A Framework for Safe Navigation of Unmanned Aerial Vehicles in Unknown Environments

Mariusz Wzorek, Cyrille Berger, Patrick Doherty
Department of Computer and Information Science
Linköping University Linköping, Sweden
email: mariusz.wzorek@liu.se

***Abstract*—This paper presents a software framework which combines reactive collision avoidance control approach with path planning techniques for the purpose of safe navigation of multiple Unmanned Aerial Vehicles (UAVs) operating in unknown environments. The system proposed leverages advantages of using a fast local sense-and-react type control which guarantees real-time execution with computationally demanding path planning algorithms which generate globally optimal plans. A number of probabilistic path planning algorithms based on Probabilistic Roadmaps and Rapidly-Exploring Random Trees have been integrated. Additionally, the system uses a reactive controller based on Optimal Reciprocal Collision Avoidance (ORCA) for path execution and fast sense-and-avoid behavior. During the mission execution a 3D map representation of the environment is build incrementally and used for path planning. A prototype implementation on a small scale quad-rotor platform has been developed. The UAV used in the experiments was equipped with a structured-light depth sensor to obtain information about the environment in form of occupancy grid map. The system has been tested in a number of simulated missions as well as in real flights and the results of the evaluations are presented.**

I. Introduction

The use of Unmanned Aerial Vehicles (UAVs) in recent years has been becoming common place in military and civilian markets. We are seeing an increased number of deployed platforms in many real world applications, such as search and rescue, physical structure inspection, agriculture, to name a few. The necessary changes to the aviation laws are still under discussion in many countries with the main concern on how to safely allow insertion of UAVs in the common airspace. From that perspective, the need for safe navigation is of great importance in order to minimize risks associated with the deployment of this technology, that is to minimize the risk of property damage and, most importantly, human injury. In this paper we address the problem of safe navigation of multiple UAVs operating in unknown environments. The UAVs are equipped with range sensors capable of sensing their surroundings for the purpose of collision avoidance as well as building of a map.

Typical solutions applied to solve this problem include using reactive controllers that guarantee real-time performance at the expense of optimality. Additionally, most sense-and-react type controllers do not include any predictive capability hence given range-limited sensor information about the environment they tend to suffer from the local minima problem which in this context means getting stuck between multiple obstacles in the environment.

An alternative solution is to use deliberative techniques developed for solving navigation tasks, namely path planners. Sample-based approaches such as Probabilistic Roadmaps (PRM) [1], Rapidly-Exploring Random Trees (RRT) [2] [3] have been already successfully deployed in the UAV domain. They are probabilistically optimal and complete but require a 3D world model for collision checking. Additionally, those techniques are computationally more demanding than reactive controllers making them infeasible for direct application to problems where real-time execution guarantees are required.

In this paper we propose a framework that combines sample-based path planning techniques with reactive control and sensing for solving the problem of multi UAV navigation in unknown environments. The system leverages advantages of fast sense-and-avoid control with more time consuming path planning techniques which in turn generate globally optimal plans. In the context of this work, a globally optimal plans are considered to be collision-free and optimal in relation to the 3D map knowledge of the environment acquired so far, since the UAV is operating in an unknown environment.

The integration of the reactive and deliberative components is not trivial due to several factors. During a mission execution the control signals guaranteeing collision-free navigation have to be provided in a timely fashion at all times. Additionally, the large amount of noisy sensor data has to be processed and integrated in form of a map in order to be used for optimal plan generation. Finally, smooth transitions between updated plans provided by path planners need to be assured.

The proposed framework builds upon the ideas presented in [4] [5], where path planners used a set of strategies in order to provide repaired plans during the mission execution in an anytime fashion. The underlaying control mode used for the path execution was based on the 3D trajectory following reference controller [6]. Its goal was to minimize the tracking error between the planned path and the actually

executed trajectory. In the framework presented in this paper the Optimal Reciprocal Collision Avoidance (ORCA) [7] controller is used instead. Plans generated by path planners are represented as a set of sub-goals and are used as an input to the controller. The controller is guided by globally optimal plans and makes sure to reactively avoid any collisions with static and dynamic (other UAVs) obstacles. This property is especially important when dealing with navigation of multiple UAVs in unknown environments.

A system based on the ORCA algorithm deployed on multiple UAV platforms was presented in [8]. The authors propose an extension to the original approach in order to deal with the perceived static obstacles at the reactive layer. This solution still suffers from the local minima problem since the control signals are generated based on the locally sensed obstacles and no predictive functionalities are used. In our framework this problem is aliviated by using the path planners for the generation of globally optimal plans which take into account all the perceived obstacles.

Several other approaches have been presented in the literature that combine reactive and deliberative functionalities relevant to the problem considered in this paper. The two most notable are the following. A system that uses a combination of RRT path planning with Model Predictive Control has been presented in [9]. An alternative approach has been described in [10] where a stochastic optimization technique called a Particle Swarm Optimization (PSO) was applied. The system first uses a simple one-at-a-time strategy to compute a collision-free non-optimal solution which is then refined by PSO in order to improve its optimality.

Our work in contrast not only considers combining reactive and deliberative components but also includes the aspect of environment sensing. In the proposed framework functionalities for 3D map building are integrated. The path planning algorithms rely on fast and efficient collision checks, therefore, a method for incremental updates of the collision checker data structures is proposed. The system presented in this paper has been tested in simulation and verified in real flight tests. The results are presented and discussed.

The structure of the paper is as follows. Section II presents the dynamic collision avoidance framework including descriptions of all its components and their integration. The experimental validation in simulations and real flights is described in Section III. The paper concludes in Section IV including a discussion of the future work.

