Examples of Real-Time UAV Data Processing with Cloud Computing

Peter Šuľaj, Renát Haluška, Ľuboš Ovseník , Stanislav Marchevský
Technical University of Košice
Košice, Slovak Republic
Peter.Sulaj@tuke.sk, Renat.Haluska@tuke.sk,
Lubos.Ovsenik@tuke.sk, Stanislav.Marchevsky@tuke.sk

Vadim Kramar
Oulu University of Applied Sciences
Oulu, Finland
Vadim.Kramar@oamk.fi

***Abstract*—This article explores cloud-based data processing object detection for future anti-collision system and georeferencing. At present, data processing from Unmanned Aerial Vehicles (UAVs) is based, in most cases, on sequential image acquisition and subsequent processing, but there is a demand for real-time resolution. Real-time data processing can be beneficial to track changes accurately. The help of real-time kinematic sparse data generation is a possibility for new remote sensing applications. Sensors of light-weight and low-cost UAVs provide the opportunity to create target data in real-time.**

I. Introduction

These days the cloud services for data storage, processing and access are widely spread due to their easy access, data loss prevention and flexibility. According to Gartner research, the global market of cloud services will expand to more than 300 billion of U.S. Dollars in 2021 comparing to about 150 billion in 2017, showing a double grow [1].

In the modern world of robotics that assumes fast and low-latency data transfers, cloud-based data processing may be considered appropriate for signals processing applications. Using the cloud services makes it possible to reduce the development time and effort by moving the computational power into the cloud and automating many elements of a data-processing chain.

Unmanned Aerial Vehicles (UAVs) sense the environment and process data for two purposes, operational (e.g. [2]) and mission-specific data collection. Since UAVs are utilised in different operational conditions and for a great variety of application domains, such as monitoring, anti-collision, sensing, security, military, surveying, and many other, nature of data may be very diverse. Due to limitations brought by battery life, limited computational and data storage ability, utilisation of cloud services for data processing and storage looks attractive.

Different software solutions could provide different data products and different properties. A wide range of impressive software solutions took a unique and diverse approach to data processing. Essentially, the software packages offer diversity in mechanisms and algorithms. Different approaches provide different results with similar data sets. Gradually, more advanced processing options have emerged that have enabled the performance to be tightened and the available decision-making capabilities to make the programs best suited to their use and standards [3].

The rest of the paper is organised as follows. In Section II, a short overview of cloud computing concerning UAV data is given. Object detection system is introduced in Section III. Georeferencing is overviewed in Section IV. In Section V, experimental setup and test results are described. Concluding remarks are given in the Conclusion section.

II. Cloud Computing and UAV Data

By spreading computing technologies to a variety of areas, data processing is more often moved to and performed in the cloud. With this approach, it is possible to obtain a manageable computational power and extended 'end-to-end' workflow automation. Cloud-based platforms can enhance the availability of data processing performance without the need to invest in server infrastructure or high-performance computers. Such a great advantage is successfully adopted many start-ups [4].

However, in that kind of approach the transparency and processing capabilities of locally-operated software modules are hidden behind the automation veil or completely eliminated for operational simplicity. Still, there are several issues associated with the use of cloud services. Among those are data security in the cloud, the bandwidth required for data transfer, available for data processing load and delays caused by cloud computing [5]. Those all are particularly significant in case of UAVs.

Cloud services supporting robot technologies can be used for real-time knowledge sharing and maintaining datasets. This increases the potential for robot's intelligence. Cloud support can handle difficult tasks by being less expensive comparing to built-in-robot intelligence and bring opportunity to easier maintenance. For example, system updates can be arranged in real-time and without the data loss. Already now it is possible to anticipate that the future robotic systems will be developed in the form of hardware that may be reused and software that will utilise the cloud infrastructures.

For example, for automating the workflow of processing the signal from the camera, the automated software must produce a large set of assumptions about the data it encounters. The role of the data processor is to minimise the assumptions and ensure the expected outcome. If the images in the dataset include differences between light, camera positions, and orientation derived from the inertial navigation system, these values should always be introduced before their

introduction into the system and their revised values then reviewed. This process is stricter if an estimated default deviation of inputs can be included a priori. The proposed by many cloud-based platforms instruments have a wide range of processing options, they are mostly aimed at terrain-structures, RGB or colour-infrared only [6].

