2020 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. September 1-4, 2020

Bebop 2 Quadrotor as a Platform for Research and Education in Robotics and Control Engineering

Wojciech Giernacki[1], Piotr Kozierski[2,1], Jacek Michalski[1], Marek Retinger[1], Rafal Madonski[3], Pascual Campoy[4]

***Abstract*— In conducting research and teaching in fields related to unmanned aerial vehicles (UAVs), it is particularly important to select a universal, safe, open research platform and tools for rapid prototyping. Ready-to-use, low-cost micro-class UAVs such as Bebop 2 are successfully used in that regard. This article presents how to use the potential of this flying robot with Robot Operating System (ROS). The most important software solutions for the developed experimental testbed FlyBebop are characterized here. Their capabilities in research and education are exemplified using three distinct cases: 1) research results on the method of optimal, in-flight, iterative self-tuning of UAV position controller parameters (based only on current measurements), 2) the use of the reinforcement learning method in the autonomous landing of a single drone on a moving vehicle, 3) planning the movement of UAVs for autonomous video recording along the planned path in the arrangement: "cameraman drone" and "lighting technician drones".**

SUPPLEMENTARY MATERIAL

`https://youtu.be/7B0l9mJeCSM`

I. INTRODUCTION

The last decade is a period of dynamic expansion of the unmanned aerial vehicles market, in particular low-cost, lightweight constructions of small size [1]. In many countries, these UAVs can be used according to aviation regulations without the need for special permissions regulated by law, which increases the number of target recipients. Among them, the community of people interested in conducting research and educating students in the area of UAVs is still growing. Literature studies show a strong increase in the number of publications in recent years dedicated to this subject. Attention is paid to solid, in-flights validation of results in laboratory and real-world conditions. For this purpose, two things, correlated with each other, are needed: a universal research platform (real flying robot) and a simulation environment that allows rapid prototyping on an adequate model of the corresponding robot [2]. It is expected that the used research and teaching tools should be scalable and standardized, which increases the potential number of people who may quickly reflect, compare, validate, and expand published results. Therefore, the authors of this article devoted years of attention to trends and new products in the field of miniature unmanned aircraft with open architecture and commonly used simulation environments, such as ROS/Gazebo-based simulators (including RotorS, Kelpie, OpenUAV, CrazyS [2]-[7]), Paparazzi [8], AirSim [9], MAV3dSim [10], 4Fly [11] or ROSMAT [12]. More information about currently used simulators and comparisons between them can be found, for example in [13]-[16].

Fig. 1. Bebop 2 during the experiment in the `AeroLab` laboratory: in-flight, iterative self-tuning of altitude controller parameters, in an experiment with payload (empty bottle) and simulated wind gusts [30]

Among the group of numerous commercial UAVs that can be successfully used in conditions of safe work with students (while maintaining minimum safety rules) and in research (e.g. in the field of methods of autonomous control, techniques of estimation and fusion of sensory data, machine learning, optimization algorithms or path planning methods), three very popular and inexpensive constructions deserve attention: AR.Drone 2.0 [17], Crazyflie 2.0 [11] and Bebop 2 [18]. In this paper, it was decided to focus on the last UAV (Fig. 1), which, unlike the others, offers much longer flight times and a good quality camera. This RGB camera definitely increases the area of applications of this drone. Only in recent years one may find the use of Bebop in: autonomous drone race [19]-[20], accurate specified-pedestrian tracking [21], path planning for cinematography [22]-[23], 3D modelling and image analysis [24]-[27], vision-based approach for

[1]W. Giernacki, J. Michalski, M. Retinger are with the Institute of Robotics and Machine Intelligence, Faculty of Control, Robotics and Electrical Engineering of Poznan University of Technology, Poznań, Poland, corresponding author: `wojciech.giernacki@put.poznan.pl`, `http://www.uav.put.poznan.pl`.

[2] P. Kozierski is with the Institute of Computing Science (Faculty of Computing and Telecommunications) and with the Institute of Robotics and Machine Intelligence (Faculty of Control, Robotics and Electrical Engineering) of Poznan University of Technology, Poznań, Poland.

[3] R. Madonski is with the International Energy College, Energy Electricity Research Center, Jinan University, Zhuhai, Guangdong, China.

[4] P. Campoy is with Computer Vision and Aerial Robotics (CVAR), Centre for Automation and Robotics, Technical University of Madrid (UPM), Madrid, Spain.

978-1-7281-4277-7/20/$31.00 ©2020 IEEE

autonomous landing [28], cooperative exploration by UGV (unmanned ground vehicle) and UAV [29].

The goal of this article is to show the UAV community how the potential of low-cost UAV Bebop 2 may be used in conducting advanced research and teaching – with three examples of i) optimal, in-flight, iterative self-tuning technique of the UAV position controller based only on current measurements [30], ii) reinforcement learning for autonomous landing of a drone on a moving vehicle [31], as well as iii) group of drones performing autonomous video recording in the arrangement: "cameraman drone" and "lighting technician drone" along the planned path [32]. The presented results are obtained by extension of the control system of unmanned aerial vehicle, passing from the speed-control (nominal for Bebop 2) to a cascade system for tracking a reference position of the drone in time. All the solutions mentioned above and discussed in following sections were implemented in the ROS/Gazebo environment. They extend the base capabilities of the *bebop_autonomy* and *Parrot-Sphinx* packages for the simulation and conducting of autonomous flights of UAVs with the support of machine learning techniques and reference measurements from the OptiTrack motion capture system (ground truth). These solutions are the most important parts of the FlyBebop testbed, which is an experimental platform developed by the Authors in their laboratory called `AeroLab`[1].

In Section II, the most important research directions using UAV Bebop 2 as a test robot, were listed. The UAV was also characterized in the following subsections with reference to its physical layer and available open source software. The Section III presents the FlyBebop testbed and selected extensions for the architecture of autonomous control of group of flying robots for which the Section IV presents a method of automated selection of controller gains (on the example of a single UAV position controller). In Section IV, representative results of using the FlyBebop block architecture in conducting didactic projects on UAVs are presented.

