3D MAPPING WITH A DRONE EQUIPPED WITH A DEPTH CAMERA IN INDOOR ENVIRONMENT

Stanislav ALEXOVIČ, Milan LACKO, Ján BAČÍK
Department of Electrical Engineering and Mechatronics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, Tel. +421 55 602 4220,
E-mail: stanislav.alexovic@tuke.sk, milan.lacko@tuke.sk, jan.bacik.2@tuke.sk

ABSTRACT

This paper presents a quadrotor drone designed for the 3D reconstruction of indoor environments using a depth camera and a tracking camera. The drone's hardware and software components are described, including the Holybro X500V2 kit, the Intel RealSense T265 tracking camera, and the Intel RealSense D435i depth camera. The paper outlines the process of building a complete TF tree, capturing point clouds with RtabMap, and streaming video. The drone's pose stabilization and successful test flight in an indoor environment are also discussed, including its ability to navigate through indoor spaces and provide accurate data for 3D reconstruction. The paper concludes with recommendations for future development, such as autonomous navigation and exploration of unknown areas. The results demonstrate the feasibility and potential of this drone for creating detailed and comprehensive maps of indoor environments.

Keywords: *3D scanning, visual SLAM, drone*

1. INTRODUCTION

In today's technological era, digitisation has become an integral part of our lives. With the advancements in technology, capturing the physical world is now easier than ever before, offering endless opportunities to explore the space around us. However, traditional 2D photography is limited in the sense that it only captures a two-dimensional representation of a three-dimensional reality, losing out on valuable information. This is where 3D scanning comes into play, providing us with the ability to create digital copies of reality with 3D attributes. The field of 3D scanning has gained enormous popularity in recent years, offering new ways to interact with our environment.

Acquiring a high-quality 3D scan of an object or environment requires the use of various specialised sensors. The stereo vision and structured light methods, use 2D cameras and projectors, the Time-of-Flight principle (ToF) and LiDAR require more specialised sensors [1]. In the ToF, a camera is used to capture depth information. ToF cameras emit a light signal and measure the time it takes for the signal to bounce back to the camera, creating a depth map of the object or environment. They are commonly used in applications such as gesture recognition and augmented reality, where real-time depth information is crucial [2]. LiDAR, on the other hand, uses lasers to measure distance, generating a 3D point cloud of the object or environment, a method useful across various fields. It has gained popularity in recent years due to its ability to detect obstacles and navigate self-driving cars around them. This technology has been widely used in geology too, as it aids to generate detailed 3D models of terrain [3].

In addition to these sensors, other technologies are used in 3D scanning. For example, some 3D scanners use structured light sensors that emit light in the form of a grid pattern [4], while others use photogrammetry [5], which involves taking multiple photographs of an object or environment from different angles and using software to generate a 3D model. Each sensor has its strengths and weaknesses, and the choice of the sensor depends on the specific application and the requirements of its user.

Technology is advancing rapidly and there is an ever-growing need for high-precision 3D scanning solutions to capture complex environments with increasing accuracy. One of the most promising areas of development in this regard appear to be equipping drones with 3D scanning sensors to capture data in GPS-denied environments, such as indoor spaces and the like [6]. This technology has the potential to revolutionise a wide range of industries, from architecture and engineering to entertainment and education. By using drones to capture detailed 3D data in these environments, businesses and researchers can gain valuable insights into the physical world around them, enabling them to make more informed decisions and create more realistic simulations [7].

The field of drone technology is constantly evolving, and recent advancements in affordable components have opened up new possibilities for 3D mapping applications. This article focuses on a quadrotor drone that is built from such components, yet it is capable of high-quality 3D mapping.

