Adaptive Path Planning for a Vision-Based quadrotor in an Obstacle Field

J. Junell*, and E. van Kampen†
Delft University of Technology Aerospace Engineering, Control and Simulation
Delft, The Netherlands.

ABSTRACT

This paper demonstrates a real life approach for quadrotor obstacle avoidance in indoor flight. A color-based vision approach for obstacle detection is used to good effect conjointly with an adaptive path planning algorithm. The presented task is to move about a set indoor space while avoiding randomly located obstacles and adapting a path to prevent future confrontation with the obstacles all together. The goal is to complete this task with a solution that is simple and efficient. The result is an adaptive path planning algorithm that evades obstacles when necessary and uses these interactions to find an obstacle-free path with simple logic. The whole task is implemented within Paparazzi, an open source autopilot software. Flight tests are performed in an indoor flight arena with simulated GPS from a camera tracking system. Through these flight tests, the approach proves to be reliable and efficient.

1 INTRODUCTION

A wide range of applications for Micro Aerial Vehicles(MAVs) has created a demand for smaller vehicles with greater autonomy. Along with a decrease in size comes the need for lighter or fewer sensors and efficient utilization of the sensors onboard. Autonomy of the vehicle in many applications requires the ability to navigate in unknown environments and decide on paths or trajectories that ensures the safety of the vehicle in addition to the completion of the task at hand.

Collision avoidance is a broad topic and the problem approach inherently changes with differences in vehicle size, sensors available, type of aerial vehicle, and properties of the obstacles to avoid. In a survey of Collision Avoidance Systems (CAS), Albaker et al., has categorized all approaches into one of five key CAS functions; they are roughly: *sensing the environment*, *conflict detection*, *awareness*, *escape trajectory*, and *maneuver realization* [1]. From this all-encompassing view of CAS, this paper explores: 1) The combined functions *conflict detection and awareness* using a color-based detection method, and 2) The function *escape trajectory* using a simple adaptive path planning scheme within an obstacle field. This paper uses the terminology *Obstacle detection* and *Guidance*, for these 2 functions respectively.

Camera sensing is desirable because cameras are a lightweight sensor option and many MAV applications already require the video output as part of the main task. Detection of obstacles using vision methods has made much progress in MAV applications. Optical flow uses texture in the sensed environment to track points over time, ascertaining information about the distance of the lens from these points [2, 3, 4]. Another vision-based detection approach is stereo vision which uses two images with a known offset to create depth perception and recognize objects which are closer than the background [5, 6]. This approach is best used with a binocular camera setup, but can be implemented with a monocular vehicle by taking two side by side pictures with the same heading and a known offset in the vehicle position [7]. Finally, simultaneous localization and mapping (SLAM) has been used to fully map a room [8, 9], however not every task benefits from such a complex and information rich approach.

Approaches for guidance in collision avoidance tasks depend greatly on system and environment details. Looking only into static-obstacle avoidance and using a vehicle without communication with other agents there are a few guidance strategies popular among researchers. E-Field methods treat all agents/obstacles as a charged particle and model the repulsive forces between them. These forces drive the actions of the vehicle [1, 10]. Optimized, genetic, or learning approaches [11, 12] focus on finding optimal paths or policies. These can be computationally extensive and may still not guarantee convergence to an optimal path.

The conflict resolution guidance approach presented in this paper is categorized as one of predefined logic which is based on a set of predefined rules to avoid collisions. While these types of approaches do not attempt optimization, they are simple and therefore fast acting for collision avoidance. This research's specific guidance law takes advantage of the use of waypoints which is a strength of the chosen autopilot software. In addition, the logic is such that the planned path will adapt to obstacle placement and will find an obstacle-free path without the need of computationally heavy tasks of mapping or storing any obstacle information.

*j.l.junell@tudelft.nl. PhD Student
†Assistant Professor

This paper presents an adaptive path planning logic which is then tested on a quadrotor micro aerial vehicle with a monocular camera sensor. The color filtering approach for obstacle detection is presented in Section 2 and is shown to be highly reliable for particular obstacles. The adaptive path planning and collision avoidance scheme is presented in Section 3. The complete system and experimental setup is detailed in Section 4. The results from test flights in different configurations are shown in Section 5. Finally, conclusions are drawn and discussed in Section 6.

