Chapter 1

Autonomous Quadrocopter for Search, Count and Localization of Objects

Nils Gageik, Christian Reul and Sergio Montenegro

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/63568

Abstract

This chapter describes and evaluates the design and implementation of a new fully autonomous quadrocopter, which is capable of self-reliant search, count and localization of a predefined object on the ground inside a room.

A camera attached to the quadrocopter and directed at the ground is used to find the searched objects and to determine its positions during the autonomous flight in real time. Hence, objects that fulfil the scanning parameters can be found in different positions. Based on its own known position and the position of the object in the picture of the camera, the position of the detected objects can be determined. Thus, repeated detections of objects can be excluded. Consequently, objects can be counted and localized autonomously.

The position of the object is transferred to the ground station and compared with the true position to evaluate the system. Two different search situations and two different strategies, breadth first search (BFS) and depth first search (DFS), are investigated and their results are compared. The evaluation shows the potential, constraints and drawbacks of this approach just as the effects of the search strategy, and the most important parameters and indicators such as field of view (FOV), masking area (MA) and minimal object distance. Moreover, the accuracy, performance and completeness of the search are discussed. The entire system is composed of low-cost components and constructed from scratch. Its integration in the innovative real-time operating system RODOS (Real-time Onboard Dependable Operating System) developed by the German Aerospace Centre is described in detail. RODOS has been developed for embedded systems such as satellites and comparable aerospace systems.

Keywords: autonomous UAV, quadrocopter, quadrotor, search and rescue, count, object localization

1. Introduction

Equipping unmanned aerial vehicles (UAVs) such as quadrocopters with more and more autonomous abilities is an interesting field of research. Furthermore, it is a requirement for challenging autonomous search and rescue missions, which are still a field of interest [1–14]. Especially, fully autonomous systems are challenging since they cannot rely on external systems like Global Positioning System (GPS) or optical tracking for accurate positioning. State of the art is the usage of a laser scanner for obstacle detection, collision avoidance and via a simultaneous localization and mapping (SLAM) algorithm for positioning [15, 16]. But laser scanners are heavy, expensive and fail in some situations like a smoking environment. Other approaches are vision based, but the high computational burden often requires an external computer for computation [17–19].

Therefore, a solution for a fully autonomous system is presented using a new hardware design combining optical and PMD cameras with infrared and ultrasonic distance holders for a reliable system capable of search and rescue missions. This chapter focuses on the concept, implementation and evaluation of the search, count and localization of red balls (example search targets) with the mentioned autonomous system based on an innovative new hardware design.

In a preliminary calibration scan, the parameters of the object are defined: a red ball is used as an example object. The scan determines the colour and radius of the ball. The implementation and principles of the object recognition and search will be described in detail. After determining the scanning parameters, the autonomous search can be executed. This is done autonomously by the quadrocopter, which uses inertial, infrared, ultrasonic, pressure and optical flow sensors to determine and control its orientation and position in six DOF (degree of freedom).

This research is part of the AQopterI8 Project of the Chair Aerospace Information from the University of Würzburg [20].

2. Terms and background

To clarify different terms, parameters and algorithms, which will be used later, those are defined in this chapter. The main idea of the presented search approach is that the quadrocopter uses a camera directed at the ground and by flying through the search area; it scans all possible locations on the floor for a target (red ball). If a target is detected, it is added to the list of detected targets unless a target has already been detected at this position. Thus, the whole area can be searched for targets and the amount of targets as well as their positions can be determined.

The most significant parameters for the performance of the search, the virtual field of view (VFOV), the masking area (MA) and the search strategy are investigated, and therefore need to be defined.

http://dx.doi.org/10.5772/63568

In this case, the field of view (FOV) is defined as the area on the floor which the camera covers (**Figure 1**). Using computer vision object detection, a target can be found on this single picture of the floor. The field of view is specified by the camera (hardware), whereas the virtual field of view is the area which the search strategy uses in order to cover the whole search area at least once. For VFOV, a smaller value than the true FOV may be used to leave no room for inaccuracy. Detection might fail if the quadrocopter does not fly exactly as expected by the search strategy or if the target is located between two pathways and cannot be seen completely. A smaller VFOV leads to a higher coverage and a longer search pathway (compare **Figures 2** and **3**).

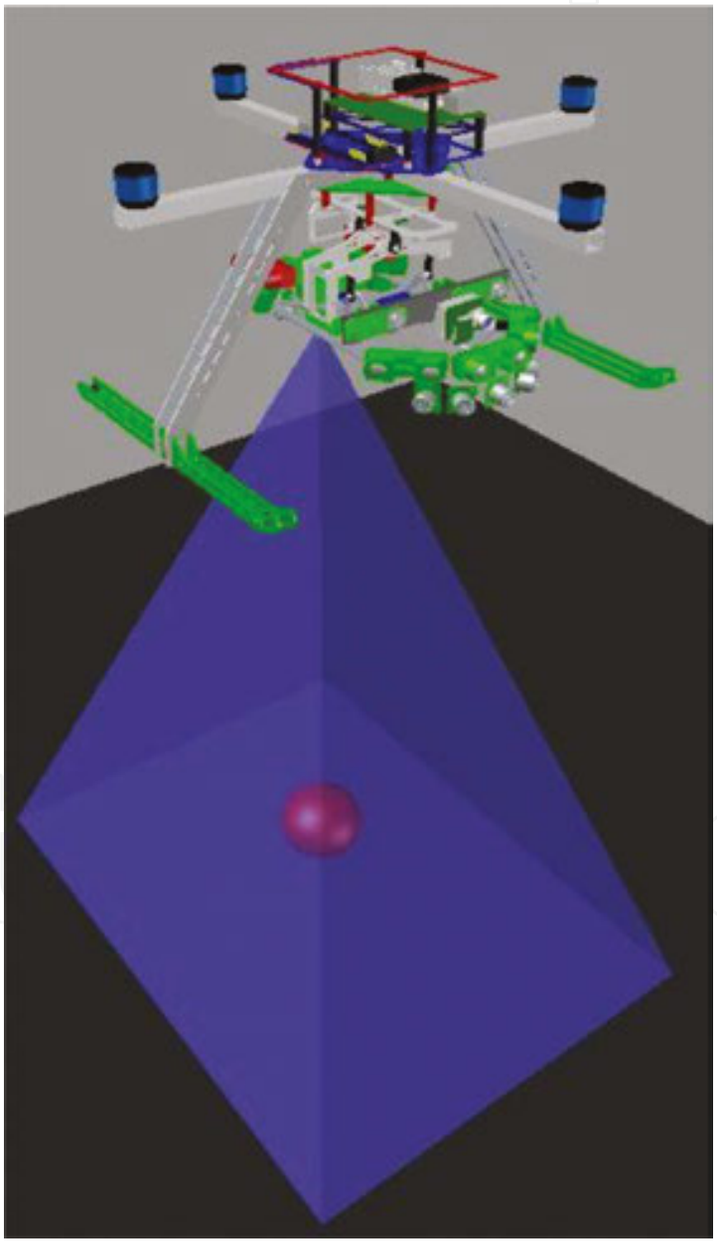

Figure 1. Field of view (FOV).

