PROCESS AUTOMATION OF WAREHOUSE INSPECTION USING AN AUTONOMOUS UNMANNED AERIAL VEHICLE

JAROMIR STANKO, FILIP STEC, JEZEF RODINA

Slovak University of Technology in Bratislava, *Faculty of Electrical Engineering* and Information *Technology*, Institute of Robotics and Cybernetics

DOI: 10.17973/MMSJ.2022_10_2022063

jaromir.stanko@stuba.sk

Inventory management applications using unmanned aerial vehicles (UAVs) appear to have the highest potential for use in warehouse operations. They remove the need for manual inspection, which is in most cases performed in heights using ladder, pen and paper. Moreover, it provides a competitive edge in the field of logistics while decreasing costs and improving stock management efficiency. Deployment of autonomous UAVs in an indoor environment puts higher requirements for the level of autonomy due to the more complex nature of the environment (narrow crossings, absence of Global Navigation Satellite System (GNSS), smaller manoeuvring space, dynamic environment, etc. ...). In this paper, we would like to introduce a system for automation of planning, managing, and executing a warehouse inspection. The goal is to provide an easy-to-use framework of the inspection task that any warehouse operator is capable of using it. The system consists of two main parts - user interface and path planner. Using a created human-machine interface (HMI), an operator is able to define the workspace and the inspection task. The path planning module takes as an input the definition of the inspection task defined by the operator and plans an optimised collision-less path. The operator can visualise the generated mission, launch it, observe the mission execution progress in real time and receive the mission results. The functionality of the system is verified by a simulation as well as by test flights made in an environment simulating a warehouse by a real UAV called AV Discovery built by Airvolute s.r.o.

KEYWORDS

Autonomous UAV, Path planning, Mission planning, Warehouse inspection, Automation, User interface

1 INTRODUCTION

The autonomous movement of UAVs in the indoor environment is a problem that is often solved in mobile robotics. Different approaches to a given problem differ themselves depending on the application, the size of the UAV (as a platform), available hardware and algorithms used for simultaneous localization and mapping (SLAM) and path planning. Despite existing technologies and rapid development in the field of autonomous UAVs, their application in warehouses is not yet widespread. There are not many companies offering solutions for warehouse inspections based on autonomous UAVs. Deployment of their solution is often in the development or experimental phase [Wawrla 2019]. We are not aware of any study dealing with the deployment of the autonomous UAVs for warehouse inspection that can evaluate and compare existing solutions. Further information on the technologies used by the companies, their integration and experimental results are usually not freely available.

Companies are approaching the deployment of autonomous UAVs for warehouse inspection in various ways. The first approach is autonomous UAV moving independently in the warehouse, collecting application data that are sent to a server. Another approach, that is more adapted to the warehouse environment, is the UAV wired-connected to the automatic guided vehicle (AGV). The AGV carries out an inspection at ground level and performs all data processing. It also serves as a power supply for the UAV. The UAV serves as a flying camera. It performs the inspection of the upper floors and sends all the data to the AGV (used by doc. Innovation, RAWview). Each of the approaches has its pros and cons. While the first approach is more versatile (UAV can be adapted to a different kind of applications outside of the warehouse), in the second approach the UAV coupled with the AGV is able to continuously perform the inspection for hours without interruption.

In [Beul 2017] the authors present an autonomous UAV built for the warehouse inspection. In the paper, the authors focus on the control system architecture, their approach to localization, mapping and path planning which is described in detail. However, the paper does not focus on mission management and visualisation, so it only provides a brief description of how the mission is planned and how the data is visualised for the operator. The approach to static environment representation and application-related waypoints optimization is similar to the one described in this paper. Based on the provided description of the current state, their mission planning and visualisation is suitable for development and testing purposes, but not ready for deployment.

The system introduced in this paper focuses on the management of warehouse inventory and surveillance tasks. For the purpose of visualising and controlling this task, an HMI was created. The HMI is built as a web application that allows the operator to easily setup and execute an inspection task, also called mission. A warehouse can be considered a structured environment with a limited number of relevant objects, which allows to predefine the environment (dimensions, static obstacles, no flight zones …), placement of racks, pallets, docking stations and others. When a model of the warehouse is created, the operator can specify the mission requirements, which are processed and performed by the UAV.