2 Obstacle Detection

A color-based vision algorithm was used to identify objects within the quadrotor's $90°$ field of view. This approach is limited to obstacles of a chosen color which is distinguishable from the background, however the approach is very reliable within these limitations. This method was selected due to its reliability in practice and its ease of use as there is already a color filtering module implemented in the Paparazzi software for the AR Drone's camera (See Section 4.1.1 for details of the ARdrone quadrotor used for this research). Other vision approaches which are more robust to diverse obstacles could be attempted in place of this, including edge detection [13], optical flow [4, 3], or a combination method [13, 14]. Each of these vision approaches will have its own challenges, strengths, and weaknesses. The important thing to note is that the setup for this application is such that it would be easy to take another approach and attempt to implement it within the modular structure of the Paparazzi software.

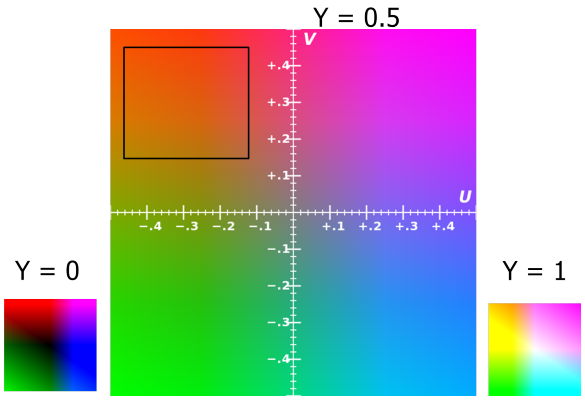

Figure 1: Color filtering detects colors within a specific range.

The algorithm works with the YUV images coming from the camera, downscaled to one quarter size for faster processing. The filter checks, pixel by pixel, if the three channels (one carrying the brightness and the remaining two the color) lie in the intervals defining the color shades of the obstacles. Figure 1 demonstrates the range interval of the color to detect.

The color filtering is already implemented in the AR.Drone 2 vision module of the Paparazzi software. The result is a binary image where "$1's$" mark the regions of the chosen color. Supposing first there is only one obstacle in the image, its position can be determined by finding the centroid of the detected pixels. As the vehicle approaches the obstacle, the number of detected pixels will increase. Thus, this number can be used as a metric of distance to the obstacle.

In practice, however, multiple obstacles can (and often will) be seen by the camera. Because the obstacles can be of any shape and can overlap, it would be difficult to detect them one by one. Thus, a different approach is taken. The image is split into five segments (Figure 2**a**, **b**) and the detected pixels are counted in each of them. If this value is greater than a certain threshold, that segment is considered blocked; otherwise it is free (Figure 2**c**). Finally if there exists three free neighboring segments then the quadrotor flies in the direction of the middle segment of the three (Figure 2**d**). In case of multiple options, the $0°$ heading has the priority. In situations where two or less neighboring segments are free (meaning there is not enough space for the drone between the obstacles), the drone turns by $+90°$ and repeats the detection.

The threshold value needs to be tuned depending on the size of the obstacles. Too low values will result in constant turning, as most segments will always be blocked. On the other hand, the threshold needs to be low enough, so that the thinnest obstacle is detected still at a safe distance.

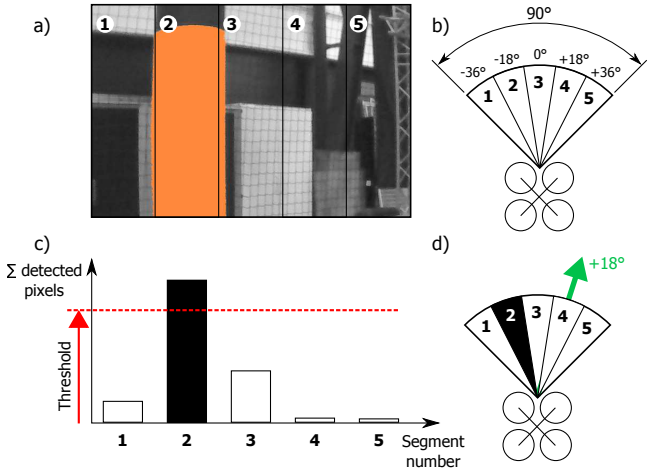

Figure 2: Obstacle detection based on color filtering: **a**) Video frame split into five segments, **b**) each segment corresponds to one fifth of the field of view, **c**) blocked segments are determined by a pixel threshold, d) a safe heading is planned if at least three neighboring free segments exist; otherwise the drone turns by $+90°$ and repeats the procedure.

