CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS

Petr Ješke

Autonomous helicopter control by a mobile phone with android for precision agriculture

Department of Control Engineering

Thesis supervisor: **Dr. Martin Saska**

March 2018

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date ....

signature

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Ješke**	Jméno:	**Petr**	Osobní číslo:	**456893**
Fakulta/ústav:	**Fakulta elektrotechnická**				
Zadávající katedra/ústav:	**Katedra řídicí techniky**				
Studijní program:	**Kybernetika a robotika**				
Studijní obor:	**Systémy a řízení**				

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Řízení autonomní helikoptéry v úlohách přesného zemědělství mobilním telefonem se systémem Android.

Název bakalářské práce anglicky:

Autonomous helicopter control by a mobile phone with android for precision agriculture

Pokyny pro vypracování:

The goal of the thesis is to design, implement in Android and ROS (Robot Operating System), and experimentally verify in Gazebo simulator and real experiments an application to support control of an Unmanned Aerial Vehicles (UAVs) in environment monitoring for precise agriculture.
1. Design and implement an application and an interface that enable to control a UAV equipped with onboard computer with ROS by mobile phone with Android.
2. Design, implement and compare different algorithms of coverage to be able to capture a compact picture of the selected area.
3. Test algorithms of stitching of individual pictures with and without using data from UAV onboard sensors and state estimators. The user should be able to edit areas, where the environment monitoring has to be realized, to confirm/adapt obtained plans of the mission, to download a composed image of the area.4. Verify system functionalities in Gazebo and with a real platform equipped by a down-looking camera in outdoor conditions.
5. Prepare a review of UAVs possibilities in precision agriculture, e.g. based on [2-7].

Seznam doporučené literatury:

[1] T. Baca, P. Stepan and M. Saska. Autonomous Landing On A Moving Car With Unmanned Aerial Vehicle. In The European Conference on Mobile Robotics (ECMR), 2017.
[2] Lukas, V.; Novák, J.; Neudert, L.; Svobodova, I.; Rodriguez-Moreno, F.; Edrees, M.; Kren, J. "The Combination of Uav Survey and Landsat Imagery for Monitoring of Crop Vigor in Precision Agriculture ", ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLI-B8, 2016.
[3] P. Katsigiannis, L. Misopolinos, V. Liakopoulos, T. K. Alexandridis and G. Zalidis, "An autonomous multi-sensor UAV system for reduced-input precision agriculture applications," 2016 24th Mediterranean Conference on Control and Automation (MED), Athens, 2016.
[4] C. Zhang, J. Kovacs. The application of small unmanned aerial systems for precision agriculture: a review, Precision Agriculture, Volume 13, Issue 6, pp 693?712, 2012.
[5] L. Piermattei, at al. Multispectral data processing from unmanned aerial vehicles: application in precision agriculture using different sensors and platforms. EGU General Assembly, 2017.
[6] T. Ad?o, at al. Hyperspectral Imaging: A Review on UAV-Based Sensors, Data Processing and Applications for Agriculture and Forestry. Remote Sensing, 9(11): 1110-1125, 2017.
[7] E. Honkavaara, at al. Processing and Assessment of Spectrometric, Stereoscopic Imagery Collected Using a Lightweight UAV Spectral Camera for Precision Agriculture, Remote Sensing, 5(10): 5006-5039, 2013.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Saska, Dr. rer. nat., Multirobotické systémy FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **31.01.2018** Termín odevzdání bakalářské práce: _____________

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Martin Saska, Dr. rer. nat. podpis vedoucí(ho) práce	prof. Ing. Michael Šebek, DrSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání	Podpis studenta

Acknowledgements

Firstly I would like to thank my family for their support during my studies at the university. Furthermore, I thank my supervisor Martin Saska for his support and advice. Finally, I would like to thank Tomáš Báča, Vojtěch Spurný and others from the MRS group for their comments and advice.

Abstract

The goal of this thesis is to create a mobile application which serves to map agricultural areas using aerial photographs taken from a UAV. An analysis was made in order to examine the options available to create an application for communicating between the phone and a ROS environment, which is the meta operating system running on the UAV. The best-found option was implemented on an Android platform. The thesis covers the issues of planning trajectories for complete coverage, of taking photos from a flying object and of stitching images into a single image to create a map of the desired area. A series of experiments is presented. These show the performance of the system under various conditions, both in a simulation environment and with a real platform.

Keywords: precision agriculture, ROS, Android, path planning

Abstrakt

Cílem práce je vytvoření mobilní aplikace, která bude sloužit pro mapovaní zemědělských ploch formou leteckých snímků pořízených z bezpilotní vícerotorové helikoptéry. Byla provedena analýza možností, jak vytvořit aplikaci pro komunikaci mezi telefonem a prostředím ROS, které využívá koptéra. Nejvhodnější varianta byla implementována na platformě Android. Práce pokrývá problematiku plánování trajektorie pro kompletní pokrytí zadané oblasti a problematiku sběru snímků z letícího objektu a jejich následného skládání pro vytvoření mapy zadané oblasti. Byla uskutečněna série experimentů, které testují chování výsledné aplikace, jak v simulaci, tak s reálnou platformou.

Klíčová slova: přesné zemědělství, ROS, Android, plánovaní trajektorie

Contents

1 Introduction

An unmanned aerial vehicle (UAV) is a unified term for a flying robot with no pilot on board. It can be remote controlled (i.e., controlled by a human), it can fly autonomously based on a preprogrammed trajectory, or it can present even higher levels of autonomy (AI methods, response to the environment, etc.). In the past few years, the popularity of UAVs is increasing rapidly, especially the popularity of multi-rotor helicopters, also called MAVs (micro aerial vehicles - Figure 1), which have a lot of advantages compared with other UAVs.

- The MAV is easy to learn to pilot even for a non-professional pilot, unlike remote-controlled planes or helicopters.
- The user does not have to concentrate on piloting for the entire flight time. The MAV can retain its position, while the pilot can focus on other tasks such as planning a new trajectory, evaluating data from sensors, taking photos, etc.
- With a precise guidance system, multicopters are available to reach the desired position with a high level of accuracy.
- Due to vertical landing and take off sequence, landing areas can be relatively small, which permits using the MAVs in crowded areas or even indoors.

Figure 1: MAV - Parrot Bebop 2 [1].

UAVs can be used in many different applications for various purposes. A significant potential of UAVs is in mapping areas damaged by earthquakes or floods. In this case, UAVs can locate survivors, bring necessary medical equipment or locate houses which are likely to collapse. Defense forces use UAVs for exploring potentially dangerous areas, and first concepts of using UAVs as weapons are presented. MAVs can be used to deliver post or in the film industry for taking shots from the air. A large number of UAVs is intended for the general public for entertainment only, which brings developers to the task of creating simple controllers for everyday use, which are easy to understand and to work with.

1.1 Precision agriculture

Precision agriculture is one of the areas where UAVs could replace existing technologies.

Precision agriculture (PA) as defined in (V. Lukas at al., 2016 [2]) "*is a unified term for land management using modern technologies that began to be developed in the eighties and early nineties of the twentieth century. The aim of precision agriculture is an optimization of production inputs (fertilizers, pesticides, fuel, etc.) based on the local crop requirements and soil condition. Crop management in this way can lead to the economically efficient use of agrochemicals and minimization of environmental risks. Site-specific management takes into consideration spatial variability within fields and optimizes the production inputs, thus fulfilling the objectives of sustainable agriculture.*"

The most common approach to PA is high-resolution satellite imagery. Although it is a popular approach among agricultural companies, it is time-consuming, dependent on weather conditions and the cost is not negligible, all of which are motivations to find an alternative solution.

The answer could be UAVs. Low-cost of operation, the potential of a higher image resolution, the possibility of using multiple sensors (e.g., multispectral cameras, thermal cameras, etc.) and high flexibility are just some of the reasons why a lot of recent studies concentrate on the application of UAV imagery.

The goal of this thesis is to design, implement in Android and ROS (Robot Operating System), and experimentally verify in Gazebo simulator and real experiments an application to support control of UAVs in an environment monitoring for precise agriculture, which is motivated by the needs of Czech industry to apply UAVs to smart agriculture and water protection.

2 Related work

The idea of using UAVs for PA is not new. Therefore a lot of papers dealing with this problem can be found. A general overview of the problem is in (C. Zhang, J. Kovacs, 2012 [3]). In this paper, different unmanned vehicles are compared and limitations of using UAVs in PA are studied. Attention is also paid to image processing and information extraction from UAVs images. The task of collecting and processing the UAV imagery is also discussed in (K. C. Swain at al., 2010 [4]). This study examines possibilities of using unmanned helicopters to measure chlorophyll concentration in a rice crop. An interesting topic, independently discussed by several research groups, is the usage of UAVs in hyperspectral imaging for the purposes of PA (P. Katsigiannis, at al., 2016 [5]), (L. Piermattei, at al., 2017 [6]), (T. Adao, at al., 2017 [7]) or (E. Honkavaara, at al., 2013 [8]).

Relevant research to the topic of this thesis was conducted at Mendel University in Brno, well documented in (V. Lukas, J. Novák, 2016 [2]). This research aimed to compare unmanned and multispectral satellite imaging for estimation of basic crop parameters during the growing season and focused mainly on processing collected data and evaluating impacts of fertilizers and water supply on tested plants.

An important task related to PA is the issue known as complete coverage path planning (CCPP). CCPP is a problem of finding a path that passes through all the waypoints in the desired area from a starting point to a final point while avoiding obstacles. Waypoints can have many interpretations such as positions from which the robot performs measurements, places from which the MAV takes a photo or spots on the floor for a vacuum cleaning robot. The area of robotic applications dealing with CCPP is as wide as the number of waypoints interpretations: vacuum cleaning (Yasutomi et al., 1988 [9]), robot painting (Atkar et al., 2005 [10]), harvesting (Ollis and Stentz, 1996 [11]), ground mapping (Demim et al., 2016 [12]) etc. The problem of CCPP is described in detail in section 4.

As the altitude in which the MAV can fly is limited, mainly because of the resolution of the on board camera, it is necessary to deal with the problem of combining multiple photos into a single image in order to obtain a complete image of the desired area. This problem is known as photo stitching. A general overview of photo stitching methods can be found in (R. Szeliski, 2006 [13]). Currently used stitching algorithms are well documented in (Q. Fu, H. Wang, 2017, [14]) and (X. Zhou et al., 2016 [15]). The problem of UAV image stitching was discussed in (Ch. Cheng et al., 2017 [16]). In their paper, image stitching is established using Speed-up Robust Features (SURF) techniques. Finally, the approach used in this bachelor thesis is based on (Y. Ha, H. Kang, 2017, [17]). One of the advantages of their method is the easy integration of an image stitching algorithm into the mobile application. More information about image processing can be found in section 5.

2.1 A review of UAV possibilities in precision agriculture

Over the past decade a numerous researches studying possibilities of the UVAs in PA has been conducted using not only MAVs, but also other types of UAVs such as powered gliders (V. Lukas, J. Novák [2]), helicopters (K. C. Swain et al. [4]), powered parachutes, fixed-wing aircrafts (L. Piermattei, at al. [6]) etc. In the past, most of the published applications ended up only as research, due to a strict aviation regulations, low sensor reliability and high initial cost. This may change with the progress of UAV technology, falling prices of the components and a relaxing trend of aviation regulations. Today it is possible to create a cheap system providing a complete set of functions for PA application, which requires only a standard smartphone as a controller, as will be demonstrated in this thesis, which may increase the interest of agricultural companies in embedding UAVs to PA.

3 Problem specification

The following chapters will cover these points:

- Designing and implementing an application that enables the control of UAV equipped with an onboard computer with ROS by mobile phone with an Android operating system.
- Designing, implementing and comparing different algorithms of coverage to be able to capture a compact image of the selected area.
- Testing algorithms of stitching of individual pictures with and without the use of data from UAV onboard sensors and state estimators.
- Designing and implementing an application in which the user is able to edit areas, where the environment monitoring has to be realized, to confirm/adapt obtained plans of the mission, to download a composed image of the area.
- Verifying system functionalities in Gazebo and with a real platform equipped with a down-looking camera in outdoor conditions.
- Creating a review of UAV possibilities in precision agriculture.

The final application assumes using a smartphone with an Android operating system with API (Application Programming Interface) level 15 or higher in order to use all application components, such as Google Maps advanced features, SQL libraries, recent Android application structures, etc.

The concept of the presented ground mapping application is conceived as a “light solution”, meaning that the goal was to present a fully working application using minimal hardware equipment. To use all the application possibilities, MAV should be equipped with the following components (Figure 2).

- a computer running a ROS
- a GPS to obtain the position of the MAV
- a high resolution down looking camera
- an altimeter to measure the altitude of the MAV above the ground (e.g., Garmin rangefinder, LIDAR-lite laser rangefinder)
- a unit estimating the tilt of the MAV (e.g., Pixhawk Flight Controller with IMU-Inertial Measurement Unit)

The last assumption is the ability of the Android device and the MAV to communicate via WiFi connection on the same frequency (usually 2.4 or 5 GHz).

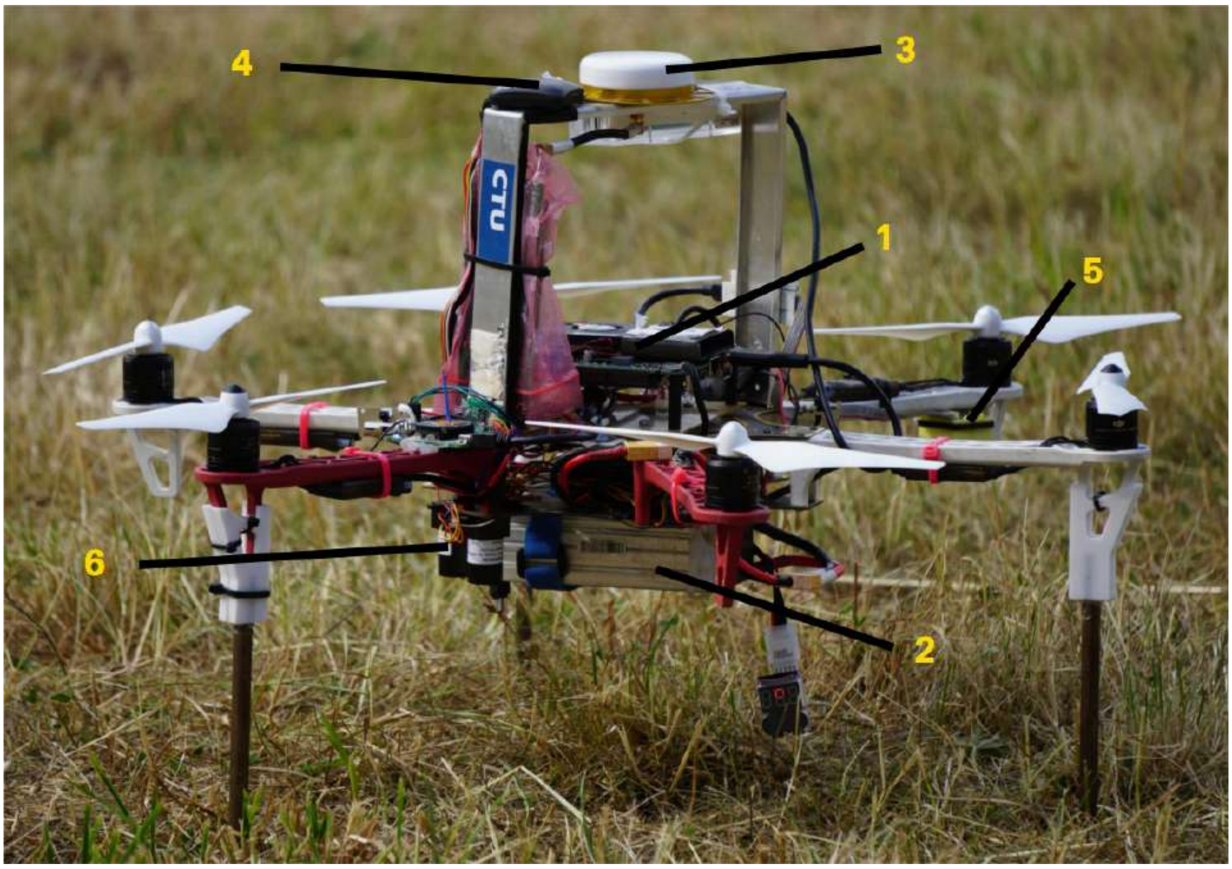


Figure 2: MAV used in real platform experiments equipped with an essential list of components. (1) computer running ROS, (2) battery, (3) differential GPS, (4) standard GPS (for the case of differential GPS malfunction), (5) rangefinder (6) Garmin rangefinder (for the case of ranger rangefinder malfunction) [18].

3.1 Application structure

The application can be divided into three parts: GUI (Graphical User Interface), Android application (application running on the smartphone), ROS application (application running on the MAV). In the GUI the user selects mapping areas, manages generated trajectories, observes the progress of the mission etc. The Android application reacts to the user input (plans trajectory, shows mission results, etc.) and sends appropriate commands to the ROS application. The ROS application directly controls the MAV (e.g., sends MAV to specified positions, controls the camera, etc.), collects data from onboard sensors and sends the data to the Android application. A graph of the application structure is shown in Figure 3.

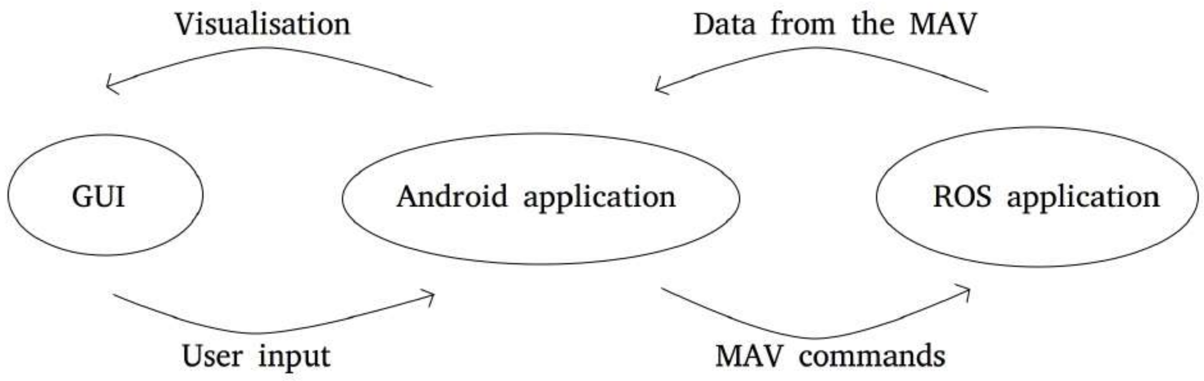


Figure 3: Graph of the application structure. The Android application provides a GUI, reacts on the user input, visualize application results and sends commands to MAV. The ROS application reacts on MAV commands, controls the MAV and sends data (GPS position, images) back to the Android application.

4 Complete coverage path algorithms

Because of the wide range of applications dealing with complete coverage, trajectory calculating algorithms became a widespread problem among the science community. Requirements on the trajectory are usually the same: Visit all waypoints in the selected area. Avoid obstacles which may occur. Find the shortest path. Due to this similarity, it is possible to apply the same approach on different robotic tasks. Algorithms solving the CCPP problem can be divided into two categories.

(a) Algorithms adapting trajectory based on data from onboard sensors. This approach is more complex and computationally demanding. The ability to detect obstacles goes with expensive sensors and software which is not always available. This approach can be used in applications such as mine or seabed mapping.

(b) Algorithms which rely on a consistent environment and advance map of the area. In other words, obstacles are not present in the mapping area, or their position is known at a time of planning the trajectory. This approach is simpler but fully satisfactory in many cases such as ground mapping applications from a flying object.

Most of the complete coverage algorithms used in PA applications counts with an assumption of a minimal altitude of 20 meters, usually more. Mapping larger areas from a lower height is considered inefficient. This assumption enables to neglect the possibility of collision with other objects, because, at such heights, obstacles are usually not present or their position is known, and the pilot can adapt the trajectory to avoid collisions. Based on this ascertainment two of the implemented algorithms (the Wavefront algorithm, the Zigzag algorithm) omit the possibility of an obstacle inside the mapping area. The third algorithm (the Trapezoidal algorithm) counts with a possibility of a forbidden area, and it is able to avoid it.

4.1 The Wavefront Algorithm

The Wavefront algorithm (sometimes called a distance transform algorithm) was first presented in (Zelinsky et al. 1993 [19]) and the implemented algorithm was based on their paper. The Wavefront algorithm uses the principle of cellular decomposition. The principle of cellular decomposition is to divide the mapped area into a grid of cells. Those cells are subsequently interpreted as nodes of an adjacency graph and boundaries between the cells as edges of the graph which transforms the problem of a CPPP into a problem of finding a path through the adjacency graph which is a well-known math task.

A cell represents a fundamental area in a given task (e.g., in grass cutting applications the fundamental area corresponds to the area which the robot is able to cut from its position, in ground mapping applications a fundamental area corresponds to the area which the UAV is able to capture in a single picture, etc.). The cells in the grid are considered neighboring if they have a common vertex (e.g., in Figure 4 cells neighboring with the G cell are all cells with a number one).

The area contains two significant cells. The cell marked as a Start cell (S) is a starting cell of the algorithm (usually a starting position of the robot). The cell marked as a Goal cell (G) is a cell, where the algorithm ends. The Goal cell can be any cell of the area including the S cell. The planning process starts with assigning a cost D to all the cells. In general, the cost corresponds to the distance of a cell from the Goal. The cost is generated as a propagation of a wavefront of values through all the cells in the area. The algorithm for assigning the cost D is as follows:

Algorithm 1 Assigning the cost D

```
1: procedure ASSIGN COST TO CELLS
2:     $parrentCells, childCells, goalCell$
3:     $Dcost = 0$
4:     for (cell c: allCells) c.cost = none
5:     goalCell.cost = Dcost
6:     $Dcost{+}{+}$
7:     $parrentCells = goalCell$
8: $loop$:
9:     if $allCellsReceivedCost()$ then break
10:    end if
11:    $childCells = getNeighbors(parrentCells)$
12:    $childCells = selectCellsWithNoValue(childCells)$.
13:    for (cell c : childCells) c.cost = Dcost
14:    $Dcost{+}{+}$
15:    $parrentCells = childCells$
16:    goto $loop$.
17: end procedure
```

Example of a cost propagation is shown in Figure 4. The assigned cost can be used for either calculating the shortest path between the S and G cell or calculating the complete coverage path.

As defined in (Zelinsky et al. 1993 [19]) "*the shortest path to the G cell from any point in the environment is obtained by following the steepest descent path.*" In other words, the neighboring cell of the last point in trajectory with the lowest cost is selected for the next point in the trajectory. If there is no unvisited adjacent cell, the algorithm returns to the previous cell. Otherwise, the algorithm returns the shortest path.

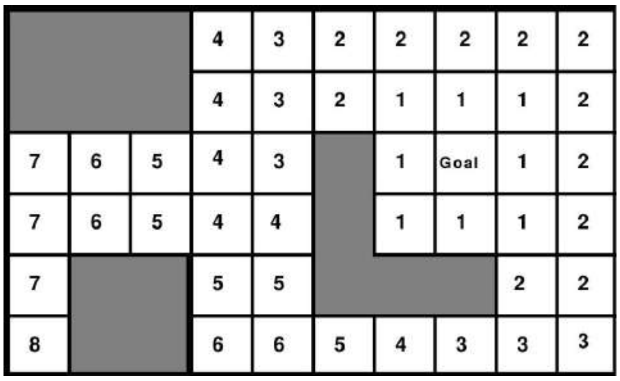


Figure 4: Assigning the cost D. In the first iteration, all cells neighboring with the Goal cell receive a cost one (one higher than the Goal cell). In the second iteration, all cells which are the neighbors of the cells with the cost one, with no assigned cost, obtain a cost two, etc. Source: [19].

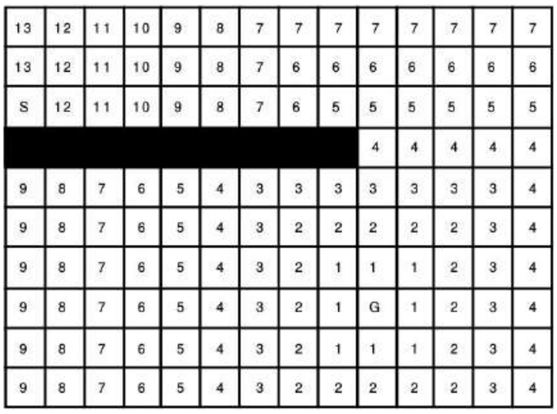


Figure 5: Cost of the cells based on the wavefront propagation started in a cell marked as G. Source: [19].