II. BEBOP 2 FLYING ROBOT

A. Backgroud

Bebop 2 is a low-cost, compact size unmanned aerial vehicle produced by the Parrot company. It is the next generation of ready-to-use, stable, safe multi-rotor micro class UAVs, which after AR.Drone 2.0 [17] arouses great interest to the UAV community. Currently, research is successfully conducted using this drone towards path planning for infrastructure inspections [33], CNN-based single image obstacle avoidance [34], simple and advanced control techniques [35]-[38], SLAM techniques [39], user interaction with flying cameras [40]-[41], control software for choreographed UAVs [42], etc. These studies are possible due to the physical characteristics of the Bebop articulated below and the comprehensive, open source software.

[1] `AeroLab` – research laboratory at the Institute of Robotics and Machine Intelligence of Poznan University of Technology, enabling experimental flights supported by motion capture system and augmented reality.

TABLE I
CHARACTERISTICS OF BEBOP 2 SENSORS.

Parameter	Value
accelerometer & gyroscope	InvenSense 3-axes MPU 6050
pressure sensor (barometer)	Measurement Specialist MS5607 (flight altitude beyond $4.9\,\mathrm{m}$)
ultrasound sensor	analyses the flight altitude up to $8\,\mathrm{m}$
magnetometer	Asahi KASEI 3-axes AKM 8963
geolocalization	Furuno GN-87F GNSS module
Wi-Fi	IEEE 802.11ac 2.4 and 5 GHz dual dipole
vertical stabilization camera	MT9V117 camera (photo every $16\,\mathrm{ms}$)
camera	CMOS 14 Mpx 3-axis Full HD 1080p with Sunny 180° fish-eye lens: $1/2.3$"
lens stabilizer	3-axis digital system
video resolution	1920×1080 pixels (30 fps)
image resolution	4096×3320 pixels

B. Hardware

Bebop 2 is relatively small ($0.328 \times 0.382\,\mathrm{m}$). The takeoff mass of the UAV is $0.5\,\mathrm{kg}$. It is equipped with four propulsion units ($4 \times$ KV BLDC Motor, $7500 - 12000\,\mathrm{rpm}$), 6" propellers and $2700\,\mathrm{mAh}$ battery, thus it is able to provide maximum flight time up to $25\,\mathrm{min}$ without payload (which can reach up to $0.55\,\mathrm{kg}$) [30]. In nominal case it can flight with the maximum horizontal speed up to $60\,\mathrm{km/h}$ and maximum vertical speed up to $21\,\mathrm{km/h}$. Bebop 2 resists head winds up to $63\,\mathrm{km/h}$ [18].

The UAV is equipped with BusyBox Linux operating system used for $P7$ dual-core CPU Cortex 9 processor, 1 GB RAM memory and 8 GB of flash memory. Based on current data from the 9-DOF inertial measurement unit (IMU) and on-board sensors from the Table I, Extended Kalman Filter (EKF) [43] is used for state estimation of the Bebop 2.

C. Controlling the drone

It is possible to manually control the Bebop 2 flight using the FreeFlight Pro application for mobile phones and using the Parrot Skycontroller 2 device (it increases the maximum range from approx. $300\,\mathrm{m}$ to $2\,\mathrm{km}$). Skycontroller 2 (having the highest priority in `AeroLab`) is used to secure experiments (in case of loss of communication) in autonomous flights using a ground control station (GCS) equipped with Wi-Fi. For the GCS, currently the most popular solutions are based on the open source driver *bebop_autonomy* [44] developed natively for Linux Ubuntu 16.04 LTS and Robot Operating System (ROS) in the Kinetic Kame version. For Bebop 2, a dedicated device based on ROS for simultaneous localization and mapping (SLAM), was produced: S.L.A.M. dunk for developers, allowing the drone to build an environment map during the flight with a $5\,\mathrm{Hz}$ frequency.

The *Bebop_autonomy* package for communication and programming of control commands for the Bebop 2.0 from the GCS was developed by Monajjemi et al. It is described in details in [44]. This open-source ROS package (driver) is based on the Parrot's official Software Development Kit, i.e. ARDrone SDK3. By matching ROS standards, *bebop_autonomy* using exacutable node (*bebop_driver_node*) enables to subscribe and publish to Bebop's topics with

TABLE II

ROS MESSAGES & TOPICS FOR BEBOP 2 CONTROL.

Action	Command
Takeoff	rostopic pub –once /bebop/takeoff std_msgs/Empty
Land	rostopic pub –once /bebop/land std_msgs/Empty
Emergency	rostopic pub –once /bebop/reset std_msgs/Empty
Piloting	rostopic pub –once /bebop/cmd_vel geometry_msgs/Twist

TABLE III

IP CONFIGURATION FOR REAL AND SIMULATED UAVS.

Bebop unit	Name	Real UAV	Simulated UAV
1	Leader	192.168.100.1	10.202.0.1
2	Follower no.1	192.168.100.2	10.202.1.1
3	Follower no.2	192.168.100.3	10.202.2.1

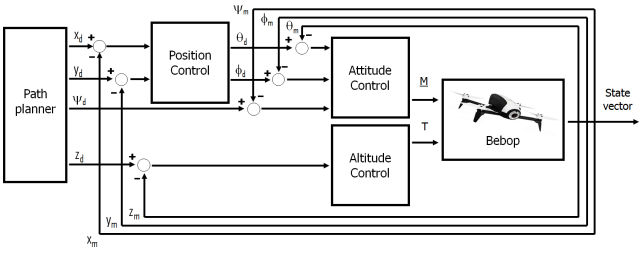


Fig. 2. Diagram of control system for Bebop 2 autonomous flights

navigation data (*/bebop/odom* & */bebop/fix*), as well as to receive video from the UAV in the real-time mode (*/bebop/image_raw*). The full list of available topics can be found in [44] or using command *rostopic list* in the Linux terminal. The most important control commands are presented in Table II.

This package can be effectively used to control the Bebop 2 model in *Parrot-Sphinx* [45] simulator for ROS/Gazebo. The package was initially developed to cover the needs of Parrot's engineers. The main advantage of using Gazebo in this case is the ability to simulate the physical behavior of the drone in a modeled, own environments/scenes with which Bebop 2 can interact in a simulated virtual flight. *Parrot-Sphinx* enables to observe flight information in three ways: runtime observtion via a web HMI (Human Machine Interface) (*sphinx dashboard*), runtime observation via command line tool (*tlm-data-logger*) and post-flight analysis with recorded data files (*.tlmb* files). Of course, since *bebop_autonomy* and *Parrot-Sphinx* use ROS architecture, there is a possibility to record and play after the flights the *rosbag* files with data from selected sensors. In [45], one can find a comprehensive documentation for *Parrot-Sphinx*, which, together with *bebop_autonomy*, was used to develop the FlyBebop testbed.

