Improving the px4 Avoid Algorithm by Bio-Inspired Flight Strategies

A. Thoma[1,2], A. Fisher[2], C. Braun[1]
1 Department of Aerospace Engineering, FH Aachen UAS, Aachen, Germany
2 School of Engineering, RMIT University, Melbourne, Australia

Abstract

We present the concept of extending the px4 avoid algorithm for more reliable and flexible obstacle avoidance of non-cooperating, static obstacles. Besides the hard requirements that the algorithm has to fulfill, several soft requirements are presented. As the latter are less stringent, they have to be weighed against one another for each mission because their importance differs according to mission type. To fit the needs of different mission types, the px4 avoid algorithm is used as a basis and is controlled by a master function adjusting its cost parameters. This master function takes importance of the different soft requirements, depending on mission time, into account. To improve robustness, the master function additionally contains modules inspired by the behavior of bumblebees in challenging obstacle situations. This behavior is achieved by a speed controller that adapts the flight speed according to the lateral distance to obstacles, thereby influencing the turning radii of the UAV. Additionally, bumblebee behavior inspires a module, which triggers the vertical evasion of distant obstacles and the horizontal evasion of close obstacles. Finally, the master function has a separate element for goal approaches that overrules the normal flight behavior. This overruling allows different speed adaptions and specific obstacle approaches close to the goal.

Nomenclature

Δdown	=	vertical distance between goal and waypoint (if the waypoint is lower than the goal point)
Δup	=	vertical distance between goal and waypoint (if the waypoint is higher than the goal point)
Δyaw	=	horizontal distance between goal and way point
3dVFH*	=	3D Vector Field Histogram with integrated A*
3DVFH+	=	improved 3D Vector Field Histogram
a	=	deceleration parameter
c_{goal}	=	cost of non-goal oriented behavior
c_{smoot}	=	current flight direction deviation cost
c_{total}	=	total cost
d_{goal}	=	distance between current position and goal
k_{down}	=	weighting factor for Δdown
k_{goal}	=	weighting factor for c_{goal}
k_{smooth}	=	weighting factor for c_{smooth}
k_{up}	=	weighting factor for Δup
LiDAR	=	Light Detect and Range
SLAM	=	Simultaneous Localization And Mapping
SURF	=	Speed Up Robust Features
UAV	=	Unmanned Aerial Vehicle
v	=	velocity
VFH*	=	Vector Field Histogram with integrated A*

1. INTRODUCTION

Unmanned aerial vehicles (UAV) can be used for various applications. Nowadays, UAVs usually require a pilot, while most use and business cases require fully autonomously operating UAV in the long run. Even though several obstacle avoidance algorithms exist, reliable evasion of static and moving obstacles is still a great challenge [1]. Hence, more robust and efficient obstacle avoidance algorithms are required. One of the main challenges is reliably detecting obstacles with lightweight sensor systems and low computational effort in real-time. While much progress was recently made in this field of research, there is still the remaining challenge of choosing adequate evasion maneuvers after an obstacle is detected [2].

In robotics, evading an obstacle is referred to as local path planning. Cost functions, which are specially designed for local path planning, are often used to identify the best possible local path around an obstacle. These functions evaluate various waypoints under consideration of several different parameters; for example, by considering the deviation between the new path and the initially planned path regarding angle, distance, and velocity. Here, the balance between efficiency and robustness is particularly challenging as robustness is hardly measurable by classical engineering approaches. In this context, robustness is defined as resilience towards critical failure, e.g., collision with an obstacle or the ability to react adequately in novel situations. However, in general, biological systems are proven to have a good balance between efficiency and robustness. The way a biological system solves a novel situation is often simple but still convenient and efficient; they are flexible by nature. Additionally, many biological systems have a limited number of neurons, which equals a low computational power of a technical system. Nonetheless, biological systems can effectively manage accountable resources and quickly find viable solutions. One example of a successful biological system is the bumblebee. Since bumblebees are efficient fliers that need to navigate in cluttered environments, they are perfectly suitable test objects when seeking avoidance strategies. Therefore, the behavior of bumblebees serves as a basis for developing a reliable obstacle avoidance algorithm.

This paper is structured the following way: After this introduction, a comprehensive summary of path planning and obstacle avoidance algorithms is given, together with recent findings in biology and bio-inspired systems. Section 3 presents the requirements of a new control approach, followed by a concept to fulfill these requirements in section 4. Finally, a comprehensive conclusion is given in section 5.