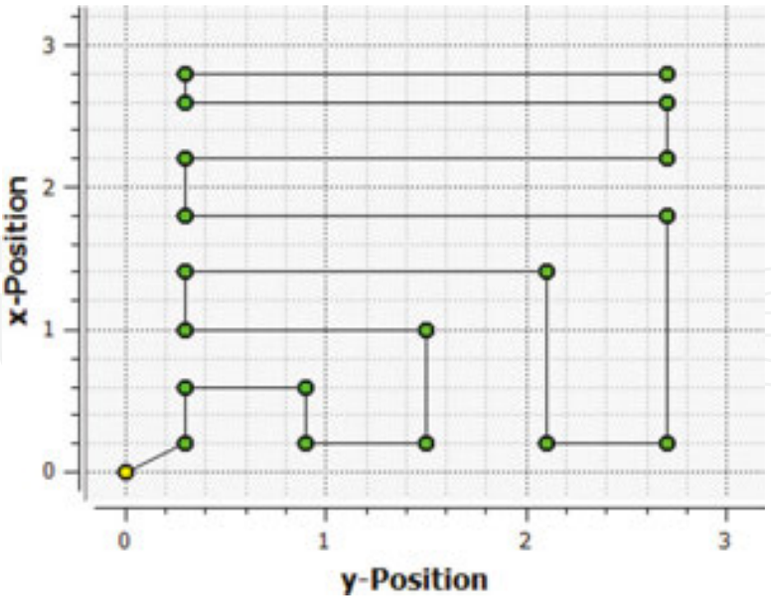

Figure 2. BFS waypoint list (VFOV 40 × 60).

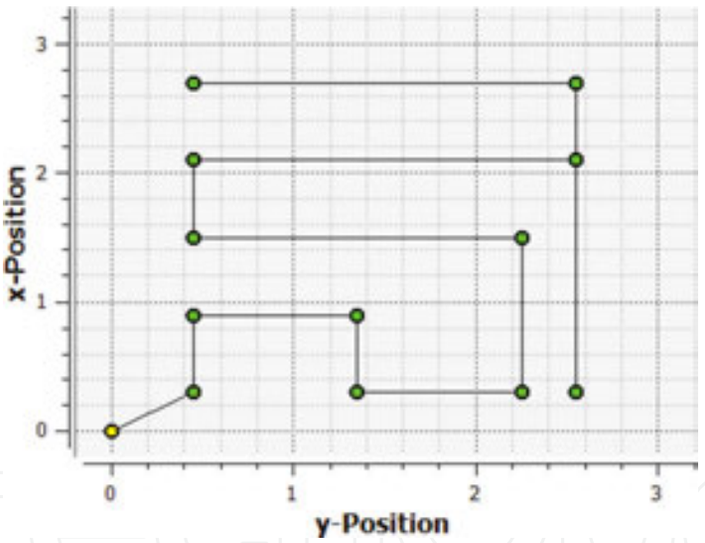

Figure 3. BFS waypoint list (VFOV 60 × 90).

Two different search strategies, which are referred to as BFS (breadth-first-search) and DFS (depth-first-search) later, are investigated. They correspond to the original algorithms, which are used to search nodes in a graph. For reasons of simplification, all search algorithms start at the bottom left corner of the search area.

The idea of the BFS strategy is shown in **Figure 2**. This strategy follows the general rule, which says that closer positions are reached before farther ones. In general, the used algorithm follows the iterative rule Up-Right–Down-Right–Up-Left.

http://dx.doi.org/10.5772/63568

In contrast to the BFS, the DFS does not search nearer but farther positions first. At first, the algorithm covers the sides of the search area and proceeds with smaller iterations until the complete search area is covered (compare **Figures 4** and **5**).

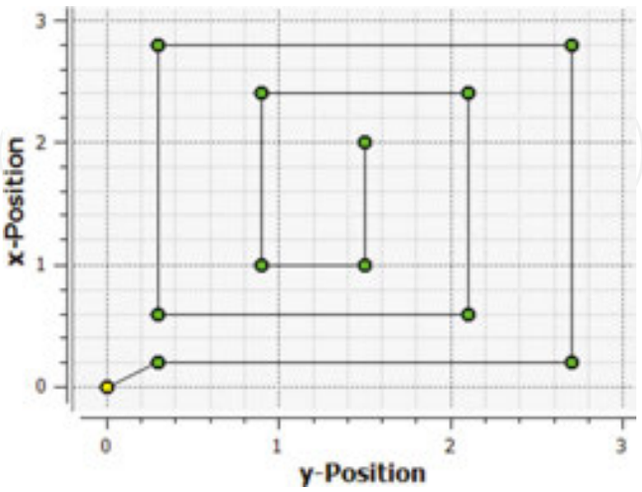

Figure 4. DFS waypoint list (VFOV 40 × 60).

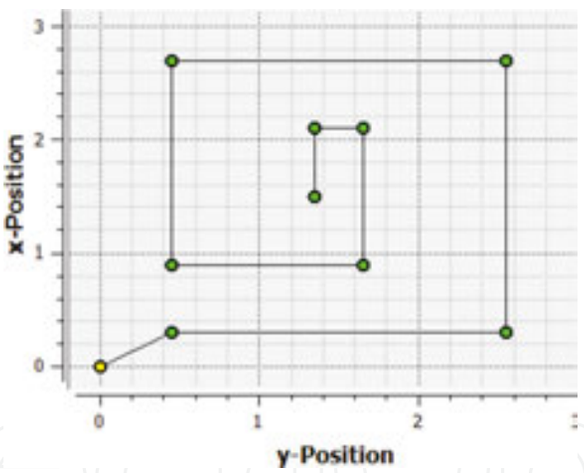

Figure 5. DFS waypoint list (VFOV 60 × 90).

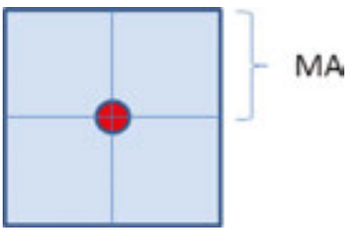

Figure 6. Masking area around an accepted target (red dot).

The masking area (**Figure 6**) determines a square which is set around a found object to avoid multiple detections of the same target. It is determined by a distance named MA. During the search a target might be seen several times from different positions. Because of errors and noise the target is never detected exactly twice at the same position, and therefore would be considered as a new object multiple times. The masking area is subtracted and added to the *X*-coordinate and the *Y*-coordinate of every accepted target and it is proved if the newly detected target is located within one of these coordinates. If so, the newly detected target is discarded, otherwise it is accepted. Instead of a circle a square masking area can be chosen for reasons of simplification and the FOV being also a square.

3. Concept

The concept of the search is separated into two parts: the object or target search and the flight search

3.1. Object search

The task of the object search is to determine the amount and positions of the targets by fusing the results of the object detection with the current position of the quadrocopter (**Figure 7**). It manages the list of found objects and adds new ones if necessary.

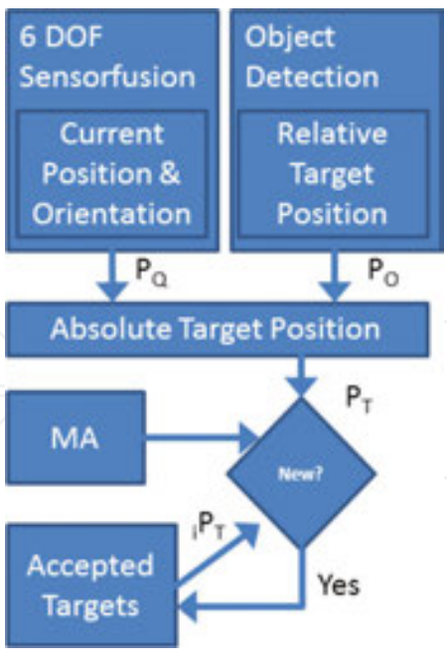

Figure 7. Object search concept.

