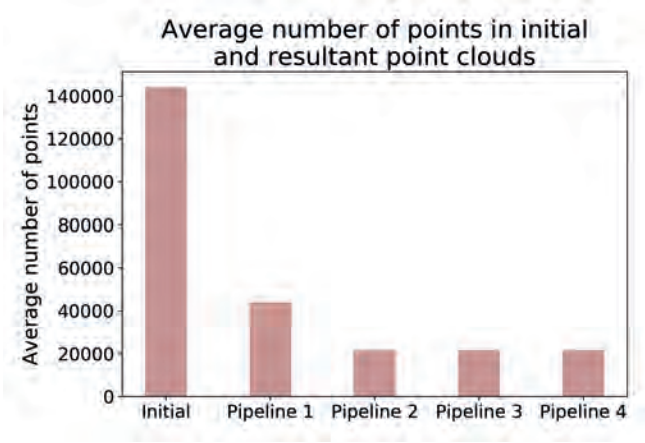

Fig. 6. Comparison of the four filtering pipelines by their average number of points in resultant point clouds and average number of points in corresponding initial point clouds
Rys. 6. Porównanie czterech proponowanych sposobów filtracji pod względem średniej liczby punktów w uzyskanych chmurach punktów oraz średniej liczby punktów w odpowiadających im początkowych chmurach punktów

timeefficient option) demonstrates the successful elimination of the floor plane and a reduction in the number of small outliers, thereby providing accurate yet efficient environmental reconstruction. The differences in memory consumption across the pipelines are negligible, amounting to around 200 MB of RAM memory on average for each pipeline. This indicates that the proposed methods can be effectively applied in mobile robotics, enhancing autonomous navigation capabilities.

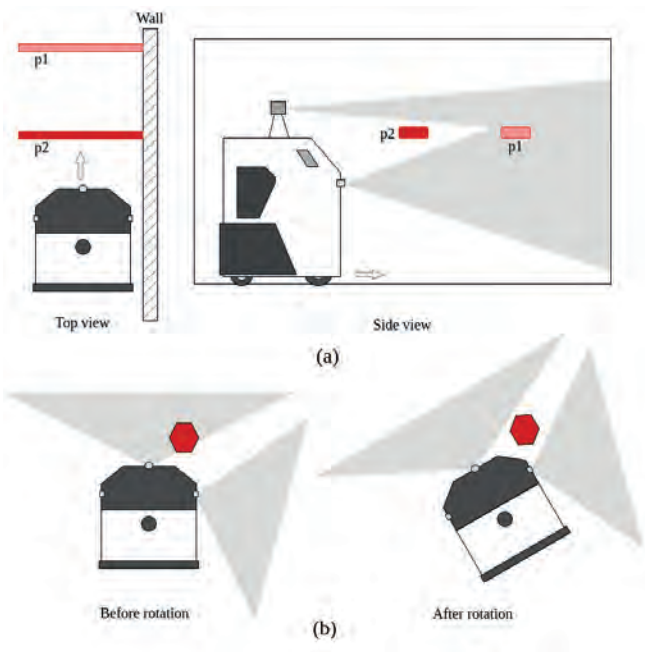

Fig. 7. Examples illustrating navigation challenges caused by partial sensor coverage. (a) The obstacle protruding from the wall is detectable in pose p1 relative to the robot, but after the robot's linear displacement, it becomes undetectable in the new relative pose p2. (b) An obstacle below the Lidar's detection range is identified by the front sensor, but after the robot's rotation, it is no longer detectable
Rys. 7. Przykłady ilustrujące wyzwania nawigacyjne spowodowane częściowym pokryciem przestrzeni przez czujniki. (a) Przeszkoda wystająca ze ściany jest wykrywalna w położeniu p1 względem robota, ale po liniowym przemieszczeniu robota staje się niewykrywalna w nowym położeniu p2. (b) Przeszkoda znajdująca się poniżej zasięgu detekcji LIDARa jest identyfikowana przez czujnik przedni, jednak po obrocie robota przestaje być wykrywana

4. Mitigation of limited field of view

Once the detailed analysis is conducted and the optimal arrangement of sensors is not achievable due to the system costs, one can resort to the algorithmic solution to eliminate the threats caused by the blind spots of the sensory rig during the navigation. More specifically, with the limited FOV of sensors, achieving full coverage of the surroundings at all times is not possible. An obstacle may be detected in one position, but as the robot moves, the same obstacle could fall into the sensors' blind spot (Fig. 7). This makes navigation challenging, especially when moving close to the walls and obstacles. To address this issue, we propose an approach that utilises spatio-temporal sensor's information.