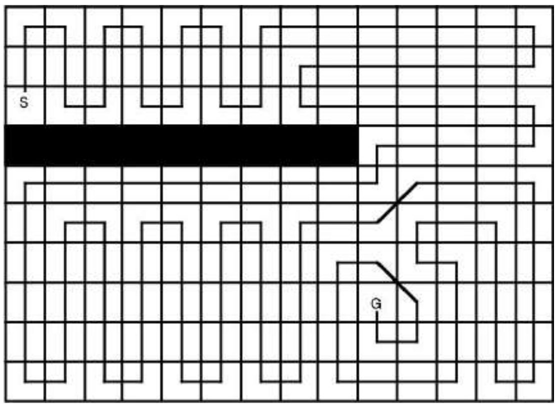


Figure 6: Calculating the complete coverage path from cell S to cell G. Source: [19].

As defined in (Zelinsky et al. 1993 [19]) "*calculating the complete coverage path begins at the Start cell. The neighbor with the highest cost is always selected for the next point of the trajectory. In other words, the robot moves away from the goal, keeping track of the cells it has visited. The robot only moves into a grid cell which is closer to the goal if it has visited all the neighboring cells which lie further away from the goal.*" This approach forces the robot to go through all the cells. The algorithm for complete coverage, whose results are shown in Figure 5, 6, is as follows

Algorithm 2 Calculating the complete coverage

```
procedure CALCULATE COMPLETE COVERAGE
    setAllCellsUnvisited()
    $currentCell = startCell$
$loop$:
    $neighbor = findUnvisitedNeigborWithHighestDcost(currentCell)$
    if neighbor equals null then
        goToPreviousCell()
    else
        markAsVisited()
        $currentCell = neighbor$
        if currentCell equals goalCell then return sucess
        end if
    end if
    if $allCellsVisited()$ then return failure
    end if
    goto $loop$.
end procedure
```

Embedding the Wavefront algorithm into the application

When the mapping area is selected (see selecting mapping area in user manual 7.2.5) it is split into a set of squares whose size matches the size of photos taken from the MAV flying in the desired altitude. Squares are inflated by a necessary photo overlap (photo overlap is dependent on a stitching algorithm, usually, at least 30 % of a photo size is required). Each photo is interpreted as one cell. Then the Wavefront algorithm is applied to the set, which generates the desired trajectory (Figure 7).

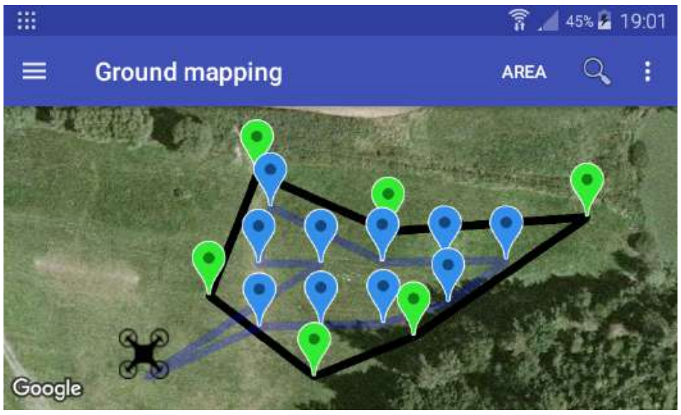


Figure 7: Example of the Wavefront algorithm in the application. Green markers represent boundaries of the area, blue markers points of the trajectory.

Advantages of the Wavefront algorithm

The biggest advantage of this approach is that when the MAV flies the trajectory, it stops at each waypoint and before the photo is taken, time for stabilization of the platform is left. This time can be directly set by the user, which means that they can influence the accuracy of the whole process. However, this algorithm is usually inappropriate for larger areas, because stopping at each point reduces the average speed and extends the flight time.

4.2 ZigZag algorithm

The most common approach to CCPP in ground mapping applications is an algorithm usually called the ZigZag algorithm, which covers the desired area using back and forth motions (Figure 8). This method does not consider the possibility of obstacles inside the mapping area and it is up to the navigator to choose the correct altitude to avoid collisions.

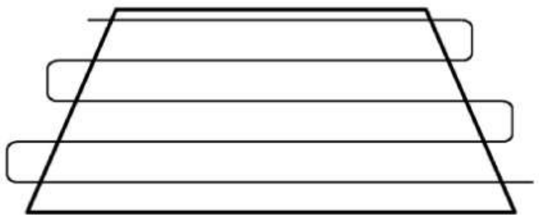


Figure 8: ZigZag principle, mapping using back and forth motions. Source: [20].

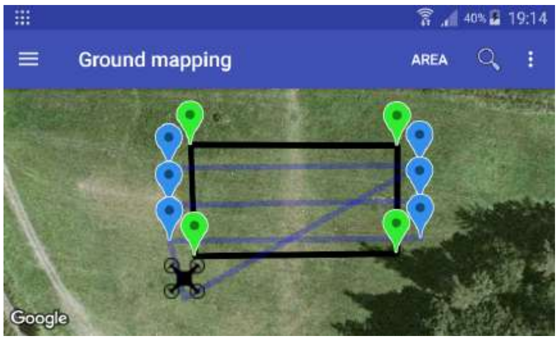


Figure 9: Example of the ZigZag algorithm in final application.

Embedding ZigZag algorithm into the application

The MAV flies in corridors (as shown in Figure 8) above the mapped area while taking photos periodically (with such frequency that the images have the necessary photo overlap). A corridor is a straight line above the mapped area. The width of the corridor corresponds to the width of the area, which the MAV is able to capture in a single picture while flying at the desired altitude, where the width of the corridors is subsequently reduced by necessary picture overlaps. An example of visualization of the results of the ZigZag algorithm in the application is in Figure 9.

Advantages of the ZigZag algorithm

The biggest advantage and also the reason why the ZigZag algorithm is used in most PA applications is that it eliminates disturbances on aerial photography caused by a change of tilt while the MAV accelerates/decelerates or changes the flight direction. The ZigZag algorithm removes those effects by moving changes of the velocity and the flight direction outside the mapping area. This means that the MAV accelerates to the mapping velocity outside the area, flies through the area at a constant speed (i.e., with a constant tilt), then decelerates, moves to the next transition and accelerates back to the mapping velocity before reaching the border of the area. A second advantage is that calculating the ZigZag trajectory is fast even with low mobile processor computational capacity.

4.3 Trapezoidal algorithm

Although most PA applications do not consider the possibility of a forbidden area inside the mapping region, in some cases it is a necessary assumption (mapping area can contain a no-fly zone, or contain high voltage pylons, or other obstacles which need to be avoided). For these cases, the application offers an algorithm, which is able to avoid forbidden areas. This algorithm is based on trapezoidal decomposition, therefore it is called the Trapezoidal algorithm.

The idea of the Trapezoidal algorithm is to decompose a complex polygon (representing the mapping area) into triangles and trapezoids, which are easier to cover using back and forth motions. Moreover, if the correct way of a transition between the simpler polygons is respected, the final trajectory avoids obstacles inside the area as will be shown in the following paragraph.

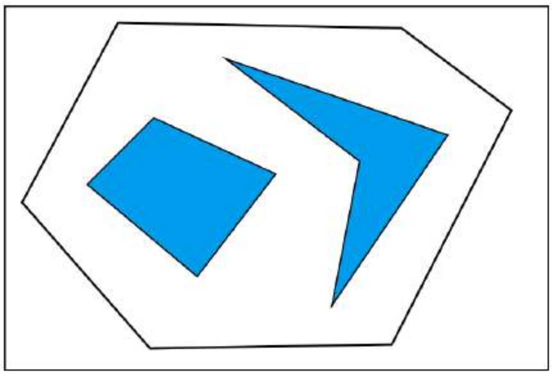


Figure 10: Polygonal environment, external polygon represents the mapping area, blue polygons represent obstacles. Source: [21].

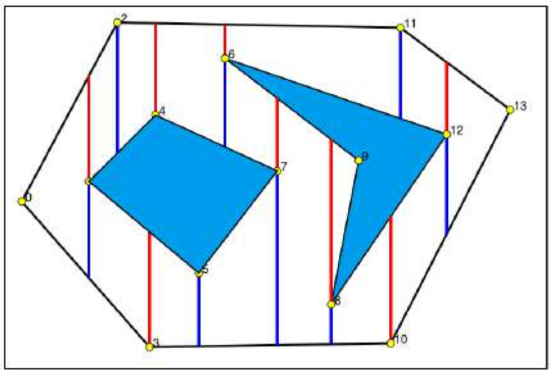


Figure 11: Example of a division of the mapped area to the set of simple polygons using trapezoidal decomposition. Source: [21].

Consider a non-intersecting polygonal area and polygonal obstacles as shown in Figure 10. Trapezoidal decomposition is achieved by creating a set of parallel lines, one line at each of the area and obstacle vertexes. The presented algorithm uses a north-south (i.e., parallel to the y-axis of the plane of the polygon) orientation of the lines, but the orientation can be arbitrary. After the lines are created the area is composed of simple polygons only, as can be seen in Figure 11. Each of the simple polygons can now be interpreted as a node of an adjacency graph and the edges of the graph are common sides of trapezoids. The process of interpreting the decomposed polygon as the adjacency graph is indicated in Figures 12, 13.

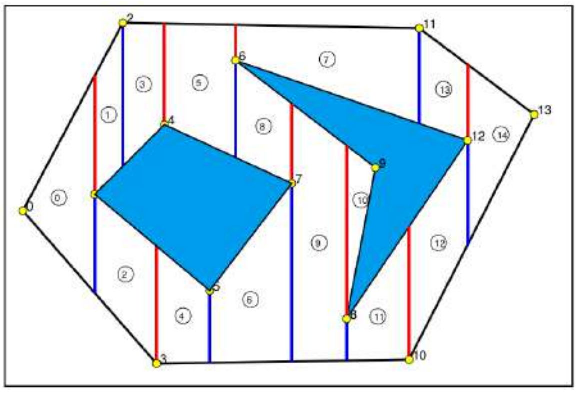


Figure 12: Interpretation of simple polygons received by trapezoidal decomposition as vertexes of an adjacency graph. Source: [21].

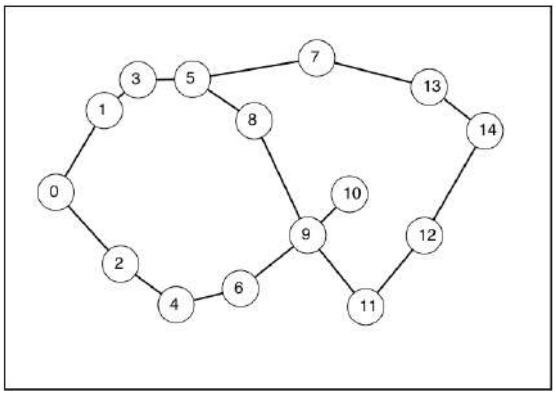


Figure 13: Adjacency graph corresponding to the result of trapezoidal decomposition shown in Figure 12. Source: [21].

The next step is finding a path through the adjacency graph. Based on the specific application, it is possible to search for the shortest path through the graph, using algorithms such as Deep-first search (DFS), Breadth-first search (BFS), A* or D*, or for a complete coverage path, which is the case of the typical traveling salesman problem (TSP). TSP is an NP (non-deterministic polynomial-time) hard problem, which means that there is no universal algorithm which would guarantee finding a solution in polynomial time for this task. Although for a small number of nodes, TSP can be solved using iteratively improving algorithms or repeatedly triggered randomize algorithm. Such algorithms can find an ideal solution, or a solution which is satisfactorily close to the ideal. The size of the mapped area (i.e., number of possibly created trapezoids and a number of nodes in the graph) is limited by the flight time of the MAV, which enables using one of the simple solvers for TSP. Based on this prediction, the Trapezoidal algorithm uses a randomized DFS algorithm to calculate the trajectory through the adjacency graph. Randomize DFS starts in one node of the graph, expands its successors, then chooses one successor at random which has not been expanded yet, and goes to the next iteration. If there are no unvisited successors, the algorithm goes back using backtracking. The algorithm is repeatedly triggered, and the shortest solution is selected. Using 100 iterations for tested areas consisting of more than 20 points, with multiple obstacles inside the area, the algorithm always found the optimal solution. Even for 1000 iterations the total computation time on the mobile processor is shorter than 1 second, which shows that the algorithm is satisfactorily fast to be used in a real application.

Once the path through the graph is found, the last step is to go along the path and map each unmapped polygon using a simple mapping algorithm such as the Zigzag algorithm, with the modification that the trajectory does not exceed boundaries of the polygon (as shown in Figure 16). It is important to note that the transition between the two polygons has to go through their common edge (Figure 14) otherwise it can result in the path coinciding with obstacles (Figure 15). A result of the Trapezoidal algorithm is shown in Figure 16.

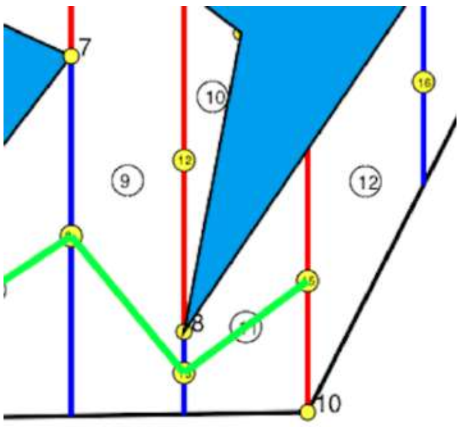


Figure 14: Transition between two polygons while avoiding the obstacle. Source: [21].

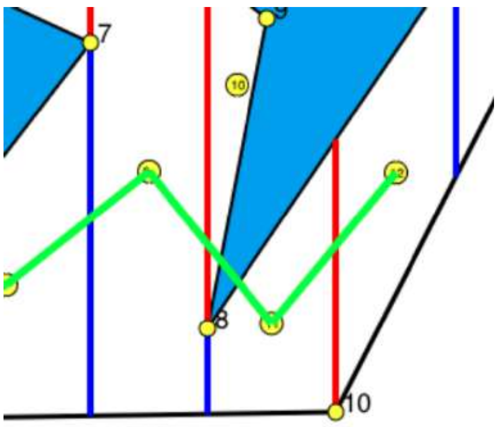


Figure 15: Collision with an obstacle due to wrong transition approach. Source: [21].

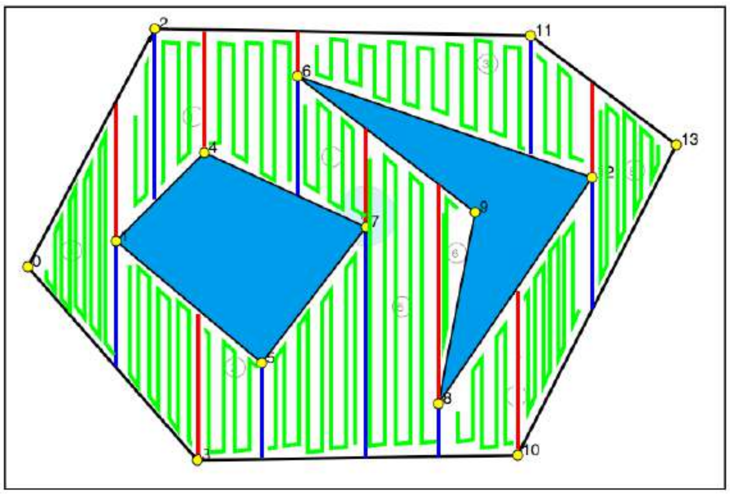


Figure 16: Demonstration of the Trapezoidal algorithm. Source: [21].

Trapezoidal algorithm in the application

In the following Figures, the Trapezoidal algorithm applied on a given area can be seen. The MAV flies above the mapped area while taking photos periodically, as in case of the ZigZag algorithm. Figure 17 shows an area to be mapped (green markers mark its corners) with a forbidden subarea (red markers). Figure 18 shows a trajectory generated using the Trapezoidal algorithm. Notice that no traversion crosses the forbidden area.

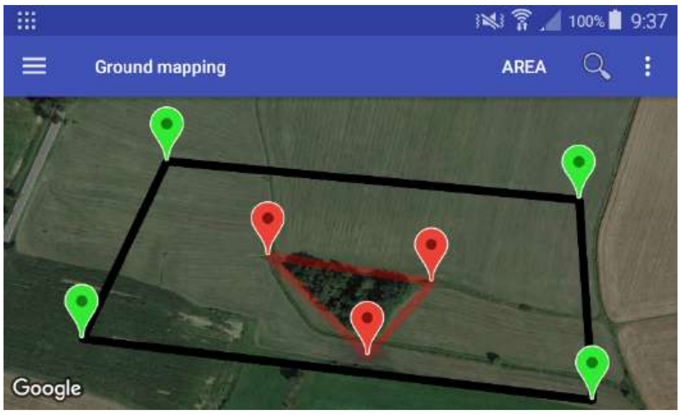


Figure 17: Selecting mapping area in the application.

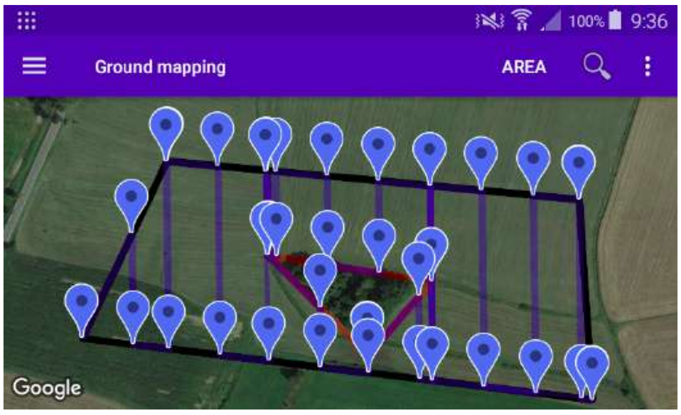


Figure 18: Generated trapezoidal trajectory represented by blue markers.

The biggest advantage of the Trapezoidal algorithm is the ability to avoid the forbidden area. Also, the Trapezoidal algorithm can be easily modified to find the shortest trajectory between two points of the area, which guarantees avoiding obstacles inside the area.

5 Aerial photography

This chapter describes the process of taking a compound picture of the mapped area from the MAV. At first the undesirable impacts of photography from flying object are presented, together with possible solutions followed by real platform distortions and their compensations.

5.0.1 Coordinate system

The tilt of the MAV in space is determined by a set of three parameters, roll, pitch and yaw as shown in Figure 19. The camera is set to the center of the coordinate system. The x and y-axes are parallel to booms of the quadcopter. The z-axis is perpendicular to both of them and points downwards. Arrows symbolize a positive direction of the coordinates.

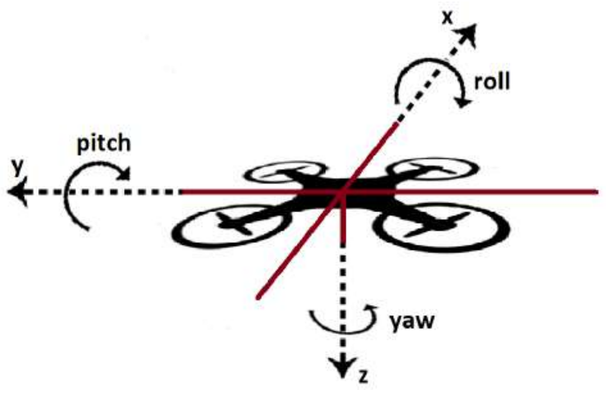


Figure 19: MAV coordinate system.

5.1 Image stitching

Three ways of stitching are used in the presented application. Stitching based only on a sensor data (estimated MAV position, roll, pitch, and yaw while taking the photo). On-board stitching using an OpenCV library. Stitching using a professional stitching software Autopano.

5.1.1 Sensor based stitching

The method of sensor based stitching is suitable for an initial preview of the mapped area in a system with limited computational power. The advanced image processing methods can be subsequently applied on the set of collected photos on a computer with higher computational power. In the process of the exact image stitching based on a sensor data, several distortions may occur. The following part studies their impacts and suggests possible solutions. As the application assumes usage of additional stitching software, the trajectories are calculated concerning an average demand on a photo overlap of the stitched photos. Most of the stitching algorithms require at least 30% photo overlap. Trajectories in this application are calculated in a way that taken images have ideally 40% overlap. The 10% of the photo overlap is a precision which is necessary adhere to in order to obtain a full map of the desired area without blind spots.

The precision of the image placement to the grid of photos depends on the accuracy of the GPS sensor used. A standard GPS achieves accuracy of 4-5 meters, which would require an increase of the average photo overlap to fulfill the overlap demands in every case. This effect can be limited to several centimeters using a differencial GPS, which is enough concerning the range of the desired photo overlap.

Without automatic camera stabilization, the image evaluating program had to deal with effects caused by changing tilt of the MAV in space. In the ideal case, roll and pitch of the MAV are zero (i.e., x and y-axes are parallel to the ground, see Figure 20).

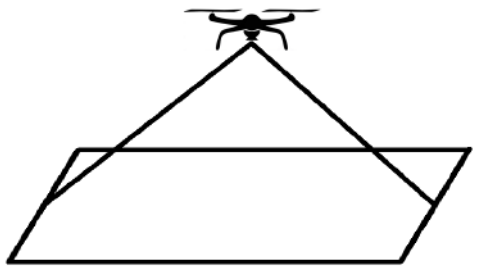


Figure 20: Taking a photo from the MAV in an ideal case, when roll and pitch are exactly zero.

In a real platform, the tilt of the MAV changes according to the current flight requirements. Using the roll and pitch, the MAV changes the flight velocity and the flight direction. Tilt also adjusts according to current wind conditions to preserve desired flight velocity. The nonzero roll causes the change of the image content (Figure 21) together with a change of the image perspective.

The effect of the changed perspective is negligible with respect to a high altitude (20m and more) used while mapping areas, together with a flat surface of the agricultural areas (i.e., surface without obstacles where the change of the perspective could appear). With the knowledge of the MAVs roll α and pitch β the dislocation of the photo can be calculated as follows

$$\Delta x = h \tan \alpha \tag{1}$$

$$\Delta y = h \tan \beta \tag{2}$$

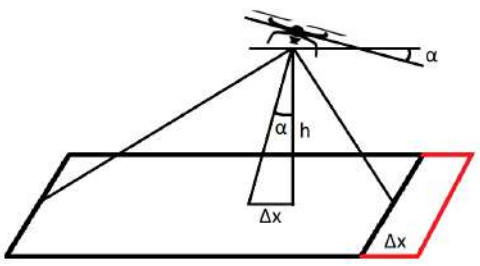


Figure 21: Influence of the nonzero roll on a taken photo. Dislocation of the photo (red) depends on the roll (α) and the altitude (h).

The last parameter which requires compensation is the yaw. In order to stitch the photos correctly, it is necessary to take the photos with a constant yaw. Inaccuracy in the yaw could cause rotation distortion, as shown in Figure 22. The influence of inaccurate yaw is compensated by additional rotation of the photo by an angle δ, which is a deviation from the desired yaw. Examples of sensor based stitching are presented in chapter 8.

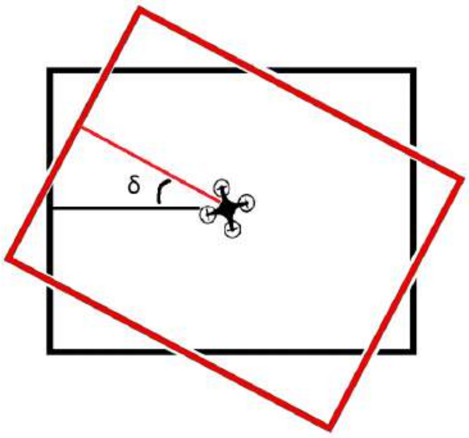


Figure 22: Influence of rotation distortion on the taken photo.

5.1.2 Stitching using the OpenCV library

The OpenCV library [22] is a free C++ image processing library. It was designed with respect to computational efficiency and with a strong focus on real-time applications, which are the reasons why it is abundantly used in many robotic applications. Besides the simple functions (image composition, image rotation, etc.) the OpenCV library offers a set of tools for advanced image processing (image transformation, filter design, image stitching, etc.).

For the purpose of image stitching, the OpenCV library offers a stitching API provided by cv::Stitcher. The Stitcher class operates with two camera models. A Homography model expecting perspective transformations between images is useful for creating photo panoramas and an Affine model for stitching images with a constant perspective suitable for composing images from aerial photography. The result of the OpenCV stitching process is presented in chapter 8.

5.1.3 Stitching using the Autopano software

The Autopano application [23] was selected as an example of advanced photo stitching. The Autopano software, with results shown in chapter 8, was selected based on the following features:

- Possibility of stitching up to thousand images
- Automatic color and exposition correction
- Multiple different projections
- Large number of input/output formats
- Possibility of showing individual pictures inside the stitched map
- Compatibility with Linux OS, Windows OS and Mac OS

5.2 Real platform distortions

Three distortions occurred while testing the application with a real platform. Tests showed that the used Mobius C2 camera has a small unconstant delay 0.2-0.3s based on current camera load, which caused an image dislocation. In order to compensate the delay, MAVs tilt and position is saved 0.25s before the photo, which minimalized the inaccuracy.