2. 3D SENSORS

Sensors are fundamental to 3D scanning because their features affect the quality of scanned objects. The Intel Realsense D435i depth camera was used as the primary sensor in the project. It was designed primarily for robotics, and it has the appropriate features for the purpose of this research. The camera is small and lightweight and can be used indoors as well as outdoors. D435i's depth-sensing technology is based on active stereo vision, which is built from two infrared cameras, one RGB camera and an infrared projector casting a random pattern on scanned objects. An Inertial Measurement Unit (IMU), which can sense the acceleration and rotation of the camera, is also included among the features. This unit is used in Visual Simultaneous Localisation and Mapping (V-SLAM) algorithms. However, in this project, a separate component was selected to track its movements.

Table 1 Intel RealSense D435i specification

Specification	Value
Depth Field of View (FOW)	87° x 58°
Depth output resolution	Up to 1280 x 720
Depth frame rate	Up to 90 fps
RGB sensor FOV (H × V)	69° x 42°
RGB frame resolution	1920 x 1080
RGB frame rate	30 fps
Weight	75g
Dimensions	(90 x 25 x 25) mm

The setup includes a tracking camera from the same manufacturer as the first camera. The Intel RealSense T265 tracking camera can run Visual SLAM and outputs the exact location of itself in the scanned environment. It features a dual fisheye lens that has a combined 163° field-of-view and a low-power visual processing unit capable of running V-SLAM. It also integrates IMU, which is essential for V-SLAM. Although the depth camera can provide input for V-SLAM, a dedicated camera used for localisation is used instead. It provides greater accuracy and even lifts some load from the mission computer, which does not have to run CPU-intense algorithms.

Table 2 Intel RealSense T265 specification

Specification	Value
FOW	163° x 99°
Output resolution	Up to 1920 x 1080
Odometry output frequency	Up to 200 Hz
Weight	55g
Dimensions	(108 x 25 x 13) mm
Power consumption	1.5W

Both the depth and tracking cameras are mounted together to allow for their joint movements. Since the information about the position came from the tracking camera and scanned point cloud from the depth camera, a coordinate transformation system must be defined to align both inputs into one coordinate system. This was achieved by using the TF package from the ROS framework.

2.1. Complete TF tree

The TF tree is a directed acyclic graph (DAG) that represents the coordinate frames of a robotic system and the transformations between them. Each node in the tree represents a coordinate frame, and each edge represents the transformation between two frames. Nodes in the TF tree have a name and a parent node, which determine the frame of reference for that node. Additionally, each node has a transformation matrix that specifies the transformation between that node and its parent node.

ROS provides a range of libraries and tools for working with the TF tree, including a TF broadcaster and listener for publishing and subscribing to transformation updates and a TF viewer for visualising the tree. In order to use the TF tree effectively, all nodes must have a defined edge to their parent. In our case, all edges are defined with static transforms that are produced by the URDF model of a drone. The entire drone is treated as a rigid body object, and the only dynamic transformation relates to the drone's position on the map. This transformation is defined as a transformation between the *odom* (odometry) and *base_link* frame, with *base_link* representing the middle point of the drone. This transformation must be broadcasted with every change of position.

