

Situational Graphs for Robotic First Responders: an application to dismantling drug labs

W.J. Meijer[§], A.C. Kemmeren[§], J.M. van Bruggen, T. Haije, J.E. Fransman, J.D. van Mil

Abstract—In this work, we support experts in the safety domain with safer dismantling of drug labs, by deploying robots for the initial inspection. Being able to act on the discovered environment is key to enabling this (semi-)autonomous inspection, e.g. to open doors or take a closer at suspicious items. Our approach addresses this with a novel environmental representation, the Behavior-Oriented Situational Graph, where we extend on the classical situational graph by merging a perception-driven backbone with prior actionable knowledge via a situational affordance schema. Linking situations to robot behaviors facilitates both autonomous mission planning and situational understanding of the operator. Planning over the graph is easier and faster, since it directly incorporates actionable information, which is critical for online mission systems. Moreover, the representation allows the human operator to seamlessly transition between different levels of autonomy of the robot, from remote control to behavior execution to full autonomous exploration. We test the effectiveness of our approach in a real-world drug lab scenario at a Dutch police training facility using a mobile Spot robot and use the results to iterate on the system design.

I. INTRODUCTION

Illegal drug labs pose a difficult problem for the Dutch society and police force. The labs are increasingly found in urbanized areas, posing a high risk to local residents, and dumped drug waste has devastating effects on the surrounding nature¹. Naturally, quick and safe dismantling of these labs is a priority. Various toxic or explosive chemicals and even suspects can be present in the labs, making dismantling a dangerous and complex mission. It is therefore important that the human specialists carefully plan these dismantling efforts since oversights have previously led to (severe) injuries of the staff. A potential solution to improve the safety of operation is to use a robotic system for the initial inspection of the drug lab (Figure 1). As illustrated in Figure 2, robots could autonomously explore the lab and build a situational awareness - incl. detection of dangerous substances - which could then be used by the police staff to make a plan to properly dispose of any chemicals.

Unfortunately, commercially available robots are currently not yet suitable to support the police in this. Companies providing state-of-the-art solutions focus on using autonomous robot platforms for routine inspection tasks: first, an operator records a mission in a known environment (e.g. a factory) through remote-control, which the robot can then replay².

^{*}This work is supported by TNO and Dutch National Police.

[§]These two authors contributed equally to this work.

¹<https://open.overheid.nl/documenten/50b66102-48f4-4afe-a2ad-bbe034896e2f/file> (Dutch)

²www.bostondynamics.com/case-studies/energy-savings-at-ab-inbevs-largest-european-brewery

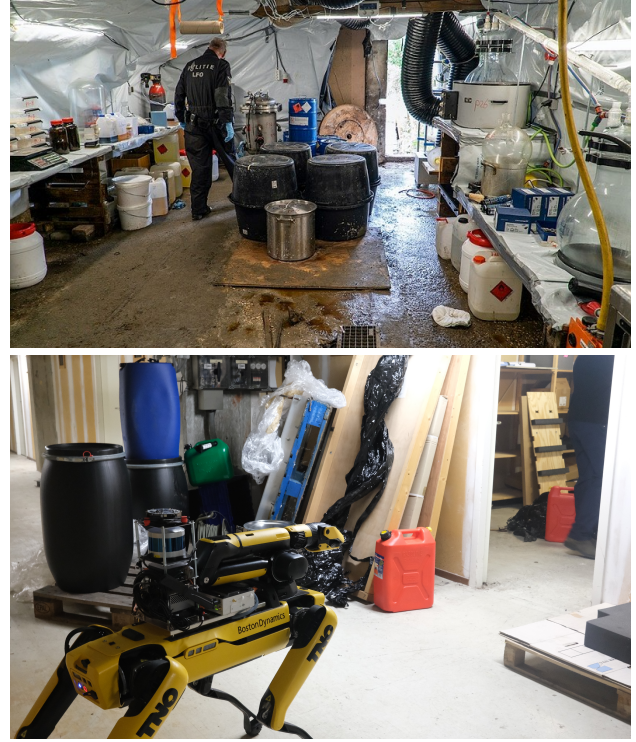


Fig. 1: Top: photo from a real drugslab (source:politie.nl). Bottom: Robot in operation during experiment in mock-up drugslab.

Inspecting a drug lab for safe dismantling does not fit this set-up, since the main aim is gaining situational awareness by the exploration of an unknown environment as opposed to performing the same known job instances routinely.

The main challenge to deploy autonomous robotic systems for exploration of drug labs is to have a predictable and reliable system in an unknown environment. This requires a design where a human operator has insight into and control over the behavior of the robotic system at all times. We propose to use the Behavior-Oriented Situational Graph to achieve these goals. The graph can be updated and extended in real-time upon exploration of new areas, while intuitively presenting any salient information in the environment such as tracked objects and traversability of the terrain. Moreover, it provides a representation that human operator and (autonomous) planner can simultaneously use to determine what behavior can be executed next. This allows for seamless handover between different levels of autonomy, from immersive remote-control to selecting specific autonomous behaviours (e.g. opening a door) to fully autonomous exploration.