Whenever the object detection has a hit, the absolute position of this new target is computed by Formula (1),

http://dx.doi.org/10.5772/63568

$$\boldsymbol{P}_T = \boldsymbol{P}_O + \boldsymbol{P}_Q + \boldsymbol{C}_O \tag{1}$$

where $\boldsymbol{C}_O$ is the offset between the camera and the centre of the quadrocopter or its position sensor, $\boldsymbol{P}_Q$ is the current position of the quadrocopter and $\boldsymbol{P}_O$ is the relative position of the found object determined by Formula (2)–(5):

$$\mathrm{P}_O^{\square} = \frac{\mathrm{h}}{C_h} \cdot (\boldsymbol{M} - \boldsymbol{Z}) \cdot \boldsymbol{C}_w \tag{2}$$

$$\boldsymbol{M} = \begin{bmatrix} \frac{M_x}{R_x} & \frac{M_y}{R_y} \end{bmatrix} \tag{3}$$

$$\boldsymbol{Z} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \tag{4}$$

$$\boldsymbol{C}_w = \begin{bmatrix} C_x & 0 \\ 0 & C_y \end{bmatrix} \tag{5}$$

M_x and M_y in Formula (2) are the coordinates of the object's centre point determined by the object detection, C_x and C_y are the calibration width in X and Y, respectively, at a height of C_h, h is the current height and $\boldsymbol{Z}$ is a constant. C_x and C_y are determined by the true FOV of the camera. R_x and R_y are the resolution of the camera in the X - and Y-directions, respectively.

Next, the position P_T is compared with all positions ${}_iP_T$ with i indicating the index of the already accepted position. If the new position P_T occurs within the masking area of any target ${}_iP_T$, it is discarded, otherwise it is accepted.

3.2. Flight search

The task of the flight search is the waypoint generation. It ensures that the quadrocopter with a determined VFOV covers the whole search area at least once. The VFOV is a static parameter, which represents the FOV. A bigger FOV leads to less waypoints and a shorter flight search, while with a smaller FOV more waypoints and resulting pathways are created. In general, waypoints are not generated next to each other iteratively in small steps because of the bad flight performance of this approach [21], but with maximal distance according to the search strategy.

Then, the flight search is executed statically. That means the waypoint list is generated once at the beginning and it is not changed during the flight. The waypoint list is determined by

the search strategy, the search area and the VFOV. **Figures 2–5** illustrate the effect of these parameters on the waypoint list.

4. Object recognition

The algorithm for object recognition is based on significant information about an object's shape and colour and determines if it is a target or not [22]. The implemented search algorithm (object recognition) expects targets, which are round and monochrome, such as red balls. Extending the system to enable a detection of more than one target colour at the time can easily be achieved. To provide the possibility of searching for other shapes like rectangles or human bodies the object recognition needs to be replaced or changed fundamentally. For the experiments described in this chapter, red balls with a diameter of approximately 7 cm are considered as search targets.

In the following sections, the necessary image processing fundamentals will be briefly discussed. Afterwards, the circle detection algorithm used to identify the balls will be introduced. Finally, the recognition procedure consisting of an initial scan to determine the search parameters and a subsequent search will be explained.

4.1. Image processing fundamentals

For implementation and guidance, the open source computer vision library OpenCV can be recommended [23]. It contains a variety of basic image processing core algorithms as well as advanced procedures for applications such as object recognition, feature extraction and machine learning.

4.1.1. Image representation

A standard way to represent a picture while using a PC is the RGB model. Each pixel is described by three intensity values: *r*ed, *g*reen and *b*lue. Here we assume a resolution of 8 bit. Therefore, the range for each value is 0–255 ($=2^8-1$).

For a bench of calculations, the RGB representation of a pixel is impractical and a single value per pixel is preferred. Using the so-called greyscale image the value grey of a pixel x can be determined by its original RGB values (Formula (6)):

$$grey(x) = 0.299 \cdot x.R + 0.587 \cdot x.G + 0.114 \cdot x.B \tag{6}$$

The simplest representation is the binary image in which only two values exist: 0 and 1. If a pixel meets certain requirements, for example if it has a specific RGB or a greyscale value, a value of 1 is assigned, otherwise it has a value of 0.

http://dx.doi.org/10.5772/63568

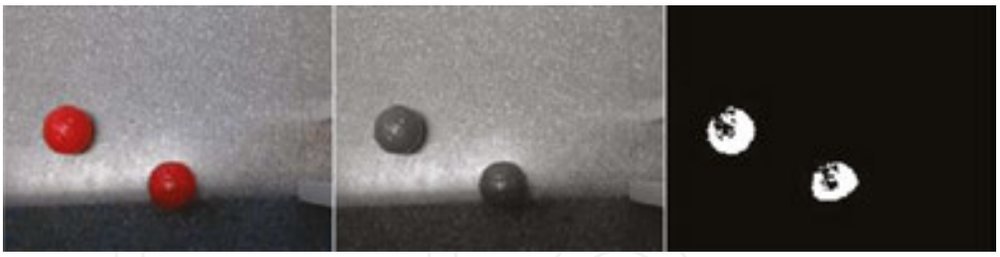

Figure 8. Image of two balls using its RGB (left), greyscale (middle) and binary (right) representation.

An example image of two balls and the corresponding greyscale image can be seen in **Figure 8**. Furthermore, a binary image was created by assigned value 1 to each pixel with a red value higher than 100 and with green and blue values lower than 50.

4.1.2. Filters

In contrast to the operations already introduced, filters use a variety of pixels and not just a single one to determine the new value of a pixel. The idea behind the filters is to perform 2D convolution: a so-called filter matrix is slid over the original image and simple multiplications of the filter elements with the underlying values of the image pixels are performed. The calculated outcomes are summed up and the result is stored as the new value of the pixel, which is located under the so-called hot spot, the centre of the filter matrix. The entire process is shown in **Figure 9**.

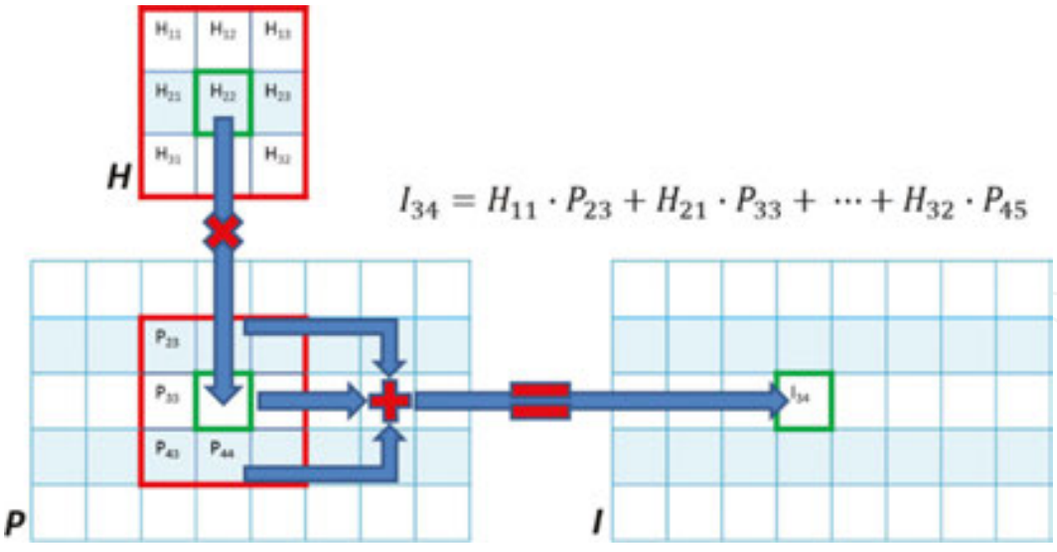

Figure 9. Mode of operation of a linear filter [24].

