International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 2021.

Autonomous Flight in Unknown GNSS-denied Environments for Disaster Examination

Daniel Schleich, Marius Beul, Jan Quenzel, and Sven Behnke

***Abstract*— Micro aerial vehicles (MAVs) have high potential for information gathering tasks to support situation awareness in search and rescue scenarios. Manually controlling MAVs in such scenarios requires experienced pilots and is error-prone, especially in stressful situations of real emergencies. The conditions of disaster scenarios are also challenging for autonomous MAV systems. The environment is usually not known in advance and GNSS might not always be available.**

We present a system for autonomous MAV flights in unknown environments which does not rely on global positioning systems. The method is evaluated in multiple search and rescue scenarios and allows for safe autonomous flights, even when transitioning between indoor and outdoor areas.

I. Introduction

Micro aerial vehicles (MAVs) are increasingly used in search and rescue scenarios for fast disaster examination and situation awareness. They are mostly used to map outdoor environments, but MAVs also have potential for information extraction in indoor areas, especially in industrial environments: Structurally damaged buildings or the presence of hazardous chemical substances, might prohibit exploration by humans, and the use of ground robots might be limited due to untraversable terrain.

In most applications, MAVs are remotely controlled by human operators. However, when flying indoors or in the vicinity of obstacles, manually controlling a MAV is a challenging task, which requires an experienced pilot. Since a direct line-of-sight between operator and MAV cannot always be maintained, communication latency and restricted awareness of the MAV's surroundings increase the difficulty even further. Human errors or potential communication losses might result in crashes.

Autonomous flights help to reduce the strain on the operator, increase safety and make the applicability of MAVs feasible for less-experienced pilots. However, many approaches for autonomous navigation either rely on GNSS-based localization or on pre-captured maps of the environment [1]. For real disaster scenarios in indoor environments, both are often not available. In this work, we present our integrated system for safe, autonomous navigation in GNSS-denied, initially unknown environments. Our framework includes

- a method for precise LiDAR-based odometry in unknown environments,
- fast navigation planning considering MAV dynamics and optimizing for safety and flight time,
- and time-optimal trajectory generation and control.

Our system was evaluated in multiple scenarios, like outdoor and indoor flights and a simulated CBRNE-scenario.

Fig. 1: Our MAV guides the teleoperation of a ground robot in a simulated indoor CBRNE-scenario.

II. Related Work

MAVs are commonly used in search and rescue scenarios for exploring large outdoor environments, i.e., for forest fire monitoring [2] or searching survivors during floods [3]. In such scenarios, MAVs usually navigate a sequence of GPS-waypoints and are operated on sufficiently high altitudes such that no obstacle avoidance is necessary. The use of MAVs for indoor fire detection has been studied by Pecho et al. [4]. However, autonomously navigating MAVs in such scenarios requires advanced localization and obstacle avoidance methods.

State estimation methods in GNSS-denied environments are mainly vision-based or LiDAR-based. Mohta et al. [5] employ visual odometry for autonomous flight in indoor areas. During flights in warehouse-like environments, they encountered drift in the state estimation due to a low number of detectable visual features. Mostafa et al. [6] address such challenges by using a combination of radar and visual odometry.

A different approach for autonomous flights in warehouses is proposed by Beul et al. [1]. It employs LiDAR-based localization but relies on a pre-captured environment map. Spurny et al. [7] use two different state estimation filters for

This work has been supported by the German Federal Ministry of Education and Research (BMBF) in the project "Kompetenzzentrum: Aufbau des Deutschen Rettungsrobotik-Zentrums (A-DRZ)"

Institute for Computer Science VI, Autonomous Intelligent Systems, University of Bonn, Friedrich-Hirzebruch-Allee 8, 53115 Bonn, Germany, {schleich, ..., behnke}@ais.uni-bonn.de

outdoor and indoor environments. The outdoor filter uses GNSS and magnetometer data while the indoor filter relies on simultaneous localization and mapping (SLAM) using a 2D LiDAR. The combination of two different state estimators enables smooth transitions between outdoor and indoor environments. A survey on different odometry methods for autonomous navigation in GNSS-denied environments has been compiled by Mohamed et al. [8]. Our approach employs LiDAR-based odometry which does not require previous knowledge of the environment.

Planning collision-free trajectories in cluttered environments is commonly done in a two-stage approach: A low-dimensional spatial path is planned first, and subsequently refined to a high-dimensional trajectory, i.e., using B-Spline path planning [9], quadratic programming [10] or gradient-based optimization [11]. The methods for indoor flights presented above also rely on low-dimensional path planning in 3D or 4D (position and yaw).

