Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1177–1185
DOI: 10.15439/2017F184
ISSN 2300-5963 ACSIS, Vol. 11

Visual simulator for MavLink-protocol-based UAV, applied for search and analyze task

Piotr Śmigielski, Mateusz Raczyński, Łukasz Gosek
Student Scientific Association AI LAB,
Faculty of Automation, Electrical Engineering, Computer Science and Biomedical Engineering,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
Email: smigielski.piotr@gmail.com

***Abstract*—In this paper the authors present the results of research to develop the visual system for autonomous flying agent. The core elements of the vision system which were designed and implemented in the earlier stage of the project are brought together. The second aim is to show capabilities of a simulation environment designed and developed by the authors in order to enable testing of the vision systems (dedicated for Unmanned Aerial Vehicles) in the artificial environment. The first section of the paper introduces the testing (simulation) environment for MavLink-protocol-based autonomous flying robots. Next, the core elements of a vision system, designed for Unmanned Aerial Vehicle (UAV), are discussed. This includes pre-processing and vectorization algorithms, object recognition methods and fast three-dimensional model construction. The third part introduces a set of algorithms for robot navigation, solely based on vision and altitude sensor and compass. The paper concludes with the description of the tests and presentation of results where designed simulator was applied to show mentioned vision system elements operating together to execute complex task.**

I. Introduction

THE SIMULATION has always been important in the development of advanced robotic systems. Such a need is driven by a number of factors. The key one is the cost of the robotic solutions. Utilizing such environments for testing of the early versions of elements of the system, such as vision system and navigation algorithms, may help avoiding costly accidents [1]. Additional benefit of using artificial testing environments lays in short time between bug discovery, patch implementation and re-testing. This can lead to greatly minimized overall development and testing time.

The key questions that arise during the development of advanced UAV systems may be:

1) Are we sure how the robot will response to the input from navigation procedures and sensors?
2) Will it be able to accomplish the task within given time regime?
3) What are the limitations of communication protocol?
4) How will robot react if emergency situation, such as loss of communication with navigation module, occurs and how procedures for such event will work?

Artificial testing environment can help minimize the risk associated with these questions. Nevertheless, some uncertainty is related specifically with communication protocol which is utilized between navigational module (which includes AI functions) and autopilot module which is responsible for execution of low-level tasks, such as robot's movement, power management, basic sensors reading (such as barometer, GPS, gyroscope). There is a number of existing simulators for particular UAV controllers and, on the other side, more general approaches, where testing environment is designed for abstract robot, not associated with particular communication protocol or controller model [2], [1]. The challenge is to create the testing environment which directly incorporates the communication protocol (such as MavLink) that will be used in real UAV. In such scenario the AI module sends the instruction directly to the navigation module utilizing the given communication protocol, but the system can be connected either to the emulated or real UAV.

The solution presented in this paper introduces an approach, which enables testing solutions that are ready to be applied on real UAV. It faces that challenge by combining emulation of robot which communicates via MavLink protocol with visualization of the simulated environment. The MavLink protocol (Micro Air Vehicle Communication Protocol) is one of the most popular protocols for communicating with robots' control stations (also called *autopilots*), sending commands and exchanging telemetry information. It is a lightweight, header-only protocol utilized by controllers such as Pixhawk PX4 (see https://pixhawk.org/), APM 2.6 (see http://ardupilot.org/copter/) or SLUGS Autopilot [3], [4]. Such approach in the design of UAV simulator allows not only to test a considerably wide range of solutions but also to directly use existing code to connect to real UAV, utilizing MavLink protocol, immediately after finishing simulated tests. This can be accomplished by simply changing the connection from the server representing emulated UAV to a real robot connected to PC via telemetry transmitter/receiver.

Such simulator can be applied in various scenarios. The most basic one would be testing of robot's reactions to movement requests, sent from the controlling application, where a graphical interface would allow observation of potential responses of the real robot and tracking unexpected behaviors. This simulator can be applied for testing of more sophisticated solutions. One of the typical classes of algorithms that can be tested in such simulator are object recognition [5], [6] and scene analysis [7], [8]. These problems are widely studied in

robotics and can contribute to the solution of more complex tasks, such as scene understanding and robot localization [9], [10].

The paper is structured as follows. Section II discusses the design and functionalities of the proposed simulator. Next section (see III) describes the core elements of scene analysis system. It briefly introduces algorithms for vectorization of the images, recognition of the objects in the scene and three-dimensional model building. In section IV the authors propose the set of methods for navigation of a UAV to allow operating in an urban environment, utilized in tests for the solution. Section V is dedicated for presentation of test showing capabilities of scene analysis system as well as the simulator. The scenario of the tests is based on the idea of UAV, equipped with visual sensor, operating in an urban environment. The main task of this robot is to locate predefined object in the scene and build a three-dimensional model of the located building (later in this paper the terms *object* and *building* will be used interchangeably due to assumption that the robot operates in an urban environment). The last section is dedicated for concluding remarks.

