IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 14, 2020, accepted October 27, 2020, date of publication October 30, 2020, date of current version November 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034948

Development of an Automated Camera-Based Drone Landing System

MALIK DEMIRHAN[1] AND CHINTHAKA PREMACHANDRA[2], (Member, IEEE)
[1]Department of Electronic Engineering, Shibaura Institute of Technology, Tokyo 135-8548, Japan
[2]Department of Electronic Engineering, School of Engineering/Graduate School of Engineering and Science, Shibaura Institute of Technology, Tokyo 135-8548, Japan

Corresponding author: Malik Demirhan (malik.demirhan@web.de)

This work was supported in part by the Branding Research Fund of Shibaura Institute of Technology.

ABSTRACT Digital image processing serves as a multifunctional tool for measurement and positioning tasks in robotics. The present paper deals with the development of a camera-based positioning system for quadrocopters in order to automate their landing process. In this regard, a quadrocopter equipped with classical radio-control components is upgraded with applicable hardware, such as a Raspberry Pi 3B+ and a wide-angle camera. Hereupon, black-box system identifications are executed to attain the relevant plants of the attitude control performed by the flight controller. Thereby, a PID-controller for the altitude control including a back-calculation anti-windup as well as two PD-controllers for the horizontal plane are designed using a pole placement method. Effective tests of the controller gains are conducted by simulating the closed loops respectively. Since the camera functions as a position sensor, an image processing algorithm is then implemented to detect a distinctive landing symbol in real time while converting its image position into compliant feedback errors (pixel-to-physical distance-conversion). Ultimately, the developed system allows for the robust detection and successful landing on the landing spot by a position control operating at 20 hertz.

INDEX TERMS Autonomous drone, quadrocopter, landing, PID, control engineering, system identification, image processing, wide angle camera, region of interest.

I. INTRODUCTION

Over the last years the popularity of unmanned aerial vehicles increased enormously together with their use cases. While in the beginning they were mainly used for military purposes, they are now being introduced more and more in economic as well as academic sectors. Especially automated functions such as flying through specific coordinates autonomously are gaining importance. In this matter, our research group has been conducting studies about various possible applications of (camera-equipped) quadrocopters [1]–[8]. Rescue and aid missions after natural disasters fall under the applications amongst others. Critical for such missions are the autonomous and safe landings of the drones. However, random objects and uneven grounds can have a negative impact on the landing process itself. Therefore, an automated landing system based on a depth camera is desired within a bigger project. Such cameras namely can provide information about the terrain evenness, which allows for the finding of appropriate landing spots in various environments.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

But beforehand, a framework respectively useful base is to be developed from the beginning. This includes the design and validation of an automated landing system based on a classical 2D-camera combined with a symbolized landing pad. For this purpose, a quadrocopter equipped with radio-control (RC) components must be upgraded through additional hardware and software. We explicitly settled on the usage of RC-/"plug and play" components as well as easily accessible sensors/hardware since it accelerates the development process and enables other researchers to conveniently apply the presented methods of this paper to their own systems. Later, a position control must be implemented in order to execute suitable landings. Other works rely on relatively complex approaches for generating controller feedback respectively estimating the drone's position and carrying out the control process, e.g., the implementation of Kalman filters as in [9], [10] and the implementation of an adaptive PID controller as in [11]. In this work however, we kept the system complexity rather low by only relying on a Time-of-Flight (ToF) sensor, a camera and innovative software solutions regarding the control system. The challenges in this project, among others, include the development of fast and robust image processing algorithms to provide the desired position feedback while maintaining the desired loop frequency. Instead of detecting the landing symbol through methods based on a neural network respectively deep learning (e.g., [12]), we focused on classical image processing algorithms, as these can be highly efficient when designed deliberately. Another major challenge includes the design of an innovative fail-safe switch logic allowing the pilot to abort the autonomous mode and control the drone manually at any time, hence preventing accidents during tests.

Chapter 2 covers the basics of quadrocopter dynamics as well as the upgrade process of the used model. In Chapter 3, the implementation of the quadrocopter's altitude control through a PID controller with a back-calculation anti-windup is presented. The design of two PD controllers for the horizontal plane based on black-box system identifications is then described in Chapter 4. Subsequently, Chapter 5 includes the development and optimization of the image processing algorithm for generating horizontal feedback. Finally, Chapter 6 deals with the implementation/test of the horizontal control and the overall landing process. Finally, Chapter 7 concludes the paper.

II. QUADROCOPTER DYNAMICS

A quadrocopter is a multiple-input-multiple-output-system with 6 degrees of freedom and 4 motors/inputs (Fig. 1). Generally, a body fixed coordinate system (index B) as well as inertial one (index I) are defined. Converting the body fixed accelerations into the latter, assuming the quadrocopter is a point mass, is possible by applying the rotation matrix as in [13]–[15]:

$$\begin{bmatrix} m\ddot{x}_I \\ m\ddot{y}_I \\ m\ddot{z}_I \end{bmatrix} = \mathbf{R}\begin{bmatrix} 0 \\ 0 \\ \Sigma F_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = \begin{bmatrix} (c\psi s\theta c\phi + s\psi s\phi)\Sigma F_i \\ (s\psi s\theta c\phi - c\psi s\phi)\Sigma F_i \\ (c\theta c\phi)\Sigma F_i - mg \end{bmatrix} \quad (1)$$

$$\mathbf{R}(\phi, \theta, \psi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \quad (2)$$

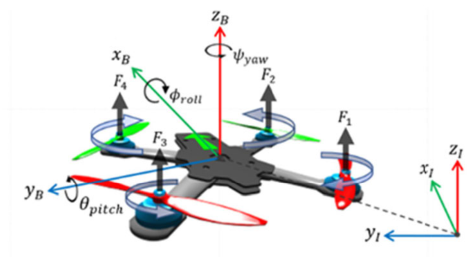


FIGURE 1. Quadrocopter Euler angles and coordinate systems.

F_i describes the uplift force of the rotors respectively and ϕ, θ and ψ the roll, pitch and yaw Euler angles (s = sin, c = cos).

A. MICROCONTROLLER AND SIGNAL GENERATION

A quadrocopter based on RC components is usually maneuvered through the roll, pitch, yaw and throttle flight commands coming from the transmitter operated by the pilot.

At first, our system only consisted of a FUTABA T6L Sport transmitter, a FUTABA R3106GF receiver with six channels (using the PWM-protocol) and an SP Racing F3 flight controller. In order to replace the pilot and enable autonomous control, we need a microcontroller to give out the necessary commands to the flight controller. Since the desired position control will be camera-based, CPU-intensive image processing must be considered while choosing the right hardware. Hence, a Raspberry Pi Model 3B+ (RPi) is used, which offers high performance in addition to fast development by using available libraries such as OpenCV. Lastly, Fig. 2 illustrates the implemented signal routing/switch logic for achieving autonomous signal generation. By using a 4 × 2:1 multiplexer (TC74HC157A [16]) the drone can be controlled either manually by the pilot/receiver or the RPi. Therefore, the RPi reads in the channel 6 receiver signal, which is linked to a shaft encoder on the transmitter. Depending on this PWM signal (ranging between 1-2 ms) and a defined threshold, the RPi sends out either a Low or a High signal to the multiplexer. The multiplexer then switches its inputs accordingly, whereby only the flight commands of either the receiver or the RPi are passed to the flight controller. Additionally, a pull-down resistor always provides a defined Low signal in case the RPi should have a defect and turns off randomly. Hence, we can still control the quadrocopter manually after mid-flight failures of the single-board computer (fail-safe system). Since this setup allows for a continuous switching between the two described modes, severe accidents through unwanted behaviors of the autonomous mode can be prevented.

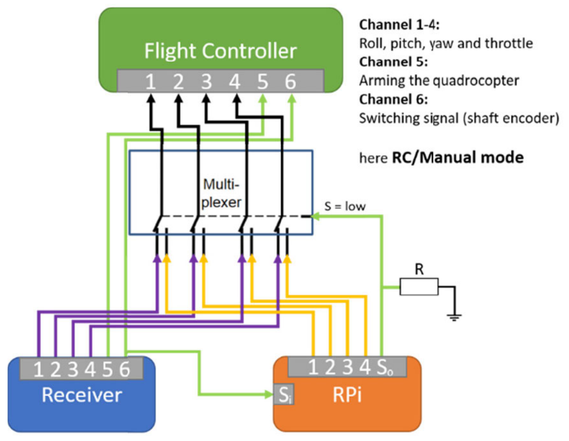


FIGURE 2. Switch logic between autonomous control (Raspberry Pi) and manual/RC control.