4.1.3. Edge detection

One of the most frequent applications of filters is edge detection [24]. Edges can be defined as regions in which big intensity changes occur in a certain direction. To detect those changes one

or several filters have to be applied to the greyscale image. In most applications, the so-called Canny edge detector is used [25]: after deploying a Gaussian filter in order to remove noise, the intensity gradients are computed by applying the Sobel operator, which consists of two separate filter matrices. The first filter computes the gradient in the x-direction:

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{7}$$

The second one highlights the change of intensity in the y-direction:

$$H_Y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{8}$$

The local edge strength E can then be calculated by combining the resulting images D_x and D_y for each pixel (u, v):

$$E(u,v) = \sqrt{\left(D_x(u,v)\right)^2 + \left(D_y(u,v)\right)^2} \tag{9}$$

Furthermore, the local edge orientation angle $\Phi(u, v)$ can be determined as

$$\Phi(u,v) = \tan^{-1}\left(\frac{D_y(u,v)}{D_x(u,v)}\right) \tag{10}$$

In general, the described procedure leads to blurry edges. Therefore, an edge thinning technique called non-maximum suppression is applied. The computed first derivatives are combined into four directional derivatives and the resulting local maxima are considered as edge candidates.

Finally, a hysteresis threshold operation is applied to the pixels. Two thresholds have to be defined: an upper one and a lower one. If the local edge strength of a pixel is higher than the upper one, the pixel immediately is accepted as an edge pixel. Pixels whose gradient is below the lower threshold are rejected. If the local edge strength is between the lower and the upper threshold, only pixels adjacent to pixels with gradients above the upper threshold are accepted. This process promotes the detection of connected contours. In this work, values of 20 and 60 were used for the lower and upper thresholds, respectively.

http://dx.doi.org/10.5772/63568

4.2. Hough circle transformation

A circle is defined by its centre $C(x_C, y_C)$ and its radius r. All points $P(x_P, y_P)$ on the outline of the circle satisfy the circle equation:

$$(x_C - x_P)^2 + (y_C - y_P)^2 = r^2 \tag{11}$$

Identifying circles from an edge image by using this equation and the simple approach of checking for every centre candidate, how many edges lie on a circle around it, is very inefficient and highly inadvisable. In the following sections two much faster and more robust approaches are presented.

4.2.1. Basic method

The basic idea behind the Hough transformation can be seen in **Figure 10**. Given a target circle with radius r. If circles with the same radius r are drawn around an edge point of the target circle, they will intersect. The main accumulation of intersection will occur in the centre of the target circle.

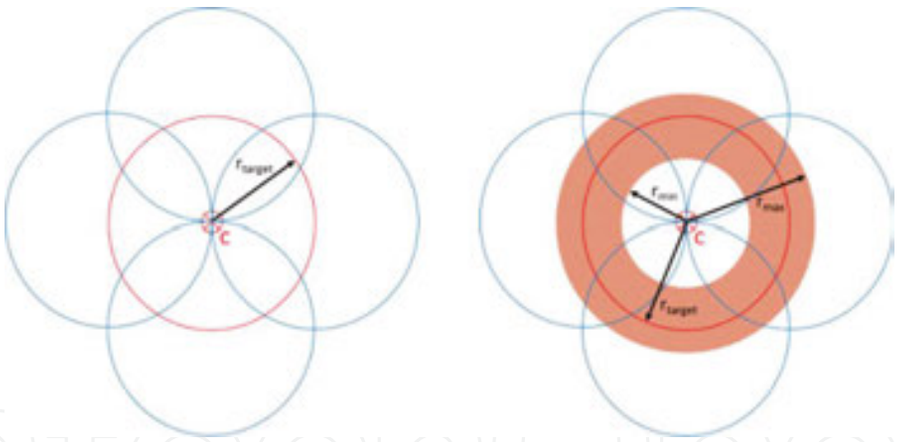

Figure 10. Intersecting circles (blue) drawn around the edge pixels of the target circle(s) (red) using one radius (left) and a range of radii (right) [26].

To identify a circle with known radius r in an edge image, a so-called accumulator array is used. Typically, it has the same dimension as the edge image or is scaled down by a low integer number. If the target radius is exactly known, a two-dimensional array is sufficient. After initializing every cell with zero the voting process starts. Each edge pixel is treated as a possible circle outline and all corresponding centre candidates in the accumulator array get a vote. This means that their value is incremented by 1. After voting all detected circles can be identified by checking, in which cells in the accumulator array earned enough votes. Consequently, a threshold is needed. As the value of each cell roughly corresponds to the number of circle

outline pixels in the edge image, a useful threshold can be derived from the maximal number of votes, i.e., the circle's circumference [25].

However, in real-world application the target radius often is not exactly known and the algorithm has to search for a range of radii. This is implemented by extending the accumulator array to three dimensions: two for the already described two-dimensional arrays and one for each radius. During the voting process each edge pixel votes for all possible circle centres by incrementing the corresponding values in every radius plane.

It can easily be seen that the standard approach is not suitable to most real-world applications despite being very robust. First of all, it is slow because for every edge pixel approximately $2\pi r$ centre candidates have to be calculated per radius r. Another problem is that the accumulator arrays can be very memory intensive, especially if the input edge image resolution is high and the target radius is not exactly known. Hence, several improvements were introduced and will be discussed in the following section.

4.2.2. Gradient method

As shown in Formula (10), the local edge orientation angle can be easily determined. By exploiting this, the circle detection algorithm can be executed with much higher efficiency. The key observation is that all edges are perpendicular to the line that connects the edge pixel and the centre of the circle. Therefore, it is not necessary to calculate up to $2\pi r$ centre candidates for each edge pixel and vote for them in the accumulator array. Because of the edge orientation angle the amount of candidates can be narrowed down to only a few pixels. This is shown in **Figure 11**.

The vertical line in the centre of **Figure 11** shows the respective local edge orientation. If the radius is exactly known, in theory only two possible centres correspond to the given edge direction: C_1 and C_2. They are located on a line perpendicular to the edge direction and their distance to the considered edge pixel is r. If the algorithm is searching for a range of radii, the sets of possible centres $C_1[\,]$ and $C_2[\,]$ are located on aforementioned line and the distances of the centres to the initial edge pixel vary from r_{min} to r_{max}.

Hence, the accumulator array can be reduced to two dimensions even when searching for several radii [27]. During the voting process the location of each edge pixel that casts a vote is stored. After the vote the centre candidates are selected. To be taken into consideration the accumulator value of a valid centre candidate has to be above the given threshold and higher than the values of all its immediate neighbours. The approved centre candidates are sorted in descending order according to their accumulator values.