II. Dynamic Collision Avoidance Framework

This section presents an overview of the proposed dynamic collision avoidance framework. The system consists of five functional components which execute independently: *3D Map, Path Planning Server, Execution Coordinator, Reactive Collision Avoidance and Low-level Control*. Schematic of the system is depicted in Fig. 1.

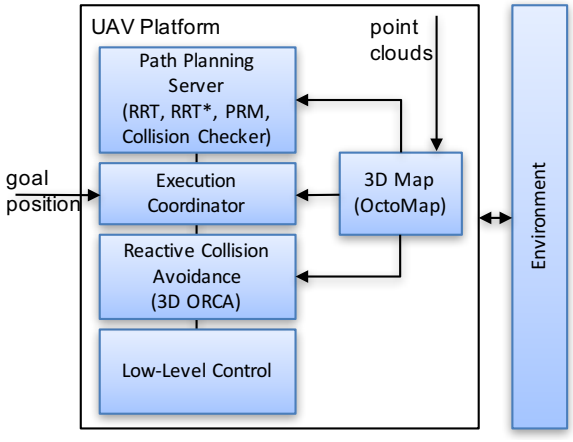


Figure 1: Overview of the dynamic collision avoidance framework.

The *3D Map* module is used to store and continuously update the representation of the environment. Map updates are based on the perceived range sensor data in form of point clouds. Such data can be generated by a number of sensors, for example, a structured-light depth camera, a laser range finder or a stereo system.

The *Path Planning Server* includes a number of path planners capable of generating globally optimal plans given the current knowledge of the environment encoded in the collision checker data structures.

The *Reactive Collision Avoidance* module implements a reactive controller which executes paths generated by planners. It makes sure that there is no collision with static and dynamic obstacles such as other UAV platforms operating in the environment. The controller generates velocity commands which are then passed to the *Low-Level Control* system of the UAV platform.

The *Execution Coordinator* (EC) is the central component integrating the above listed functionalities in order to perform the navigation mission, that is safely reaching the goal position. It achieves its function by performing the following tasks:

- monitoring execution of the current plan for potential collisions given newly acquired sensor data,
- triggering path planner collision checker updates,
- querying path planners for new or updated plans given the current map knowledge.

The following subsections provide detailed descriptions of the five functional components of the proposed framework.

A. 3D Map

In order to navigate safely in an unknown environment it is essential to perceive and represent it in form of a map. Both, path planners, and reactive controller make use of the information stored in the map.

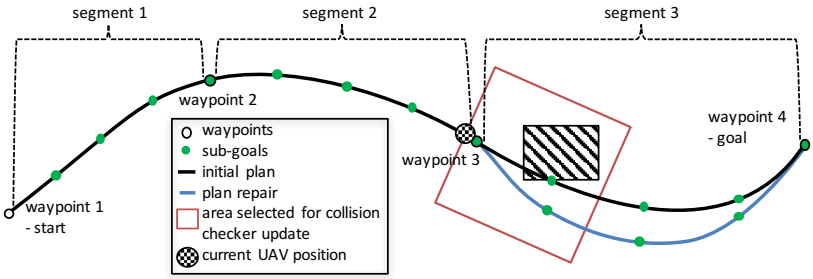


Figure 2: An example plan generated by the path planner including a plan repair and a collision checker update.

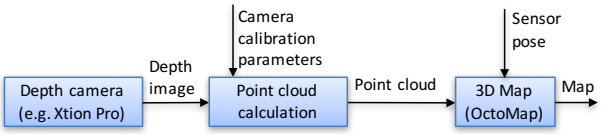


Figure 3: Schematics of the map generation process.

The Dynamic Collision Avoidance framework makes use of an octree-based 3D map structure, the OctoMap [11] to represent the perceived environment. The OctoMap is an open source framework suitable for efficient map representation. It uses probabilistic occupancy estimation which makes it suitable for incorporating noisy sensor data. The 3D map is built incrementally based on the information about the environment in form of point clouds.

Point clouds can be generated using a number of sensors and techniques. These include structured-light depth cameras, laser range finders, stereo vision systems or monocular structure from motion techniques, to name the most commonly used ones. In the context of this paper we are focusing on using a depth camera as the main source of range data. Sensors of this type are characterized by low weight and low power consumption, fast update rate and are therefore suitable for small scale UAV platforms.

The process of building a map of the environment for the UAV platform used for experimental validation presented in Section III-A is schematically depicted in Fig. 3. First, a depth image is obtained from the sensor. Each of the image pixels encodes a distance to an object in the environment. If no information is available for certain pixels then this is encoded with a reserved values denoting out-of-range measurements. Construction of a point cloud from a depth image is achieved by first rectifying the image (i.e. removing the lens distortion) and applying the camera pin-hole model with the intrinsic camera parameters. The required parameters are obtained through the calibration procedure. Finally, the point cloud is transformed from the sensor's coordinate frame to the global one by applying the extrinsic parameters (i.e. translation and rotation of the sensor) and added to the map.

B. *Path Planning Server*

Using path planning techniques for navigation allows for finding collision-free and optimal paths to reach mission goal positions given a 3D map of the environment. Commonly used planners for the UAV domain take advantage of sample-based probabilistic approaches and include Probabilistic Roadmaps (PRM), Rapidly-Exploring Random Trees (RRT), and its variation (RRT*) [12], [13].

PRM planners work in two phases, one offline and the other online. In the offline phase a discrete roadmap in form of a graph representing an approximation of a free state space is generated using a 3D world model encoded in the collision checker. In the online (querying) phase initial and goal states are added to the previously generated roadmap and a graph search algorithm such as A* is used to find a path. Since the PRM requires the offline phase which is computationally expensive this type of algorithm is more suited for multi query usage in static environments.