B. GIMBAL, CAMERA AND HEIGHT SENSOR

Fig. 3 illustrates the necessity of a camera stabilization. By combining two servos with 3D printed parts, we built a cheap and light-weight gimbal (Fig. 4).

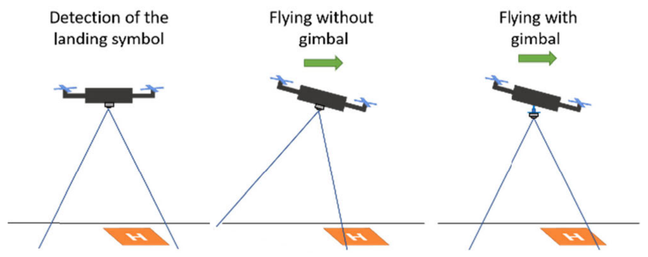
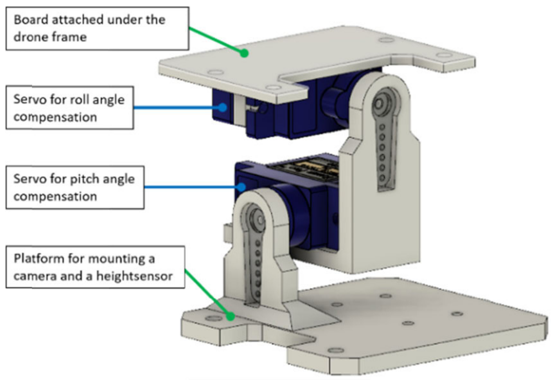


FIGURE 3. Necessity of a gimbal for camera usage.



Board attached under the drone frame
Servo for roll angle compensation
Servo for pitch angle compensation
Platform for mounting a camera and a heightsensor

FIGURE 4. 3-D printed servo gimbal.

To compensate the current roll and pitch angles of the quadrocopter, their shafts must move accordingly in the opposite direction. For this purpose, the CAMSTAB function of the flight controller is employed.

As for the camera, a modified Raspberry Pi v1.3 camera with a fisheye lens is used (Longrunner LC20). While it provides a wide field of view with a diagonal angle of 160°, the resulting distortions must be mentioned as a downside. However, it can flexibly be set to different fps and resolutions.

In addition, a VL53L1X Time-of-Flight (ToF) module serves as the feedback sensor of the quadrocopter's height. It offers millimeter resolution and is operated at 33 Hz, allowing for height measurements of up to 3 m. Finally, Fig. 5 depicts the upgraded quadrocopter.

III. ALTITUDE CONTROL

Next, the height control operating at 20 Hz will be implemented by designing a PID controller (parallel form):

$$G_{PID}(s) = \frac{K_D \cdot s^2 + K_P \cdot s + K_I}{s} \tag{3}$$

The integral part of the controller will help us in compensating the loss of uplift force due to the continuous discharge of the battery mid-air. Using the black-box measurement feature of the flight controller, we can measure the vertical acceleration (output) of the quadrocopter together with the throttle flight command (input) over time. After collecting enough data, we can deduce the throttle command for hovering/compensating gravity as well as a PT1 element to approximate the acceleration-throttle relation (Fig. 6).

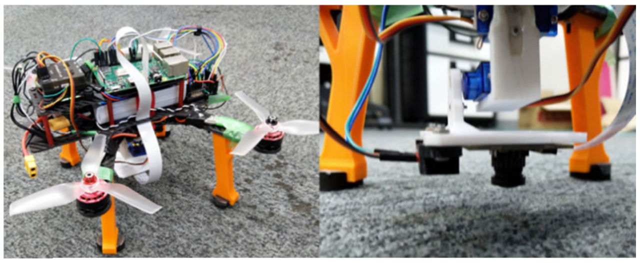


FIGURE 5. The used quadrocopter with its components.

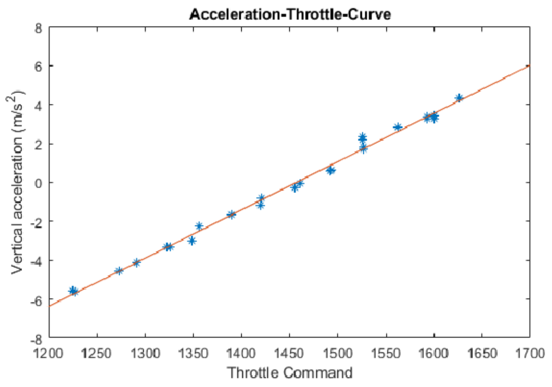


FIGURE 6. Acceleration characteristics of the quadrocopter.

The overall plant for the height z can thus be described as:

$$\frac{Z(s)}{CM_{throttle}(s)} = \frac{\ddot{Z}(s)}{CM_{throttle}(s)} \frac{1}{s^2} = \frac{0.0248\text{m/s}^2}{(0.08s+1)s^2} \tag{4}$$

The controller gain values are then calculated by pole placement respectively coefficient comparison between the closed loop transfer function and a desired transfer function according to [17], [18]. However, the pole of the throttle-PT1 element as well as the zeros of the closed loop are neglected for this purpose:

$$\frac{\frac{\omega_{0,w}^2}{T_w}}{s^3 + \left(2D_w\omega_{0,w} + \frac{1}{T_w}\right)s^2 + \left(\frac{2D_w}{T_w} + \omega_{0,w}\right)\omega_{0,w}s + \frac{\omega_{0,w^2}}{T_w}} = \frac{K_{S,T}K_I}{s^3 + K_{S,T}K_Ds^2 + K_{S,T}K_Ps + K_{S,T}K_I} \tag{5}$$

At first, we calculated the controller gains for a closed loop simulation in MATLAB Simulink by choosing a desired damping of $D_w = 0.85$, a frequency of $\omega_{0,w} = 1.95$ rad/s and a time constant of $T_w = 1$ s to $K_P = 2.87$, $K_D = 1.74$ and $K_I = 1.53$ (Fig. 7).

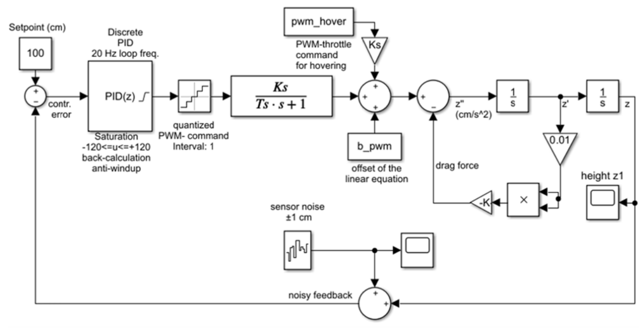

Setpoint (cm)
100
Discrete PID 20 Hz loop freq.
PID(z)
contr. error
Saturation -120<=u<=+120 back-calculation anti-windup
quantized PWM- command Interval: 1
Ks / Ts·s+1
pwm_hover
PWM-throttle command for hovering
Ks
b_pwm
offset of the linear equation
z'' (cm/s^2)
1/s
z'
z
0.01
drag force
-K
height z1
sensor noise ±1 cm
noisy feedback

FIGURE 7. The Simulink simulation model for the height control.

The resulting controller gains combined with a ±150 output saturation of the throttle-command around its hovering-state value (control ratio) and a classic clamping anti-windup resulted in a rather unsatisfying overshoot not suited for landings (Fig. 8). Nevertheless, the simulation's step response was compared to experiments using the real system with the same control parameters.

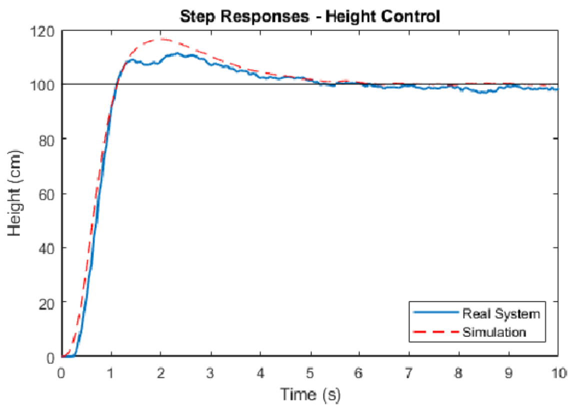


FIGURE 8. Height control step responses - real system and simulation with a clamping anti-windup.