This article shows our proposed framework, with the following contributions:

- 1) Initial design and formalization of a versatile environmental representation data structure that is directly actionable for robots and human operators, the Behavior-Oriented Situational Graph;
- 2) The process for real-time creation and adaptation of this Situational Graph from data, which makes it suitable for missions in unknown environments;
- 3) A workflow for operator interaction with this Situational Graph during teleoperation to build situational awareness in high-stakes missions where the human should stay in control;
- 4) Implementation of autonomous exploration and exploitation including the execution of behaviors, through algorithms that solve job selection and planning problems over the Behavior-Oriented Situational Graph;
- 5) A workflow where the shared control between human and autonomy is facilitated, by expressing missions in terms of the Situational Graph.

In collaboration with the Dutch police force, we are actively testing and iterating on this design. Using an initial version of the system, we gather feedback from potential users in a field test in a mock-up drug lab, resulting in iteration on requirements and suggested improvements.

II. RELATED WORK

The DARPA SubT challenge³ spurred applied scientific research on building situational awareness in challenging unknown environments [1]. The focus was to aid the situational awareness of an operator by reconstructing a 3D representation of the environment and localizing objects with decimeter accuracy in environments spanning kilometers. However, where the DARPA SubT challenge solely considered ‘passive’ exploration, new challenges arise when collecting information through active interaction with the environment. The following sections consider different methods to support active interactions, with methods that determine which actions can be executed where, and system designs that allow for adjustment of the level of autonomy.

A. Actionable Environment Representation

Environment representations have been steadily improving in terms of information richness and the situational awareness they provide both for the robot itself and human operators [2]. The most prominent related field is Simultaneous Localization and Mapping (SLAM), which builds geometric representations of the environment [3]. Current state-of-the-art representations extend regular SLAM pipelines to form their Situational Graphs [4]. These capture walls as conceptual geometric features and combine them to arrive at a conceptual understanding of rooms and floors which make up buildings. Other methods such as 3D scene graphs (3DSG) [5] include relations between objects but don’t

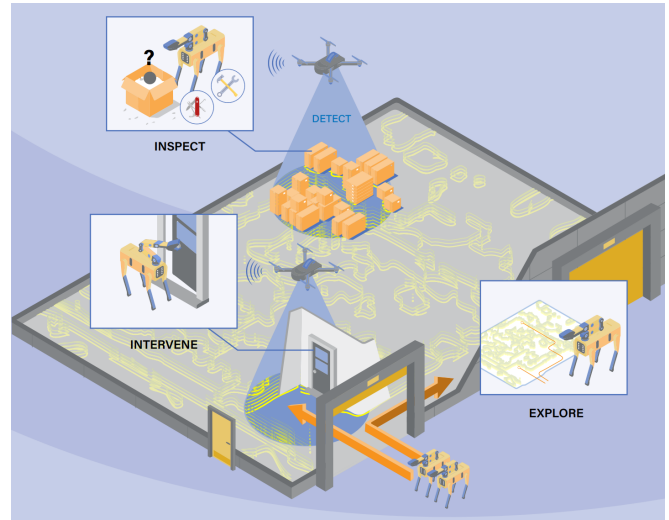


Fig. 2: The envisioned usage of robots in a drug lab exploration scenario, including improvements identified in this formative study. Multiple robots collaborate to explore the environment, intervene on obstacles such as closed doors, and execute mission-relevant jobs such as inspection.

explicitly model walls, rather relying on clustering of nodes to form rooms. However, the high dimensionality of these representations often make them unusable for planners. Even for simple tasks, the computational burden explodes with the size and complexity of the scene [6].

Instead, other works focus on sparser graphs to encode actionable information, e.g. Grey, Liu, and Ames [7] dynamically constructed a *possibility graph*, where they solely focused on low level actions for traversing environments. Moreover, Chen et al. [8] connected the observed environment to higher level navigational behaviors such as “follow the corridor” and “take a right turn”. Loo, Hsu, and Lab [9] extended this by proposing behavior graphs that map these distinct navigation behaviors in an environment. Here, they use *navigational affordances* to refer to the system seeing which behaviors are relevant in the environment.

Along the same lines as these studies, we propose a robot-centric graph that encodes behaviors and ignores object-object relations to simplify the building and use of the graph. Similarly, we draw inspiration from 3DSG and add objects to the graph to make it more diversely applicable, and enable us to enrich the graph with detected affordances [10].

