Vision based hardware-software real-time control system for autonomous landing of an UAV

Krzysztof Blachut [1], Hubert Szolc [1], Mateusz Wasala [1], Tomasz Kryjak [2], and Marek Gorgon [1]

[1]Affiliation not available
[2]AGH University of Science and Technology

October 30, 2023

Abstract

In this paper we present a vision based hardware-software control system enabling autonomous landing of a mul-tirotor unmanned aerial vehicle (UAV). It allows the detection of a marked landing pad in real-time for a 1280 x 720 @ 60 fps video stream. In addition, a LiDAR sensor is used to measure the altitude above ground. A heterogeneous Zynq SoC device is used as the computing platform. The solution was tested on a number of sequences and the landing pad was detected with 96% accuracy. This research shows that a reprogrammable heterogeneous computing system is a good solution for UAVs because it enables real-time data stream processing with relatively low energy consumption.

Vision based hardware-software real-time control system for autonomous landing of an UAV

Krzysztof Blachut
AGH University of Science
and Technology Krakow, Poland
E-mail: kblachut@agh.edu.pl

Hubert Szolc
AGH University of Science
and Technology Krakow, Poland
E-mail: szolc@agh.edu.pl

Mateusz Wasala
AGH University of Science
and Technology Krakow, Poland
E-mail: wasala@agh.edu.pl

Tomasz Kryjak, *Senior Member IEEE*
AGH University of Science
and Technology Krakow, Poland
E-mail: tomasz.kryjak@agh.edu.pl

Marek Gorgon, *Senior Member IEEE*
AGH University of Science
and Technology Krakow, Poland
E-mail: mago@agh.edu.pl

***Abstract*—In this paper we present a vision based hardware-software control system enabling autonomous landing of a multirotor unmanned aerial vehicle (UAV). It allows the detection of a marked landing pad in real-time for a 1280 x 720 @ 60 fps video stream. In addition, a LiDAR sensor is used to measure the altitude above ground. A heterogeneous Zynq SoC device is used as the computing platform. The solution was tested on a number of sequences and the landing pad was detected with 96% accuracy. This research shows that a reprogrammable heterogeneous computing system is a good solution for UAVs because it enables real-time data stream processing with relatively low energy consumption.**

I. Introduction

Unmanned Aerial Vehicles (UAV), commonly known as drones, are becoming increasingly popular in commercial and civilian applications, where they often perform simple, repetitive tasks such as terrain patrolling, inspection or goods delivering. Especially popular among drones are the so-called multirotors (quadrocopters, hexacopters etc.), which have very good navigation capabilities (vertical take-off and landing, hovering, flight in narrow spaces). Unfortunately, their main disadvantage is high energy consumption. With the currently used Li-Po batteries the flight time is relatively short (up to several dozen minutes). Therefore, autonomous realization of the mentioned tasks requires many take-offs and landings to replace or recharge batteries.

The start of a multirotor, assuming favourable weather conditions (mainly lack of strong wind) and while the distance from other obstacles is preserved, is simple. Landing in a selected place, however, requires relatively precise navigation. If a tolerance of up to several meters is allowed, the GPS (Global Positioning System) signal can be used. Nevertheless, it should be noted that in the presence of high buildings the GPS signal may disappear or be disturbed. What is more, even under favourable conditions, the accuracy of the determined position is limited to approximately 1–2 m. Performing a more precise landing requires an additional system. There are primarily two solutions to this problem – the first is based on computer vision and the second uses a radio signal to guide the vehicle.

The main contribution of this paper is the proposal of a hardware-software vision system that enables precise landing of the drone on a marked landing pad. As the computational platform, a heterogeneous Zynq SoC (System on Chip) device from Xilinx is used. It consists of programmable logic (PL or FPGA – Field Programmable Gate Array) and a processor system (PS) based on a dual-core ARM processor. This platform allows to implement video stream processing with 1280×720 resolution in real time (i.e. 60 frames per second) in programmable logic, with relatively low power consumption (several watts). The processor facilitates communication with the drone controller and is responsible for the final part of the vision algorithm. To the best of our knowledge, this is the first reported implementation of such a system in a heterogeneous device.