2. THEORETICAL BACKGROUND

2.1. General Differentiation

For a technical application, the problem of obstacle avoidance is commonly split up into three steps [3]:
Obstacle detection is the first step in obstacle avoidance and the basis of any evasive maneuver. While animals mostly rely on passive systems to observe their environment, e.g., different kinds of eyes, many technical applications rely on active systems. The automotive industry is one industry that increases the level of autonomy of its products at a fast pace. Many modern cars have hundreds or even thousands of sensors and assisting systems. LiDAR and radar systems are commonly used for obstacle detection [4, 5]. However, optical cameras are also frequently used for various detection tasks. LiDAR systems can provide the most accurate representation of an observer's environment in all directions, as those systems can detect even the smallest objects at medium distances [4]. Unfortunately, this can only be achieved at the cost of a relatively heavy system. Radar systems, on the other hand, are of lighter weight and work in low visibility conditions but generally provide a narrow field of view [6]. Finally, cameras can provide the most accurate picture of an observer's surrounding at low weight and cost. Unlike radar and LiDAR systems, cameras cannot directly measure any distances. Complex and usually computationally heavy methods are required to estimate the distance to an object. SURF and SLAM are two commonly used algorithms to estimate the distance between the camera and the object [7–10].
Path Planning is an essential part of obstacle avoidance as well. In many applications, path planning results are called global paths [11]. The global path is the path from the start or current position to the goal point. Fix information from maps is usually considered when determining the global path [11]. However, variable information, for example, the position of a movable or unknown obstacle, is not considered in global path planning. Different methods are available for global path planning. Most of them are 2D methods or 2D methods extended to 3D. Rapidly-exploring Random Tree (RRT) and different variations of this algorithm are most commonly used in robotics [11–13].
Obstacle evasion is complementary to path planning, also frequently called local path planning. Here, only local, recently updated information from a sensor system is used to determine a specific path section [11]. Most of the currently available methods are for 2D navigation. The avoidance algorithm developed for the px4, the 3DVFH*, is one of the most advanced methods for 3D obstacle avoidance [14]. This algorithm discretizes the sensor field of view with a grid. If a grid contains an obstacle or is close to an obstacle, it is marked as unsafe. All other grids are marked as safe and evaluated by a cost function. Then a waypoint within the safe grid with the lowest cost is defined [15]. Unfortunately, this approach is not very flexible. Despite the availability of many parameters to evaluate the cost of a grid, only view control parameters to steer the flying platform are used. One of the main drawbacks is constant parameters that do not adapt to the current situation. Also, only one cost function is used [15].
Finally, most local planners define the shortest possible path as the best solution [16–18], which might be a minimal deviation from a global path or directly via the path length of the local path [19]. While this assumption might hold for most ground-bound applications, this is questionable for flying applications. Depending on the environment a drone is operating in, increasing the flight level might be advantageous compared to horizontal evasion. Even if the distance is longer or the energy consumption is higher for a single evasion, this might pay off in the long run over a whole mission [20].

2.2. The px4 avoid algorithm, the 3DVFH*

The 3DVFH* is one of the most advanced, publicly available algorithms for static obstacle avoidance, developed explicitly for small drones.
The algorithm is based on a 3DVHF* [15], which is a combination of the 3DVFH+ [21] and the VFH* [22] algorithm together with a novel memory strategy. The 3DVHF* algorithm is a local path planner that reacts to obstacles without building a global map. However, the memory part keeps track of previously detected obstacles by transforming the previous polar histogram to the current location. The algorithm consists of several submodules designed explicitly for certain obstacles:
The core part of the 3DVFH* is based on a 2D polar histogram (Figure 1). The histogram consists of a binary occupancy layer, a distance layer, and an age layer. Each layer consists of a cell grid. The cells of the occupancy layer contain the binary information on whether more or less 3D points than a specific threshold were detected in this sector; if more points are detected within one grid cell, the cell is considered occupied. Additionally, a safety margin is added around occupied cells. The size of the safety margin depends on the obstacle distance and dimension.

The distance layer contains information on the distance of the closest obstacle point inside each sector. The age layer contains the information age in each cell of the other layers. The age is required to allow an extended field of view, which is greater than the actual field of view of the sensor. The data from the previous occupancy and distance cells is re-projected to the current position and fused with the current sensor data. If the data outside the current field of view is too old, it is discarded.

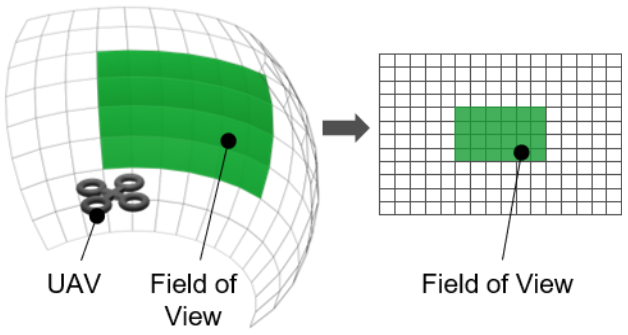

Figure 1: A histogram according to the VFH approach represents the surrounding world. The 3D world around the UAV is rasterized into sectors (left) and represented in a 2D grid cell (right), where one cell represents one sector.

Then, the directions of the histogram's free cells are considered candidates and evaluated by a cost function. The cost function consists of a goal-oriented term (c_{gaol}, equation (1)) and a flight path smoothness term (c_{smooth}, equation (2)). The goal-oriented term compares the candidate direction to the goal direction by taking the yaw difference Δ_{yaw}, and the pitch difference Δ_{pitch} into account. Both are weighted by weighting factors k_{yaw} and k_{pitch}. The differences are determined by projecting the candidate direction to a point with the same distance from UAV to goal point. The yaw difference is the distance of this point to the goal point in the XY-plane; the pitch difference is the distance in Z-direction. The smoothness term compares the candidate direction with the current one and sums the yaw $\Delta_{yaw,prev}$and pitch $\Delta_{pitch,prev}$ difference to the previous path. The cell with the lowest total cost under differently weighted consideration of the goal, smoothness cost, and obstacle cost $c_{obstcale}$ is chosen as the next waypoint (equation (3)). The obstacle cost $c_{obstacle}$ is proportional to the distance of a candidate point to the next obstacle.