If it is not possible to modify the software to meet the processing requirements above the level of automated processes proposed by the provider, it is necessary to program the required software packages. Simple data products, such as elevations and orthoses, are quite well generated in these systems. For further data manipulation, third-party software will be required to exchange the necessary data with the cloud provider. Many software modules from the service provider offer not only problem recognition during each data processing step, but also the tools needed to resolve problems before continuing processing [7].

III. Object Detection System

In the world of automated vehicles such as UAVs, the very important task is to detect and recognise nearby objects. These tasks require high computational power and therefore affect the battery consumption. One of the essential solutions for that is to process the demanding operations in the cloud.

The object may be identified with the help of different sensors such as cameras, lidars, depth cameras, infrared sensors, ultrasonic sensors, etc. The detection of live or inanimate objects in the nearby space is mostly acquired with expensive equipment. For detecting the objects, 2D Light Detection and Ranging (LiDAR) and Pan/Tilt platform for the camera (Fig. 1) were used. The system uses LiDAR to detect the object in the space and the camera to recognise which object it is.

Typically, LiDAR sensors use polar coordinates for measurement. More expensive models include multiple laser sources attached at different angles, allowing you to scan data in 3D [7]. By connecting 2D lidar, it is possible to detect an obstacle/object and use the camera to determine what object it is.

The main task in building an object detection system is to merge data from various sensors [8], e.g. UAV/mounted LiDAR and RGB camera. A suitable means of communication for the acquisition of data from sensors and subsequent processing is the Robotic Operating System (ROS) framework.

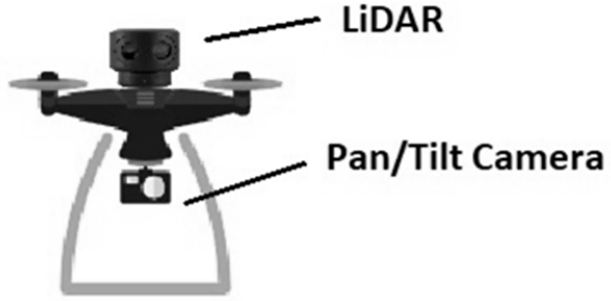

Fig. 1. The concept of an object detection system

The ROS has support for several UAVs depending on the release used. Supported UAVs are as follows:

- Erle-copter
- AR.Drone from Parrot
- Bebop from Parrot
- Skybotix Coax Helicopter
- DJI Matrice 100 Onboard SDK ROS support
- PIXHAWK MAVs
- Penn's AsTec Hummingbird
- Bitcraze Crazyflie
- Gapter
- Lily Camera
- AsTEc Pelican

ROS includes a variety of software modules for processing data from different sensors. One of the ROS features is the ability to create MASTER/SLAVE communications. This allows data to be processed on the remote server. In the given case, Raspberry Pi single-board microcomputer was utilised in the slave function that obtains sensor data and sends it for further processing to a more powerful computer located in the cloud [9].

ROS contains some object recognition packages, such as *find_object_2D*, which has implemented detectors: SURF, SIFT, FAST, and BRIEF. Labelling objects and saving them allows a graphical user environment. In case 3D coordinates of the object are needed, it is possible to use a deep camera such as a Kinect [10]. To use the ROS with Kinect, one need to install the driver package.

Subsequently, it is possible to identify an object from the RGB data and calculate the distance from the UAV using the depth map. For that, it is, necessary to have more kinetics and cover the area around the UAV and thereby the volumes of transmitted data to the cloud increases.

A. Data Acquisition and Detection Objects by LiDAR

Most 2D LiDARs scan the space at an angle of 180 ° over 13 ms with an angular resolution of 1° [11]. The higher resolution of scanning space will cause slow scanning. When scraping 0.25°, scanning time is 52 ms [12]. The data from these sensors is in a simple form and therefore is suitable for detecting objects with mapping the environment. LiDAR communicates with an external device, in this case, Raspberry Pi, via the USB <-> TTL converter. If Raspberry Pi sends a system command to LiDAR, LiDAR then sends the corresponding response back to Raspberry Pi (Fig. 2).