II. Artificial testing environment

In this part the environment for simulating execution of a real UAV's tasks is introduced. The designed system is dedicated for testing UAV which utilizes MavLink protocol. The solution is based on combination of lightweight SITL (Software In The Loop) simulator and scene model which is generated in OpenGL environment. The whole system was implemented in Python language. The SITL simulator is provided alongside with libraries for communicating in MavLink protocol. It is aimed to allow simple tests of sending command to the robot and receiving telemetry information from it. When the simulator is started as a server in local machine it responds to the input as if it was a physical UAV using MavLink protocol. This is enough for testing simple commands of movement and receiving information from the robot. To allow the simulator to execute more complex tasks it was combined with OpenGL module which enables visualization of actions executed by a simulated UAV. The important outcome of this is the possibility to use OpenGL camera to work as a visual sensor of the robot. The input from that virtual camera is a source of information that can be provided to the complex scene analysis algorithms. By combining these two elements the researchers and developers are allowed to directly switch from using simulator to executing actions on real MavLink-enabled UAV, equipped with visual sensors.

The overall design of the simulated environment is shown in Fig. 1. In this system, the *UAV module* and *Image processing module* operate together as the central control unit. It processes information from sensors (video feed from OpenGL in the presented simulated environment), executes complex tasks related to image processing and conducts interaction with a physical flying robot or a simulated one (*SITL simulator*) like it is presented in this paper. Practically, this control unit is the main part of an unmanned flying agent responsible for

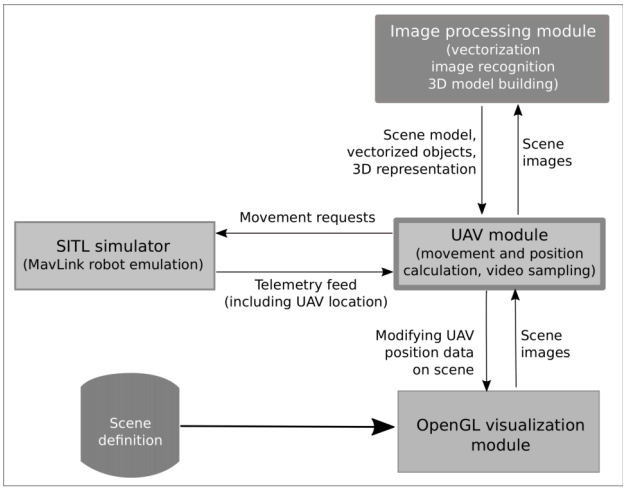


Fig. 1: Overview of the system modules and communication in the simulated environment.

all higher level functionalities over those that are ensured by *autopilots* such as PX4 or APM 2.6 and communicates with the *autopilot* with a use of the MavLink protocol.

III. Scene analysis algorithms

This section is aimed to briefly introduce a set of scene analysis methods that were combined to form a complete task for a UAV, tested in the simulator. These algorithms were designed and implemented in another part of the project to create the core elements of visual system for autonomous flying agent. More details about these methods and their possible applications can be found here [11], [12], [13]. These algorithms can also be utilized for solving more complex problems, such as cognitive approach to scene analysis and recognition [14]. Let us briefly introduce these algorithms below.

A. Vectorization method

This method is aimed to obtain memory-efficient, vector representation of objects in the examined scene. The amount of information describing the shape of an object is limited to a list of points in Euclidean space which are located in the corners of the object. The steps which lead to creation of that representations are:

1) **Pre-processing.** As the pre-processing of the images is outside of the scope of discussed project is is assumed that objects that are subject for analysis have distinctive colors. This allows extracting objects from the image by filtering specific colors from predefined set.
2) **Border extraction.** The result of pre-processing which consists of extracted color shapes on the black background are subject for edge detecting process. To extract edges of the shapes, a recursive algorithm for two-dimensional edge detection is used [15].
3) **Point sequence generation.** A dense sequence of points, located along the edge of each object, is generated. The

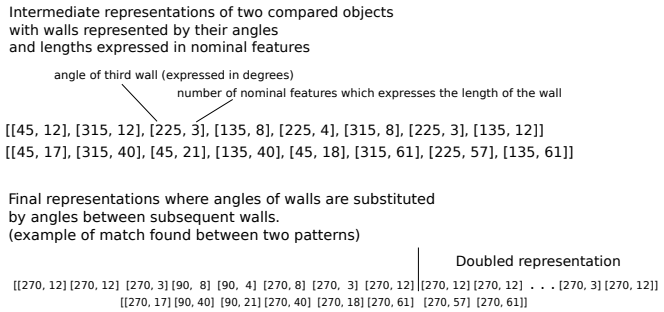


Fig. 2: Representation obtained from vectorized object, used for recognition process.

points are evenly distributed and the distance (calculated in number of pixels from original image) between them depends on predefined parameter. The distance has impact on shape description accuracy and the greater distance is, the less accurate description can be obtained with the increase of efficiency of the algorithm at the same time.