3 Guidance

In an unknown environment with randomly place obstacles, it is likely that an exploring vehicle will encounter an obstacle and need to avoid it. However, once those obstacles

have been detected and circumnavigated it would be advantageous to learn from the encounter and prevent taking that same path. Having to detect and avoid the obstacle again will only take more time than is necessary. Mapping of the entire flight area and/or storing information about the obstacle location in order to optimize a route takes a lot of computational resources. In this section, a guidance law is presented which learns from past obstacle encounters in order to plan an obstacle-free path without the use of obstacle information or complex algorithms.

The guidance approach was designed to be 1) simple to implement, 2) fail-safe against hitting obstacles, and 3) efficient in the sense that it would adapt to create an obstacle-free path. The final algorithm plays to the strengths of the Paparazzi software as it uses waypoints and logic within a flight plan in order to execute commands to the quadrotor. For more information on the paparazzi autopilot, see 4.1.3.

3.1 Adaptive path planning

The guidance concept is illustrated in Figure 3. An initial path is determined a priori without knowledge of the obstacle locations. This path and waypoint placement can be based on a specific task, for example: maximizing total distance traveled, traveling along the perimeter of an area, or trying to explore as much of the area as possible. Choosing initialization parameters will be discussed shortly in subsection 3.2.

Without knowing the environment, the initial path will likely have some points of collision with the obstacles. Although an obstacle may be detected earlier, the vehicle will continue on its path until an obstacle is nearby. An obstacle is determined "nearby" when the front segment and one of its adjacent segments are blocked. At this point, collision is within 1-2 meters if the obstacle is not thinner than approximately 20 cm. At this point, the vehicle will be commanded to stop and move the waypoint it was heading towards to a new location in a safe region. The waypoint will be placed 150cm in the direction of the safe heading output of the vision module and a predefined distance. If there is no safe heading or the waypoint is placed out of bounds, the vehicle will yaw and search for an acceptable place for its new waypoint. The logic is such that a last resort will be to go back the way it came and therefore should never get stuck in a corner.

Techniques for waypoint placement in difficult scenarios are put in place so that an in-bounds, safe heading can be found as quickly as possible. If the quadrotor avoids an obstacle and the resulting waypoint is placed outside the boundary, the vehicle will turn away from the boundary back towards the obstacle and search for another safe heading on the other side of the obstacle. If the logic were left here, there would be instances in the corner or in areas with many obstacles near the boundary where the vehicle could get stuck in yawing one way and then another in an infinite loop. To combat this worse case scenario, a counter, i, is used to count the number of times a waypoint is placed out of bounds. Each time a wp is placed out of bounds, $42 * i$ degrees will be turned (in the same direction). This will ensure that a $360°$ turn will be made if necessary. The value $42°$ was chosen rather than a value that divides into 360, so that if multiple turns were necessary to find a safe heading, the quadrotor camera will get as many viewpoints as possible.

The path will eventually converge to an obstacle-free path. In the example in Figure 3, it only takes two loops through the waypoints to come up with a converged path. If the next diagram in red is drawn by connecting the waypoints, it can be seen there are no longer any collision paths.

Managing special cases:

No safe zone is detected The quadrotor rotates its heading clockwise until it finds a safe area.

Waypoint is placed outside of a boundary The quadrotor searches for a safe zone on the other side of the obstacle by rotating incrementally in the opposite direction of the safe heading and searching again.

Between an obstacle and a corner There can be a case in a corner or when surrounded by obstacles where it will detect no safe zone, rotate clockwise, then rotate counter clockwise when it encounters a border. This has been addressed by setting a counter and incrementally turning more degrees each time a border is detected. A whole $360°$ turn is guaranteed if it is necessary. In a worst case scenario, the quadrotor can always go back the way it came.

There are two obstacles nearby each other The quadrotor will stop further away from obstacles because it merely detects if a threshold is surpassed. When a new waypoint is place only 2 meters away, it may not be enough to completely circumnavigate the obstacles and will result in the need for 2 waypoints needing to be relocated. This problem would be bypassed with a reasonable estimate of how far away the obstacles are. Then a variable waypoint displacement distance could be used.