The second additional distortion is also caused by the camera. The wide angle of the Mobius camera causes an image distortion known as the fisheye effect (Figure 23), which was not present in the simulated camera. A solution to this distortion is to apply the rectification process on the photo, which changes the perspective of the image in order to obtain planar photography. For the process of image rectification, tools from the OpenCV library were used. The accuracy of the rectification process is dependent on the accuracy of the camera model which is used as a reference when correcting the real camera distortion. Real experiments showed that the Mobius camera model is not ideal. It compensates distortion on edges of the photo well (Figures 23, 24), but the resulting image gets compressed in a direction of the wider FOV of the camera (in the x-axis) as can be seen, when the rectified image (Figure 24) is compared with an image (Figure 25) taken from another MAV (DJI Phantom), whose camera generates a minimal fisheye effect, flying 100m above the mapped area. After the rectification process is finished, only the center of the image, which is enlarged in order to compensate the compression, is used in the map. An experimentally measured coefficient for image expansion in the x-axis is 1.5. The issue of the fisheye effect suggests using another camera, generating a lower fisheye effect, for the final product.

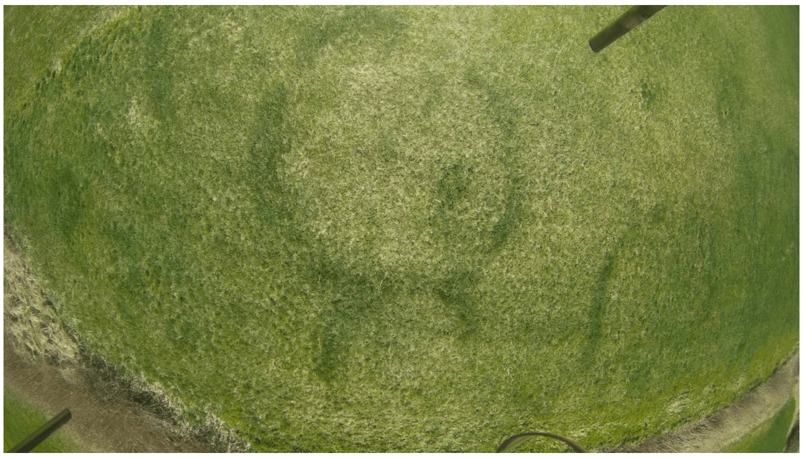


Figure 23: Example of a camera image without a rectification process. Notice the fisheye effect on the edges of the photo.

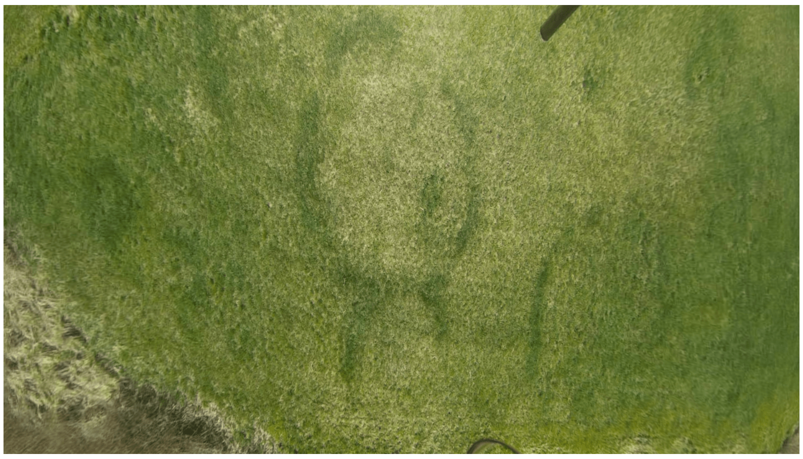


Figure 24: Example of a camera image with additional rectification, using the OpenCV model of the Mobius camera. Notice the reduction of the fisheye effect together with the change of the perspective mainly in the center of the photo (i.e., deformation of the circular shape on the ground).

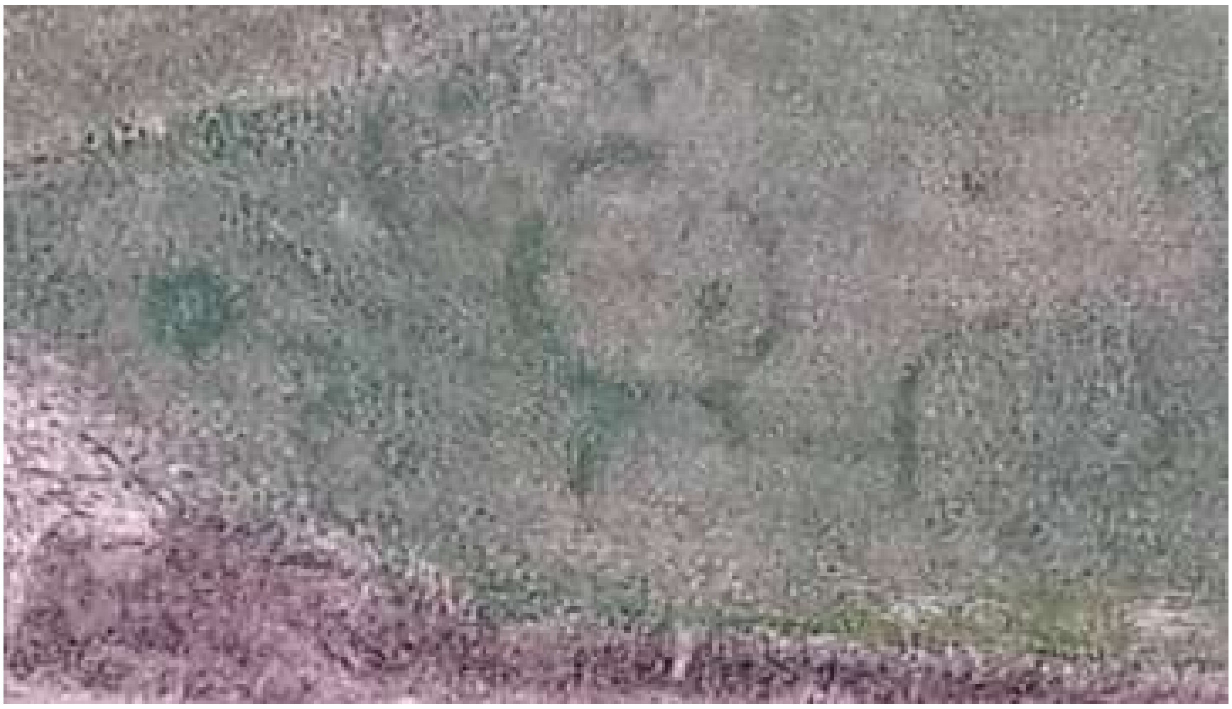


Figure 25: An image taken from 100m using a DJI Phantom, which shows the actual proportion of the circular shape on the ground.

The last distortion was caused by inaccurate knowledge of the MAV yaw. For evaluating the yaw, a Pixhawk flight controller is used. In the process of changing yaw, MAV uses a sliding change of the yaw to the desired value, which (depending on the size of the yaw change) takes several seconds. Within this time, IMU (inertial measurement unit) of the pixhawk controller already returns the final value of yaw. Due to this inaccuracy, some of the photos taken with the wrong yaw cannot be properly rotated to the correct position (the information about the yaw is incorrect). A possible solution for this problem can be the usage of another yaw sensor, for example a pair of two differential GPS sensors (where the yaw would be calculated from a difference of sensors position). Without additional sensors, the problem of inaccurate yaw was partially compensated by adding extra time, before each mapping sequence, left for the MAV to complete the rotation process.

6 Application description

The content of this chapter assumes a basic knowledge of the ROS and Android operating systems. If the reader is not acquainted with a ROS operating system and the Android operating system, it is suggested to read (M. Quigley, B. Gerkey, W.D. Smart, 2015 [24]) for an introduction to ROS and (S. James, 2013 [25]) for an introduction to Android development.

6.1 Wireless communication methods

Several different approaches can be used to establish communications between the MAV and the mobile device:

- The first option is a Bluetooth technique. The advantage of this approach is that most of modern mobile devices support the Bluetooth connection. However, parameters of this connection vary according to the Bluetooth performance class which they use. Class 1 has a range of approximately 100m with a transmission speed of 24 Mbit/s. Class 2 also reaches 24 Mbit/s, but the range is limited to 10m, which is too short concerning the manual control requirements. While most mobile devices are equipped with the Bluetooth of class 2 this communication method is rather theoretical.
- The second option is controlling using radio control. This method is widely used in commercially sold UAVs (DJI, power vision, ...). With a transmission frequency of 900 MHz, the range could reach up to 7000m. The disadvantage of this approach is the requirement of a remote controller, which enables the communication, in addition to a mobile device, which is against the requirement for a light solution to the problem.
- The third option is a Wi-Fi connection. It is a compromise between the Bluetooth technique and the radio technique. The biggest advantage of this option is the same as the advantage of a Bluetooth connection, almost all modern mobile devices support a Wi-Fi connection. Standard 2.4 GHz band reaches up to 46m indoors and 92m outdoors, which is satisfactory for manual control. While mapping larger areas, the trajectory can be loaded while the connection is established and the MAV flies the trajectory autonomously, when the mapping is finished the MAV returns to the starting point and reestablishes the connection.

Based on the presented comparison, the Wi-Fi connection was selected as the best option, and it is used as a communication method between the MAV and the mobile device.

6.2 Establishing communication through Wi-Fi

Two options can be used to establish the communication between the ROS operating system and the application running on the Android device.

6.2.1 Terminal

The first option is based on the possibility of connecting to a terminal of the MAVs computer directly from the Android application using an SSH (Secure Shell) communication protocol. When the connection is established, it is possible to call ROS services and send ROS messages from a command line without creating ROS publishers and subscribers. The advantage of this approach is that no program evaluating incoming messages from the phone is necessary, while the messages are sent directly on a specific topic. A disadvantage of thus created communication is that it is only unidirectional (the SSH communication is bidirectional, but most mobile devices do not support a command line control of the applications, which makes sending messages from UAV to phone problematic). However, more complex applications require a bidirectional communication in order to obtain the data from the UAV such as GPS position, battery state, camera images, etc., which limits the range of applications where this approach can be used and makes this approach inappropriate for a ground mapping application.

6.2.2 RosJava

The second option is based on an extension of ROS in a Java language, called Rosjava (D. Kohler, K. Conley, 2011 [26]). The official definition, as defined in [27]: "*Rosjava is an implementation of ROS in pure Java with Android support. It provides a client library that enables Java programmers to quickly interface with ROS Topics, Services, and Parameters. It also provides a Java implementation of a roscore.*"

The term "implementation of ROS in Java" means, that the developer is able to create standard publishers and subscribers directly in the Java application with all requisites. For example, all nodes running in the rosjava can be seen in a list of the nodes or in an rqt_graph after connection with a rosmaster is established. Still, there is one difference between subscribers in rosjava and standard subscribers, which will be studied in the next section.

The rosjava project was aimed at creating a tool for Android developers through which they can access the ROS environment, which robots use if desired without a deep knowledge of ROS. For this purpose, there is a repository on Github, called an android core (D. Kohler, K. Conley, 2011 [26]), which contains a collection of components and examples that are useful for developing ROS application on the Android device. The presented application is built on top of one of those examples. Although the rosjava project is still under active development most of the work was done 5-6 years ago, which sometimes leads to collisions with modern Android applications, but all of them can be solved with only small modifications of rosjava code.

6.3 Publishers and subscribers in RosJava

A standard node in a ROS application is able simultaneously to publish on multiple topics and subscribe to multiple topics (Figure 26), but publishers and subscribers in Rosjava are treated differently. Each subscriber/publisher in the rosjava is created as one node, and each node uses one thread from Android OS, which means that the application has to operate with a limited number of publishers and subscribers. Experiments with the rosjava showed that the number of nodes (i.e., the number of publishers/subscribers) running simultaneously on the Android platform is limited to six, which is not sufficient for a complex application. A possible solution is using one subscriber and one publisher running on the Android application, each of them operating with one status message (one additional subscriber for receiving images from the MAVs camera is used).

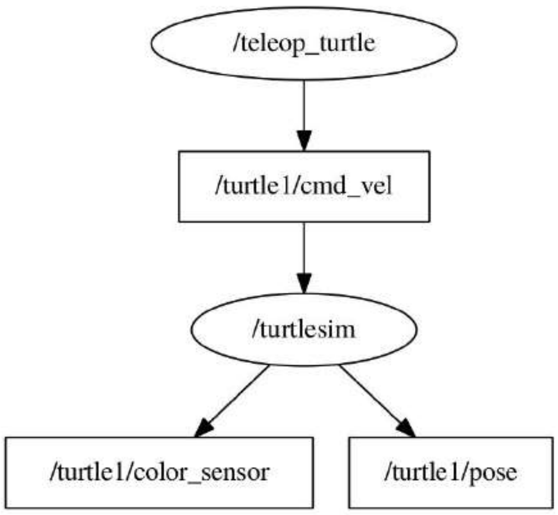


Figure 26: Ilustrative rqt graph for a standard ROS application. Notice, that the node */turtlesim* has one subscriber, that subscribes to topic */turtle/cmd_vel*, and two publishers, one publishes on topic */turtle1/color_sensor* and the second on *turtle/pose*. Source: [28].

6.4 Status messages

The concept of status messages is based on using one message carrying a set of instructions or collection of data from various sensors instead of having one communication channel (one topic) for each sensor and each instruction. Both messages (incoming to the application and outgoing from the application) are of a type stdFloat64MultiArray where every element of the array has a special meaning.

6.4.1 Outgoing messages

The outgoing messages are generated at the Android application as a result of user actions (e.g., if the user clicks on take off button the Take off message is generated and sent to MAV, if the joysticks, which provide the manual control, are used the manual control message is sent to MAV, etc.). The type of the message is determined by the first element of the message as shown in Table 1.

message[0]	meaning
1	Take off message
2	Manual control message
3	Trajectory message
4	Wavefront message
5	ZigZag message
6	Trapezoidal message
7	Change settings message

Table 1: A list of message types based on the first element of the status message.

Other elements of the message differ according to the message type:

- **Take off message**

  Take off message does not require any other elements as it is just an impulse for the MAV to take off.

- **Manual control message**

  Manual control of the MAV is provided through a set of two joysticks (see the manual control in the user manual 7.2.4). Controlling is provided through the angle of rotation of the joystick and its deflection. Example of a manual control message is presented in Table 2

Element in array	meaning
message[1]	Angle of the left joystick
message[2]	Deflection of the left joystick
message[3]	Angle of the right joystick
message[4]	Deflection of the right joystick

Table 2: Elements of a manual control message, which carries information about the position of joysticks which are used in manual control.

- **Trajectory message**

  Trajectory message carries the list of points through which the MAV is supposed to fly (Table 3).

Element in array	meaning
message[1]	n (number of points in trajectory)
message[2]	x_1 (latitude of a first point)
message[3]	y_1 (longitude of a first point)
message[4]	z_1 (altitude of a first point)
message[5]	x_2 (latitude of a second point)
message[6]	y_2 (longitude of a second point)
message[7]	z_2 (altitude of a second point)
...	...
message[3n]	y_n (longitude of a last point)
message[3n + 1]	z_n (altitude of a last point)
message[3n + 2]	v (desired flightVelocity)

Table 3: Elements of a trajectory message. The length of the trajectory message depends on the number of points in the trajectory. Each point in the trajectory is defined using three coordinates.

- **Wavefront message, ZigZag message, Trapezoidal message**

  Mapping, ZigZag and Trapezoidal messages use the same format. It is the same format as the trajectory message complemented with additional elements (Table 4), following after the message element representing the flight velocity. The next element after the velocity represents if onboard stitching will be used when the mapping is finished. The last item is the time used for rotation correction at the beginning of the flight.

Element in array	meaning
...	...
message[3n + 2]	v (desired flightVelocity)
message[3n + 3]	us (use onboard stitching)
message[3n + 4]	tr (time for initial rotation correction)

Table 4: Elements which are added to the trajectory message in order to use the trajectory message as a Mapping, ZigZag and Trapezoidal message.

6.4.2 Incoming messages

Incoming message (Table 5) is produced by the MAV. It consists of a list of information about the MAV, such as GPS position, altitude, battery state, etc. and a list of control signals such as isMAVInAir or messageReceived (confirm variable which is set to one when the MAV receives a new message).

Element in array	meaning
message[0]	UTMx (MAVs x possition in UTM coordinates)
message[1]	UTMy (MAVs y possition in UTM coordinates)
message[2]	alt (current altitude of the MAV)
message[3]	lat (current latitude of the MAV)
message[4]	lon (current longitude of the MAV)
message[5]	bv (current voltage of the batery)
message[6]	bp (aproximate percentage of the batery)
message[7]	MAVInAir (1 if the MAV is flying)
message[8]	messageRecieved (confirm command receival)

Table 5: Elements of the incoming message which carries a list of information about the current state of the MAV and a list of control signals.

6.5 Complete communication structure

The whole communication structure between the Android application and the ROS program running on the onboard pc of the MAV is shown in Figure 27.

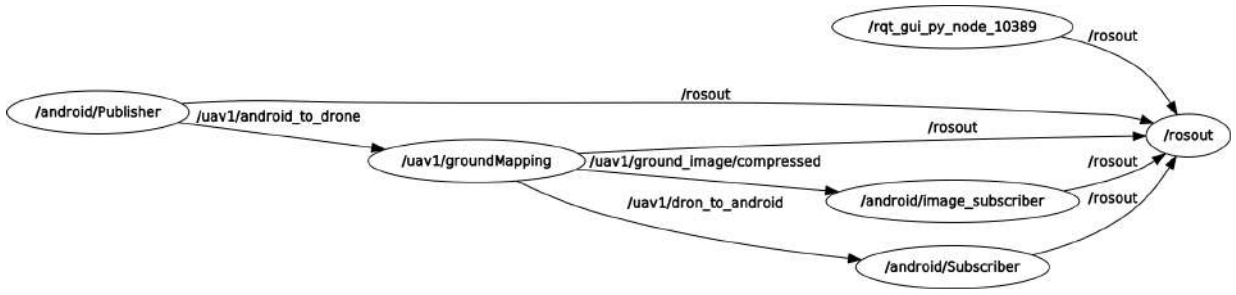


Figure 27: Project structure. Node running on UAV is called */uav1/groundMapping*. This node subscribes to a topic */uav1/android_to_drone*, which is used for status messages incoming from the Android application. Data from the MAV are published to a topic */uav1/dron_to_android*. The last topic */uav1/ground_image/compressed* is used for transporting camera images to the phone.

6.6 Android user interface

The rest of the Android application was created by author's coworker Š. Klouček [29], solving an issue of bird repulsing using UAVs and creating an easily expandable Android application for the MAV control. Klouček used the concept of fragments for the application structure, which made the application scalable. This approach is based on dividing the application to the set of cooperating components, called fragments, where each fragment is responsible for one part of the application logic (e.g., one fragment is used for manual control via a pair of joysticks 7.2.4, a second serves for saving the trajectories into the SQL database 7.2.3, the third fragment contains Google maps together with a logic of selecting trajectories and displaying the position of the MAV on the map 7.2.5, etc.).

The necessary modifications of the Klouček's application for the purpose of ground mapping are as follows:

- adding the logic for entering obstacle to the mapping area 7.2.5
- embedding the coverage algorithms to the application
- adding the possibility of displaying resizable images, to display the result of the mapping process to the user
- adding additional fragments and dialog windows for entering and evaluating mapping parameters

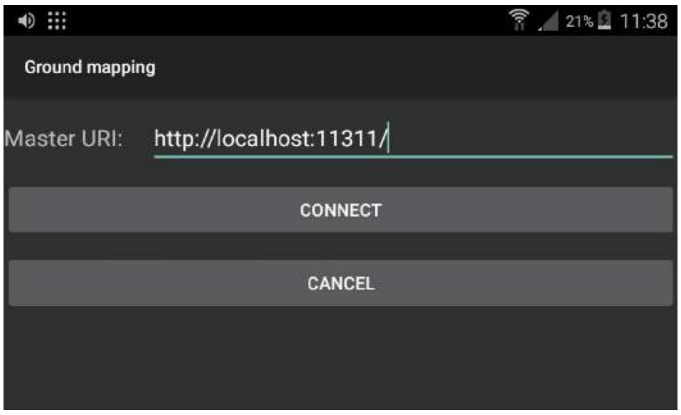
More information about the Android application structure can be found in Klouček's thesis (Š. Klouček, 2018 [29]).

7 User manual

This chapter serves as a user manual for the final application. After reading the manual a reader should be able to use all features of the application. A video demonstrating the system behavior can be found at `https://www.youtube.com/watch?v=epG3XwxIZeU&feature=youtu.be`.

7.1 Main screen

The Main screen contains a connection dialog (Figure 28) enabling the establishment of a connection with the MAV based on the MAVs IP address.



Master URI: http://localhost:11311/

Figure 28: Main screen. A Master URI defines the connection with the MAV. A connect button is used to establish the connection and enter the application. The Cancel button is used to enter the application without a previous connection.

For connecting to the MAV, the formula *localhost* in the Master URI had to be replaced with the IP address of the MAV. After the Master URI had been completed, the user can enter the application by clicking on the *Connect* button. The *Cancel* button serves for opening the application without establishing the connection in order to modify existing trajectories or plan new trajectories. It is possible to return back to the main screen and establish the connection later.

7.2 Application tabs

The menu on the left side of the screen (Figure 29) enables moving through the application. The menu opens by swiping from left to right on the screen or when clicked on the menu button in the top navigation bar.

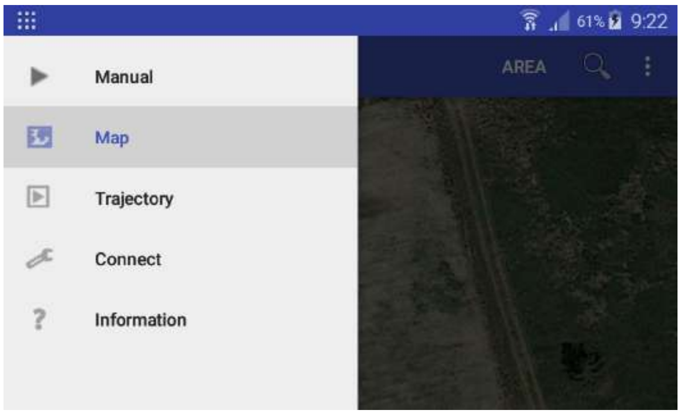
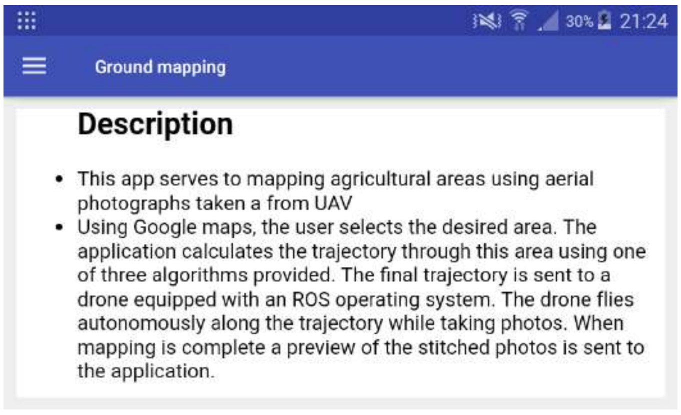


Figure 29: Navigation bar. After an item from the navigation bar is selected the application moves into the selected tab.

7.2.1 Information tab

Information tab contains a brief description of the application. A person unfamiliar with the application should be able to control basics of the application based on this description, without the requirement of reading the full user manual.



Description

- This app serves to mapping agricultural areas using aerial photographs taken a from UAV
- Using Google maps, the user selects the desired area. The application calculates the trajectory through this area using one of three algorithms provided. The final trajectory is sent to a drone equipped with an ROS operating system. The drone flies autonomously along the trajectory while taking photos. When mapping is complete a preview of the stitched photos is sent to the application.

Figure 30: Information tab containing the description of the application.

7.2.2 Connection tab

The Connection tab contains information about the connection with the MAV. If the MAV is fully connected, the user should see the MAV's IP address, also as *Is connected: true*. The *Refresh* button can be used to update the connection data. If the application was opened without a previous connection to the MAV the user can return to the main screen using the *Connect* button to establish the connection.

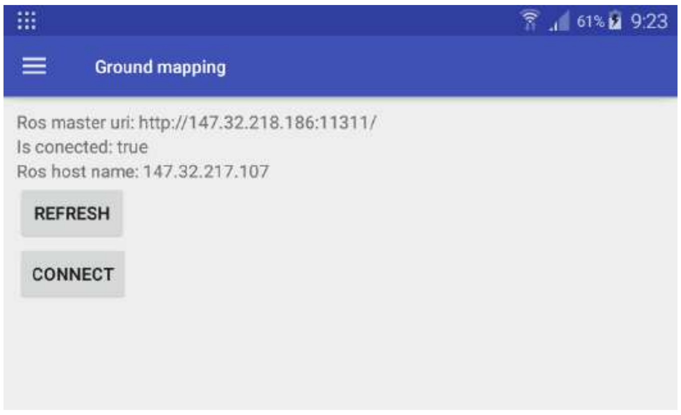
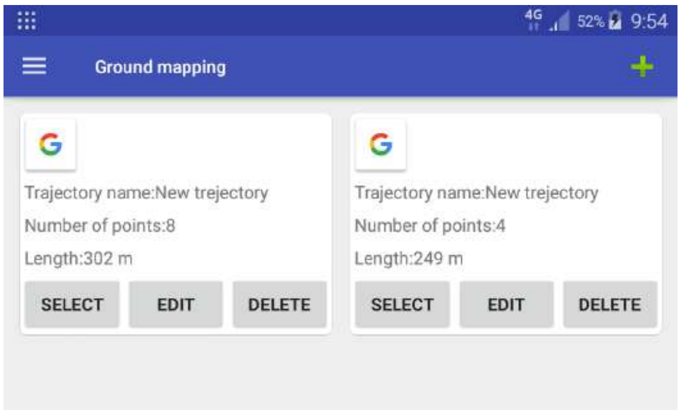

Ros master uri: http://147.32.218.186:11311/
Is conected: true
Ros host name: 147.32.217.107

Figure 31: Connection tab containing information about the established connection.