B. Adjustable Level of Autonomy

Depending on the complexity of the task, environment, and system, the operator may have varying levels of trust in the autonomous capabilities of the robot [11]. This highlights the importance for an operator to control or aid the system during operation by interacting at varying levels. For example, [12, 13] use human supervision for a high-level task selection problem, where the human operator is able to edit the selected tasks. Additionally, [14] supervises low-level motion control directly. When engaging in teleoperation of the system, it is beneficial to provide an immersive and

³<https://www.darpa.mil/program/darpa-subterranean-challenge>

intuitive experience for the operator [15, 16]. Transferring sensory modalities such as vision, audio or force feedback give the operator a feeling of being in control and contributes to the effectiveness of the operator [17]. Moreover, studies have also investigated how to integrate these different levels of human supervision to make a system where the level of autonomy is adjustable [18, 19]. These works inspired us to create a traditional 2D view as the interface to interact with the system at a high level and adjust between levels of autonomy, and an immersive mixed reality view for intuitive teleoperation.

III. PROBLEM STATEMENT

We consider the exploration of a large unknown area. A human operator stands remotely, and is in charge of a robot. We address this challenge by dividing it into the following three sub-problems:

- The robot perceives the environment and records this information in a Behavior-Oriented Situational Graph.
- The graph stores actionable information (i.e. affordances), which the planning layer uses to autonomously execute behaviors.
- This Situational Graph is presented to the operator in a graphical user interface (GUI), providing information about the mission in real-time and allowing the operator to control the mission.

We elaborate on these three sub-problems in the following sections and from now on we refer to the Behavior-Oriented Situational Graph, simply as Situational Graph.

A. Situational Graph

Let us first consider the problem of constructing a Situational Graph from robot perception. We define the graph as a directed multigraph $\mathcal{G}$, of which a schematic view is shown in Figure 3.

$$\mathcal{G} := (\mathcal{V}, \mathcal{E}), \quad (1)$$

where $\mathcal{V}$ is defined as the set of nodes and $\mathcal{E}$ as the set of edges. Every node $v \in \mathcal{V}$ represents a place and consists of a pose $\mathbf{p}_v \in SE(2)$ in the ground plane and a situation s_v . Here, $SE(2)$ is the 2-dimensional Special Euclidean group.

$$v := (\mathbf{p}_v, s_v), \quad v \in \mathcal{V}. \quad (2)$$

For each node, situation s_v stores data specific to a location, which is relevant to determine possible behaviors. In our definition, the situation contains an $m \times n$ local gridmap $M_v \in \mathbb{R}^{m \times n}$ and a set of objects $\mathcal{O}_v$ that are currently present at place v .

$$s_v := (M_v, \mathcal{O}_v) \quad (3)$$

$$o := (\text{id}_o, l_o, \mathbf{p}_o), \quad o \in \mathcal{O}_v,$$

where object o has a unique identifier id_o , a label of the object type l_o (e.g. *door* or *person*), and pose $\mathbf{p}_o \in SE(3)$.

A directed edge $e_{ij} \in \mathcal{E}$ connects two nodes v_i and v_j iff some behavior $b \in \mathcal{B}$ allows the robot to transition from v_i to v_j . In other words, v_i implicitly encodes behavior pre-conditions, and v_j the post-conditions. Multiple edges

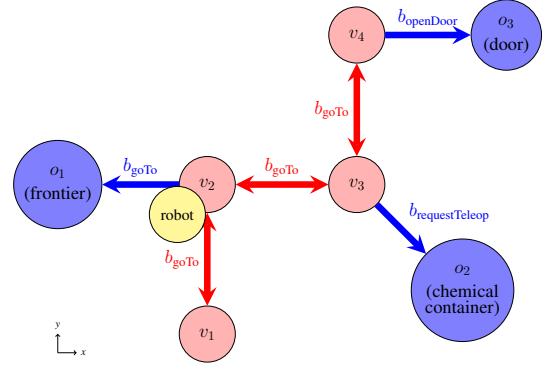


Fig. 3: An example of how the Behavior-Oriented Situational Graph $\mathcal{G}$ encodes potential behaviors b_k in the environment. Conventional edges e between two nodes are visualized in red. Behaviors that take world objects as parameters are visualised by the blue arrows, connecting a node v with a world object o . The robot can e.g. goTo and explore a frontier (o_1) at v_2 , open a door (o_3) at node v_4 or request the operator for assistance on a chemical container (o_2) from node v_3 .

are allowed between two nodes v_i to v_j , since there can be multiple behaviors b_k that have the same pre- and post-conditions. $\mathcal{O}_i^k$ is the set of objects at v_i that are relevant for the execution of b_k . e_{ij}^k is therefore defined as

$$e_{ij}^k := (b_k(\mathcal{O}_i^k), v_i, v_j), \quad e_{ij}^k \in \mathcal{E}, \quad v_i, v_j \in \mathcal{V}. \quad (4)$$

We use the notion of affordance h to find the behavior that corresponds to an edge, which we define as a mapping

$$h : v_i \rightarrow b_k, v_j. \quad (5)$$

Examples include detecting where new area's can be explored, or detecting that the environment can be changed by opening a door as shown in the top right of Figure 3.