The RRT and RRT* are variants of the sample-based algorithms that, unlike the PRM planners, do not use a precompiled roadmap. Instead, they use a specialized search strategy to construct a roadmap online to find solutions quickly during runtime. This is a strong advantage since the RRT and RRT* do not require an a-priori 3D model of the environment and that makes them applicable for solving planning problems in dynamic and unknown environments. The RRT* algorithm is probabilistically complete, provides anytime solutions and provably converges to an optimal solution.

The *Path Planning Server* (PPS) of the presented system includes all of the above algorithms. However, for the above mentioned reasons, the RRT* has been chosen as the main path planner used in the context of this paper (i.e. solving the navigation problem in unknown environments).

All path planners rely on efficient collision checking when looking for a solution and, in fact, it is one of the most computationally demanding elements of the process. The OctoMap which is used for the 3D map representation is well suited for map building considering noisy sensor data. Unfortunately, this representation is not directly well suited for fast and efficient collision checking required for path

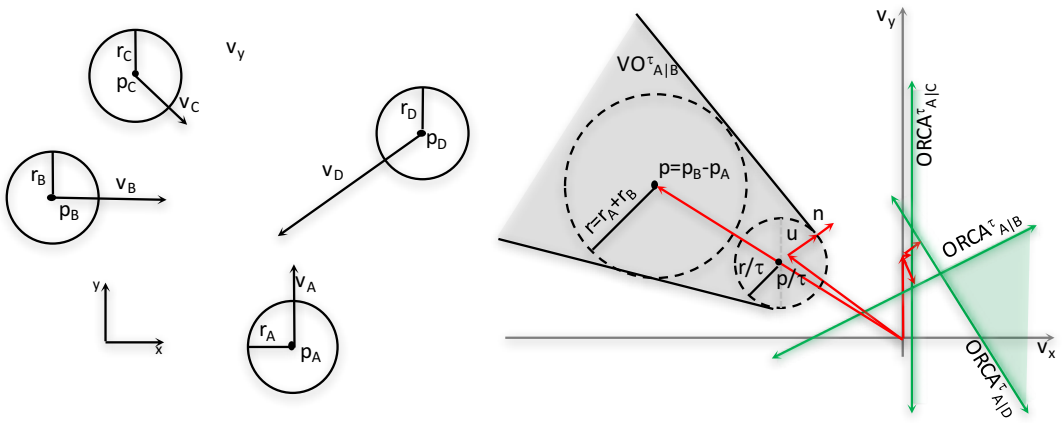


Figure 4: An example 2D navigation scenario with four robots on a collision course depicted on the left. On the right, the corresponding *velocity obstacle* $VO^{\tau}_{A|B}$ for $\tau = 2$ (gray), ORCA half-planes (green lines) and the region of allowed velocities (green area) calculated from the perspective of robot A.

planning. A more suitable approach it to use a specialized collision checker such as the OBBTree algorithm [14]. The OBBTree relies on spatial subdivision of the 3D space and constructs a tree of Oriented Bounding Boxes (OBBs). It allows for efficient collision checking not only for a single state but also whole 3D path segments (solved analytically).

The system presented in this paper takes advantage of a custom implementation of OBBTree collision checker which is tailored to the UAV domain. Detailed description of this implementation is presented in [15].

The path planners integrated in the presented system generate plans in form of a segmented cubic polynomial curves which are then executed by a reactive collision avoidance controller. Fig. 2 depicts an example path consisting of 3 segments. The plan is initially collision-free (black splines) with respect to the current map of the environment encoded in the OBBTree collision checker. As the mission is being executed the new point cloud data provided by a range sensor is added to the OctoMap (stripped rectangle). The process of building new OBBs for collision checking is time consuming, therefore adding of new OBBs into the OBBTree structure is performed incrementally and limited to areas directly influencing the current path (red square). After the OBBTree update is completed the newly generated paths are again collision-free (blue spline).

C. Reactive Collision Avoidance - 3D ORCA

The *Reactive Collision Avoidance* (RCA) module implements the Optimal Reciprocal Collision Avoidance (ORCA) algorithm which is used for fast real-time sense-and-avoid behavior as well as plan execution. ORCA addresses the problem of navigation for multiple robots operating in a common environment. The algorithm works in velocity space and each robot uses relative position and velocity to independently and simultaneously select a new velocity in order to ensure collision-free navigation for at least a preset amount of time.

Consider a 2D navigation example scenario presented in Fig. 4 where four robots $R_{i=A..D}$ are represented as discs with radius $r_{i=A..D}$ located at positions $p_{i=A..D}$. ORCA defines a *velocity obstacle* $VO^{\tau}_{A|B}$ as a set of relative velocities v for robot R_A with respect to robot R_B that will lead them to collision within a time window τ when keeping their current velocities.

$$VO^{\tau}_{A|B} = \{\boldsymbol{v} | \exists t \in [0,\tau] : : t\boldsymbol{v} \in D\left(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B\right)\} \tag{1}$$

where $D\left(\boldsymbol{p}, r\right)$ is an open disc of radius r centered at p:

$$D\left(\boldsymbol{p}, r\right) = \{\boldsymbol{q} | \|\boldsymbol{q} - \boldsymbol{p}\| < r\} \tag{2}$$