Now the best fitting radius has to be determined. For this, the previously stored edge pixels are considered. The distances between each of these pixels and the centre candidates are calculated. Using these distances, the best supported radius can be determined. Finally, it has to be checked, if the resulting centre is not too close to any previously accepted centre and if it is supported by a sufficient number of edge pixels.

http://dx.doi.org/10.5772/63568

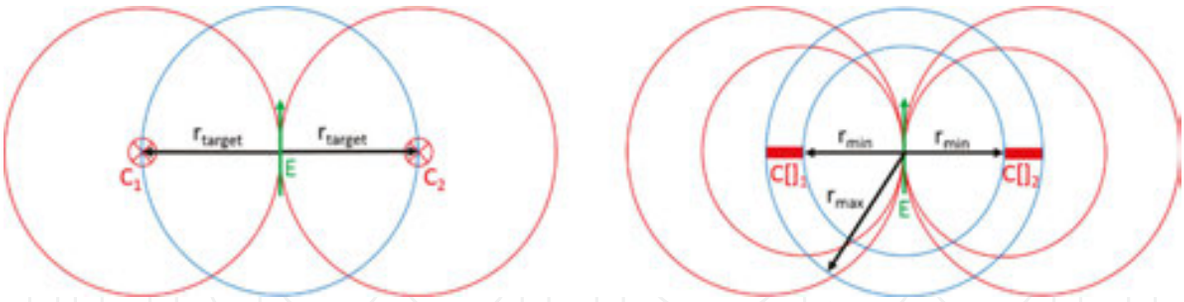

Figure 11. Possible locations of centres given a specific edge direction (green) using one radius (left) and a range of radii (right) [26].

4.2.3. Run-time comparison

In Ding *et al.* [22], the run times of the described basic method and the gradient method are compared by detecting red balls in an image using a resolution of 192 × 144 pixel. The achieved results were averaged over 10 measurements and are displayed in **Table 1**.

	1 ball	**5 balls**
Basic method	35.2 ms	63.4 ms
Gradient method	12.0 ms	12.5 ms

Table 1. Results of the run-time comparison.

When only one ball is detected, the gradient method is already about three times faster than the basic method. The difference becomes even more significant when the number of balls and therefore the amount of edges is increased.

4.3. Recognition procedure

The recognition procedure is split into two phases: the initial search to determine the target parameters pre-flight and the actual search which is performed mid-air by the quadrocopter.

4.3.1. Initial scan

Prior to the search, some parameters have to be predefined. Thus, a picture of the search target is taken on a plane background and from a height close to the flying height of the quadrocopter. The radius can be directly determined by detecting the ball and storing the radius, in which the maximum number of votes in the accumulator array was achieved.

Furthermore, the dominating colour within the detected circle is calculated by combining several of the most frequent red, green and blue values. It is notable that the average values are not used because they can be heavily influenced by bright spots on the search target which emerge because of unfavourable lighting conditions. The final target colour does not consist of exactly one set of RGB values but of ranges for each colour channel which are derived from the original values.

Furthermore, the algorithm searches for more than one radius. Therefore, the initially detected radius ±2 can be chosen as a target range to compensate for light variations of height during the flight. To allow bigger chances in flying altitude, the radius range would have to be adjusted according to the currently measured distance of the quadrocopter to the floor.

4.3.2. Search

The actual search is performed using a resolution of 192 × 144 pixels. This allows quick processing while still preserving all the information necessary for a successful detection.

After taking a picture it is converted into a greyscale image and the Hough detection is performed. The number and quality of detected circles heavily depend on the threshold used during the Hough circle detection. A good value for the required number of votes is 30% of the circumference of the smallest radius. With significantly higher values, target objects tend to get missed far too often because the constant movement of the quadrocopter tends to prevent the camera from taking sharp pictures.

All detected circles, i.e., all target candidates, are then analysed for their colour. For each pixel inside a candidate's enclosing circle, it is checked if its RGB values lie within a certain range. The range is determined during the initial scan. For a candidate to get confirmed as a target, a certain percentage of its pixels has to be target pixels. Setting this threshold to about 40% was a good value here.

5. Hardware design

Figure 12 depicts the hardware design of the quadrocopter (**Figure 13**). The brain of the system is composed of two processing units, an AVR 32 bit MCU (microcontroller unit) UC3A and the LP-180 providing an AMD-x86 processor and 2 × 1.6 GHz system clock [28].

The CPUs can be seen in the centre of the picture. The MCU interfaces all sensors except those connected via USB and performs the control part with real-time computing, while the task of the LP-180 contains all functions with a high computational burden such as object recognition and mapping.

The quadrocopter uses a couple of sensors for obstacle detection and is capable of distance-controlled collision avoidance [29]. For object recognition, the C270 camera from Logitech is used [30]. All processing is done on-board the quadrocopter, so it is capable of a fully autonomous flight.

http://dx.doi.org/10.5772/63568

Figure 12. Hardware design.

Figure 13. AQopterI8 picture.

6. Software implementation

6.1. Overview and background (RODOS)

The underlying software problem with multiple, here three, real-time processing units interacting with each other is a typical application of the real-time operation system RODOS (Real-time Onboard Dependable Operating System). An important aspect in the selection of RODOS is its integrated real-time middleware. Developing the control and payload software on the top of a middleware provides the maximum of modularity. Different functions can be developed independently and simultaneously and it is very simple to interchange modules later without worrying about side effects because all modules are encapsulated as building blocks (BB) and they interact only through well-defined interfaces.

RODOS was originally developed for space applications at DLR (German Space Agency) and is now distributed as open source for many applications such as Robotics [31, 32]. RODOS was designed for application domains demanding high dependability (e.g., space) and targets the

irreducible complexity in all implemented functions. It follows the quote 'Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.' from Antoine de Saint-Exupery.

The quadrocopter firmware—ranging from attitude control to route planning, and payload software, e.g., identification of objects—is implemented as a (software) network of communicating building blocks. A useful feature of the RODOS middleware is the location transparency of building blocks. BBs can interact and communicate in the same way independently of the location of communication partners. This includes the same computer, a different computer in the same vehicle, on a different vehicle or between vehicles and ground station (operator interface).

RODOS was designed as the brain of the Avionic system and introduced for the first time (2001) the NetworkCentric concept. A NetworkCentric core avionics machine consists of several harmonized components, which work together to implement dependable computing in a simple way. In a NetworkCentric system we have a software network of BBs and a hardware Network interconnecting vehicles (radio communication), computers inside of vehicles (buses and point-to-point links), intelligent devices (attached to buses) and simple devices attached to front end computers. To communicate with external units, including devices and other computing units, each node provides a gateway to the network and around the networks several devices may be attached to the system. The communication is asynchronous using the publisher–subscriber protocol. No fixed communication paths are established and the system can be reconfigured easily at run time. For instance, several replicas of the same software can run in different nodes and publish the result using the same topic without knowing each other. A voter may subscribe to that topic and vote on the correct result. Application can migrate from node to node or even to other vehicles without having to reconfigure the communication system. The core of the middleware distributes messages only locally, but using the integrated gateways to the NetworkCentric network, messages can reach any node and application in the network. The communication in the whole system includes software applications, computing nodes and even IO devices. Publishers make messages public under a given topic. Subscribers (zero, one or more) to a given topic get all messages which are published under this topic. As mentioned before, for this communication there is no difference in which node (computing unit or device) a publisher and subscribers is running, and beyond this they may be any combination of software tasks and hardware devices! To establish a transfer path, both the publisher and the subscriber must share the same topic. A topic is a pair consisting of a data type and an integer representing a topic identifier. Both the software middleware and the hardware network switch (called middleware switch) interpret the same publisher/subscriber protocol.