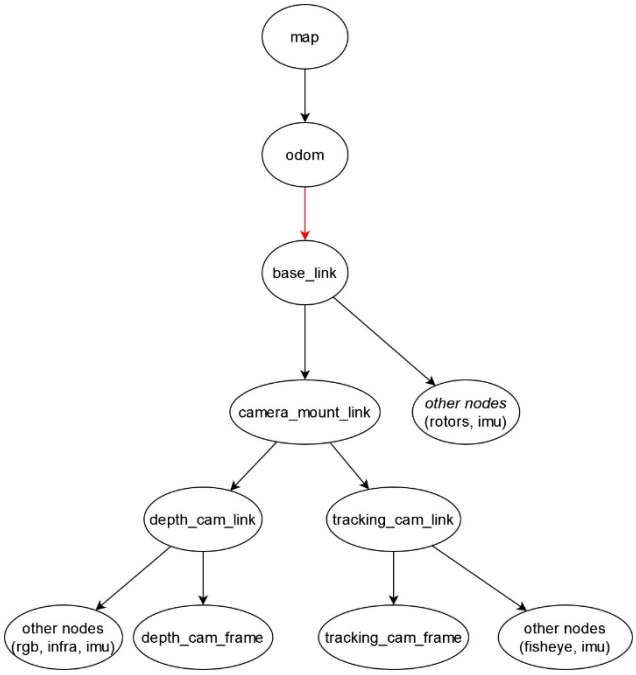


Fig. 1 Part of the TF tree showing relations among transform matrixes

However, the pose of the drone is obtained from the *tracking_cam_frame* rather than the *base_link* frame. Therefore, we need to recalculate the obtained pose relative to the *odom* frame. The transformation from *base_link* to the *tracking_camera_frame* is known, and we can use it to transform it to *base_link*.

To calculate this transformation, we need to recalculate the pose obtained from the *tracking_cam_frame* relative to the *odom* frame. This means we need to consider the static transform between the *base_link* and *tracking_cam_frame* and effectively cancel it from the obtained pose.

Let ${}^{o}_{b}T$ define the homogeneous transformation matrix that defines the transformation between the *odom* frame and the drone's *base_link*. Then it can be expressed as:

$$ {}^{o}_{b}T = {}^{b}_{t}T^{-1} \cdot P_t \cdot {}^{b}_{t}T \tag{1} $$

where ${}^{b}_{c}T$ represents the transformation matrix between the *base_link* and *tracking_cam_frame*, ${}^{b}_{t}T^{-1}$represents an inverted matrix from ${}^{b}_{t}T$, and P_t is a current pose represented as a transformation matrix relative to the *tracking_cam_frame*.

By applying the ${}^{b}_{t}T^{-1}$ matrix and then the ${}^{b}_{t}T$ matrix, we effectively cancel the transformation between the *base_link* and *tracking_cam_frame*, producing a pose as if it came from *base_link*. The resulting transformation can then be applied to the transformation between the *odom* and *base_link* frames to make TF tree complete.

2.2. Transforming points from the depth camera

Another transformation which needs to be applied is that of 3D points captured from a depth camera. The depth image produced by the depth camera is relative to the *depth_cam_frame* – the distance of points is measured directly from the camera. Nevertheless, our need is to have these points relative to *base_link*, which currently describes the pose of the drone according to the pose obtained from the tracking camera.

To recalculate the depth image, we again used the TF package for calculations and obtaining transformation matrices. We need to define a homogenous transformation matrix which defines the transformation between the *depth_cam_frame* and the drone's *base_link* denoted as ${}^{d}_{b}T$. It is defined as:

$$ {}^{d}_{b}T = \begin{bmatrix} {}^{d}_{b}R & \vec{t} \\ \vec{p} & s \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

where ${}^{d}_{b}R$ is the 3x3 rotation matrix that describes the orientation of the camera frame with respect to the *base_link* frame, $\vec{t}$ is the 3x1 translation vector that describes the position of the camera frame origin relative to the *base_link* frame origin, $\vec{p}$ is the 1x3 perspective vector, which is not used and therefore, its value is $\vec{0}$, and s is a scaling factor which is always equal to 1.

To transform a 3D point P_d captured from the depth camera frame to the *base_link* frame, we can apply the transformation matrix as follows:

$$ P_b = {}^{d}_{b}T \cdot P_d \quad (3) $$

where P_b is the transformed point in the *base_link* frame.

Aligning the tracking camera and depth camera into a common reference frame allows us to obtain a more accurate and complete representation of the environment around the drone. By aligning the two camera frames into the *base_link* frame, we can obtain a consistent and coherent view of the world, which is necessary for localisation and mapping.