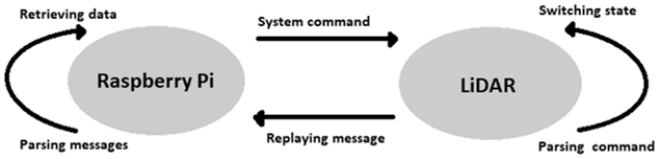

Fig. 2. System communication mechanism

Information about the angle that is currently being read by LiDAR is encoded in the First registers Sample Angle (FSA) and Last Sample Angle (LSA). Each scan angle includes the following data structure.

$$Distance_i = \frac{Sampling\ data}{4} \quad (1)$$

$$Angle_{FSA} = \frac{Rbit(FSA,1)}{64} + AngCorrect_1 \quad (2)$$

$$Angle_{FSA} = \frac{Rbit(LSA,1)}{64} + AngCorrect_{LSN} \quad (3)$$

$$Angle_i = \frac{diff(Angle)}{LSN-1} * (i-1) + Angle_{FSA} + AngCorrect_i \quad (4)$$

Rbit represents the accuracy of the data shift by one bit. The difference of the initial angle to the end angle in the clockwise direction is *diff(Angle)*. Angle correction *AngCorrect* is achieved using the formula.

$$AnglCorrect_i = tand^{-1}\left(23.6 * \frac{180 - Distance_i}{180 * Disatnce_i}\right) \quad (5)$$

Data obtained by scanning space is published as */scan* topic to cloud. The Rviz program, which contains the ROS desktop version, can be used for data visualisation in the cloud. To view data using the Rviz program, it is necessary to add a topic name in this case */scan* to the display panel. Laboratory testing in a space with flat edges of objects does not require any further modification of a two-dimensional map plotted in coordinates (x, y). With the increasing variability of the space, the resulting readability of the two-dimensional map deteriorates [13].

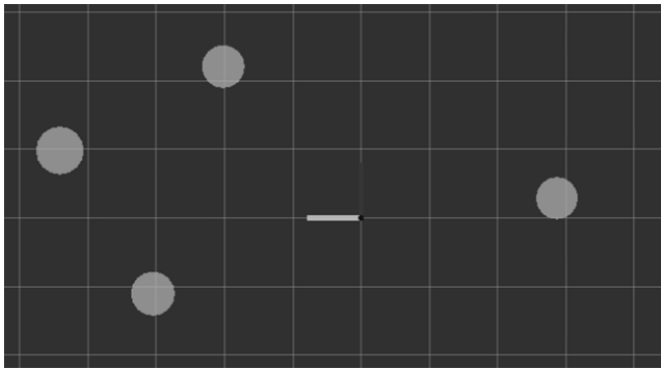

Fig. 3. Example of the output from the software

The appropriate path is to use the *obstacle_detector* package for ROS, which provides *scans_merger*, *obstacle_extractor*, and *obstacle_tracker*. Obstacles are displayed in the circle. At this point, the larger the circle is, the larger the object. This software uses the Kalman filter to filter out obstacles. The object tracking algorithm only applies to circular-shaped obstacles, and it is also possible to obtain information about the speed of detected objects [14]. The parameters needed to set extraction, tracking, and publishing can be changed through the graphical user interface (Fig.3). After detecting the closest object of its speed and direction, it is possible to set the camera to the given direction and to recognise the object.

B. Camera Object Detection

Object detection based on machine learning needs to select a specific object from a group of objects. Various image processing algorithms (segmentation, Haar patterns, etc.) can be used for detection. The disadvantage of some algorithms is that they cannot be used to detect multiple objects [15]. Using machine learning algorithms, it is possible to train a neural network with a large collection of data. This makes it possible to obtain accuracy and scalability. Using machine learning is a new approach control for autonomous UAVs in which data from sensors can be sent to the cloud, and consequently, the result can be used to control the unmanned vehicle.

Python programming language is one of the most widely used languages for machine learning. Many of the library libraries developed by many developers have been created. Since Python is the interpretive language, NumPy and SciPy libraries have been developed to increase the computing speed. There are many options for object recognition, one of the options is multiple detection algorithms such as You Only Look Once (YOLO), RetinaNet, Single Shot Multibox Detector (SSD), Faster R-CNN. These algorithms can be implemented using a Machine Learning (ML) library such as TensorFlow, Theano, Torch or deep learning library such as Caffe or optical character recognition (OCR) engine such as Tesseract [16].