The high similarity between the curves' progression serves as a sufficient validation of the built simulation and system parameters. Nonetheless, manipulating the step response of the real system to make it suitable for landings by simply varying the controller gains poses a difficult task. Reason being is that the PID controller with its two zeros can only compensate two poles of the existing third order plant [19]. Therefore, we replaced the existing anti-windup method with a back-calculation logic according to [20] (Fig. 9).

This anti-windup method allows for the effective change of the system dynamics by ''discharging'' the integrator against its natural direction of error-integration during saturation. Hence it can be useful regarding the prevention of high overshoots. Eventually, the final controller gains were determined by tuning the desired transfer function (5) and the back-calculation factor K_b empirically. Table 1 includes the corresponding values based on $D_w = 0.95$, $\omega_{0,w} = 1.5$ rad/s and $T_w = 2$ s. Also, we decreased the already mentioned throttle saturation to ±120 around the hover-value.

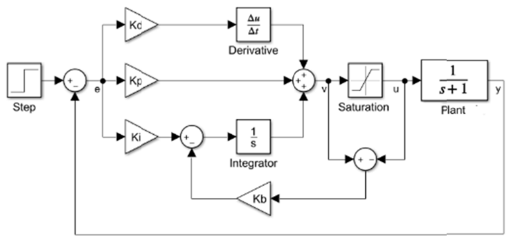


FIGURE 9. Back-calculation anti-windup principle.

TABLE 1. Height Control PID Gains.

Altitude PID Controller Gains			
$K_P = 1.48$	$K_D = 1.35$	$K_I = 0.45$	$K_b = 6$

The final throttle command to the flight controller contains the saturated PID controller output $y_{PID,sat}$ added to the hovering-throttle command $CM_{throttle,hover}$ and is corrected by the current roll and pitch angles:

$$y_{throttle}(k) = \frac{y_{PID,sat}(k) + CM_{throttle,hover}}{\cos(\phi(k))\cos(\theta(k))} \tag{6}$$

Furthermore, a PT1-filter with $\alpha = 0.6$ is applied to the control error of the derivative part decreasing the controller's sensitivity against sensor noise and improving the performance of the back-calculation anti-windup [21]:

$$e_{z,filt}(k) = (1-\alpha)\, e_{z,filt}(k-1) + \alpha e_z(k) \tag{7}$$

With this configuration we can achieve a sufficiently damped step response required for adequate landings (Fig. 10).

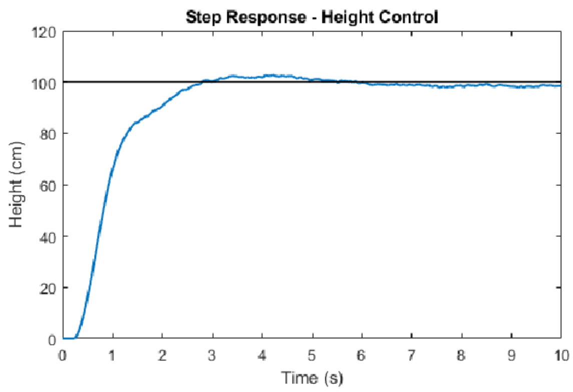


FIGURE 10. Step response of the height control - real system with a back-calculation anti-windup.

IV. CONTROLLER DESIGN FOR POSITIONING IN THE HORIZOTAL PLANE

A. BLACK-BOX SYSTEM IDENTIFICATION

Horizontal movements can be achieved through roll and pitch maneuvers without needing the yaw command. In order to design PD controllers, we first simplify (1) with the following ideas:

- Linearization of the trigonometric functions by approximating $\sin x \approx x$ and $\cos x \approx 1$ for small angels.
- The height setpoint during the horizontal control/approach of the landing spot will not change. Hence, we can assume an almost hovering state, so that the uplift force of the rotors roughly equals the gravitational force ($\Sigma F = mg$).
- No yaw movements during the landing process allow us to set the yaw angle to $\psi = 0°$.

By that, the accelerations in the horizontal plane can be approximated as:

$$\begin{bmatrix} \ddot{x}_I \\ \ddot{y}_I \end{bmatrix} \approx \begin{bmatrix} \theta_g \\ -\phi_g \end{bmatrix} \tag{8}$$

At this point, the Euler angles are unknown to us since they are being controlled by the flight controller as outputs depending on the flight commands. Hence, we deduce models for the roll and pitch plants of the quadrocopter by applying black-box system identifications. First, measurements are prepared by letting the RPi handle the height control at a specific setpoint, while the pilot solely focuses on exciting the quadrocopter with roll and pitch maneuvers respectively. In this regard, the roll, pitch and yaw inputs of the flight controller will temporarily be connected to the receiver directly instead of passing them to the multiplexer. With the resulting 4 min long data sets suitable models are identified by applying the instrumental variable method (IV) [22]. We therefore used MATLAB's system identification toolbox to determine PT1 elements for the controller design as well as models of higher order for accurate simulation purposes. Cross validations have been executed through additionally prepared data sets. Furthermore, different resampling rates have been analyzed regarding their effect on the fit values of the resulting models as in [23]. The following transfer functions have resulted with 50 Hz being the most beneficial sampling rate during the identification process (Table 2):

TABLE 2. Identified black-box models of the roll and pitch system plant.

Blackbox Models	Model Type and Fit Values	
	PT1	*Higher Order*
Roll	$\frac{0{,}08457°}{0.0896s+1}$	$\frac{0.6064s+0.2205}{s^2+6.962s+7.068}$
Pitch	$\frac{0{,}08605°}{0.103s+1}$	$\frac{0.6226s+0.472}{s^2+7.354s+10.06}$

Fig. 11 shows the graphical comparison between the measured and simulated system outputs of the stated models using the validation data sets.

B. PD CONTROLLER DESIGN

By integrating the determined PT1 roll and pitch models twice we can obtain the plants describing the horizontal movement of the quadrocopter:

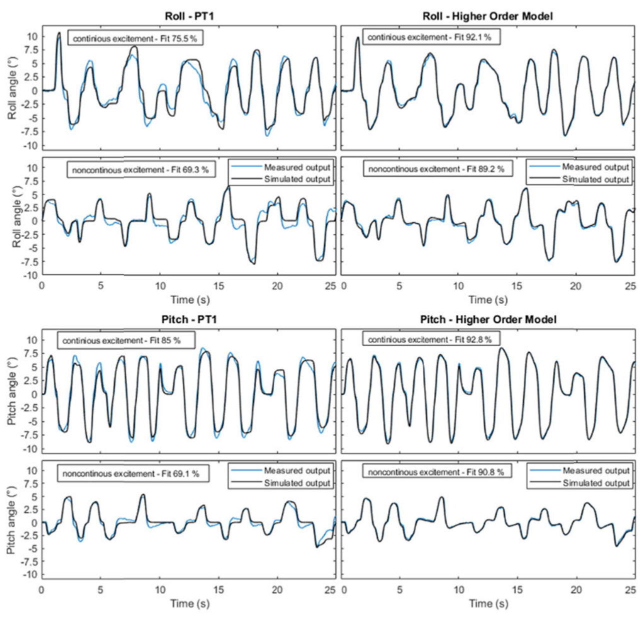


FIGURE 11. Comparing the measured system output with the ones simulated through the identified black-box models.

$$\frac{X(s)}{CM_{pitch}(s)} = \frac{K_{S,P} \cdot \frac{\pi}{180} \cdot g}{(T_{S,P} \cdot s + 1) \cdot s^2} \quad \text{and}$$
$$\frac{Y(s)}{CM_{roll}(s)} = \frac{-K_{S,R} \cdot \frac{\pi}{180} \cdot g}{(T_{S,R} \cdot s + 1) \cdot s^2} \tag{9}$$

The PD gains for the roll and pitch commands (R/P) are then again calculated by pole placement through the following equation (neglecting the plant's PT1 element and the zero of the closed loop respectively):

$$\frac{\omega_{0,w}^2}{s^2 + 2D_w\omega_{0,w}s + \omega_{0,w}^2} = \frac{gK_PK_{S,R/P}}{s^2 + gK_DK_{S,R/P}s + gK_PK_{S,R/P}} \tag{10}$$

Based on $D_w = 0.8$, $\omega_{0,w} = 1.2$ rad/s, $g = 981$ cm/s^2 and $K_{S,R/P} = \frac{\pi}{180°}\text{mean}(K_{S,R}, K_{S,P})$ the gains are $K_{Dx/y} = 0.88$ and $K_{Px/y} = 0.66$. Additionally, an output saturation of ± 30 will be introduced to limit the Euler angles ($<3°$). The step responses of the simulation model in Fig. 12 can be seen in Fig. 13. Even with a feedback delay of 30 ms, representing a rough/temporary estimation of the image processing time consumption, the results are satisfactory.