To guarantee collision free navigation, robots R_A and R_B have to choose relative velocities $v_A - v_B$ and $v_B - v_A$ that are outside of $VO^{\tau}_{A|B}$ and $VO^{\tau}_{B|A}$, respectively. Ultimately there are infinitely many pairs of velocities meeting this criteria and in ORCA, pairs of sets are chosen that maximize the allowed velocities close to individual *optimization velocity* (i.e. preferred velocity) imposed by a control system on each robot. Formally the set of collision free velocities of robot A imposed by robot B is a half-plane defined as follows:

$$ORCA^{\tau}_{A|B} = \left\{\boldsymbol{v} | \left(\boldsymbol{v} - \left(\boldsymbol{v}^{opt}_A + \frac{1}{2}\boldsymbol{u}\right)\right) \cdot \boldsymbol{n} \geq 0\right\} \tag{3}$$

where $\boldsymbol{v}$ and $\boldsymbol{v}^{opt}_A$ are current robot velocity and its optimization velocity, respectively. u is a vector from $\boldsymbol{v}^{opt}_A - \boldsymbol{v}^{opt}_B$ to the closest boundary of velocity obstacle $VO^{\tau}_{A|B}$ and the half-plane $ORCA^{\tau}_{A|B}$ is starting at the point $v^{opt}_A + \frac{1}{2}\boldsymbol{u}$. Note that each robot takes half of the responsibility of avoiding the other by applying $\frac{1}{2}\boldsymbol{u}$ of minimal velocity change.

In case of n-robot navigation scenario, each robot calculates a set of half-planes imposed by other robots based on relative position and velocity information. The intersection of the half-planes is calculated and a new velocity is chosen to minimize the following function:

$$ORCA_A^\tau = D\left(\mathbf{0}, v_A^{max}\right) \bigcap \left(\bigcap_{B \neq A} ORCA_{A|B}^\tau \right) \quad (4)$$

$$v_A^{new} = \underset{v \in ORCA_A^\tau}{\operatorname{argmin}} \|v - v_A^{pref}\| \quad (5)$$

The calculation is done using quadratic programming and in case it is infeasible the problem is relaxed by decreasing τ to ensure collision free navigation.

In the system presented in this paper a 3D variant of the ORCA algorithm is used. It is partially based on RVO2-3D implementation [16] with an extension for handling static obstacles proposed in [8], that is a static obstacle perceived in the environment is treated as a non-collaborative agent and the UAV will apply full value $\boldsymbol{u}$ of minimal velocity change instead of $\frac{1}{2}\boldsymbol{u}$.

The *Reactive Collision Avoidance* module actively monitors point cloud sensor data and selects a number of closest points as a potential obstacle for collision avoidance (i.e. obstacle points, see Fig. 5). The obstacle points are also used for checking potential collisions with the currently executed global plan (see Section II-E). When the original plan becomes invalidated during the flight due to a newly perceived obstacle and a new plan in not yet available, the use of the reactive controller (i.e. ORCA) will guarantee collision free operation.

D. Low-Level Control

The task of the *Low-Level Control* module is to execute a set of velocity commands provided by the *Reactive Collision Avoidance* component. The velocities are generated based on the set of sub-goals as described in the following section.

A velocity control mode is a typical functionality offered by UAV platforms implemented in their low-level control systems. It allows for flying with velocities given as a 3-dimensional vector. When all components of the velocity vector are zero the UAV platform remains stationary (i.e. hovers). Specific platform used in this paper is described in Sec. III-A and details of its low-level control system are provided in [17].

E. Execution Coordinator

The role of the *Execution Coordinator* (EC) is to manage the interactions between different functional components of the presented framework. Fig. 5 depicts those interactions in detail including the data flow. The EC is implemented as a state machine with 5 states: *Wait for command, Plan, Send to 3D ORCA, Monitor and Update OBBTree*. Transitions between each state are triggered by events which are depicted in Fig. 6.

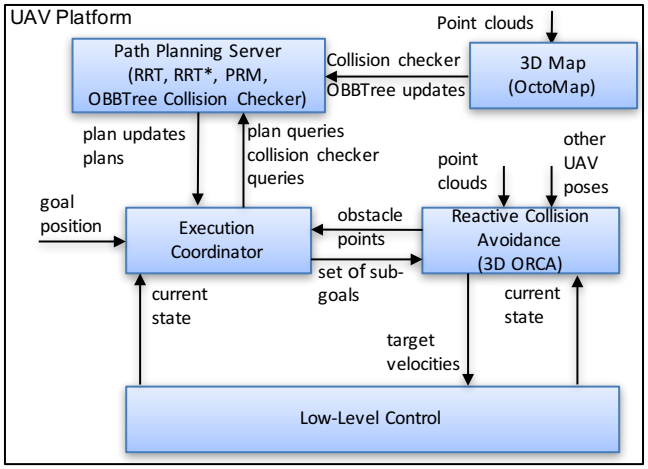


Figure 5: Interaction between different functional modules during a mission execution.

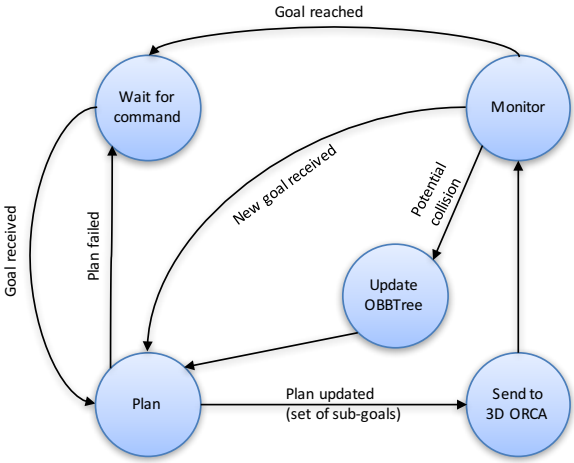


Figure 6: State machine used for mission execution in the Execution Coordinator.