The core concept of our proposal is to update the navigation map by integrating both current sensor data and previous detections, along with the robot's movements. This approach is implemented within the framework of the costmap [21], a crucial component of the navigation stack in the Robot Operating System (ROS) [22]. Nevertheless, it is adaptable and can be applied to other navigation systems as well. In the following section, we provide a brief overview of the navigation costmaps. This is followed by an explanation of the proposed approach, called Hit Map, and the presentation of empirical experiment results.

4.1. Costmap-Based Navigation

Navigation systems require a well-defined configuration space. Costmaps provide an efficient way to represent the robot's operational environment, facilitating safe and optimal path planning. As illustrated in Fig. 8, the Costmap module receives

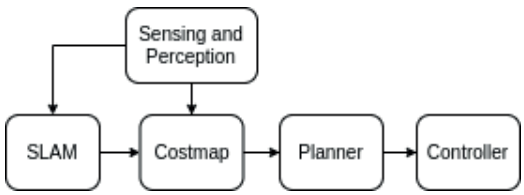

Fig. 8. Main components of the navigation system
Rys. 8. Główne komponenty systemu nawigacji

the map and the robot's pose from the SLAM (Simultaneous Localization and Mapping), along with processed sensor data from the Perception system [23]. It processes this information into a 2-D grid of cells, known as an occupancy grid, which serves as the primary input for the planning system. Each cell in the occupancy grid has a value representing the cost of traversing through that grid cell [24].

To enhance flexibility and accommodate various navigational constraints, a layered approach is proposed that separates the processing of costmap data into semantically distinct layers [21, 25]. In this approach, each layer handles data from a specific source using designated functionalities. The Costmap features a primary occupancy grid, known as the master grid, while each layer maintains its own layer grid. Three basic layers are commonly used in navigation systems:

- Static Layer: Represents data from the Mapping component, providing a representation of the environment.
- Obstacle Layer: Aggregates data from sensors to represent obstacles as a unified obstacle grid. Each grid cell is either free or occupied.
- Inflation Layer: Accounts for the robot's dimensions, specifically its footprint, by inflating the obstacles in the final grid.

The obstacle grid is the main input for the proposed Hit Layer.

4.2. Hit Map

In this section, we propose a new Costmap layer, referred to as the Hit Layer. The occupancy grid associated with this layer, termed the Hit Map, contains cell values (hit) that are proportional to the duration of time each cell has been detected either as an obstacle or free.

Figure 9 depicts the structure of the Costmap module, incorporating the Hit Layer. The Costmap module is responsible for updating the master grid, a fundamental element for all autonomous planning functions, at a predefined frequency. Throughout this process, each layer refreshes its respective occupancy grid, which in turn contributes to the update of the master grid. Although the Costmap module may consist of several layers, the Obstacle Layer plays a critical role in the creation and continuous updating of the Hit Layer.

Obstacle Layer Update The Perception module receives data from sensors and applies preprocessing operations (e.g. the actions described in Section 3). Processed data is then published into specific channels. The Obstacle layer maintains an Observation Buffer for each specified data channel. During the update process, it modifies the obstacle grid based on the most recent data in the Observation Buffers. Each cell in the obstacle grid will have one of two values, indicating whether it is free or occupied. Then, the master grid is modified based on the updated obstacle grid.