Thus, based on the local gridmap M_{v_i} and the objects $\mathcal{O}_{v_i}$ present in the local environment of v_i , the robot infers which behaviors b_k are possible, and to what resulting desirable situation v_j this will lead. This way, we effectively encode one-step look-ahead planning into the graph.

B. Planning Layer

The path planning problem concerns how some robot can reach node v_j from starting position v_i . If the robot starts from node v_i , the optimal plan π_{ij}^* to reach node v_j is simply the shortest path between v_i and v_j in the graph.

$$\pi_{ij}^* := \arg \min_{\pi_{ij}} \sum_{e \in \pi_{ij}} w_e, \quad \pi_{ij} \in \mathcal{P}_{ij} \quad (6)$$

where w_e is the cost of executing the behavior corresponding to edge e , and $\mathcal{P}_{ij}$ is the set of all (acyclic) paths between nodes v_i and v_j in graph $\mathcal{G}$. A plan is thus an ordered sequence (i.e. a path) of directed edges, encoding consecutive behaviors for the robot to traverse the graph.

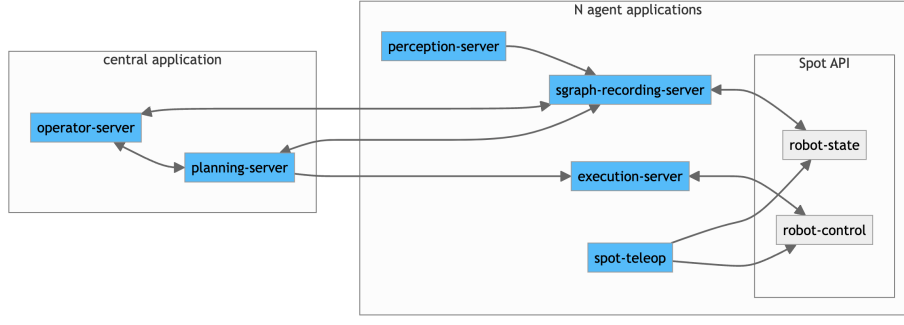


Fig. 4: A simplified overview of the centralized architecture of the system. In the agent app running on the backpack PC a perceptions server publishes perception events from sensor data, which the sgraph-recording-server uses to update the Situational Graph. On a central app outside of the drug lab, the operator-server hosts the graphical user interface for the operator and allows the operator to select the level of autonomy. Based on this level, the planning-server and execution-server allocate and execute jobs respectively. Finally, when in teleop mode, the spot-teleop server allows direct communication with the spot API to improve teleop latency.

The job selection problem concerns selecting the optimal job τ^* given the agent's current node v_i . We define the optimal job to be selected as

$$\tau^* = \arg \max_{\tau} R(\tau) - C(\tau, v_i) \quad (7)$$

where $R(\cdot)$ is the reward function and $C(\cdot)$ is the cost function for the robot at v_i to reach the selected node τ . The cost C to execute the job is the same as the cost of the optimal plan π_{ij}^* , given v_i is the current place of the robot, and $v_j = \tau$ the allocated node.

The proposed system then allows for easy switching between the following levels of autonomy:

- 1) Full autonomy with job selection and path planning
- 2) Job selection by human operator, but autonomous path planning
- 3) Human operator directly selects a behavior to execute that is available at the current node
- 4) Motion directly remotely controlled by the operator

C. Human Operator

The situational graph is presented to the operator in a GUI, with which we aim to achieve three goals: (1) facilitate the situational awareness of the operator, (2) provide the operator with insight into the expected mission progress via visualization of the job assignment, plans for job execution and behavior execution, and (3) presenting an intuitive interface to adjust the level of autonomy of the robot, switching from e.g. immersive tele-operation, to execution of a specific behavior or to full autonomous exploration.

These goals have been identified based on conversations with stakeholders, and experience with similar projects in TNO's department of human machine teaming.

IV. METHODS AND ALGORITHMS

In this section, an initial implementation of the Situational Graph framework is presented. To do this we first describe the servers for perception, Situational Graph recording, planning, behavior execution, operator interaction, and immersive

teleoperation. The overall system architecture is outlined in Figure 4. The operator and planning server run centrally, while the rest run on the PC in the robot's backpack.

A. Perception Server

The perception server processes camera, IMU and LiDAR data in real-time to detect objects o , update the node locations p_v and produce gridmaps M .

From video data, fiducials can be detected with highly accurate pose estimates and unique IDs. Moreover, a YOLOv8 model [20] is able to detect a range of objects as defined in the COCO dataset [21]. YOLOv8's tracking capability is used to update object locations p_o in real-time. As of yet, positions of the YOLOv8 detections are limited to image coordinates, and do not yet have 3D pose estimates.