The reminder of this paper is organized as follows. In Section II previous works on autonomous drone landing are briefly presented. Then, the multirotor used in the experiments, landing procedure and landing pad detection algorithm are described. In Section IV the details of the created hardware-software system are presented. Subsequently, in Section V, the evaluation of the designed system is discussed. The paper concludes with a summary and further research directions discussion.

II. Previous works

The topic of the autonomous landing with the use of visual feedback has been addressed in a number of scientific papers. In the following review we focus mainly on the landing on a static pad.

The authors of the work [1] reviewed different autonomous landing algorithms of a multirotor on both static and moving landing pads. They compared all previously used landing pad tags, i.e. ArUco, ARTag, ApriTag, tags based on circles and a "traditional" H-shaped marker. They also pointed out that landing in an unknown area, i.e. without a designated landing pad, is a challenging task. Then they analysed two types of

drone controllers during the landing phase: PBVS (Position-Based Visual Servoing) and IBVS (Image-Based Visual Servoing). The first compares drone's position and orientation with the expected values. The second one compares the location of feature points between the pattern and subsequent image frames. In the summary, the authors pointed out three main challenges related to this type of systems: development of a reliable vision algorithm with limited computing resources, improvement of a state estimation and appropriate modelling of the wind in the control algorithm.

In the work [2] the authors proposed an algorithm for landing of an unmanned aerial vehicle on a stationary landing pad. It was used when replace or re-charge the drone's battery was necessary during high-voltage line inspection. For the detection of the landing pad they used thresholding, median filtration, contour extraction, determination of geometrical moments and an SVM classifier. In addition, the Extended Kalman Filter (EKF) was used to determine the position of the drone. It processed data from inertial sensors (IMU – Inertial Measurement Unit), radar and vision system. During tests, the drone approached the landing pad with a GPS sensor and then switched to the vision mode, in which the landing pad was detected with a camera. Out of 20 sequences, 14 ended with a position error of less than 10 cm, and the remaining ones below 20 cm. At the same time, in 12 tests the orientation error was below 10 degrees, while in the remaining ones below 20 degrees. As a computing platform, the authors used Raspberry Pi. They obtained a processing time of a single frame of 0.19 s, that is about 5 frames per second, which is sufficient for slow-moving drones (no information about the camera resolution was provided).

The authors of the work [3] presented an algorithm for controlling autonomous landing of an unmanned aerial vehicle on a stationary T-shaped landing pad. The proposed vision algorithm was based on so-called image key points (feature points). The algorithm consisted of colour to greyscale transformation, thresholding, morphological operations, contour extraction (by using the Laplace operator) and matching the polygon to the detected points. Based on this, 8 angular points were determined, which were used to find the position of the drone corresponding to the landing pad. The algorithm worked in real-time as the processing of one frame with a resolution of 480×320 pixels took about 50 ms. The authors did not state on what platform the algorithm was implemented.

The article [4] presents the possibility of using reinforcement learning to generate the control necessary for autonomous landing. In the learning phase a simulation environment was used. Real experiments were also carried out on a hexacopter. It was equipped with a custom computer based on a TI microcontroller, NVIDIA Jetson TX2 platform and HD camera. The authors reported a single frame processing time of 200 ms (5 fps). In addition, they mentioned that this was one of the reasons for the observed oscillations.

The article [5] presents a complex vision algorithm for landing pad detection. A marker in the form of the letter H placed inside a circle with an additional smaller circle in the centre was used. The algorithm operation was divided into three phases that were performed depending on the distance between the drone and the landing pad. In the first one, the outer circle, then the letter H, and finally the middle circle were detected. The authors devoted much attention to the reliability of the algorithm, providing correct operation in the conditions of partial occlusion and shading of the marker. Odroid XU4 computing platform with a Linux operating system was used and the OpenCV library was applied. The source of the video stream was a camera with a resolution of 752×480 pixels. Processing of 12 to 30 frames per second was obtained depending on the phase of the considered algorithm.

In the work [6] the authors proposed the RTV (Relative Total Variation) method to filter the unstructured texture to improve marker (a triangle inside a circle) detection reliability. They also used median filtering, Canny edge detection and polygon fitting by the Ramer-Douglas-Peucker algorithm. In addition, they proposed a method of integrating data obtained from the GPS sensor and vision algorithm. There was no information about used computing platform, camera resolution or the possibility of performing calculations in real-time.