Before the UAV performs the mission, the path planner must generate a suitable path for it. This process is fully automatic, not requiring other inputs from the operator. Firstly, the path planner sorts racks so that the path is optimised for length. Next, when the racks are sorted, a collision-less path is found by the path planning algorithm connecting all selected inspection sites. The resulting path is sent back to the HMI where it is visualised and checked by the operator. Architecture and detailed description of the application, the path planner and the HMI are covered in upcoming chapters.

2 DESCRIPTION OF INSPECTION APPLICATION

This paper focuses on automation of an inspection task of a warehouse using autonomous drones. For this application, we used a drone named AV Discovery (Fig. 1) built by a drone company named Airvolute s.r.o. This drone is capable of localization [Mraz 2020] in the confined and structured

environment of the warehouse. The drone is also equipped with sensors which are used to collect data relevant to the inspection.

The inspection begins by an operator placing the drone on a starting point and turning it on. The operator then chooses the parameters of the inspection, from which the mission is generated. After the start of the mission, the drone takes off to a predefined altitude where it can see using onboard camera sensor a specific fiducial marker that is used to precisely position the drone in the global frame of the warehouse. After this, the drone starts to follow the predefined path that connects all selected inspection sites. The onboard sensors collect the data which can be later translated into items stored in the warehouse and their locations. After finishing the inspection, the drone returns to a closest landing site, which concludes the mission. The results of the inspection task can then be passed to an upper management system of the warehouse.

Figure 1. AV Discovery performing inspection task in a simulated warehouse

3 CHAPTER HUMAN MACHINE INTERFACE

The purpose of the HMI is to control and visualise the automated process of warehouse inspection. The HMI is built as a web application making it available on any web browser capable device. The focus of the HMI is to produce an easy and user-friendly experience of running an inspection task. Being an autonomous UAV, the control elements are comparatively smaller to the visualisation part. In this section, firstly, the description of the architecture of the HMI is outlined. Then the sequence of task preparation, mission selection and observation, and results visualisation are explained. The design of the application is described in more detail in the paper "Designing Human-Machine Interface for UAVs in a structured environment using Robotic Operating System (ROS) and Flask" [Stec 2020].

3.1 *Architecture*

The HMI connects the operator with the autonomous drone. The application consists of two parts as seen in Fig. 2: backend and frontend. The backend is built using Python programming language and runs as a server on the drone. It provides a middleware for the communication between the drone and the HMI. The frontend is built using web application languages: HTML, CSS, JavaScript. The frontend is the graphical user interface displayed in a web browser.

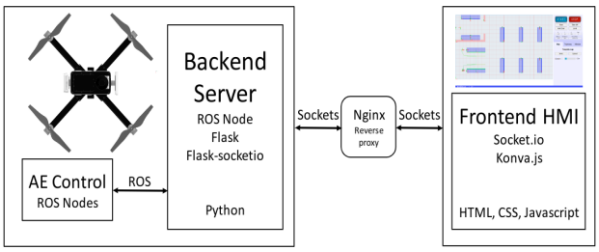

Figure 2. Overview of the architecture of the HMI application

In order for the backend to be able to communicate with the drone, the drone's communication protocol is adopted. The drone is built using ROS, a set of software libraries and tools that facilitate building robot applications. ROS uses its own protocols for data communication called TCPROS and UDPROS. The application creates a ROS node, which allows the backend to use the ROS defined message transport types called services, topics and actions.

The communication between the application backend and frontend is handled by a microframework called Flask. It provides application programming interface (API) end-points for socket communication. The exchanged messages are in JavaScript Object Notation (JSON) format, which is natively supported in both Python and JavaScript. Because the application runs on the drone as a server, the frontend has to be made accessible to the connected operator, which is done by reverse proxy handled by Nginx. It sits in front of the backend application and forwards client (e.g. browser) requests to the application.

