Autonomous navigation of a tracked unmanned ground vehicle ⋆

Marija Seder, Anđela Jurić, Ana Šelek, Filip Marić, * Marija Lovrić, ** Ivan Petrović *

* *University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering, Laboratory for Autonomous Systems and Mobile Robotics (LAMOR), Zagreb, Croatia (e-mail: name.surname@fer.hr).*
** *University of Zagreb, Croatian Military Academy Dr. Franjo Tuđman, Zagreb, Croatia (e-mail: marija.lovric@morh.hr)*

Abstract: This work proposes a complete autonomous navigation system for a tracked vehicle. The system enables a complete autonomous execution of waypoint and patrolling tasks selected by the user. It also enables user-vehicle shared autonomy, switching between the user teleoperation and the vehicle autonomous operation. Our navigation system uses the model predictive control scheme based on a navigation function. We propose the navigation function which takes into account changing environments, any-shape footprint, and non-holonomic motion of the tracked vehicle. Besides the waypoint and patrolling tasks, we implemented a fail-safe scenario in case of the user-vehicle communication loss, in which the vehicle returns autonomously to the previously visited goal where the communication was stable. The efficiency of the proposed system is validated by experimental results on the Komodo tracked vehicle.

Keywords: Robot navigation, path planning, obstacle detection and avoidance, motion control, map building.

1. INTRODUCTION

Unlike human teams, robots and remotely controlled vehicles can easily access dangerous sites, remove existing and potential threats and find and extract victims. Recently, the use of the Unmanned Ground Vehicles (UGV) for military purposes raise great attention (Czarnowski et al., 2018; Nohel et al., 2020). However, these UGVs are semi-autonomous and not suitable for completely autonomous tasks.

The paper presents the autonomous navigation of the Komodo UGV made by DOK-ING company (see Fig. 1). It can sustain the conditions in the extremely hot zone, in which humans cannot survive. We propose the control architecture for the Komodo UGV, which enables user-vehicle shared autonomy, switching between the user teleoperation and the vehicle autonomous execution of the GPS-based waypoint and patrolling tasks. In our previous work (Šelek et al., 2019) we presented the autonomous execution of the waypoint and patrolling tasks on a Husky A200 mobile robot. Here, we redesign our previous control architecture for the vehicle weighing 17 tons and driving at around 4 times higher speeds. The vehicle's footprint is narrow and long, which requires a complex calculation of collision with the obstacles. Furthermore, the caterpillar tracks constrain the motion of the vehicle requiring the employment of kinodynamic planners or RRT-based high dimensional planners (Li et al., 2016; Yoon et al., 2017). However, these planners provide a solution that usually requires post-optimization and the additional trajectory tracking controller (Li et al., 2016; Klančar and Blažič, 2019). To avoid computationally intensive decoupled planning and tracking control, we propose a model predictive control (MPC) scheme based on a navigation function (Ogren and Leonard, 2005; Seder et al., 2017). We design the navigation function to take into account the non-circular footprint and non-holonomic motion. Then, we employ the navigation function as a cost function inside the model predictive control scheme.

Fig. 1. Komodo UGV made by DOK-ING company.

On top of the proposed navigation, we implemented the user interface for task assignment and supervision based on the Quantum geographic information system (QGIS). Furthermore, we developed the communication module which connects QGIS and the Robot Operating System (ROS) holding the navigation software. The communication mod-

⋆ The authors thank DOK-ING company for providing the Komodo UGV for the development and testing the autonomous navigation system.