7.2.3 Trajectory tab

The Trajectory tab serves for managing trajectories. While mapping agricultural areas, a possibility of periodical area mapping is considered in order to evaluate long-term effects of used fertilizers. For this purpose, the application offers an option to prepare the trajectory once (with setting of obstacles and any necessary adjustments) and save it for future use.


Trajectory name:New trejectory
Number of points:8
Length:302 m

Trajectory name:New trejectory
Number of points:4
Length:249 m

Figure 32: Managing trajectories. Each tab represents one trajectory. The tab contains a short trajectory description (name, number of points, length of the trajectory) together with buttons for managing the trajectory.

- By clicking on the *Plus* button a new empty trajectory will be added to the list.

- Option *Select* moves the user to the map and zooms on the selected trajectory.

- Option *Edit* opens a popup window with additional options, such as adding a description to the trajectory, changing waypoints height in bulk, etc.

- Option *Delete* removes the trajectory (both from the application context and from the SQL database).

7.2.4 Manual control tab

The Manual control tab provides an option of a non-autonomous take off in problematic situations such as crowded areas or places where a precise starting sequence is necessary in order to avoid obstacles close by a take-off spot. Switching to the manual mode can be also used to interrupt the automatic flight and take control of the MAV in a critical situation.

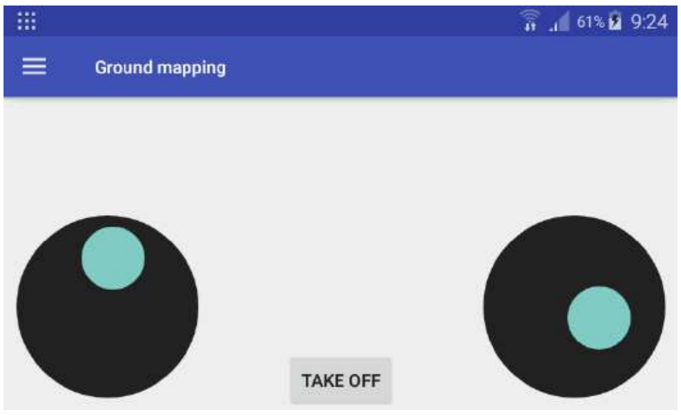


Figure 33: Manual control. The left joystick controls the vertical motions of the MAV together with the MAV's a rotation. The right joystick influences the horizontal motions of the MAV.

Joysticks

- Left joystick
  - Vertical deflection - upwards and downwards motions
  - Horizontal deflection - rotation of the MAV
- Right joystick
  - Vertical deflection - left and right motions
  - Horizontal deflection - forward and backward motions

7.2.5 Map tab

The Map tab is the key tab of the application. Here, the user creates the trajectories, calculates the coverage and observes the progress of a currently flown mission.

When entering the Map tab a view similar to Figure 34 will appear. If at least one point of the trajectory was set in a previous run of the application, the map will automatically zoom on the first point of the last used trajectory, otherwise it will zoom on the user's location. To search for a specific location the *magnifying glass* button can be used.

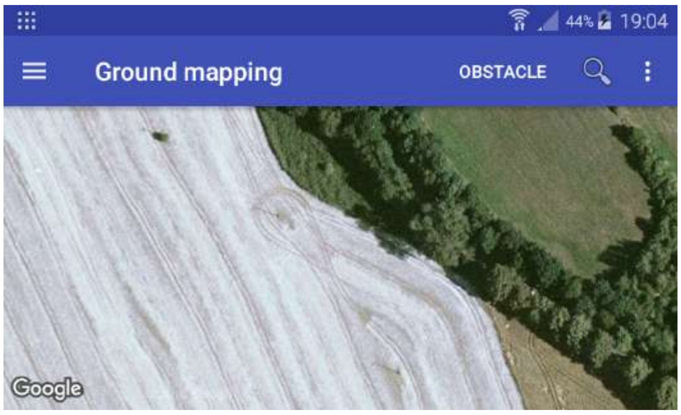


Figure 34: Map tab used for creating trajectories, defining mapped areas and observing the progress of the flown mission.

By default, each tap on the map will add one marker to the map. The user can add two types of markers to the map. Green markers represent boundaries of the mapped area while red markers serve for defining an obstacle inside the mapped area. The button highlighted in Figure 35 serves as a switch between defining the area and the obstacle.

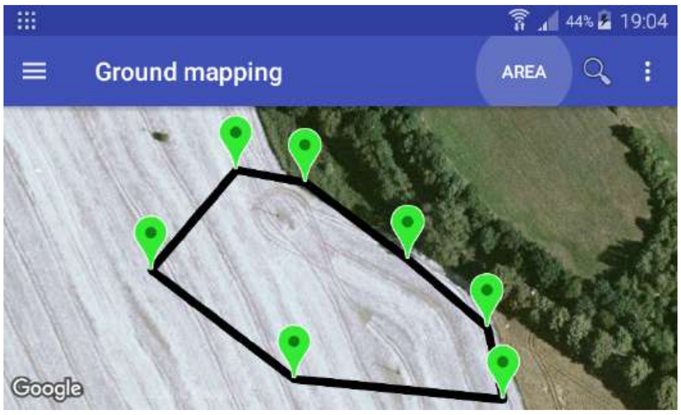


Figure 35: Defining area/trajectory using the green markers. Notice the Area button which defines if the markers belonging to area or obstacle are being inserted into the map.

If the marker symbol is tapped a pop up window for marker modification will appear. If the marker symbol is held, it can be moved on the map to the new location. If it is necessary to avoid some sub-region of the mapping area, the user can define this sub-region using red markers. When the trapezoidal algorithm is selected, the calculated trajectory will cover the whole mapping area while avoiding the forbidden sub-region. When the area is defined click on the menu button on the right side of the top navigation bar to access advanced commands.

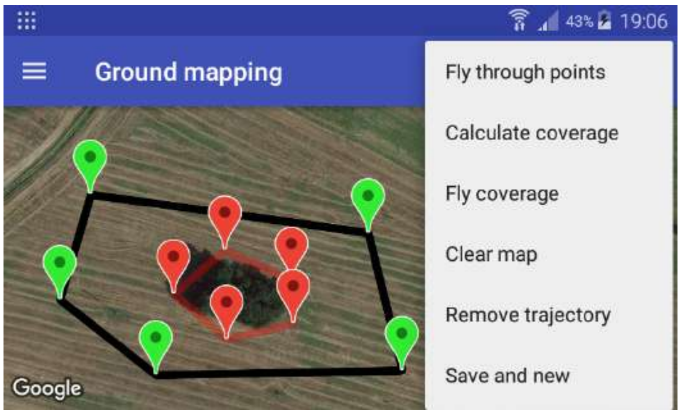


Figure 36: The right dropdown list, containing a list of advanced commands for calculating the trajectory and managing markers.

- *Fly through points command* sends the MAV a command to fly along the perimeter of the mapping area (along the green markers) starting from the first defined point.[1]. The MAV icon will move along the trajectory as the mapping progresses.
- *Calculate coverage command* opens a pop-up window to select the mapping algorithm and desired mapping height. Based on the selected algorithm the application will calculate coverage through the area. The coverage will be added to the map as a sequence of blue markers (Figures 37, 38, 39).
- *Fly coverage command* sends the MAV a command to fly along the trajectory calculated using *Calculate coverage* function [1].
- *Clear map command* removes all markers from the map.
- *Remove trajectory command* removes only trajectory markers from the map (e.g., the blue markers).
- *Save and new command* saves the currently defined area into the SQL database and clears the map.
- *Zoom on drone command* zooms on the MAVs current position.

[1]If the MAV is not in the air, it will take off automatically

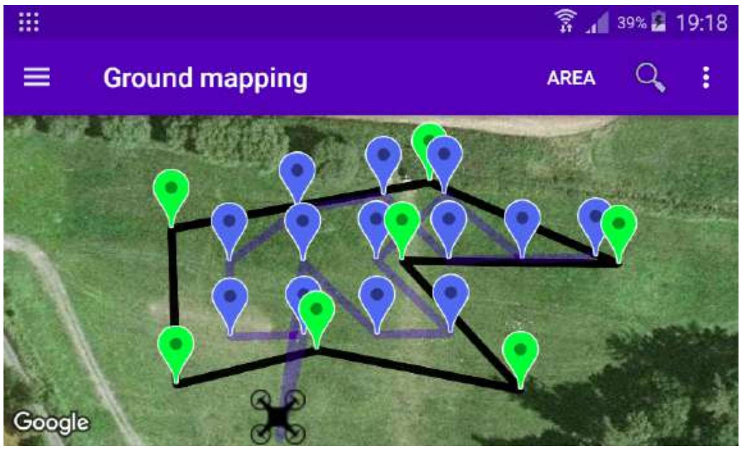


Figure 37: Example of the mapping trajectory calculated using the Wavefront algorithm.

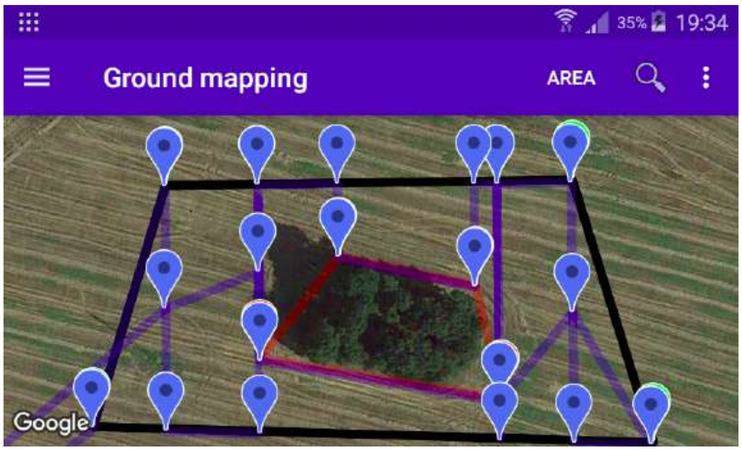


Figure 38: Example of the mapping trajectory calculated using the Trapezoidal algorithm.

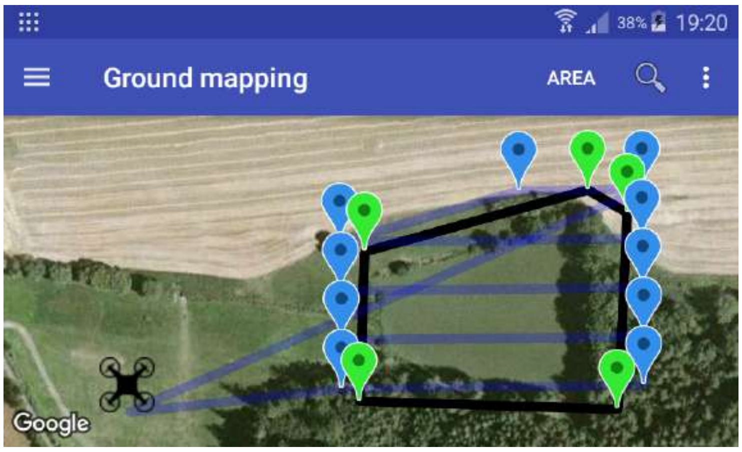


Figure 39: Example of the mapping trajectory calculated using the Zigzag algorithm.

Once the mapping is complete, the user receives a stitched photo of the mapped area (Figure 40) created using sensor based stitching. It is important to note that the received image is only a preview of the mapped area (sensor based stitching is able to achieve only a limited accuracy). Using the preview the user can check if the mapping finished properly (e.g., if the MAV mapped the whole area with no blind spots etc.). It is assumed, that for future analysis, images taken from the MAV will be stitched using additional software such as Autopano or Mircosoft ICE.

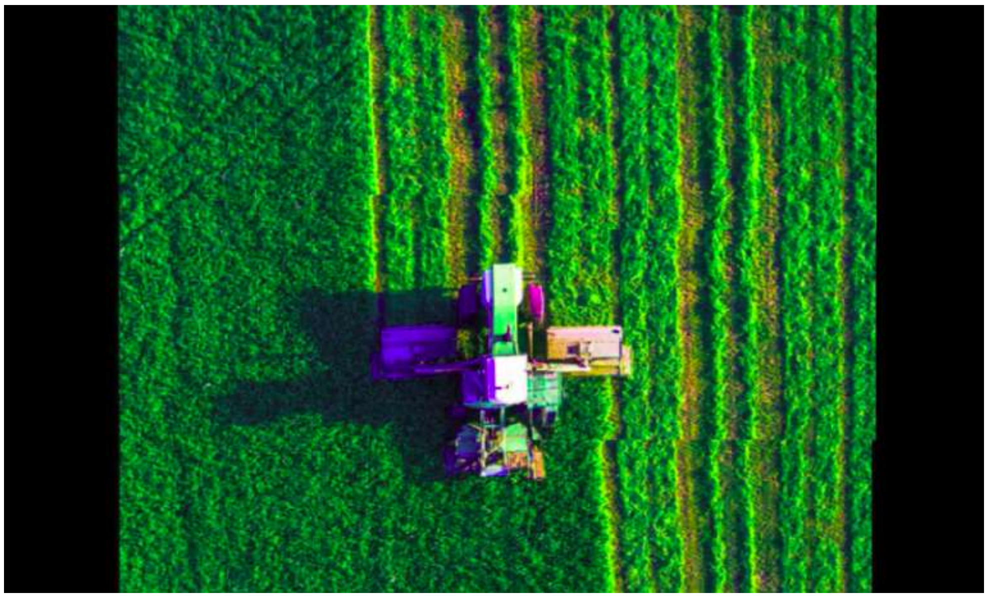


Figure 40: Result of the ZigZag mapping algorithm in the Gazebo simulation stitched using the presented sensor based stitching approach.

8 Verifying system functionalities

This chapter contains results of the realized experiments. At first behavior of the system in the simulation environment is examined, applying the three mapping algorithms on the same mapping areas. Their properties can be compared, especially their precision and speed. Followed by real platform experiments, which concentrate on evaluating results of the stitching process applied on real photos. Large version of selected images from the following chapter can be found in Appendix C *Images from experiments*. A video from the real platform experiments can be found at `https://www.youtube.com/watch?v=hLEZ9cOUXyM&feature=youtu.be`.

8.1 Simulation experiments

For the purpose of simulation experiments a Gazebo simulator containing a copy of a real MAV was used. An aerial photo of an agricultural area was used in the simulation to create conditions similar to reality. Several different areas were mapped in order to compare the precision of the presented algorithms. Each area was mapped without obstacle using all implemented algorithms and with obstacle using the Trapezoidal algorithm. The following pages show outputs of the algorithms applied on the first area (Figure 41). The rest of the results can be found on the enclosed CD. Statistical results of the simulated experiments are shown in Table 6 and the end of this section.

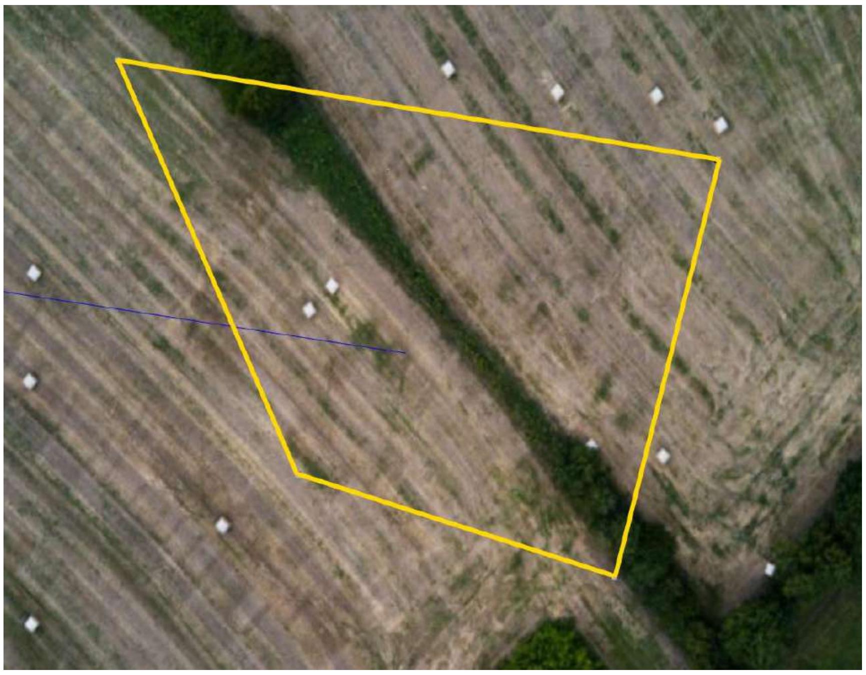


Figure 41: Mapped area in the Gazebo simulator. Full image - Figure 68.

8.1.1 Wavefront algorithm

At first the behavior of the Wavefront algorithm was examined. As expected, the Wavefront algorithm (Figure 42, 43) reached the highest accuracy thanks to the extra stabilization time, before taking a photo. The precision of the taken photos also positively influences the results of the stitching algorithms, as can be seen in Figures 44, 45.

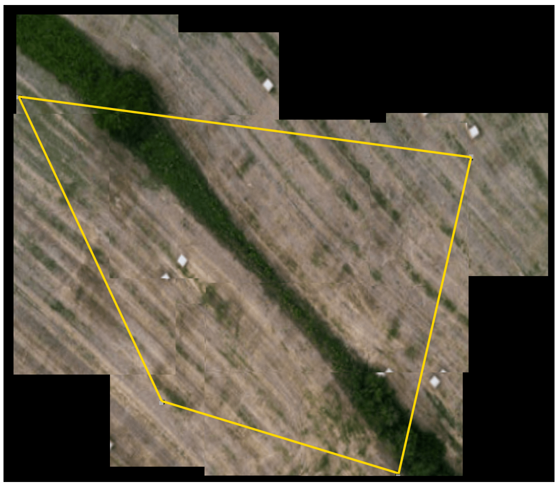


Figure 42: Result of the sensor based stitching of mapping by Wavefront algorithm. Full image - Figure 69.

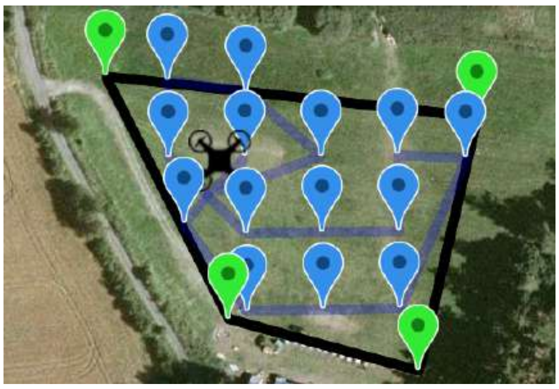


Figure 43: Preview of the flown trajectory.

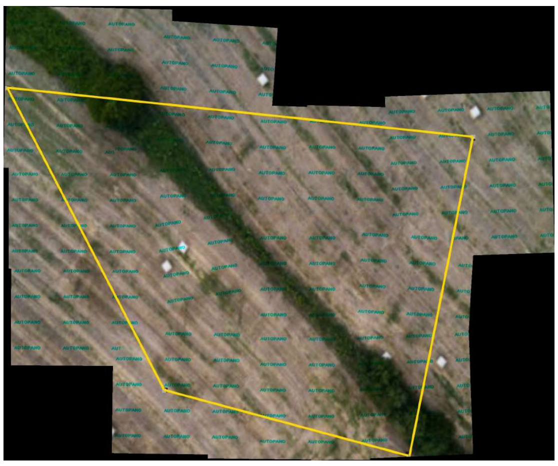


Figure 44: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Wavefront algorithm. Full image - Figure 70.

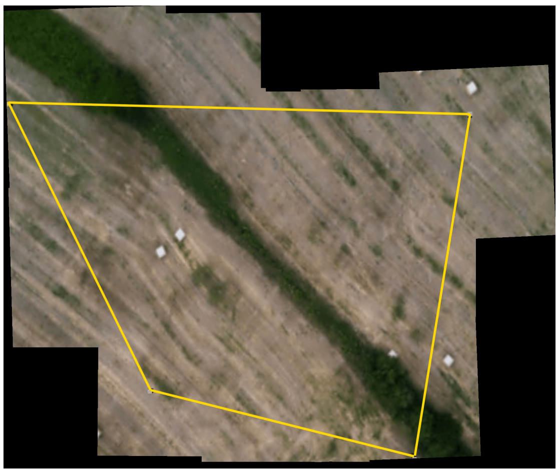


Figure 45: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Wavefront algorithm. Full image - Figure 71.

8.1.2 Zigzag algorithm

The Zigzag algorithm (Figure 46, 47) does not reach the accuracy of the Wavefront algorithm but requires shorter mapping time. The worse accuracy is mainly caused by the photo taking during the MAVs movement, at which to retain a constant tilt, which causes the biggest distortion, is more difficult. A second significant distortion is the delay of the simulated camera which depends on current computer usage and which causes image shift inside the canvas. The results of the Autopano and OpenCV stitching algorithms are given in Figures 48, 49.

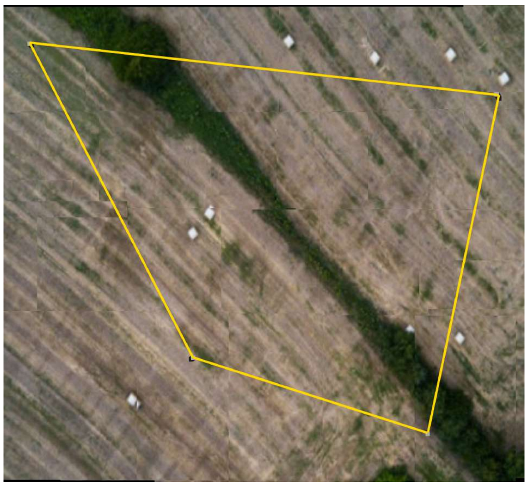


Figure 46: Result of the sensor based stitching of mapping by Zigzag algorithm. Full image - Figure 72.

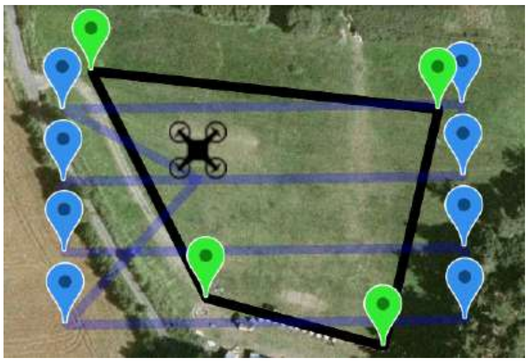


Figure 47: Preview of the flown trajectory.

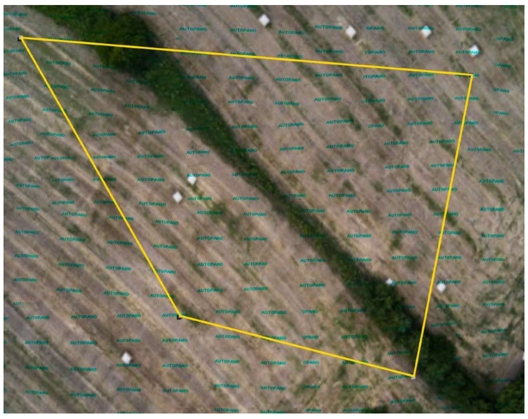


Figure 48: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Zigzag algorithm. Full image - Figure 73.

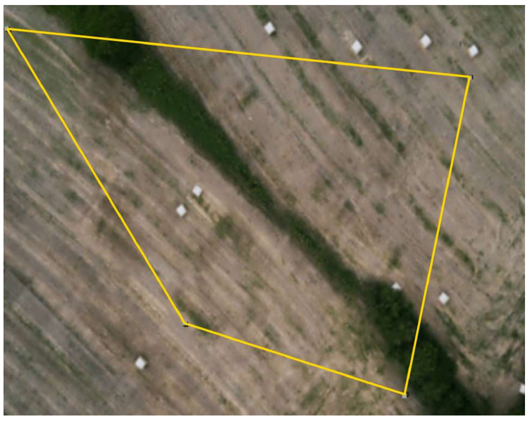


Figure 49: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Zigzag algorithm. Full image - Figure 74.

8.1.3 Trapezoidal algorithm without an obstacle

The accuracy of the Trapezoidal algorithm (Figure 50, 51) applied on the mapping area without an obstacle is similar to the results of the Zigzag algorithm both in a sensor based stitching also as in stitching using the additional stitching algorithms (Figure 52, 53).