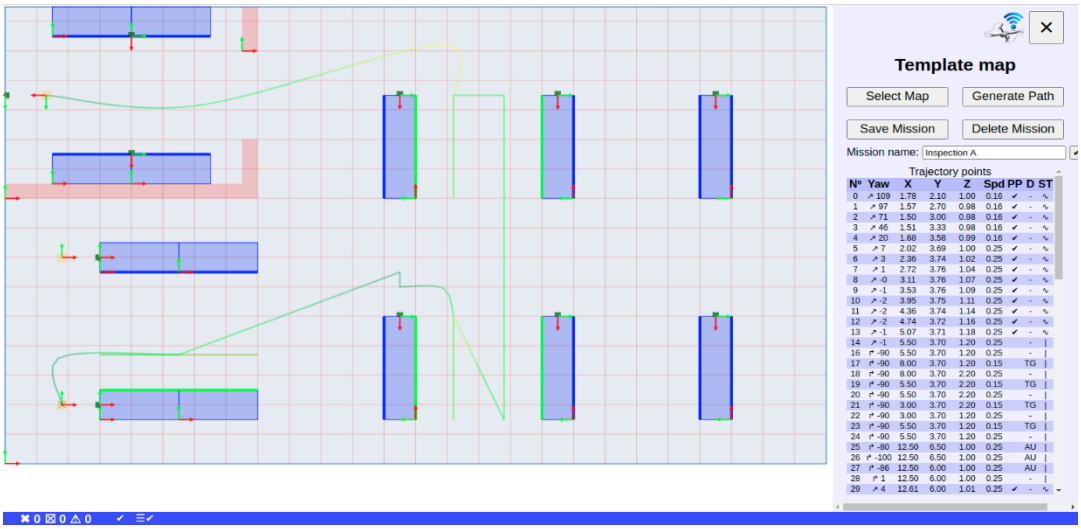

Figure 3. GUI designed for the preparation of the mission.

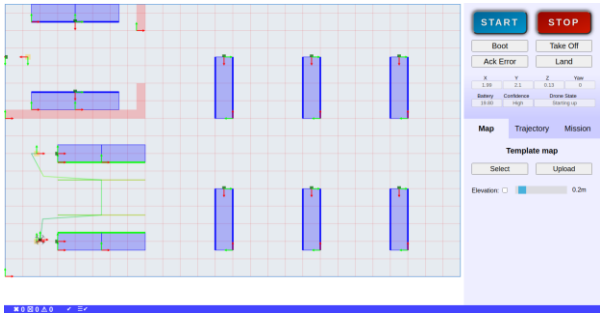

Figure 4. Graphical user interface of the HMI

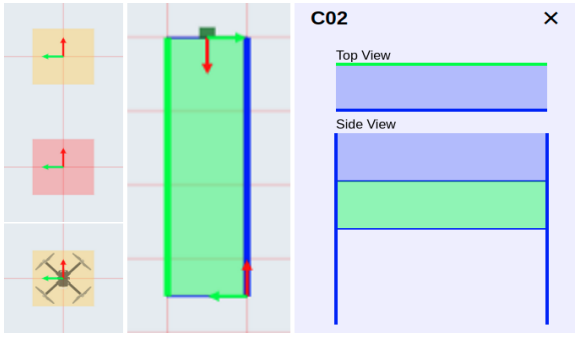

Figure 5. GUI for selection of the inspection specifications: docking station with its three states, selected rack, selected inspection side and inspection floor

3.2 *Mission selection and observation*

The application provides a graphical user interface that is used to prepare inspection tasks as we can see in Fig. 3. The mission preparation consists of four parts: selection of the map of the environment, selection of the inspection, generation of the global path, and revision of the generated mission. The map of the warehouse represents a structured environment with a known set of objects and dimensions. The important objects of a warehouse for the application of warehouse inspection are: racks, docking stations, fiducial markers, and restricted areas. The locations and dimensions of these objects are defined in a json map file that is saved in the drone's filesystem. Based on this map, an octomap is created, which defines the obstacles and limits of the space with specific resolution.