6.2. Software design

A simplified RODOS-based software design of the AQopterI8 quadrocopter can be seen in **Figure 14**. The different BBs exchange services by middleware topics. These BBs are located on three different CPUs, the on-board MCU UC3A, the on-board x86 PC LP-180 and the ground

segment (GS) with an off-board CPU. The GS provides the user with a GUI and is used for commanding (**Figure 15**).

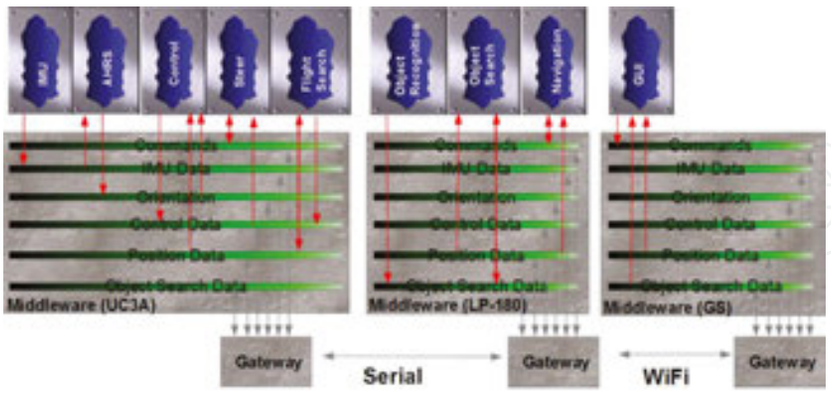

Figure 14. RODOS-based software design (simplified).

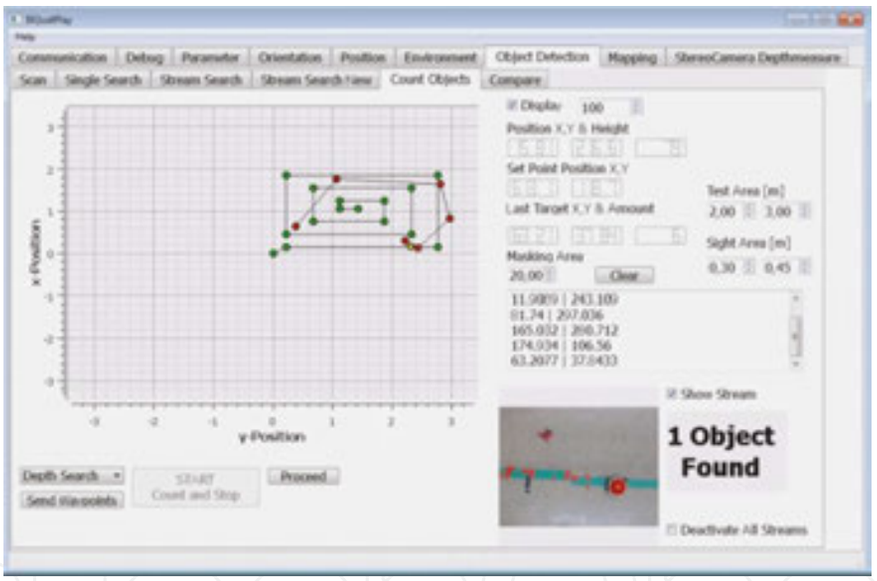

Figure 15. I8Quatplay (Qt-based Commanding Software GUI).

The IMU BB updates the IMU readings every 10 ms with already calibrated and conditioned sensor values. The attitude heading reference system (AHRS) computes from these data the 3D orientation of the quadrocopter using a complementary quaternion filter. The control BB performs the six degree of freedom (DOF) position control based on the position and orientation given by other BBs. The Steer BB drives the motors and executes the commands of the operator and navigation.

The position is further required by the Object Search BB and sent via the gateway using the serial communication link as well as to the GS via WiFi. Thanks to the Gateway and Middleware of RODOS, these data can be used in the same way on another device as on the same

device. The kernel of RODOS provides support for thread execution, time management, synchronization and transparent access to external devices such as sensors by the HAL (hardware abstraction layer) interface.

7. Evaluation

7.1. Overview evaluation

To investigate the performance, accuracy and limitations of the proposed system and to compare both search strategies (DFS, BFS) as well as to discuss the optimal parameters for the masking area and VFOV, the results of 63 experiments from two setups are presented.

7.2. First setup

The first setup contained 42 experiments. The search area consisted of a 3 m × 2 m square with two randomly placed balls at the positions (50, 50) and (240, 140) according to **Figure 16**.

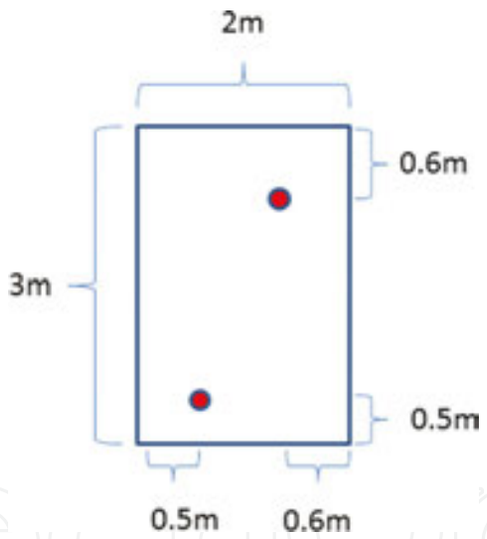

Figure 16. First setup.

In this setup the experiment was repeated for both search strategies, BFS and DFS, for four different masking areas with MA = 0.1 m, MA = 0.15 m, MA = 0.2 m and MA = 0.3 m and with three different VFOV: 0.3 m × 0.45 m, 0.40 m × 0.60 m and 0.60 m × 0.90 m. Then the computed position of the target was compared with the manually measured one, supposed to be the true position. For every single parameter setting, the average error dx in the X- and dy in the Y-direction was computed, first over both targets and then over the entire run together. Also the number of double detections D (fail positive) and misses M (fail negative) was counted (**Table 2**). In the second run, the experiments for MA = 0.3 m were skipped because MA = 0.2 m showed no problem in this setup.

http://dx.doi.org/10.5772/63568

Detection failures		DFS		BFS	
		M	D	M	D
VFOV	30–45	0	0	0	3
	40–60	0	1	1	0
	60–90	4	0	2	1
MA	10	2	1	1	3
	15	0	0	1	1
	20	0	0	1	0
	30	2	0	0	0

Table 2. Detection errors first setup: M missing and D double detections.

From these data no clear difference in accuracy between DFS and BFS or between the different parameter settings could be identified, but it could be concluded that the average error in one axis is less than 15 cm. This setup of randomly placed balls is predominantly affected by coincidence. It might be that one setting leading to one flight path fits well to the placement of the balls.

By taking a look at the detection failures (**Table 2**), clear conclusions can be made. The real FOV is about 65 cm × 45 cm and it can clearly be seen that a VFOV of 40 cm × 60 cm or higher leads to misses. The bigger the VFOV is, the more misses occur, as expected. A proper VFOV of 30 cm × 45 cm leads to no misses for both search strategies. The data show that a lower MA can lead to double detections. This is the case because a target might be seen several times. As the position error in one direction is about 15 cm, MA should be at least in the same range. Conclusively, it can be seen that the DFS performed better and also that there is still a dominating systematical error.