(1) $$c_{goal} = \mathrm{k}_{\mathrm{yaw}} \cdot \Delta_{yaw} + k_{pitch} \cdot \Delta_{pitch}$$

(2) $$\mathrm{c}_{\mathrm{smooth}} = \Delta_{yaw,prev} + \Delta_{pitch,prev}$$

(3) $$c_{total} = c_{goal} + k_{smooth} \cdot c_{smooth} + c_{obstacle}$$

Additionally, an A* path planning algorithm is used to extend the algorithm to a series of candidate points. For any candidate point, the vector field histogram is estimated with the available sensor data and the predicted movement of the UAV. The prediction generates a search tree from the current position with a predefined length of nodes. Every note in the tree is evaluated by the cost function given in equation (3). The first node of the branch with the lowest total cost is then chosen as the next waypoint (Figure 2).

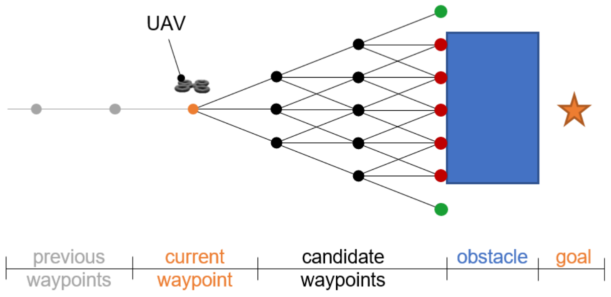

Figure 2: Tree of the A* algorithm from the current position (orange dot) towards the goal (orange star). This tree consists of three levels of branches. Initially, for every level, the branch with the lowest cost is chosen, i.e., the algorithm directly approaches the goal until all sub-branches of the best branch are invalid or the maximal tree length is reached (in this example 3). If a candidate point is not valid, e.g., the point is too close to an obstacle, this sub-branch is considered impossible (red dot). If all sub-branches of a branch are invalid, the algorithm starts to expand the tree by following the second optimal branch of the previous level of branches. This procedure repeats until an acceptable path is found (green point). The first waypoint of the acceptable path with the lowest cost is chosen as the next waypoint

However, even though a safety margin around the obstacle is defined in the 2D histogram, it does not ensure a minimum safety distance between UAV and the obstacle in flight direction. Therefore, a minimum longitudinal distance to obstacles is defined within the px4 avoid algorithm. The algorithm defines two three-dimensional spheres; the UAV sphere, around the UAV, and the obstacle sphere, around the obstacles. The center of the UAV sphere is the UAV; the center of the obstacle sphere is the mean of all detected obstacle points within the UAV sphere (Figure 3). A minimal distance between the UAV and the obstacle sphere is defined to maintain a safe distance between UAV and the obstacle.

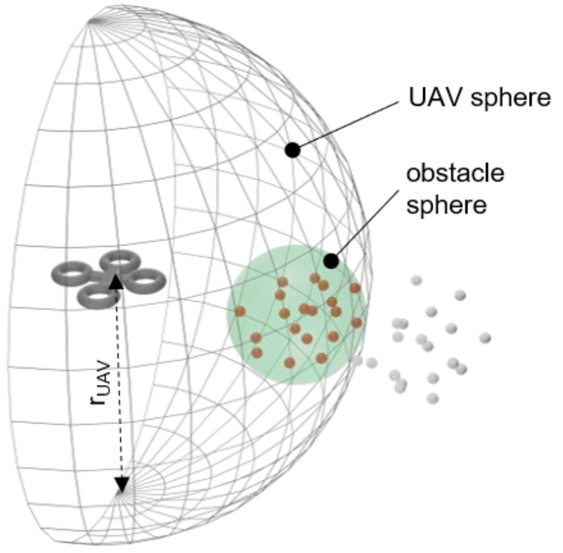

Figure 3: Representation of the UAV sphere and the obstacle sphere, which is a sphere around all obstacle points inside the UAV sphere.

Additionally, a ground detection model is implemented, which keeps a minimum distance to the ground. In a setup with a forward-looking camera, the actual distance above the ground is unknown because the ground below the UAV is not in the current field of view. However, the ground in front of the UAV is in the field of view. The vertical distance to the UAV is stored in a height map; The height map is based on a bounding box, which crops the point cloud from the sensor system. A RANSAC algorithm estimates a horizontal plane from the point cloud inside the bounding box. If the UAV is flying towards a ground patch stored inside the height map, it needs to adapt its height before reaching the ground patch and rising with a UAV-specific maximal rising angle. If the UAV is too low, all candidate points from the polar histogram lower than the maximum rising angle are considered as blocked. If the current altitude is close to the minimal altitude, all candidate points lower than the horizontal axis are blocked.

2.3. Recent advances in bio-inspired obstacle avoidance

Honeybees and bumblebees forage between rewarding food sources and their hive in often complex environments. They must fly through clutter consisting of obstacles of different sizes, shapes, orientations, and textures. To avoid physical damage [23] and to perform the task of food collection as efficiently as possible, insects need to move around objects obstructing their way [24]. Flying insects, in particular, can apply various strategies and flight maneuvers to avoid objects and reach their goal efficiently, fast, and safely [25–27].
Currently, much knowledge is available about the behavior of bumblebees in specific situations. Several research groups identified critical aspects of bee behavior when flying through tunnels. For example, bees maintain equidistance to both walls [28, 29] by maintaining equivalent optic flow on both eyes [30]. Additionally, by keeping the optic flow constant, the flight speed is adapted to the width of the tunnel [30, 31]. Even though the behavior in an empty tunnel is understood quite well, none of the above gives information on obstacle encounters.
If challenged with a series of vertical or horizontal obstacles within a flight tunnel, bumblebees do not show any significant difference in the maneuver when avoiding the obstacles horizontally or vertically [32]. However, body size has a more significant influence on the flight behavior of bumblebees than obstacle orientation. With increasing body size, flight performance is impaired [32]. However, this work investigates only one possibility to evade the obstacle in a confined space such that the bee cannot decide between several alternatives.
When bumblebees can decide between two horizontally aligned gates, they tend to choose the wider gap [33]. However, this situation may change when gaps are not horizontally aligned, but the bees must avoid an object by moving upward or sideward. If bumblebees can choose between avoiding an obstacle vertically or horizontally, e.g., flying over the obstacle or flying around it, their choice depends on the distance to the obstacle when noticing it for the first time [34]. If the obstacle is close, bumblebees tend to evade horizontally; if the obstacle is far, they tend to evade vertically. However, the reason for this behavior is not known yet.
Thus, we want to combine these recent advances in biology with recent avoidance algorithms.