The map can be selected in the right panel using the Select Map button. This will show the list of all the maps that are saved in the drone's filesystem and are ready to be used for a mission. The selected map is drawn in 2D canvas in the map view part of the graphical user interface (GUI). The objects in the map are well distinguishable by colour. The racks are drawn as blue rectangles with a thick line specifying a side from which the rack can be inspected. The docking stations are orange when available, red when disabled/occupied, and orange with an image of a drone when selected as the starting point of the mission (Fig. 5). The fiducial markers are small green rectangles and the restricted areas are drawn in red.

With the selected map, starting point and inspection sites, the mission is prepared to be generated. For this serves the button Generate Path, which will send all needed information to the path planner. During the planning, a progress bar is displayed showing the progress state of the planning. When the path is planned, it is displayed in the map with a line connecting all points of the path. The colour of the line stands for relative altitude of the point in the warehouse with blue colour being low and red being high with rainbow colours in between. The generated path is also shown in a table specifying each target's position, orientation, velocity, type of detection and type of movement. The generated path can be saved in the drone's filesystem using the Save Mission button by specifying the name of the mission. This mission can then be loaded any number of times and executed.

Every saved mission holds all the information that was used to generate it. This allows it to notify the operator when the same selection was specified. The operator can choose to cancel the path planner or generate a duplicate mission or rewrite the existing mission with a newly generated one. This allows high flexibility with generating missions and to save time on duplicates.

3.3 *Mission selection and observation*

As seen in Fig. 4, the GUI contains three tabs in the right panel. These tabs are shown in Fig. 6. The Map tab is used to select the map file in which the mission will be executed and to display information about map objects. The Trajectory tab is used to select the mission from the list of saved missions for the specific map and to show the table of targets for the UAV navigation. The Mission tab is designed to be shown during the inspection task execution by displaying a live video feed from one of the drone's cameras and the table of all detected items of the inspection.

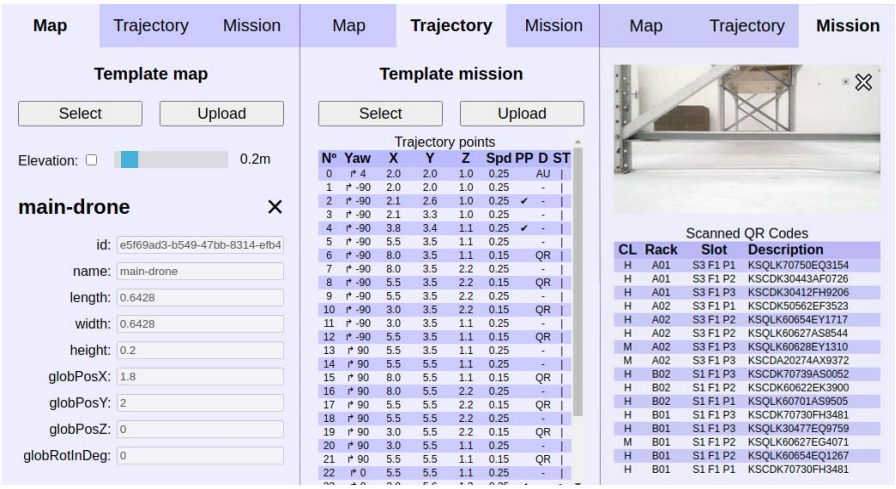

Figure 6. Map, Trajectory and Mission tabs respectively

When the mission is selected, the drone is ready to perform its action. To start the mission execution, the operator holds down the big Start button, shown in Fig. 4, for at least two seconds, so that it cannot be started by accident. This will send the command to the drone, which will begin its autonomous task. The drone AV Discovery is fully autonomous and does not require any other assistance while performing the mission. Only in case of emergency, the operator can push the Stop button for at least half a second to stop the drone.

During the mission, the position of the drone is periodically updated based on the real position of the drone in the space of the map as seen in Fig. 7. This allows the operator to easily follow the progress of the inspection and the movement of the UAV. After reaching a target, the trajectory line is also updated to point to the next target of the trajectory. The past drone's positions are periodically drawn into the map using small dots to show the real path the drone has taken. Some other important parameters can be seen in the information panel, as seen in Fig. 4, with the drone's actual position and orientation, battery voltage, camera confidence level for camera used for localization, and the state of the drone.