V. DETECTION OF THE LANDING SYMBOL

A. ALGORITHM TO DETECT THE H-SYMBOL

To test the horizontal control, camera-based feedback is needed. We used a landing pad with high color contrasts and a unique symbol for easy segmentation. Initially, basic operations like the transformation of the RGB image into grayscale and its binarization through a simple threshold method after applying a Gaussian filter are executed [24].

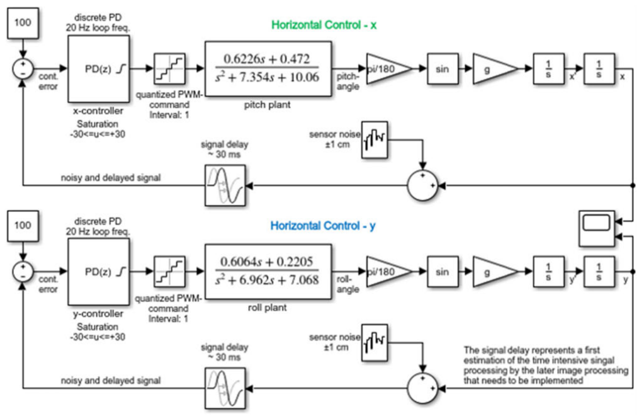

Horizontal Control - x
discrete PD 20 Hz loop freq.
PD(z)
x-controller Saturation -30<=u<=+30
quantized PWM-command Interval: 1
0.6226s + 0.472 / s² + 7.354s + 10.06
pitch plant
pitch-angle
pi/180
sin
g
noisy and delayed signal
signal delay ~ 30 ms
sensor noise ±1 cm
Horizontal Control - y
roll plant
0.6064s + 0.2205 / s² + 6.962s + 7.068
y-controller
roll-angle
The signal delay represents a first estimation of the time intensive singal processing by the later image processing that needs to be implemented

FIGURE 12. The Simulink simulation model of the horizontal control.

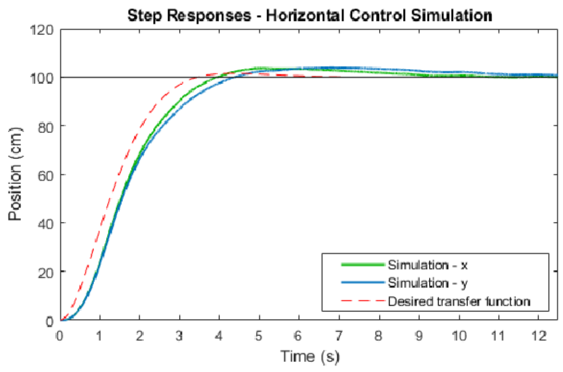


FIGURE 13. Step responses of the horizontal control – simulation.

A slight dilation by a 3×3 mask increases the recognizability of the 20 cm large H-symbol from further distances (Fig. 14).

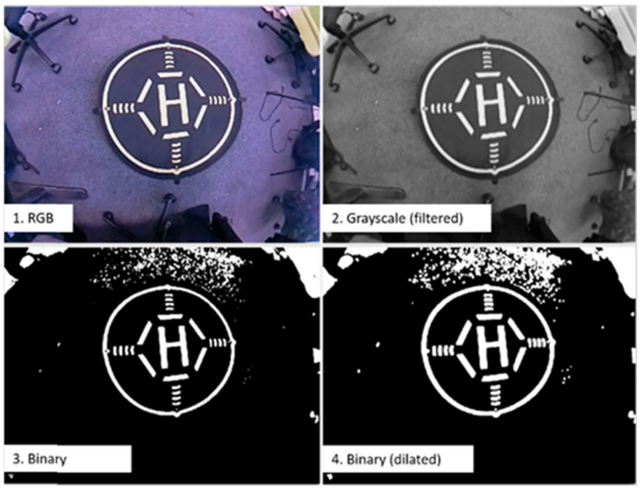


FIGURE 14. Binarizing and dilating the RGB image.

The flowchart of Fig. 15 describes the successive algorithm to detect the H-symbol in the image. First, the closed contours are identified through the cv.findContours function from the OpenCV library based on [25]. Hereafter, the H-symbol is searched for in the resulting contour vectors inside a loop. Only those contours, which neither include a child contour nor are described with less than 30 points are passed for further examination. Subsequently, the current contour is approximated to such a degree that ideally 12 corner points should remain if it was the H-symbol (by cv.approxPolyDP). The next condition requires the contour to have a total length of at least 200 pixels. Ultimately, the H-symbol can be robustly determined by the final two conditions. They are verifying, whether the contour has 5 lines between the two longest ones and whether those two are at least four times longer than their direct neighbours respectively. Small tolerances in step 3 and 5 of the flowchart allow the detection of the H-symbol even if it should be approximated by 13 corner points or has up to 7 lines between its two longest ones. A positive detection results in the immediate abortion of the loop (we presume that maximally only one H-symbol will appear in the image). By that, we can successfully identify the landing symbol in heights of 15-165 cm as shown in Fig. 16. The green center point is essentially the desired landing spot and can be deduced by averaging all the corner points of the

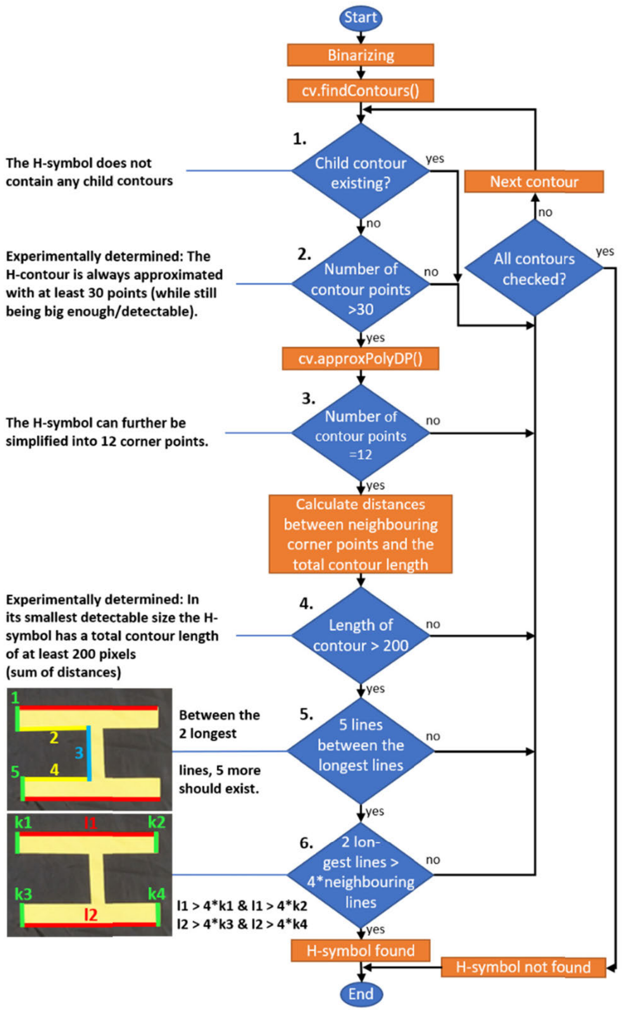

Start
Binarizing
cv.findContours()
1.
Child contour existing?
yes
no
Next contour
All contours checked?
The H-symbol does not contain any child contours
2.
Number of contour points >30
Experimentally determined: The H-contour is always approximated with at least 30 points (while still being big enough/detectable).
cv.approxPolyDP()
3.
Number of contour points =12
The H-symbol can further be simplified into 12 corner points.
Calculate distances between neighbouring corner points and the total contour length
4.
Length of contour > 200
Experimentally determined: In its smallest detectable size the H-symbol has a total contour length of at least 200 pixels (sum of distances)
5.
5 lines between the longest lines
Between the 2 longest lines, 5 more should exist.
6.
2 longest lines > 4*neighbouring lines
l1 > 4*k1 & l1 > 4*k2
l2 > 4*k3 & l2 > 4*k4
k1
l1
k2
k3
k4
l2
H-symbol found
H-symbol not found
End

FIGURE 15. Flowchart of detecting the landing symbol (H).