3. REQUIREMENTS AND ASSUMPTIONS

A clear definition of the requirements is key to developing any algorithm. We divide requirements into hard requirements that must be fulfilled in any case and soft requirements that are formulated less stringent and allow more flexibility. Additionally, assumptions must be made, which will be discussed in this chapter.

3.1. Hard Requirements

The essential requirement of an obstacle avoidance algorithm is that it has to be safe. Therefore, a collision has to be avoided in any case. The second hard requirement for an obstacle avoidance algorithm is to control the UAV, so it flies from the current position to a specified goal position. It requires some drive or goal-directed behavior, which leads the UAV to the goal point until its final position is reached. Those two aspects need to be fulfilled in any case. A mission is considered failed if one of those two aspects is not fulfilled. Another hard requirement is for the algorithm to work in real-time because a local planner reacts to local, unforeseen obstacles; it needs to react in an adequate time frame to avoid the obstacle before it is hit.

3.2. Soft Requirements

Unlike the hard requirements, some less stringent or specific soft requirements are defined. These requirements often contradict each other or are derived from hard requirements. A typical example is a system that needs to be as light, safe, and cheap as possible. These factors must be weighed against each other to find an optimal solution for the overall system. Similarly, several soft requirements apply to an obstacle avoidance algorithm as well. The soft requirements considered in this approach are:

1. The algorithm has to react as fast as possible.
2. The algorithm has to be as computationally efficient as possible.
3. The UAV has to maintain an adequate safety distance to obstacles.
4. The UAV has to fly as far as possible.
5. The UAV has to fly as fast as possible.
6. The UAV has to fly as long as possible.
7. The energy consumption of the maneuvers has to be as low as possible.
8. The accelerations have to be in a reasonable range.

The soft requirements must be weighed against each other to find an optimal solution. However, what is understood by the word “optimal” might be different from mission to mission. For a military surveillance drone, maximal flight endurance might be more important than flight speed. Flight endurance might be less critical for a drug delivery drone, but flight speed might be critical. A general flight control system must adapt to a specific weighting of these soft requirements and behave according to the needs of the flown mission. Therefore, means have to be implemented to optimize for one or another mission type flexibly.

3.3. Assumptions

Some assumptions must be made to define the functional frame of the algorithm. First of all, the algorithm has to avoid static, non-cooperating obstacles. Non-cooperating means that the obstacle does not actively send out any information about its size, orientation, position, or presence at all. However, the obstacle might be arbitrary in texture, shape, size, and position. Additionally, the number of obstacles also might be arbitrary.
It is assumed that the sensor range data of the surrounding of the UAV is available. The maximum and minimum sensor range is assumed to be known and used as input parameters for the algorithm. The field of view of the sensor system is assumed to be arbitrary but known and constant. Feature identification, point tracking, and similar functions are not required. It is also assumed that the sensor system and flight controller are equipped onboard the UAV, and no downlink to a ground station is required. The UAV shall be able to navigate autonomously and be self-sufficient.

4. CONCEPT

The obstacle avoidance algorithm has to fulfill all hard requirements in any case, while finding the optimal trade-off between the different soft requirements simultaneously. To achieve both types of requirements, the 3DVFH*, implemented in the px4 avoidance algorithm, is used as a basis and combined with an intelligent master function. The master function sets the px4 avoidance control parameters according to the current situation and the importance of a specific soft requirement for the current mission.
Most flying animals face the same problem of finding an optimum between several different requirements. Therefore, the strategies applied by bumblebees to fly through cluttered environments are used as a basis for the intelligent master function. Even though more research is required to develop a functional algorithm based on the behavior of bumblebees, some of the strategies applied by bumblebees are presented here and possible ways to implement them as a part of an existing obstacle avoidance algorithm.

4.1. The basis

The 3DVFH* is used as a basis. The algorithm has shown that it can solve various situations reliably if the px4 control parameters are chosen correctly. The authors tested the performance of the px4 avoid algorithm in 3754 different worlds with different obstacle situations. The tests were conducted in a software in the loop (SITL) environment with GAZEBO simulator and ROS. Three types of worlds were tested. The first type, 900 worlds, had one to forty randomly placed rectangular obstacles with different dimensions and positions. One world is exemplarily depicted in Figure 4.