4) **Removing redundant points.** Some points are redundant in the sequence if they lay on the same line or change the path insignificantly. To remove them the Ramer-Douglas-Peucker curve simplification algorithm [16], [17] is utilized.

The final step is the enrichment of the object representation with information about its color. To do this a pixel located inside the outline of examined object is selected and its RGB color description saved along with the vector representation. Further it will be used to prepare images for 3D object model construction. The particular images, showing examined building from various sides (see section III-C) will be filtered to select only this color which is associated with the color of the object which was first vectorized and recognized based on the image taken from above. This is done to ensure that the contours that are discovered in the images taken during examination of the building belong to the desired (examined) object.

B. Object recognition

For object recognition a syntactic algorithm was proposed. It allows a rotation and scale invariant matching which is crucial for UAV application as altitude and direction of flight may differ from one scenario to another.

In Fig. 2 the representation used in matching algorithm is presented. To allow rotation invariant matching, the representation of first object is doubled as it is highly probable that starting corner (first in vector representation) of one object is different from the one that belongs to the representation of compared object. More comprehensive information about this algorithm can be found in [11].

C. Three-dimensional model building

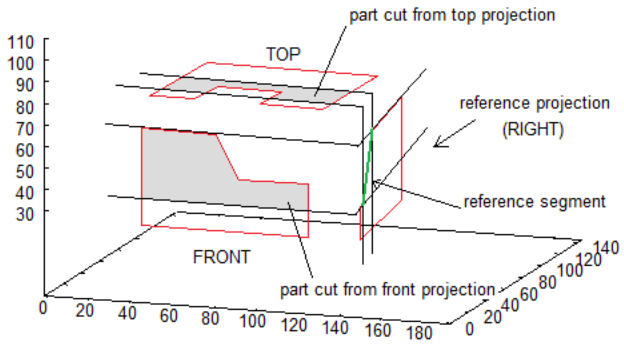
Another application where the vector representation described above is utilized is three-dimensional object model construction. Such a representation can be used in various applications. As an example we can consider ground inclination and obstacle shape approximation [18] and objects shape modeling for collision-free navigation and inspection tasks in an urban environment [13]. Such model is built using projections which are obtained from images taken from three sides of the examined object - top, front, right. This allows to create a simplified model carrying information about overall shape of the object as well as configuration of separate walls and features such as holes in object's structure. The process of model construction is divided into steps in which particular walls of the resulting 3D structure are derived separately, based on *reference* projection and two other projections which are cut into fragments and adjusted to form the third dimension. In Fig. 3 a process of single wall creation is presented, where a *reference* projection is the right one and top and front ones are being cut.

IV. Navigation methods

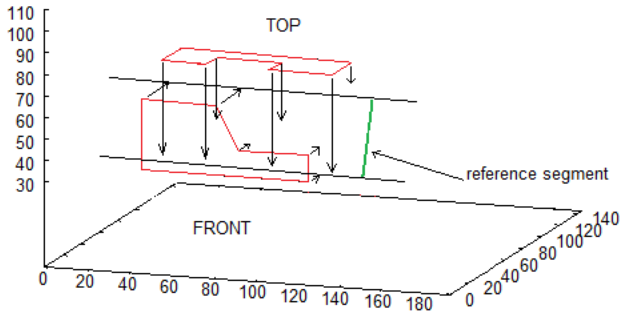
Some of the most important tasks for a UAV are target position calculation and path planning. They are exceptionally critical when operating in urban environment and during inspection tasks. In such applications robot is required not only to avoid obstacles (even when flying on high altitude) but also to position itself precisely to execute given tasks [10], [19], [20]

The scenario presented in section V require utilization of precise navigation as well. To support UAV with necessary capabilities the following methods were provided:

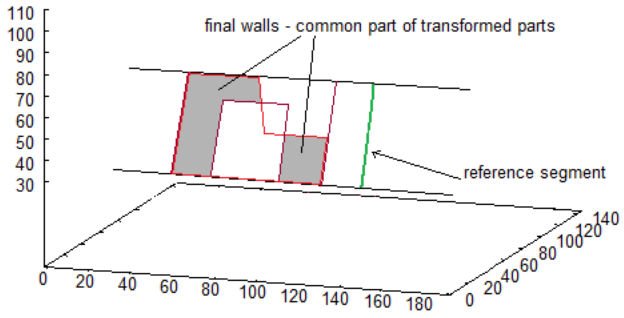
1) Calculating position of Points Of Interest (later referred to as POI) in the area photographed from high altitude. The POIs are related with objects (buildings) that, in subsequent steps, are subjects for recognition in order to find the specific one which shape is similar to one's that was initially stored in the robot's memory. Each POI has to be visited in order to collect an image revealing the exact shape of the object observed from position straight over it.
   This calculation is done by a function *calcMoveToTargetHorizont()* executed for each POI. The function returns distance to North and East to a point in the photo, given as an argument. First the distances along X and Y axis (in pixels) are calculated with reference to the center of the photo which represents the point over which the UAV is currently located. Next, using information about drone's altitude (read from barometer sensor) given as an argument, those X and Y values are converted to real distances in meters. Finally the function uses drone's heading direction (read from compass) to count the real shift to North and East (as the image of the scene is not necessarily taken while the drone is facing North).
2) Calculating target position to collect images necessary for building three-dimensional representation of an object. This is executed once the UAV is positioned straight over the building which is recognized as the one to be examined.