Refining position-only paths to high-dimensional trajectories only generates locally optimal solutions, though. In contrast to these works, we directly incorporate the system dynamics into the planning to generate globally optimal trajectories with low execution times. Dynamically feasible motion primitives are used to generate high-dimensional state lattices [12], which are searched using A*. In contrast to previous methods, we additionally apply local multiresolution [13] to the state lattices: The spatial resolution is decreased with increasing distance to the current MAV position. Other approaches of using multiresolution with state lattices include the work of González-Sieira et al. [14], where the resolution level is based on the complexity of the local environment. Andersson et al. [15] plan in high-dimensional space until a time threshold is reached, after which they continue planning in a lower-dimensional space.

An extension of the MAV controller used in our framework is presented in [16]. For future work, we plan to integrate it into our navigation pipeline. In this paper, we rely on fast trajectory replanning to react to newly perceived obstacles.

III. System Setup

In the following sections, we describe the different components of our MAV system, an overview of which is depicted in Fig. 2. First, we present our hardware setup in Sec. III-A. We continue by describing the environment perception modules, including occupancy mapping (Sec. III-C), LiDAR-based odometry (Sec. III-B), and a State Filter (Sec. III-D), which fuses IMU measurements and LiDAR odometry into an estimate of the high-dimensional MAV state. Finally, we describe our navigation and control pipeline: Safe, dynamically feasible trajectories are planned by searching a high-dimensional state lattice including velocities (Sec. III-E). The next waypoint is sampled from the planned trajectory (Sec. III-F) and forwarded to our MAV controller (Sec. III-G).

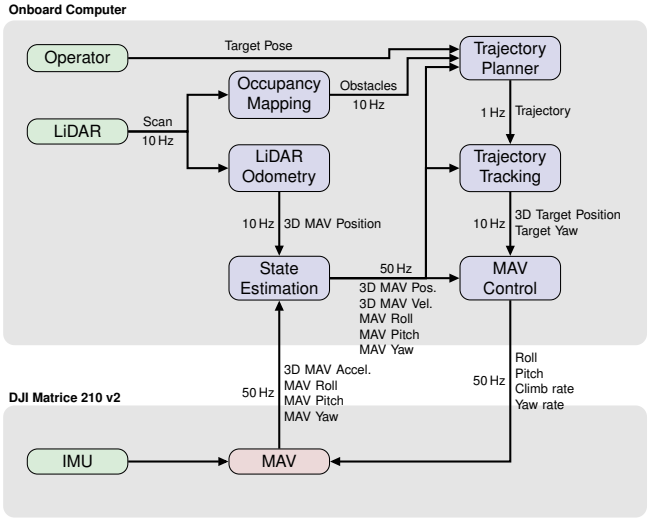

Fig. 2: System overview.

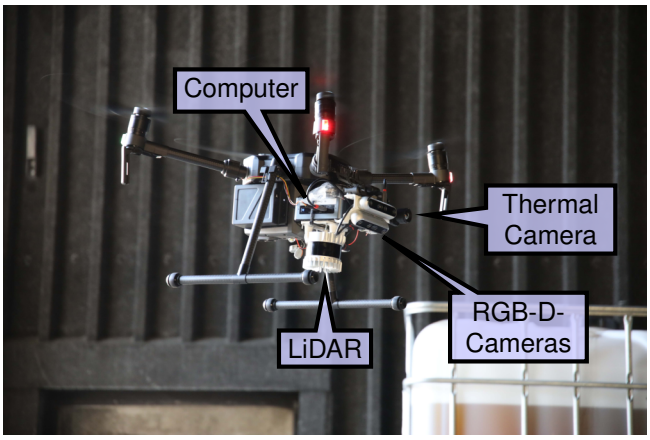

Fig. 3: Hardware design of our MAV.

A. Hardware Design

We base our MAV (Fig. 3) on the commercially available DJI Matrice 210 v2 platform. For onboard computing, we equip it with an Intel Bean Canyon NUC8i7BEH with Core i7-8559U processor and 32 GB of RAM. An Ouster OS-0 3D-LiDAR is used for odometry and obstacle detection. To provide useful information in search and rescue missions, our MAV additionally features a FLIR AGX thermal camera for, e.g., fire detection, and two Intel Realsense D455 RGB-D cameras which are mounted on top of each other to increase the vertical field of view. An example of onboard footage from those sensors is shown in Fig. 4.

B. MARS LiDAR Odometry

We used an early development version of our MARS LiDAR odometry [17] during the experiments. We model surfaces within LiDAR scans with normal distributions derived from measured points on a uniform sparse voxel grid. This surface elements (surfel) map is created with multiple resolutions to increase detail close to the map origin. We half the side length and cell size per level for better representation of the sensor geometry. Our odometry uses a

Fig. 4: Onboard footage from Target 1 of the outdoor experiment described in Sec. IV-B. a) Thermal camera. b) Forward-facing RGB-D camera. c) Downward-facing RGB-D camera.

sliding registration window to simultaneously register multiple surfel maps against a local surfel map. Fig. 5 shows an example from the DRZ Living Lab[1]. A continuous-time Lie group B-Spline [18] describes the UAV trajectory within the sliding registration window. After a certain travelled distance, we add the last scan in a keyframe-based sliding window approach to the local surfel map and shift if necessary the local map to maintain its egocentric property.