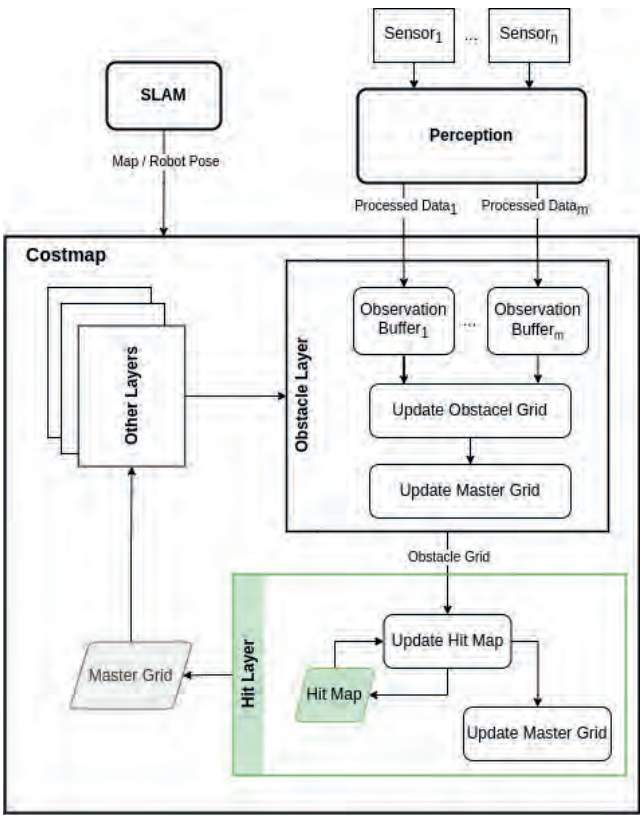

Fig. 9. Update process overview of the costmap module which includes the proposed Hit Layer: Sensor data is processed and published into designated channels, where observation buffers temporarily store it. During each update cycle, the Obstacle Layer refreshes the Obstacle Grid using the data from the observation buffers and updates the Master Grid. The Hit Layer then revises the Hit Map based on the current Obstacle Grid and finalizes the Master Grid update
Rys. 9. Przegląd procesu aktualizacji mapy kosztów, który obejmuje proponowaną warstwę Hit Layer: Dane z czujników są przetwarzane i publikowane w wyznaczonych kanałach, gdzie tymczasowo przechowywane są w buforach obserwacji. Podczas każdego cyklu aktualizacji, warstwa przeszkód odświeża siatkę przeszkód, korzystając z danych z buforów obserwacji, i aktualizuje mapę główną. Następnie warstwa Hit Layer dokonuje rewizji mapy trafień na podstawie aktualnej mapy przeszkód i finalizuje aktualizację mapy głównej

Hit Layer Update Each Hit Map cell is an integer within the range [0, hit_{max}]. Initially, all cell values are set to zero. The Hit Layer updates the Hit Map based on the latest obstacle grid received from the Obstacle Layer, as well as the robot's movements in three main steps. Figure 10 illustrates the Hit Map update process using a simplified example.

First, the Hit Map is relocated using a Sliding Window approach, which accounts for the robot's displacement. Subsequently, the portion of the Hit Map corresponding to newly covered areas is reset to 0, while the cells that remain in the common area before and after the update are left unchanged.

In the next stage, the Hit Map is updated. During this process, the value of each cell in the Hit Map is recalculated according to the corresponding cell in the obstacle grid, as shown in equation 8. Let h_i represent the i-th cell in the Hit Map, and o_i represent the i-th cell in the obstacle grid. If o_i is occupied, h_i is increased according to the hit and hit_{max} parameters. Conversely, if oi is free, hi is decreased by one.

$$h_i = \begin{cases} \min(h_i + hit, hit_{\max}) & \text{if } h_i \text{ is occupied,} \\ \max(0, h_i - 1) & \text{otherwise} \end{cases} \tag{8}$$

Finally, the master grid is updated based on the Hit Map. Let m_i denote the i-th cell in the master grid, and let $c_{occupied}$ represent the cost value of an occupied cell. The value of m_i is then updated according to equation 9:

$$m_i = \begin{cases} c_{occupied} & \text{if } h_i > 0, \\ m_i & \text{otherwise} \end{cases} \tag{9}$$

By adding the Hit Layer to the set of layers in the costmap, obstacles that have been detected for a longer duration are more likely to be retained in the Hit Map, even if they are no longer detected. The parameters hit and hit_{max} control the cautiousness of the Hit Layer. Higher values for these parameters result in safer navigation, as obstacles are more likely to be retained. However, this can also lead to more cells being marked as occupied in the costmap. Choosing appropriate values depends on the specific environment in which the robot is operating and the coverage provided by the sensors.