3.2 Initial waypoint settings

As mentioned previously, the initial path and waypoint placement is chosen a priori and should be based on a specific task. In this example, the quadrotor could be following a mail route avoiding trees and sign posts; or it could be patrolling specific spots for crime. In another configuration, it could be trying to explore and video as much of the area as possible, in which case the path would cover more of the total area. The waypoint placement should reflect that of the task.

Secondly, the number of waypoints must be chosen. As shown in Figure 3, there are waypoints placed on top of each other. Two waypoints are placed at the end of a journey along the side of the square, and three waypoints are placed at the

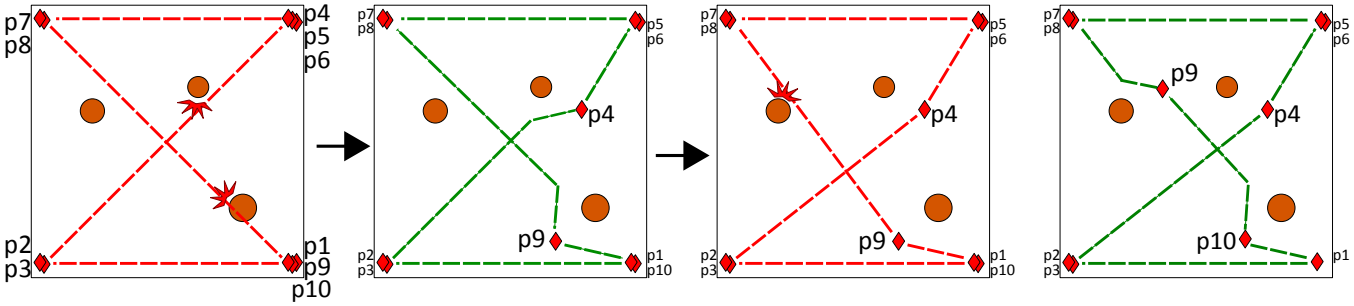

Figure 3: The path that an unobstructed vehicle would fly is in red and the actual path flown is in green. An unobstructed quadrotor would fly looping between waypoints. (eg. $p1 \rightarrow p2 \rightarrow p3... \rightarrow p_n \rightarrow p1...$). However, when an obstacle is detected in the path, the quadrotor stops and the waypoint it is currently moving toward is relocated to a safe zone N meters away from its current position. The obstacle is circumnavigated as the quadrotor continues to loop through the waypoints. This will happen each time an obstacle is encountered. The next round through the waypoint looping, the path will have changed so as to divert from previously encountered obstacles all together. However, the change can put other obstacles in the resultant path. The process then repeats. Given that there are enough waypoints created at the beginning, an obstacle-free path will be created.

end of a diagonal journey. The more distance there is between two points, the greater the chance of multiple obstacle encounters, and the more need for waypoints to create a proportionally complex path. Therefore, an obstacle rich environment will require more waypoints to capture a complicated path to a desired point. However, it is not so simple as to just put a hundred waypoints in each location. If that spot is truly inaccessible, there should be few enough waypoints so that they will be moved away from that spot and the vehicle will stop trying in vain to get there.

4 Flight Test Setup

4.1 Resources

This experiment utilized several resources which will now be described. For a more in depth description of these resources in a previous experiment, reference Junell et al. [11]. The most up to date information can be found at the websites in the footnotes.

4.1.1 AR-Drone 2 quadrotor

Since designing and building a quadrotor is not the main research intention for this experiment, it is very welcome to have a commercial quadrotor available which makes this research not only more convenient for this project but also more accessible to other researchers in the field. A quadrotor is an unmanned aerial vehicle consisting of 4 engines that spin independently of each other. Adjusting the thrust produced by each engine gives the necessary control to perform general motion as well as other aggressive maneuvers. The Parrot® AR-Drone 2 [1], is a commercially available quadrotor intended for recreational use. The AR-Drone 2 is also promising for research and educational purposes [15] as it is relatively inexpensive, and robust to damage. The AR-Drone features a forward facing camera, GPS, WiFi, and precision control. Most importantly for this project is that its built-in software can be overwritten.