C. Environment Mapping

We perform a simple voxel-based occupancy mapping to enable high-level obstacle avoidance and trajectory planning. After registration, we transform the point cloud with the sensor pose into the local map frame and bin the points into voxels with 25 cm side length. Then, we ray-trace with a 3D version of the Bresenham line-search [19], starting from the sensor pose towards the retained voxels. Every voxel in between start and end point that was not previously retained will be counted towards being empty, while every retained voxel will be counted towards being occupied. We improve the robustness against drift by removal of measurements that are older then $N = 30$ scans, thus, reducing the time needed to measure the old obstacle location free.

D. State Estimation

The LiDAR-based odometry estimates the current MAV 3D position and orientation with a frequency of 10 Hz. However, to enable closed-loop control, we need information about the MAV state at the control frequency of 50 Hz, including additional velocity estimates. The MAV platform only provides GNSS-based velocity measurements, which are not available for indoor flights. Thus, we use IMU acceleration measurements to estimate the velocity and correct these estimates using position information from LiDAR Odometry.

During all of our experiments, the IMU provided accurate 3D orientation data and the magnetometer was not affected by buildings or other metallic structures. Since these measurements are provided at a much higher rate than our odometry data, we decided to only fuse IMU orientation measurements and omit LiDAR-based orientation data.

To fuse IMU and LiDAR-Odometry data, we utilize the implementation of an Extended Kalman Filter (EKF) from the robot_localization library [20]. Input to the EKF are 3D positions from LiDAR-Odometry, and 3D orientations (which have to be transformed into the map frame first) and linear accelerations measured by the IMU. The EKF outputs high-dimensional MAV states with 50 Hz.

[1] https://rettungsrobotik.de/living-lab/

E. Trajectory Planning

In disaster response scenarios, low mission execution times are essential, especially since MAV flight times are limited. Thus, flight trajectories should not only be optimized with respect to safety but also for execution time. This can be achieved by directly incorporating MAV dynamics into the planning.

We employ our trajectory planning method from [21], which is based on the framework of Liu et al. [12]. The MAV state is modeled as a 6-tuple $s = (\mathbf{p}, \mathbf{v}) \in \mathbb{R}^6$ consisting of a 3D position $\mathbf{p}$ and velocity $\mathbf{v}$. A state lattice graph $\mathcal{G}$ is generated by unrolling motion primitives from the initial MAV state s_0. Each primitive corresponds to applying a constant acceleration control $\mathbf{u}$ over a short time interval τ, i.e., can be expressed as a time-parameterized polynomial

$$F_{\mathbf{u},s}(t) = \begin{pmatrix} \mathbf{p} + t\mathbf{v} + \frac{t^2}{2}\mathbf{u} \\ \mathbf{v} + t\mathbf{u} \end{pmatrix}, \quad \text{for } t \in [0, \tau].$$

Here, $s = (\mathbf{p}, \mathbf{v})$ denotes the initial state of the motion primitive and the corresponding action costs are defined as the weighted sum of control effort and primitive duration, i.e.,

$$C(F_{\mathbf{u},s}) = ||\mathbf{u}||_2^2 \tau + \rho\tau.$$

Trajectories are generated by applying A* search to the state lattice graph.

Searching high-dimensional state spaces is computationally expensive, but frequent replanning is necessary in unknown environments to react to newly perceived obstacles. Thus, we extend the method of Liu et al. to achieve fast replanning times. We restrict the state positions of the lattice graph $\mathcal{G}$ to the corners of a MAV-centered local multiresolution grid to significantly reduce the state space size. An example is depicted in Fig. 6. The MAV vicinity is represented at a high spatial resolution, while the resolution decreases with increasing distance to the MAV position.

Additionally, we propose a search heuristic that considers the resolution of the motion primitives by solving 1D subproblems along the x-, y- and z-axis. For each pair of