Summarizing the review, it is worth noting that in most works vision systems were able to process just a few frames per second. The authors of [4] pointed out that that this could be the reason the drone was oscillating during landing. Moreover, in [1] one of the mentioned challenges was the use of an energy-efficient platform to perform calculations in real-time. In this work, we claim that the use of a heterogeneous computing platform can address the both mentioned issues.

III. The proposed automatic landing algorithm

In this section we present the used hardware setup, the proposed automatic landing procedure and the landing pad detection algorithm.

A. Hardware setup

In this research we used a custom multirotor (hexacopter) built from the following components:

- frame: DJI F550,
- propeller: reinforced, with the designation 9050, i.e. with a propeller diameter equal to 9.0" (22.86 cm) and 5.0" pitch (12.7 cm),
- engines: DJI 2312 / 960KV controlled by 420 LITE controllers,
- power supply: four-cell Li-Po battery with a nominal voltage of 14.8V (maximum 16.8V) and with a capacity of 6750 mAh,
- radio equipment: FrSky Taranis X9D Plus,
- receiver: FrSky X8D,
- autopilot: 3DR Pixhawk.

The drone was adapted to the considered project. We added a heterogeneous computing platform Arty Z7 with Zynq-7000 SoC device. We connected an Yi Action camera to it (as the source of a 1280×720 @ 60 fps video stream), a LiDAR for measuring altitude above the ground level (LIDAR-Lite v3 device, SEN-14032) and a radio transceiver module for

Fig. 1: The used hexacopter

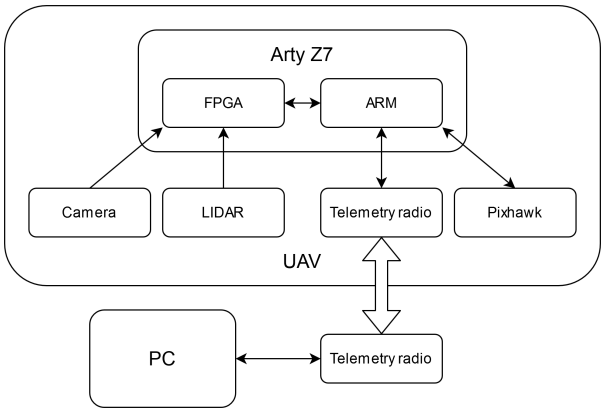


Fig. 2: A simplified scheme of the proposed system

wireless communication with the ground station (3DR Mini V2 telemetry modules). Moreover, the Arty Z7 board was connected to the Pixhawk autopilot via UART (Universal Asynchronous Receiver-Transceiver). The drone is shown in Figure 1, while the simplified functional diagram of the proposed system is presented in Figure 2.

B. The proposed automatic landing procedure

In the initial situation, the drone flies at an altitude of about 1.5–2 m above the ground level and the landing pad is in the camera's field of view. Until then, the vehicle is piloted manually or in another automated mode (using GPS-based navigation). After fulfilling the conditions mentioned above, the system is switched into autonomous landing mode (currently implemented by a switch on the remote controller).

In the first phase, a rough search for the landing pad is executed. After finding it, the drone autonomously changes its position so that the centre of the marker is around the centre of the image. In the second phase, the altitude is decreased to approx. 1 m and the drone's orientation relative to the landing pad is additionally determined. Based on this information, the vehicle is positioned accordingly.

In the last phase, the landing is carried out. The altitude above the landing pad is measured using the LiDAR sensor. The position of the pad is constantly determined and corrections, if required, are made. The drone is lowered and the engines are turned off after landing.