III. FLYBEBOP TESTBED FOR AUTONOMOUS FLIGHTS

In order to conduct research and didactic projects in `AeroLab` [30], FlyBebop testbed (Fig. 3) was build using the modular architecture of ROS depending on the needs of particular test. The FlyBebop allows prototyping of autonomous flights in simulated and real-world

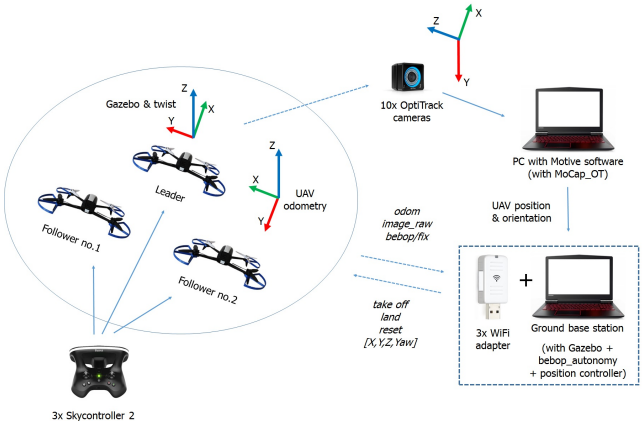


Fig. 3. Simplified diagram of the FlyBebop testbed

conditions for a group of Bebop 2 units with the support of additional sensory information from the MoCap system (OptiTrack). For this purpose, one can use the functionality of packages, such as *bebop_autonomy*, *Parrot_Sphinx*, *MoCap_OT* [30], *position_controller* [32], *drl-landing* [46], *object_detection_tensorflow* [31], *Optical-Flow-based-Obstacle_Avoidance* [31], etc. Extending the base capabilities of the testbed (see Fig. 7 from [30]) for the control needs of the group of drones is possible by using separate communication interfaces for each flying unit. Real robots communicate with the ground station using separate Wi-Fi cards. In the simulation *Parrot-Sphinx* uses a virtual ethernet interface. ROS with publisher/subscriber mechanisms for proper communication with drones (virtual and real units) requires the use of **.launch* and **.drone* files with separate IP addresses and identifiers (names of drones) – see Table III.

The proposed FlyBebop testbed requires transformations between particular coordinate systems (see Fig. 3). The following orientation has been adopted in the global system: X axis – forward, Y – left, Z – up, $+\psi$ (yaw angle) – CCW. This orientation is associated with the topics */bebop/cmd_vel* and */gazebo/default/pose/info*. Data from the MoCap system and drone on-board odometry need to be recalculated to the global system. By default, the *bebop_autonomy* driver allows sending $u(t)$ control commands (from -1 to 1) to Bebop 2 (converted to specified angles: pitch θ & roll ϕ, as well as vertical & rotational velocities) using *bebop/cmd_vel* topic in the format:

$$u(t) = \begin{bmatrix} \text{linear.x} & \text{linear.y} & \text{linear.z} & \text{angular.z} \end{bmatrix}^T. \quad (1)$$

Note that restrictions introduced in Table IV, resulting from physical sizes of the `AeroLab` [30], according to [44], allow to adjust and scale the dynamics of UAV flight to the space of laboratory according to:

$$Ku(t) = \begin{cases} \text{roll} = \text{max\_tilt\_angle} \cdot \text{linear.y} \\ \text{pitch} = \text{max\_tilt\_angle} \cdot \text{linear.x} \\ \text{ver\_vel} = \text{max\_vert\_speed} \cdot \text{linear.z} \\ \text{rot\_vel} = \text{max\_rot\_speed} \cdot \text{angular.z} \end{cases} \quad (2)$$

Considering the control architecture from Fig. 2 for each drone, it is necessary to develop a cascade system for tracking of X and Y positions expressed in the global reference system. For automatic tuning of the controllers in these loops, as well as the altitude and ψ angle controllers, the methodology presented in [30] and characterized briefly in Section IV was adopted. The obtained best gains of particular controllers is presented in Table IV.

IV. BEBOP IN RESEARCH & EDUCATION

Due to limited space in the article, from a wide range of research and didactic projects using the Bebop 2 flying robot as an experimental platform, three selected were briefly described in the subsequent subsections.

A. Research – In-flight tuning of Bebop's controllers

The UAV control architecture from Fig. 2 was considered. In addition to the previously published results of research on the optimal, in-flight self-tuning of UAV controllers: altitude [30] and ψ angle [32], the method to obtain gains for controllers in the X and Y axes (see Table IV), is presented below. Respectively, a symmetrical drone structure was adopted, which enables tuning controllers for both axes simultaneously, based on measurements from iterative position changes in one of axis – here the X axis (see Fig. 4). For controllers, a PID type structure with unconstrained control signal $u(t)$, was adopted, described by the equation:

$$u(t) = k_P e(t) + k_I \int_0^{t_h} e\,(t)\,dt + k_D \frac{d}{dt} e(t), \tag{3}$$

where $e(t)$ is the control error, k_P is the proportional gain, k_I is the integral gain, k_D is the derivative gain, and t_h is a flight time horizon.

The integral gain was given as a very small value (the only goal: minimization of the steady-state error), according to the tuning methodology for controllers from the paper [30]. Gains k_P and k_D are expected to be found using one of iterative learning methods from [30], [47] – here: Golden-search Method (GLD). The optimization problem for a search of the PID type controller gains (for fixed k_I) is defined as

$$\begin{aligned} \underset{k_P,k_D}{min} \quad & J\,(t) = \textstyle\int_0^{t_a} (\alpha\,|e\,(t)| + \beta\,|u\,(t)|)\,dt, \\ s.t. \quad & 0 \le k_P \le k_P^{max} \\ & 0 \le k_D \le k_D^{max} \end{aligned} \tag{4}$$

where α and β are set weights in cost function J, t_a is the time of gathering information (to calculate new controller gains) in the optimization procedure, k_P^{max} and k_D^{max} are upper bounds of the predefined ranges of exploration in the optimization procedure of the PID controller parameters. The process of tuning the controller for the X axis, supported by the distance measurement from the MoCap system, was conducted in *Robotics Arena of the Center for Automation and Robotics* in *Technical University of Madrid* (for which the system was scaled from FlyBebop). It is illustrated in the video attached to this article. Figure 5 shows the percentage improvement of results in the following 56 steps of the GLD algorithm, which leads to the gains of PID controllers for the X and Y axes from Table IV.