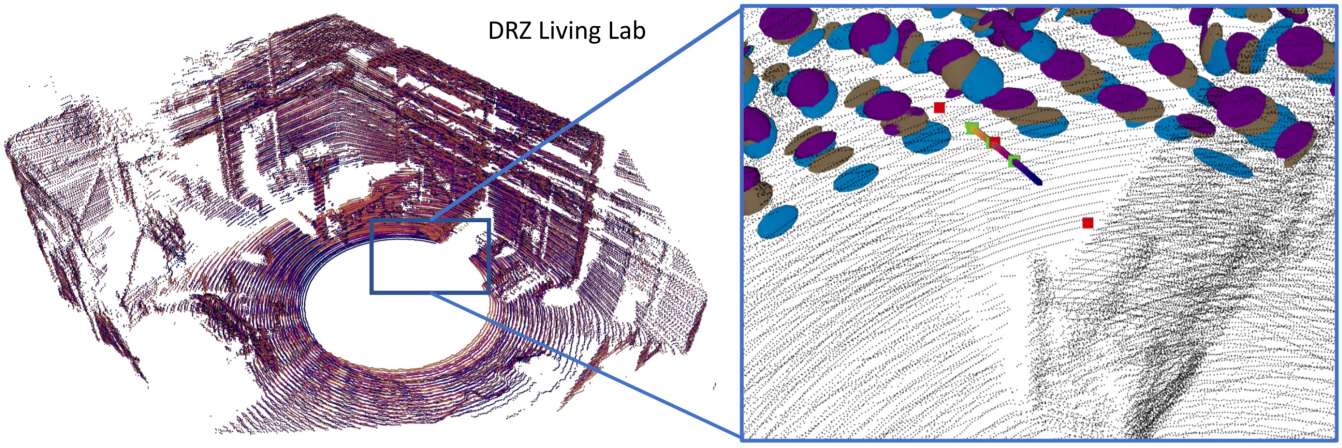

Fig. 5: Aligned point clouds of the sliding registration window described in III-B. Control points (red) define the spline (blue to yellow line) and interpolate the scan poses (green). The ellipsoids (surfel, colored per scan) describe the distribution of points in their vicinity.

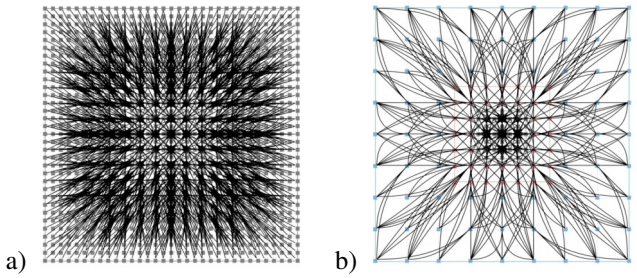

Fig. 6: State lattice graphs. a) Uniform. b) Local multiresolution. The spatial position of nodes is fixed to the corners of a multiresolution grid with high resolution at the current MAV position, and coarser resolution for more distant areas.

signed distance to the goal position and start velocity, we precompute the costs of the optimal 1D trajectory without obstacle consideration. During search, we combine the 1D costs into an estimate for the 3D costs: The flight time is calculated as the maximum over the individual axes, and the control costs are those of the sub-problem with highest execution time. For further details on local multiresolutional state lattices and the proposed 1D heuristic, we refer to [21].

To increase the safety of the planned trajectories, we extend our previous work by adding additional obstacle costs to our planning framework. The center points of the obstacles detected by the environment mapping module (cf. Sec. III-C) are inserted into a k-d tree [22] for fast distance queries. During search, we sample positions on the motion primitives for collision checking. If the distance $d(s, o)$ between a sample s and the nearest obstacle o is lower than a safety distance d_{min}, the position is assumed invalid. For distances larger than an upper threshold d_{max}, no obstacle costs are added to the motion primitive. Otherwise, we add costs c that linearly decrease with increasing distance to the nearest obstacle, i.e., we set $c = \frac{d_{\text{max}} - d(s,o)}{d_{\text{max}} - d_{\text{min}}}$.

Due to trajectory tracking errors, the MAV might enter the safety area around obstacles. Replanning would fail in such cases, since the initial MAV position is invalid. Therefore, we allow invalid start positions during collision checking, but require that the obstacle distance between two adjacent position samples does not decrease. By adding high obstacle costs to invalid positions, we ensure that the MAV immediately returns to a safe distance on the shortest possible trajectory.

To react to newly perceived obstacles, replanning is triggered at 1 Hz. This frequency was empirically determined from maximum replanning times. As described in [21], replanning takes less than 1 s in most of the cases, even for large environments. For smaller environments, a higher replanning frequency is possible.

As described in Sec. III-F, a waypoint is sampled from the current trajectory and forwarded to the MAV controller. The start state s_{replan} for replanning is selected from the trajectory at the time t_{plan} ahead of the current waypoint. Here, t_{plan} corresponds to the estimated replanning duration. The trajectory from the current MAV state up to s_{replan} is not updated.

F. Trajectory Tracking

Our planning framework generates second-order trajectories, which need to be further refined for execution. Thus, we sample waypoints on the planned trajectory with a fixed timestep of 0.1 s. The first waypoint p_0 is selected at a fixed time interval t_0 after the trajectory start and is forwarded as a goal pose to the MAV controller. Afterwards, we keep publishing waypoints at a frequency of 10 Hz while moving them accordingly along the trajectory. If the distance between the MAV and the current waypoint p_i exceeds a threshold d_{tracking}, the MAV was not able to track the trajectory. In this situation, we distinguish between two cases:

1) The MAV tracks the trajectory spatially but at a lower velocity.
2) The MAV does not track the trajectory spatially.