Figure 4: The AR-drone 2 quadrotor MAV. Photo credits by Parrot.com

4.1.2 TU Delft Cyber Zoo

The TU Delft Cyber Zoo [2] was officially opened in March 2014 as a research and test laboratory for ground robots and aerial vehicles. Its space consists of a floor area that is 10 x 10 meters and extends 7 meters high. The space is contained inside an aluminum truss frame which holds nets to ensure the safety of people and surrounding equipment. Also supported on the truss structure are 24 cameras. These cameras are part of the *Optitrack* [3]*: Motive Tracker* optical tracking system. The quadrotor "wears" a special hull with reflective balls so that the cameras can track it. Within the Cyber Zoo boundaries, up to 32 rigid bodies can be tracked with high

[1] Parrot AR-Drone 2. http://ardrone2.parrot.com/

[2] TU Delft Robotics Institute. http://robotics.tudelft.nl

[3] Optitrack™. http://www.naturalpoint.com

precision accuracy. The system can also be used as a simulated GPS. Therefore, as long as an experiment is not using any feedback from the Cyber Zoo, everything done indoors can be translated to an outdoor environment.

This flight test will use the tracking system only for simulated GPS and for evaluating performance. The complete system as it is used is illustrated in Fig. 5. The computer running the OptiTrack software has a wired connection to the laptop running Paparazzi and from there can broadcast the GPS signal via WiFi to the quadrotor.

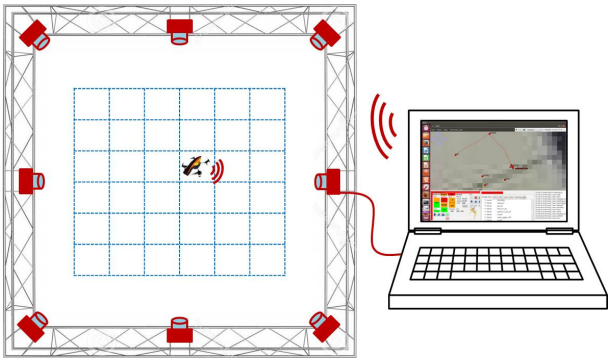

Figure 5: Cyber Zoo experimental setup: 24 cameras track the quadrotor. From this system, the quadrotor receives a simulated GPS signal. That position is transmitted via WiFi between a laptop and the quadrotor. Based on the location state and the memory state, a waypoint command will be sent to the quadrotor for it to execute.

4.1.3 Paparazzi Open Source Autopilot

Paparazzi [4] is a fully open source free autopilot for fixed wing and multi-copter UAVs [16]. A version of the software can be acquired at GitHub [5].

The quadrotor capabilities in Paparazzi include a model of a standard quadrotor that is close enough to the AR-Drone to function as a good inner loop controller given good tuning and standard non-aggressive maneuvers. Simulation of a flight plan in an intuitive GUI is available in addition to use as a Ground Control Station (GCS) for real flights. An example of the Paparazzi GCS interface can be seen in Fig. 6.

In the indoor setting, Paparazzi connects via WiFi to the quadrotor to give commands and via a wired connection to the Cyber Zoo computer to receive the position of the vehicle. In an outdoor situation, GPS would be picked up by the AR-Drone and used there directly.

Paparazzi is a modular platform and therefore has made it easy to add functionality by means of custom modules. Initializations, periodic and event functions can be added without effecting the main autopilot software. This makes customization very flexible and safe.

The high level position control of the vehicle is controlled by the flight plan. Waypoints and commands involving these waypoints are written into the flight plan. The autopilot software and the flight plan is compiled and uploaded onto the quadrotor computer. Once uploaded, the names and numbers of the waypoints and types of commands cannot be changed; however, the location of the waypoints can be modified at any time. Therefore, it is important to plan the method for which the waypoints will be used. Two new modules were created in this experiment. The first module processes the video from the forward facing camera on the AR-Drone and output the binary 5-tuple obstacle detection output as described in Section 2. The guidance is a set of commands which work together with the flight plan to move the waypoints based on predetermined logic.