In parallel, data from a LiDAR and IMU are processed by the SLAM algorithm LIO-SAM [3]. This algorithm updates the robot pose used to update the node locations p_v and processes 3D pointclouds to maintain a voxel map of the environment. These pointclouds are also converted to local 2D gridmaps M_v by using Octomap [22]. A constructed global 2D map is shown in Figure 8.

B. Situational Graph Recording

The Situational Graph Recording server has an *observer module* and *prediction module* to update the graph nodes and edges. The observer module receives perception events from the Perception Server. The module uses these events to edit the graph by storing the latest gridmaps and object detections in the nodes and re-evaluating the validity of the edges.

Furthermore, the robot uses the most recent estimate of its position to determine whether it is in a new place $v_{new} \notin \mathcal{V}$. Here, a place is considered new if it is located > 2 m from the nearest node. Then, a new node is added to the graph and connected by an edge e_{ij}^k , where behavior b_k describes how the robot transitioned from v_i to the new node v_j .

Then, the prediction module uses the recorded situation data and pre-defined affordances to further refine the graph, by detecting if a robot can execute any behaviors. However,

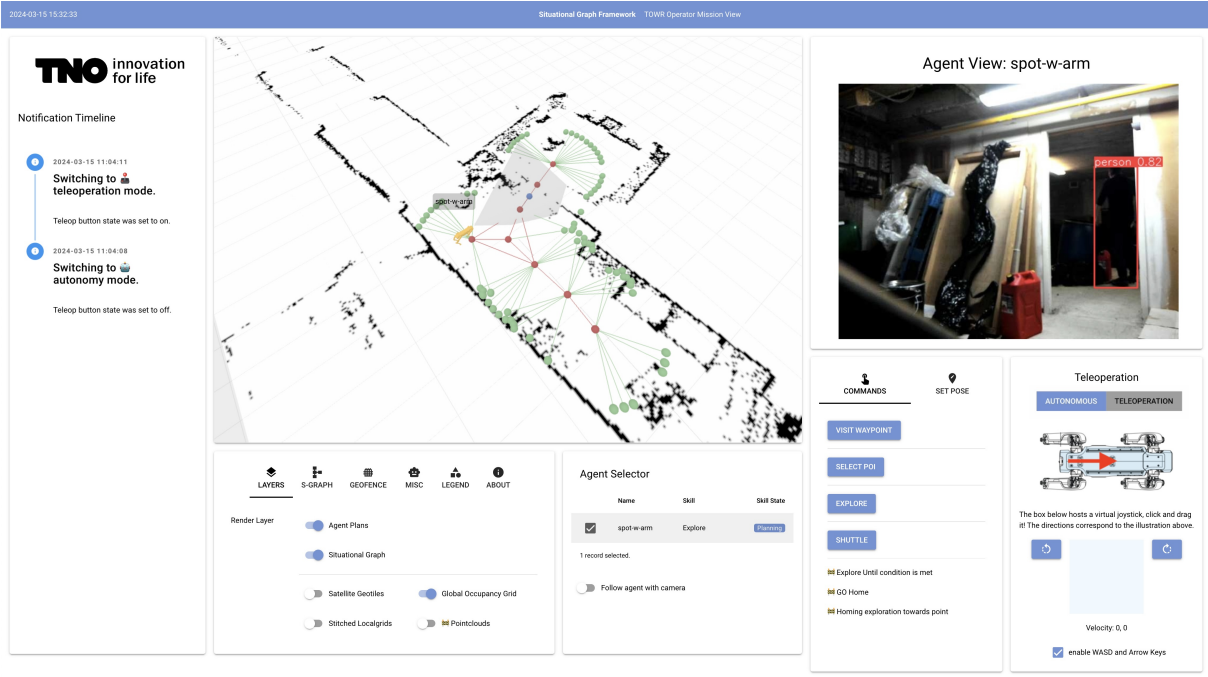


Fig. 5: The operator interface. The 3D scene in the center shows the Situational Graph. Red nodes are waypoints visited by the robot, green nodes are unexplored frontiers and blue nodes are detected objects. Detections are represented by a unit vector from the position where the detection was done towards the detected object. The left side shows a timeline of events that occurred during the mission. The bottom elements house general options for the system, such as toggling layers on and off. The right side shows robot-specific elements such as the live stream (with detections), commands, and basic teleoperation controls.

this is limited to the behaviors and affordances defined before the mission. In our implementation, we can execute the following behaviors:

- b_1 : Go to node
- b_2 : Open a door
- b_3 : Request tele-operation from the human operator

Moreover, we have defined the following affordances h to map situations at node v_i to potential behaviors b_k :

- h_1 : If terrain in M_v is traversable and leads to unknown areas, execute b_1 to the frontier;
- h_2 : If a closed door is detected at some node, execute b_2 ;
- h_3 : If a human is detected, execute b_3 ;
- h_4 : If a container is detected, execute b_3 .