(a)



(b)



(c)

Fig. 3: Steps of creating walls in 3D model building, (a) - cutting from projections, (b) - projecting onto a plane, (c) - intersection of intermediate walls

To accomplish this, the function *calcHeadingChangeForImage()* was introduced. It chooses the place and direction for the front and right-side photo of the building. The idea is to calculate the smallest rectangle that can be circumscribed on a figure of a building. For the *front* image we choose the side of the building associated with one of the longer edges of obtained rectangle. It is done so, because the *front* image is supposed to give the best overview of the examined building. After the front side is identified, the distance from the building is calculated using the camera's vertical and horizontal angles of view. The position for the second photo is set in front of the side of the building associated with shorter edge of the circumscribed rectangle. It means that direction in which camera has to be pointed is perpendicular to the direction in which the *front* side is photographed.

Finally, it is checked if any of two chosen points collides with any other object on the scene. If so, another possible point is searched. It means that the opposite side of the building is selected or distance from the building is changed if opposite side also gives colliding position. If there is no collision, the function returns heading changes and coordinates for both of the chosen spots.

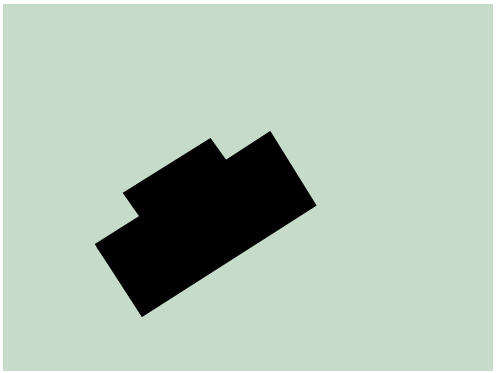


Fig. 4: Bitmap image of searched object provided to UAV.

V. Tests

This section presents the results of the tests to show capabilities of the simulator introduced in this paper. These test also bring in action the scene analysis algorithms discussed in previous sections - vectorization, image recognition and 3D model building. Test scenario is revealed step by step in this section and supported by figures showing most essential output. Let us briefly outline the scenario:

1) First, the UAV is provided with an image of the object (taken from above) which is going to be searched in the simulated environment.
2) The robot, using the image of the scene taken from high altitude, locates objects in the scene and calculates route to take more detailed pictures of each of them.
3) It flies over each object and takes detailed images from lower altitude. Each photographed object is compared with the searched one.
4) Once the searched object is located the UAV performs closer investigation of the building to collect images necessary for 3D model building.

In Fig. 4 and 5 a searched object and its vector representation is presented. This object is provided to the UAV to be located in the simulated environment. The vectorized object will be an input for image recognition algorithm discussed earlier in this paper.

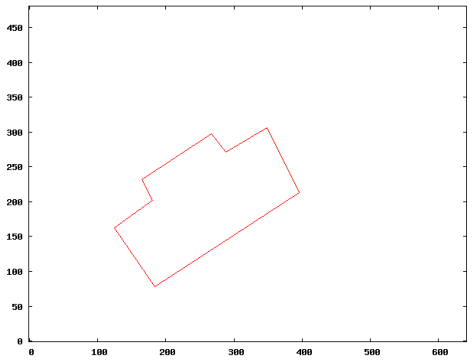


Fig. 5: Vector representation of searched object.

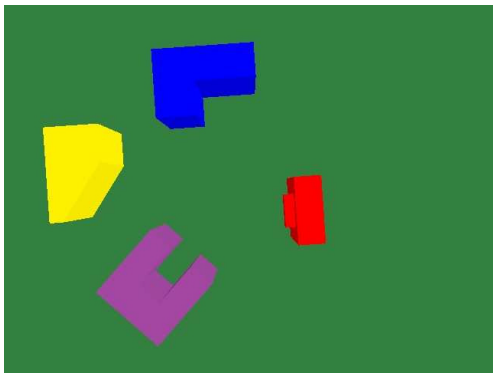


Fig. 6: Scene generated in simulated environment, photographed by the UAV from high altitude.