Remark 4.1 To perform GLD optimization and be able to control Bebop autonomously using *bebop_autonomy*, it is necessary to use the discrete form of the PID controller instead of the one from the equation (3), i.e.

$$u_n = k_P \cdot e_n + k_I \cdot \sum_{m=0}^{N_h} e_m + k_D \cdot (e_n - e_{n-1}), \tag{5}$$

where $u_n \in [-1,\,1]$ is the normalized control signal at n-th time step, e_n, e_{n-1} is the control error in current and previous moments, and N_h is the control horizon.

B. Education – UAVs flight path planning

In autonomous flights of a group of UAVs, the quality and precision of the flight is determined by the path planning mechanisms used for the group of UAVs, as well as the mechanisms of maintaining a given arrangement of tracking units relative to the position and orientation of the leader setting the course to the others. This is especially important in the video recording using several UAVs equipped with cameras and lighting devices so that during flying in limited light conditions, the fragments of drones construction, their shadows, etc. do not appear in the frames, and the cameraman drone keeps a constant perspective/distance to the object. This problem was the motivation behind paper [30]. Its promising results (validated on the Bebop group) were used for further research (see Fig. 6 and attached video material) on leader movement planning and stabilization of the drone group flight in conditions inspired by [48].

The block architecture and functionality of the FlyBebop testbed allows a rapid implementation of path planning algorithms and their validation on Bebop in simulated and real-world conditions. An original node (*formation_controller*) has been developed and integrated [30] for the mechanism of movement of the UAVs group in V-shaped formation. In Fig. 7, demonstrative screen shots from one of the experiments were presented. In this experiment, the cameraman drone (leader) and two followers (conventionally equipped with lighting points directed to the center of the leader frame), are designed to record video flying around a convex object at a constant altitude (here: h = 1.5 [m]). Video recording from a camera leader moving along a curve flight path is directly influenced by the following quantities related to the observer (global) system:

a) $d\,[\mathrm{m}]$ – distance of the follower (equipped with a light point) to the leader in a straight line,

b) $\alpha\,[^\circ]$ – the angle of deviation of the abovementioned straight line from the leader axis,

c) $r\,[\mathrm{m}]$ – distance of the leader to the recorded object,

d) $\underline{\epsilon} = [\epsilon_x,\,\epsilon_y,\,\epsilon_z,\,\epsilon_\psi]^T$ – tolerance of achieving a given coordinate

e) $k\,[\mathrm{m}]$ – distance between successive points of the trajectory (concentration of intermediate points in flight).

In planning the leader flight path of a group of three units flying coplanar around an object with a known sizes

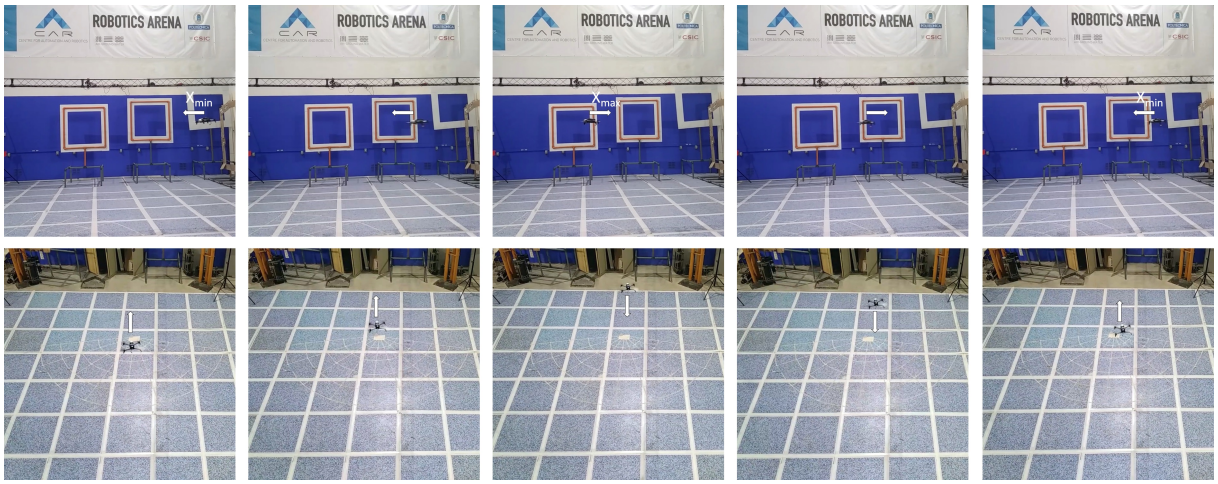


Fig. 4. Flight in the X axis with the goal: acquisition of the current position from MoCap in one iteration of the GLD algorithm

TABLE IV

PARAMETERS FOR BEBOP FLYING IN AEROLAB CONDITIONS.

Parameter	**Default value**	**Used value**	
SpeedSettings-MaxVerticalSpeedCurrent	1.0 [m/s]	1.0 [m/s]	
SpeedSettings-MaxRotationSpeedCurrent	100.0 [°/s]	20.0 [°/s]	
SpeedSettings-MaxPitchRollRotationSpeedCurrent	300.0 [°/s]	80.0 [°/s]	
PilotingSettings-MaxTiltCurrent	20.0 [°]	5.0 [°]	
PilotingSettings-MaxAltitudeCurrent	30.0 [m]	2.5 [m]	
PilotingSettings-MaxDistanceValue	2000.0 [m]	3.0 [m]	
PilotingSettings-NoFlyOverMaxDistance- Shouldnotflyover	0 (false)	1 (true)	
	Gain	**Gain**	**Gain**
	k_P	k_I	k_D
X-axis	0.69	0.00015	50
Y-axis	0.69	0.00015	50
Z-axis	1.32	0.0003	10.2
θ	default	default	default
ϕ	default	default	default
ψ	0.07	0.00001	0.9

