IMAV2016-10
http://www.imavs.org/pdf/imav.2016.10

Applicability of Vector Field Histogram Star (VFH*) on Multicopters

R.J. van Breda* and W.J. Smit†
Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Stellenbosch, South Africa

ABSTRACT

This paper investigates the applicability of the Vector Field Histogram Star (VFH*) obstacle avoidance algorithm on multicopters using two-dimensional LIDAR systems. Modifications to the existing VFH* algorithm are made to account for position and yaw uncertainties of multicopters. The effects of tilt angles caused by pitch and roll manoeuvres are also addressed by modifying range measurements from the LIDAR system. The need for steering control and the effects of changes in the search depth of the VFH* algorithm are also investigated.

1 INTRODUCTION

The possible uses for multicopters are seemingly endless. From search and rescue operations, photography, surveying, emergency aid, and even drone racing it would seem their uses are limited only by our own imaginations. However, there are inherent dangers posed by multicopters. The loss of control over such a drone can easily result in injuries occurring from the falling mass or its spinning rotors. As safety concerns grow over the use of such devices the need for autonomous obstacle avoidance capabilities and autonomy have become essential.

Obstacle avoidance can either be done locally, globally or through a combination of the two. A local obstacle avoidance system uses real time sensor data to detect obstacles in the immediate vicinity of a robot and steers the robot clear of any obstacles encountered in the robot's path. On the other hand global obstacle avoidance system determines the most appropriate path for a robot to follow based on a map of the robot's known environment [1].

In 1989 Johann Borenstein and Yorem Koren suggested the Virtual Force Field (VFF) method which combined two existing concepts, that of certainty grids for obstacle representation and potential fields for navigation [2] . The method also made use of the Wall Following Method (WFM) to get out of minimum trap situations. For the WFM the robot follows the obstacles contour until it can resume its path to the target position [2] . In 1991 they suggested an improved algorithm called the Vector Field Histogram (VFH) method which uses a two-dimensional Cartesian histogram grid for a world model [3]. Onboard range sensors were used to update the world model and then a two-stage data reduction process carried out to compute the desired steering commands for the robot [3]. In the first stage the histogram grid is reduced to a one-dimensional polar histogram grid that is centered on the robot. In the second data reduction stage the area around the robot is divided into sectors each having its own polar obstacle density value and the most suitable sector is chosen for the robot to travel towards. Then, again in 1998 the VFH method was improved and appropriately named the VFH+ method. The VFH+ algorithm accounts for the robots width and approximates the available robot trajectories when selecting its optimal path [4]. This resulted in smoother robot trajectories and greater reliability. Then in 2000 the method was further enhanced and called the VFH* method [1]. The VFH* algorithm makes use of the A* search algorithm and appropriate cost and heuristic functions in order to verify whether candidate directions do in fact guide the robot around obstacles. This helps the robot to deal with situations that purely local obstacle avoidance algorithms such as the VFF, VFH, VFH+ do not take into consideration.

Our aim is to establish the applicability of the VFH* algorithm to multicopters since the algorithm was originally created with with mobile ground robots in mind. We also investigate what modifications can be made to better the algorithm for application on multicopters through simulations. We consider only the case where the multicopter would make use of a two-dimensional LIDAR system for range measurements.

In specific we investigate the effect of multicopter position and yaw uncertainties. The effects of pitch and roll movements on range measurements are also investigated and the effects of steering control and the VFH* algorithm's search depth in a multicopter capacity.

Section 2 provides a more detailed overview of the workings of the VFH* algorithm. Our suggested modifications are discussed in Section 3 with a conclusion on the applicability of the algorithm in Section 4.

2 THE VFH* ALGORTIHM

The VFH* algorithm makes use of a two-dimensional Cartesian histogram grid C for obstacle representation [3].