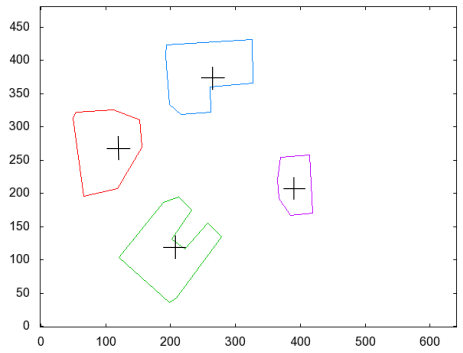


Fig. 7: Vectorized image of the scene. Calculated destination points to take precise images are marked.

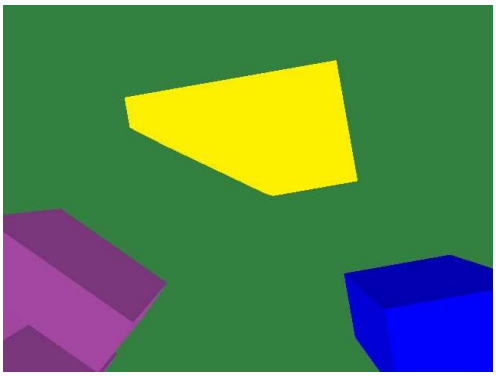


Fig. 8: Detailed image of yellow building.

The next figure (see Fig. 6) shows the first image taken by the UAV and represents whole scene with four buildings. To take this picture the UAV climbed up to predefined altitude of $h = 45$ meters. In the following figure (see Fig. 7) the same scene is shown vectorized. This vectorization is done to allow UAV to find all structures in the scene and calculate target positions from where it will be possible to take more precise images. These images will further be compared with searched object to find the one for which the 3D representation will be built.

In the next step the destination points were added to the list. It was then provided to UAV module to initiate a task of visiting each point. After reaching each of the points from the list (at predefined altitude of $h_2 = 20$ meters) a detailed image was taken (see Fig. 8, 10, 12, 14), vectorized (see Fig. 9, 11, 13, 15) and provided to image recognition algorithm.

The following figures (see Fig. 14 and 15) show the image and vector representation of the building that was recognized in the scene as the one that was searched. During this step, when the object was recognized, its RGB color code is stored along with vector representation. It will be used in next steps for proper extraction of objects photographed horizontally against other objects in the scene.

In this particular test it turned out that the *red* building was the last on the list and all buildings were visited before this one. In more general scenario it is not necessarily the case as the searched object can be found earlier and the task of flying over all targets can be terminated.

After the searched object was recognized the UAV initiates next task to scan the object and build 3D representation. To accomplish this, the following target locations are calculated and provided to simulated UAV:

1) position over the building with UAV heading set to take a proper image which will be used as one of three projections (the *top* one).

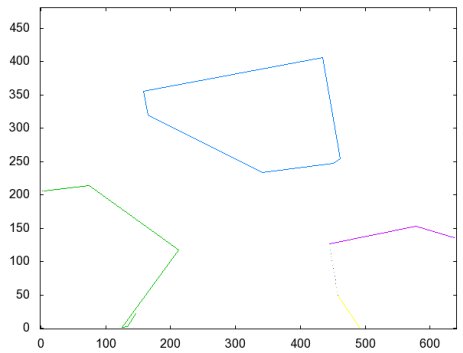


Fig. 9: Vector representation of yellow building.

Fig. 10: Detailed image of purple building.

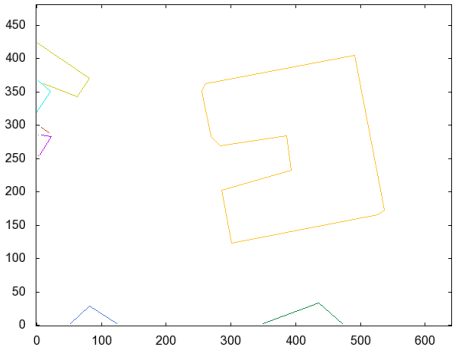


Fig. 11: Vector representation of purple building.

2) turn the camera from pointing downwards to front in order to take horizontal images. In real UAV it can be done with the use of gimbal stabilizer.
3) position on the front side of the building to take *front* projection.
4) position on the right side of the building to take *right* projection.

Figures 16, 17 and 18 present images of the examined object taken by the robot from the above mentioned positions respectively.

In the next set of figures (see Fig. 19, 20 and 21) there are vector projections of the examined object, obtained from above mentioned images. The RGB color code (red tint in this case), stored with vector representation of *top* projection, is utilized in this step in order to properly extract shapes of the objects in the images taken horizontally. This is done in image pre-processing by filtering out all colors that are different from the stored one.

The final step is the creation of three-dimensional representation of the object that has been found and closely examined. Figures 22 and 23 show the final result of three-dimensional model building algorithm.