Affordance h_1 represents an important capability of our module, based on a frontier exploration algorithm. It checks whether any of the local grids afford traversable ground to go to unexplored areas. In those locations, new frontier nodes $\mathcal{V}_f \subset \mathcal{V}$ are sampled, while existing frontiers in recently explored areas are pruned.

C. Planning Server

The Planning Server considers two optimization problems: finding the best plan to execute a job (Equation (6)), and optimizing selection of the optimal job itself (Equation (7)).

The path planning problem in Equation (6) is solved using

Dijkstra's algorithm in NetworkX⁴. Furthermore, we define some positive reward R for any frontier nodes $\mathcal{V}_f \subset \mathcal{V}$ such that the job selection problem in Equation (7) reflects an exploration problem.

It then depends on the selected level of autonomy (subsection III-B) which of these problems are actively being solved. Both problems are optimized when the robot is in the highest level of autonomy, where it explores the environment without any human input. One level lower, the human operator selects the jobs, and the planning server only needs to consider the path planning problem. For the lowest two levels of autonomy, the planning server is inactive.

D. Behavior Execution Server

In the highest three levels of autonomy (subsection III-B), the Behavior Execution Server can be called to execute the following behaviors: go to a node, open a door, or request teleoperation from a human operator.

For the goTo node behavior a navigation stack is used, i.e. the Boston Dynamics navigation stack⁵ for Spot. The behavior to open a door is currently only implemented for Spot, using a combination of the native Boston Dynamics

⁴https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html

⁵https://dev.bostondynamics.com/docs/concepts/autonomy/autonomous_navigation_services

API⁶ and custom detection of door handles and hinges. Lastly, the tele-operation request simply hands the control of the robot over to the operator.

E. Operator Interface

The operator interacts with the system using a web interface in the browser as shown in Figure 5. The view is split into generic elements on the left and robot-specific elements on the right. Various layers can be toggled on and off in this 3D view, for example specific node types of the Situational Graph, a global 2D occupancy grid and plan visualizations. Also, the visualized Situational Graph allows the human to interact with the system on a high level, since e.g. a job can be re-allocated by clicking on one of the nodes in the graph.

F. Immersive Teleoperation

Whenever the operator feels the necessity to intervene, or when the perception server detects an object or situation where intervention is required, the system can switch to tele-operation mode. For this, the robot is equipped with a stereo pair of 180 degree fisheye lenses creating depth vision for the operator on the base of the arm. The two video streams are rendered through a Head Mounted Display (HTC Vive Pro Eye, HTC Corporation, Taiwan). The camera inputs are displayed through the Unity game engine on half domes matching the shape of the lens to compensate for distortion. The arm is controlled through end-effector position control. Input for the arm control is gathered with the VR controller (HTC Vive Pro Eye). Locomotion of the robot is controlled through a custom foot pedal, measuring the weight distribution (towards toes, or towards heels) and rotation of the operator feet. Figure 6 gives an impression of the operator station and the first person VR view.

V. EXPERIMENTAL SET-UP

In order to iterate with our stakeholders on the system requirements, an experiment was devised. This section de-

⁶https://dev.bostondynamics.com/docs/concepts/arm/arm_services.html#door-service



Fig. 6: The setup for teleoperation. On the left the operator with VR goggles and VR controller, and his feet on the haptic plate. On the top right the view as the operator sees it. The bottom right shows the teleoperated robot.

scribes the experimental robot platform and the setting of the experiment.

A. Experimental Platform

The experimental robotic platform is a Spot robot with arm and a custom backpack, as shown in Figure 7. The backpack comprises a 4G-router (HMS), A LiDAR Velodyne VLP-16), IMU (microstrain), a microphone array (ReSpeaker) and a Nvidia Jetson PC (Jetson AGX Orin). At the base of the arm, two fisheye cameras (ELP) are mounted for use during teleoperation. Figure 4 shows an overview of the software architecture, and on which hardware the two applications run.

B. Experiment for Formative Evaluation of the Complete System

An experiment was conducted at a mock-up drug lab normally used for training of the Dutch Police. The goal was twofold: testing the initial version of the system in a relevant environment and evaluating it with potential end users.

For the experiment, the participants from the Dutch Police were tasked with building situational awareness of the area and placing any detected dangerous objects into a safe container. Specifically, the mission consisted of the following tasks:

- 1) Supervise the system while it autonomously searches for the 3 hazardous situations in the environment.
- 2) Perform 2 immersive tele-operation interventions. A bottle representing a "toxic chemical" should be manipulated and placed securely in the waste box.

The total number of hazardous situations and required interventions were unknown to the police officers.