A mission execution starts after receiving of a goal position. The EC queries the Path Planning Server (PPS) with the initial and goal positions (*Plan* state). As previously described the plan consists of a segmented cubic polynomial curve (Fig. 2). The length of a segment depends on the platform's flight capabilities. Furthermore, each segment of the plan is sampled along the path at specified distances and the resulting positions are used as sub-goals (Fig. 2, green dots) which are provided to the reactive collision avoidance controller for execution (*Send to 3D ORCA* state). Both, the segment length and the density of the sub-goal positions depend on the platform and control system capabilities and are chosen empirically.

Figure 7: LinkQuad platform with an Asus Xtion PRO depth camera.

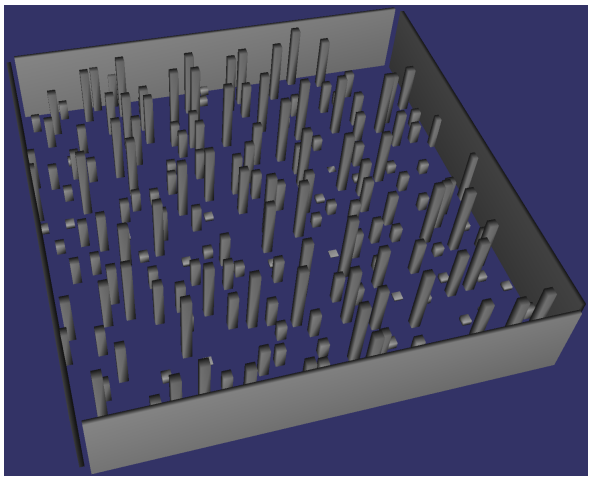


Figure 8: A world model example generated for the simulation.

While the current sub-goal positions are being followed by the 3D ORCA algorithm, the EC monitors for potential plan violations caused by the perceived obstacles (*Monitor* state). The check for a potential collision is based on a number of obstacle points (i.e. a set of closest points to the UAV position, cf. Section II-C) which are selected from each point cloud reading. If any of the points are within a predefined safety distance to the currently executed path, the EC triggers OBBTree collision checker update (*Update OBBTree* state). For efficiency reasons the new OBB data structure is generated only in a selected area in the direction from which the latest sensor data was acquired (cf. Fig. 2, red square).

When the OBBTree update has been completed the EC queries the PPS for a new collision-free plan based on the extended knowledge of the environment (*Plan* state). Note that the collision checker data structures are continuously updated by adding new OBBs as the UAV is perceiving new obstacles. The system will make use of the gained map knowledge and effectively rely less on the reactive control for avoiding static obstacles and rely more on the optimal plans generated by the path planner. The execution continues as described until the final goal position is reached. This results in a transition into the *Wait for command* state. Alternatively, when a new goal position is received during the mission execution, a transition to the *Plan* state is triggered.

III. Experimental validation

This section presents results of experimental validation of the proposed framework. First, the UAV platform used is presented. Followed by the description of the two types of experiments that were performed: (i) a set of simulations in randomly generated dense worlds and (ii) a real flight in an indoor environment. The first type of the experiment was performed to show successful collision-free operation of multiple UAVs in unknown dense environments. The second type shows application of the proposed framework on a real UAV platform.

A. UAV Platform

The UAV platform used in the evaluation of the proposed framework is the LinkQuad - a highly versatile autonomous Micro Aerial Vehicle. The platform's airframe is characterized by a modular design which allows for easy reconfiguration to adopt to a variety of applications. Thanks to its compact design (below 70 centimeters tip-to-tip) the platform is suitable for both indoor and outdoor use. Depending on the required flight time, one or two 4.6Ah batteries can be placed inside an easily swappable battery module. The maximum take-off weight of the LinkQuad is 2kg with up to 600g of payload and an endurance of up to 30 minutes.

The LinkQuad is equipped with an advanced flight control board - the LinkBoard [18]. The LinkBoard has a modular design that allows for adjusting the required computational power depending on mission requirements. Due to the available onboard computational power, it has been used for computationally demanding applications such as the implementation of an autonomous indoor vision-based navigation system with all computation performed on-board. In the full configuration, the LinkBoard weighs 30 grams, has very low power consumption and has a footprint smaller than a credit card (45mm $\times$ 80mm).

The system is based on two ARM-Cortex microcontrollers running at 72MHz (or 168MHz) which implement the core flight functionalities and optionally, two Gumstix Overo boards for user software modules. The LinkBoard includes a three-axis accelerometer, three rate gyroscopes, and absolute and differential pressure sensors for estimation of the altitude and the air speed, respectively. The LinkBoard features