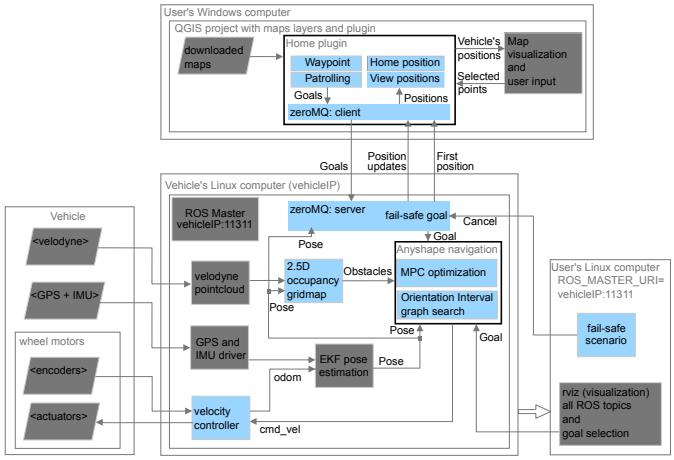


Fig. 2. Control system architecture of a tracked UGV.

ule allows multi-user task assignment and supervision via the client-server framework. Besides the waypoint and patrolling tasks, we implemented the fail-safe scenario in case of communication loss. In that case, the vehicle returns to the previously reached goal where the communication connection was permanent.

The contribution of the paper comprises of the following: i) a complete autonomous navigation system for the tracked vehicle weighing 17 tons and driving up to 4 m/s, ii) graph search suitable for non-holonomic vehicles with a non-circular footprint, and iii) task supervision with the fail-safe scenario in case of communication loss.

The paper is structured as follows: Sec. 2 describes the system architecture, Sec. 3 presents our solution for the autonomous navigation of a tracked UGV, Sec. 4 presents experimental results, and Sec. 5 concludes the paper.

2. CONTROL SYSTEM ARCHITECTURE

The control system architecture is presented in Fig. 2. The system consists of QGIS integration with ROS navigation software. Modules developed in this work are presented with blue blocks, while the gray blocks represent standard ROS packages or known inputs. All ROS software components are installed on the vehicle's Linux computer. The user's Linux computer is used to visualize all ROS topics from the vehicle's Linux computer remotely. The user's Windows computer has a QGIS user interface deployed and the communication module connecting QGIS and ROS. In the following, we shortly describe three main submodules: the autonomous navigation system deployed on the vehicle, task assignment, and supervision system deployed on the remote user computer, and the communication module deployed on both computers.

2.1 Autonomous navigation

The velocity controller built specifically for the Komodo UGV calculates the odometry by transforming the measured velocity of each caterpillar tread to the translational-rotational velocity pair in the local vehicle frame. It also transforms the control inputs in the local vehicle frame to the command velocity of each caterpillar tread. The EKF fuses odometry with the GPS and IMU data to give

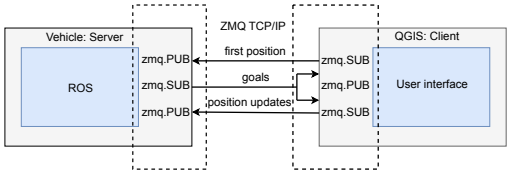


Fig. 3. Communication between the user-facing QGIS software and the vehicle.

the absolute geo-referenced position of a vehicle and a 6D position estimate in the global map frame. The mapping module continuously builds a 2.5D occupancy gridmap of the environment from Velodyne's data for fast visualization on a remote computer. This module also extracts a 2D range data of possible colliding obstacles and serves as the input for the anyshape navigation. Anyshape navigation provides the optimal control inputs for the vehicle, taking into account the vehicle's motion model and constraints on the vehicle's controls and states.

2.2 Task assignment and supervision

This module is implemented as a QGIS plugin developed in our previous work (Šelek et al., 2019). The user can select *waypoint* and *patrolling* tasks. A waypoint is a goal point in the physical world that the vehicle needs to reach. The user can select the points one by one on a map containing geo-referenced data. On the other hand, patrolling focuses on completing a whole sequence of reference points. Furthermore, the user can supervise the position of the vehicle in the map. If the vehicle does not move, the user can find out its current position through the *Home position* request. During the motion, the user can track the route of the vehicle through the *View positions* request.

2.3 Communication module

Communication between the user-facing QGIS software (client) and the ROS module on the vehicle (server) is achieved using one of the ZeroMQ messaging patterns – Publish (PUB)/Subscribe (SUB) pattern (Hintjens, 2013). This classic pattern, which is based on the TCP/IP protocol, enables the stacking of PUB/SUB sockets on the server and clients according to functional needs. In our case, two SUB sockets and one PUB socket are installed on the user side, and for compatibility, there are two PUB sockets and one SUB socket on the server-side. The data sent by the PUBs are sent to all tethered SUBs (see Fig. 3). Thus, the communication module allows multi-user connection to the vehicle.