4.3. Motion Condition

The Hit Map update process involves two main operations: incrementing and decrementing. By decrementing, occupied cells that are no longer detected by the Obstacle Layer will eventually be freed. This helps to prevent costmap from becoming cluttered. However, this approach also introduces a potential issue. If the robot remains stationary for an extended period, the Hit Map continues to decrement and free the cells. If some of these cells correspond to obstacles that have moved into a blind spot, they might not be considered when the robot starts moving again.

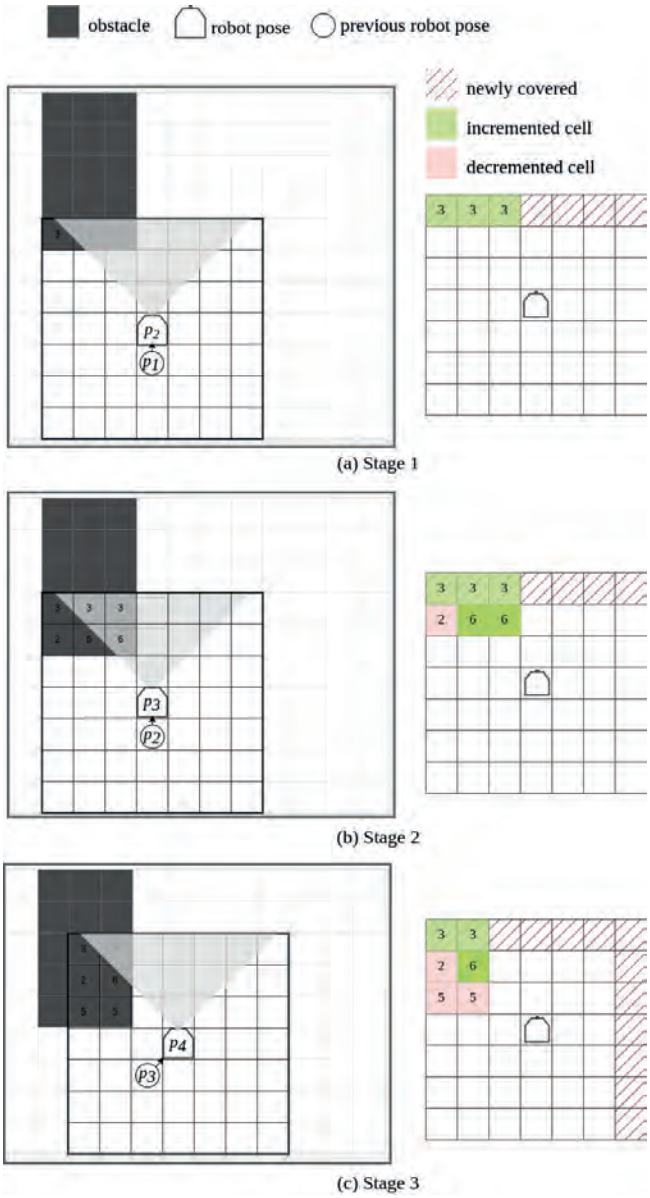

Fig. 10. An illustrative example of the Hit Map update process. In this example, the Hit Map is updated when the robot moves to a new cell, though in practice, costmaps typically have a higher update frequency. On the left, the positions of the obstacle and costmap are shown, while the Hit Map is displayed on the right. Cells where an obstacle is detected and the hit value is increased are shown as incremented cells. Additionally, cells that are free in the obstacle grid but have a hit value greater than zero (still considered occupied in the final master grid) are displayed as decremented cells

Rys. 10. Przykład ilustrujący proces aktualizacji mapy trafień. W tym przykładzie mapa trafień jest aktualizowana, gdy robot przemieszcza się do nowej komórki, chociaż w praktyce mapy kosztów zazwyczaj mają wyższą częstotliwość aktualizacji. Po lewej stronie przedstawiono pozycje przeszkody i mapa kosztów, natomiast po prawej stronie widoczna jest mapa trafień. Komórki, w których wykryto przeszkodę i wartość trafienia została zwiększona, są oznaczone jako komórki inkrementowane. Dodatkowo komórki, które są wolne w mapie przeszkód, ale mają wartość trafienia większą niż zero (wciąż uznawane za zajęte w finalnej mapie głównej), są wyświetlane jako komórki dekrementowane

Fig. 11. Experiment setup for navigation test with and without the Hit Layer

Rys. 11. Stanowisko eksperymentalne do testów systemu nawigacji z warstwą Hit Layer oraz bez niej

To address this issue, the motion condition is evaluated during the decrement step using two data sources: (1) velocity commands from the controller and (2) the robot's displacement over a predefined time interval. Specifically, if the robot's displacement within the specified time period, denoted as $d_{duration}$, falls below a defined threshold $d_{threshold}$, the robot is considered stationary, and the decrement operation is skipped.