A map of the experimental environment used is shown on the left in Figure 8. It represents an unknown indoor cluttered environment, roughly 15x20 meters in size and included a separate room connected by a door already opened. The remote control station is envisioned to be at a distant location (e.g. police van or police control room), however,

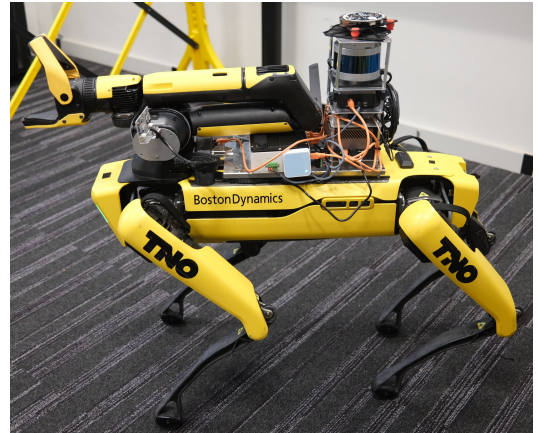


Fig. 7: The experimental platform is a Spot robot with arm equipped with a custom backpack.

for practical reasons in this experiment it was placed nearby in the same room.

The group of participants consisted of four officers from the Dutch National Police working in the discovery and dismantling of drug labs. The participants received a short 5 minute instruction on the controls of the tele-operation setup and an short explanation of the user interface before the start of the experiment.

VI. EXPERIMENTAL RESULTS

This section will describe the results of the formative study we conducted together with the stakeholders. The experimental results contribute to an iteration of the system requirements. We focus especially on how the Situational Graph should be adapted to optimally connect to the new requirements.

A. Results Formative Evaluation with Stakeholders

Feedback on the demonstrated system was overall positive. The envisioned task of building situational awareness was supported by the autonomous exploration and mapping and the task of placing dangerous objects in a safe container was supported by the identification of these objects and switching to tele-operation mode. Operators appreciated the approach of adjustable levels of autonomy in itself, as it improved operator trust in tackling the dynamic environments in which they work by providing them with a means to act as a fallback for the robot, and seize control if needed.

Even during a fully tele-operated mission with the system, the operators experienced benefits of the passive autonomy services running in the background:

- The green frontier nodes helped in spotting missed and undiscovered rooms.
- When reaching the end of a room the operators could rely on the autonomous navigation to move the system to the start of an unexplored room or back to the start,

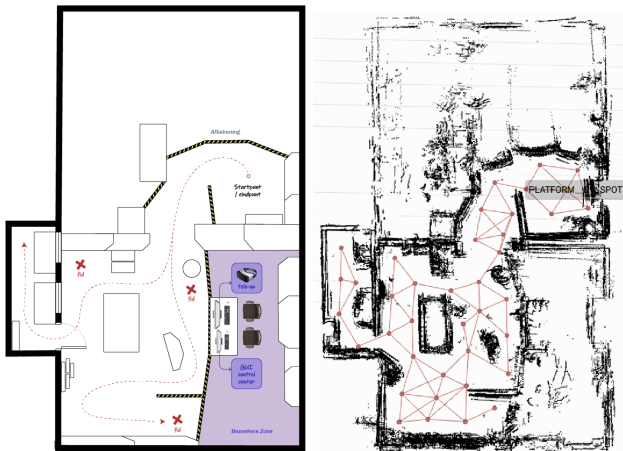


Fig. 8: Schematic drawing of the mock-up drug lab on the left. The resulting environment representation of the test location on the right.

rather than having to control the system there manually, this saved mental overhead.

- When experiencing communication issues they could rely on the system to autonomously return back into communication range.

They did note that the general approach of “autonomy first, teleoperation second” was a big step compared to their current usage of robots, and felt more comfortable with a “teleoperation first, autonomy second” approach.

VII. CONCLUSION

Through the multidisciplinary process presented, we can iterate on robotic solutions for challenging unknown environments together with stakeholders in the national security domain. We have shown that Situational Graphs are powerful representations for storing actionable information. They are interpretable for humans while for robots they are directly usable for planning. By perceiving and storing actionable elements in the environment the used planning methods can simply be graph search, allowing the planning to occur at a high frequency while scaling to extensive areas of operation. Moreover, the representation allows the human operator to easily adjust the level of autonomy, and possibly switch to immersive teleoperation.

A. Future Work

The early and continuous evaluation of the system provides directions for future extensions of the Situational Graph. Given the experimental results, a number of requirements were distilled for a subsequent development iteration.

- The robot should be able to pinpoint detected objects relevant to a specific mission in the graph.
- The robot should have specific execution modes for exploration: faster exploration, or slower but thorough.
- The robot framework should support deploying multiple robots that coordinate their efforts.
- The robot framework should work with robots of differing type and make (interoperability).
- The robot should be operated in an intuitive manner.