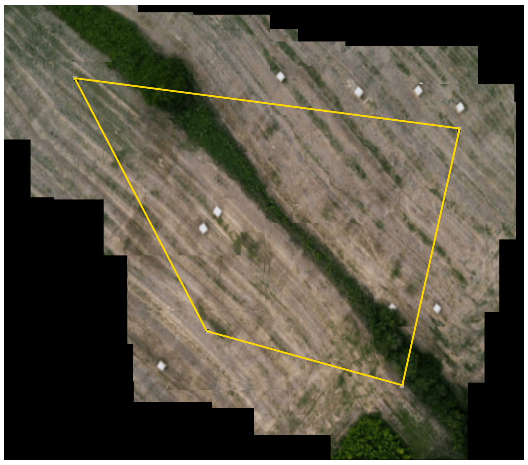


Figure 50: Result of the sensor based stitching of mapping by Trapezoidal algorithm. Full image - Figure 75.

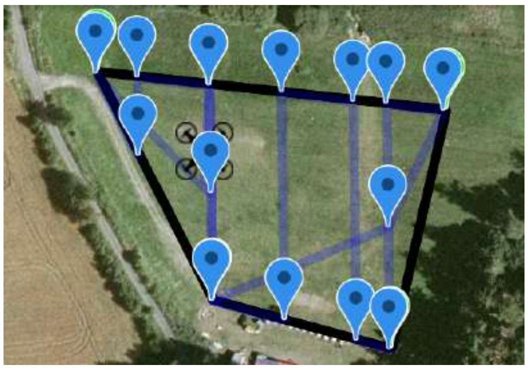


Figure 51: Preview of the flown trajectory.

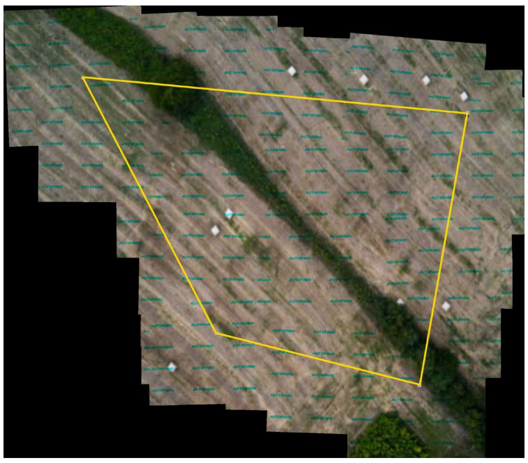


Figure 52: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Trapezoidal algorithm. Full image - Figure 76.

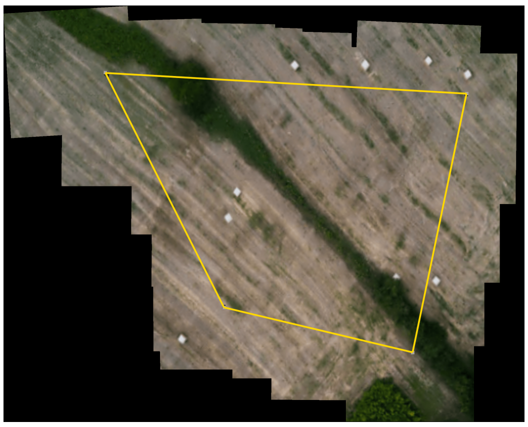


Figure 53: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Trapezoidal algorithm. Full image - Figure 77.

8.1.4 Trapezoidal algorithm with an obstacle

The quality of the sensor stitching process (Figure 54) decreases while mapping an area with an obstacle due to an increasing number of flight direction changes above the mapping area, which causes the changes of the MAVs tilt resulting in image placement distortions. The quality of the Autopano and OpenCV stitching algorithms remain constant (Figures 56,57).

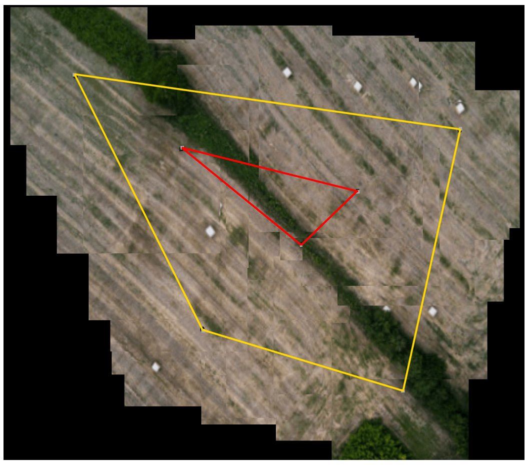


Figure 54: Result of the sensor based stitching of mapping by Trapezoidal algorithm applied on area with an obstacle

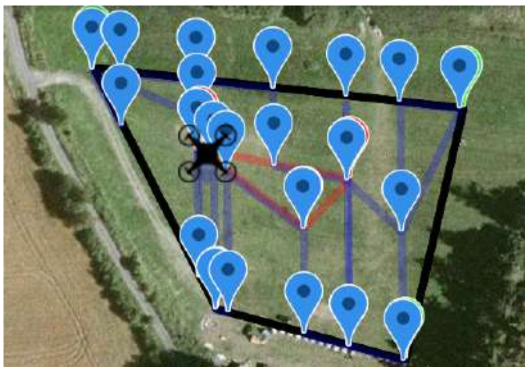


Figure 55: Preview of the flown trajectory.

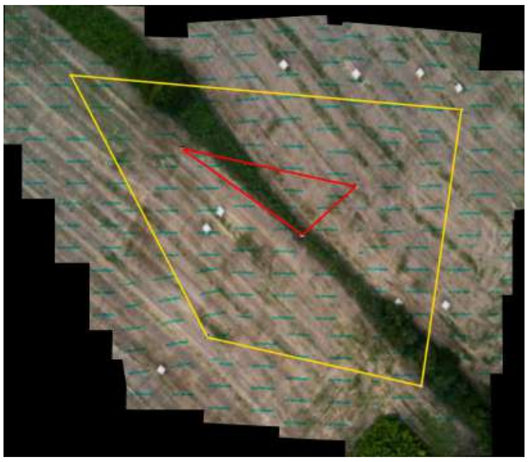


Figure 56: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Trapezoidal algorithm.

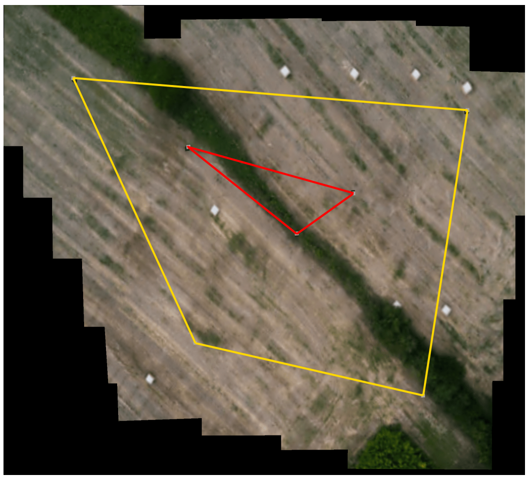


Figure 57: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Trapezoidal algorithm.

The following table shows statistical results of the realized experiments. Distortion in image placement was measured at each map using a set of points in which it was possible to measure a dislocation of the placed photo. The inaccuracy of single image placement measured in pixels was recalculated to a percentual inaccuracy with respect to the size of the image. The total inaccuracy of image placement in the map is calculated as an average of fractional inaccuracies. The inaccuracy of each algorithm is calculated from the distortions measured in individual experiments.

algorithm	experiment	time[s]	photos taken	dislocation[%]	average disloc. [%]
Wavefront	1	360	14	0.040	0.037
	2	417	18	0.033	
	3	472	24	0.037	
Zigzag	1	322	26	0.057	0.060
	2	332	26	0.066	
	3	297	37	0.056	
Trapezoidal	1	396	37	0.046	0.058
	2	400	36	0.063	
	3	433	57	0.064	
Trapezoidal with obstacle	1	460	43	0.085	0.068
	2	440	40	0.057	
	3	495	66	0.063	

Table 6: Table of results of the presented flights. Experiments with the same number, mapped the same area from the same altitude using the same speed. Experiments number one were executed in 20m altitude with a flight velocity 1.5m/s, experiments number two at 19m and 1.5 m/s and experiments three at 17m and 2ms/s.

Although the precision of the algorithms calculated from the position of individual pictures inside the map is only approximate, it is possible to confirm that the Wavefront algorithm reaches the highest accuracy with the lowest number of photos. The Zigzag and Trapezoidal algorithms reach a similar accuracy at an area without obstacles. The biggest distortion occurs in the Trapezoidal algorithm while avoiding obstacles.

8.2 Real platform experiments

8.2.1 Hardware specifications

The camera used in experiments, both simulated and real, is the latest product from Mobius action cameras called *Mobius C2 lens*, a model of which was integrated into the Gazebo simulator (Figure 58). It is a full HD sports camera, abundantly used in UAV applications, because of its low weight and damage resistance. The advantage of this camera against other cameras is a wide FOV (field of view), which is 132 degrees.

Figure 58: Mobius C2 lens [30].

List of camera specifications:

- 1080p HD @ 30fps, 720p HD at 60 or 30fps
- Micro SD Cards up to 32GB
- Photo resolution 2304 x 1536 (default) / 1920 x 1080 / 1280 x 720 / 848 x 480
- USB Specification: Mini USB 2.0 port
- Weight: 42 grams (camera only)/ 52 grams (camera and mounting sleeve)

Other camera specifications can be found on Mobius official web page [31];

Details about the MAV platform can be found in (G. Loianno, at al., 2018 [32]), (V. Spurný, at al., 2018 [33]) or (T. Báča, at al., 2018 [34]). Details about the MAV control system are well described at (T. Báča, G. Loianno and M. Saska, 2016 [35]), (T. Báča, P. Štěpán and M. Saska, 2017 [36]) or (T. Báča, D. Heřt, at al., 2018 [37]).

8.2.2 Experiments

Two experiments were carried out in order to examine the behavior of the system with a real platform. Results of the Trapezoidal algorithm and the Wavefront algorithm from the first experiment are presented. The Trapezoidal algorithm was applied on a surface composed of a cultivated grassy area, a field and uncultivated high grass (Figure 59 polygon A), which offers a lot of patterns for the stitching software. The Wavefront algorithm was applied on a large planar grassy area (Figure 59 polygon B) with a minimal number of patterns, which could be used by the stitching software as reference points, in order to test the capabilities of the stitching software. An area with a minimal number of patterns was also selected for the second experiment. This area (Figure 60) was mapped using the ZigZag algorithm.

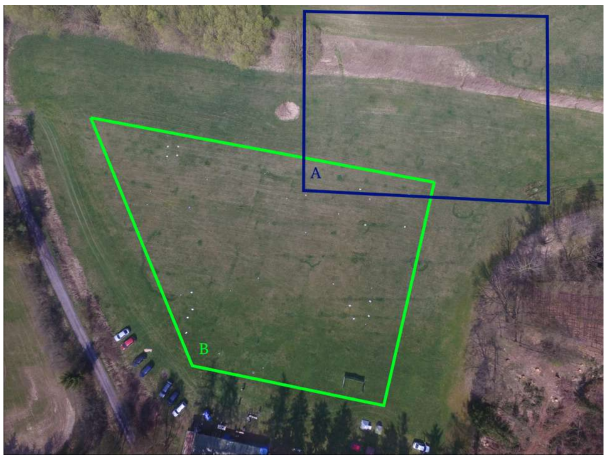


Figure 59: Picture of mapped areas in the first experiment taken from a DJI Phantom flying at 100m. Full image - Figure 78.

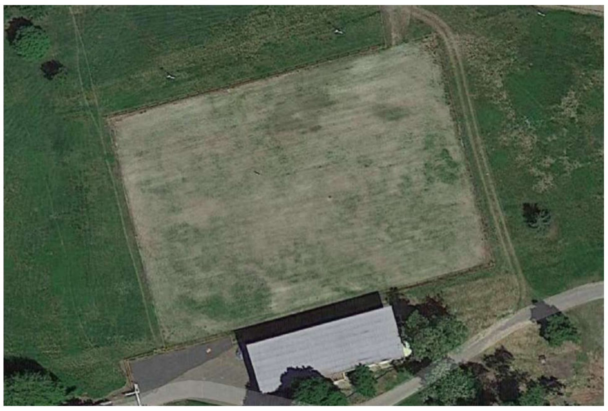


Figure 60: Satelite image of the area mapped in the second experiment. Full image - Figure 83.

8.2.3 Trapezoidal algorithm

The Trapezoidal algorithm used a MAV flying at 10m altitude in 1m/s. The sensor-based stitching algorithm behaved as expected (Figure 61), also did the Autopano stitching software (Figure 62). However, the onboard stitching algorithm using the OpenCV components was able to stitch only several photos (Figure 63), which suggests that the OpenCV stitching algorithm may be inappropriate for stitching images of environment with a small number of reference point such as agricultural images.

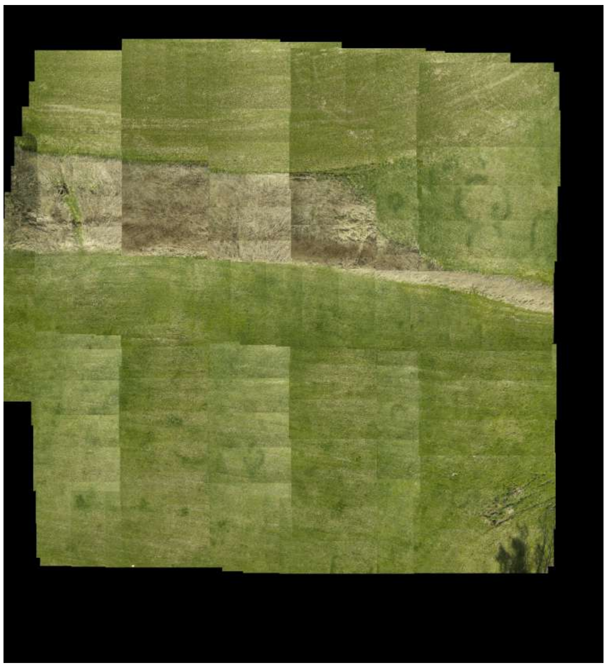


Figure 61: Demonstration of the Trapezoidal algorithm in a real system, during mapping an area from an altitude of 10m. Full image - Figure 79.

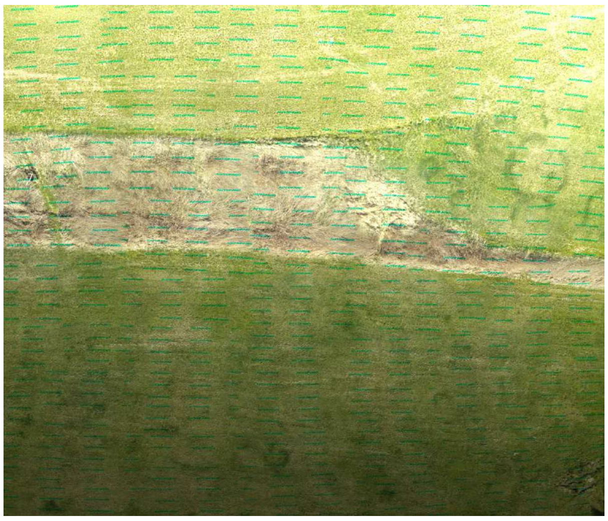


Figure 62: Result of the Autopano stitching software applied on a set of real photos collected using the Trapezoidal algorithm. Full image - Figure 80.

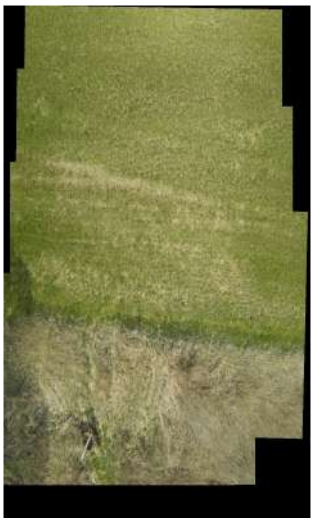


Figure 63: Result of the OpenCV stitching software applied on a set of real photos collected using the Trapezoidal algorithm.

8.2.4 Wavefront algorithm

The Wavefront algorithm used the MAV flying 2m/s in an altitude of 15m. The accuracy of the sensor based stitching (Figure 64) corresponds to the accuracy of the simulated results. A wrong camera mode was selected in this experiment which caused that the camera used a different FOV than expected so the collected images had a lower overlap (20 %) in the y-axis. Due to the smaller overlap even the Autopano stitching software had a problem to stitch all the images (Figure 65), which shows the importance of the image overlap while stitching images with a low number of refernce points.

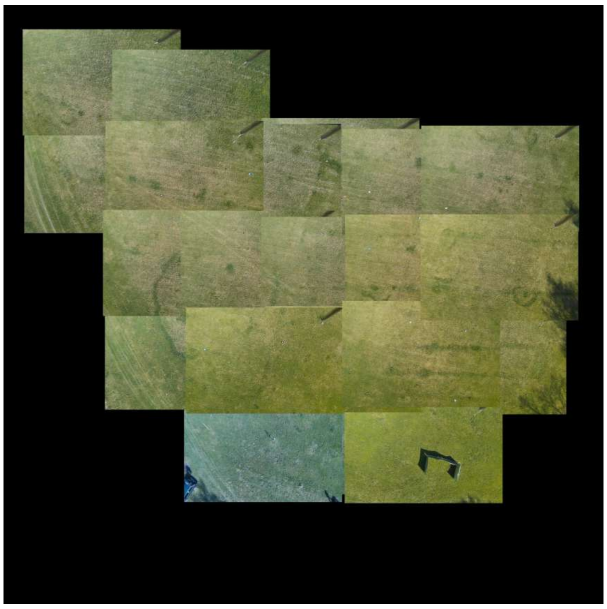


Figure 64: Demonstration of the Wavefront algorithm in a real system, while mapping an area from the altitude of 15m. Full image - Figure 81.

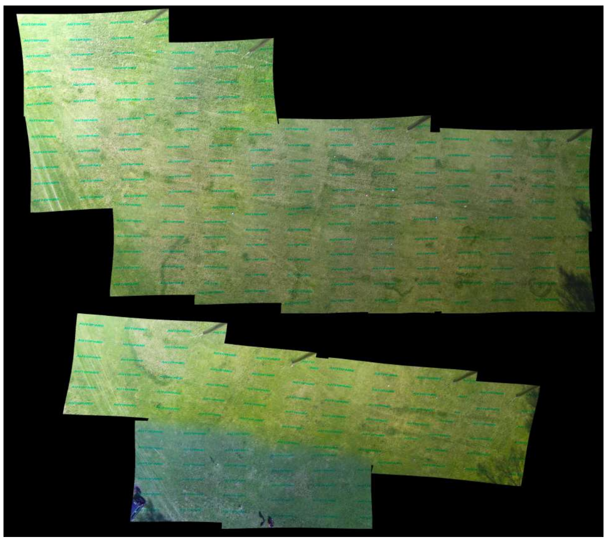


Figure 65: Result of the Autopano stitching software applied to real photos collected during mapping using the Wavefront algorithm. Full image - Figure 82.

8.2.5 Zigzag algorithm

The ZigZag algorithm mapped the area from the 20m altitude with a flight velocity 1.5m/s. A result of the sensor based stitching is shown in Figure 66. With the correct overlap the Autopano stitching software was able to create a complete map of the desired area (Figure 67) although the OpenCV stitching software was again able to stitch only several photos, which confirms that the OpenCV stitching library is inappropriate for stitching agricultural images.

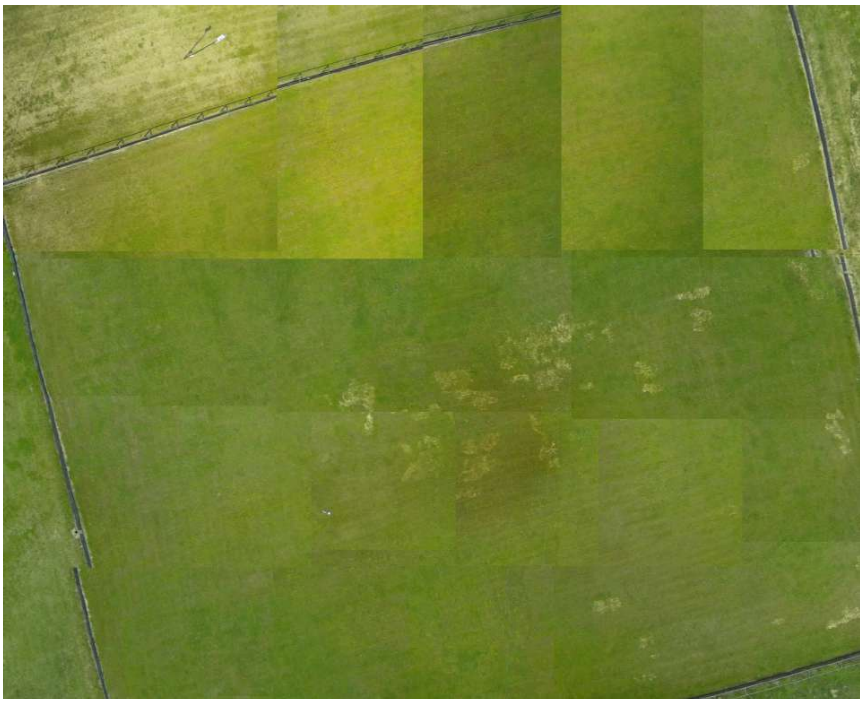


Figure 66: Demonstration of the ZigZag algorithm in a real system, while mapping an area from the altitude of 20m. Full image - Figure 84.

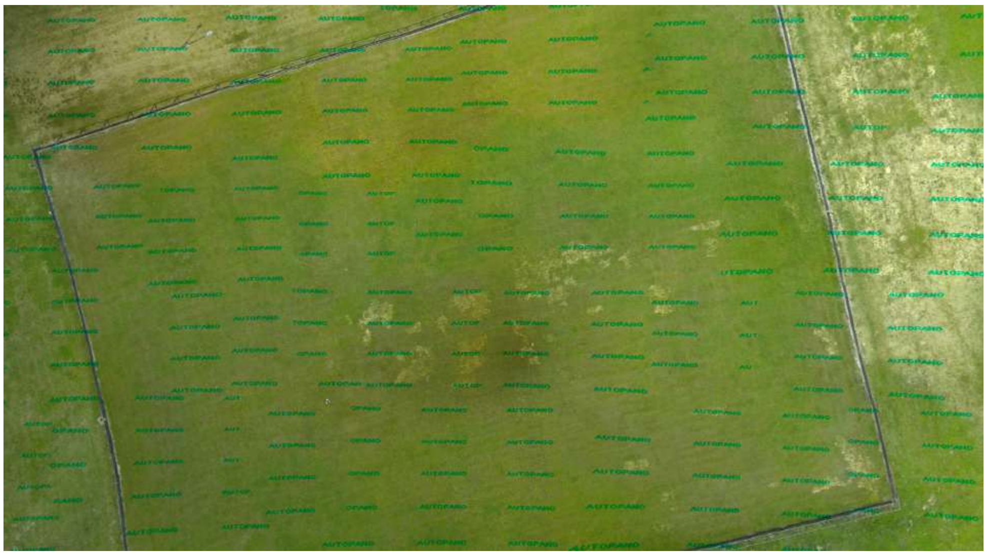


Figure 67: Result of the Autopano stitching software applied to real photos collected during mapping using the ZigZag algorithm. Full image - Figure 85.

9 Conclusion

The presented thesis dealt with designing and implementing an application for precision agriculture in order to offer a light solution of area scanning, available to work on a MAV equipped with minimal hardware equipment. The application aimed at monitoring field and water areas which offer minimal information for stitching algorithms. Therefore a sensor based stitching algorithm, which uses an estimated position of the MAV in the stitching process, was implemented in the application. The position of the MAV is estimated using a sensory fusion described in (T. Báča, G. Loianno and M. Saska, 2016 [35]), (T. Báča, P. Štěpán and M. Saska, 2017 [36]) and (M. Saska, T. Báča et al., 2017 [38]). Three different algorithms of CCPP were implemented to examine the influence of a MAV movement on the precision of the stitching process. The application was created using a RosJava tool for Android developers and shows that it is possible to create an Android application providing a complete set of functions for MAV piloting and real-time trajectory construction, available to immediately display a preview of a mapped area. The application was successfully tested on multiple mobile devices with different levels of the Android operating system.

The simulation experiments presented at the beginning of chapter 8 compared behavior of the implemented algorithms. The Wavefront algorithm proved to be most accurate algorithm, but requires the longest time for mapping areas. The Zigzag algorithm reaches the shortest mapping times with a satisfying level of accuracy, which makes it the best algorithm for mapping large areas. The Trapezoidal algorithm offers the ability to avoid an obstacle inside the mapping area. Without the obstacle, the Trapezoidal algorithm reaches similar results and mapping times as the Zigzag algorithm.

Real platform experiments presented in the second part of chapter 8 showed, that the application works well even on a real system. The Autopano stitching software proved to be appropriate for stitching agricultural images providing a small number of reference points. On the contrary, the OpenCV stitching algorithm showed as inappropriate for agricultural image stitching. A video from the real platform experiments can be found at `https://www.youtube.com/watch?v=hLEZ9cOUXyM&feature=youtu.be`. Results from the presented algorithms and videos demonstrating the application are also provided on the enclosed CD.