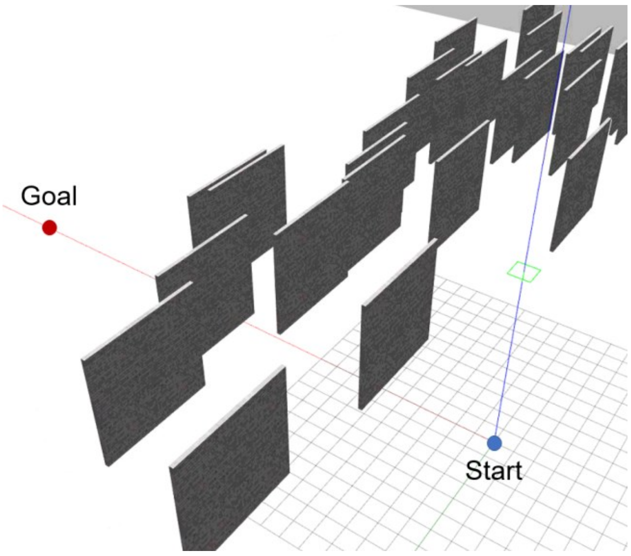

Figure 4: Example World with 40 obstacles, 6 m x 6 m in size. The drone's start point (blue) and goal point (red) are 6 m away from the first/last obstacle.

The second type, 254 worlds, consists of cylindrical obstacles placed randomly between the start and the goal point. These worlds partly had a hill or valley between the start and the goal point. The third type, 2600 worlds, representing cities, had differently sized cuboids (buildings), randomly arranged in specified distance ranges (streets). This type of world is shown in Figure 5.

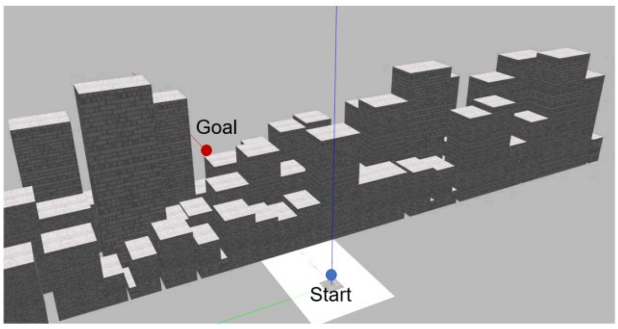

Figure 5: Example world with building-like obstacles arranged in rows from left to right with spaces typical for 3 to 5 lane roads between each row of obstacles. The blocks represent buildings with 7 to 50 floors and are generated and placed randomly. Additionally, several Obstacles have a small space (1-5 m) between each other; fewer obstacles have wider spaces (5-20 m). Goal and start point position are varied from close to buildings (5 m) to far from buildings (50m)

The simulations showed that the goal position of most worlds could be reached without crash with at least one combination of px4 avoid parameters. However, the parameter combination, which leads to a successful flight, differed from world to world. Additionally, the analysis showed that there is still much improvement for the px4 avoid algorithm. An average of 20% of the simulations were not successful. Here, not successful means the UAV does not reach the goal point. The reason might be a crash, safe landing, or inability to pass an obstacle and reach the goal.

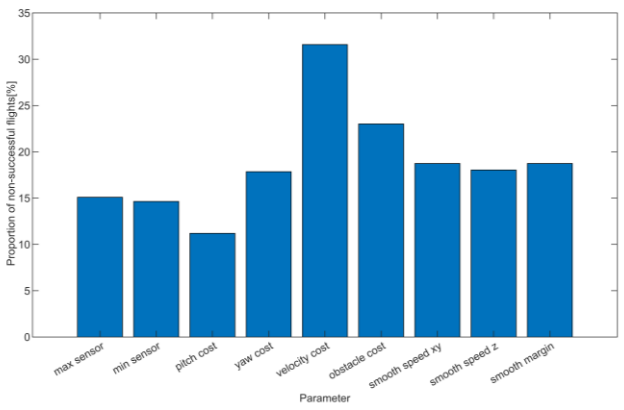

Figure 6: Proportion of non-successful flights for variations of the choosable px4 avoid parameters max_sensor_range (max sensor), min_sensor_range (min sensor), pitch_cost_param (pitch cost), yaw_cost_param (yaw cost), velocitiy_cost_param (velocity cost), obstacle_cost_param (obstacle cost), smoothing_speed_xy (smooth speed xy), smoothing_speed_z (smooth speed z), smoothing_margin_degrees (smooth margin)

Figure 6 shows that some parameters, e.g., the variation of the velocity cost parameter or the pitch cost parameter, have a more significant influence on the success rate of the px4 algorithm than other parameters. However, Figure 6 also shows that at least 10% of flights were unsuccessful regardless of which px4 avoid parameter varies within useful bounds.
Nonetheless, the px4 avoid algorithm already contains various features and can prevent the UAV of crashes in many cases. It is also able to fulfill its mission and reach its goal on the majority of flights. Therefore, the current px4 avoid algorithm, release 0.3.1, is chosen as a basis.

4.2. Situation-dependent speed adaption

The compound eyes of insects work differently than human eyes and directly measure optic flow. Optic flow is the pattern of objects' relative motion in an observer's field of view; it equals an angular velocity map around an observer [35]. Bees and other insects keep the optic flow on their eyes constant to adapt their flight speed to any situation. Recently, Dynamic Vision Sensors (DVS), also known as event cameras, silicon retina, or neuromorphic cameras, were developed, which directly measure optic flow and do not work like classical cameras or the human eye. If the UAV is equipped with such a system, a situation-

based flight velocity control with a flight controller keeping the optic flow constant can be directly implemented. The disadvantage of such a system is that obstacles in flight direction are hard to detect or not detectable at all.
Therefore, a classic 3D vision system or a combination of a DVS camera and an optic range detector are considered a more reliable solution. Here, careful choice of the available field of view is essential. The distance between UAV and the obstacle should directly influence flight speed. If obstacles are close, the flight speed should be adapted to an adequate, lower speed. However, this should happen for obstacles in front and around the UAV. Taking obstacles lateral to the UAV into account allows passing such obstacles at an adequate speed, which increases safety and flexibility. The behavior of bumblebees can be mimicked by linearly reducing the flight speed from cruise velocity to minimal flight velocity with decreasing obstacle distance (Figure 7). The minimal flight speed and goal distance must be chosen according to the UAV's weight. The minimal flight speed should be lower at a higher minimal distance for heavier UAVs than for lighter UAVs. This speed is chosen similarly to the difference in the behavior of heavier and larger bumblebees compared to lighter and smaller ones.