4.4. Empirical Experiment Results

A test scenario was designed to evaluate the efficiency of the proposed Hit Layer. This scenario involves a wallfollowing operation, where the navigation system must guide the robot to maintain a close distance to the wall. This is particularly challenging in industrial environments, where various objects of differing heights may be present near the walls and could fall into the sensors' blind spots. Due to the proximity to the wall, such scenarios increase the risk of collisions. Specifically risky are the situations where the objects do not stretch vertically from the ground but "stick out of the wall".

Figure 11 illustrates the test setup, which includes a table with a foam layer on top. The foam extends beyond the edges of the table and is only detectable by the robot's front depth camera. We conducted tests both with and without the Hit Layer. Table 3 summarizes the parameters chosen for the test scenario.

Tab. 3. Hit layer parameters selected for the wall-following navigation test

Tab. 3. Parametry warstwy Hit Layer wybrane do testów nawigacji w trybie śledzenia ściany

hit	$hit_{\max}$	d_{thresh} [mm]	$d_{duration}$ [s]
3	1500	0.1	10

Figure 12 (a) shows the costmap shortly after the wall-following behaviour begins, but before the robot reaches the table. At this stage, the front sensor detects the foam. As the robot approaches the foam, it starts navigating around it. Figure 12 (b) and (c) represent the costmap in this pose, with and without the Hit Layer, respectively. When using the Hit Layer, the foam that was previously detected by the camera remains in the costmap for an extended period, allowing the robot to navigate around it successfully. In contrast, without the Hit Layer, the foam disappears from the costmap, causing the robot to move to close to the table and hit the foam.

5. Conclusions

In this paper, we have presented three strategies that allow to engineer an end-to-end perception system of a robot. The presented selection process can be used to any autonomous robot in a generic way. We also evaluated the applicability of various filtration pipelines and experimentally evaluated their performance. Finally, we introduced a method to mitigate limited sensors FOV thus allowing to fill perception system gaps and potentially limit the number of sensors. All strategies were successfully applied in the autonomous floor scrubber UR\Cleaner and can be reused in other autonomous robots.

Acknowledgments

Project co-financed by the European Union through the European Regional Development Fund under Action 1.1 of the Smart Growth Operational Programme 2014-2020. The project is implemented within the competition organized by the National Centre for Research and Development: Call number: 1/1.1.1/2017 – Industrial research and development work conducted by enterprises. Project Number POIR.01.01.01-00-0206/17: „Designing a prototype of an autonomous platform moving in a production environment".