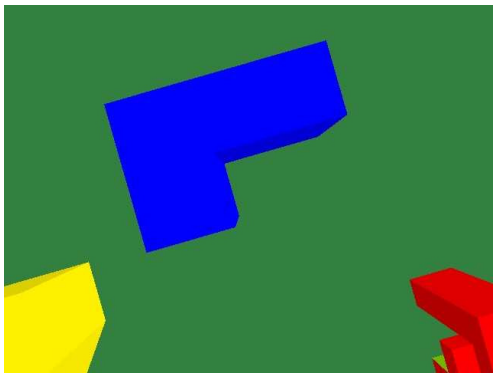


Fig. 12: Detailed image of blue building.

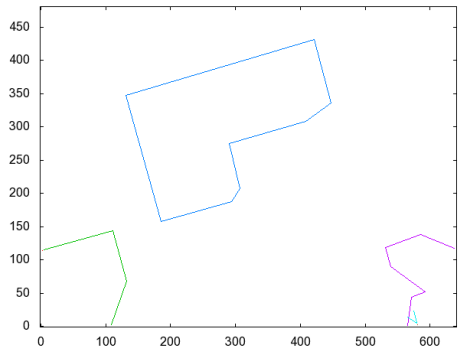


Fig. 13: Vector representation of blue building.

Fig. 14: Detailed image of red building.

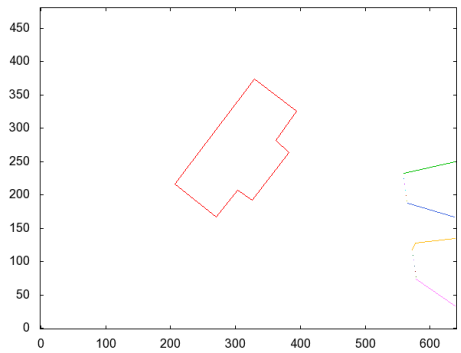


Fig. 15: Vector representation of red building.

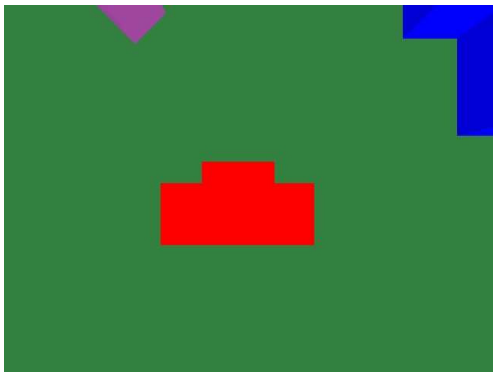


Fig. 16: Image of examined object taken from above. Direction is the same as for front image which makes it directly suitable for 3D model building.

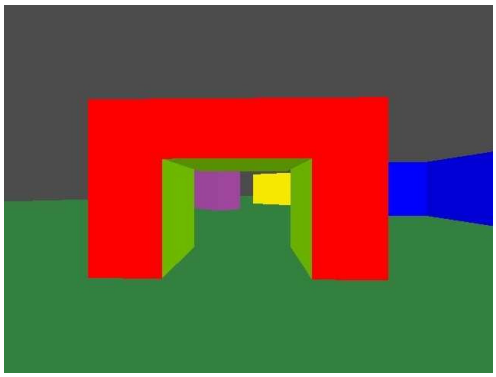


Fig. 17: Image of examined object taken from frontal position of UAV.

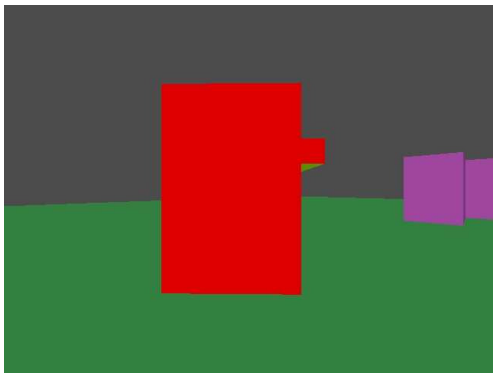


Fig. 18: Image of examined object taken from the right side of the examined object.

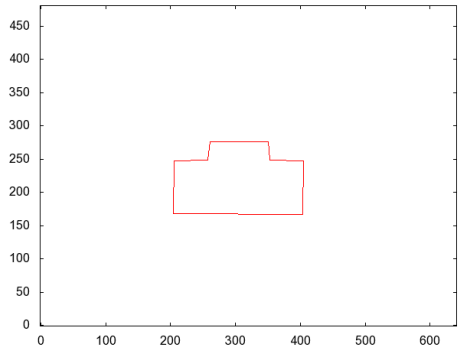


Fig. 19: Vector representation of the object photographed from above.

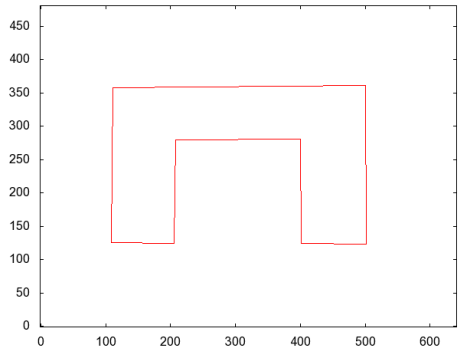


Fig. 20: Vector representation of the object photographed from frontal position of the UAV.