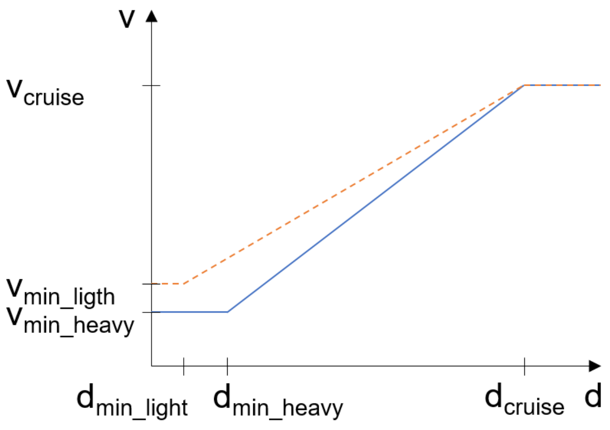

Figure 7: the linear relationship between forward flight velocity and lateral distance to the closest obstacle for lighter UAV (orange, dotted line) and heavier UAV (blue, solid line)

4.3. Differentiation between close and far obstacles

Recent investigation on the behavior of bumblebees indicates that bumblebees use two fundamentally different obstacle avoidance strategies. Close obstacles are evaded horizontally, and far obstacles are evaded vertically. The px4 avoid algorithm, on the other hand, is often stuck in front of wide obstacles and meandering in front of them because the cost to evade the obstacle in one direction becomes too high.

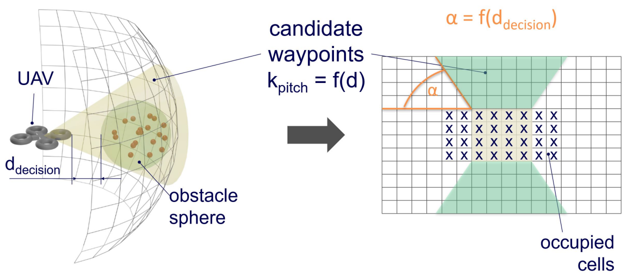

Figure 8: Adaption of the px4 avoid algorithm and limiting the considered candidate points according to the distance between UAV and obstacle sphere $d_{decision}$. A core section of candidate points will be evaluated in any case (yellow), while the evaluation of the surrounding points depends on $d_{decision}$ (green).

Implementing a strategy to evade far obstacles vertically, e.g., flying over them, will solve this problem partly. This behavior can be achieved by significantly decreasing the cost of vertical avoidance k_{pitch}. If the UAV starts increasing flight altitude directly after identifying the obstacle and if the distance is higher than the minimal distance for vertical evasion, $d_{decision}$, the obstacle will be overflown when reaching it (Figure 8). If this is not possible because the obstacle is too high, the drone is forced to reevaluate the situation when the UAV is close to the obstacle, and the obstacle blocks the whole FoV. In this case, the horizontal avoidance maneuver is performed with a high cost of vertical avoidance k_{pitch} until the obstacle is passed. If the obstacle is high and wide, the UAV will start meandering in front of the obstacle with the current px4 avoid flight algorithm. If the horizontal avoidance maneuver is performed for a specified time, this should be combined with vertical avoidance (=reduction of k_{pitch}) and decreasing of the goal cost parameters k_{goal} such that the UAV starts to fly longer distances to the left and right and increase in altitude. By this adaption of parameters, the searching behavior of bumblebees can be mimicked. Additionally, this approach can reduce the overall algorithm's computational effort because the total number of candidate points can be reduced. If the closest obstacle is below a certain threshold, $d_{decision}$, is only candidate points in a wide but short corridor need to be considered. If the distance is greater than $d_{decision,}$ points in a high but narrow corridor need to be considered (Figure 8). The shape of this corridor is defined by a core section around the goal direction, which will be evaluated in any case (yellow), and a distance-dependent section defining the waypoints, which will be considered around the core section (green).
However, this approach requires a minimal tree length longer than $d_{decision}$ such that the avoidance strategy is not changed while the UAV is approaching the obstacle. Alternatively, the corridor of possible

candidate points is fixed until the obstacle is passed or no good candidate point is available.

4.4. Final goal approach

Analysis of the px4 avoid algorithm showed no difference in behavior if the drone is close to the goal point. This behavior leads to several different problems. If the goal is close to a landing platform, the platform is identified as an obstacle. The wrong identification either leads to meandering in front of the goal point or circulation around the goal point. In this case, the goal is never reached. Therefore, the safety distance to obstacles needs to be adjusted to allow landing on platforms or hovering close to buildings. For a given flight speed, according to equation (4), a minimally required distance to an obstacle can be derived from Figure 7.
Another problem is that the UAV changes flight speed only in specific situations and rates. Flying close to the goal is none of these cases. Therefore, an overshooting of the goal point is common, as visualized in Figure 9.