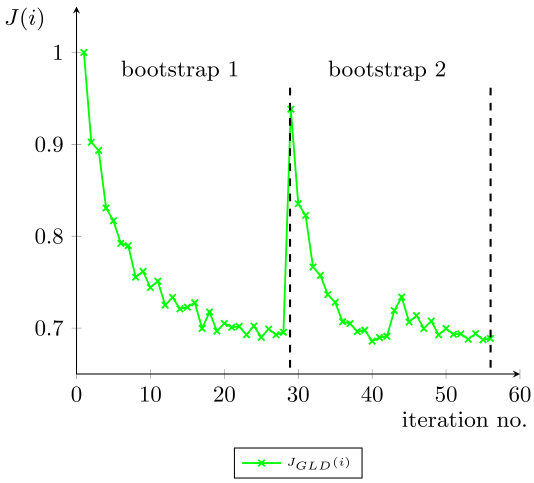


Fig. 5. The change in the value of the $J(i)$ cost function in subsequent iterations (i) of the GLD algorithm, relative to iteration No. 1 (where 1 = 100%)

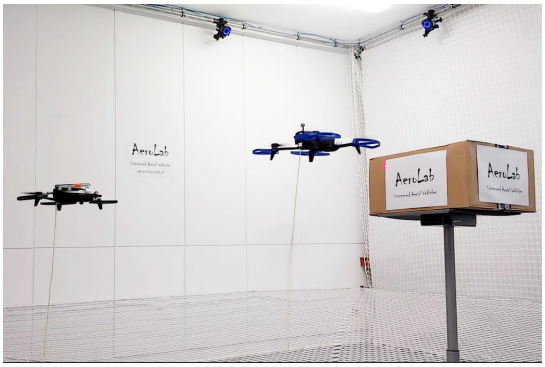
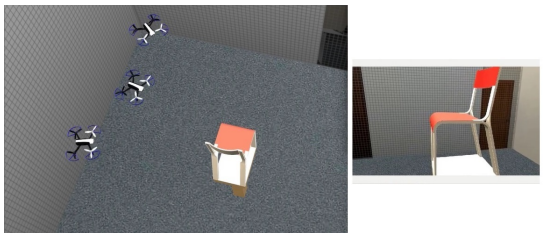


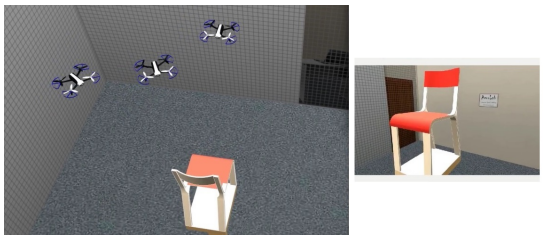
Fig. 6. "Cameraman drone" (unit on the right) and "lighting technician drone" (unit on the left) during the autonomous video recording experiment

in 3D (see Fig. 8), according to [49], the object model is the smallest possible cuboid in which the object, recorded by the UAV, can be placed. For a constant flight altitude, in the algorithm implemented in the *formation_controller* node [30], it is important to know the vertices of the rectangle O(X_i,Y_i), for $i = 1, ..., 4$, as well as the set value of r. Then, depending on the mutual position of two adjacent corners, it is possible to determine the rotation angle δ around the Z axis of the global system according to:

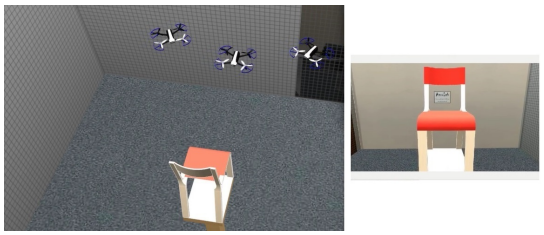
$$\delta = \arctan\left(y_1 - y_2, x_2 - x_1\right). \tag{6}$$



(a) Start of recording



(b) Curvature flight



(c) Filming front of the object

Fig. 7. Autonomous filming around the object by a group of UAVs

Assuming that the starting point $P_0 = (P_{0x}, P_{0y})$ on trajectory of the leader (distant from the object marked by the value r) will be chosen as a point located on a straight line perpendicular to the segment connecting the points O_1 and O_2:

$$P_{0x} = \frac{x_1+x_2}{2} + r \cdot \sin(\delta), \tag{7}$$

$$P_{0y} = \frac{y_1+y_2}{2} + r \cdot \cos(\delta), \tag{8}$$

the coordinates of subsequent points lying on straight line segments can be calculated relatively to the starting point according to:

$$P_{nx} = P_{(n-1)x} \pm k \cdot \sin(\delta), \tag{9}$$

$$P_{ny} = P_{(n-1)y} \pm k \cdot \cos(\delta) \tag{10}$$

for predefined k and the number of points n.

Points located on curves are calculated from the parametric equation of the circle

$$P_{nx} = x_1 + r \cdot \sin(\gamma), \tag{11}$$

$$P_{ny} = y_1 + r \cdot \cos(\gamma), \tag{12}$$

where $\gamma \in [0°, 360°]$, and x_1, y_1 are the coordinates of the corner around which the rotation occurs.

In addition to movement of the group of UAVs along straight line segments and curvatures, it is necessary to set leader's camera (expressed in the ψ angle) towards the object successively. The initial orientation of the leader at point P_0

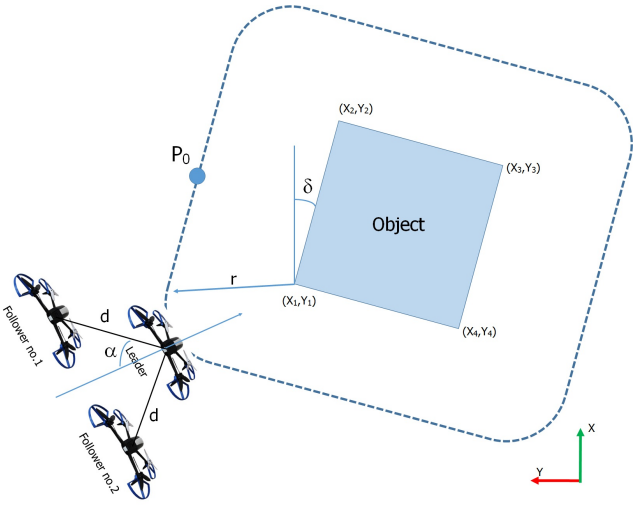


Fig. 8. Reference flight path of the leader during autonomous filming by a group of UAVs

is expressed by the relationship

$$\psi_0 = -90° - \delta. \tag{13}$$

For subsequent points of the curve, the angle ψ is reduced (for CW movement) or increased (CCW movement). The step at which this angle changes is the same as the step of changing the angle γ thanks to which the drone camera is focused on the point where the corner of the object is located [30].