In the first case, we wait until the MAV reaches a position sufficiently close to p_i before continuing to publish the next waypoints. In the second case, the distance between the MAV and the planned flight path exceeds a threshold d_{replan}. Thus, we abort trajectory tracking and restart by planning a new trajectory, starting from the current MAV position.

Fig. 7: Onboard footage during a CBRNE-scenario. By providing a top-down view of the workspace, our MAV facilitates the teleoperation of a ground robot, which has to secure hazardous substances.

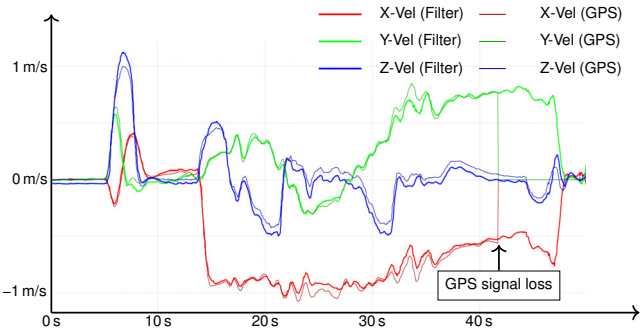

Fig. 8: Comparison of estimated velocities of our state filter and ground truth GPS velocities during a part of the flight described in Sec. IV-C. Note that estimation is robust to GPS loss.

Note that we do not aim at precisely executing the planned trajectory. Due to sampling the waypoints ahead in time, the MAV changes flight direction slightly before the trajectory does. These tracking errors do not affect the MAV safety since all planned trajectories keep sufficiently large safety margins to obstacles.

G. MAV Control

Reliably approaching the sampled waypoint close to structures despite external disturbances is a challenging task. For precise MAV control, we employ our method presented in [23] with the extensions from [24]. For brevity, in this section, we cover only the most essential aspects of the algorithm.

Our technique models the MAV as a multidimensional triple integrator with nonlinear state boundaries and jerk as system input. Based on this model, we analytically generate third-order time-optimal trajectories that satisfy asymmetrical input ($j_{min} \leq j \leq j_{max}$) and state constraints ($a_{min} \leq a \leq a_{max}$, $v_{min} \leq v \leq v_{max}$). Trajectories are computed from the current state $(p, v, a)^\intercal_{MAV}$ to the sampled waypoint state $(p, 0, 0)^\intercal_{wayp}$ with zero target velocity and acceleration. We temporally synchronize the x-, y- and z-axis to arrive at the target state at the same time.

Trajectories are replanned with 50 Hz and are directly executed by the MAV serving as model predictive controller (MPC). As stated above, our MPC generates jerk commands, but the low-level flight controller expects pitch resp. roll commands. Therefore, we assume pitch and roll to relate to $\theta = \text{atan2}(a_x, g)$ and $\phi = \text{atan2}(a_y, g)$. Thus, we send smooth pitch θ and roll ϕ commands for horizontal movement and smooth climb rates v_z instead.

MAVs are inherently unstable (triple integrator model); thus, closed-loop control is necessary to stabilize the system at all times. Since many demanding tasks are running onboard the MAV's computer, we assign real-time priority for the MPC to ensure execution of this vital task.

IV. Evaluation

We employed our MAV in two simulated CBRNE-scenarios in cooperation with firefighters and multiple teleoperated ground robots. Our MAV was used to generate an overview of the situation, while the ground robots were used

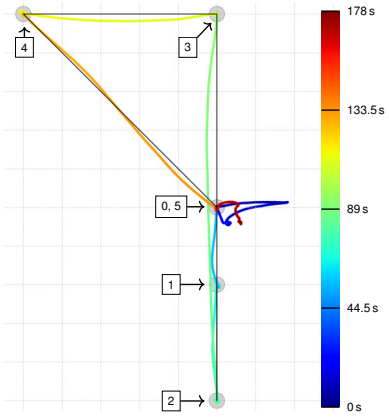

Fig. 9: Top-down view of the flight path during our control experiment. Target positions are marked with gray circles and the optimal trajectory is depicted in black. The background grid has a resolution of 1 m.

to secure hazardous substances. In both scenarios, our MAV proved useful in assisting the operators of the ground robots by providing additional top-down views of the workspace. Figure 1 shows a scene during the first CBRNE-scenario and onboard footage of the second scenario is provided in Fig. 7.

In the following sections, we evaluate the components of our framework. Here, we concentrate on the interaction between the different components and on the overall performance of our system. A more detailed evaluation of the individual components, including computation times and benchmarks against other methods, can be found in the corresponding papers [17], [21], [24].