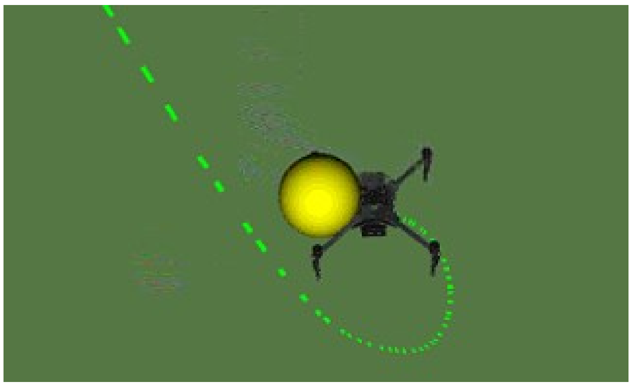

Figure 9: UAV overshooting the goal point (yellow). The green dotted line represents the flight path.

In short missions or missions with several dedicated stop and hover points, e.g., to drop a parcel, the time and energy consumption required to correct this overshooting is significant and reduces efficiency.
Therefore, a specified goal approach is required to improve performance. The px4 algorithm is already equipped with a save landing assistant. However, this algorithm is not used to land safely at a specific position but to land close by the current position.
To adequately approach a specified goal point, a new algorithm is required. Here again, insects have developed strategies to land on and in flowers. They decelerate smoothly during approach, maintaining a constant image expansion rate. This behavior can be mimicked by a proportional dependency of the flight speed *v* to the goal distance d_{goal} of the type

$$v = \sqrt{a} \cdot d_{goal} \qquad (4)$$

In equation (4), a has to be chosen according to the desired mission type because this parameter defines the energy consumption and duration of the final goal approach. The following table gives a comprehensive but not exhaustive overview of recommended values for a for different mission types and situations independent of the UAV.

Mission Type	a
Time-critical	90% of the maximally available deceleration
High Endurance	Average drag force during approach over total mass
Low accelerations	Average drag force during approach over total mass
High safety and time-critical	75% of the maximally available deceleration

Table 1: recommended *a* values for different mission types

In order to be able to react to gusts, it is not recommended to plan for theoretically maximally available deceleration rates.
Additionally, k_{smooth} should be decreased with decreasing distance to the goal to allow fast direction and velocity changes.

5. CONCLUSION

The conducted studies show that the px4 avoid algorithm is well suited to solve various challenging situations and fly autonomously from start to goal point. However, the parameters controlling the behavior of the px4 avoid algorithm have to be chosen with much care. Therefore, an approach is presented which adapts the flight speed to the current situation to simplify situations for the px4 algorithm. Additionally, close obstacles should be avoided horizontally, while far obstacles should be overflown. This behavior will diminish the problem of the px4 getting stuck in various cases. Finally, a specific goal approach is required, which adapts the flight speed independent of the smoothness parameter k_{smooth} of the px4 avoid algorithm to improve the performance of the goal approach.

6. REFERENCES

[1] S. Aggarwal and N. Kumar, “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Computer communications*, vol. 149, pp. 270–299, 2020, doi: 10.1016/j.comcom.2019.10.014.

[2] T. A. Sarmiento and R. R. Murphy, “Insights on obstacle avoidance for small unmanned aerial systems from a study of flying animal behavior,” *Robotics and Autonomous Systems*,

vol. 99, pp. 17–29, 2018, doi: 10.1016/j.robot.2017.09.002.

[3] X. Zhao, G. Wang, M. Cai, and H. Zhou, “Stereo-vision based obstacle avoidance by finding safe region,” *International Journal of Control, Automation and Systems*, vol. 15, no. 3, pp. 1374–1383, 2017.

[4] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, “3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes,” *Robotics and Autonomous Systems*, vol. 83, pp. 299–311, 2016.

[5] R. Thakur, “Scanning LIDAR in Advanced Driver Assistance Systems and Beyond: Building a road map for next-generation LIDAR technology,” *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 48–54, 2016.

[6] W. Song, Y. Yang, M. Fu, F. Qiu, and M. Wang, “Real-Time Obstacles Detection and Status Classification for Collision Warning in a Vehicle Active Safety System,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 758–773, 2018.

[7] Wilbert G. Aguilar, Verónica P. Casaliglla, and José L. Pólit, “Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles,” *Electronics*, vol. 6, no. 1, p. 10, 2017. [Online]. Available: https://doaj.org/article/4bdb5fca1ae245a9b8c2d8755e5b882e

[8] C.-H. Kim, T.-J. Lee, and D.-I. “. Cho, “An Application of Stereo Camera with Two Different FoVs for SLAM and Obstacle Detection,” *IFAC PapersOnLine*, vol. 51, no. 22, pp. 148–153, 2018.

[9] R. Renjith, Reshma R., and K. V. Arun, “Design and implementation of traffic sign and obstacle detection in a self-driving car using SURF detector and Brute force matcher,” *IEEE ICPCSI*, vol. 2017.

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: real-time single camera SLAM,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007, doi: 10.1109/TPAMI.2007.1049.

[11] M. Pittner, M. Hiller, F. Particke, Patino-Studencki. L., and J. Thielecke, “Systematic Analysis of Global and Local Planners for Optimal Trajectory Planning,” *ISR 2018; 50th International Symposium on Robotics*, vol. 2018.

[12] X. Liang, G. Meng, Y. Xu, and H. Luo, “A geometrical path planning method for unmanned aerial vehicle in 2D/3D complex environment,” *Intelligent Service Robotics*, vol. 11, no. 3, pp. 301–312, 2018.