REFERENCES

- [1] Harel Biggie et al. “Flexible Supervised Autonomy for Exploration in Subterranean Environments”. In: *Field Robotics* 3.1 (Jan. 2023), pp. 125–189. ISSN: 27713989. DOI: 10.55417/fr.2023004. arXiv: 2301.00771 [cs]. (Visited on 07/27/2023).
- [2] Hriday Bavle et al. “From SLAM to Situational Awareness: Challenges and Survey”. In: *Sensors* 23.10 (Jan. 2023), p. 4849. ISSN: 1424-8220. DOI: 10.3390/s23104849. (Visited on 05/26/2023).
- [3] Tixiao Shan et al. *LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping*. July 2020. DOI: 10.48550/arXiv.2007.00258. arXiv: 2007.00258 [cs]. (Visited on 03/15/2024).
- [4] Hriday Bavle et al. “Situational Graphs for Robot Navigation in Structured Indoor Environments”. In: *arXiv:2202.12197 [cs]* (Feb. 2022). arXiv: 2202.12197 [cs]. (Visited on 05/03/2022).

- [5] Antoni Rosinol et al. *Kimera: From SLAM to Spatial Perception with 3D Dynamic Scene Graphs*. Jan. 2021.
- [6] Christopher Agia et al. “TASKOGRAPHY: Evaluating Robot Task Planning over Large 3D Scene Graphs”. In: *ok* (), p. 20.
- [7] Michael X. Grey, C. Karen Liu, and Aaron D. Ames. “Traversing Environments Using Possibility Graphs with Multiple Action Types”. In: *arXiv:1610.00701 [cs]* (Oct. 2016). arXiv: 1610.00701 [cs]. (Visited on 02/03/2022).
- [8] Kevin Chen et al. “A Behavioral Approach to Visual Navigation with Graph Localization Networks”. In: *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN: 978-0-9923747-5-4. DOI: 10.15607/RSS.2019.XV.010. (Visited on 10/26/2021).
- [9] Joel Loo, David Hsu, and Adacomp Lab. “Behaviour Graphs: A Semantic-Contextual Representation for Robot Navigation”. In: ().
- [10] Gen Li et al. *One-Shot Open Affordance Learning with Foundation Models*. Nov. 2023. DOI: 10.48550/arXiv.2311.17776. arXiv: 2311.17776 [cs]. (Visited on 03/18/2024).
- [11] Phillip Jeff Durst and M. Wendell Gray. *Levels of Autonomy and Autonomous System Performance Assessment for Intelligent Unmanned Systems*. Report. Geotechnical and Structures Laboratory (U.S.), Apr. 2014. (Visited on 03/18/2024).
- [12] Julie A. Adams, Joshua Hamell, and Phillip Walker. “Can A Single Human Supervise A Swarm of 100 Heterogeneous Robots?” In: *Field Robotics* 3.1 (Jan. 2023), pp. 837–881. ISSN: 27713989. DOI: 10.55417/fr.2023026. arXiv: 2308.00102 [cs]. (Visited on 03/18/2024).
- [13] Giada Galati, Stefano Primatesa, and Alessandro Rizzo. “Auction-Based Task Allocation and Motion Planning for Multi-Robot Systems with Human Supervision”. In: *Journal of Intelligent & Robotic Systems* 109.2 (Sept. 2023), p. 24. ISSN: 1573-0409. DOI: 10.1007/s10846-023-01935-x. (Visited on 03/18/2024).
- [14] Mario Selvaggio et al. “Autonomy in Physical Human-Robot Interaction: A Brief Survey”. In: *IEEE Robotics and Automation Letters* 6.4 (Oct. 2021), pp. 7989–7996. ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3100603. (Visited on 03/18/2024).
- [15] Jeanine van Bruggen et al. “I-Botics Avatar System: Towards Robotic Embodiment”. In: ().
- [16] Jan B.F. Van Erp et al. “What Comes After Telepresence? Embodiment, Social Presence and Transporting One’s Functional and Social Self”. In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2022, pp. 2067–2072. DOI: 10.1109/SMC53654.2022.9945544. (Visited on 03/12/2024).
- [17] Michael Walker et al. “The Cyber-Physical Control Room: A Mixed Reality Interface for Mobile Robot Teleoperation and Human-Robot Teaming”. In: Mar. 2024, pp. 762–771. DOI: 10.1145/3610977.3634981.
- [18] Kapil D. Katyal et al. “Approaches to Robotic Teleoperation in a Disaster Scenario: From Supervised Autonomy to Direct Control”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sept. 2014), pp. 1874–1881. DOI: 10.1109/IROS.2014.6942809. (Visited on 03/15/2024).
- [19] Naresh Marturi et al. “Towards Advanced Robotic Manipulation for Nuclear Decommissioning: A Pilot Study on Tele-Operation and Autonomy”. In: *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. Dec. 2016, pp. 1–8. DOI: 10.1109/RAHA.2016.7931866. (Visited on 03/15/2024).
- [20] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLO*. Jan. 2023. (Visited on 03/15/2024).
- [21] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. Feb. 2015. DOI: 10.48550/arXiv.1405.0312. arXiv: 1405.0312 [cs]. (Visited on 03/15/2024).
- [22] Kai M Wurm et al. “Octomap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems”. In: (), p. 7.