In all experiments, environment maps for collision avoidance and localization are generated online during flight. No prior knowledge about the environment is necessary. Since our MAV features a 3D-LiDAR with a vertical field of view of 90°, many obstacles are perceived while the MAV is still on the ground. Thus, initial environment maps are already available before takeoff. Target poses are manually defined by placing a marker in those maps.

We analyze data from multiple autonomous flights captured during simulated search and rescue scenarios. Our

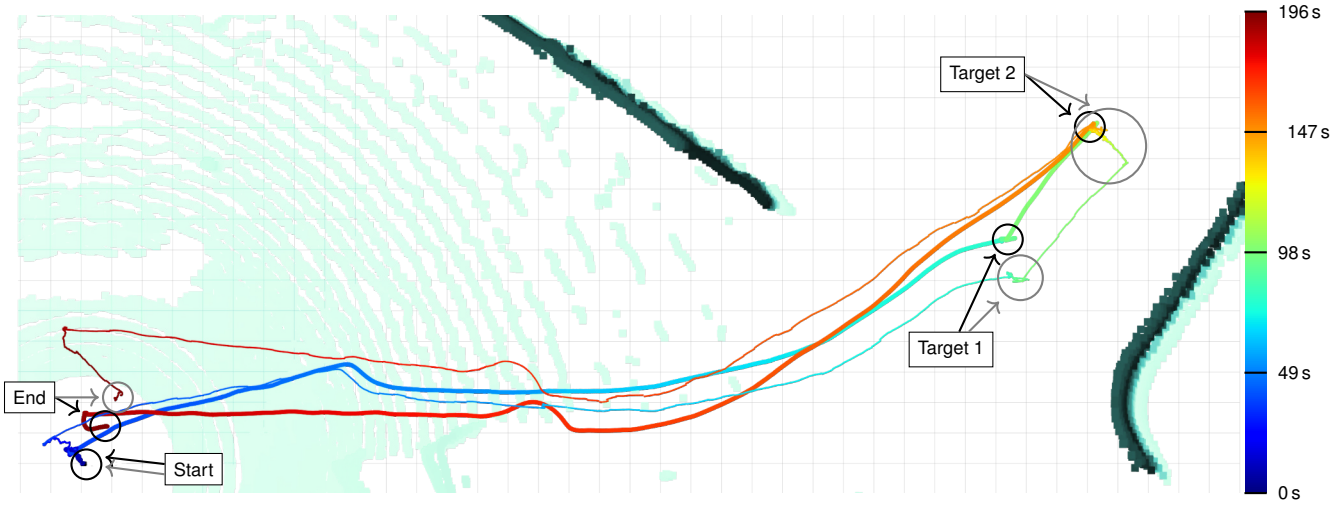

Fig. 10: Comparison of our LiDAR-based odometry (thick) and GPS (thin) during execution of the outdoor mission (top-down view). The parts of the trajectory corresponding to start, end and the two target poses are encircled (black for LiDAR-based odometry and gray for GPS). The background depicts the initial obstacle map. Darker color indicates a larger height value. The background grid has a resolution of 1 m.

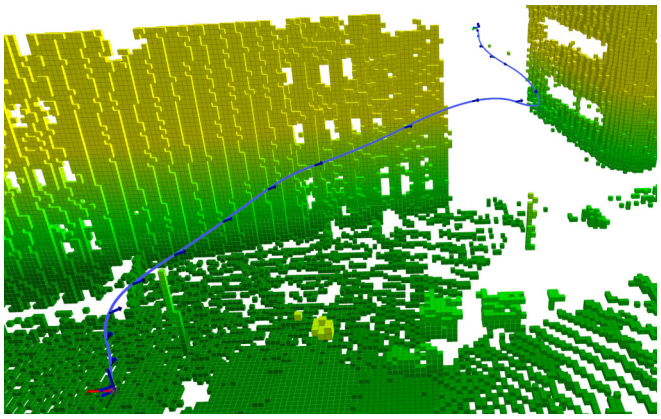

Fig. 11: A planned trajectory in the outdoor scenario. Blue arrows depict the flight direction. The temporal distance between two consecutive arrows is 0.5 s.

system works in outdoor and indoor environments, including transitions between both. Since we fly close to structures in initially unknown environments, we chose to restrict the maximum flight velocity to 1 m/s.

A. State Estimation and Control

Reliably controlling the MAV close to structures requires precise state information. Thus, we evaluate whether our state estimation (cf. Sec. III-D) is sufficiently accurate to be used with the MAV controller (cf. Sec. III-G) for precise navigation. As described above, our state filter infers velocities from fused position and acceleration measurements. In a first experiment, we compare the estimated velocities against velocities measured by the onboard DJI GPS, which we consider as ground truth. The corresponding data was captured during an autonomous flight from the outside into an industrial building, which will be detailed in Sec. IV-C. Figure 8 shows that the estimated velocities align with the GPS-based ground truth until we enter the building. While our state filter continues to generate velocity estimates, no GPS-data is available inside the building. Note that the DJI interface still provides z-velocities since those are not only inferred from GPS but also from barometric data and a downward facing ultrasonic sensor.