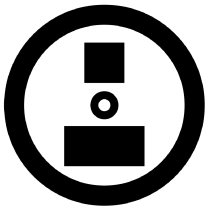


Fig. 3: The used landing marker

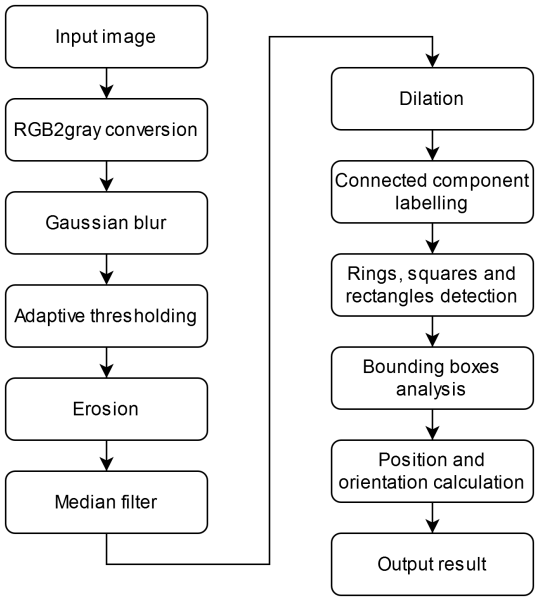
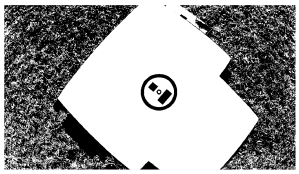


Fig. 4: Simplified dataflow of the vision algorithm

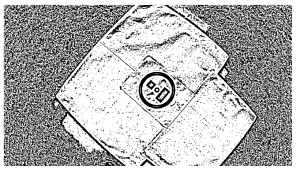
C. Landing spot detection algorithm

The key element of the proposed autonomous landing system is the landing pad detection algorithm. Based on the analysis of previous works and preliminary experiments, we decided to choose the landing pad marker as shown in Figure 3. It consists of a large black circle, inside which we placed three other geometrical figures – a square, a rectangle and a small circle. The largest of the figures made it possible to roughly detect the landing pad and determine the approximate centre of the marker. The small circle was used in the case of a low altitude of the drone i.e. when the large circle was no longer fully visible in the image. It also allowed to improve the accuracy of the marker centre detection. The purpose of placing the square and the rectangle was to determine the orientation of the entire marker. Therefore, we are able to land the drone in a certain place with a given orientation. Here we should note, that the used shapes are relatively easy to detect in a vision system implemented in reconfigurable logic resources.

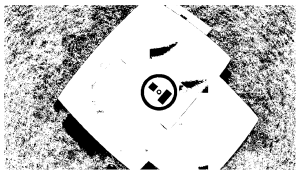
A simplified block diagram of the entire vision algorithm is depicted in Figure 4. The application was prototyped in the C++ language with the OpenCV library version 4.1.0, which includes most basic image processing operations. We provide the source code of our application [7]. This allowed to compare



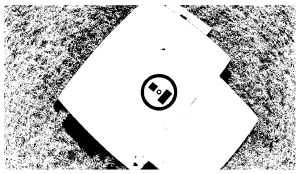
(a) Global thresholding



(b) Local thresholding



(c) Adaptive in windows



(d) Adaptive with interpolation

Fig. 5: Comparison of thresholding methods

different approaches, select parameters, etc. The obtained results could then be used during hardware implementation on the target platform.

Firstly, the recorded image is converted to greyscale. Then a standard Gaussian blur filtering is applied with a 5×5 kernel. In the next step adaptive thresholding is used. As a result of its use, the segmentation of the landing pad works well in different lighting conditions. The input image is divided into non-overlapping windows of 128×128 pixels. For each of them, the minimum and maximum brightness is determined. Then thresholding is performed. The threshold is calculated according to Equation (1).

$$th = 0.25 \cdot (max - min) + min \tag{1}$$

where:

th – local threshold for the window,
max – maximum pixel brightness in the window,
min – minimum pixel brightness in the window.

The second stage of adaptive thresholding is the bilinear interpolation of the threshold's value. For each pixel, its four neighbours are determined (for pixels on the edges two neighbours, while in the corner just one) and the binarization threshold is calculated. As a result, the output binary image is smoother and the boundaries between the windows are not visible. This is presented in Figure 5d.

It is worth noting that alternative binarization approaches were also considered, i.e. based on a global threshold – Figure 5a, "fully local" (the threshold for each pixel calculated on the basis of its local neighbourhood) – Figure 5b, and adaptive in non-overlapping windows but without interpolation Figure 5c. Figure 5 presents a comparison of the described above approaches performed on a sample image. It can be concluded that the adaptive interpolation method is the best choice for the considered system. It provides very good results and works correctly in case of uneven illumination.