*Email address: 16546660@sun.ac.za
†Email address: wjsmit@sun.ac.za

Each cell of the histogram grid C has its own certainty value $c(i,j)$. On-board range sensors are used to continuously and rapidly update the histogram grid in real time. Originally ultrasonic sensors were used but we propose the use of a two-dimensional LIDAR system for use on a multicopter because of its superior range capabilities compared to ultrasonic sensors . Each cells' certainty value $c(i,j)$ is updated as the robot moves. If an obstacle is detected the cell's certainty value is incremented otherwise its certainty value slowly decays away over time. Thus, if an obstacle is detected multiple times it will have a high certainty value while on the other hand random noise or misreadings will fade away with time [3].

VFH* makes use of a four stage data reduction process to reduce the data from the histogram grid C [4]. The aim of which is to obtain candidate directions in which the robot can move. Then the candidate directions are evaluated based on a cost function and look-ahead verification and the most suitable one is chosen as the robot's direction of motion [1].

2.1 *First Stage - The Primary Polar Histogram*

The first data-reduction stage maps the active region C_a from the histogram grid C onto the primary polar histogram, H^p [4]. The active region C_a is located around the robot's momentary location and has a radius of $r_{active-region}$. Each cell acts as an obstacle vector with a vector direction and magnitude [4]. The vector direction, $\beta_{i,j}$, is based on the direction of the active cell relative to the robot center point (RCP):

$$\beta_{i,j} = \arctan\left(\frac{y_o - y_i}{x_o - x_i}\right) \tag{1}$$

where:

x_o, y_o : Momentary coordinates of the RCP.
x_i, y_i : Coordinates of active cell $C_{i,j}$.

The vector magnitude, $m_{i,j}$ of an active cell $C_{i,j}$ is in turn calculated as follows:

$$m_{i,j} = c_{i,j}^2(a - bd_{i,j}^2) \tag{2}$$

where:

$c_{i,j}$: Certainty value of active cell $C_{i,j}$.
$d_{i,j}$: Distance from active cell $C_{i,j}$ to the RCP.

The parameters a and b are chosen according to:

$$a - b\left(\frac{r_{active-region} - 1}{2}\right)^2 = 1 \tag{3}$$

Note that the certainty value, $c_{i,j}$, is squared when the vector magnitude is calculated. This means that recurring range readings will result in high certainty values and thus we can be more confident that an obstacle resides in the obstacle cell $C_{i,j}$. In contrast noise or once-off range readings do not result in high certainty values.

The distance between the active cell and RCP, $d_{i,j}$, is also squared when calculating the vector magnitude. This means that occupied cells will have greater vector magnitudes the closer they get to the robot.

To prevent the robot from cutting corners we have to compensate for the robot's width [4]. This is done by enlarging occupied cells by the robot radius, r_{robot}. Additionally, a minimum allowed distance between the robot and obstacle, d_{safety}, can be added. The resulting radius r_{r+s}, is then defined as $r_{r+s} = r_{robot} + d_{safety}$. By enlarging occupied cells by r_{r+s} the robot can be treated as a point-like vehicle [4].

The primary polar histogram, H^p, is constructed by dividing the active window surrounding the robot into sectors [4]. An arbitrarily chosen angular resolution, α, is used so that the number of sectors is always an integer, i.e. $n = 360°/\alpha$. In our simulations we chose $\alpha = 5°$. Each angular sector corresponds to a discrete angle $\rho = k \cdot \alpha$.

Instead of updating only the sectors in which occupied cells fall all histogram sectors that are affected by the enlarged occupied cells are updated while building the primary polar histogram. The enlargement angle $\gamma_{i,j}$ is calculated as follows:

$$\gamma_{i,j} = \arcsin\left(\frac{r_{r+s}}{d_{i,j}}\right) \tag{4}$$

The polar obstacle density for each sector k is calculated as follows:

$$H_k^p = \sum_{i,j \epsilon C_a} m_{i,j} \cdot h'_{i,j} \tag{5}$$

with:

$$h'_{i,j} = 1 \qquad \text{if} \quad k \cdot \alpha \epsilon \, [\beta_{i,j} - \gamma_{i,j}, \beta_{i,j} + \gamma_{i,j}] \tag{6}$$

$$h'_{i,j} = 0 \qquad \text{otherwise} \tag{7}$$

The h' function acts as a low-pass filter and smooths the polar histogram [4]. The resulting polar histogram grid considers the robot width.