In a second experiment, we evaluate MAV control based on the state estimation. We manually took off and then switched to autonomous control. We commanded the MAV to five different target positions, as shown in Fig. 9. The MAV precisely reached all of the targets without overshooting. Furthermore, the position at each waypoint was stably maintained for several seconds.

B. Outdoor Flight

In this experiment, we apply our system in an outdoor scenario where the MAV has to observe a facade from two different manually defined poses. The MAV starts at the ground and has to move into a passage between two buildings. An example for a planned trajectory is depicted in Fig. 11. This scenario is challenging for autonomous navigation since the GPS signal quality is significantly reduced in the narrow passage between the two buildings. However, our approach successfully guides the MAV on a safe trajectory towards both target poses and back again due to our robust LiDAR-based odometry. We compare the localization with our approach against GPS in Fig. 10. Especially when hovering at Target 2, which is located at the center of the narrow passage, GPS shows significant drift while our odometry correctly estimates that the MAV is not moving.

C. Combined Outdoor and Indoor Flight

In a last experiment, we apply our system in a scenario where the MAV has to explore the inside of an industrial

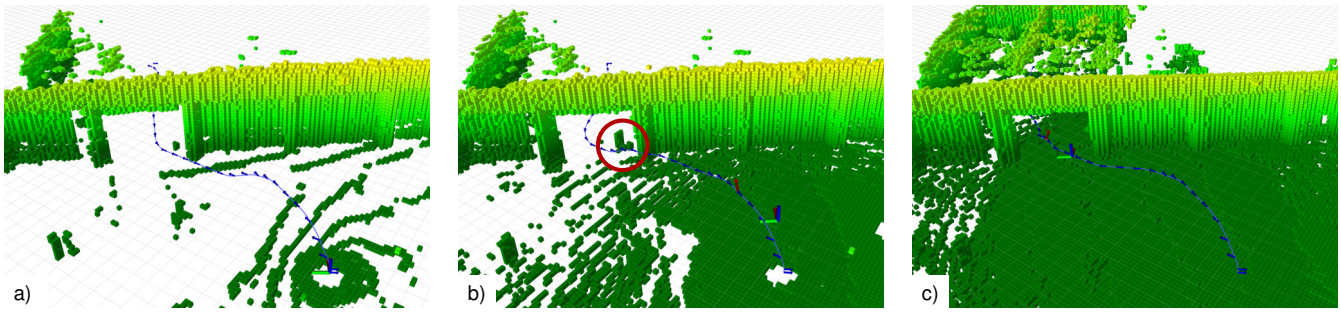

Fig. 12: Replanning during the indoor experiment. A red arrow corresponds to the currently sampled waypoint for the MAV controller. Blue arrows depict the flight direction. The temporal distance between two consecutive blue arrows is 0.5 s. Note how the trajectory is updated to react to a dynamic obstacle (red circle).

Fig. 13: Our MAV during autonomous flight. a) Reaching the observation pose inside an industrial building. b)Autonomously exiting the building through the gate while being supervised by a safety pilot.

building. The MAV starts outside and has to enter through a gate to reach a manually defined observation pose. There, it rotates to generate an overview of the environment and leaves the building again. Example images of the flight are shown in Fig. 13. The full mission from takeoff until we left the building again took 105 s. Since we plan shorter trajectories and have to find a path through a narrow doorway, we disable local multiresolution planning and use uniform state lattices instead, which is more suitable in such scenarios.

Additionally, we evaluate how our framework reacts to newly perceived obstacles. The initially planned trajectory is shown in Fig. 12 a. After takeoff, a person moves into the doorway. The new obstacle is added to the occupancy map (red circle in Fig. 12 b). The trajectory is updated accordingly to avoid a collision. In Fig. 12 c, the person has moved away

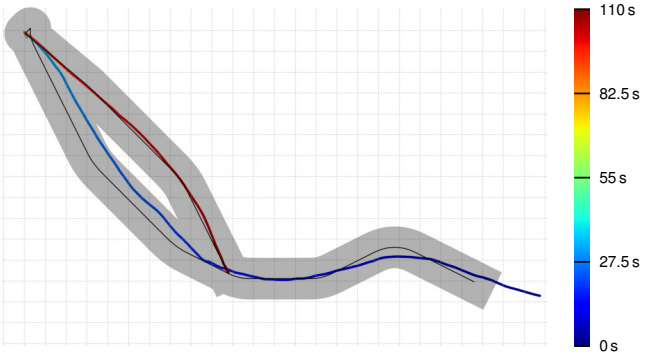

Fig. 14: Trajectory of the combined outdoor/ indoor scenario. The actual flight trajectory is colorized with respect to flight time. The black line connects the waypoints sampled from the planned trajectory. Since the first waypoint is sampled t_0 =1 s ahead of the initial position, there is some offset to the start of the flight trajectory. The gray area marks a tunnel with radius 1 m around the planned trajectory. The background grid has a resolution of 1 m.

and the corresponding cells in the obstacle map are cleared again. Since the MAV has already committed to a point in the doorway (red arrow), only parts of the trajectory inside the building are updated.