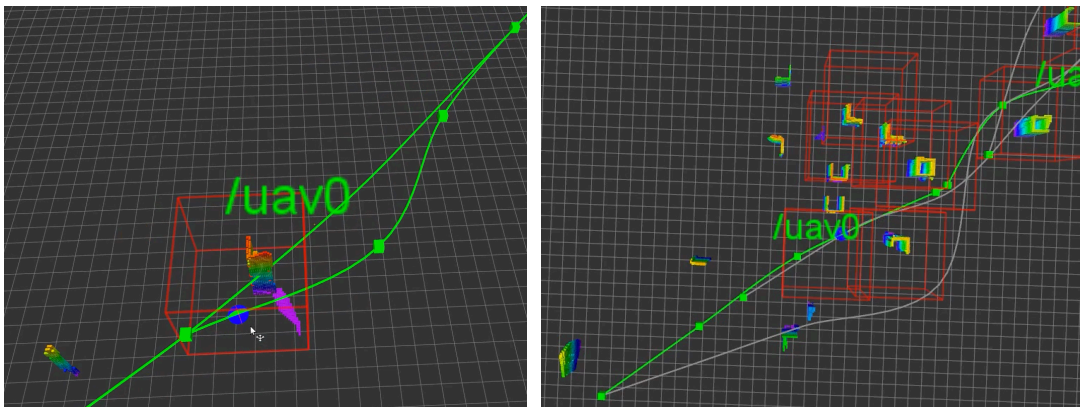
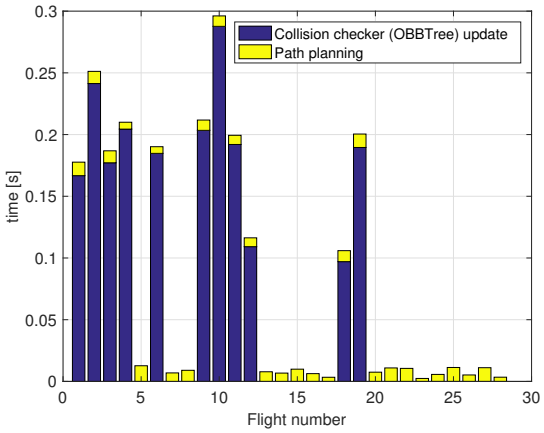
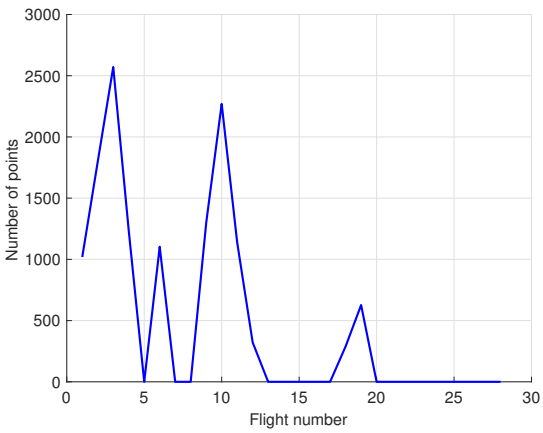


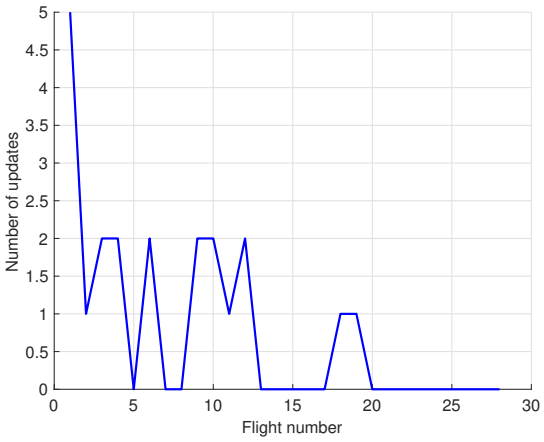
Figure 9: Visualization of an example navigation mission executed in simulation. Initial and later stages of the flight are depicted on the left and right, respectively. Original and repaired plans are represented by green splines. Gray splines visualize plans from previous flights within the mission. Red cubicles mark the update areas for the OBBTree collision checker. Multi-color voxels show occupied areas in the OctoMap.



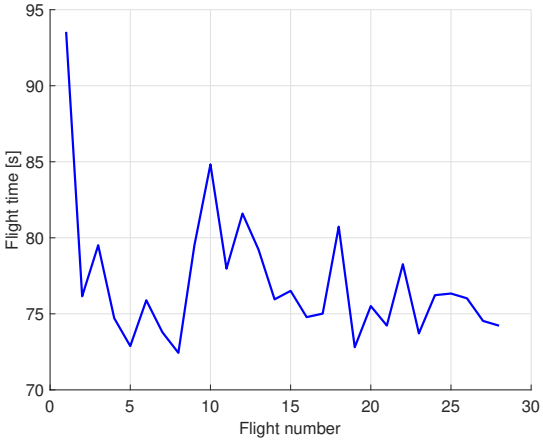
(a) Average times for path planning and collision checker (OBBTree) updates.



(b) Average point cloud sizes for collision checker (OBBTree) updates.



(c) Number of collision checker (OBBTree) updates.



(d) Total flight times.

Figure 10: Empirical results for 28 navigation missions executed in an example simulation environment.

a number of interfaces which allow for easy extension and integration of additional equipment. It supports various external modules such as laser range finders, analogue and digital cameras, GPS receivers, and magnetometers. The LinkQuad can also be equipped with an external Intel NUC computer which is connected to the LinkBoard.

Commonly used sensors for sensing of the environment in the case of small scale platforms are sweeping laser range finders (as presented in [19]) or cameras in different configurations (monocular or stereo). For the purpose of this work an Asus Xtion PRO[1] depth camera is used. Compared to a laser range finder, the sensor delivers range information within the field of view of the camera. It uses structured light technology and illuminates the environment with infrared light to obtain the depth information.

The dynamic collision avoidance framework has been implemented in Robot Operating System[2] (ROS) and integrated with the LinkQuad. Each of the five components were implemented as separate ROS nodes. Exchange of information was realized using ROS topics and common visualization tools were used for monitoring progress of the missions. The framework is a part of the hybrid deliberative/reactive architecture (HDRC3) presented in [20].