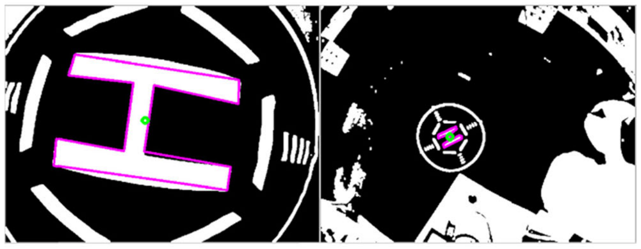


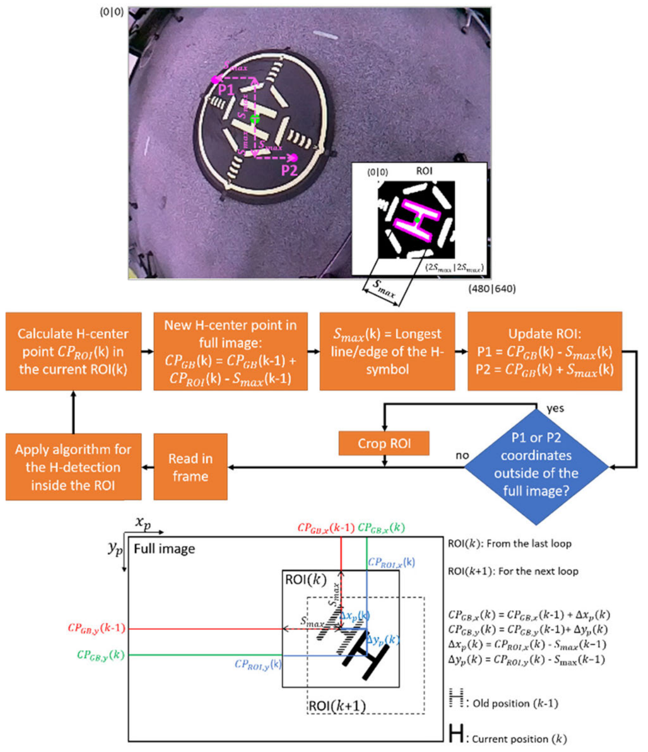
FIGURE 16. Detection of the H-symbol from 15 cm (left) and 165 cm (right).

contour. In this form the algorithm needs about 12-22 ms from the start to the end of the flowchart in Fig. 15. The computing time variation depends on the image's lineup and tends to be higher when the H-symbol is in the picture.

B. INTRODUCING AN ADAPTIVE REGION OF INTEREST

After the very first detection of the landing symbol, pixels distant to the tracked object lose their relevance. The reason for this is that between two frames the object cannot change its position in the image too much due to the quadrocopter's inertia and limited speed. Hence, a Region of Interest (ROI) can be introduced, which allows us to speed up the algorithm by only processing the relevant parts of the image. Fig. 17 depicts the implementation of the ROI. After reading in the image, the detection algorithm only processes the ROI determined in the last loop. Therefore, a square window spanned by the points P1 and P2 is cut out of the whole/full frame (Fig. 17, top). However, the ROI is regarded as a new image and hence makes us lose the global coordinates of the H-symbol in the full image (Fig. 17 edge coordinates of the ROI, top). Thereby, we would not know in which part of the frame we must place the ROI for the detection in the next loop. Thus, we require a (recursive) transformation of the locally determined center point of the H-symbol CP_{ROI} (k) back into global coordinates CP_{GB} (k). We know the ROI has been placed around the global H-symbol coordinates CP_{GB} (k − 1) in the last loop (k − 1). Due to the physical inertia of the drone the H-symbol can only move away slightly from the ROI's center in between two loops. This displacement is described by Δx_p and Δy_p (Fig. 17, bottom). Accordingly, only these two values are of interest in order to update the global coordinates of the H-symbol CP_{GB} (k). For this purpose, we simply add the displacement to the global center point from the last loop CP_{GB} (k − 1). Δx_p and Δy_p can be deduced by subtracting the ROI's center coordinates from the newly acquired local center point of the H-symbol CP_{ROI} (k).

The ROI's center is half of its height respectively width S_{max} (k − 1), which represents the longest line of the H detected in the last loop (if the ROI wasn't cropped). After calculating CP_{GB} (k), the ROI needs to be placed around these new global coordinates. Its height and width are updated to be twice as large as the H-symbol's current longest line S_{max} (k), making it adaptive to the size of the landing symbol


(0|0)
P1
P2
S_{max}
(0|0)
ROI
$(2S_{max}|2S_{max})$
(480|640)
Calculate H-center point CP_{ROI}(k) in the current ROI(k)
New H-center point in full image: CP_{GB}(k) = CP_{GB}(k-1) + CP_{ROI}(k) - S_{max}(k-1)
S_{max}(k) = Longest line/edge of the H-symbol
Update ROI: P1 = CP_{GB}(k) - S_{max}(k) P2 = CP_{GB}(k) + S_{max}(k)
yes
P1 or P2 coordinates outside of the full image?
no
Crop ROI
Read in frame
Apply algorithm for the H-detection inside the ROI
x_p
y_p
Full image
$CP_{GB,x}(k\text{-}1)$
$CP_{GB,x}(k)$
$CP_{ROI,x}$(k)
ROI(k)
S_{max}
S_{max}
Δx_p(k)
$\Delta y_p(k)$
$CP_{GB,y}(k\text{-}1)$
$CP_{GB,y}(k)$
$CP_{ROI,y}$(k)
ROI(k+1)
ROI(k): From the last loop
ROI(k+1): For the next loop
$CP_{GB,x}(k) = CP_{GB,x}(k\text{-}1) + \Delta x_p(k)$
$CP_{GB,y}(k) = CP_{GB,y}(k\text{-}1) + \Delta y_p(k)$
$\Delta x_p(k) = CP_{ROI,x}(k) - S_{max}(k-1)$
$\Delta y_p(k) = CP_{ROI,y}(k) - S_{max}(k-1)$
H: Old position (k-1)
H: Current position (k)
FIGURE 17. Implementation and theoretical operation of the ROI.

in the image (Fig. 18 and attached video). In the case of P1 or P2 being out of the image boundaries, the ROI will be cropped appropriately (e.g., H-symbol too large or nearly at the edge of the frame). Finally, Fig. 19 shows the effectivity of the ROI by depicting the computing time over the camera's vertical distance to the landing symbol respectively the size of the ROI. As seen, the computing time drops from about 12-14 ms down to 3 ms with steadily increasing vertical distance respectively shrinking ROI size. Starting with a bisection above 50 cm we obtain high savings of time, which justifies the usage of the ROI and could even allow us to use higher loop frequencies.

C. GENERATING COMPLIANT FEEDBACK ERRORS

Within the horizontal control the displacement of the landing symbol to the image's center point and hence to the quadrocopter's physical center is to be minimized. Currently, we only attain position values in pixels, while the PD controllers are configured for physical feedback (cm). Thus, a continuous conversion of the landing symbol's image position into compliant feedback errors is needed. Innovatively deducing a reliable regularity for this task was possible by conducting multiple measurements of the H-symbol's image position in respect to the image center at equally spaced physical distances (both vertically and horizontally). Later, we reversed the data so that the image position (in pixels) at constant heights (cm) serves as the input, whereas the horizontal distance (cm) represents the output (Fig. 20).