The fail-safe scenario in case of communication loss is implemented on the vehicle's computer and can be triggered from the remote user's Linux computer. Usually, when the connection signal weakens, a delay of data from sensors (e.g. Velodyne) occurs on the remote user's computer. The module allows the user to cancel all current missions and return the vehicle to the previously visited goal. The module simply remembers the last visited goal. On user request, it cancels the current set of goals received from QGIS and sends the remembered goal to the navigation module.

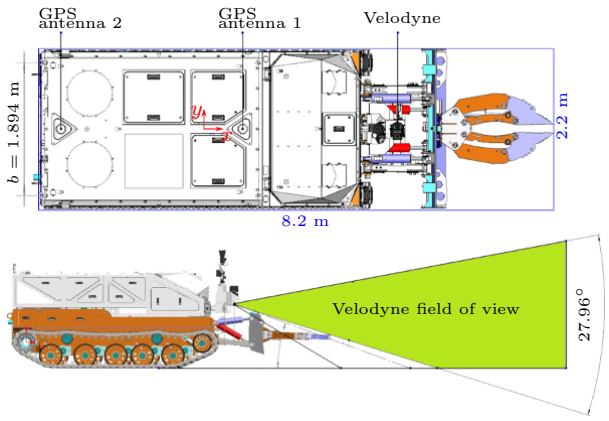


Fig. 4. The top and side view of the Komodo UGV with the position of sensors and center of rotation (x, y).

3. AUTONOMOUS NAVIGATION OF A TRACKED UNMANNED GROUND VEHICLE

This section presents autonomous navigation based on a model predictive control scheme. A prerequisite of the control scheme is a kinematic model and constraints of the vehicle which we identified through the odometry calibration procedure. Then we present the navigation function that takes into account the vehicle's rectangular footprint and holonomic constraints. Finally, we present the model predictive control scheme that integrates the vehicle motion model and the navigation function.

3.1 Kinematic model and constraints

The Komodo UGV drives on caterpillar tracks, which velocities are accessible over CAN bus. To provide reliable odometry, we developed a ROS driver which decodes and encodes CAN messages received from and sent to the tracks' motors, and implements the kinematic model of the UGV. The Komodo UGV can be described with a differential drive kinematic model, where the translational velocity v in its forward direction (x axis in Fig. 4) is the mean value of the left and right track velocities, v_L, and v_R, while rotational velocity ω around the vehicle's vertical z axis is determined as the difference in the right and left track velocity divided by the distance b between tracks:

$$\begin{aligned} v &= \tfrac{v_L+v_R}{2}, \\ \omega &= \tfrac{v_R-v_L}{b}. \end{aligned} \tag{1}$$

To take into account uncertainty in the center of rotation, wheel skid and small errors in left and right track velocities, we used the extended kinematic model with three parameters for calibration (Ivanjko et al., 2007):

$$\begin{aligned} v &= \tfrac{c_L \cdot v_L + c_R \cdot v_R}{2}, \\ \omega &= \tfrac{c_R \cdot v_R - c_L \cdot v_L}{c_b \cdot b}, \end{aligned} \tag{2}$$

where parameters c_L and c_R compensate the error in left and right track velocities, and c_b compensates the error of the center of rotation. These parameters can be found by performing two special trajectories: straight motion and in-place rotation. Optimization computes parameter values that minimize the pose and orientation error of the last point between the simulated trajectory and the real trajectory. We repeated the calibration experiment 10 times and obtained the average values $c_L = c_R = 1$ and $c_b = 1.5$.

By integrating the velocities in (2) over time, we write a discrete kinematic model of the vehicle:

$$\begin{aligned} x(k+1) &= x(k) + v(k)T\cos(\theta(k+1)), \\ y(k+1) &= y(k) + v(k)T\sin(\theta(k+1)), \\ \theta(k+1) &= \theta(k) + \omega(k)T, \end{aligned} \tag{3}$$

where T is the sampling time, $x(k)$, $y(k)$ and $\theta(k)$ are vehicle position and orientation at discrete time k.