7.3. Second setup

Based on the outcome of the first setup, in the second setup more balls were placed to reduce the effect of coincidence. In addition, the search area was changed to a 2 m × 3 m square (**Figures 17** and **18**), which aimed to equalize the results between the two search strategies and to improve the results of the BFS. This time positions were selected, which should cause troubles for all settings (**Table 3**).

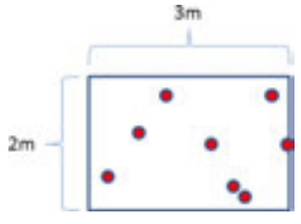

Figure 17. Second setup.

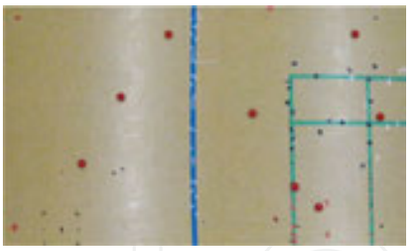

Figure 18. Picture of second setup from above.

Detection failures		DFS		BFS	
MA	**VFOV**	**M**	**D**	**M**	**D**
15	**25–35**	**Skipped**		**2**	**4**
	30–45	**0**	**2**	**2**	**0**
	40–60	**0**	**0**	**1**	**0**
	60–90	**2**	**0**	**4**	**0**
20	**25–35**	**Skipped**		**3**	**2**
	30–45	**0**	**0**	**1**	**0**
	40–60	**0**	**0**	**2**	**0**
	60–90	**2**	**0**	**4**	**0**
30	**25–35**	**Skipped**		**2**	**0**
	30–45	**0**	**0**	**1**	**0**
	40–60	**1**	**0**	**1**	**0**
	60–90	**3**	**0**	**4**	**0**

Table 3. Detection errors second setup: M missing and D double detections.

Figure 19 depicts the results shown in the QT Control-Software for a run with the settings MA = 20 cm and 30 × 45 cm for VFOV. It demonstrates that for these settings all targets were detected properly.

The second setup more clearly showed the effect of each parameter or setting and underlined the already expected results. More targets reduced the effect of coincidence, and therefore one run was seen to be enough.

The average position error for the DFS was 16 cm and for the BFS it was about 20 cm. According to these data the DFS can already be concluded as more accurate, but a clearer distinction between both search strategies can be made by taking the detection failures into account. For the DFS there are 10 detection errors in nine experiments compared to 20 detections errors of the BFS in the same setup. Considering these bad results, a value of 25 cm × 35 cm for VFOV

http://dx.doi.org/10.5772/63568

was tested with the BFS, but this led to even worse results. There is no setting for the BFS without detection error, but there are four settings with no detection error for the DFS.

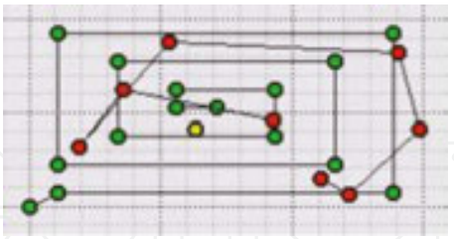

Figure 19. GUI picture of search result (20–30–45): red: found targets; green: waypoints; yellow: position.

7.4. Summary evaluation

To sum up, it can be said that all settings, the search strategy, the masking area and the VFOV have a significant effect on the performance of the search. Although still other, partly random parameters and circumstances have an important influence on the result, optimal values of these parameters are required. This is underlined by **Figures 20–22**, which show that the DFS with MA = 20 cm and a VFOV of 30 cm × 45 cm or 40 cm x 60 cm detected eight balls exactly and nothing else mistakenly. This means there exist settings, which solved this challenging setup. It shall be mentioned that an MA of 30 cm led to a miss in one of the two cases because

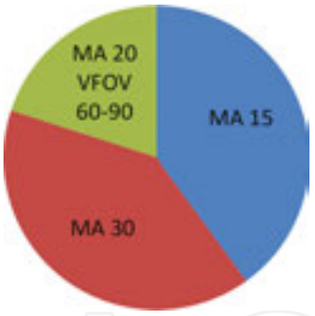

Figure 20. DFS detection failures (general distribution).

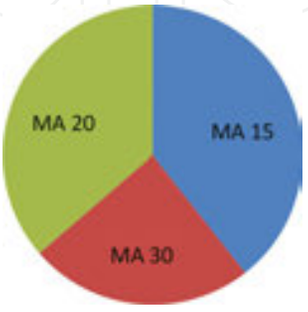

Figure 21. BFS detection failures (distribution after MA).

then one of the closest balls, which are 20 cm away, was not accepted. The BFS showed no good results here and only a VFOV of 30 cm × 45 cm and of 40 cm × 60 cm led to acceptable results. Altogether these results also made the BFS look worse than it was. Some balls were positioned in such way that the BFS failed them by a few centimetres.

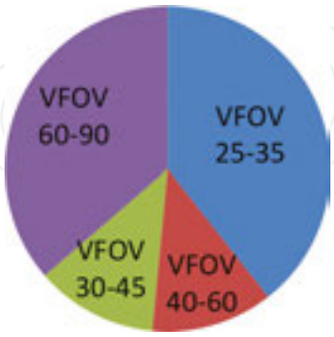

Figure 22. BFS detection failures (distribution after VFOV).

8. Conclusion and discussion

The evaluation demonstrated that the system is capable of autonomously detecting, counting and localization of objects with an accuracy of about 15–20 cm. It was proven that an optimal value for MA (20 cm) has to be a bit higher than the accuracy of the position system and that objects with the distance of 20 cm (MA) in each axis can still be distinguished. Also the coherence of the parameters MA and VFOV on the performance of the search and the detection errors was demonstrated. A smaller VFOV with a smaller MA leads to more double detections, while a too high MA leads to misses of nearby objects. As a general rule, too high VFOV leads to misses because some areas are not searched properly. In this context the acceptance tolerance, which was set to 25 cm in setup 2, is a parameter, which comes into effect. A waypoint is already marked as reached if the current position of the quadrocopter is within this tolerance. This can result in an incomplete cover of the search area and it explains why the BFS misses some targets at the side of the search area.

The best parameter for VFOV was 30 cm × 45 cm. This setting together with the best value for MA showed no detection error even in a challenging room with eight objects.

Furthermore, the evaluation proved that the DFS performed better than the BFS. The reason for that is the fact that smaller waypoint steps are less accurate than less big ones because of the set point jumps and the jump effect as well as the control and sensor system. These result simplified mean that also for a flying robot such as a quadrotor using the underlying on-board sensors less turns and commands are better. This could already be demonstrated in previous experiments [21]. The reason for that can be found in the dynamic of the quadrotor as an aerial vehicle with very little friction (air) and the on-board optical sensors, which are especially affected by the behaviour of the system. Rotations, which mainly occur after set point changes, are a source of error for the position determination.