References

[1] Parrot Beebop 2 [photo]. Parot corp. [online]. Available at *https://www.parrot.com*

[2] V. Lukas, J. Novák, L. Neudert, I. Svobodova, F. Rodriguez-Moreno, M. Edrees, J. Kren, *The Combination of Uav Survey and Landsat Imagery for Monitoring of Crop Vigor in Precision Agriculture*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLI-B8, 2016.

[3] C. Zhang, J. Kovacs, *The application of small unmanned aerial systems for precision agriculture: a review*, Precision Agriculture, Volume 13, Issue 6, pp 693-712, 2012

[4] K. C. Swain, S. J. Thomson and H. P. W. Jayasuriya, *Adoption of an unmanned helicopter for lowaltitude remote sensing to estimate yield and total biomass of a rice crop.* Transactions of the ASABE , vol. 53, pp 21-27, 2010

[5] P. Katsigiannis, L. Misopolinos, V. Liakopoulos, T. K. Alexandridis and G. Zalidis, *An autonomous multi-sensor UAV system for reduced-input precision agriculture applications*, 24th Mediterranean Conference on Control and Automation (MED), Athens, 2016.

[6] L. Piermattei, at al., *Multispectral data processing from unmanned aerial vehicles: application in precision agriculture using different sensors and platforms*, EGU General Assembly, 2017.

[7] T. Adao, at al., *Hyperspectral Imaging: A Review on UAV-Based Sensors, Data Processing and Applications for Agriculture and Forestry.* Remote Sensing, 9(11): 1110-1125, 2017

[8] E. Honkavaara, at al., *Processing and Assessment of Spectrometric, Stereoscopic Imagery Collected Using a Lightweight UAV Spectral Camera for Precision Agriculture*, Remote Sensing, 5(10): 5006-5039, 2013

[9] F. Yasutomi, M. Yamada and K. Tsukamoto, *Cleaning robot control*, In Proc. IEEE International Conference on Robotics and Automation, pp. 1839–1841, 1988.

[10] P. Atkar, A. Greenfield, D. Conner, H. Choset and A. Rizzi, *Uniform cover-age of automotive surface patches*, The International Journal of Robotics Research, 24(11), 883-898, 2005.

[11] M. Ollis and A. Stentz, *First results in vision-based crop line tracking*, In Proc. IEEE International Conference on Robotics and Automation, Vol. 1, pp. 951–956, 1996.

[12] F. Demim, A. Nemra, K. Louadj, M. Zakaria, M. Hamelain and A. Bazoula, *Simultaneous localization and mapping algorithm for unmanned ground vehicle with SVSF filter*, In Proc. IEEE 8th International Conference on Modelling, Identification and Control, pp. 155-162, 2016.

[13] R. Szeliski, *Image Alignment and Stitching:A Tutorial*, Now Foundations and Trends, pp.116-125, 2006

[14] Q. Fu, H. Wang, *A fast image stitching algorithm based on SURF*, International Conference on Communications in China (ICCC), 2017

[15] X. Zhou, H. Zhang, Y. Wang, *A Multi-Image Stitching Method and Quality Evaluation*, 4th International Conference on Information Science and Control Engineering (ICISCE), 2017

[16] Ch. Cheng, X. Wan, X. Li, *UAV image matching based on surf feature and harris corner algorithm*, 4th International Conference on Smart and Sustainable City (ICSSC), 2017

[17] Y. Ha, H. Kang, *Evaluation of feature based image stitching algorithm using OpenCV*, 10th International Conference on Human System Interactions (HSI), 2017

[18] Micro Aerial Vehicle [photo]. MRS CTU [online]. Available at *http://mrs.felk.cvut.cz/research/micro-aerial-vehicles*

[19] A. Zelinsky, R. Jarvis, J. C. Byrne and S. Yuta, *Planning paths of complete coverage of an unstructured environment by a mobile robot*, Proceedings of International Conference on Advanced Robotics, pp. 533–538, 1993.

[20] A. Nedjati, et al. *Complete Coverage Path Planning for a Multi-UAV Response System in Post-Earthquake Assessment.* Robotics, 2016.

[21] *Trapezoidal Cell Decomposition and Coverage* [online]. Middle east technical university, [Accessed December 2017]. Available at *http://user.ceng.metu.edu.tr/ akifakkus/-courses/ceng786/hw3.html*

[22] The OpenCV library [online], [Accessed March 2018]. Available at *https://opencv.org*

[23] The Autopano stitching software [online], [Accessed March 2018]. Available at *http://www.kolor.com/autopano*

[24] M. Quigley, B. Gerkey, W.D. Smart, *Programming robots with ROS*, Sebastopol: O'Reilly Associates Incorporated, ISBN 1449323898, 2015

[25] S. James, *Android application development for Java programmers*, Boston: Cengage Learning, 2013

[26] D. Kohler and K. Conley, *rosjava–an implementation of ros in pure java with android support*, [online], [Accessed December 2017]. Available at $https://github.com/rosjava/rosjava_core$, 2011

[27] Rosjava [online], [Accessed September 2017]. Available at $http://rosjava.github.io/rosjava_core/latest/index.html$

[28] J. M. O'Kane, *A Gentle Introduction to ROS*, J. M. O'Kane, 2014.

[29] Š. Klouček, *Android application for control of an unmanned helicopter carrying a bird repeller, CTU, 2018*

[30] Mobius C2 lens [photo]. Mobius Action cams [online]. Available at *https://www.banggood.com/Mobius-New-Version-Wide-Angle-Lens-C2-1080P-HD-Mini-Action-Camera-p-985644.html.*

[31] Mobius action cameras [online], [Accessed January 2018]. Available at *https://www.buymobius.com/pages/mobius-actioncam-specifications*

[32] G. Loianno, V. Spurný, T. Báča, J. Thomas, D. Thakur, T. Krajník, A. Zhou, A. Cho, M. Saska and V. Kumar. *Localization, Grasping, and Transportation of Magnetic Objects by a team of MAVs in Challenging Desert like Environments.* IEEE Robotics and Automation Letters (RAL), 2018.

[33] V. Spurný, T. Báča, M. Saska, R. Pěnička, T. Krajník, G. Loianno, J. Thomas, D. Thakur and V. Kumar. *Cooperative Autonomous Search, Grasping and Delivering in a Treasure Hunt Scenario by a Team of UAVs.* Under review of Jurnal of Field Robotics (JFR), 2018.

[34] T. Báča, P. Štěpán, V. Spurný, D. Heřt, R. Pěnička, M. Saska, J. Thomas, G. Loianno and V. Kumar. *Autonomous Landing on a Moving Vehicle with an Unmanned Aerial Vehicle.* Under review of Jurnal of Field Robotics (JFR), 2018.

[35] T. Báča, G. Loianno and M. Saska. *Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles.* In 21st International Conference on Methods and Models in Automation and Robotics (MMAR), 2016.

[36] T. Báča, P. Štěpán and M. Saska. *Autonomous Landing On A Moving Car With Unmanned Aerial Vehicle.* In The European Conference on Mobile Robotics (ECMR), 2017.

[37] T. Báča, D. Heřt, G. Loianno, M. Saska and V. Kumar. *Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles.* Under review of Robotics and Automation Letters (RAL), 2018.

[38] M. Saska, T. Báča, J. Thomas, J. Chudoba, L. Přeučil, T. Krajník, J. Faigl, G. Loianno and V. Kumar. *System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization.* Autonomous Robots 41(4):919–944, 2017.

A CD Content

Table 7 lists names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
thesis_sources	latex source codes
android_sources	android application source codes
ros_sources	ros application source codes
experiments	results of the experiments
videos	demonstration videos

Table 7: CD Content

B List of abbreviations

Table 8 lists abbreviations used in this thesis.

Abbreviation	**Meaning**
ROS	robot operating system
API	application programming interface
GUI	graphical user interface
UAV	unmanned aerial vehicle
MAV	multi-rotor helicopter
CCPP	complete coverage path planning
OS	operating system
API	application programming interface
DFS	deep first search
BFS	breadth-first search
TSP	traveling salesman problem
SSH	secure shell
FOV	field of view

Table 8: List of abbreviations

C Images from experimnts

Large version of images from the chapter *Verifying system functionalities* 8.

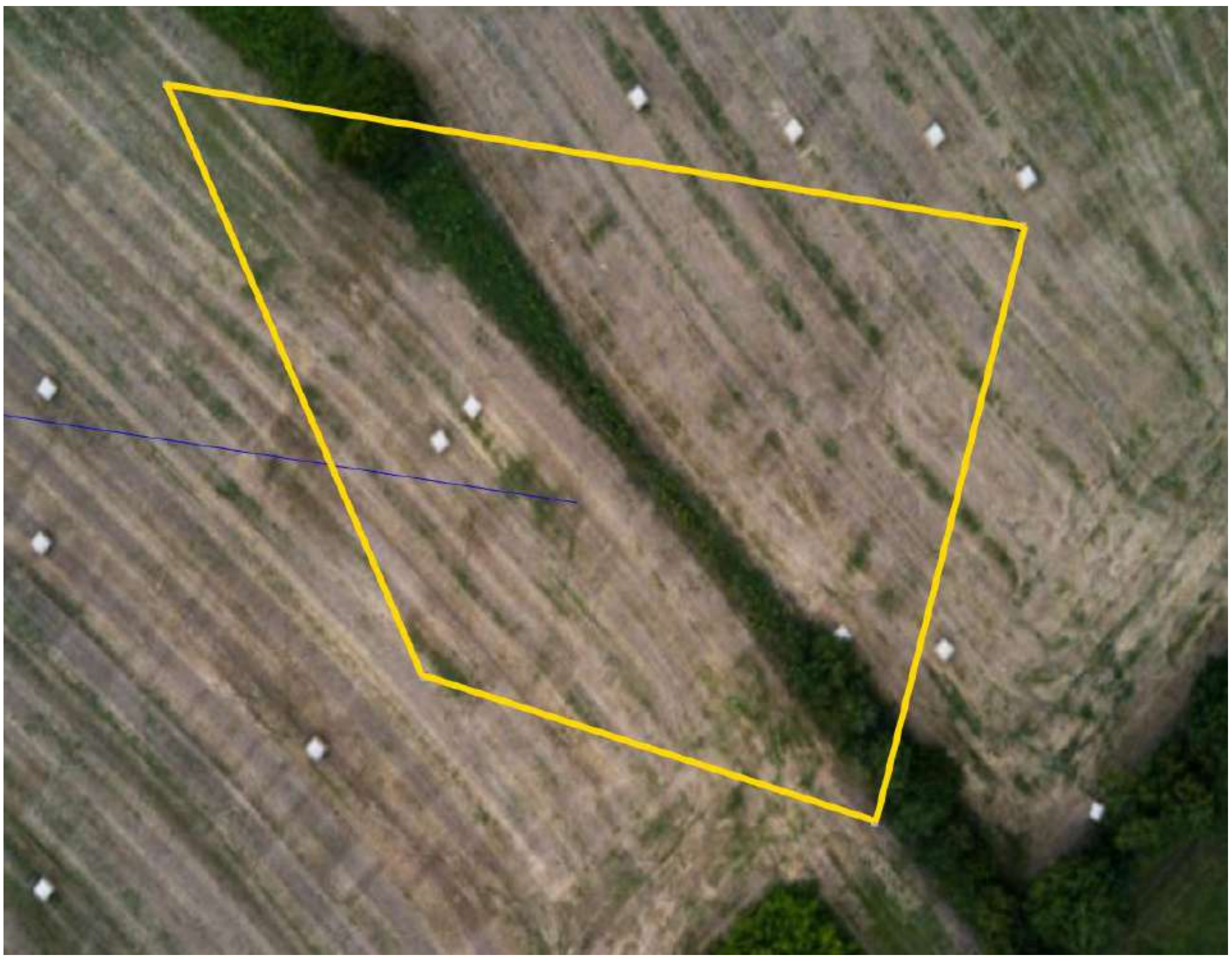


Figure 68: Mapped area in the Gazebo simulator.

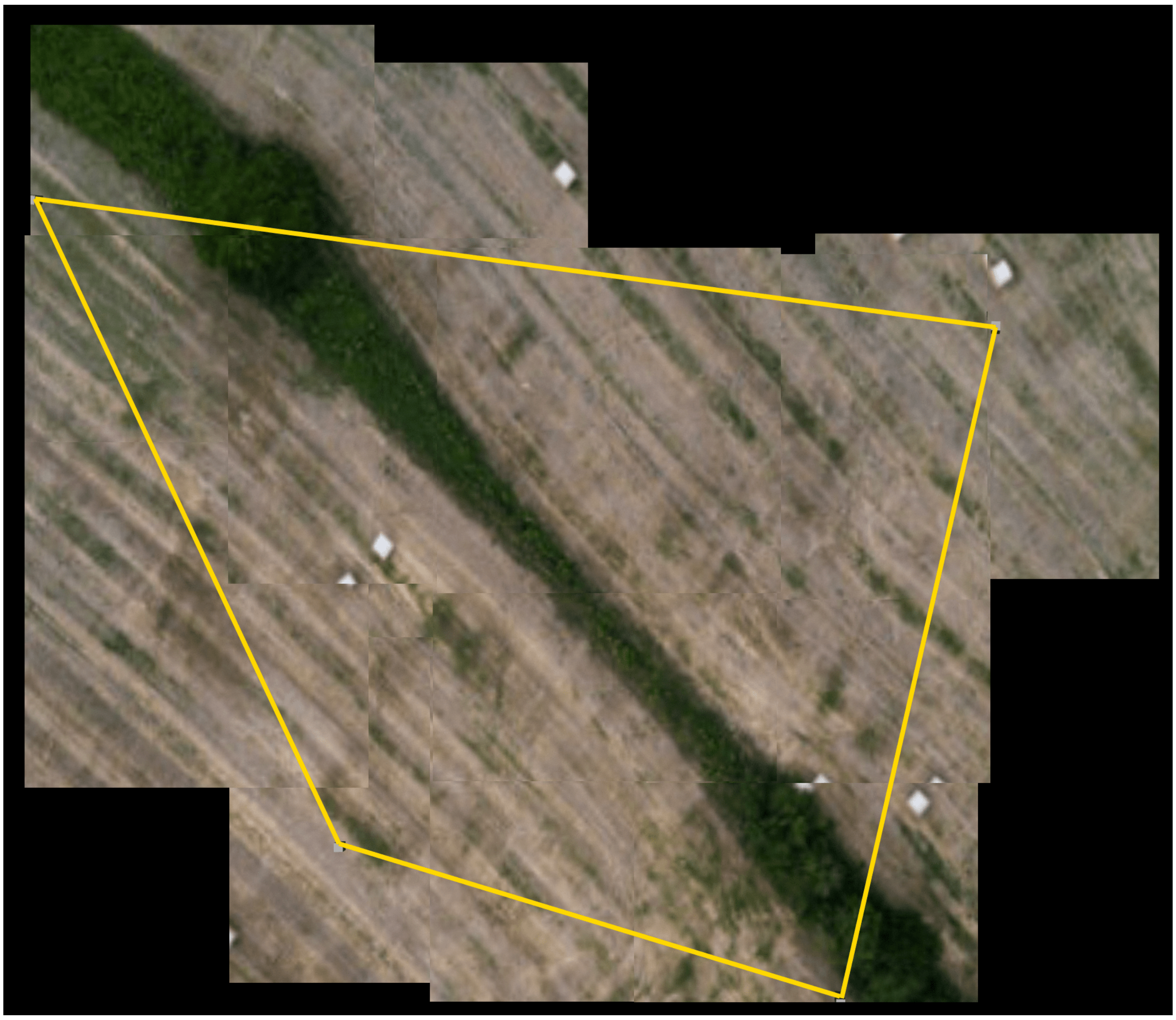


Figure 69: The result of the sensor based stitching of mapping by Wavefront algorithm.

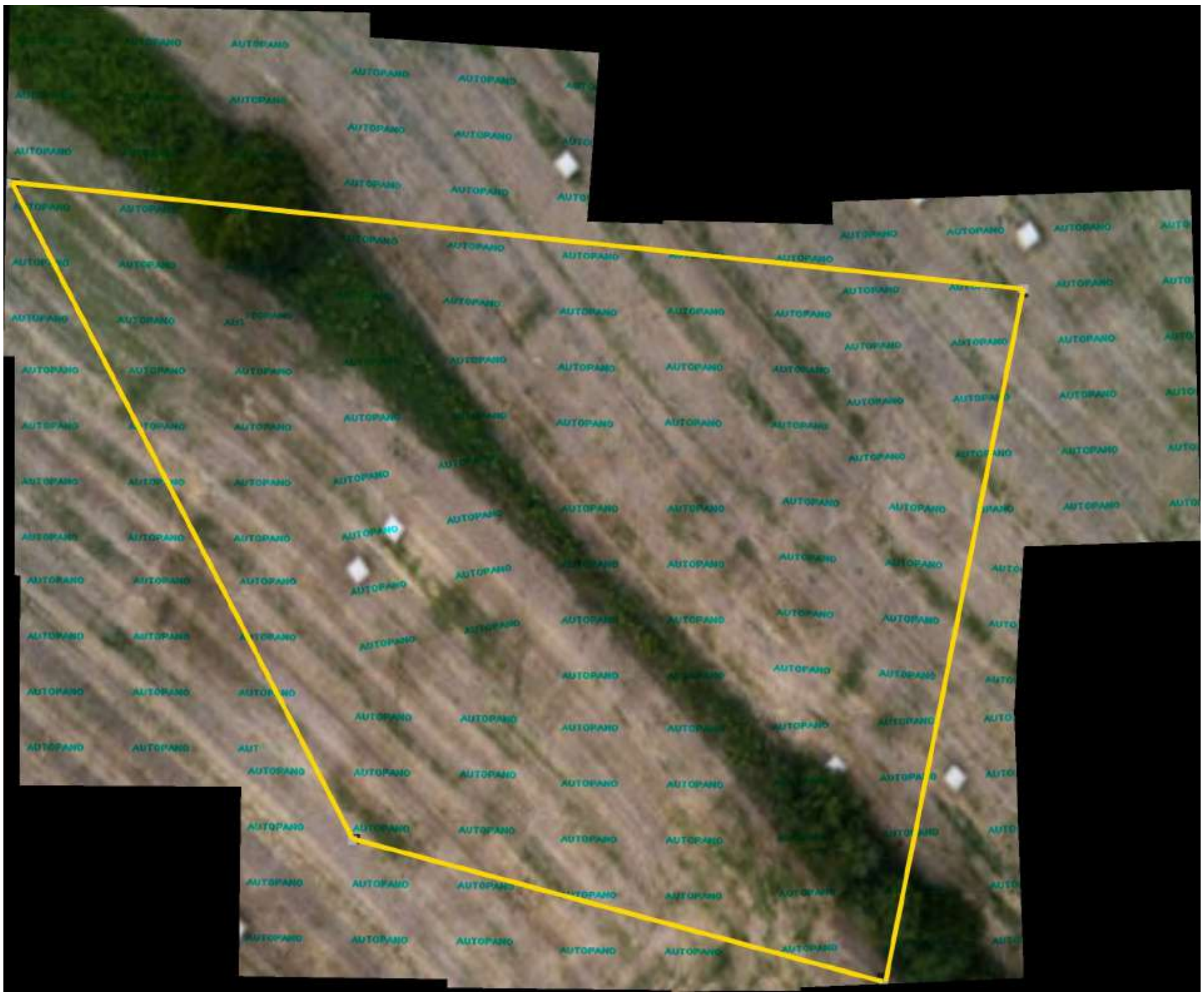


Figure 70: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Wavefront algorithm.

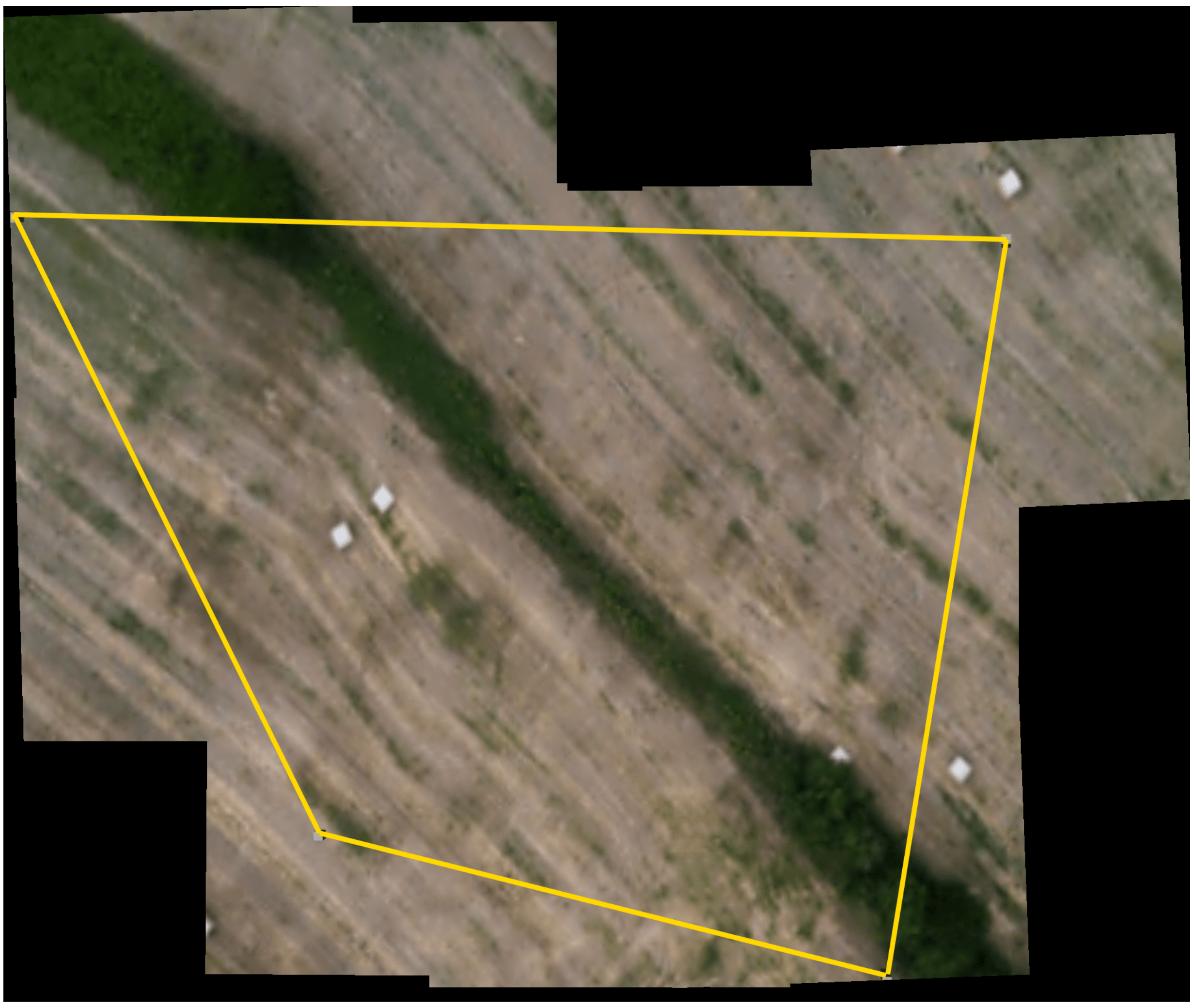


Figure 71: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Wavefront algorithm.