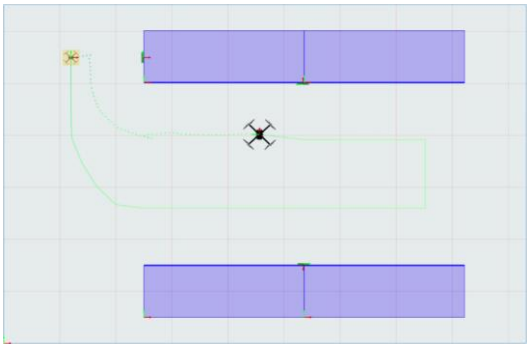

Figure 7. Visualisation of the drone performing its mission

3.4 *Mission results*

The result of the mission of warehouse inspection is a list of detected items in the warehouse. Each item consists of a detection confidence, location within the warehouse, and a description. The list can be seen in Fig. 6 in the Mission tab. The confidence shows how well the system has detected the specific item, which is determined by the number of detections in the images and the precision of location estimation. The location of the item contains the name of the rack and a slot description (rack side, inspection floor, floor part). This list is saved in the drone's filesystem in comma-separated values (CSV) format, which is a well compressed and easily parsed data format.

4 PATH PLANNER

The goal of path planning is to find a set of valid states that navigates an object from a start state to a goal state. It solves the problem of how to get from point A to point B. When planning warehouse inspection, another problem arises: where are points A and B. Purpose of the path planner is to find an optimal (or at least near-optimal) path, so the UAV is able to accomplish its mission autonomously. Path planner performs mission optimization by calculating an order in which the racks are going to be inspected. Then it focuses on finding a safe and well optimised path through the environment. This section describes the architecture of the path planner and the process of finding the resulting path.

4.1 *Architecture*

The path planner resided in the UAV, making the solution self-contained. Fig. 8 depicts the main structure of the path planner. Path planner communicates primarily with the HMI. Input to the path planner is the definition of the inspection task as it is described in chapter 3.2 - Mission preparation. The output is the resulting path that the UAV follows while executing the inspection task. The path planner consists of 2 submodules: path planning and mission planning.

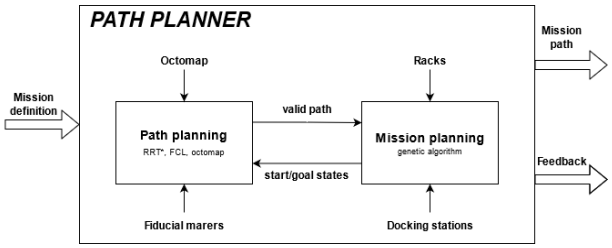

Figure 8. Block diagram of path planner

Due to the fact that the used UAV is built on ROS, the path planner is implemented as a ROS node with an action server ready to receive a request for planning from the HMI. While planning the path for the inspection task, the path planner provides feedback about the current state of planning so that the operator can keep track of it.

4.2 *Mission planning*

The UAV's flight time is limited by the amount of energy it carries onboard in a battery. Thus, it is desired to utilise the flight time as efficiently as possible. As it is mentioned in chapter 3.2, the mission definition contains the list of the racks that are supposed to be inspected and the list of docking station from where the UAV can take-off or land. In some cases, to pass through all the racks in the order they were received from HMI, the solution wouldn't be efficient because of unnecessary transitions leading to the longer resulting path.