In the experiment presented in Fig. 7, in subsequent attempts, following values were adopted: $d = 0.8\,[\mathrm{m}]$, $\alpha = 80\,[°]$, $r = 0.6\,[\mathrm{m}]$, $\epsilon_x = \epsilon_y = 0.05\,[\mathrm{m}]$, $\epsilon_z = 0.07\,[\mathrm{m}]$, $\epsilon_\psi = 1.15\,[°]$, $d_N = 0.8\,[\mathrm{m}]$.

C. Education – Autonomous Landing on Moving Platform

In 2017, at the prestigious Mohamed Bin Zayed International Robotics Challenge (MBZIRC), Challenge 1 required the UAV to locate, track and land on a moving vehicle. Based on the experience gathered by the best research teams [50], many approaches to the problem of autonomous drone landing were proposed. One of the most interesting concept was the use of Reinforcement Learning technique proposed in [46], implemented in the Aerostack [51] environment, based on ROS. This concept was the background for the paper [31], where once again the validity of Bebop as a test platform, was proved. It was assumed that based on the image from the drone's front camera and measurements only from on-board sensors (without MoCap), deep reinforcement learning method was used to teach the agent (UAV) how to efficiently land on moving platform. To locate it and calculate the relative distance, ArUco markers (from ArUco Library based on OpenCv) were used (see Fig. 9).

For the purposes of [31], the *dlr-landing* [52] library was used. The agent was taught for reward function R from the reference [46] during 4800 trials using a neural network composed of 3 hidden layers (with 400, 300 and 200 neurons) and input and output layers with 7 and 5

neurons, respectively. The reward function R consists of four components

$$R = \begin{cases} r_1 = -100 & if\, x > x_{max} \\ & or\, y > y_{max} \\ & or\, z > z_{max} \\ r_2 = -50 & if\, \tau > 0.8 \\ & and\ (z^{\tau} > z^{\tau_{max}} \\ & or\, z^{\tau} < 0) \\ r_3 = 100e^{k(z^{\tau^*} - z^{\tau})} & if\, \tau > 0.8 \\ & and\ (z^{\tau} < z^{\tau_{max}} \\ & or\, z^{\tau} > 0) \\ r_4 & otherwise, \end{cases} \tag{14}$$

$$r_4 = shaping[t] - shaping[t-1], \tag{15}$$

$$\begin{aligned} shaping[t] = -\alpha_1 \sqrt{x^2 + y^2 + z^2 + \psi^2} - \\ \alpha_2 \sqrt{\dot{x}^2 + \dot{y}^2} - \alpha_3 \sqrt{\dot{\psi}^2 + \theta^2 + \phi^2 + \dot{z}^2}, \end{aligned} \tag{16}$$

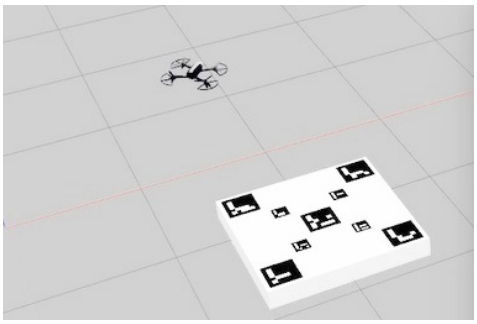
and it includes the penalty r_1 for exceeding the maximum position of UAV relatively to the moving platform (x_{max}, y_{max}, z_{max}); values r_2 and r_3 "encouraging" UAV to cut off propulsion units at a safe altitude, and the "shaping" component r_4 informs about the progress of the given iteration t relatively to the previous $t-1$, where x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$, ϕ, θ, ψ mean the current position, speed and orientation of UAV; z^{τ}, $z^{\tau_{max}}$, z^{τ^*} – mean the current, maximum, desired thrust cutoff altitude, and α_1, α_2, α_3, k are experimentally defined constants with values: 100, 10, 1 and 1.55, respectively.

In the project from [31], it was necessary to adjust the functionality of the *dlr-landing* node to ROS standards for the FlyBebop. For this purpose, reading data from the following topics was used: */bebop/image_raw* (reading the image from the front camera tilted down) and */bebop/odom_conv* (data in the current position of the UAV), which after being processed, were published to topics: */bebop/cmd_vel* (allowing control of angle and thrust changes), and to */bebop/aruco_camera* (publish a camera image with information about the detected markers and calculated UAV distance from the moving platform). Autonomous control in individual flight phases (e.g. take-off, ascent and flight to the selected position) was provided by *position_controller* node.

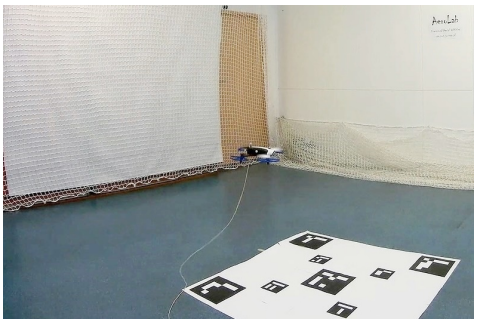
Illustratively, representative results from experiments in which Bebop lands autonomously on a fixed and moving platform in the simulator and in `AeroLab` are included in the attached video.

V. CONCLUSION

In the paper, the Bebop UAV was presented as a low-cost, open source and universal flying platform for research and education in control engineering and robotics. Extension of the drone control system for the open source available libraries: *bebop_autonomy* and *Parrot-Sphinx* for Robot Operating System enables the implementation of advanced research and student projects in the field of low and high-level control of a single unit and a group of drones, methods of planning their movement, estimation techniques of robot's position and orientation or the development of machine intelligence methods and optimization algorithms. In indoor conditions of the `AeroLab`, it is possible (quickly and reliably) to validate solutions using the developed experimental testbed FlyBebop with measuring support from Bebop sensors and the MoCap system (used as a ground truth).



(a) Bebop model in ROS/Gazebo



(b) Bebop drone in `AeroLab`

Fig. 9. Autonomous landing on (a) moving, (b) static platform

In the near future, it is planned to make available the developed program codes and expand the functionality of FlyBebop, so that using *object_detection_tensorflow*, *Optical-Flow-based-Obstacle_Avoidance* nodes and Matlab-based libraries (mostly associated with neural networks) be able to conduct various experimental flights safely using collision avoidance mechanisms with the support of the possibilities offered by virtual reality.