Nowadays, TensorFlow enjoys great popularity, an open-source library developed by Google. TensorFlow can be used with cloud services such as Azure from Microsoft or Amazon Web Services (AWS). This library allows you to perform neural network calculations on one or more CPUs, desktop or mobile devices, or GPUs.

For applications where UAVs are used, the emphasis is also placed on battery consumption and therefore the time for which the UAV can be in the air. For this reason, it is necessary to limit battery consumption by the central processing unit and to perform more demanding operations in the cloud [17]. Implementation of detection of other objects is possible via Nanonets Machine Learning API. Working with the NanoNets Machine Learning API is easier to implement because it requires fewer steps than Tensorflow. The entire implementation with Nanonets has the following steps:

- uploading images
- labelling of images
- training of the model
- testing and integration [18].

Once the API key is obtained, it is necessary to add it to the system using the *export* command. Subsequently, an algorithm for object detection can be implemented.

Algorithm 1 Object detection

Input: ID of the model.
Input: API key for connecting to the cloud.
Input: Image from the camera.

Output: Image with detected objects.

result ← *response from the server*
for each point in *p* on the test image **do**
  b ← □□□□, □□□□, □□□□, □□□□
  draw a rectangle to the image
end for
Print *result*
Show *image*

Fig. 4. Object detection test

There is no need for an expensive GPU, that means the system is easily applied to mobile devices like mini PCs. The system also automatically selects the best model and hyperparameter selection [19]. The training process for the size of 55 UAVs lasts more than 10 minutes. After integration, the resulting detection is shown in Fig. 4.

IV. GEOREFERENCING

Georeferencing is the process of aligning the data file raster to known map coordinates and assignment of the coordinate system. This system contains additional information within the file itself and/or the additional files that are created with the image file field that speaks GIS software to place and draw correctly. It is a very important step to create aerial and satellite imagery useful for mapping.

Different maps can use different projection systems. The most commonly used are cylindrical, conical and azimuthal (Planar) projections. The cartographic impressions have several benefits. Azimuthal has the smallest distortion in the centre of the map, conical has the least distortion at the point of contact with the globe, and the cylindrical has the smallest distortion in the equator area and the largest in the pole area. Every map projection of the Earth causes some distortion. Geo-referencing tools include methods that we can be combined to get the map as light as possible and achieve maps' overlapping with minimal distortion.

A positive feature is that if the shape of the small characters is preserved, then the x and y projection scales are always the same and have the same properties, this is true if the distorted and mapped shapes are the same as the surface area of the surface. Searching for points on large maps and land navigation is generally achieved by cartesian planar coordinates. Earth can be displayed on maps with large scale as a plane, which brings the advantage of the display as simple geometric shape and mathematics as a complex ball.

In general, the georeferencing process consists of the following steps:

- Identification of appropriate reference data. The data set is needed in the required coordinate system (preferably the same as a scanned map or digital image) which are used to match raster data (target data). Bitmap data and reference data have some common features that are visible in both data sets, such as street junctions, hydrographic functions or building contours.
- Selection of checkpoints (based on common features) to link known locations in both data sets.
- Transformation of target data into compliance with the reference given.

Well-designed maps, such as topographic maps, also consider the curvature of the Earth. Simple linear increments (i.e. meters) of planar coordinates are significantly easier for the large map using high accuracy in the field, as it is more complicated to increase the latitude and longitude (i.e., degrees).

V. THE EXPERIMENTAL SETUP

The practical setup and the workflow were inspired by experiments with hyperspectral UAV remote sensing [20].

A. Data Integration and Processing

The experimental UAV includes an irradiation sensor, single-board microcomputer Raspberry Pi 3 and a 4G modem (Fig. 5). The UAV communicates via a standard DJI Phantom 4 communication channel. The 4G modem is used to obtain real data from the RGB camera system. The UAV has a payload of 1kg and a flight time of 25 minutes.