Firstly, the mission planning submodule optimises the order in which the racks are to be inspected. For this purpose, a genetic algorithm is used. The inputs to the genetic algorithm are the entry points to each inspection site and the number of floors that are selected for that inspection site. A rack can be entered for inspection by the UAV either from left or right side and either from top or bottom floor. Also, the algorithm considers where the inspection of the previous rack finished, either left or right, based on the number of selected floors for the inspection site. Consider two racks facing each other that are subject to inspection. The inspection of the first rack starts from the left side of the bottom level of the rack and finishes at the same place in its top level. The inspection of the second rack starts from its left side in the top level and moves towards the bottom one. The distance travelled between opposite inspection sites is only the distance between them. Fig. 9 shows how the UAV would perform the inspection in this situation. The distance that the UAV has to fly during inspecting an inspection path is constant. Therefore, the criterion function of the genetic algorithm is the overall length of the path that is passed between the inspection sites. The result of the genetic algorithm is an ordered list of inspection sites.

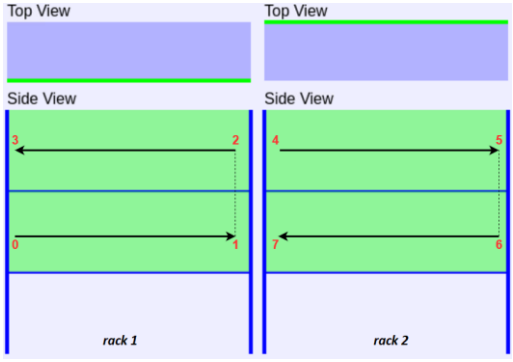

Figure 9. The UAV performs rack inspection by passing through waypoints in the given order

The mission planning submodule creates an ordered list of points (start/goal) from the result of the genetic algorithm. These are sequentially inputted into the path planning submodule to produce an optimised collision-free path through 3D space of the warehouse. Each waypoint of the path contains the information about position, orientation, maximum speed, and type of movement towards the target as can be seen in the Trajectory table in Fig. 6. The progress of processing this list of points is sent to the HMI as the feedback. Then the mission planning submodule combines the individual paths and adds additional information to the waypoints where the inspection occurs. The resulting path is sent back to the HMI.

4.3 *Path planning*

The goal of the path planning submodule is to find a path that connects all the received setpoints from the mission planning submodule and ensure collision-less transition through the environment. To find an optimal and collision-less path from start to goal, the rapidly-exploring random tree star (RRT*) planning algorithm is used.

RRT* is a sampling-based path planning algorithm derived from rapidly exploring random graph (RRG). Compared to the origin rapidly-exploring random tree (RRT), RRT* converges to the suboptimal solution [Karaman 2011]. It is thanks to the two features the RRT* algorithm introduced – nearest neighbour search and tree rewiring. Fig. 10 depicts a difference between the path found by RRT (left) and RRT* (right) and between the tree structure of those algorithms. The path found by RRT* tends to be straight and smooth compared to the path found by RRT. Implemented RRT* finds the path in 3D space and is optimised for length. It is used for global path planning in a static environment. Used collision detector [Pan 2012] detects a collision between the UAV's geometric model and the environment represented by octomap.

Localization system of the UAV uses fiducial markers for correction of the current position. Path planning submodule tries to help the localization system by ensuring that UAV will be able to detect fiducial markers if the marker is close to the path the UAV is taking. The path planning submodule receives the list of fiducial markers with their positions and orientations. When RRT* finds a path, the path is checked if any of the markers are close to this path. If there is such a marker, the origin path is modified so the UAV will get into the position from where it will have a direct view of the marker. This can be seen in Fig. 3, where the path goes from bottom left racks to a middle one by checking a fiducial marker (small green rectangle) on the top side of the middle rack.

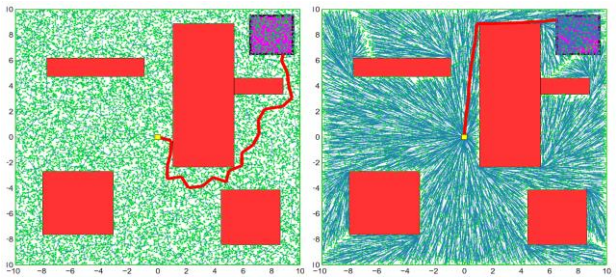

Figure 10. Comparison of solution path found by RRT and RRT* planning algorithm [Karaman 2011]

Path planning algorithm (RRT*) is implemented with the open motion planning library [Sucan 2012]. For collision detection the flexible collision library (FCL) is used. Octomap representing the warehouse as a working environment is created with the OctoMap library [Hornung 2013].