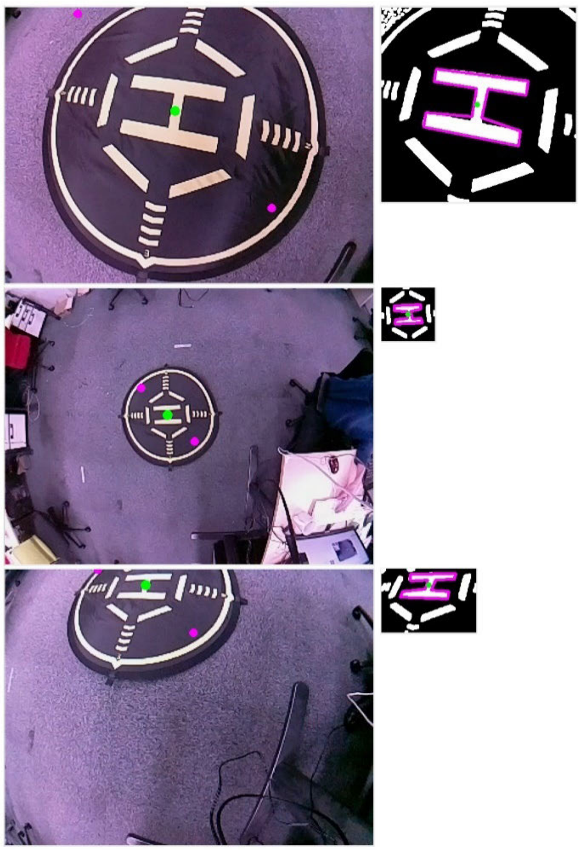


FIGURE 18. Adaptation of the ROI.

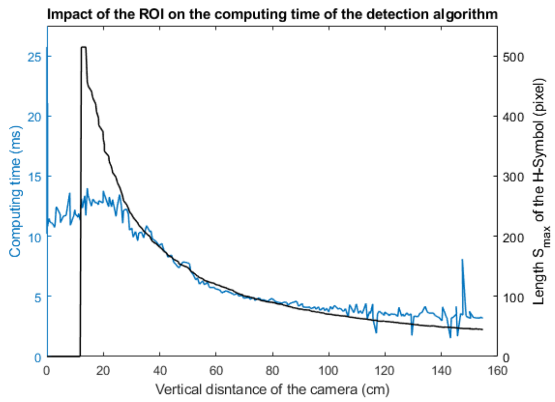


FIGURE 19. Computing time of the detection algorithm using the ROI.

Furthermore, regression lines have been laid through the measurements at constant heights.

Generally, the relation between the horizontal distances and the image position at a constant height can be approximated as linear. As expected, the slopes of those lines increase together with the height of the camera, since zooming out results in less pixels describing the same physical length. Hence the conversion should be a function of both the pixel distances p_x and p_y as well as the height z:

$$x(m(z), p_x) \approx m(z)\, p_x$$
$$y\left(m(z), p_y\right) \approx m(z)\, p_y \tag{11}$$

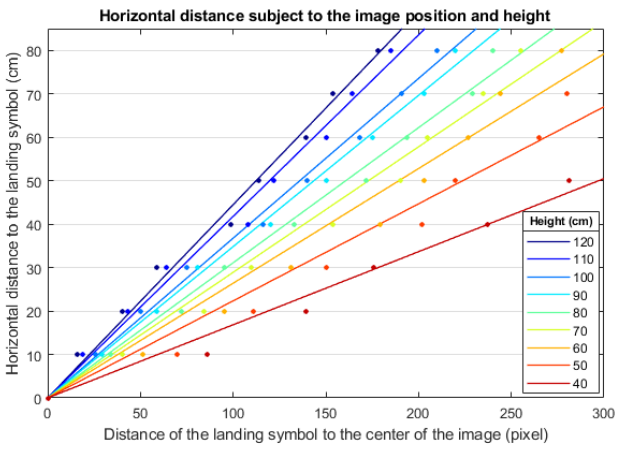


FIGURE 20. Inverted measurements representing the relationship between the horizontal physical distance as a function of the height and the distance of the landing symbol to the image center in pixels.

The slope-function m (z) can again be approximated by a regression line through the slopes derived from the individual lines at constant heights (Fig. 21).

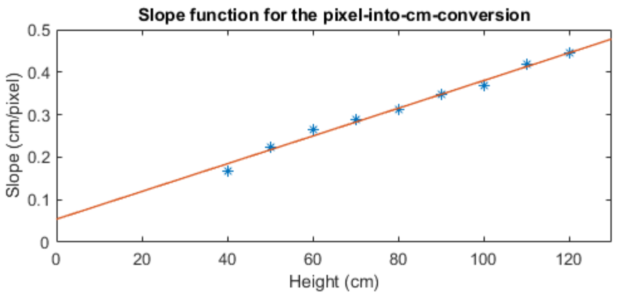


FIGURE 21. Determining the slope function m(z) by regressing a line through the deduced slopes at different heights (from Fig. 20).

With that, the final functions converting the pixel values into physical ones are:

$$x(m(z), p_x) \approx \left(0,00326\frac{\text{cm}}{\text{pixel}} z + 0,0541\frac{\text{cm}}{\text{pixel}}\right) p_x$$
$$y\left(m(z), p_y\right) \approx \left(0,00326\frac{\text{cm}}{\text{pixel}} z + 0,0541\frac{\text{cm}}{\text{pixel}}\right) p_y \tag{12}$$

VI. IMPLEMENTING THE POSITION CONTROL

A. HORIZONTAL CONTROL

With the controller design and the feedback generation being finished, we are now able to test the horizontal control. Fig. 22 illustrates the orientation of the camera mounted onto the quadrocopter and the defined axes, while Fig. 23 describes the general program flow. First, the current camera frame is read followed by the detection algorithm and the update of the current height (ToF-sensor-value). A successful detection of the H-symbol only leads to the activation of the autonomous mode respectively the horizontal control if the pilot enables this through a switch on the remote control. For that matter, a successful detection is defined as follows:

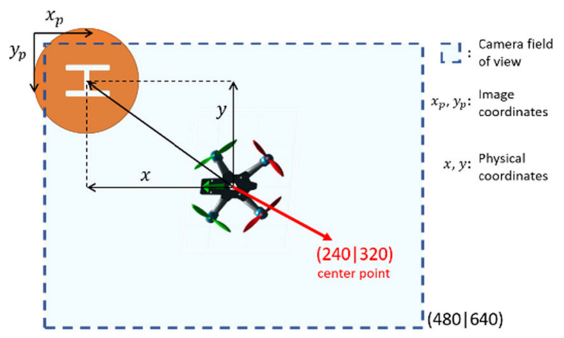


FIGURE 22. Orientation of the camera/image on the drone.

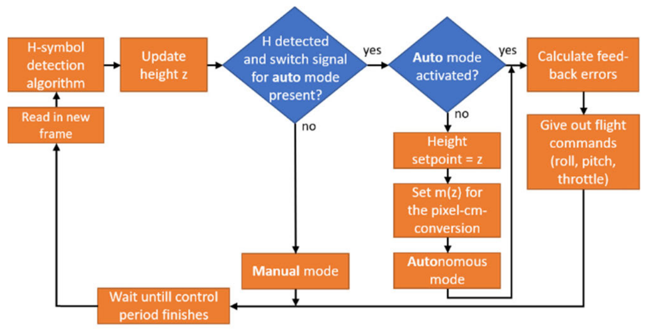

H-symbol detection algorithm
Update height z
H detected and switch signal for auto mode present?
yes
Auto mode activated?
yes
Calculate feed-back errors
Read in new frame
no
no
Height setpoint = z
Give out flight commands (roll, pitch, throttle)
Set m(z) for the pixel-cm-conversion
Manual mode
Autonomous mode
Wait untill control period finishes

FIGURE 23. Flowchart of the horizontal control test.

- 3 consecutive respectively uninterrupted detections, if the autonomous mode was previously deactivated. The first takes place in the full image, while the last two only process the emerging ROI.
- 2 consecutive failed detections at most during the autonomous mode. After the third, the manual mode is reactivated and the ROI reset.

These precautions allow for a safer start of the horizontal control as well as a higher robustness against randomly failed detections during said process. Moreover, the horizontal control should take place during the hovering state of the quadrocopter. Hence, the height setpoint is set one-time to the last measured vertical distance z at the beginning of the autonomous mode. In the same way, the slope m(z) for the pixel-into-cm conversion is fixed uniquely. While this prevents adaptions of the horizontal feedback to slight changes in the real height, it also decouples said feedback from the ToF-sensor's noise. Finally, the RPi calculates the feedback errors and sends flight commands to the flight controller accordingly. Due to the low mechanical accuracy of the servo gimbal (low angle resolutions) as well as general noise in the image processing, we also applied PT1-filters for the x-axis and y-axis respectively:

$$e_{x/y,\mathrm{filt}}(k) = 0.5e_{x/y,\mathrm{filt}}(k-1) + 0.5e_{x/y}(k) \tag{13}$$

The tests were then conducted by flying the quadrocopter manually towards the landing symbol until the RPi takes over the control by itself. After several experiments we additionally decided on slightly tuning the controller gains in order to get a smoother/more damped step response and limit the drone's oscillation around its setpoint. With $D_w = 0.87$ and $\omega_{0,w} = 1$ rad/s the final gains result in $K_{Dx/y} = 0.8$ and $K_{Px/y} = 0.46$. Additionally, the pitch and roll command saturations were further decreased to ± 25.