Fig. 6: Sample image after filtration

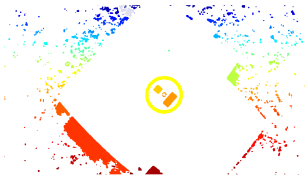


Fig. 7: Sample result of the CCL module

The next steps are few simple context filtering operations used to remove small objects. At first, erosion with a 3×3 kernel is applied. Then median filtering with a 5×5 window is performed, which helps to remove single outliers. Finally, dilation with a 3×3 kernel is used in order to keep the original dimensions of the remaining objects. An example result obtained after performing these filtrations is shown in Figure 6.

The next step is connected component labelling. An example result is shown in Figure 7. As an output: area, coordinates of the bounding box and the centroid for each object are obtained. These values are used to calculate specific parameters, such as the size of the bounding box, its shape or the ratio of pixels belonging to the object to the total number of pixels inside it. The analysis of these parameters allowed us to determine a set of conditions that distinguishes the circles, squares and rectangles from other objects. For more details please refer to the provided source code [7]. In addition, the expected sizes of each of the mentioned figures are determined using altitude data provided by the used LiDAR, as their dimensions in pixels depend on the distance from the camera. The main motivation for this approach was to reduce the number of false detections.

To accurately detect the entire marker, not just individual shapes, we performed a detailed analysis of the bounding boxes. The square, the rectangle and the small circle are considered as correctly detected if their bounding boxes are inside the bounding box of the large circle. This approach has significantly reduced the number of false detections.

Finally, the location and orientation of the detected marker is determined (cf. Figure 8). The centroids of the square and rectangle are used to calculate the orientation. Firstly, the distance between these points is determined and compared to the diametre of the large circle to avoid incorrect detections. If that distance is in the appropriate range, the ratio of the difference between them along the Y axis to the corresponding difference along the X axis is specified. Values calculated in this way enable to determine the angle using the $\arctan$ function.

Then the centroids of the square and the rectangle are used to calculate the position of the marker on the image. The average values of the two mentioned centroids proved to be more reliable than the centroid of the big circle, especially in case of incomplete marker or low altitude. However, the prior detection of the big circle is crucial as the squares and

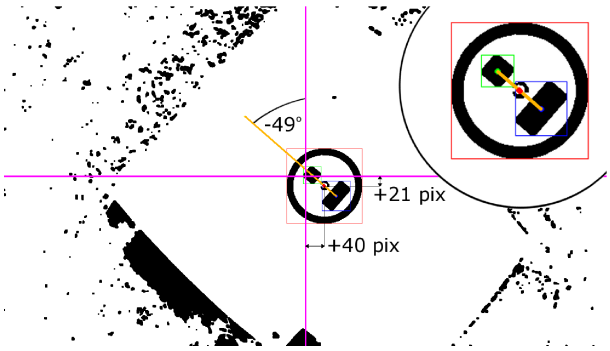


Fig. 8: Sample result of the proposed vision algorithm. The centre of the camera field of view is indicated by two crossing magenta lines. The bounding box surrounding the circle is marked in red, the square in green and the rectangle in blue. The red dot marks the centre of the marker and the orange line marks its orientation. The small circle is not marked by its bounding box, because it is too small to be detected correctly. In the top right corner of the image the marker is enlarged for better visualization. The obtained drone position and orientation are also presented.

rectangles were analysed only inside it. That means all three figures are necessary to estimate the position and orientation of the marker. At low altitude, when the big circle is not entirely visible, the centroid of the small circle is used as the location of the landing pad centre. The analysis of the image shown in Figure 8 was used in the example below to calculate the position and orientation of the drone relative to the landing pad. The following results were obtained:

- horizontal distance from the centre of the image: $+40$ pixels
- vertical distance from the centre of the image: $+21$ pixels
- deviation from the fixed marker orientation: $-49°$

After taking into account the current drone altitude and known dimensions of the marker, the calculated values were converted into centimetres. The following results are obtained:

- horizontal offset from the centre of the image: 6.2 cm
- vertical offset from the centre of the image: 3.3 cm

IV. HW/SW Implementation

We implemented all necessary components of the described algorithm in a heterogeneous system on the Arty Z7 development board with Zynq SoC device. All image processing and analysis stages are implemented in the programmable logic of the SoC. It receives consecutive frames from the camera, performs the required image preprocessing operations i.e.: conversion from RGB to greyscale, Gaussian low-pass filtration, median filtering, erosion, dilation and two more complex image analysis algorithms: connected component labelling (CCL) from our previous work [8] and adaptive thresholding. Finally, it sends the results to the processing system (ARM processor) via AXI interface. Moreover, in the prototyping phase the image processing results were transmitted via HDMI and visualized on a temporarily connected LCD screen.

TABLE I: PL resource utilization

Resource	System
LUT	14897 (28.00%)
FF	21368 (20.08%)
BRAM	24 (17.14%)
DSP	25 (11.36%)

The adaptive thresholding algorithm, mentioned in Section III-C, consists of two stages. The first one finds the minimum and maximum values in 128×128 windows and calculates the appropriate thresholds, according to Equation 1. The raw video stream does not contain pixel coordinates, but only pixel data and horizontal and vertical synchronization impulses, which is why the coordinates need to be calculated in additional counters implemented in programmable logic. In the second stage, local thresholds are obtained based on the values in adjacent windows (4, 2 or 1). It should be noted that thresholds computed for frame $N-1$, are used on frame N. This approach, which does not require image buffering, is justified by the high sampling frequency of the camera and thus small differences in brightness of pixels on subsequent frames.

In addition, we use the programmable logic part to supervise the altitude measurement using LiDAR. We use a simple state machine to control the LiDAR in PWM (Pulse Width Modulation) mode. It continuously triggers the distance measurement and reads its result. We implemented all of these modules as separate IP cores in the Verilog hardware description language.

For the processing system, i.e. the ARM Cortex-A9 processor, we developed an application in the C programming language. Its main task is to fuse data from the vision system and LiDAR and send appropriate commands to the Pixhawk controller via the MAVLink 2.0 protocol using UART. In particular, it filters the objects present in the image using data obtained from the CCL module, searches for proper geometric shapes and determines the position of the drone relative to the landing pad[1]. In addition, the application controls the radio module for communication with the ground station. Thanks to this, we can remotely supervise the algorithm. The application also uses interrupts available in the processing system, to correctly read all input data streams. In this manner, it can stop the algorithm execution at any time, especially when an unexpected event occurs.

Resource utilization of the programmable logic part of the system is presented in Table I. It can be concluded that it is possible to implement improvements to the algorithm or to add further functionalities to the system. The estimated power usage is 2.191 W. A photo of the working system is presented in Figure 9.

V. Evaluation

To evaluate the system, several test sequences were recorded using the Yi camera mounted on the drone. The altitude data from the LiDAR sensor was also stored. Diverse images of the landing pad, i.e. for drone altitude from 0 to 1.5 m, for

[1]This is the same code as used in the software model [7]

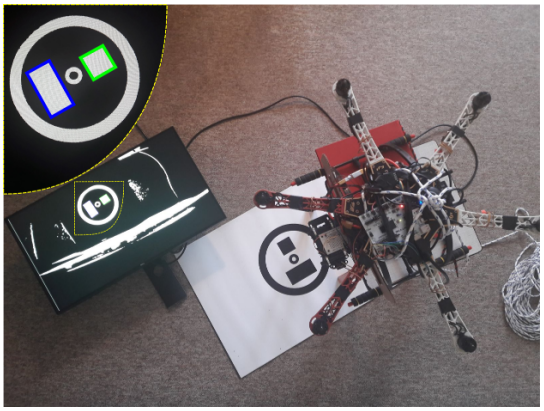


Fig. 9: Working hardware-software system. The image is processed by the Arty Z7 board on the drone at a altitude of about 0.5 m and the detected rectangle (blue bounding box) and square (green bounding box) are displayed on the monitor.

different orientation of the marker, for its different position in the image and on different background (outside and inside) were obtained.