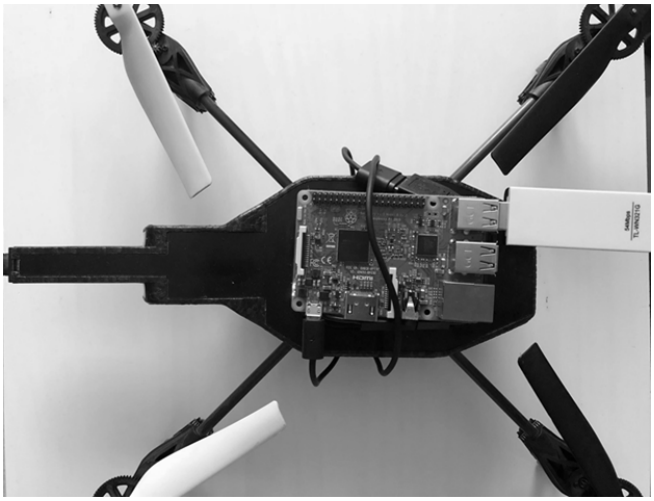

Fig. 5. UAV with Raspberry Pi and 4G modem

The camera image processing is divided into real-time processing and post-processing. In real time it is possible to get approximate sets of data with a lag of about a few seconds after capturing each image. The quality of data is achieved by subsequent processing. The main steps of real-time processing include Preliminary image processing, GNSS/IMU trajectory calculations and irradiation data.

B. Workflow After Downloading a Cloud Image

Estimation of real-time motion is based on a tune-up block of packages that use SIFT points. Raspberry Pi uses an integrated GPU and is responsible for calculating data points.

The SiftGPU library is used for the calculation. The resulting data is expressed in a local coordinate system based on the definition of an internal limitation. This co-ordination system is independent of the global GNSS and IMU co-ordinate system. All internal restrictions are then processed for weighted GNSS and IMU on the observation board upon availability.

The resulting sequential image is compared using the SIFT functions computed on a GPU in a multi-pyramid style system designed to find a large set of high-frequency attachment

points. Putting in and pairing a local block with multiple images improves the network estimate. The side overlays are calculated and considered comparing the original calculation versus the cloud calculation.

Repetitive BBA is implemented using the Jacobi matrix. The main emphasis on this implementation is placed on processing speed and low RAM load. Depending on the size of input images (8 - 22 MPix), extraction of key points and roughness, it requires at least 3-4 seconds for processing by a regular computer.

In case of the shot taken in a time of poor lighting, it will affect the processing time which may reach 18 seconds. This situation can only happen if the number of high-quality matches is extremely low (less than 39 robust matches). To determine the relative orientation is calculated <1s, these speeds can be accelerated by CPU parallelisation.

At the output, we get the point cloud representing the object. The structure and external orientation of the images in the local or global coordinate system is in a form. An incremental solution is the orientation of the image and the structure of the image-by-image object that allows the application to be used in real time. This real-time result is almost the same as a result achieved by optimisation calculations with a complete set of data. This method is a combination of both options [21].

After processing the image, GNS data is processed by PPK RTKLIB and processed in cloud computing. Then the cloud services are used to process partial droning images further.

To create an overall 3D map of the subject, the image can be processed by mapping the data from ultrasound sensors from the Phantom 4 UAV device. To measure the depth of the building, the above-mentioned ultrasonic sensors can be used all measured data and images can be processed at skycatch.com, which is a commercial cloud solution for final positioning and image orientation and cloud formation point. The result of processing is shown in Fig. 7

The higher resolution was used to capture the selected area - the better geometric accuracy can be achieved in the result. The camera calibration can be performed with an image orientation (self-calibration) and with a pre-calibration value used for the initial approximation value. Ground Control Points (GCP) transform data into the ETRS-TM35FIN Coordinate System. Geometric accuracy was evaluated using independent control points.

C. Testing in the City

The system was tested at the Technical University of Kosice, Slovakia (coordinates are 48° 43'56.1"N 21°14'39.5"E). UAV Phantom 4 with a built-in camera was used. For the main building, a flight altitude of 70 meters was set for a safe distance above the building. The UAV has been fixed to 2.4 m/s with fine swirls caused by a stronger wind that is typical for the area and season. For the given UAV type the flight time was 24 minutes. The time difference between adjacent exposures is 0.075 seconds. The image size is 1024 x 648 pixels with a pixel size of 11 μm. The field of view (FOV) is in the direction of flight ± 18 °, ± 27 ° in the crossing direction and ± 31 ° in the corner of the format. In addition to additional cameras, we also used the standard camera used by the UAV device.