4.4 *Planning result*

The result of the path planner is the path the UAV will pass autonomously while performing the warehouse inspection. The planned path is optimised so the limited flight time is used efficiently and the UAV can inspect more racks per flight. Path planner ensures that the UAV will be able to scan a fiducial marker during its transition between racks if the marker is close enough to the planned path. The resulting path is sent to the HMI where it is visualised (Fig. 7) and to the control system [Rau 2020] which is passing position commands to the flight control unit (FCU).

5 CONCLUSION

In this paper, we presented the system for automation of warehouse inspection using an autonomous drone. The system consists of a HMI, mission and path planner. As a web application, the HMI with GUI allows an operator to easily define and control the execution of warehouse inspection. The overall architecture of the HMI and its use for path planning was described in detail.

Based on the inspection task defined in the HMI, the path planner finds an optimised and collision-less path that the UAV can autonomously follow while it performs the inspection task. Besides that, the path planner helps the UAV's localization system by modifying the path considering the fiducial markers placed in the warehouse. Thanks to that, the path planner ensures the UAV can detect fiducial markers that are close enough to the planned path and so the localization can perform correction of the actual UAV's position.

The resulting system was deployed onto autonomous drones called AV Discovery developed by Airvolute s.r.o. and flight tested.

ACKNOWLEDGMENTS

This publication was created thanks to the support from project 313011ATR9 – "Research and development of the usability of autonomous aircraft in the fight against the pandemic caused by COVID-19", number of the call: OII-VA/DP/2020/9.4-01, contract number: 5594984. This publication was also supported by the project VEGA 1/0599/20 - „Robust localization for Drones in Industry 4.0" and by the company Airvolute s.r.o.

REFERENCES

[Beul 2017] Beul, M., Krombach, N., Nieuwenhuisen, M., Droeschel, D. and Behnke, S. Autonomous Navigation in a Warehouse with a Cognitive Micro Aerial Vehicle, 2017. DOI:10.1007/978-3-319-54927-9_15

[Hornung 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. and Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees, 2013. Autonomous Robots, 34(3), 189-206. doi:10.1007/s10514-012-9321-0

[Karaman 2011] Karaman, S., Frazzoli, E. Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research, 30(7), 846-894. doi:10.1177/0278364911406761

[Mraz 2020] Mraz, E., Rodina, J. and Babinec, A. Using fiducial markers to improve localization of a drone, 2020. 23rd International Symposium on Measurement and Control in Robotics (pp. 1-5), doi: 10.1109/ISMCR51255.2020.9263754

[Pan 2012] Pan, J., Chitta, S. and Manocha, D. FCL: A general purpose library for collision and proximity queries, 2012. In 2012 IEEE International Conference on Robotics and Automation (pp. 3859–3866).

[Rau 2020] Rau, D., Rodina, J. and Stec, F. Generating instant trajectory of an indoor UAV with respect to its dynamics, 2020. 23rd International Symposium on Measurement and Control in Robotics (pp. 1-5), doi: 10.1109/ISMCR51255.2020.9263769.

[Sucan 2012] Sucan, I. A., Moll, M. and Kavraki, L. E. The Open Motion Planning Library, 2012. IEEE Robotics & Automation Magazine, 19(4), 72-82. doi:10.1109/mra.2012.2205651

[Stec 2021] Stec, F. and Rodina, J. Designing Human Machine Interface for UAVs in structured environment using ROS and Flask, 2021. Elitech`21: 23rd Conference of Doctoral Students, Faculty of Electrical Engineering and Information Technology, Bratislava, 26 May

[Wawrla 2019] Wawrla, L., Maghazei, O. and Netland, T.H. Applications of drones in warehouse operations, 2019.

CONTACTS:

MSc. Jaromir Stanko
Slovak University of Technology in Bratislava, Faculty of Electrical Engineering and Information Technology
Ilkovicova 3, 81219 Bratislava, Slovakia
jaromir.stanko@stuba.sk