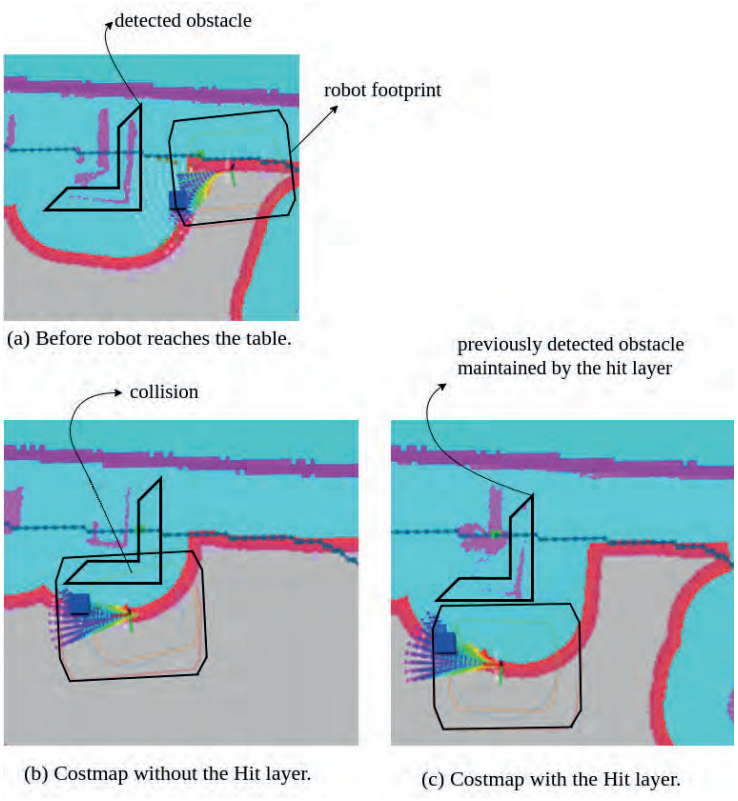

Fig. 12. Costmaps from the navigation test during wall-following, with and without the Hit Layer. The graphics are generated using rviz tool. (a) Before the robot reaches the table, which is within the observation range of the front camera. A few seconds later, the robot attempts to navigate around the table and the foam placed on it. Due to the robot's rotation, the foam falls into the camera's blind spot. (b) Without further detection, the robot moves too close to the table and collides with the foam. (c) Using prior detections, the Hit Layer preserves the foam as an obstacle in the costmap, allowing the robot to successfully navigate around the table

Rys. 12. Mapy kosztów z testu nawigacji w trybie śledzenia ścian, z warstwą Hit Layer oraz bez niej. Grafiki zostały wygenerowane za pomocą narzędzia rviz. (a) Przed dotarciem robota do stołu znajdującego się w zasięgu kamery przedniej. Kilka sekund później robot podejmuje próbę ominięcia stołu oraz pianki umieszczonej na jego powierzchni. Ze względu na obrót robota, pianka znika z pola widzenia kamery. (b) W wyniku braku dalszej detekcji robot przemieszcza się zbyt blisko stołu i uderza w piankę. (c) Dzięki wcześniejszym detekcjom warstwa Hit Layer zachowuje piankę jako przeszkodę na mapie kosztów, umożliwiając robotowi skuteczne ominięcie stołu