According to the differential model it is not possible to achieve the maximal translational and the maximal rotational velocity of the vehicle at the same time. For example, the maximal translational velocity is achieved when the rotational velocity is zero, and vice versa. It is straightforward to derive the following constraint on the velocities:

$$\begin{aligned} \tfrac{|v(k)-v(k-1)|}{T} \le a_{\max}\ ,\ \tfrac{|\omega(k)-\omega(k-1)|}{T} \le \alpha_{\max}, \\ |\omega(k)| \le \omega_{\max}\ ,\ |v(k)| < -\tfrac{v_{\max}}{\omega_{\max}} \cdot |\omega(k)| + v_{\max}, \end{aligned} \tag{4}$$

where $v_{\max}$, $\omega_{\max}$, $a_{\max}$, $\alpha_{\max}$ are maximal translational and rotational velocities and accelerations of the vehicle.

3.2 Navigation function

Denote $\mathcal{S} \subset \mathbb{R}^3$ a bounded set of UGV states, also called the configuration space, where $\mathbf{s}_G = [x_G, y_G, \theta_G]^T \in \mathcal{S}$ is the goal state. The navigation function $N : \mathcal{S} \to [0, \infty\rangle$ is used to safely control the vehicle towards locations where N decreases, where the minimum is at the goal, $N(\mathbf{s}_G) = 0$. The easiest way to create the navigation function is to employ the graph search algorithm such as D* and use the calculated cost-to-goal values as the navigation function values. Such a discrete navigation function needs to be modified to obtain a unique value for any configuration inside the sampled search state (Ogren and Leonard, 2005).

A graph is created by sampling the configuration space of the vehicle, where each sample represents a node of the graph, while edges are defined between two neighbor (close) samples. The configuration space is easy to compute if the vehicle's footprint is escribed by a circle. Then, all obstacles in the environment need to be enlarged for the radius of the circle and what remains free is the configuration space.

A rectangular footprint of the Komodo UGV (see Fig. 4) requires complex geometric transformations for calculating the configuration space. We used the efficient grid-based C-space algorithm for fast calculation of the discretized configurations in dynamic environments (Lau et al., 2013). C-space defines a 3D grid map of the environment (positions and orientations), where the orientation resolution is defined such that no point inside the footprint translates more than one grid cell if the robot changes its orientation by one increment (see Fig. 5-left). Merging discrete orientations into orientation intervals leads to a compact configuration space representation called the orientation interval graph (OIG). OIG reduces the search space for path planning significantly (Đakulović et al., 2013). Each orientation interval represents a node of OIG, while edges are defined between two neighbor nodes if their x or y coordinates differ for only one increment and their orientation intervals overlap.

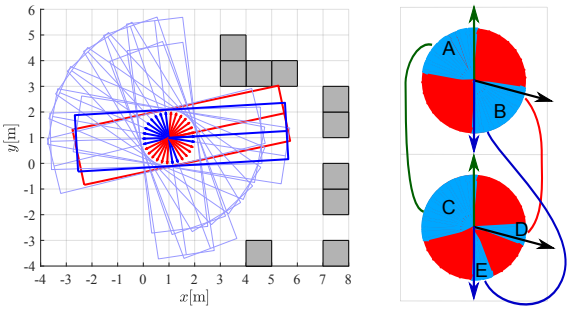


Fig. 5. Discretized configurations of the vehicle: (left) admissible configurations (blue) and colliding configurations (red); (right) two neighbor grid locations with their orientation intervals representing the nodes A, B, C, D, E, where edges are defined between the neighbor nodes that contain the orientation aligned with the forward or backward transition on that edge: $\{A, C\}$ and $\{B, E\}$ are edges, while $\{B, D\}$ is not an edge since it requires moving sideways.