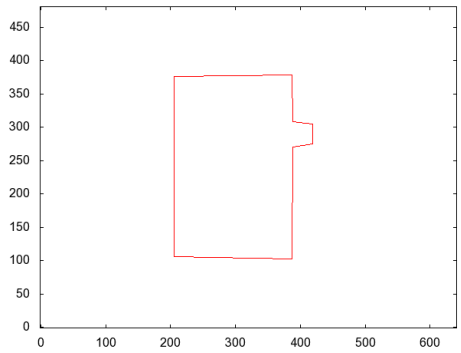


Fig. 21: Vector representation of the object photographed form the right side.

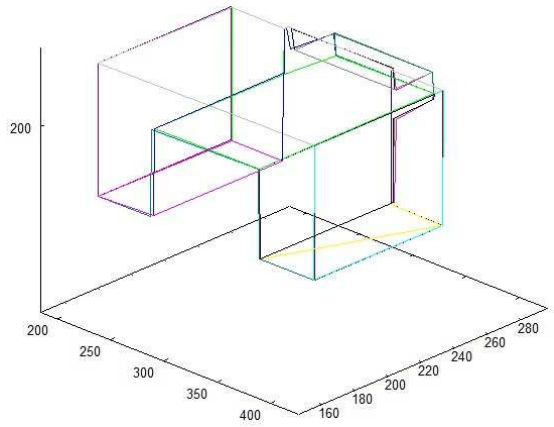


Fig. 22: The result of three-dimensional model building algorithm.

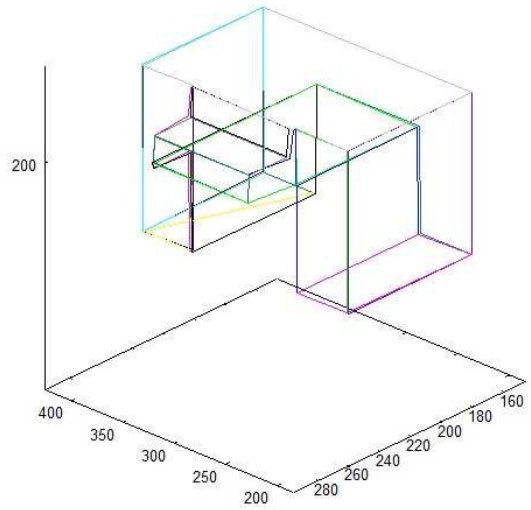


Fig. 23: The result of three-dimensional model building algorithm presented from a different angle.

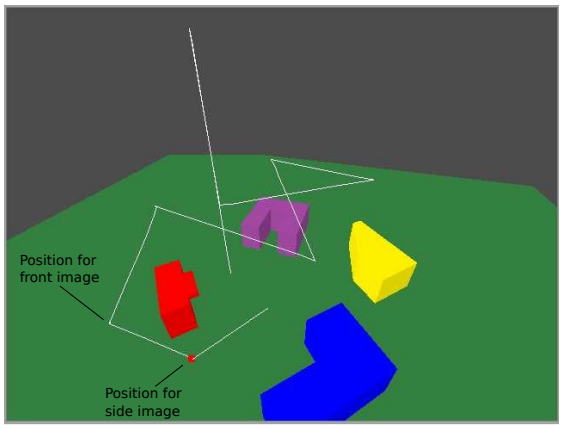


Fig. 24: The route of the UAV executing tasks in simulated environment. The locations from which the robot took images of examined object are marked.

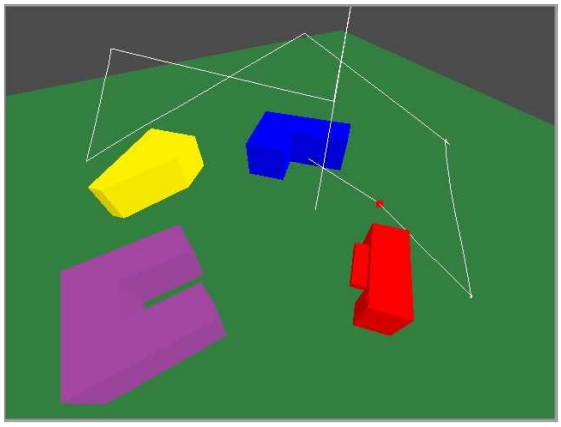


Fig. 25: The route of the UAV shown from different angle.

Additional functionality of the presented simulator is the tracking of movement of the UAV. Figures 24 and 25 show the path that was followed by the UAV during the tests. It can be noticed that the robot started from the middle of the scene and immediately climbed to high altitude to take overview image of the scene (shown in Fig. 6). Then, on a lower altitude, all objects on the scene were visited in the sequence: *yellow*, *purple*, *blue*, *red*. Additionally, in Fig. 24 the points from which the UAV took images for 3D model building are marked.