Although the system was proven capable of performing autonomous and challenging search, count and localization missions, the evaluation of the system did not show a very high accuracy according to the determined positions and the fact that optical sensors were used, which generally can reach higher accuracies. There are multiple sources for accuracy errors, which start from the manually measured and placed target positions in a region of several centimetres. The next major source of error is the starting error, which means the wrong position measured by the optical flow during lift off and the wrong initial position and orientation or placement error of the quadrocopter on the starting position. An initial orientation error for yaw of only 1° leads to a position error of 5 cm after 3 m. It is most likely that the initial yaw orientation error was sometimes in the range of a few degrees. These are good explanations for the high systematical error, which can be seen in the data. A proof of this fact is given by a closer look at some raw data. They demonstrate that the accuracy for the closer object is much better than for the farer object, even if the closer object is detected later in some cases. The best explanation for this is an initial yaw orientation error or missing alignment.

In general, it can be concluded that for this setup proof of high accuracy is challenging and the accuracy of the system might be better than the data show, but at the same time this is not the presented work.

Other sources of error are wrong calibration values for the relative position of the detected object Po (Formula (2))) and simplifications of Formula (2), an incorrectly measured height, a wrong scaling factor for the optical flow and bad lighting and surface conditions, which lead to position errors measured by the optical flow sensor.

The current orientation of the quadrocopter is not considered in the computation of the position P_T. This was intended because the effect of an orientation error should be excluded from the evaluation. In some cases this led to double detection errors.

9. Perspective

Although the system performed quite well in general, there is potential for optimization. The effect of the already mentioned acceptance tolerance and an improved procedure for the waypoint navigation would allow higher values for VFOV. Furthermore, the system can be improved by using two phases. In the first phase, the object search just tries to find something with a low resolution reducing the computational burden and increasing the possible sample time. The focus of the first phase is to overlook nothing. If it has a hit, the quadrocopter suspends the waypoint search and flies to the position of the hit. Then, the second phase is executed using a high resolution and accuracy and only in this phase the accepted position is determined. Computational burden is unimportant in the second phase because the quadrocopter is on position hold.

A different approach with a moving camera and flexible height could also be investigated. In this case, the quadrocopter would possibly not need to search the whole area or at least the waypoint list could be much smaller. In our setup, the quadrocopter could simply fly 4 m up

and could see the complete search area. But that will not be possible in every situation as usually rooms are not that high. However, it needs to be compared that which accuracies and detection performance could be achieved then. Taking obstacles and unknown limitations into account as well as the fact that objects might not be detected properly from a distance and at an angle, this approach is much more sophisticated, but also offers more potential and might save flight time, and therefore could reduce the energy consumption.

Another interesting improvement would be to use the obstacle detection sensors to improve the position computation, and therefore the accuracy of the localizations. A challenging part here is a reasonable distribution of the limited resources of the LP-180 to the different demanding tasks.

Acknowledgements

The author would like to thank Universitätsbund Würzburg, IHK Mainfranken, Simone Bayer, Barbara Tabisz and Sascha Dechend for their support and help on this work.

Author details

Nils Gageik[2*], Christian Reul[1] and Sergio Montenegro[2]

*Address all correspondence to: nils.gageik@uni-wuerzburg.de

1 Artificial Intelligence and Applied Computer Science, University of Würzburg, Würzburg, Germany

2 Aerospace Information Technology, University of Würzburg, Würzburg, Germany

References

[1] Nonami K., Autonomous Flying Robots, Springer. 2010, ISBN-10: 4431538550

[2] Aibotix GmbH, www.aibotix.de

[3] Microdrones GmbH, www.microdrones.com

[4] ArduCopter, http://code.google.com/p/arducopter

[5] HiSystems GmbH, www.mikrokopter.de

[6] Mellinger D. et al, Trajectory generation and control for precise aggressive maneuvers with quadrotors, The International Journal of Robotics Research, 31(5), 2012.

[7] Gageik N., Mueller T., Montenegro S., Obstacle detection and collision avoidance using ultrasonic distance sensors for an autonomous quadrocopter, UAVveek 2012.

[8] Autonomous Flying Robots, University Tübingen, Cognitive Systems, www.ra.cs.uni-tuebingen.de

[9] Gronzka S., Mapping, state estimation, and navigation for quadrotors and human-worn sensor systems, PhD Thesis, Uni Freiburg, 2011.

[10] Lange S., Sünderhauf N. Neubert P., Drews S., Protzel P., Autonomous Corridor Flight of a UAV Using a Low-Cost and Light-Weight RGB-D Camera, Advances in Autonomous Mini Robots, 2012, ISBN: 978-3-642-27481-7

[11] Ding W. et al, Adding Optical Flow into GPS/INS Integration for UAV navigation, International Global Navigation Satellite Systems Society, IGNSS 2009.

[12] Wang J. et al, Integration of GPS/INS/VISION Sensor to Navigate Unmanned Aerial Vehicle, The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII Part B1, Beijing 2008.

[13] Blösch M.et al., Vision Based MAV Navigation in Unknown and Unstructured Environments, Robotics and Automation (ICRA), 2010 IEEE International Conference, 21–28.

[14] Corke P., An Inertial and Visual Sensing System for a Small Autonomous Helicopter, Journal of Robotic Systems 21(2), 43–51, 2004.

[15] Dryanovski I., Valenti R., Xiao J., Pose Estimation and Control of Micro-Air Vehicles, August 2012.

[16] Grzonka S. et al, A Fully Autonomous Indoor Quadrotor, IEEE Transactions on Robotics 28(1), 2012.

[17] Kendoul F. et al, Optical-flow based vision system for autonomous 3D localization and control of small aerial vehicles, Robotics and Autonomous Systems 2009, Elsevier.

[18] Herisse B. et al, Hovering flight and vertical landing control of a VTOL Unmanned Aerial Vehicle using Optical Flow, 2008 IEEE International Conference on Intelligent Robots and Systems.

[19] Zing S. et al, MAV Navigation through Indoor Corridors Using Optical Flow, 2010 IEEE International Conference on Robotics and Automation.

[20] Gageik, Reul, Montenegro, Autonomous Quadrocopter for Search, Count and Localization of Objects, UAV World 2013.

[21] Gageik N., Strohmeier M., Montenegro S., Waypoint Flight Parameter of an Autonomous UAV, IJAIA, 4(3), May 2013, pmdtechnologies GmbH, www.pmdtec.com

[22] Reul, C., Implementierung und Evaluierung einer Objekterkennung für einen Quadrocopter, BA Thesis, April 2013.

[23] Itseez. OpenCV [Internet]. Available from: http://www.opencv.org [Accessed: 30.01.2016]

[24] Burger, W. and Burge, M. Principles of Digital Image Processing—Fundamental Techniques. 1st ed. London: Springer; 2009. 260 pp.

[25] Rhody H. Lecture 10: Hough Circle Transform [Internet], 2005.

[26] Smereka M.. Duleba, I. Circular object detection using a modified hough transform. International Journal for Applied Mathematics and Computer Science. 2008; 18(1):85–91.

[27] Bradski G., Kaehler A. Learning OpenCV. 1st ed. Sebastopol, CA: O'Reilly Media; 2008. 555 pp.

[28] Commell, www.commell.com.tw.

[29] Gageik N., Benz P., Montenegro S. Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors. IEEE Access 3: 2015; 599–609.

[30] Logitech, http://www.logitech.com.

[31] Rodos Wiki, wikipedia.org/wiki/Rodos_(operating_system).

[32] Rodos Framework, DLR Webpage, www.dlr.de/irs/desktopdefault.aspx/tabid-5976/9736_read-19576/.