A graph search algorithm computes a path as a sequence of edges in OIG. Since OIG extends the changes from the dynamically updated C-space, a suitable choice is the D* search algorithm due to its fast replanning in changing graphs. However, such a path does not take into account the feasibility of edge transitions for non-holonomic vehicles. For example, some transitions may require moving sideways. Therefore, we introduce the holonomic constraints in OIG by redefining edges only between neighbor nodes that contain the orientation aligned with the forward or backward transition on that edge. For example, Fig. 5 (right) shows two grid locations with orientation intervals as nodes A, B, C, D, E. The transition between nodes C and A contains forward direction orientation, while the transition from A to C contains backward direction orientation. A similar case is for transition between E and B. The transition between B and D requires moving sideways and these nodes are not connected with an edge.

Each node of OIG has a *desired orientation* set to be aligned with the edge transition, where preferred ones are in the forward direction as the UGV sensor field of view is limited to a forward direction. After searching with D*, the path cost of each node is calculated as a sum of inter-states translations and rotations between their desired orientations. The navigation function N of a continuous state is defined as the path cost of the closest discrete state plus the distance from that state (translational and rotational difference).

The navigation function is presented in Fig. 6, where arrows indicate the desired orientations while lighter colors indicate higher costs. Transitions to cells that do not have all admissible orientations are weighted 10 times more than the others. The control applied to such navigation function will repel the vehicle away from the obstacles.

3.3 Model predictive control

Model predictive control (MPC) is applied to find optimal controls $\mathbf{u}(k) = [v(k), \omega(k)]^T$ for the UGV over a prediction horizon H that minimize the values of the navigation function N at current state $\mathbf{s} = [x, y, \theta]^T$:

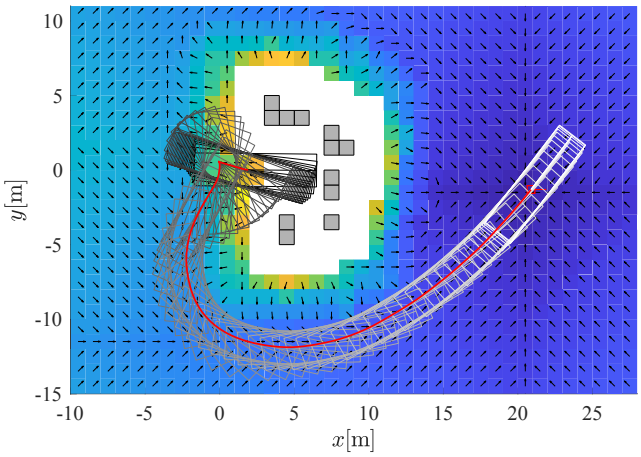


Fig. 6. Navigation function with the driven MPC trajectory.

Fig. 7. Velocity profile, and Lyapunov function during the MPC execution.

$$J(\mathbf{s}) = \min_{\mathbf{u}(k-1)} \sum_{k=1}^{H} N(\mathbf{s}(k)). \tag{5}$$

In horizon interval, future robot state $\mathbf{s}(k)$ is predicted using kinematic model (3), while control actions are constrained by (4). Optimal control sequence $\{\mathbf{u}_k^*\}_0^{H-1}$ that minimizes (5) defines the best feasible future trajectory and its first control action is applied to the robot in current time. In next time sample the procedure repeats.

To prove the convergence of the MPC scheme the value function J (employed as a Lyapunov function) needs to decrease each time sample. To lower computational burden, we used a fixed candidate optimization to solve the MPC problem (Seder et al., 2017). It considers only nine values of acceleration combinations $\{0, \pm a_{\max}\} \times \{0, \pm \alpha_{\max}\}$ to produce the circular control sequences. An example of the MPC execution is presented in Fig. 6 with the velocity profiles and Lyapunov function in Fig. 7. Backward velocity is limited to 0.5 m/s, while forward velocity is limited to 4.167 m/s. Note the slow backward motion at the beginning, while shortly the vehicle rotated to move forward due to the desired forward orientations.