Figure 1 shows a robot surrounded by obstacles. The primary polar histogram is drawn around the robot indicating sectors with high polar density values or in other words sectors that contain obstacles. The effect that the enlargement angle $\gamma_{i,j}$ has on the primary polar histogram is also indicated. The sectors to the sides of each obstacle that fall within the enlargement angle are also deemed blocked and assigned a polar density value using Equation 5. The sectors that are blocked due to the enlargement angle are the filled sectors as shown in Figure 2.

2.2 *Second Stage - The Binary Polar Histogram*

When a robot moves in an environment with several narrow openings these openings can alternate between open and blocked between consecutive readings [4]. This causes the robot to keep changing its steering direction between alternative openings. To get rid of this indecisive behaviour the polar histogram is reduced to a binary polar histogram.

The binary polar histogram H^b uses a hysteresis based on two thresholds τ_{low} and τ_{high} [4]. By using these threshold values the primary polar histogram H^p's sectors are reduced from polar density values to either being free(0) or

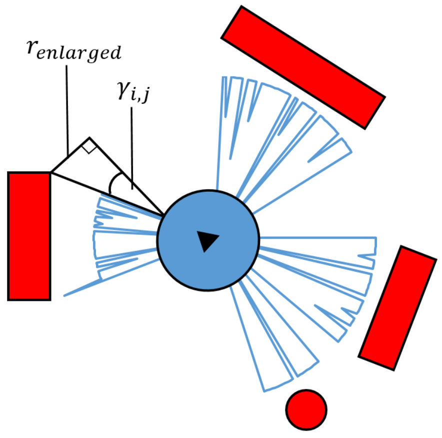

Figure 1: Primary polar histogram constructed around robot.

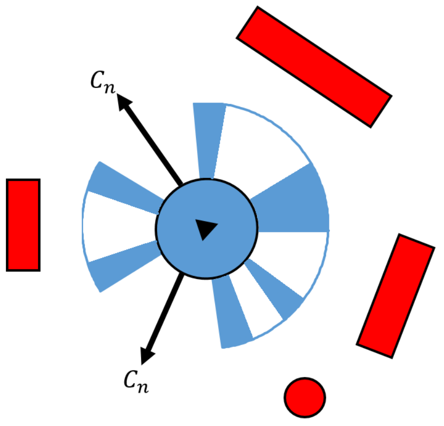

Figure 2: Binary polar histogram constructed around robot with two candidate directions.

blocked(1). The set of rules used to update the binary polar histogram H^b are shown:

$$H^b_{k,i} = 1 \qquad \text{if } H^p_{k,i} > \tau_{high} \tag{8}$$

$$H^b_{k,i} = 0 \qquad \text{if} \qquad H^p_{k,i} < \tau_{low} \tag{9}$$

$$H^b_{k,i} = H^b_{k,i-1} \qquad \text{otherwise} \tag{10}$$

This concept is illustrated in Figure 2 where the polar histogram from Figure 1 has been reduced to a binary polar histogram where the sectors are either open or closed.

2.3 *Third Stage - The Masked Polar Histogram*

VFH* approximates robot trajectories as circular arcs (constant curvature curves) to account for the fact that most mobile robots are unable to change their direction of motion instantaneously and are subject to turning circles [4]. The curvature of a curve is defined as $\kappa = 1/r$ where r is the steering radius. The values for minimum steering radii of a robot can be found experimentally and are functions of velocity. The steering radii to the right and left of the robot are defined as $r_r = 1/\kappa_r$ and $r_l = 1/\kappa_l$.

If the trajectory circle and an enlarged obstacle cell overlap all directions of motion are blocked to that side of the robot. The masked polar histogram shows which directions are accessible at the robot's current speed. If all sectors are blocked the robot would have to decrease its speed and reconstruct the masked polar histogram [4].

2.4 *Fourth Stage - Determining of Primary Candidate Directions*

By checking which sectors in the masked polar histogram are free and which are blocked primary candidate directions are identified [1, 4]. Then by evaluating each candidate direction at the hand of a cost function the new direction of motion is calculated [1].