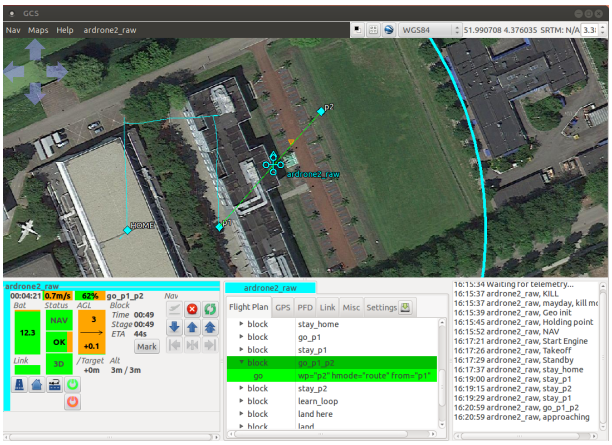

Figure 6: Paparazzi GCS (Ground Control Station) is the main interface for the user. In the GUI it shows a 2D map with a flight path (cyan line). The quadrotor status is shown in strips box (bottom-left) along with some buttons for common commands. Commands such as “go p1” or “stay p2”, can be seen in the Flight Plan tab in the Notebook frame (bottom-middle). Commands that have been made are printed in the console box (bottom-right).

4.2 System Setup

The controller consists of the Paparazzi autopilot with a vision module which continuously outputs the 5-tuple binary detection output representing the obstacles detected in each segment, and a guidance module consisting of functions used in the paparazzi flight plan. Once the software is uploaded to the quadrotor computer, the user can send commands from the GCS flight planvia WiFi to the quadrotor. By calling the block “Loop Route p2”, the quadrotor will begin a loop from p1 to p2 to p3, etc. Within each of these blocks are exceptions which will be triggered if an obstacle is detected and in the path of the quadrotor. If the exception is triggered, then the

[4] Paparazzi Free Autopilot. http://wiki.paparazziuav.org
[5] Paparazzi GitHub account. https://github.com/paparazzi/paparazzi

quadrotor will stop, move the waypoint it was headed towards to a safe zone and continue towards it.

Paparazzi's inner loop controller, which is already uploaded onboard, will translate the waypoint command to specific low level control commands.

Figure 7 demonstrates the communication within the paparazzi controller. The vision module receives input from the camera and the GPS signal. It processes the images, and outputs information that will be used by both the flight plan in its exception block, and by the guidance module to determine the next action should an obstacle be encountered. The GPS signal is also used by many functions in the paparazzi software. The guidance module uses this information to determine the placement of waypoints and heading. The guidance module and flight plan, along with other standard functions within the software, provide the commands to the quadrotor which ultimately moves it to the desired heading and location.

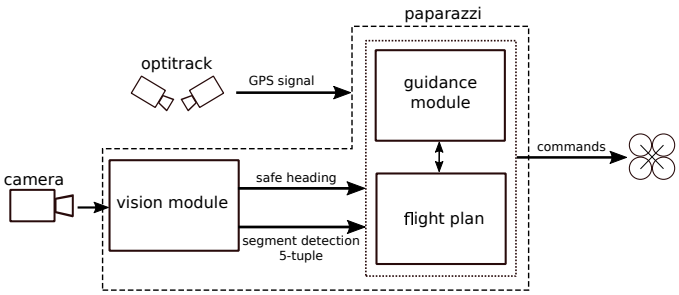

Figure 7: Communication between the modules and the flight plan within the paparazzi autopilot.

5 Flight Results

The results will now be presented and briefly discussed.

5.1 Easy configuration: Three obstacles spaced apart

The first test is configured just as the example from Section 3. Figure 8, shows the tracks of the flight test as received from the Optitrack tracking system. The final waypoint and obstacle positions are shown on the image.

The waypoints original configuration doesn't create an exact square because the original waypoints were placed in latitude/longitude coordinates based on the trackable area of the arena. The idea is still the same. It is shown on the tracks that the first approach toward an obstacles stops, backs up, and reroutes around the obstacle. The next route goes directly to the newly placed waypoint off to the side of the obstacle. This is seen easily in the routes from $p8 \rightarrow p9$. There are clearly 3 different angles leaving $p8$ from the three different positions that $p9$ was at.

The backtracking away from the obstacle is not seen for the $p4$ placement and the waypoint was placed far before the obstacle. This is because the obstacle was detected very early during the quadrotors flight from $p3 \rightarrow p4$. It is not known why the obstacle was detected from so far away, as it should not have been. One possibility is that a false positive detection was caused by seeing other obstacles during a quick yaw maneuver which, for an instant, triggered the detection conditions. This behavior is not necessarily undesirable since having a false positive detection does not result in a crash and in this case there was no harm done since the early waypoint placement still resulted in the desired effect.