4. EXPERIMENTAL RESULTS

The experiments are performed on the Komodo UGV with the proposed control system architecture. First, we present the experimental setup describing the used sensors, network devices, and computers. Then we present the

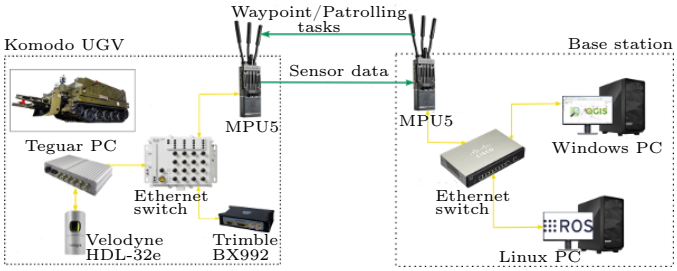


Fig. 8. Experimental setup of the autonomous UGV.

results of Komodo UGV executing the tasks selected by the user and a fail-safe scenario.

4.1 Experimental setup

The experiment was conducted on a system illustrated in Fig. 8, and it consists of UGV and a base station that monitors UGV's missions. UGV is equipped with a rugged box Teguar PC, GPS dual-antenna receiver Trimble BX992 with integrated IMU, CISCO industrial switch, Velodyne HDL-32e, and MPU5 radio system. The rugged box PC runs Linux Ubuntu 16.04 and it is equipped with an i5 CPU and 8 GB RAM. Velodyne HDL-32e is used for map building, and GPS Trimble gives a precise global position. IMU, integrated inside the Trimble receiver, is used for orientation estimation for all three axes. The base station is also equipped with the radio MPU5 module which is the main communication channel between UGV and the base station. The base station consists of two PCs equipped with i7 CPU and 8 GB RAM. One PC is running Windows 10 and is used for mission assignments from QGIS. The other PC is running Ubuntu 16.04 and is used for visualization of all ROS processes on the Teguar PC and to trigger a fail-safe scenario. The UGV is shown in Fig. 4 with its dimensions and sensor placement. Velodyne is positioned on the front of the vehicle with a field of view limited to a forward direction.

4.2 Task execution and supervision

Three scenarios are performed: Waypoint, Patrolling, and Fail-safe. In the Waypoint scenario carton boxes were placed in front of the vehicle to form a U-shaped obstacle, see Fig. 9(a). The user sets the Waypoint tasks in QGIS by selecting the goal positions, Fig. 9(d). All Velodyne readings collected during the experiment in Fig. 9(b) describe the environment of dimension 100m×100m. The lighter representation of the environment is the 2.5D occupancy gridmap in Fig. 9(c), where each location cell of 0.2m×0.2m contains the information of the maximal height of the Velodyne points inside it. Each time step a subset of Velodyne points whose height is above 0.5m and below 3m is compressed into the 2D laser scan for obstacle detection. These obstacles change OIG and initiate path replanning (see Sec. 3.2).

The trajectories of the three scenarios are presented in Fig. 10. The goals set for each scenario are noted with capital letters. The paths to each goal are plotted only as a reference, consisting of grid cell position and desired orientations. Accumulated 2D scan readings for obstacle detection show static obstacles and sometimes detected a floor plane due to the error in the estimation of a 6D pose. Algorithm performance for these three scenarios is presented in Fig. 11. Planning time never exceeds 20 ms which is related to the reduced number of search nodes, where grid cell is 1 m, and more nodes related to disjunctive orientation intervals only appear near the detected obstacles. The MPC execution time goes to candidate trajectory generation and optimization according to the navigation function and never exceeds 30ms. Lyapunov function mostly decreases between the selected goals except in the case when newly detected obstacles increase the costs of the navigation function. In Waypoint scenario, the velocity and Lyapunov function reach zero when the vehicle reaches the region of 0.5m around the goal. In Patrolling scenario, the goals are executed one after another when the vehicle reaches the region of 4m around the goal so the velocity and Lyapunov function do not reach zero. In Fail-safe scenario, the current goal (marked as F) is canceled by the user, and the velocity decreases to make the turn to the previous goal.

5. CONCLUSION

We presented the autonomous navigation system for the Komodo UGV. We demonstrated the necessary modifications of the navigation algorithm to include holonomic constraints of the tracked vehicle. The experiments on the Komodo UGV demonstrated successful execution of the user-set waypoint, patrolling and fail-safe scenarios. Furthermore, experiments show that our navigation system provides fast and reactive motion among obstacles on a flat terrain. Future work will focus on extending the navigation system for operation on uneven terrains.