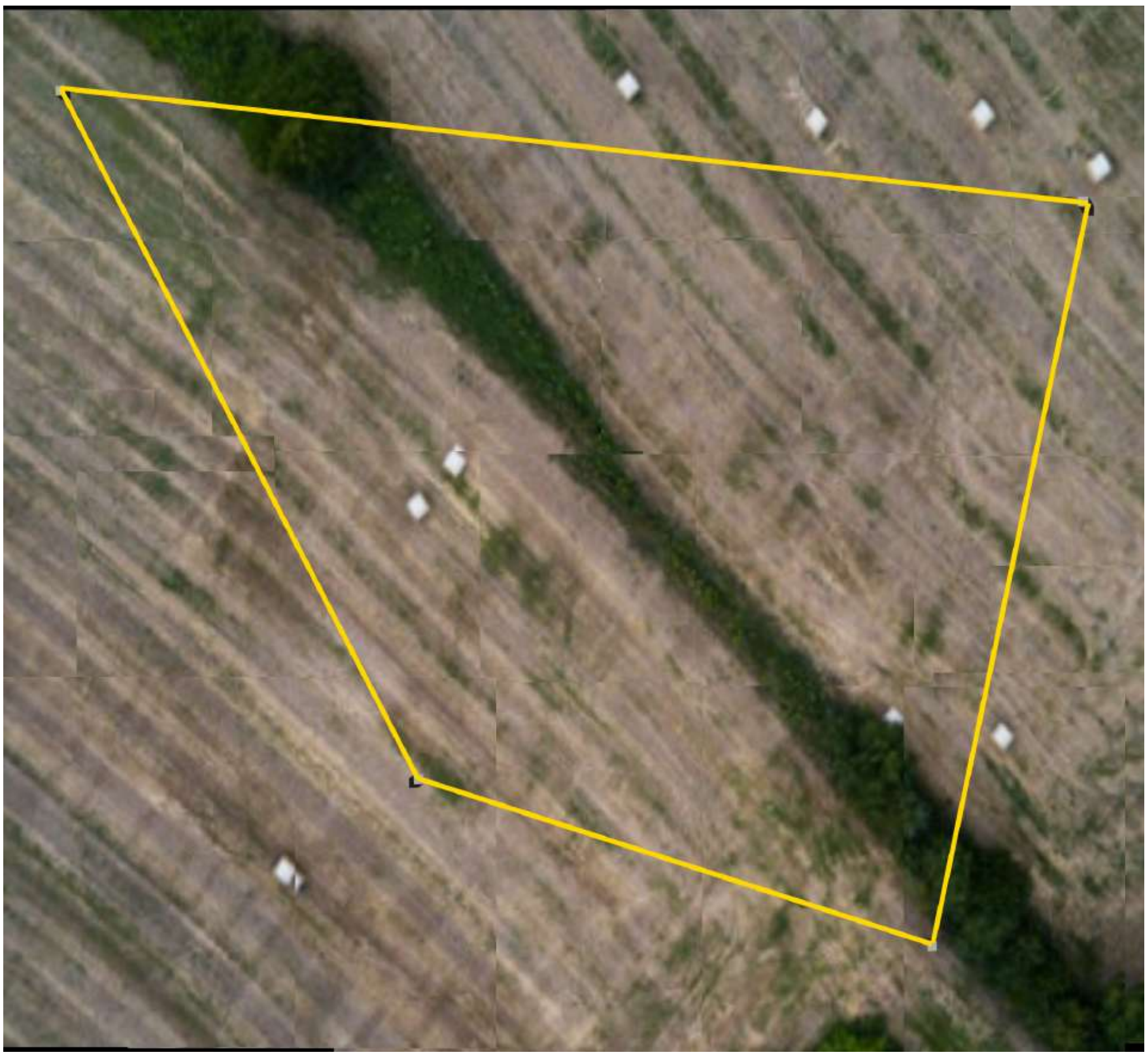


Figure 72: Result of the sensor based stitching of mapping by Zigzag algorithm

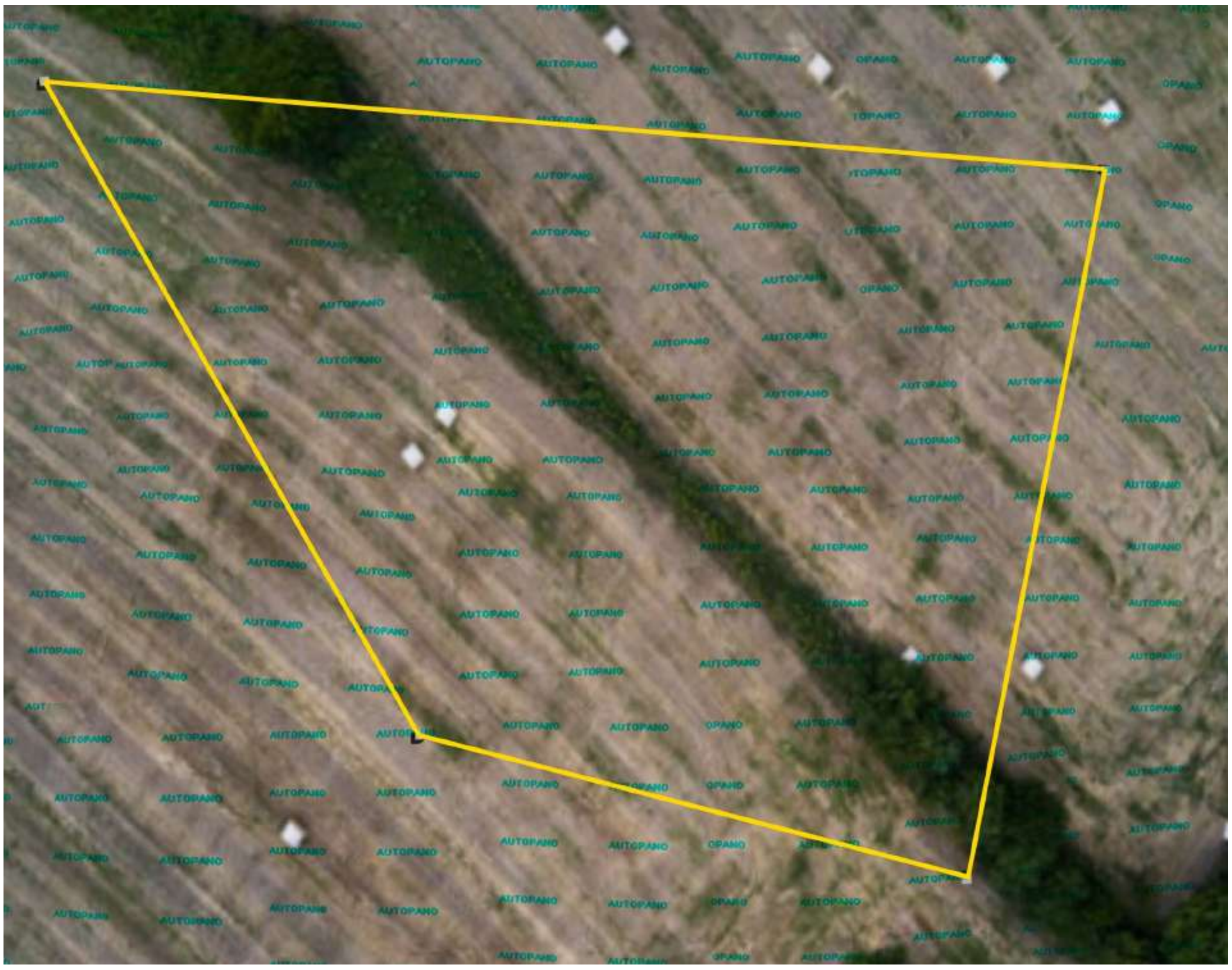


Figure 73: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Zigzag algorithm.

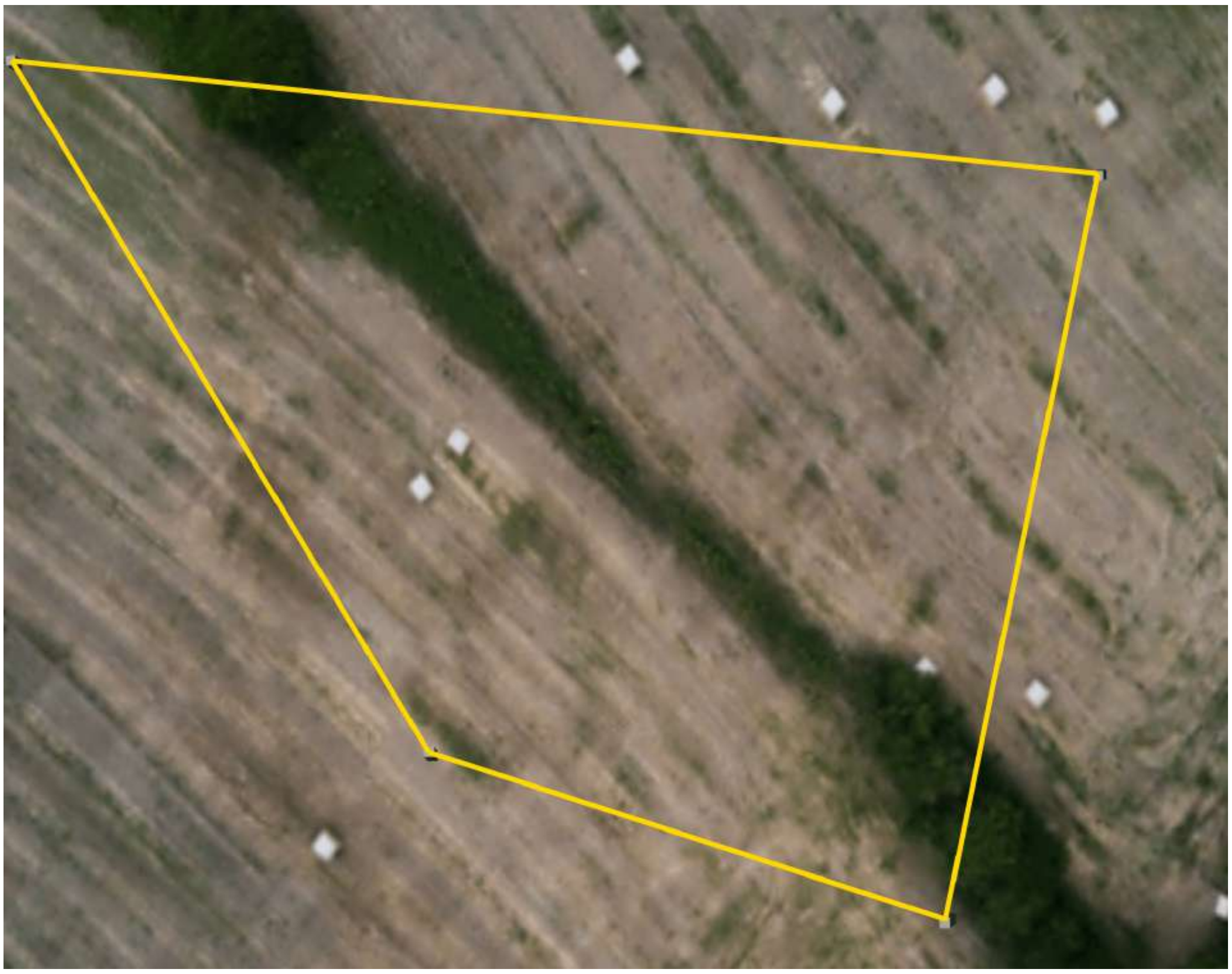


Figure 74: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Zigzag algorithm.

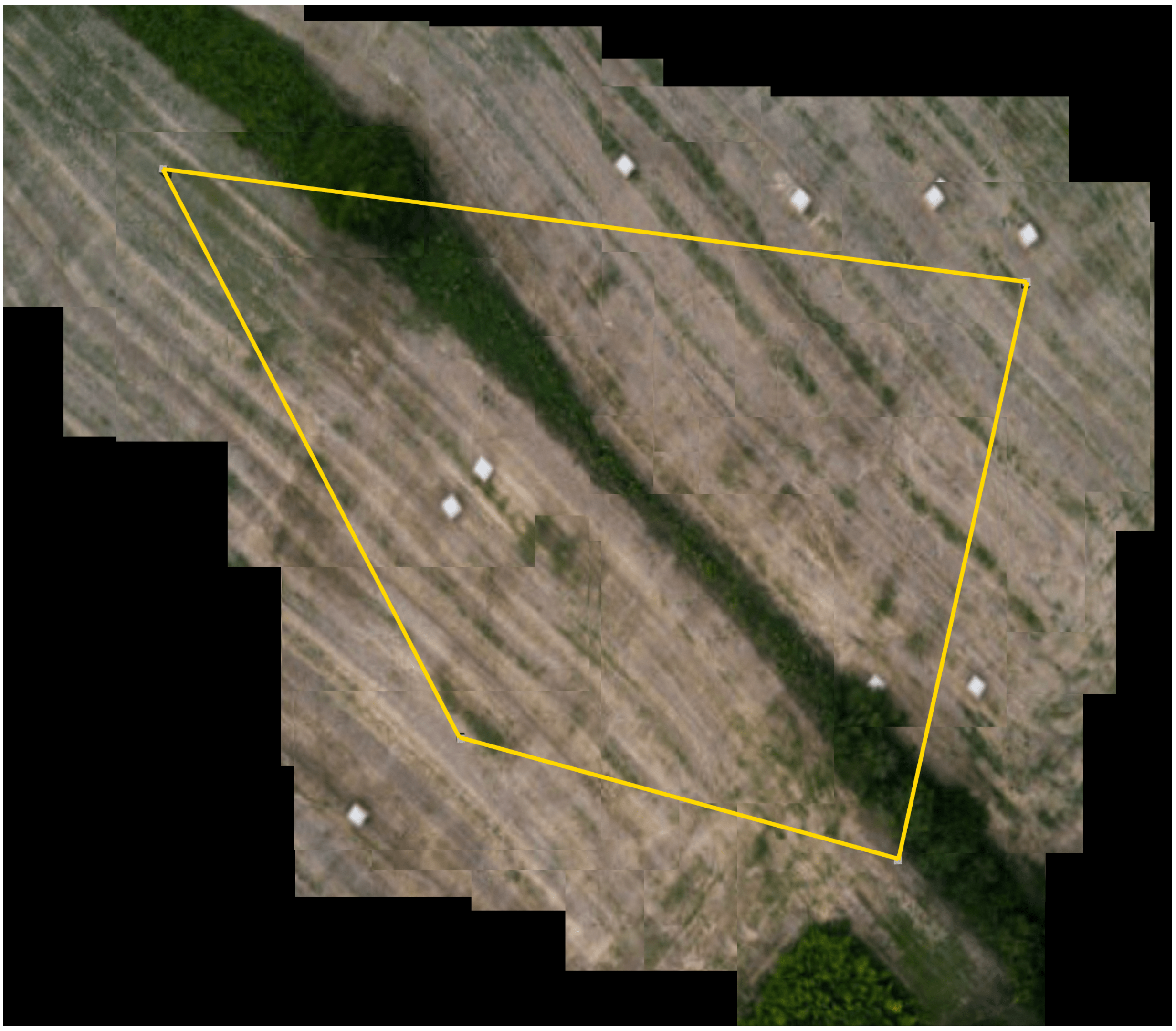


Figure 75: Result of the sensor based stitching of mapping by Trapezoidal algorithm

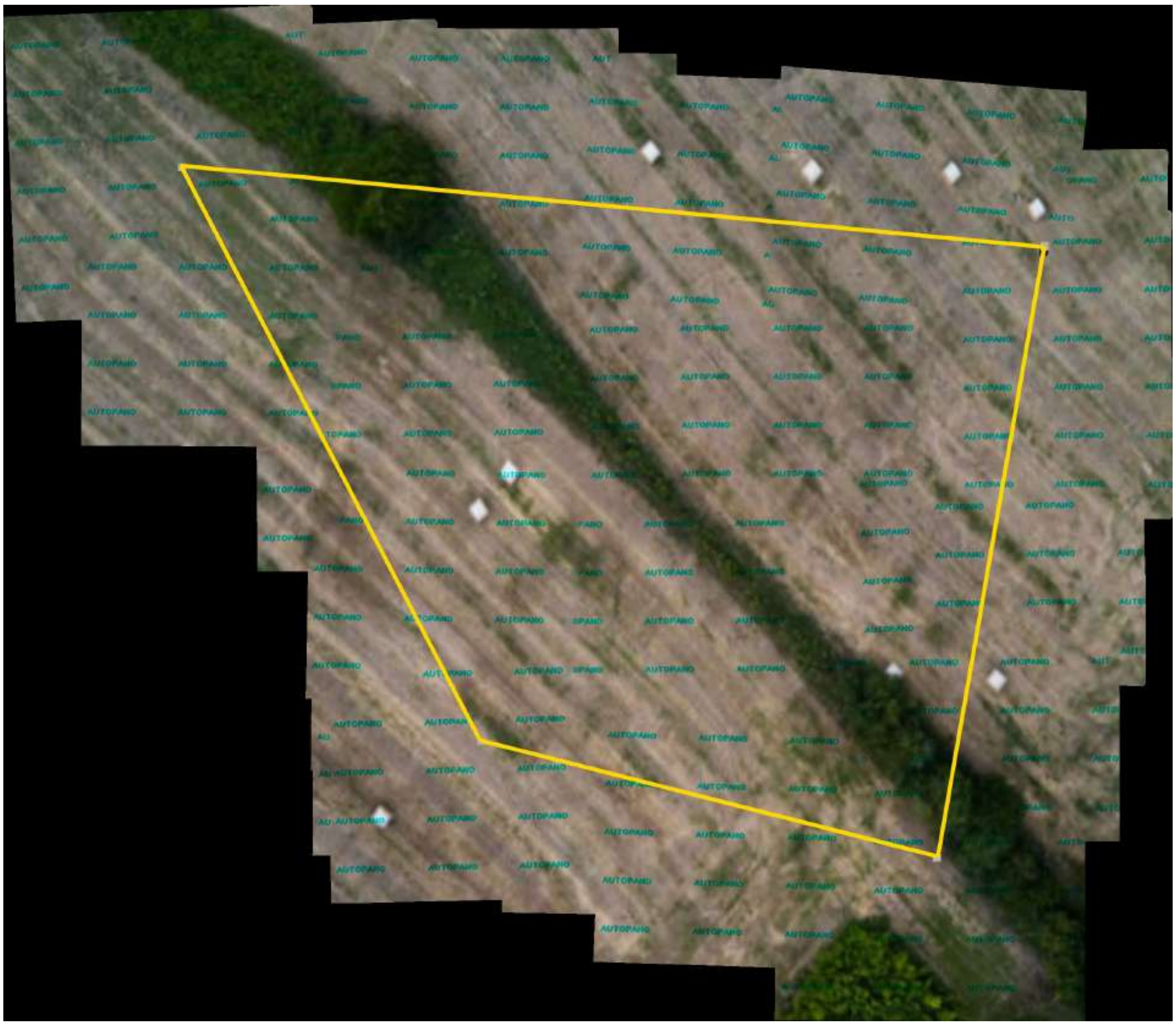


Figure 76: Result of the Autopano stitching algorithm applied on photos taken during mapping using the Trapezoidal algorithm.

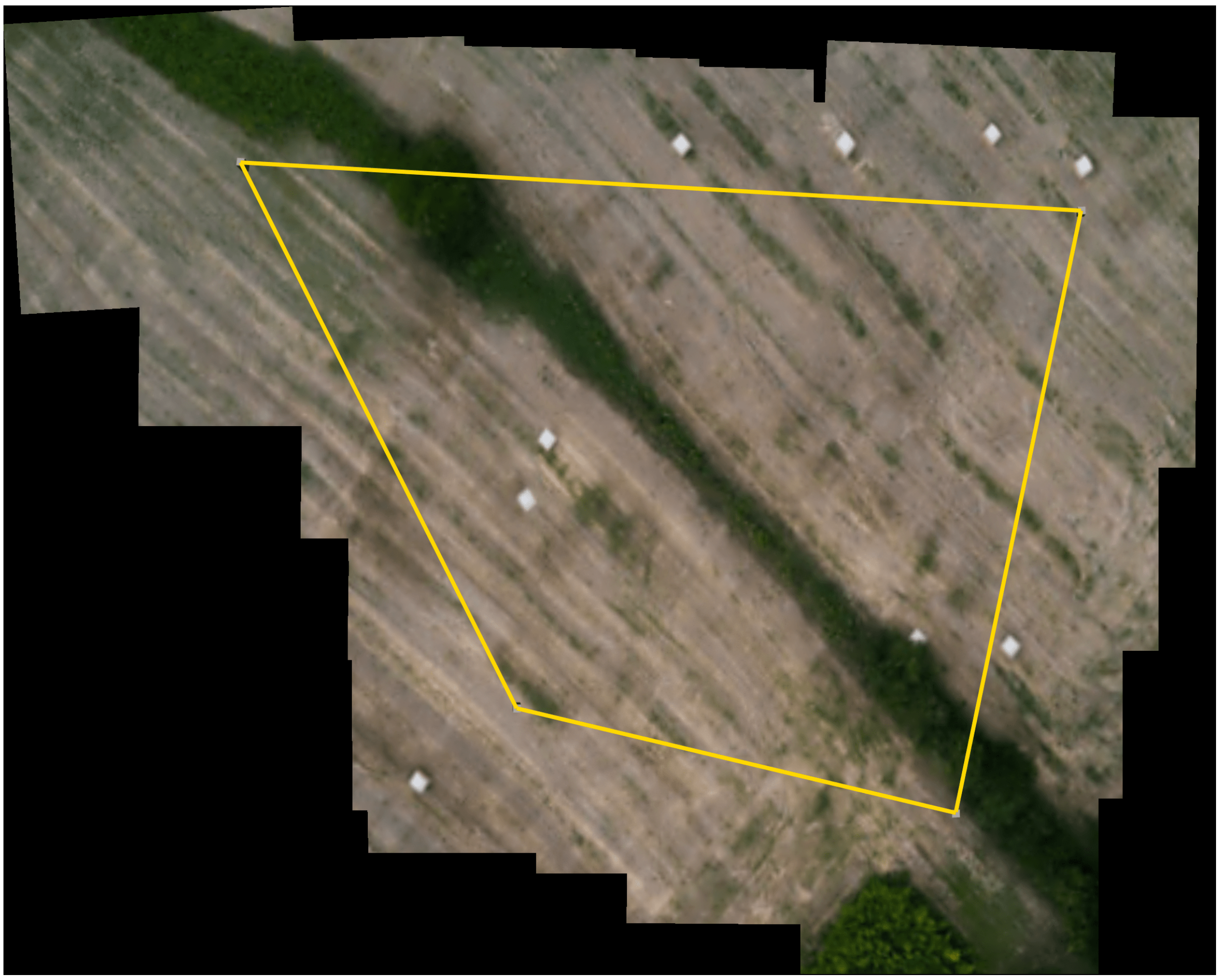


Figure 77: Result of the OpenCV stitching algorithm applied on photos taken during mapping using the Trapezoidal algorithm.

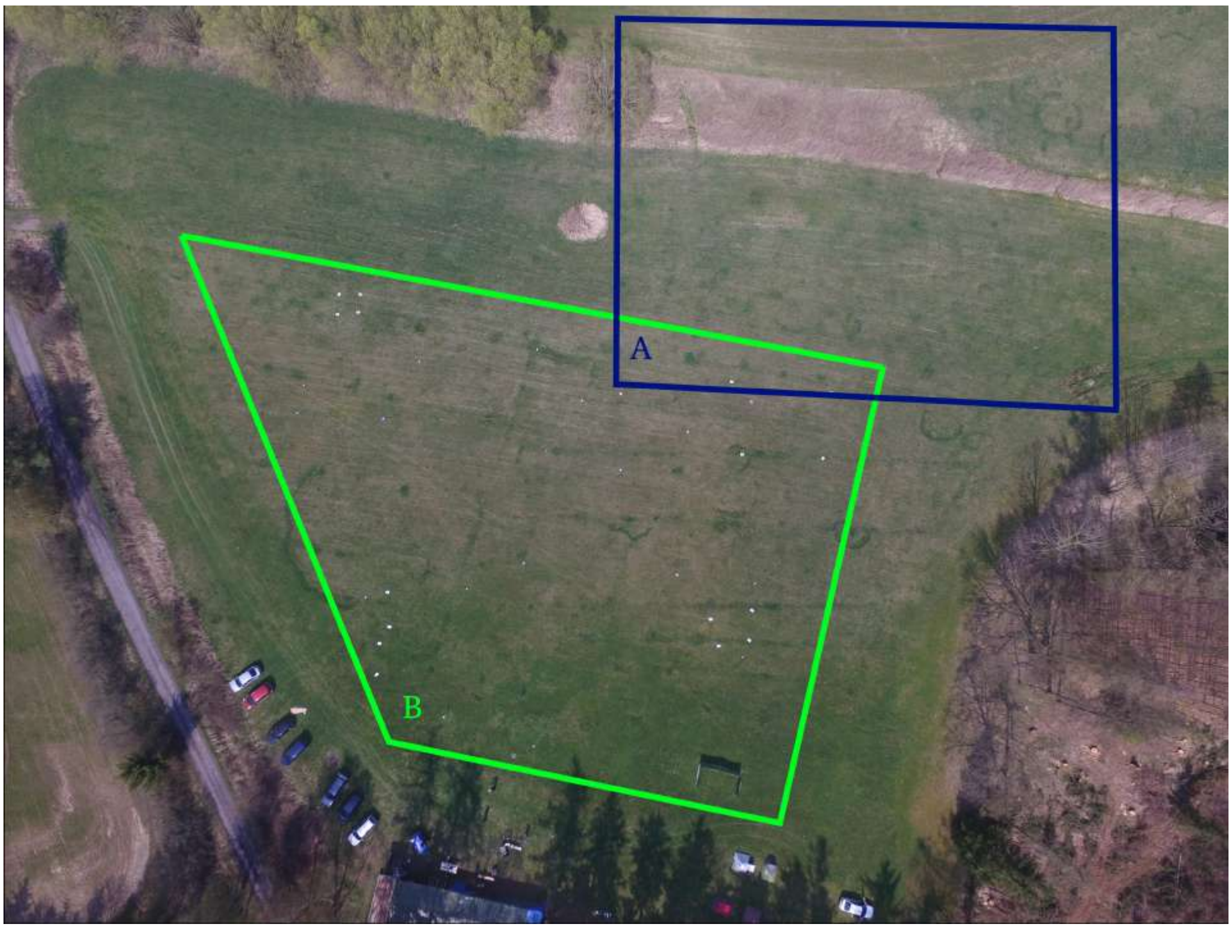


Figure 78: Picture of mapping areas in the first experiment taken from a DJI Phantom flying at 100m

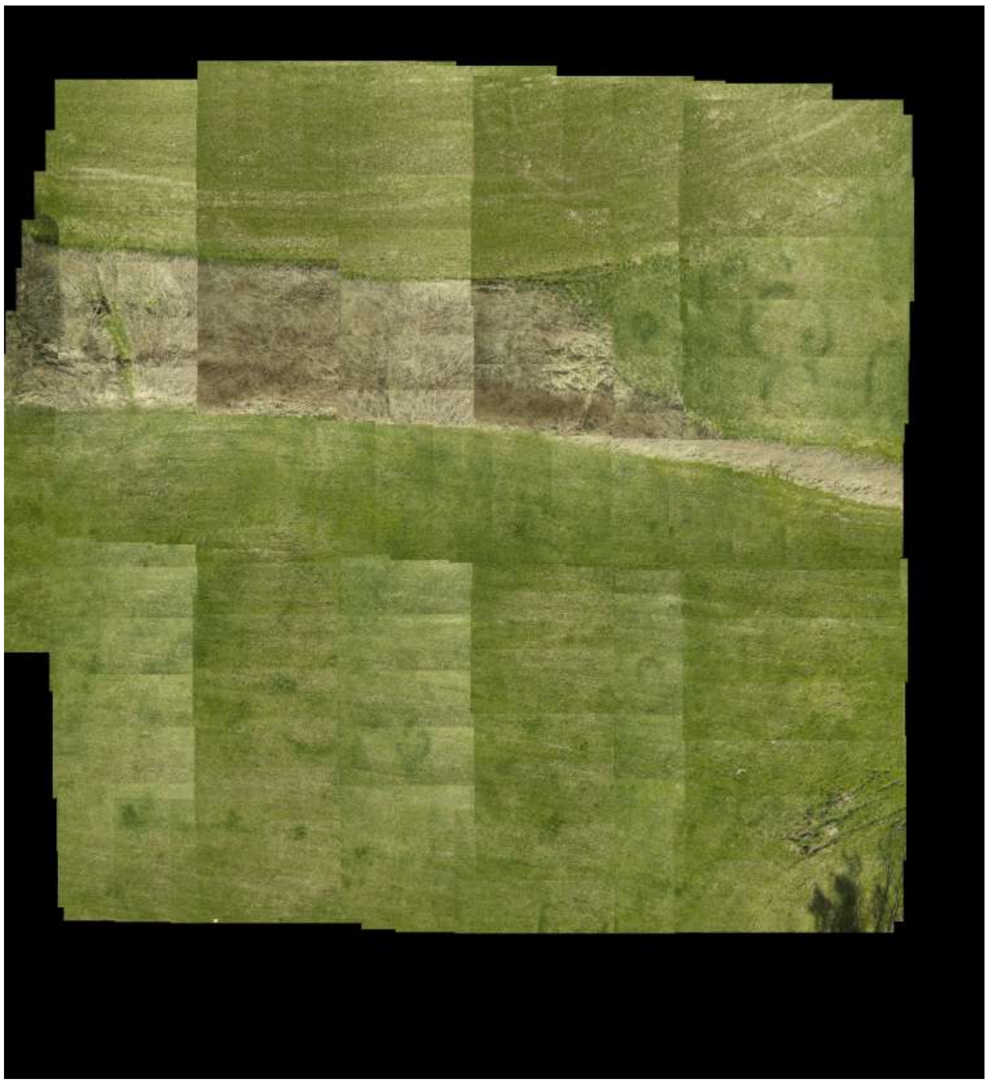


Figure 79: Demonstration of the Trapezoidal algorithm in a real system, during mapping an area from an altitude of 10m.