B. Experimental evaluation in simulations

The first set of experiments was performed in simulation executed on a single PC equipped with a 3.5GHz Intel Xeon E5-1620 CPU. The Modular OpenRobots Simulation Engine (MORSE) [21] was used for generating the depth camera sensor data based on the model of the Asus Xtion PRO sensor and the position and orientation of the UAV in the generated world model. Simulation of the UAV platform behavior was based on the flight dynamics model of the LinkQuad platform and its low-level control system allowing for easy switching between simulations and real-flight experiments.

A set of 20 dense world models (50m $\times$ 50m) were generated for the evaluation. Each model includes 200 uniformly distributed obstacles in form of rectangular blocks with 1m $\times$ 1m size and 1m $-$ 10m height. An example world model used in the simulation is depicted in Fig. 8.

Each simulation run included five UAVs navigating in an environment without any prior map knowledge. Four of the UAVs were commanded to fly to randomly generated positions within the world. While the remaining UAV was commanded to fly diagonally between two corners of the environment up to 30 times. A safety distance of 1m was used both for reactive collision avoidance and path planning.

In all the experiments the UAVs successfully navigated in the environments and avoided at all times collisions with static and dynamic (other UAVs) obstacles. The safety distance was never violated with respect to the obstacles or between UAVs. Fig. 9 shows a visualization of an example mission executed with the initial and later flight stages presented on the left and right, respectively. In the beginning of the experiment the path planner provides a plan to the UAV (green spline). Initially the UAV has no information of the environment thus the path is a straight line. As the flight progresses new objects in the environment are perceived and added to the 3D Map (OctoMap) represented as multi-color voxels (i.e. occupied space). Potential obstacles on the way of the currently executed plans are added to the collision checker OBBTree structure in form of new OBBs in selected areas (red cubicle). The path planner generates new plans taking into account the updated knowledge of the environment. The mission execution continues and the more information about the environment is gained the more optimal plans are generated by the path planner resulting in shorter flight times. This eventually leads to the situation that the reactive collision avoidance module will mostly take care of avoiding collisions with other UAVs as all the obstacles in the environment will be known and plans will be collision-free in respect to static obstacles.

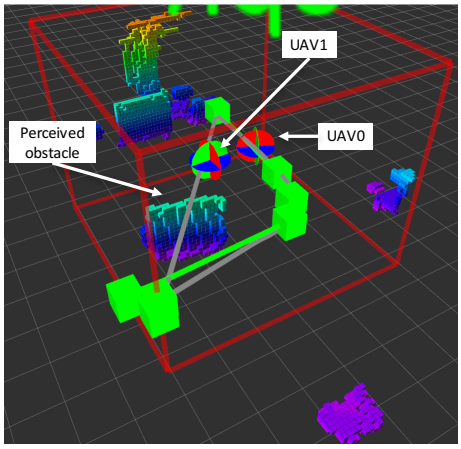


Figure 11: Visualization of the real flight experimental setup and mission result. Original and repaired plans are represented by gray and green lines, respectively. Red cubicle marks the update area of the OBBTree collision checker. Multi-color voxels show occupied areas in the OctoMap.

Fig. 10 presents quantitative results from an example simulation run which included 28 diagonal flights between two opposite corners of the environment. The more flights are performed in the environment the more obstacles are represented in the OBBTree collision checker data structures. This results in fewer and less time consuming collision checker updates. As a result, obtaining new collision-free optimal plans from the path planner becomes quicker (Fig. 10a-10c). After performing 19 flights no more updates are necessary. Since the plans generated by the path planner become more optimal in relation to the static obstacles, the flight time

[1]Asus: http://www.asus.com/Multimedia/Xtion_PRO/
[2]ROS: http://www.ros.org/

decreases (Fig. 10d). Initial flight times are as high as 94s down to 75s on average for later flights.

C. Experimental evaluation in real flight

The real-flight evaluation was performed indoors with the use of a commercial motion capture system from Vicon[3]. The system consists of ten T10 and six T40 cameras. The operation volume of the system is approximately $10\times10\times5$ meters. The cameras illuminate the scene with infrared light which is reflected by a set of markers attached to a tracked physical object (see Fig. 7). In the real-flight experiment presented in this paper the UAV position and heading information is used at the rate of 10 Hz for autonomous flight as well as at 25 Hz for the mapping purpose.

The UAV operates in an indoor environment with one static and one dynamic obstacle. The former, is a 2m high white-board positioned in the middle of the flight area. The latter is a simulated UAV executing a continues flight between two points in the environment. The target altitude and velocity for the flight of both UAVs was set to 2m and 1m/s, respectively. Fig. 11 depicts a visualization of the experimental setup and the mission progress. Start and goal positions are marked as yellow circles for both UAVs. The real platform (UAV0) was commanded first to fly behind the obstacle crossing the other UAV's path and then back to the start position. The simulated platform (UAV1) was commanded to continuously fly between two points parallel to the white-board obstacle.

The UAV0 successfully executed the mission finding a path around the static obstacle, as well as avoiding collisions with the UAV1 (Fig. 11). As can be seen, the original plan (grey line) was invalidated when the obstacle was perceived. After the OBBTree collision checker update was completed (red square area), a new plan was found (green lines). Fig. 12 depicts the actual flight trajectories including a reactive collision avoidance maneuver performed by the UAV0. The safety distance which was set to 1.4m for this flight experiment was never violated confirming the validity of the proposed dynamic collision avoidance framework in real-flight conditions.