ACKNOWLEDGEMENTS

This work has been supported by the European Regional Development Fund under the grant KK.01.2.1.01.0138 – Development of a multi-functional anti-terrorism system.

REFERENCES

Đakulović, M., Sprunk, C., Spinello, L., Petrović, I., and Burgard, W. (2013). Efficient navigation for anyshape holonomic mobile robots in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2644–2649.

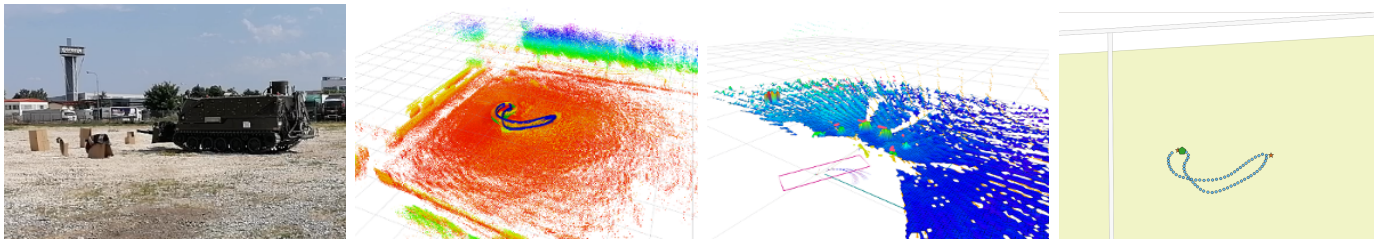
Czarnowski, J., Dąbrowski, A., Maciaś, M., Główka, J., and Wrona, J. (2018). Technology gaps in human-machine interfaces for autonomous construction robots. *Automation in Construction*, 94, 179–190.

Hintjens, P. (2013). *ZeroMQ: messaging for many applications*. "O'Reilly Media, Inc.".

Ivanjko, E., Komšić, I., and Petrović, I. (2007). Simple off-line odometry calibration of differential drive mobile robots. In *16th Int. Workshop on Robotics in Alpe-Adria-Danube Region*, 164–169.

Klančar, G. and Blažič, S. (2019). Optimal Constant Acceleration Motion Primitives. *IEEE Transactions on Vehicular Technology*, 68(9), 8502–8511.

Lau, B., Sprunk, C., and Burgard, W. (2013). Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10), 1116–1130.



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

Fig. 9. Task supervision during the Waypoint scenario: (a) five carton boxes placed in U-shape; (b) accumulated Velodyne readings of the environment with color indicating height level; (c) 2.5D occupancy gridmap with the UGV footprint, detected U-shaped obstacles, and the path to the goal; (d) trajectory in QGIS (blue circles) with goals (red star) and the first position (green square). The video is available at `https://www.youtube.com/watch?v=5DlaxV-3P7Y`.

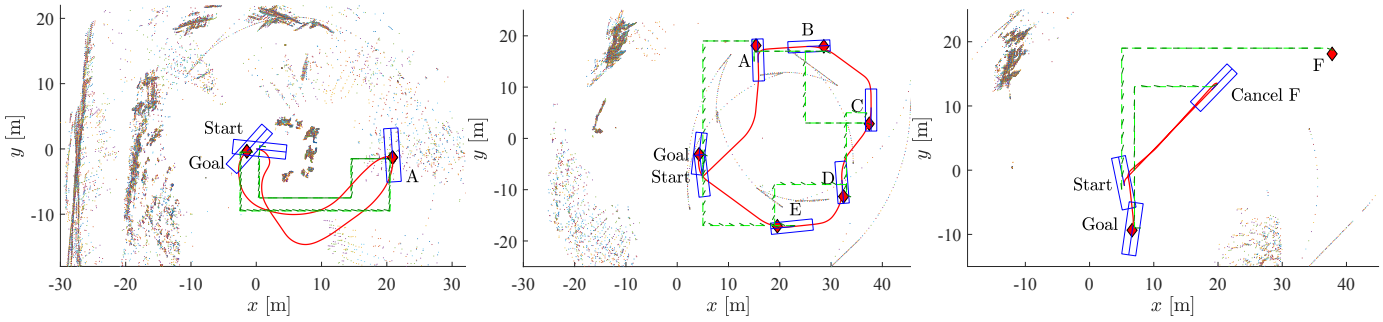


Fig. 10. Trajectory comparison for three scenarios: the Waypoint (left), Patrolling (middle), and Fail-safe (right).