5.2 Moving configuration: Adapting to a moved obstacle

The second configuration begins from the previously learned first configuration. As the the quadrotor is flying in its newly learned obstacle-free path, obstacleX is moved and placed in its path. The results show that the quadrotor successfully detects the obstacle and circumnavigates it in the same way as before. The result is a new obstacle-free path.

6 Discussion and Conclusion

This paper demonstrates a simple and effective obstacle avoidance approach within an indoor obstacle field. Past research is discussed for vision based obstacle detection and obstacle avoidance guidance tasks. The experimental approach and setup is presented in detail, and the results of the real-life flight tests are visualized. The results show that an obstacle field can be navigated and adapted to if an obstacle were to change location. However, limitations still do exist.

The benefits and limitations of this approach will now be itemized:

Benefits

- Both vision and guidance is simple to implement
- Does not use resources to map or store information about the obstacle location
- Converges to an obstacle-free path
- Color-vision detection is very reliable
- Logic implemented for special cases so that an inbound and safe heading can always be found if it exists
- If a spot is not accessible, waypoints will be moved and vain attempts to reach that spot will be abandoned

Limitations

- Constant waypoint displacement distance is used, thus wider obstacles would not be circumnavigated in one iteration of waypoint movement, but could take several.
- Number of waypoints must be chosen and placed a priori based on the desired task and number of anticipated obstacles.
- Limitations of the vision approach regarding type of obstacles:

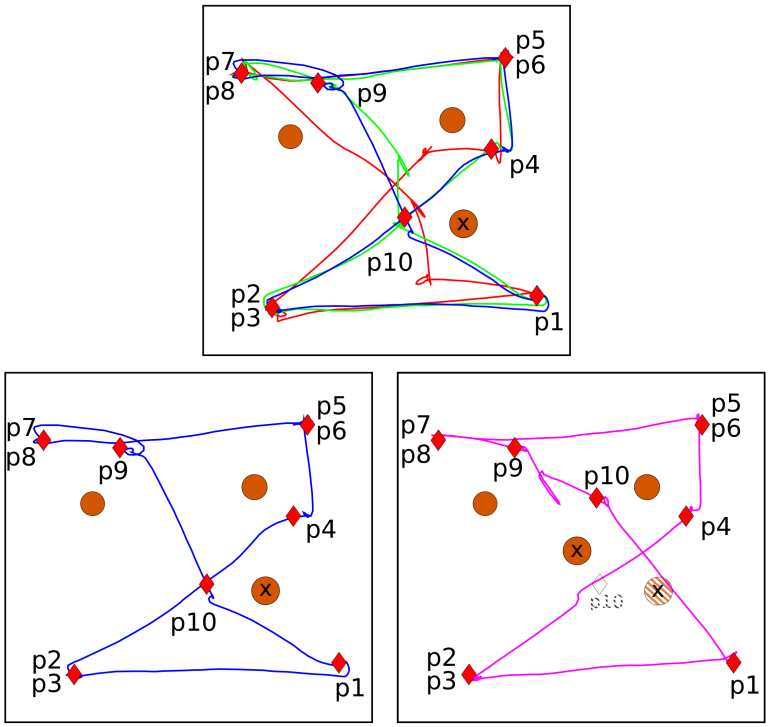

Figure 8: Tracks of the quadrotor during flights 1 & 2 from a birds-eye view. The final waypoint and obstacle positions are shown on the image. The waypoints are depicted as diamonds and are labeled. The obstacles are depicted as orange circles. *(Top figure):* Test 1 includes 3 rounds before a final adapted path is found. The first pass is in red, the second in green, and the third in blue. *(Bottom-left):* the final path learned in Test 1. *(Bottom-right):* After the path was solved in test 1, an obstacle was misplaced into the path. The path adapts to this new configuration. The tracks from the flight are shown along with the final waypoint positions.

- All obstacles must be of a similar (known) color and differentiable from background
- Cannot acquire information about obstacle width or distance

In order to address the limitations and to, for example, avoid obstacles of varying colors and texture, a more mature vision approach would need to be used. Luckily, much progress is being made in the realm of vision for guidance. With access to obstacle information such as width and distance, many of the guidance limitations could be eliminated.

For this research, the color-based vision approached works well. The necessary information is delivered reliably to the guidance algorithm for effective functionality. The guidance approach would only get better if more knowledge of the obstacles was acquired before hand or during flight from the vision algorithm. For example, if other types of detection methods were used separately or in collaboration, then the distance from the obstacle could be determined and the next waypoint could be placed in a more intelligent manner.