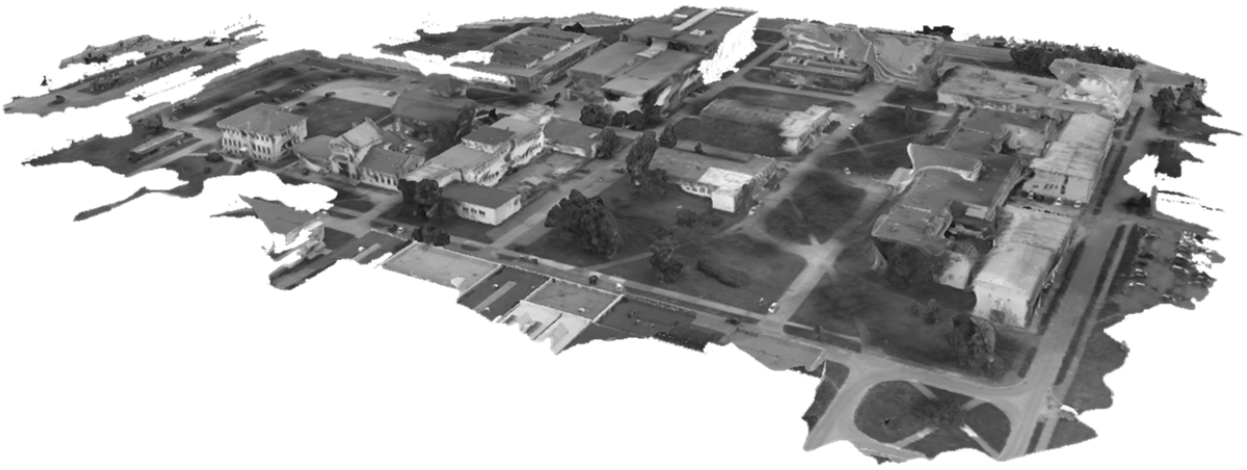

Fig. 6. 3D model Campus of University

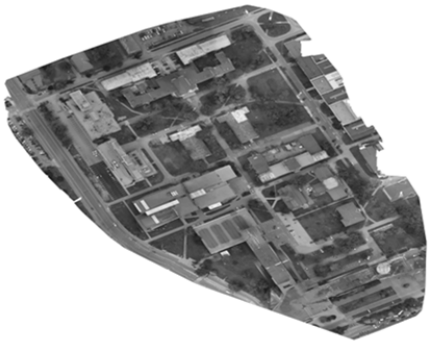

Fig. 7. 2D model Campus of University created from 165 frames

The UAV took a series of snapshots using the built-in camera and following the pre-programmed flight route. Manual control was not used to exclude a deviation from the route due to the operator's errors. For the real-time imaging, the UAV had aboard the Raspberry Pi microcomputer with the attached 4G USB dongle image transmission network, which was almost obscure assembly of the overall frame.

The obtained results were compared to post-production results from skycatch.com cloud computing services. Visual comparison and measured geometrical dimensions were basically identical. The result of experiments to use ultrasonic sensors and artificial intelligence processing from cloud services is shown in the Fig. 6. As it is seen, the quality of reconstructed 3D map of the area is quite reasonable.

VI. Conclusion

By summarising results of practical experiments, it is possible to conclude that both presented applications of UAV sensors, near object detection and building the 3D images of the scanned area, benefit from utilising the computing capacity and available resources of cloud services. Therefore, cloud services can be considered of high potential for a variety of UAV applications.

In the case of the object detection, the scan from lidar and camera video sensors is transferred to the cloud and further processed under the ROS using the NanoNets Machine Learning APIs. The NanoNets cloud service has proven to be very useful for detecting UAVs. In the second case, the scan from ultra-sonic and video sensors is transferred to the cloud and further processed using skycatch.com cloud services.

The status of work may still be considered as work in progress. More experiments with other applications and different cloud services are needed. Also, experiments with a different type of sensors and other hardware improvements are needed.

The next step will be to compare the results of the described experiments with results of performance the industry-grade UAV that has access to a novel real-time processing, capable of fixing impaired frames. The speed of real-time image processing and quality of results obtained from images taken in harsh conditions are to compare.

The other experiments are to be conducted by using a group or swarm of UAVs to capture the selected area, and a group of UAVs that will not use Lidars but just cameras for the object detection along with the automated central control system to process results from several UAVs.