[13] Kun Wei and Bingyin Ren, “A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm,” *Sensors*, vol. 18, no. 2, p. 571, 2018. [Online]. Available: https://doaj.org/article/f3e03758ca0e41a3b3ba56a45256f25b

[14] J. García and J. M. Molina, “Simulation in real conditions of navigation and obstacle avoidance with PX4/Gazebo platform,” *Pers Ubiquit Comput*, 2020.

[15] T. Baumann, “Obstacle Avoidance for Drones Using a 3DVFH Algorithm,” Masters Thesis, ETH Zürich;, 2018.

[16] E. de Lellis, G. Morani, F. Corraro, and V. Di Vito, “On-line trajectory generation for autonomous unmanned vehicles in the presence of no-fly zones,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 227, no. 2, pp. 381–393, 2012, doi: 10.1177/0954410011430173.

[17] J. Gonzalez, A. Chavez, J. Paredes, and C. Saito, “Obstacle Detection and Avoidance Device for Multirotor UAVs through interface with Pixhawk Flight Controller,” in *IEEE CASE*, pp. 110–115.

[18] A. Alexopoulos, A. Kandil, P. Orzechowski, and E. Badreddin, “A Comparative Study of Collision Avoidance Techniques for Unmanned Aerial Vehicles,” in *IEEE Syst Man Cy C*, pp. 1969–1974.

[19] R. He, R. Wei, and Q. Zhang, “UAV autonomous collision avoidance approach,” *Automatika*, vol. 58, no. 2, pp. 195–204, 2017, doi: 10.1080/00051144.2017.1388646.

[20] S. Ahmed, A. Mohamed, K. Harras, M. Kholief, and S. Mesbah, “Energy efficient path planning techniques for UAV-based systems with space discretization,” in *2016 IEEE Wireless Communications and Networking Conference*, 2016, pp. 1–6.

[21] I. Ulrich and J. Borenstein, “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots,” in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, 1998.

[22] I. Ulrich and J. Borenstein, Eds., *VFH*: Local Obstacle Avoidance with Look-Ahead Verification*, 2000.

[23] A. M. Mountcastle, T. M. Alexander, C. M. Switzer, and S. A. Combes, “Wing wear reduces bumblebee flight performance in a dynamic obstacle course,” *Biology letters*, vol. 12, no. 6, 2016, doi: 10.1098/rsbl.2016.0294.

[24] J. L. Osborne *et al.,* “The ontogeny of bumblebee flight trajectories: from naïve explorers to experienced foragers,” *PloS one*, vol. 8, no. 11, e78681, 2013, doi: 10.1371/journal.pone.0078681.

[25] F. A. Zabala, G. M. Card, E. I. Fontaine, M. H. Dickinson, and R. M. Murray, “Flight Dynamics and Control of Evasive Maneuvers: The Fruit Fly’s Takeoff,” *IEEE T Bio-Med Eng*, vol. 56, no. 9, pp. 2295–2298, 2009.

[26] F. T. Muijres, M. J. Elzinga, J. M. Melis, and M. H. Dickinson, “Flies evade looming targets by executing rapid visually directed banked turns.,” *Science*, vol. 344, no. 6180, p. 172, 2014.

[27] R. Kern, N. Boeddeker, L. Dittmarand, and M. Egelhaaf, “Blowfly flight characteristics are shaped by environmental features and controlled by optic flow information.,” *J Exp Biol*, vol. 215, no. 14, p. 2501, 2012.

[28] J. Serres, G. P. Masson, F. Ruffier, and N. Franceschini, “A bee in the corridor: centering and wall-following,” *The Science of Nature Naturwissenschaften*, vol. 95, no. 12, pp. 1181–1187, 2008. [Online]. Available: https://hal-amu.archives-ouvertes.fr/hal-02294572

[29] G. Portelli, J. R. Serres, and F. Ruffier, “Altitude control in honeybees: joint vision-based learning and guidance,” *Sci Rep*, vol. 7, no. 1, p. 9231, 2017.

[30] M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett, “Honeybee navigation en route to the goal- Visual flight control and odometry,” *J Exp Biol*, vol. 199, pp. 237–244, 1996, doi: 10.1242/jeb.199.1.237.

[31] M. V. Srinivasan, S. W. Zhang, J. S. Chahl, G. Stange, and M. Garratt, “An overview of insect-inspired guidance for application in ground and airborne platforms,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 218, no. 6, pp. 375–388, 2004, doi: 10.1243/0954410042794966.

[32] J. D. Crall, S. Ravi, A. M. Mountcastle, and S. A. Combes, “Bumblebee flight performance in cluttered environments: effects of obstacle orientation, body size and acceleration,” *J Exp Biol*, vol. 218, Pt 17, pp. 2728–2737, 2015, doi: 10.1242/jeb.121293.

[33] M. Ong, M. Bulmer, J. Groening, and M. V. Srinivasan, “Obstacle traversal and route choice in flying honeybees: Evidence for individual handedness,” *PloS one*, vol. 12, no. 11, e0184343, 2017, doi: 10.1371/journal.pone.0184343.

[34] A. Thoma, A. Fischer, O. Bertrand, and C. Braun, “Evaluation of Possible Flight Strategies for Close Object Evasion From Bumblebee Experiments,” in *Lecture Notes in Artificial Intelligence*: Springer, 2020.

[35] M. V. Srinivasan, “Visual control of navigation in insects and its relevance for robotics,” *Current Opinion in Neurobiology*, vol. 21, no. 4, pp. 535–543, 2011, doi: 10.1016/j.conb.2011.05.020.