At first, the marker detection rate was determined. 50 images were selected from the database, with the landing pad in different views. Then they were processed by the proposed algorithm. The marker was correctly detected on 48 images, which gives a 96% accuracy. The number of incorrectly detected shapes was also analysed – not a single shape outside the marker area has been falsely identified on the analysed images.

Secondly, the marker centre estimation accuracy was evaluated. The position returned by the algorithm was compared with a reference one (selected manually). The differences for the horizontal and vertical axes were calculated. Then the obtained results were averaged separately for each axis. For the horizontal axis the deviation was 0.19 pixels, while for the vertical axis 0.67 pixels.

The analysis of the presented results allows to draw the following conclusions about the performance of the vision algorithm. The percentage of frames with a correctly detected marker is high, but some cases turned out to be problematic. These were mainly images, in which the marker was far from the centre of the camera's field of view or separated into several fragments. The last mentioned case caused the failure to meet the conditions for the detection of particular shapes – this situation is presented in Figure 10. An attempt was made to solve that problem by using a less restrictive set of conditions. However, this in turn resulted in false detections, which was undesirable. Analysing the marker centre estimation result it can be concluded that this value has been determined with high accuracy (below 1 pixel), which enables precise drone navigation.

Fig. 10: Sample frame without correct marker detection. In this example the circle after filtration is separated into several pieces, so the defined detection conditions are not fulfilled.

VI. Conclusion

In this paper a hardware-software control system for autonomous landing of a drone on a static landing pad was presented. The use of a Zynq SoC device allowed to obtain real-time processing of a 1280×720 @ 60 fps video stream. The estimated power utilization of the system is 2.191 W. The designed algorithm has an accuracy of 96%. Unfortunately, due to the prevailing epidemic and the associated restrictions, it was not possible to test the fully integrated system (i.e. to perform a fully autonomous landing based on the proposed vision algorithm). This will be the first step of further work on the system and an opportunity to gather a larger database of test sequences in various conditions.

The next step will be landing on a mobile platform – moving slowly, quickly or imitating the conditions on water / sea (when the landing pad sways on waves). In the vision part, it is worth considering using a camera with an even larger viewing angle and evaluate how it affects the algorithm. In addition, the methods that allow to distinguish the marker from the background – like RTV (Relative Total Variation) from work [6], should be considered. Another option is to use an ArUco marker, although implementing its detection in a pipeline vision system seems to be a greater challenge. Moreover, adding a Kalman filter (KF) to the system should increase reliability if detection errors occur incidentally on some frames in the video sequence. Additionally, the fusion of a video and IMU (Inertial Measurement Unit) data (from the Pixhawk flight controller) should be considered.

Acknowledgement

The work was supported by AGH project number 16.16.120.773. The authors would like to thank Mr. Jakub Kłosiński, who during his bachelor thesis started the research on landing spot detection and Mr. Miłosz Mach, who was the constructor of the used drone.

REFERENCES

[1] S. Jin, J. Zhang, L. Shen, and T. Li, "On-board vision autonomous landing techniques for quadrotor: A survey," in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 10 284–10 289.

[2] R. Qiu, X. Miao, S. Zhuang, H. Jiang, and J. Chen, "Design and implementation of an autonomous landing control system of unmanned aerial vehicle for power line inspection," in *2017 Chinese Automation Congress (CAC)*. IEEE, 2017, pp. 7428–7431.

[3] C. Xu, Y. Tang, Z. Liang, and H. Yin, "Uav autonomous landing algorithm based on machine vision," in *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*. IEEE, 2018, pp. 824–829.

[4] S. Lee, T. Shim, S. Kim, J. Park, K. Hong, and H. Bang, "Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 108–114.

[5] C. Patruno, M. Nitti, A. Petitti, E. Stella, and T. D'Orazio, "A vision-based approach for unmanned aerial vehicle landing," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 645–664, 2019.

[6] Y. Huang, Y. Zhong, S. Cheng, and M. Ba, "Research on uav's autonomous target landing with image and gps under complex environment," in *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*. IEEE, 2019, pp. 97–102.

[7] Software model of the proposed application. [Online]. Available: https://github.com/vision-agh/drone_landing_static

[8] "Removed for blind review."