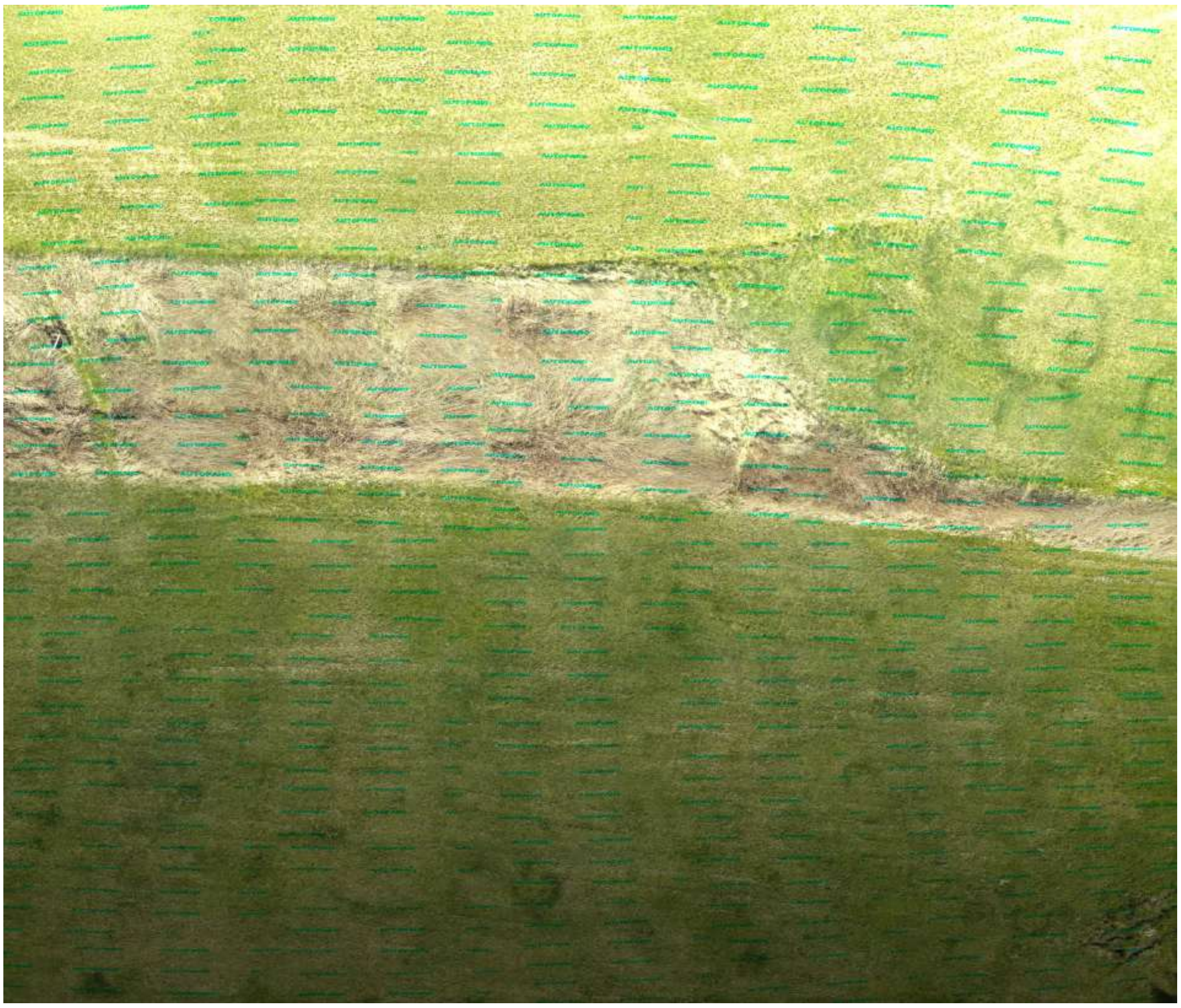


Figure 80: Result of the Autopano stitching software applied on a set of real photos collected using the trapezoidal algorithm.

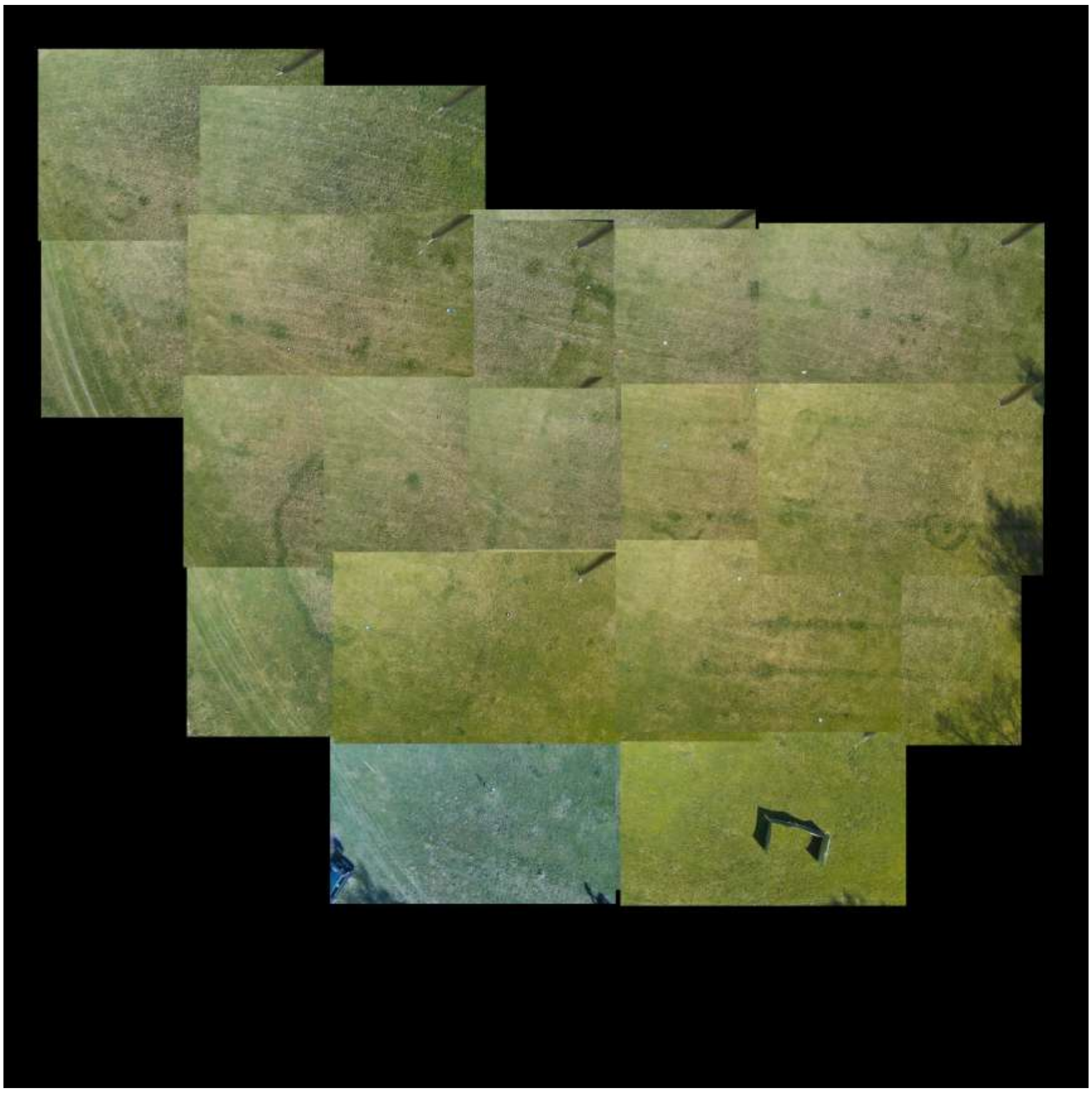


Figure 81: Demonstration of the Wavefront algorithm in a real system, while mapping an area from the altitude of 15m.

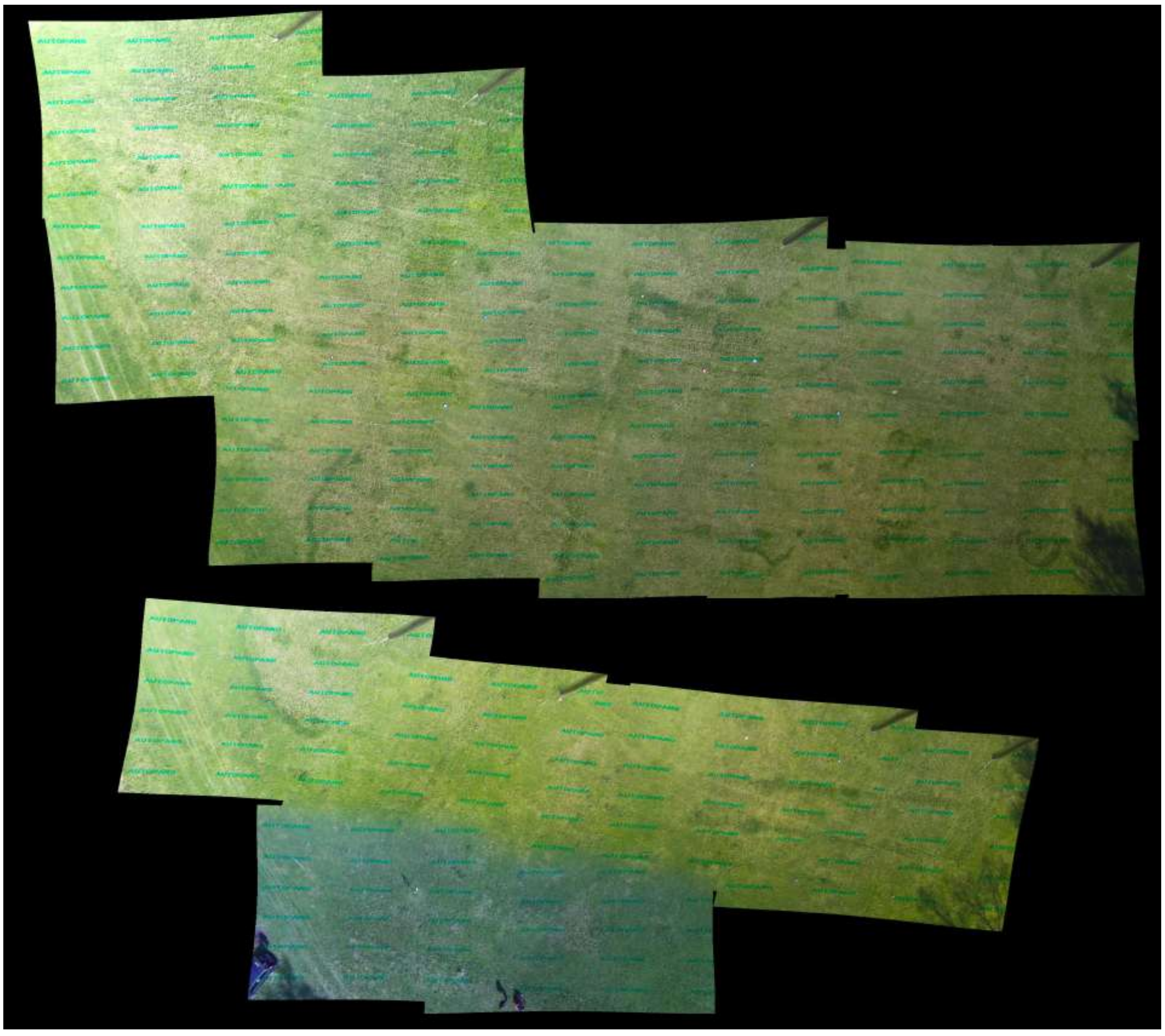


Figure 82: Result of the Autopano stitching software applied to real photos collected during mapping using the Wavefront algorithm.

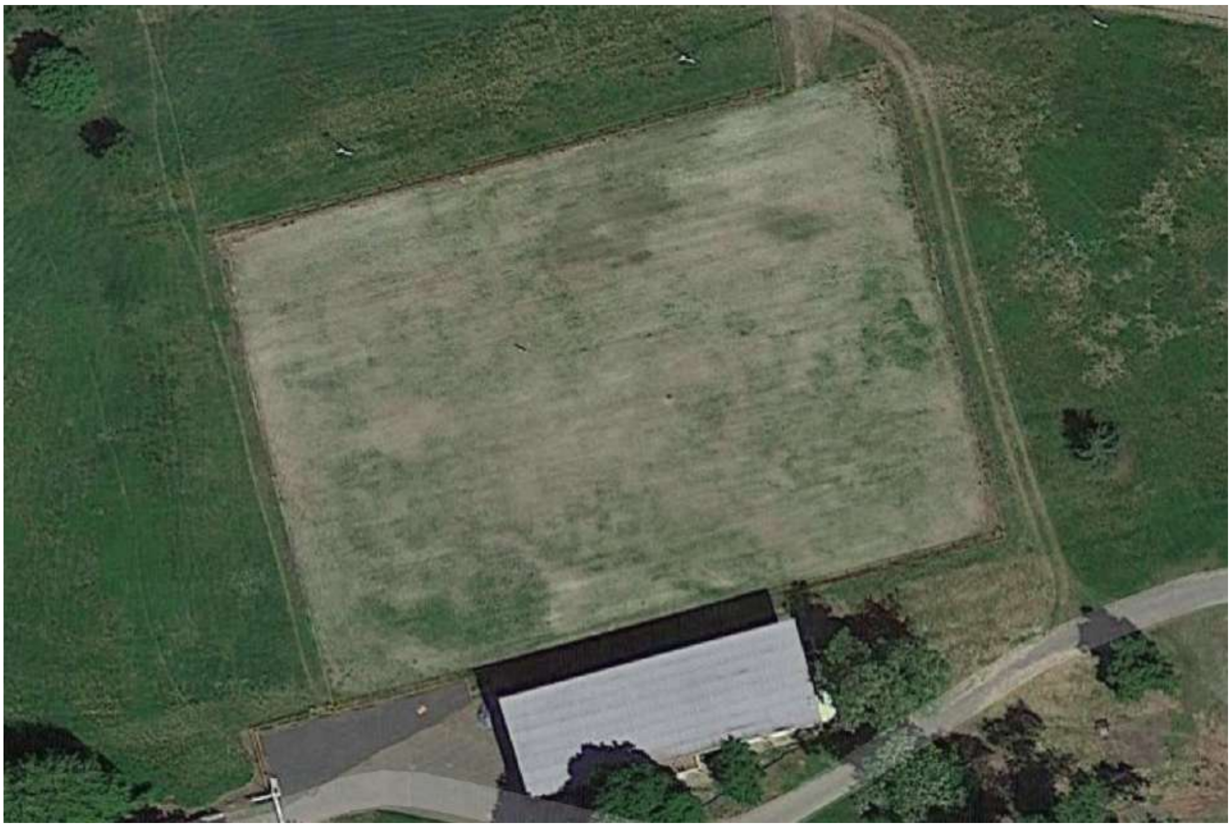


Figure 83: Satelite image of the area mapped in the second experiment

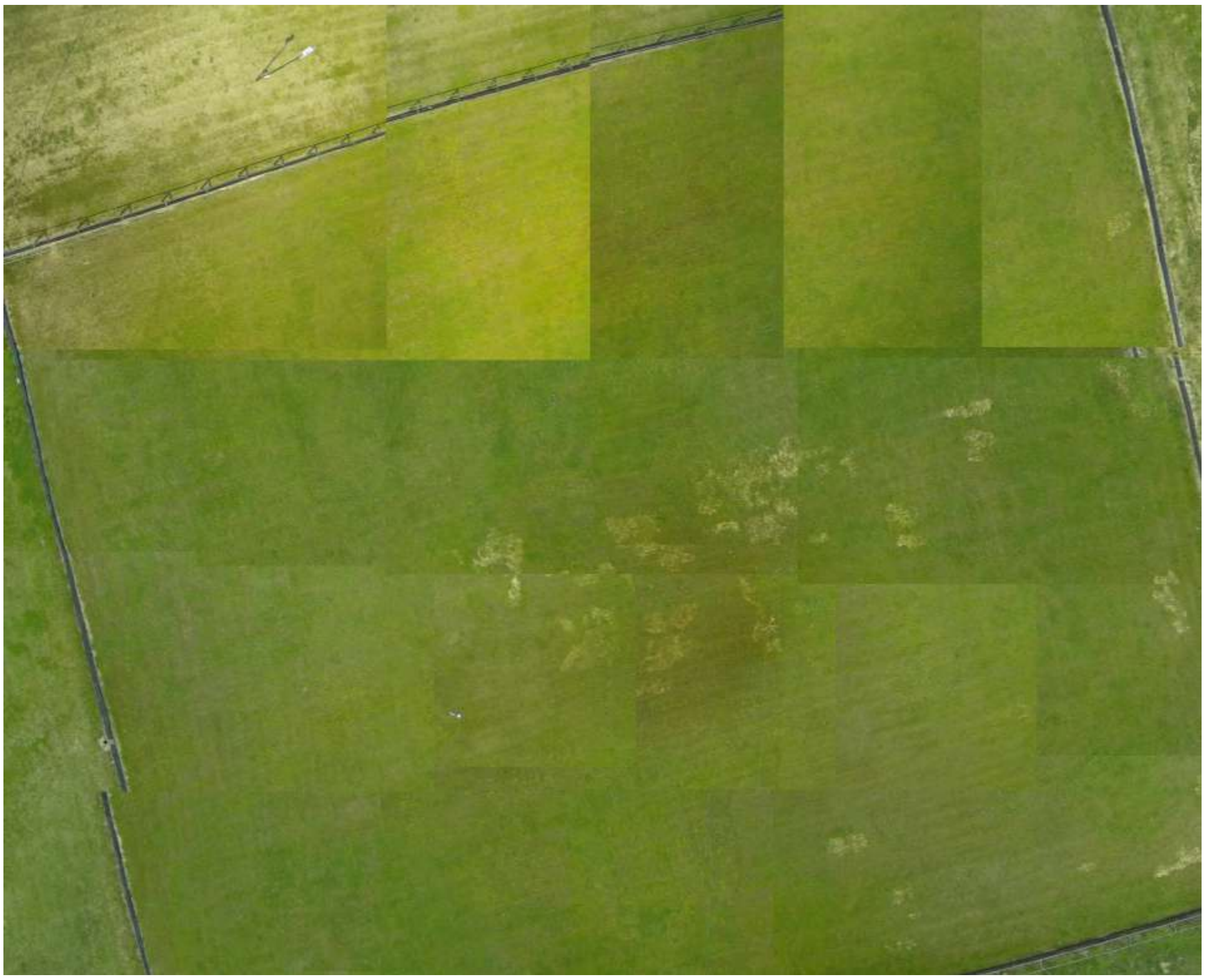


Figure 84: Demonstration of the ZigZag algorithm in a real system, while mapping an area from the altitude of 20m.

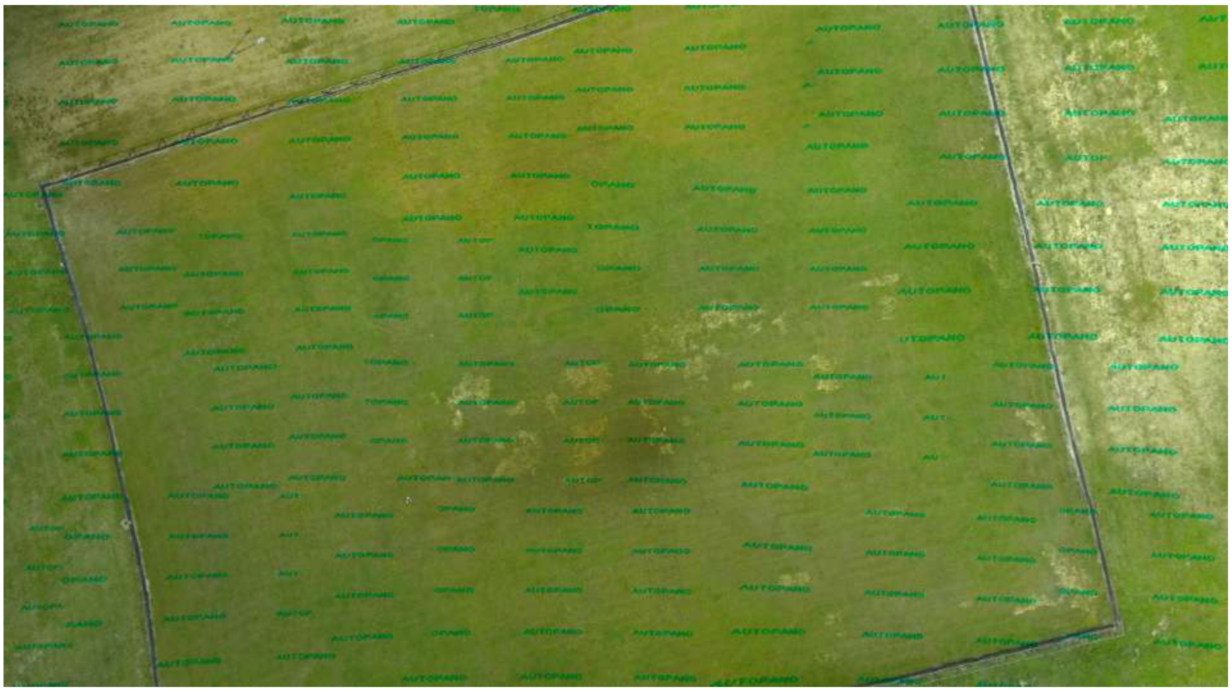


Figure 85: Result of the Autopano stitching software applied to real photos collected during mapping using the ZigZag algorithm.