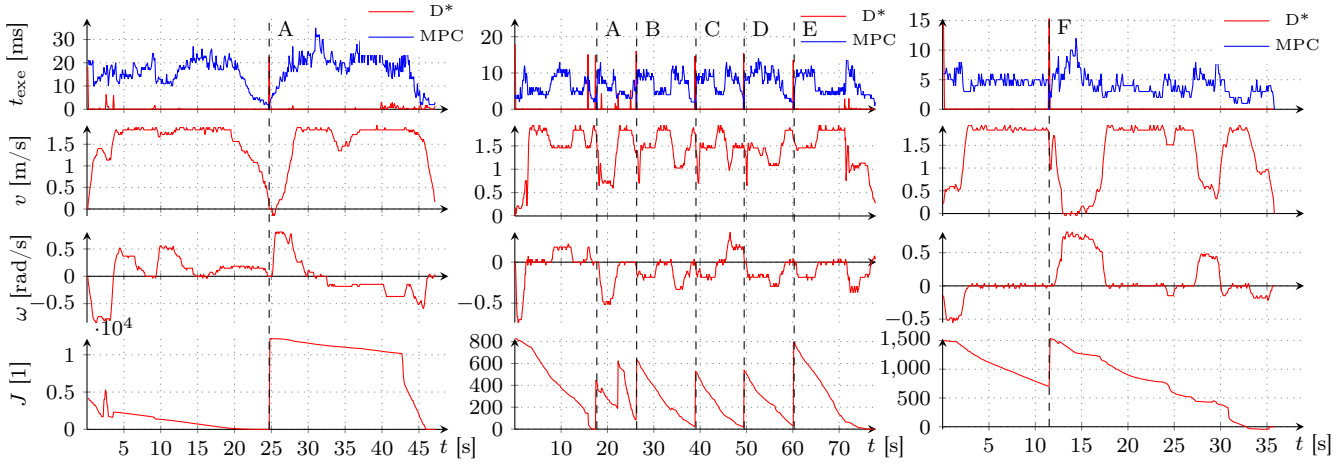


Fig. 11. Algorithm performance, velocity profile, and Lyapunov function in Waypoint scenario with intermediate goal A (left); Patrolling scenario with intermediate goals A, B, C, D, and E (middle); Fail-safe scenario with the cancelled intermediate goal F.

Li, Y., Littlefield, Z., and Bekris, K.E. (2016). Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5), 528–564.

Li, Z., Deng, J., Lu, R., Xu, Y., Bai, J., and Su, C. (2016). Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6), 740–749.

Nohel, J., Stodola, P., and Flasar, Z. (2020). Combat UGV support of company task force operations. In *International Conference on Modelling and Simulation for Autonomous Systems*, 29–42.

Ogren, P. and Leonard, N.E. (2005). A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2), 188–195.

Seder, M., Baotić, M., and Petrović, I. (2017). Receding horizon control for convergent navigation of a differential drive mobile robot. *IEEE Transactions on Control Systems Technology*, 25(2), 653–660.

Šelek, A., Jurić, D., Čirjak, A., Marić, F., Seder, M., Marković, I., and Petrović, I. (2019). Control architecture of a remotely controlled vehicle in extreme CBRNE conditions. In *2019 International Conference on Electrical Drives and Power Electronics*, 273–278.

Yoon, S., Lee, D., Jung, J., and Shim, D.H. (2017). Spline-based RRT* using piecewise continuous collision-checking algorithm for car-like vehicles. *Journal of Intelligent & Robotic Systems*, 1–13.