ACKNOWLEDGMENT

The authors are grateful to engineers: M. Molska, A. Kuźniewska and B. Kulecki for the implementation of solutions presented in Section IV (used in order to show the Bebop 2 diverse applications).

REFERENCES

[1] Valavanis, K.; Vachtsevanos, G.J. (Eds.) Handbook of Unmanned Aerial Vehicles; Springer: Dordrecht, The Netherlands, 2015.

[2] Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R. RotorS - A Modular Gazebo MAV Simulator Framework. Robot Operating System (ROS), Studies in Computational Intelligence, 2019, pp. 595–625.

[3] Hönig, W.; Ayanian, N. Flying Multiple UAVs Using ROS. Robot Operating System (ROS), Studies in Computational Intelligence, 2017, pp. 83–118.

[4] Mendonca, R.; Santana, P.; Marques, F.; Lourenco, A.; Silva, J.; Barata, J. Kelpie: A ROS-Based Multi-Robot Simulator for Water Surface and Aerial Vehicles. In Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, UK, 13-16 October 2013, pp. 1–6.

[5] Schmittle, M.; Lukina, A.; Vacek, L.; Das, J.; Buskirk, Ch.P.; Rees, S.; Sztipanovits, J.; Grosu, R.; Kumar, V. OpenUAV: A UAV Testbed for the CPS and Robotics Community. In Proceedings of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS'18): ACM/IEEE 9th International Conference on Cyber-Physical Systems, Porto, Portugal, 11 - 13 April 2018, pp. 130-–139.

[6] Zhang, M.; Qin, H.; Lan, M.; Lin, J.; Wang, S.; Liu, K.; Lin, F.; Chen, B.M. A high fidelity simulator for a quadrotor UAV using ROS and Gazebo. In Proceedings of the IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society, Yokohama, Japan, 9-12 November 2015, pp. 1–6.

[7] Silano, G.; Aucone, E.; Iannelli, L. CrazyS: a software-in-the-loop platform for the Crazyflie 2.0 nano-quadcopter. In Proceedings of the 2018 26th Mediterranean Conference on Control and Automation (MED), Zadar, Croatia, 19-22 June 2018, pp. 1–5.

[8] Paparazzi UAV http://wiki.paparazziuav.org/wiki/Main_Page (access: 22.01.2020).

[9] Shah, S.; Dey, D.; Lovett, Ch.; Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. Springer, Proceedings in Advanced Robotics book series (SPAR, volume 5), Field and Service Robotics, pp. 621–635, 2018.

[10] Lugo-Cardenas, I.; Flores, G.; Lozano, R. The MAV3DSim: A Simulation Platform for Research, Education and Validation of UAV Controllers. IFAC Proceedings Volumes, 47(3), 2014, pp. 713–717.

[11] Giernacki, W.; Skwierczyński, M.; Witwicki, W.; Wroński, P.; Kozierski, P. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering, Proceedings of the 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 28-31 August 2017, pp. 37–42.

[12] ROSMAT: ROlling Spider MAtlab Toolbox, https://github.com/Parrot-Developers/RollingSpiderEdu/blob/master/MIT_MatlabToolbox/README.md (access: 22.01.2020).

[13] Mairaj, A.; Baba, A.I.; Javaid, A.Y. Application Specific Drone Simulators: Recent Advances and Challenges. Simulation Modelling Practice and Theory 94, pp. 1–24.

[14] Hentati, A.I.; Krichen, L.; Fourati, M.; Fourati, L.Ch. Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis. In Proceedings of the 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC). Limassol, Cyprus, 25-29 June 2018, pp. 1495–1500.

[15] Noori, F.M.; Portugal, D.; Rocha, R.P.; Couceiro, M.S. On 3D Simulators for Multi-Robot Systems in ROS: MORSE or Gazebo? In Proceedings of the 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, China, 11-13 October 2017, pp. 19–24.

[16] Al-Mousa, A.; Sababha, B.H.; Al-Madi, N.; Bardhouthi, A.; Younisse, R. UTSim: A framework and simulator for UAV air traffic integration, control, and communication. International Journal of Advanced Robotic Systems, September-October 2019, pp. 1–19.

[17] Krajnik, T.; Vonasek, V.; Fiser, D., Faigl, J. AR-Drone as a Platform for Robotic Research and Education, Proceedings of the Research and Education in Robotics – EUROBOT 2011, pp. 172–186, 2011.

[18] Parrot BEBOP 2. The Lightweight, Compact HD Video Drone. Available online: https://www.parrot.com/us/drones/parrot-bebop-2 (access: 25.01.2020).

[19] Li, S.; Ozo M.M.O.I., de Wagter; C., de Croon, G.C.H.E. Autonomous drone race: A computationally efficient vision-based navigation and control strategy. arXiv:1809.05958.

[20] Moon et. all. Challenges and Implemented Technologies used in Autonomous Drone Racing. Intelligent Service Robotics. April 2019, 12(2), pp. 137—148.

[21] Yang, Z.; Huang, Z.; Yang, Y.; Yang, F., Yin, Z. Accurate Specified-Pedestrian Tracking from Unmanned Aerial Vehicles. In Proceedings of the 2018 18th IEEE International Conference on Communication Technology, Chongqing, China, 8-11 October 2018, pp. 1256—1260.

[22] Rousseau, G.; Stoica Maniu, C.; Tebbani, S.; Babel, M.; Martin, N. Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control. In Proceedings of the 2018 European Control Conference (ECC), Limassol, Cyprus, 12-15 June 2018, pp. 2897–2903.

[23] Nageli, T.; Meier, L,; Domahidi, A.; Alonso-Mora, J.; Hilliges, O. Real-time Planning for Automated Multi-View Drone Cinematography. ACM Transactions on Graphics, Vol. 36, No. 4, 132:1–10.

[24] Pagliari, D.; Pinto, L. Use of Fisheye Parrot Bebop 2 Images for 3D Modelling using Commercial Photogrammetric Software. Int. Arch. Photogrammetry and Remote Sensing. XLII-2, 2018, pp. 813–820.

[25] Suciu, G.; Dragu, M.; Hussain, I.; Iliescu, A.M.; Orza, O.; Mocanu, C. 3D Modeling Using Parrot Bebop 2 FPV. In Proceedings of the 2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC), Bucharest, Romania, 29-31 October 2018, pp. 61–65.