IV. Conclusion and future work

In this paper we have presented a framework for solving the problem of safe navigation for multiple UAVs operating in unknown environments. The framework combines reactive controller based on the Optimal Reciprocal Collision Avoidance (ORCA) algorithm with sample-based path planning techniques. We have discussed the aspects of integrating a fast sense-and-react type controller with deliberative path planners which are capable of generating globally optimal plans. In the system described we have shown how the large amount of noisy range sensor data can be processed in order to facilitate the usage of path planners. This is done by first using an octree-based map representation handling the problem of noisy sensor data. Second, by performing incremental collision checker updates to assure that potential obstacle information is considered while the path planners repair or generate new plans. Experimental evaluations in simulation and in real flight tests were performed using the presented system and the results confirmed the applicability of the method to the problem of multi-UAV navigation in unknown environments.

The future work will include performing simulations involving more UAV platforms and larger environments since the used simulation engine allows for running in a distributed manner. Additionally, focus will be put on applying the proposed framework to navigating outdoors with the use of different types of sensors.

[3]Vicon: http://www.vicon.com

Acknowledgment

This work is partially supported by the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT network organization for Information and Communication Technology, and the Swedish Foundation for Strategic Research (Smart Systems: RIT 15-0097). The authors would like to acknowledge the software development support of Tommy Persson.

References

[1] L. E. Kavraki, P. Švestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. ICRA*, 2000.

[3] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning." Computer Science Department, Iowa State University, Tech. Rep., 1998.

[4] M. Wzorek, J. Kvarnström, and P. Doherty, "Choosing path replanning strategies for unmanned aircraft systems," in *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2010.

[5] M. Wzorek, "Selected aspects of navigation and path planning in unmanned aircraft systems," Licentiate thesis, Linköping University, 2011.

[6] G. Conte, S. Duranti, and T. Merz, "Dynamic 3D path following for an autonomous helicopter," in *Proc. IFAC Symp. on Intelligent Autonomous Vehicles*, 2004.

[7] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[8] D. Alejo, J. Cobano, G. Heredia, and A. Ollero, "Optimal reciprocal collision avoidance with mobile and static obstacles for multi-uav systems," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 1259–1266.

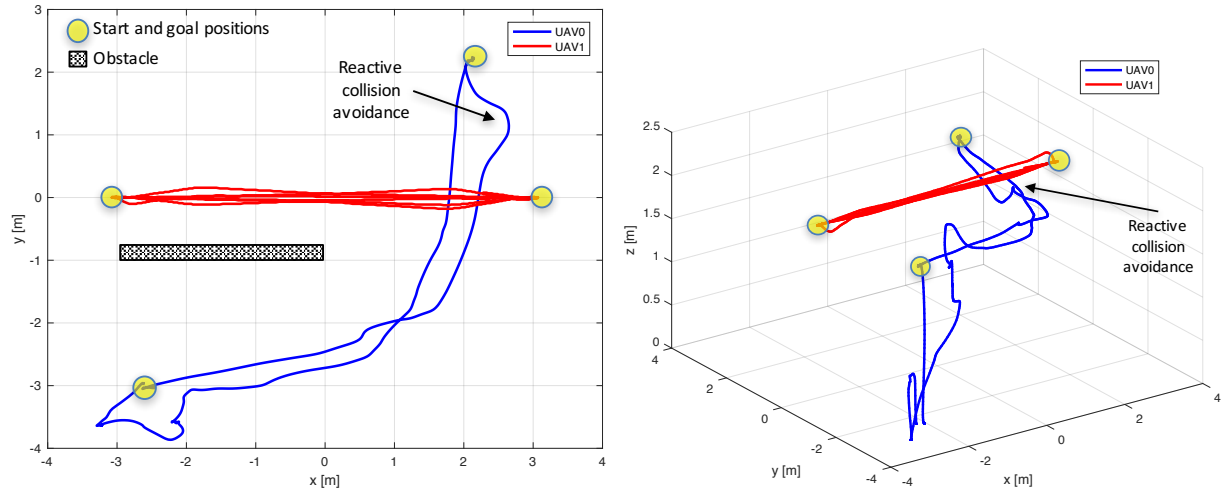


Figure 12: Real flight experiment results showing the actual trajectories executed during the flight.

[9] K. Yang, S. K. Gan, and S. Sukkarieh, *An Efficient Path Planning and Control Algorithm for RUAV's in Unknown and Cluttered Environments*. Dordrecht: Springer Netherlands, 2010, pp. 101–122.

[10] D. Alejo, J. A. Cobano, G. Heredia, and A. Ollero, "Collision-free 4d trajectory planning in unmanned aerial vehicles for assembly and structure construction," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 783–795, 2014.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[13] ——, "Incremental sampling-based algorithms for optimal motion planning," in *Robotics: Science and Systems (RSS)*, June 2010.

[14] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 171–180, 1996.

[15] P.-O. Pettersson, "Using randomized algorithms for helicopter path planning," Licentiate thesis, Linköping University, 2006.

[16] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha, "Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation," 2011.

[17] P. Rudol and P. Doherty, "Bridging the mission-control gap: A flight command layer for mediating flight behaviours and continuous control." in *IEEE International Symposium on Safety Security and Rescue Robotics*, October 2016.

[18] M. Wzorek, P. Rudol, G. Conte, and P. Doherty, "Linkboard: Advanced flight control system for micro unmanned aerial vehicles," in *IEEE International Conference on Control and Robotics Engineering*, 2017.

[19] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics (T-RO)*, vol. 8, no. 1, pp. 90–100, 2 2012.

[20] P. Doherty, J. Kvarnström, M. Wzorek, P. Rudol, F. Heintz, and G. Conte, "Hdrc3 - a distributed hybrid deliberative/reactive architecture for unmanned aircraft systems," in *Handbook of Unmanned Aerial Vehicles :*, 2014, pp. 849–952.

[21] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular openrobots simulation engine: Morse," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.