VI. Conclusion

In this paper the authors discussed the simulator for Unmanned Aerial Vehicle. Its characteristics, including MavLink protocol utilization, were introduced. The design of the simulator allows to connect AI both with navigation module directly to a real UAV which can shorten the time between development of the robotic solution and its implementation on a real UAV. The authors also introduced the vision system which was tested on the proposed simulator. The aim of the tests was to show that the vision system modules, such as object recognition and three-dimensional model building, can be combined to allow execution of complex tasks. The test results show the capabilities of the vision system which was applied for searching and analysis of the objects in the modeled scene.

The further work will be focused on extension of functionalities of the simulator. It will include monitoring of the UAV state, based on telemetry information send via MavLink protocol. Also, the visual module of the simulator will be developed to simplify switching from OpenGL view of the simulator to a vision sensor of a real UAV. In terms of vision system the cognitive module for close inspections will be developed. It will allow a robot to identify features and shape details of the examined objects and utilize the obtained information for navigation close to the structures.

REFERENCES

[1] D. Cook, A. Vardy, and R. Lewis, "A survey of auv and robot simulators for multi-vehicle operations." in *Proceedings of 2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, vol. 2014, pp. 1-8, 2014.

[2] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ros and gazebo." in *Proceedings of 2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, vol. 2016, pp. 96-101, 2016.

[3] B. Fuller, J. Kok, N. Kelson, and F. Gonzalez, "Hardware design and implementation of a mavlink interface for an fpga-based autonomous uav flight control system." in *Proceedings of Australasian Conference on Robotics and Automation*, vol. 2014, pp. 62-67, 2014.

[4] T. Dietrich, O. Andryeyev, A. Zimmermann, and A. Mitschele-Thiel, "Towards a unified decentralized swarm management and maintenance coordination based on mavlink." in *Proceedings of International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, vol. 2016, pp. 124-12, 2016.

[5] M. Flasiński, "On the parsing of deterministic graph languages for syntactic pattern recognition." *Pattern Recognition*, vol. 26, pp. 1–16, 1993.

[6] R. Tadeusiewicz and M. Flasiński, *Pattern Recognition*. Warsaw: Polish Scientific Publishers, PWN [in Polish], 1991.

[7] M. Bielecka, M. Skomorowski, and A. Bielecki, "Fuzzy syntactic approach to pattern recognition and scene analysis." in *Proceedings of the 4th International Conference on Informatics in Control, Automatics and Robotics ICINCO07, ICSO Intelligent Control Systems and Optimization, Robotics and Automation*, vol. 1, pp. 29-35, 2007.

[8] M. Flasiński, "Parsing of ednlc-graph grammars for scene analysis." *Pattern Recognition*, vol. 21, pp. 623–629, 1998.

[9] D. Filliat and J. Mayer, "Map-based navigation in mobile robots. a review of localization strategies." *Journal of Cognitive Systems Research*, vol. 4, pp. 243–283, 2003.

[10] L. Muratet, S. Doncieux, Y. Briere, and J. Meyer, "A contribution to vision-based autonomous helicopter flight in urban environments." *Robotics and Autonomous Systems*, vol. 50, pp. 195–229, 2005.

[11] A. Bielecki, T. Buratowski, and P. Śmigielski, "Syntactic algorithm for two-dimensional scene analysis for unmanned flying vehicles." *Lecture Notes in Computer Science*, vol. 7594, pp. 304–312, 2012.

[12] ——, "Recognition of two-dimensional representation of urban environment for autonomous flying agents." *Expert Systems with Applications*, vol. 40, pp. 3623–3633, 2013.

[13] ——, "Three-dimensional urban-type scene representation in vision system of unmanned flying vehicles." *Lecture Notes in Computer Science*, vol. 8467, pp. 662–671, 2014.

[14] A. Bielecki and P. Śmigielski, "Graph representation for two-dimensional scene understanding by the cognitive vision module." *International Journal of Advanced Robotic Systems*, vol. 14, pp. 1–14, 2017.

[15] J. Canny, "Finding edges and lines in images." M.I.T. Artificial Intelligence Lab., Cambridge, MA, Tech. Rep., 1983.

[16] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves." *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244–256, 1972.

[17] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature." *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[18] A. Bielecki, T. Buratowski, M. Ciszewski, and P. Śmigielski, "Vision based techniques of 3d obstacle reconfiguration for the outdoor drilling mobile robot." *Lecture Notes in Computer Science*, vol. 9693, pp. 602–612, 2016.

[19] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: a survey." *Journal of Intelligent and Robotic Systems*, vol. 53, pp. 263–296, 2008.

[20] B. Sinopoli, M. Micheli, G. Donato, and T. Koo, "Vision based navigation for an unmanned aerial vehicle." in *Proceedings of the International Conference on Robotics and Automation ICRA*, vol. 2, pp. 1757-1764, 2001.