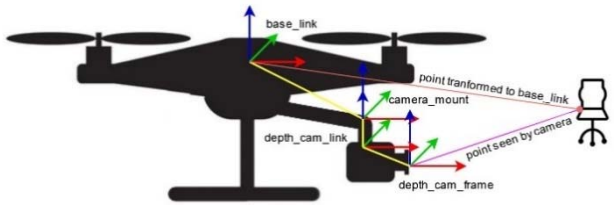


Fig. 2 Transforming the pose of the seen object to the base_link frame

2.3. 3D reconstruction from the depth image

The process of creating a 3D map involves the usage of depth images obtained from the depth camera. A depth image is, in essence, a two-dimensional matrix where each pixel corresponds to the distance from the camera to the object. To create a 3D map from these depth images, the depth data needs to be converted into a pointcloud. This is done with RTAB-Map software packages, which can read the depth image and compute the 3D coordinates of each point in the scene [8]. Once the pointcloud is created, the software can use it to generate a 3D reconstruction of the environment. In order to track the motion of the depth camera and align each pointcloud, odometry is taken from a tracking camera. This way, the software can accurately estimate the position and orientation of the depth camera for each captured depth image and combine multiple pointclouds into a single 3D map.

In addition to mapping, RTAB-Map also provides localisation capabilities. It uses odometry data from the drone's tracking camera to estimate its position and orientation within the map. This makes it possible for the drone to navigate through the environment and perform tasks based on its current location.

RTAB-Map also incorporates loop closure detection, which helps to correct errors that may have accumulated in the map over time. By identifying areas of the map that the drone had previously visited, RTAB-Map can refine its estimate of the drone's location and improve the accuracy of its map.

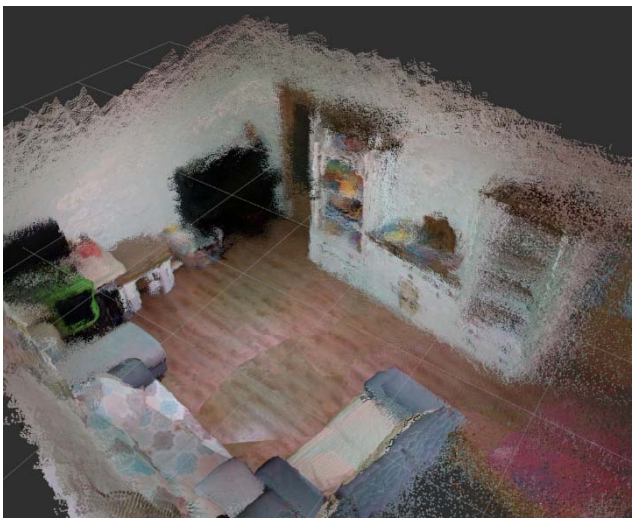


Fig. 3 Point cloud of a room created by RtabMap

3. DRONE

The drone used in this project was constructed from a Holybro X500V2 kit with a frame made of carbon fibre, a set of brushless motors, and Electronic Speed Controllers (ESCs). The drone was equipped with a Pixhawk 6x autopilot system, which operates using the open-source firmware PX4. The Pixhawk was chosen for its reliability and versatility, as well as its ability to integrate with a variety of sensors and components. The drone was also equipped with a 433MHz SIK telemetry radio for communication between the ground station and the drone.

In addition to the Pixhawk and telemetry radio, the drone was also equipped with a Futaba R617FS 7-channel receiver for remote control operation. The drone was powered by a 4S 4500mAh 14.8V LiPo battery, which was connected to the other components using a PM03D Power



Fig. 4 Fully assembled drone without protection cage

Module. The Power Module provided voltage and current monitoring, as well as protection against overvoltage and overcurrent.

To enable flight navigation, a Raspberry Pi 4B 8GB was used as the mission computer. The Pi was connected to the Pixhawk using an Ethernet cable and ran the Robot Operating System 2 (ROS2) for communication and processing output from the depth camera and tracking camera. The Pi also provided a Wi-Fi connection for communication with a ground station computer.