[26] Shah, U.; Khawad, R.; Krishna K.; Detecting, Localizing, and Recognizing Trees with a Monocular MAV: Towards Preventing Deforestation. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May-3 June 2017, pp. 1982–1987.

[27] Bamford, T.; Esmaeili, K.; Schoellig, A.P. Aerial Rock Fragmentation Analysis in Low-Light Condition Using UAV Technology. arXiv:1708.06343.

[28] Cabrera-Ponce, A.A.; Martinez-Carranza, J. A Vision-Based Approach for Autonomous Landing. In Proceedings of the 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linkoping, Sweden, 3-5 October 2017, pp. 126–131.

[29] Hood, S.; Benson, K.; Hamod, P.; Madison, D.; O'Kane, J.M; Rekleitis, I. Bird's Eye View: Cooperative Exploration by UGV and UAV. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13-16 June 2017, pp. 247–255.

[30] Giernacki, W. Iterative Learning Method for In-Flight Auto-Tuning of UAV Controllers Based on Basic Sensory Information. Applied Science 2019, 9(4), 648; https://doi.org/10.3390/app9040648

[31] Kuźniewska, A.; Molska, M. Mechanisms of machine learning in support of the flight autonomy of group of unmanned aerial vehicles, B.Sc. Thesis in polish, Poznan University of Technology, 2019.

[32] Kulecki, B. Autonomous images sequence registration using a group of unmanned aerial vehicles, B.Sc. Thesis in polish, Poznan University of Technology, 2019.

[33] Besada, J.A.; Bergesio, L.; Campana, I.; Vaquero-Melchor, D.; Lopez-Araquistain, J.; Bernardos, A.M.; Casar, J.R. Drone Mission Definition and Implementation for Automated Infrastructure Inspection Using Airborne Sensors. Sensors 2018, 18, 1170.

[34] Chakravarty, P.; Kelchtermans, K.; Roussel, T.; Wellens, S.; Tuytelaars, T.; Van Eycken, L. CNN-based Single Image Obstacle Avoidance on a Quadrotor. Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore, 29 May-3 June 2017, pp. 1–9.

[35] Orozco-Soto, S.M.; Ibarra-Zannatha, J.M.; Malo-Tamayo, A.J.; Cureno-Ramirez, A. Active Disturbance Rejection Control for UAV Hover using ROS. In Proceedings of the 2018 XX Congreso Mexicano de Robótica (COMRob), Ensenada, B.C., Mexico, 12-14 September 2018, pp. 1–5.

[36] Lopez Luna, A.; Cruz Vega, I.; Martinez Carranza, J. Gain-Scheduling and PID Control for an Autonomous Aerial Vehicle with a Robotic Arm. In Proceedings of the 2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA), Barranquilla, Colombia, 1-3 November 2018, pp. 1–6.

[37] Vajgl, M.; Hurtik, P.; Števuliáková, P. Drone real-time control based on pattern matching. In Proceedings of the 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), Otsu, Japan, 27-30 June 2017, pp. 1–6.

[38] Rosaldo-Serrano, M.A.; Aranda-Bricaire, E. Trajectory Tracking for a Commercial Quadrotor. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23-26 April 2019, pp. 1616–1621.

[39] Stumberg, L.; Usenko, V.; Engel, J.; Stückler, J.; Cremens, D. From monocular SLAM to autonomous drone exploration. In Proceedings of the 2017 European Conference on Mobile Robots (ECMR), Paris, France, 6-8 September 2017, pp. 1–8.

[40] Monajjemi, M.; Mohaimenianpour, S.; Vaughan, R. UAV, Come To Me: End-to-End, Multi-Scale Situated HRI with an Uninstrumented Human and a Distant UAV. Proceedings of the 2016 IEEE/RSJ

International Conference on Intelligent Robots and Systems (IROS). Daejeon, South Korea, 9-14 October 2016, pp. 1–8.
[41] Lan, Z.; Shridhary, M.; Hsuz, D.; Zhao S. XPose: Reinventing User Interaction with Flying Cameras. Robotics: Science and Systems 2017, pp. 1–9.
[42] Ding, H.T.; Cruz Torres, M.H.; Holvoet, T. Dancing UAVs: Using Linear Programming to Model Movement Behavior with Safety Requirements. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13-16 June 2017, pp. 326–335.
[43] Moore, T.; Stouch, D. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In: Menegatti E., Michael N., Berns K., Yamaguchi H. (eds) Intelligent Autonomous Systems 13. Advances in Intelligent Systems and Computing, vol 302. Springer, Cham, 2016.
[44] bebop_autonomy—ROS Driver for Parrot Bebop Drone (quadrocopter) 1.0 2.0. Available online: https://bebop-autonomy.readthedocs.io/en/latest/ (access: 18.01.2020).
[45] What is Sphinx. Available online: https://developer.parrot.com/docs/sphinx/whatissphinx.html (access: 8.01.2020).
[46] Rodriguez-Ramos, A.; Sampedro, C.; Bavle, H.; Gil Moreno, I.; Campoy, P. A Deep Reinforcement Learning Technique for Vision-Based Autonomous Multirotor Landing on a Moving Platform, In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1-5 October 2018, pp. 1010-1017.
[47] Giernacki W., Horla D., Báča T., Saska M., Real-time model-free minimum-seeking autotuning method for unmanned aerial vehicle controllers based on Fibonacci-search algorithm, Sensors, 2019, Vol. 19, 312.
[48] Saska, M.; Kratky, V.; Špurny, V.; Báča, T. Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control. In Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12-15 September 2017, pp. 1–8.
[49] Wang, X.; Yadav, V.; Balakrishnan, S. Cooperative UAV formation flying with obstacle/collision avoidance. IEEE Transactions on Control Systems Technology, 15(4), pp. 672–679, 2007.
[50] Saska, M.; Báča, T.; Špurny, V.; Loianno, G.; Thomas, J.; Krajnik, T.; Stepan, P.; Kumar, V. Vision-based high-speed autonomous landing and cooperative objects grasping - towards the MBZIRC competition, In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, South Korea, 9-14 October 2016, pp. 1–5.
[51] Sanchez-Lopez, J.L.; Molina, M.; Bavle, H.; Sampedro, C.; Suarez-Fernandez, R.A.; Campoy, P. A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework. Journal of Intelligent & Robotic Systems, 88, pp. 683-–709, 2017.
[52] drl_landing node, https://github.com/alejodosr/drl-landing (access: 14.02.2020)