Finally, we evaluate the accuracy of our trajectory tracking method. Figure 14 compares the actual flight trajectory against the planned trajectory. As expected, the MAV does not perfectly track the planned trajectory in curves, due to sampling the waypoints ahead in time. However, the tracking error is always below 1 m and is thus covered by the larger security margin to obstacles. The MAV accurately tracks the trajectory on straight segments.

Note that no motion capture system was available to evaluate the accuracy of our localization inside the industrial building of this flight experiment. However, Quenzel and Behnke [17] present a corresponding experiment of an indoor flight inside the DRZ Living Lab (cf. Fig. 5), which is equipped with a motion capture system.

V. Conclusion

In this paper, we provided detailed insight into our navigation framework for safe, autonomous flight in GNSS-denied environments. We showed the viability of our approach by employing our MAV system in multiple search and rescue

scenarios. Our proposed method for LiDAR-based odometry allows safe indoor flights. Furthermore, it increases navigation accuracy when flying close to structures in outdoor environments where GPS signal quality is reduced. Fast replanning and precise control enable safe flights in dynamic environments. No previous knowledge of the environment is necessary since all maps are built online during flight. Thus, our system addresses many challenges encountered in real disaster-response scenarios.

REFERENCES

[1] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, "Fast autonomous flight in warehouses for inventory applications," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3121–3128, 2018.

[2] L. Merino, F. Caballero, J. R. Martínez-de Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent & Robotic Systems (JINT)*, vol. 65, no. 1, pp. 533–548, 2012.

[3] R. Ravichandran, D. Ghose, and K. Das, "UAV based survivor search during floods," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 1407–1415.

[4] P. Pecho, P. Magdolenová, and M. Bugaj, "Unmanned aerial vehicle technology in the process of early fire localization of buildings," *Transportation Research Procedia*, vol. 40, pp. 461–468, 2019.

[5] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in GPS-denied and cluttered environments," *Journal of Field Robotics (JFR)*, vol. 35, no. 1, pp. 101–120, 2018.

[6] M. Mostafa, S. Zahran, A. Moussa, N. El-Sheimy, and A. Sesay, "Radar and visual odometry integrated system aided navigation for UAVS in GNSS denied environment," *Sensors*, vol. 18, no. 9, p. 2776, 2018.

[7] V. Spurny, V. Pritzl, V. Walter, M. Petrlik, T. Baca, P. Stepan, D. Zaitlik, and M. Saska, "Autonomous firefighting inside buildings by an unmanned aerial vehicle," *IEEE Access*, vol. 9, pp. 15 872–15 890, 2021.

[8] S. A. Mohamed, M.-H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, "A survey on odometry for autonomous navigation systems," *IEEE Access*, vol. 7, pp. 97 466–97 486, 2019.

[9] E. Koyuncu and G. Inalhan, "A probabilistic B-spline motion planning algorithm for unmanned helicopters flying in dense 3D environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 815–821.

[10] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[11] M. Nieuwenhuisen and S. Behnke, "Search-based 3D planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9123–9129.

[12] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.

[13] S. Behnke, "Local multiresolution path planning," in *Robot Soccer World Cup*. Springer, 2003, pp. 332–343.

[14] A. González-Sieira, M. Mucientes, and A. Bugarín, "An adaptive multi-resolution state lattice approach for motion planning with uncertainty," in *Robot 2015: Second Iberian Robotics Conference*. Springer, 2016, pp. 257–268.

[15] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz, "Receding-horizon lattice-based motion planning with dynamic obstacle avoidance," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4467–4474.

[16] M. Beul and S. Behnke, "Trajectory generation with fast lidar-based 3D collision avoidance for agile MAVs," in *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2020, pp. 42–48.

[17] J. Quenzel and S. Behnke, "Real-time multi-adaptive-resolution-surfel 6D LiDAR odometry using continuous-time trajectory optimization," *arXiv:2105.02010*, 2021.

[18] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative B-Splines on Lie groups," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.

[20] T. Moore and D. Stouch, "A generalized extended Kalman filter implementation for the robot operating system," in *Int. Conf. on Intelligent Autonomous Systems (IAS)*. Springer, 2014.

[21] D. Schleich and S. Behnke, "Search-based planning of dynamic MAV trajectories using local multiresolution state lattices," IEEE International Conference on Robotics and Automation (ICRA), 2021.

[22] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[23] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016.

[24] ——, "Fast full state trajectory generation for multirotors," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017.