Table 3 Drone components specification

Specification	Value
Frame	Holybro X500V2
Motors	Holybro 2216 KV920
ESC	BLHeli S ESC 20A
Propellers	1045”
Battery	4S 4500mAh 14.8V LiPo
Telemetry radio	SiK Telemetry Radio V3 433MHz
RC	Futaba R617FS 7-channel receiver
Flight controller	Pixhawk 6x
Mission computer	Raspberry Pi 4B 8GB

Apart from the main components of the drone, a camera mount was attached to the frame of the drone to hold the two cameras. The first camera, a depth camera, was mounted on the front of the camera mount, facing forward. This camera was used to capture depth information of the environment, which was used for 3D mapping and reconstruction in later processing stages. The second camera, a tracking camera, was mounted on the bottom of the camera mount, facing down. This camera was used for visual odometry, estimating the position and orientation of the drone based on the visual features of the environment. The tracking camera worked in conjunction with the PX4 firmware and the RTAB-Map software to provide accurate localisation and mapping capabilities. With the combination of these components, the drone was able to fly autonomously, navigate and map the environment, and relay the collected data back to the ground station for further processing and analysis.

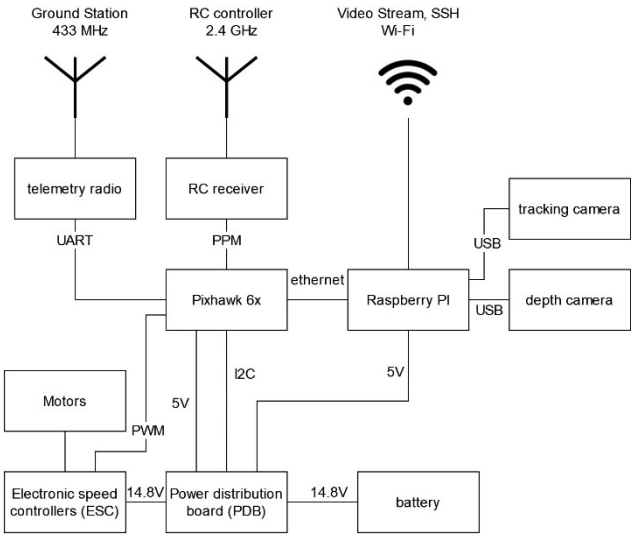


Fig. 5 Block scheme of all components used for a drone

To ensure the safety of both the drone and the environment in which it is flown, a protection cage was designed and attached to the drone. The cage was created using freeCAD software and was 3D printed using a durable material. Carbon fibre rods were used to connect the various parts of the cage together, resulting in a lightweight, yet strong structure. Then, motor base plate was 3D-printed. Its primary role was to attach the cage to the drone frame and replace the original one, as it allowed for the secure fastening of the cage. This protection cage was essential for the drone's operation in indoor environments, where collisions with walls or furniture could result in immediate damage to the drone's propellers, causing a crash and potentially damaging other components. By adding this protection cage, the drone was able to safely navigate indoor environments, providing increased protection for both the drone and its surroundings.

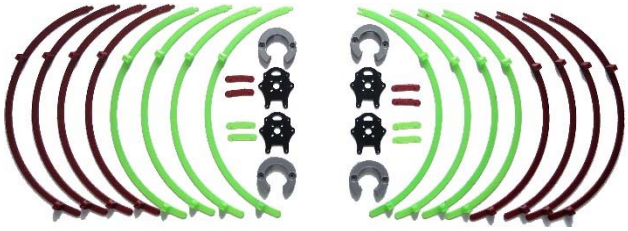


Fig. 6 3D printed parts of protection cage

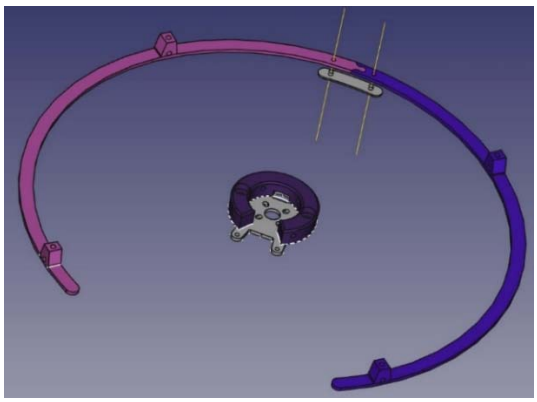


Fig. 7 Design of protection cage in FreeCAD

3.1. Video streaming

In drone piloting, the ability to stream video is an essential aspect. For the drone constructed in this project, the video stream consisted of three types of information: video from the RGB sensor, video from the depth sensor (the depth map), and metadata that included the timestamp and the current position in X, Y, and Z coordinates. However, streaming the generated map in real time was not possible due to the limited speed of the Wi-Fi connection, especially in areas with low signal coverage.