Overall, this guidance law is robust in that it is guaranteed to avoid any detected obstacles in its path as well as learn from its past encounters.

Acknowledgments

Thanks to Matěj Karásek, Kimberly Mcguire, Sjoerd Tijmons, and Kevin Lamers.

References

[1] B.M. Albaker and N.A. Rahim. A survey of collision avoidance approaches for unmanned aerial vehicles. In *International Conference for Technical Postgraduates (TECHPOS)*, pages 1–7, Dec 2009.

[2] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Mav navigation through indoor corridors using optical flow. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3361–3368. IEEE, 2010.

[3] Ted Camus, David Coombs, Martin Herman, and Tsai-Hong Hong. Real-time single-workstation obstacle avoidance using only wide-field flow divergence. In *Proceedings of the 13th International Conference on*

Pattern Recognition, volume 3, pages 323–330. IEEE, 1996.

[4] G.C.H.E. de Croon, H.W. Ho, C. De Wagter, E. van Kampen, B.D.W. Remes, and Q.P. Chu. Optic-flow based slope estimation for autonomous landing. *International Journal of Micro Air Vehicles*, 5(4):287–297, December 2013.

[5] Larry Matthies, Roland Brockers, Yoshiaki Kuwata, and Stephan Weiss. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3242–3249. IEEE, 2014.

[6] Sjoerd Tijmons, Guido de Croon, Bart Remes, Christophe De Wagter, Rick Ruijsink, Erik-Jan van Kampen, and Qiping Chu. Stereo vision based obstacle avoidance on flapping wing mavs. In Qiping Chu, Bob Mulder, Daniel Choukroun, Erik-Jan van Kampen, Coen de Visser, and Gertjan Looye, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 463–482. Springer Berlin Heidelberg, 2013.

[7] K. Lamers, S. Tijmons, K.N. Mcguire, J. Junell, and M. Karásek. Mantis inspired obstacle avoidance using monocular stereo vision. MAV course: Obstacle avoidance competition and report, April 2015.

[8] K. Celik, Soon-Jo Chung, M. Clausman, and A.K. Somani. Monocular vision slam for indoor aerial vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1566–1573, Oct 2009.

[9] Gim Hee Lee, F. Fraundorfer, and M. Pollefeys. Mav visual slam with plane constraint. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3139–3144, May 2011.

[10] J. Kim and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. In *IEEE International Conference on Robotics and Automation*, pages 790–796 vol.1, Apr 1991.

[11] Jaime Junell, Erik-Jan van Kampen, Coen de Visser, and Qiping Chu. Reinforcement learning applied to a quadrotor guidance law in autonomous flight. In *AIAA Science and Technology Forum: Guidance, Navigation and Control Conference*, pages Paper No. AIAA 2015–1990, 2015.

[12] Stefan Wender and Ian Watson. Combining case-based reasoning and reinforcement learning for unit navigation in real-time strategy game ai. In Luc Lamontagne and Enric Plaza, editors, *Case-Based Reasoning Research and Development*, volume 8765 of *Lecture Notes in Computer Science*, pages 511–525. Springer International Publishing, 2014.

[13] A. Ohya, A. Kosaka, and A. Kak. Vision-based navigation of mobile robot with obstacle avoidance by single camera vision and ultrasonic sensing. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 704–711 vol.2, Sep 1997.

[14] Nguyen Xuan Dao, Bum-Jae You, and Sang-Rok Oh. Visual navigation for indoor mobile robots using a single camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 1992–1997, Aug 2005.

[15] Tomáš Krajník, Vojtěch Vonásek, Daniel Fišer, and Jan Faigl. AR-Drone as a platform for robotic research and education. In *Research and Education in Robotics - EUROBOT 2011*, volume 161 of *Communications in Computer and Information Science*, pages 172–186. Springer Berlin Heidelberg, 2011.

[16] B.D.W. Remes, D. Hensen, F. Van Tienen, C. De Wagter, E. Van der Horst, and G.C.H.E. De Croon. Paparazzi: How to make a swarm of Parrot AR Drones fly autonomously based on GPS. In *IMAV 2013: Proceedings of the International Micro Air Vehicle Conference and Flight Competition*, Toulouse, France, September 2013.