Acknowledgement

This work was supported by the Cultural and Educational Grant Agency (KEGA) of the Ministry of Education, Science, Research and Sport of the Slovak Republic under the project No. 062TUKE-4/2017, the project No. 023TUKE-4/2017 and by the Slovak Research and Development Agency under the contract no. "APVV-17-0208 - Resilient mobile networks for content delivery".

References

[1] Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4 Percent in 2018. [Online]. Available: https://www.gartner.com/newsroom/id/3871416. [Accessed: 9-Sep-2018].

[2] P. Šuľaj, R. Haluška, Ľ. Ovseník, S. Marchevský, P. Pulli, and V. Kramar, "UAV Management System for the Smart City," *DISA 2018*, Aug. 2018.

[3] J.Blažek,"Systémy detekcie dronov z aspektov projektov VŠBM-Košická bezpečnostná revue, Košice: Vysoká škola bezpečnostného manažérstva v Košiciach, 2017, - Roč. 7 , č. 2 (2017), 8-20

[4] D. Solomitckii, M. Gapeyenko, V. Semkin, S. Andreev, and Y. Koucheryavy, "Technologies for Efficient Amateur Drone Detection in 5G Millimeter-Wave Cellular Infrastructure," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 43–50, 2018.

[5] C. O'Sullivan, N. Wise, and P.-P. Mathieu, "The Changing Landscape of Geospatial Information Markets," *Earth Observation Open Science and Innovation*, pp. 3–23, 2018.

[6] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized Search Over Encrypted Data With Efficient and Secure Updates in Mobile Clouds," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 97–109, 2018.

[7] Altavian, "WHAT IS CIR IMAGERY AND WHAT IS IT USED FOR?" [Online]. Available: https://www.altavian.com/knowledge-base/cir-imagery/. [Accessed: Aug-2016].

[8] M. Przybyla, "Detection and tracking of 2D geometric obstacles from LRF data," *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, 2017.

[9] P. Dong and Q. Chen, *LiDAR remote sensing and applications*. Boca Raton: CRC Press, 2018.

[10] L. Joseph, *ROS robotics projects: build a variety of awesome robots that can see, sense, move, and do a lot more using the powerful Robot Operating System*. Birmingham, UK: Packt, 2017.

[11] L. Joseph and J. Cacace, *Mastering ROS for robotics programming: design, build, and simulate complex robots using Robot Operating System*. Birmingham, UK: Packt Publishing, 2018.

[12] B. N. Bailey and W. F. Mahaffee, "Rapid measurement of the three-dimensional distribution of leaf orientation and the leaf angle probability density function using terrestrial LiDAR scanning," *Remote Sensing of Environment*, vol. 194, pp. 63–76, 2017.

[13] A. Harchowdhury, L. Kleeman, and L. Vachhani, "Coordinated Nodding of a Two-Dimensional Lidar for Dense Three-Dimensional Range Measurements," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4108–4115, 2018.

[14] R. P. Goebel, *ROS by example*. Raleigh, NC: Lulu.com, 2015.

[15] C. Fairchild and T. L. Harman, *ROS robotics by example: bring life to your robot using ROS robotic applications*. Birmingham, UK: Packt Publishing Limited, 2016.

[16] M. Ali, *Recent trends in applied artificial intelligence: 26th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013, Amsterdam, the Netherlands, June 17-21, 2013: proceedings*. Heidelberg: Springer, 2013.

[17] B. Ramsundar and R. B. Zadeh, *TensorFlow for deep learning: from linear regression to reinforcement learning*. Beijing: OReilly Media, 2018.

[18] Y. Zhang and Z. Yuan, "Cloud-based UAV data delivery over 4G network," *2017 Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 2017.

[19] "Industries," *NanoNets, Drone API*. [Online]. Available: https://nanonets.com/drone/. [Accessed: 23-Sep-2018].

[20] Alves de Oliveira, Raquel & Khoramshahi, Ehsan & Suomalainen, Juha & Hakala, Teemu & Viljanen, Niko & Honkavaara, Eija. (2018). Real-time and post-processed georeferencing for hyperspectral drone remote sensing.

[21] S. Jain, "How to easily Detect Objects with Deep Learning on Raspberry Pi," Medium, 20-Mar-2018. [Online]. Available: https://medium.com/nanonets/how-to-easily-detect-objects-with-deep-learning-on-raspberrypi-225f29635c74. [Accessed: 23-Sep-2018]