To generate the stream, GStreamer was used, which provided an RTSP server that ran in a separate thread in the ROS node responsible for processing the sensor data. The frames were read directly from the sensor using the librealsense SDK and then converted to OpenCV type for images. OpenCV was used to combine both frames together and embed metadata, and the final OpenCV frame was fed to the RTSP server, which handled the streaming to the user. The stream could be viewed directly in the ground station or in any video player capable of receiving a network stream, such as VLC.



Fig. 8 Video stream. up) image from RGB sensor, down) image from a depth sensor

Ensuring the stream's real-time nature was done by reducing the final video stream delay under 300ms. This was measured by re-capturing the then streamed video from the monitor, which produced an infinity mirror image with timestamp differences in each sub-image. The Raspberry Pi provided hardware acceleration for H264 encoding, which reduced the load on the main CPU. Streaming took only 7% of one core of the CPU, which demonstrated the system's efficiency.



Fig. 9 Measurement of video stream latency

3.2. Flight control and navigation

The drone was operated manually for the most part, even though it also offered autonomous flight capabilities through the use of a 2D map that was created beforehand. This was made possible by the combination of RtabMap's localisation functionality and Move Base ROS package's trajectory generation capabilities. Move Base is a powerful tool that utilises mapping, localisation, and planning algorithms to help the drone navigate to a predetermined global goal position. Prior to the autonomous flight, a map was drawn to provide Move Base with the necessary information about the environment. The tracking camera played a crucial role in determining the drone's position, which was used in conjunction with the mapping and planning algorithms to generate a safe and efficient path to the goal location while avoiding obstacles. Once the path was calculated, it was sent to the drone's control system, which utilised it to guide the drone to its destination while avoiding any obstacles along the way. The combination of RtabMap and Move Base offered a reliable and efficient way to achieve autonomous flight while ensuring the drone's safety in complex indoor environments.

4. POSE STABILIZATION

Despite the drone operating mainly indoors, external factors such as turbulence caused by the drone's own propellers can still affect the drone's position in a significant way. A stable pose is essential for precise 3D mapping, where drone stability can affect the resulting point cloud quality. Drone position stabilisation can be achieved by utilising a tracking camera and a PID controller located in the Pixhawk flight controller. PID controller is defined as the sum of the proportional component, integral component and derivative component.

$$u(t) = P(t) + I(t) + D(t) \tag{4}$$

Where $u(t)$ is the correction signal that is sent to the drone's motors at time t.

Each component can be defined as follows:

$$P(t) = K_p e(t) \tag{5}$$

Where $P(t)$ is the output of the proportional component at time t, K_p is the proportional gain, and $e(t)$ is the error at time t.

$$I(t) = K_i \int_0^t e(t)dt \tag{6}$$

Where $I(t)$ is the output of the integral component at time t, K_i is the integral gain, $e(t)$ is the error at time t, and the integral is taken from 0 to t.

$$D(t) = K_d \frac{de(t)}{dt} \tag{7}$$

Where $D(t)$ is the output of the derivative component at time t, K_d is the derivative gain, and $\frac{de(t)}{dt}$ is the rate of change of the error at time t.

The final equation will look like this:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{8}$$

For a 3D pose stabilisation, we need to define the vector for desired position $\vec{p}_{des} = [x_{des} \quad y_{des} \quad z_{des}]$ and the vector for the current position $\vec{p}_{cur} = [x_{cur} \quad y_{cur} \quad z_{cur}]$. The current position is provided by the tracking camera of the drone. This position is compared to the desired position, which produces error vector $\vec{e} = \vec{p}_{des} - \vec{p}_{cur}$. The error is then used by the PID controller to calculate an output vector that is used to adjust the drone's velocity.