Fig. 24 lastly illustrates the step responses of such an experiment with the final configuration by plotting the variation of all axes with time as well as including the H-symbol's image position over time. While the damping of the system is high enough to prevent significant overshoots, the quadrocopter slightly oscillates around the setpoint with amplitudes of maximally ± 8 cm respectively ± 18 pixels. These could result from the quadrocopter being not perfectly balanced (causing continuous drifts) as well as the servo gimbal compensating the roll and pitch angles in a rather inaccurate way. Nevertheless, the horizontal control possesses a satisfying precision and is hence usable for the overall landing process.

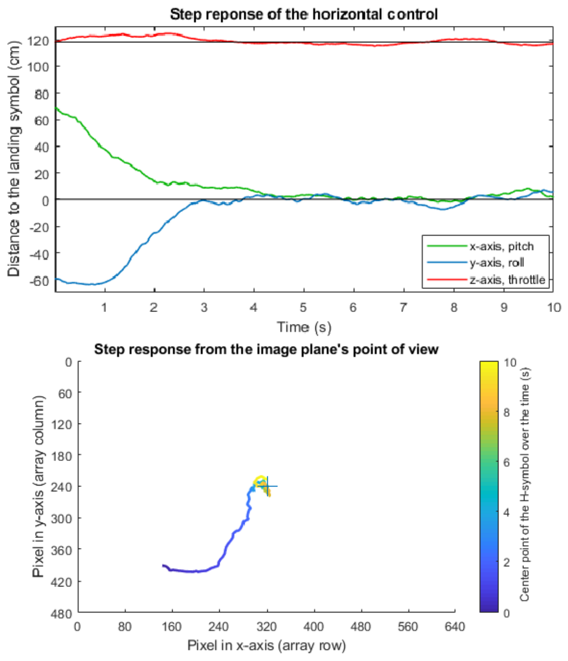


FIGURE 24. Step response of the horizontal control (physical and image plane).

B. THE OVERALL LANDING PROCESS

The overall landing process can be divided into two sub-steps as shown in Fig. 25. After detecting the landing symbol, the quadrocopter flies towards it while keeping its height. In order to activate the second sub-step, it must be airborne in a predefined horizontal tolerance zone (± 10 cm radius) for 4 s. This ensures, that the vertical landing starts after the transient effect respectively any possible overshoots. Subsequently, the height setpoint is set to -3 cm instead of 0 cm. Even though this is not possible physically, it allows for a more precise landing behavior, which is disturbed by the ground effect. The latter arises from the quadrocopter being too near to the ground, hindering the downwards airflow and hence

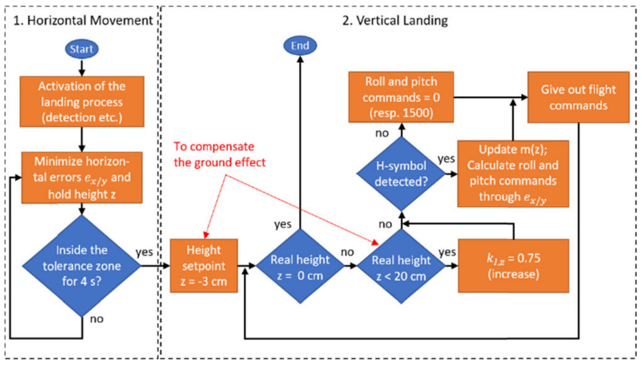

1. Horizontal Movement
2. Vertical Landing
Start
Activation of the landing process (detection etc.)
Minimize horizontal errors $e_{x/y}$ and hold height z
Inside the tolerance zone for 4 s?
yes
no
To compensate the ground effect
Height setpoint z = -3 cm
Real height z = 0 cm
End
Real height z < 20 cm
H-symbol detected?
Roll and pitch commands = 0 (resp. 1500)
Update m(z); Calculate roll and pitch commands through $e_{x/y}$
Give out flight commands
$k_{I,z}$ = 0.75 (increase)

FIGURE 25. Flowchart of the overall landing process test.

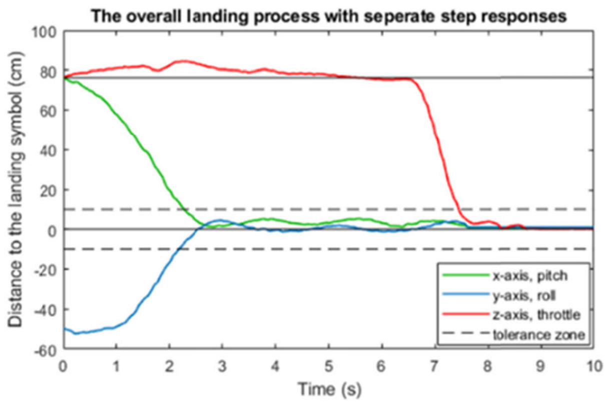


FIGURE 26. Step responses in the overall landing process (separate axes).

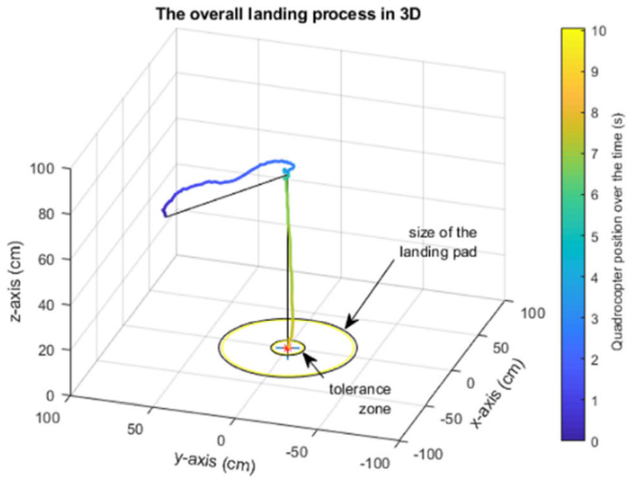


FIGURE 27. Step responses in the overall landing process (3D view).

resulting in an increasing uplift in the general case [26], [27]. However, boosting the integrator gain of the PID controller under a characteristic height of 20 cm proves to be an even more effective (whilst simple) measurement against said problem. During the vertical landing, the horizontal control stays active until the H-symbol can no longer be detected due to too low heights. Thereby, pitch and roll maneuvers are no longer executed under approximately 15 cm. Additionally, the slope m(z) for the pixel-into-cm conversion is adapted actively to the real height. Otherwise the horizontal feedback would scale to unrealistic errors. Fig. 26 and Fig. 27 finally show the plot of such a landing process (the video material also includes a test). The 3-axis position control is sufficiently functional and hence conceptionally transferable to other systems.

VII. CONCLUSION

In the present paper, an automated camera-based landing system for quadrocopters is proposed. It relies on a commercial flight controller handling the attitude control (inner loop), while the 3-axis position control (outer loop operating at 20 Hz) is taken care of by a Raspberry Pi 3B+. Therefore, we upgraded a standard radio-control model with additional hardware, such as a wide-angle camera for horizontal feedback (modified Raspberry Pi Cam v1.3), a servo gimbal, a ToF-height-sensor and a multiplexer. The latter serves as a switch, allowing the pilot to either fly the quadrocopter manually through his remote control or activate the developed autonomous mode, in which the Raspberry Pi generates the flight commands. A PID controller is designed for the height control including a back-calculation anti-windup. In the horizontal plane however, two PD controllers are implemented. Beforehand, several black-box system identifications have been conducted regarding the flight controller's attitude control in order to deduce the relevant system plants and simulations for each controller design. The image processing algorithms generate horizontal feedback by detecting a distinct landing symbol and converting its image coordinates into compliant physical distances. Implementing a region of interest helps us to speed up the computing time for that matter. Generally, the resulting step responses as well as the performance regarding the overall landing process are making up a functional autonomous system.