References

1. Shi Q., Li C., Wang C., Luo H., Huang Q., Fukuda T., *Design and implementation of an omnidirectional vision system for robot perception*, "Mechatronics", Vol. 41, 2017, 58–66, DOI: 10.1016/j.mechatronics.2016.11.005.
2. Deshpande P., Reddy V.R., Saha A., Vaiapury K., Dewangan K., Dasgupta R., *A next generation mobile robot with multi-mode sense of 3D perception*, [In:] 2015 International Conference on Advanced Robotics (ICAR). IEEE. 2015, 382–387, DOI: 10.1109/ICAR.2015.7251484.
3. Lambert J., Carballo A., Cano A.M., Narksri P., Wong D., Takeuchi E., *Performance analysis of 10 models of 3D LiDARs for automated driving*. "IEEE Access", Vol. 8, 2020, 131699–131722, DOI: 10.1109/ACCESS.2020.3009680.
4. Stoyanov T., Louloudi A., Andreasson H., Lilienthal A.J., *Comparative evaluation of range sensor accuracy in indoor environments*, [In:] 5th European Conference on Mobile Robots, ECMR 2011, Orebro, Sweden, 2011, 19–24.
5. Diaz M.G., Tombari F., Rodriguez-Gonzalvez P., Gonzalez--Aguilera D., *Analysis and evaluation between the first and the second generation of RGBD sensors*, "IEEE Sensors journal", Vol. 15, No. 11, 2015, 6507–6516, DOI: 10.1109/JSEN.2015.2459139.
6. Jin Y., Yuan X., Wang Z., Zhai B., *Filtering Processing of LIDAR Point Cloud Data*, [In:] IOP Conference Series: Earth and Environmental Science, Vol. 783, May 2021, DOI: 10.1088/1755-1315/783/1/012125.
7. Han X.-F., Jin J.S., Wang M.-J., Jiang W., Gao L., Xiao L., *A review of algorithms for filtering the 3D point cloud*, "Signal Processing: Image Communication", Vol. 57, 2017, 103–112, DOI: 10.1016/j.image.2017.05.009.
8. Moreno C., *A Comparative Study of Filtering Methods for Point Clouds in Real-Time Video Streaming*, [https://api.semanticscholar.org/CorpusID:162172879].
9. Roelofsen S., Gillet D., Martinoli A., *Collision avoidance with limited field of view sensing: A velocity obstacle approach*. [In:] 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, 1922–1927. DOI: 10.1109/ICRA.2017.7989223.
10. Bouraine S., Fraichard T., Salhi H., *Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments*, [In:] 2012 IEEE International Conference on Robotics and Automation, 2012, 174–179, DOI: 10.1109/ICRA.2012.6224932.
11. Phan D., Yang J., Grosu R., Smolka S.A., Stoller S.D., *Collision avoidance for mobile robots with limited sensing and limited information about moving obstacles*. "Formal Methods in System Design", Vol. 51, 2017, 62–86, DOI: 10.1007/s10703-016-0265-4.
12. He B., Wu S., Wang D., Zhang Z., Dong Q., *Fast-dynamic--vision: Detection and tracking dynamic objects with event and depth sensing*, [In:] 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, 3071–3078, DOI: 10.1109/IROS51168.2021.9636448.
13. Cop K.P., Peters A., Žagar B.L., Hettegger D., Knoll A.C., *New metrics for industrial depth sensors evaluation for precise robotic applications*, [In:] 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, 5350–5356, DOI: 10.1109/IROS51168.2021.9636322. Peter
14. Hausamann P., Sinnott C.B., Daumer M., MacNeilage P.R., *Evaluation of the Intel RealSense T265 for tracking natural human head motion*, "Scientific Reports", Vol. 11, 2021, DOI: 10.1038/s41598-021-91861-5.
15. Tadic T. at al. *Perspectives of RealSense and ZED depth sensors for robotic vision applications*, "Machines", Vol. 10, No. 3, 2022, DOI: 10.3390/machines10030183.
16. Pinto A.M. et al. *Evaluation of depth sensors for robotic applications*, [In:] 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, IEEE, 2015, 139–143, DOI: 10.1109/ICARSC.2015.24.
17. Haenel R., Semler Q., Semin E., Grussenmeyer P., Tabbone S., *Evaluation of low-cost depth sensors for outdoor applications*, "The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences", Vol. 48, 2022, 101–108, DOI: 10.5194/isprs-archives-XLVIII-2-W1-2022-101-2022.
18. Siegwart R., Nourbakhsh I.R., Scaramuzza D., *Introduction to autonomous mobile robots*. MIT press, 2011.
19. Rusu R.B., Cousins S., *3D is here: Point Cloud Library (PCL)*, [In:] 2011 IEEE International Conference on Robotics and Automation, 2011, DOI: 10.1109/ICRA.2011.5980567.
20. Fischler M.A., Robert C., Bolles A., *Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, "Communications of the ACM", Vol. 24, No. 6, 1981, 381–395, DOI: 10.1145/358669.358692.
21. Lu D.V., Hershberger D., Smart W.D., *Layered costmaps for context-sensitive navigation*, [In:] 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, 709–715, DOI: 10.1109/IROS.2014.6942636.
22. Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., Ng A.Y., *ROS: an open-source Robot Operating System, ICRA workshop on open source software*, 2009, Vol. 3, No. 3.2.
23. Raja P., Pugazhenthi S., *Optimal path planning of mobile robots: A review*, "International Journal of the Physical Sciences", Vol. 7, No. 9, 2012, 1314–1320, DOI: 10.5897/IJPS11.1745.
24. Carlos Andre Seara da Silva. *Robot Navigation in Highly--Dynamic Environments*. PhD thesis. University of Coimbra, 2022.
25. Lu D.V., Smart W.D., *Towards more efficient navigation for robots and humans*, [In:] 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, 1707–1713, DOI: 10.1109/IROS.2013.6696579.

Other sources

26. Microsoft. *Azure Kinect DK hardware specifications*, [https://learn.microsoft.com/sk-sk/previous-versions/azure/kinect-dk/hardware-specification].
27. Infineon. *Development Kit Brief CamBoard pico flexx*, [https://media.automation24.com/datasheet/en/PMD_DevKit_Brief_CB_pico_flexx_CE_V0218.pdf].
28. Velodyne. *VLP-16 | Ouster*, [https://ouster.com/products/hardware/vlp-16].