To calculate the next correction value, we need to take into account the current error and time which elapsed since the last correction. This will produce the following equation:

$$\vec{u} = K_p \vec{e} + K_i I + K_d D \tag{9}$$

where *I* and *D* are the integrated and differentiated error terms.

The integrated error term is calculated as the sum of all the previous errors over time:

$$I = I + \vec{e}dt \tag{10}$$

where dt is the time elapsed since the last calculation. This helps to eliminate steady-state error in the system.

The differentiated error term is calculated as the rate of change of the error:

$$D = \frac{(\vec{e} - \vec{e}_{prev})}{dt} \tag{11}$$

where $\vec{e}_{prev}$ is the error from the previous calculation, this helps to account for sudden changes in the error term and prevent overshooting.

Finally, the output vector $\vec{u}$ is used to adjust the drone's velocity in the x, y and z directions:

$$\vec{u} = \vec{v} = [v_x \quad v_y \quad v_z] \tag{12}$$

Where v_x, v_y and v_z are the velocities in the x, y and z directions, respectively. The drone's position is then updated based on these velocities over a short time interval, and the PID controller is recalculated with the new error vector. By continuously repeating this process, the drone is able to maintain its position and adjust for any changes in the environment or external factors such as wind or turbulence.

5. RESULTS

During the test flight, the drone successfully navigated through the house and captured a complete 3D point cloud of the environment using tracking and depth cameras. Thanks to the drone's pose stabilisation, it was able to fly through narrow spaces like doors without colliding with obstacles. Additionally, the protection cage proved to be helpful, as it prevented damage from minor collisions with walls or doors. The video stream provided by the depth camera allowed for real-time monitoring of the drone's progress, ensuring that it remained on course throughout the flight. The final 3D point cloud was later downloaded from the drone, and the results showed that the system was successful in mapping the indoor environment. Further development can focus on exploring unknown environments with the system and improving the real-time streaming capabilities for more efficient data collection.

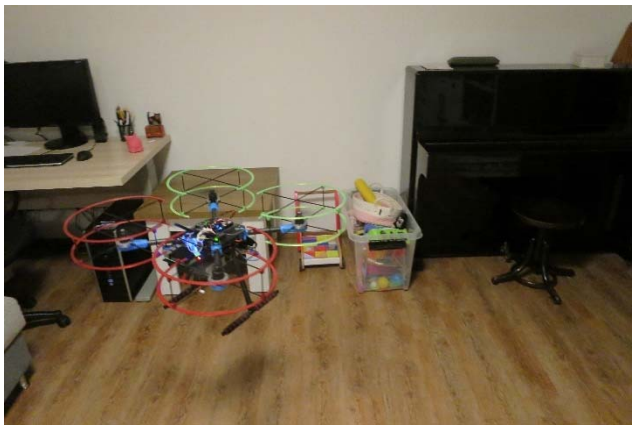


Fig. 10 Flying drone an inside indoor environment



Fig. 11 Point cloud of an indoor environment created by the flying drone

6. CONCLUSION

The focus of this paper is on the quadrotor drone designed for 3D reconstruction of indoor environments, using a combination of a depth camera for capturing depth information and a tracking camera for position sensing. The paper highlights the hardware and software components used in the development of this drone. While most of the components were readily available, integrating them proved to be a significant challenge. The integration required a combination of engineering expertise and software development skills, which were necessary to ensure the successful functioning of the drone.

A serious challenge was faced due to a bug in the Raspberry Pi USB port. This bug caused the tracking camera to stop sending pose frames, and with the camera manufacturer discontinuing support, a viable solution was not readily available. However, the authors were able to overcome this obstacle by developing custom code that interacted directly with the camera, replacing the RealSense-ROS wrapper. The custom code proved to be more efficient in consuming tracking pose frames and prevented internal queue overflow.