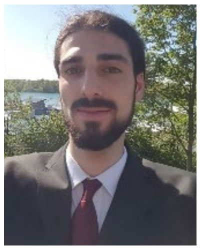
In the future, we plan on replacing the 2D-camera with a depth camera, allowing us to obtain information about the evenness of the ground. Thereby, it would be possible to locate flat surfaces to land on without depending on a landing symbol/pad. This can be especially helpful for any kind of rescuing missions after natural disasters such as earthquakes. Nevertheless, depth cameras require high computational power and suitable algorithms with respect to a sufficiently fast position control. Choosing the right hardware (camera and single-board computer) will have a huge effect on the functionality of the desired system.

REFERENCES

[1] Y. Yamazaki, M. Tamaki, C. Premachandra, C. J. Perera, S. Sumathipala, and B. H. Sudantha, "Victim detection using UAV with on-board voice recognition system," in *Proc. 3rd IEEE Int. Conf. Robotic Comput. (IRC)*, Feb. 2019, pp. 555–559.

[2] C. Premachandra, M. Otsuka, R. Gohara, T. Ninomiya, and K. Kato, "A study on development of a hybrid aerial terrestrial robot system for avoiding ground obstacles by flight," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 1, pp. 327–336, Jan. 2019.

[3] C. Premachandra, D. Ueda, and K. Kato, "Speed-up automatic quadcopter position detection by sensing propeller rotation," *IEEE Sensors J.*, vol. 19, no. 7, pp. 2758–2766, Apr. 2019.
[4] K. Nakajima, C. Premachandra, and K. Kato, "3D environment mapping and self-position estimation by a small flying robot mounted with a movable ultrasonic range sensor," *J. Electr. Syst. Inf. Technol.*, vol. 4, no. 2, pp. 289–298, Sep. 2017.
[5] C. Premachandra and M. Otsuka, "Development of hybrid aerial/terrestrial robot system and its automation," in *Proc. IEEE Int. Syst. Eng. Symp.*, Oct. 2017, pp. 1–3.
[6] C. Premachandra, S. Takagi, and K. Kato, "Flying control of small-type helicopter by detecting its in-air natural features," *J. Electr. Syst. Inf. Technol.*, vol. 2, no. 1, pp. 58–74, May 2015.
[7] Y. Yamazaki, C. Premachandra, and C. J. Perea, "Audio-processing-based human detection at disaster sites with unmanned aerial vehicle," *IEEE Access*, vol. 8, pp. 101398–101405, Jun. 2020.
[8] C. Premachandra, D. N. H. Thanh, T. Kimura, and H. Kawanaka, "A study on hovering control of small aerial robot by sensing existing floor features," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 4, pp. 1016–1025, Jul. 2020.
[9] H. Beck, J. Lesueur, G. Charland-Arcand, O. Akhrif, S. Gagne, F. Gagnon, and D. Couillard, "Autonomous takeoff and landing of a quadcopter," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 475–484.
[10] N. Xuan-Mung, S. K. Hong, N. P. Nguyen, L. N. N. T. Ha, and T.-L. Le, "Autonomous quadcopter precision landing onto a heaving platform: New method and experiment," *IEEE Access*, vol. 8, pp. 167192–167202, Sep. 2020.
[11] K. T. Putra, R. O. Wiyagi, and M. Y. Mustar, "Precision landing system on H-octocopter drone using complementary filter," in *Proc. Int. Conf. Audio, Lang. Image Process. (ICALIP)*, Jul. 2018, pp. 283–287.
[12] N. Q. Truong, P. H. Nguyen, S. H. Nam, and K. R. Park, "Deep learning-based super-resolution reconstruction and marker detection for drone landing," *IEEE Access*, vol. 7, pp. 61639–61655, May 2019.
[13] W. Dong, G.-Y. Gu, and X. D. H. Zhu, "Modeling and control of a quadrotor UAV with aerodynamic concepts," in *World Academy of Science, Engineering and Technology*. 2013, pp. 901–906.
[14] A. E. V. Moreno, "Machine learning techniques to estimate the dynamics of a slung load multirotor UAV system," Univ. Glasgow, Glasgow, U.K., Tech. Rep., 2017.
[15] A. Reizenstein, "Position and trajectory control of a quadrocopter using PID and LQ controllers," Linköping Univ., Linköping, Sweden, Tech. Rep., 2017.
[16] Toshiba. (2019). *TC74HC157AP Datasheet*. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/214509/TOSHIBA/TC74HC157AP_07.html
[17] C. Bohn, "Regelungstechnik (control theory) 1—Lecture script," TU Clausthal, Clausthal-Zellerfeld, Germany, Tech. Rep., 2017.
[18] C. Bohn, "Parameteridentifikation DC-motor und regelung—Laboratory script," TU Clausthal, Clausthal-Zellerfeld, Germany, Tech. Rep., 2018.
[19] S. Jung and R. C. Dorf, "Analytic PIDA controller design technique for a third order system," in *Proc. 35th IEEE Conf. Decis. Control*, vol. 3, Dec. 1996, pp. 2513–2518.
[20] X. Li, J. Park, and H. Shin, "Comparison and evaluation of anti-windup PI Controllers," Gyeongsang Nat. Univ., Gyeongsangnam-do, South Korea, Tech. Rep., 2010.
[21] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997.
[22] C. Bohn and H. Unbehauen, *Identifikation Dynamischer Systeme*. Wiesbaden, Germany: Springer Vieweg-Verlag, 2016.
[23] I. Kugelberg, "Black-box modeling and attitude control of a quadrocopter," Linköping Univ., Linköping, Sweden, Tech. Rep., 2016.
[24] B. Jaehne, *Digitale Bildverarbeitung Und Bildgewinnung*. Berlin, Germany: Springer Vieweg-Verlag, 2012.
[25] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Comput. Vis., Graph., Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.
[26] S. Aich, C. Ahuja, T. Gupta, and P. Arulmozhivarman, "Analysis of ground effect on multi-rotors," in *Proc. Int. Conf. Electron., Commun. Comput. Eng. (ICECCE)*, Nov. 2014, pp. 236–241.
[27] P. Wei, S. N. Chan, and S. Lee, "Mitigating ground effect on mini quadcopters with model," Univ. California Davis, Davis, CA, USA, Tech. Rep., 2019.



MALIK DEMIRHAN received the B.Eng. degree in mechanical engineering (dual study degree program) from the Ostfalia University of Applied Sciences, and Volkswagen AG, Wolfsburg/Wolfenbüttel, Germany, in 2017, and the M.Sc. degree in mechanical engineering from the Clausthal University of Technology, Clausthal-Zellerfeld, Germany, in 2020.

From 2019 to 2020, he was a Research Assistant with the Laboratory for Image Processing and Robotics, Shibaura Institute of Technology, Tokyo, Japan. He is currently a Calibration Engineer in DCT transmissions with Volkswagen AG. His research interests include control theory, mechatronic applications, calibration, measurement technology, and image processing.

CHINTHAKA PREMACHANDRA (Member, IEEE) was born in Sri Lanka. He received the B.Sc. and M.Sc. degrees from Mie University, Tsu, Japan, in 2006 and 2008, respectively, and the Ph.D. degree from Nagoya University, Nagoya, Japan, in 2011.

From 2012 to 2015, he was an Assistant Professor with the Department of Electrical Engineering, Faculty of Engineering, Tokyo University of Science, Tokyo, Japan. From 2016 to 2017, he was an Assistant Professor with the Department of Electronic Engineering, School of Engineering, Shibaura Institute of Technology, Tokyo. In 2018, he was also an Associate Professor with the Department of Electronic Engineering, School of Engineering/Graduate School of Engineering and Science, Shibaura Institute of Technology, where he is currently a Manager with the Image Processing and Robotic Laboratory. His laboratory conducts research in two main fields, such as image processing and robotics. His research interests include computer vision, pattern recognition, speed up image processing, camera-based intelligent transportation systems, terrestrial robotic systems, flying robotic systems, and integration of terrestrial robot and flying robot.

Dr. Premachandra was a steering committee member with many international conferences. He is a member of IEICE, Japan, SICE, Japan, and SOFT, Japan. He received the FIT Best Paper Award from IEICE in 2009 and the FIT Young Researchers Award from IPSJ, Japan, in 2010. He serves as the Founding Chair for the International Conference on Image Processing and Robotics (ICIPRoB). He served as an Editor for journals.