Systemy percepcji dla autonomicznych robotów mobilnych: wybór i redukcja ograniczeń

Streszczenie: Dobrze zaprojektowany system percepcji jest kluczowy dla prawidłowego działania autonomicznego robota mobilnego, jednak jego stworzenie nie jest trywialnym zadaniem. Aby stworzyć kompleksowy system, należy brać pod uwagę nie tylko cechy samych czujników, ale także sposób, w jaki dane z nich są używane do działania robota. W artykule przedstawiono zestaw ściśle powiązanych strategii, które umożliwiają stworzenie kompleksowego systemu percepcji, wykorzystującego reprezentację 3D w postaci chmur punktów. Nasza propozycja jest na tyle uniwersalna, że może być stosowana w różnych robotach mobilnych, ponieważ uwzględnia kontekst działania robota oraz ograniczenia sprzętowe. Pierwszą strategią jest wprowadzenie sformalizowanego procesu wyboru czujników, ujętego jako problem optymalizacji wielokryterialnej z relaksacją metryk, co pozwala na wybór optymalnego zestawu czujników w ramach ograniczeń budżetowych. W drugiej strategii weryfikujemy różne metody filtrowania danych i ich kombinację w kontekście systemu nawigacji, aby znaleźć kompromis między dokładnością a nakładem obliczeniowym. Wreszcie, aby zniwelować suboptymalne pole widzenia połączonych czujników i ulepszyć system percepcji, proponujemy nową koncepcję warstwy siatki zajętości, która wykorzystuje informacje o ruchu w obliczeniach dostępności obszaru nawigacyjnego. W ramach badań, przeprowadziliśmy eksperymentalną weryfikację tych strategii i zastosowaliśmy wyniki w autonomicznym robocie sprzątającym.

Słowa kluczowe: percepcja robotyczna, roboty autonomiczne, sensory robotyczne, robotyka, nawigacja, filtracja danych

Konrad Cop, MSc Eng.
konrad.cop@unitedrobots.co
ORCID: 0000-0001-8159-9307

He is a Technology Lead United Robots and a PhD student at Warsaw University of Technology. He received Bachelor of Control Engineering and Robotics from Wrocław University of Science and Technology and Master of Robotics, Systems and Control from ETH Zurich. He was a researcher in Autonomous Robots at CSIRO in Brisbane, Australia and in robotic manipulation at TUM in Munich, Germany. His experience combines a mixture of scientific research and applied development activities. His research interests include Autonomous Robotics, Application of Deep Learning to Robotics Perception and general AI topics in Mobile Robots.

Morteza Haghbeigi, MSc Eng.
morteza.haghbeigi@unitedrobots.co
ORCID: 0000-0002-8554-508X

He is an Autonomous Navigation Engineer at United Robots and a Ph.D. student at Warsaw University of Technology. He holds a Master's degree in Mechanical Engineering from the Iran University of Science and Technology. His research focuses on cooperative guidance methods, motion and path planning for both single and multi-robot systems, as well as optimization techniques using co-evolutionary methods. He also has practical experience as a robotics developer in industrial applications, where he has designed and implemented autonomous planning and navigation systems.

Marcin Gajewski, MSc Eng.
marcin.gajewski@unitedrobots.co
ORCID: 0009-0005-2593-4808

He is a Team Leader of the Architecture and Stability Team at United Robots. He received Bachelor of Control Engineering and Robotics and Master of Computer Science from Warsaw University of Technology. Additionally received Master of E-Business from Warsaw School of Economics. His experience combines applied development activities and project management. His research interest include Autonomous Robotics, Robotics Perception and Robotics for Space Exploration.

Prof. Tomasz Trzciński, DSc PhD Eng.
tomasz.trzcinski@pw.edu.pl
ORCID: 0000-0002-1486-8906

He is a Professor at Warsaw University of Technology, where he leads a Computer Vision Lab. He was an Associate Professor at Jagiellonian University of Krakow in years 2020-2023, a Visiting Scholar at Stanford University in 2017 and at Nanyang Technological University in 2019 and 2023. He worked at Google (2013), Qualcomm (2012) and Telefonica (2010). He is a Senior Member of IEEE, member of ELLIS Society and director of ELLIS Unit Warsaw, member of the ALICE Collaboration at CERN and an expert of National Science Centre and Foundation for Polish Science. He is a Chief Scientist at Toploox.