Created point cloud proved to be an accurate representation of the indoor environment, providing a comprehensive and detailed map of the scanned area. The results of the test flight demonstrated that the drone could navigate through narrow spaces and maintain its position, providing high-quality data for 3D reconstruction.

Future development of this system can focus on improving the autonomy of the drone, enabling it to navigate through complex environments without human intervention. Overall, this project has shown the promise of using quadrotor drones for indoor 3D scanning and the potential for future advancements in this field.

ACKNOWLEDGMENTS

This work was supported by Slovak Research and Development Agency under project APVV-18-0436.

REFERENCES

[1] LEE, C. – SONG, H. – CHOI, B. – HO, Y. S.: 3D scene capturing using stereoscopic cameras and a time-of-flight camera. in IEEE Transactions on Consumer Electronics, vol. 57, no. 3, pp. 1370-1376, August 2011, doi: 10.1109/TCE.2011.6018896.

[2] CHOI, O. – LEE, S.: Wide range stereo time-of-flight camera. 2012 19th IEEE International Conference on Image Processing, 2012, pp. 557-560, doi: 10.1109/ICIP.2012.6466920.

[3] BASTOS, D. – MONTEIRO, P. P. – OLIVEIRA, A. S. R. – DRUMMOND, M. V.: An Overview of Li-DAR Requirements and Techniques for Autonomous Driving: 2021 Telecoms Conference (ConfTELE), 2021, pp. 1-6, doi: 10.1109/ConfTELE50222.2021.9435580.

[4] ATIF, M. – LEE, S.: Adaptive Pattern Resolution for Structured Light 3D Camera System. 2018 IEEE SENSORS, 2018, pp. 1-4, doi: 10.1109/ICSENS.2018.8589640.

[5] BALTSAVIAS, E. P.: A comparison between photogrammetry and laser scanning. 1999 ISPRS Journal of photogrammetry and Remote Sensing, 54(2-3), 83-94, doi: 10.1016/S0924-2716(99)00014-3.

[6] MOTAWA, I. – KARDAKOU, A.: Unmanned aerial vehicles (UAVs) for inspection in construction and building industry. 2018 Paper presented at the 16th International Operation & Maintenance Conference, Cairo, Egypt.

[7] SONG, H. – YOO, W. S. – ZATAR, W.: Interactive Bridge Inspection Research using Drone. 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 2022, pp. 1002-1005, doi: 10.1109/COMPSAC54236.2022.00155.

[8] LABBÉ, M. - MICHAUD, F.: RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. in Journal of Field Robotics, vol. 36, no. 2, pp. 416–446, 2019. (Wiley)

Received March 20, 2023, accepted May 3, 2023

BIOGRAPHIES

Stanislav Alexovič received his M.Sc. degree from Faculty of Electrical Engineering and Informatics, Technical University of Košice in 2008. After twelve years as a professional software developer, he decided to pursuit his Ph.D. at the Technical University of Košice in the Department of Electrical Engineering and Mechatronics. His research interests include mobile robotic systems, autonomous exploration, SLAM algorithms and drones.

Milan Lacko received the M.S. and Ph.D. degrees in electrical engineering from the Technical University of Košice, Slovakia, in 2005 and 2010, respectively. He is currently, since 2019 an Associate professor in the Department of Electrical Engineering and Mechatronics, Technical University of Košice, where he is engaged in teaching and research. His field of interest includes microcontrollers in embedded systems, digital signal controllers, and design of power electronics converters.

Ján Bačík received the M.Sc. degree (Hons.) in electrical engineering and Ph.D. degree in mechatronics from the Technical University of Košice, Slovak Republic, in 2012 and 2016. He is currently an Assistant Professor with the Department of Electrical Engineering and Mechatronics, Technical University of Košice and Software Engineer, Phonoteo. His research interests are intensively focused on the development of mobile robotic systems, especially navigation, mapping and control of mobile robotic platforms, and drones.