The right and left borders, k_r and k_l, of all openings are determined to establish whether an opening is wide or narrow [4]. An opening is considered wide if the difference between its two borders is larger than s_{max} sectors ($s_{max} = 16$ in our simulations). In contrast if the difference between its two borders is less than s_{max} sectors the opening is considered narrow. Figure 2 shows a case where two narrow openings are present and two primary candidate directions exist.

A narrow opening has only one primary candidate direction which is the center of the opening:

$$c_n = \frac{k_r + k_l}{2} \qquad \text{centered direction} \tag{11}$$

A wide opening has at least two primary candidate directions, one to the right of its left border and one to the left of its right border [4]. When the opening presents a clear path to the target, that is when the target direction k_t lies between the two borders, the target direction k_t is also considered to be a primary candidate direction as shown:

$$c_r = k_r + \frac{s_{max}}{2} \qquad \text{towards the right} \tag{12}$$

$$c_l = k_l - \frac{s_{max}}{2} \qquad \text{towards the left} \tag{13}$$

$$c_t = k_t \qquad \text{if } k_t \,\epsilon\, [c_r, c_l] \tag{14}$$

The two primary candidate directions c_r and c_l make the robot follow the obstacle contour at a safe distance, while c_t leads the robot directly towards the target [4].

2.5 *Verification of Candidate Directions*

Look-ahead verification is used to verify which primary candidate direction is the most suitable direction of motion for the robot. VFH* does this by computing the new positions and orientations of the robot if it were to move a projected step distance d_s in each of the primary candidate directions [1] .

At every projected position a new primary polar histogram is constructed based on the histogram grid information available. Then the data reduction process as described previously is repeated. New candidate directions are found, called projected candidate directions. The process is repeated n_g times until finally we are left with a search tree of depth n_g, where the end nodes correspond to the total projected distance $d_t = n_g \cdot d_s$ [1] .

Every end node has a cost based on the path cost, which is the sum of the costs of the branches leading back to the start node, and a heuristic function that is implemented [1]. The primary candidate direction that leads to the end node with smallest total cost is then selected as the new direction of motion.

The larger d_t the greater the total look-ahead. If selected too high the algorithm is slowed down considerably while if its chosen to small the robot might not choose the best path going forward. Thus, selection of d_t is a trade-off between the speed and quality of the algorithm [1].

3 Modifications to VFH*

All multicopters are subject to inaccurate sensor data that cause uncertainties to arise with regards to their position and yaw estimations. The original VFH* algorithm does not currently take into consideration uncertainties in position and yaw estimation since the mobile ground robots that the algorithm was intended for have motor encoders and other mechanisms that can more accurately determine the robot's position and orientation (yaw). However, as will be shown, the VFH* algorithm can be modified to account for these uncertainties by updating the way in which the multicopter builds its histogram grid and primary polar histogram.

The VFH* algorithm is also modified to take into consideration a multicopter's pitch and roll angles. These parameters are important to consider when range measurements are taken since an obstacle may be closer than what is being measured due to the angle of measurement.

3.1 Additional Sensor Region

The current implementation of VFH* reacts to obstacles once they are within the multicopter's active region C_a. Though this makes sense for a mobile ground robot that makes use of short range ultrasonic sensors it is bothersome for multicopter system using a long range two-dimensional LIDAR system. The problem lies in choosing an appropriate value for the active region radius, $r_{active-region}$. If the region is too small the multicopter will not have sufficient time to react to obstacles. On the other hand if the region is too large the robot will start reacting to obstacles much earlier than what is necessary.

This could be addressed by tuning the thresholds when constructing the binary polar histogram as discussed in section 2.2. But this process would be a tedious task and may not always be reliable.

We present our solution in the form of an additional sensor region. The sensor region is used to update the histogram grid C and its size is chosen to accommodate the sensors range capabilities. In our simulations we chose a sensor region of radius $r_{sensor-region} = 20\text{m}$. This radius can be made much larger, especially for multicopters moving at high speeds. The size of the active region then determines when the multicopter will react to an obstacle. In our simulations we chose the active region to have a radius of $r_{active-region} = 5\text{m}$. Thus, the sizes of the two regions can vary as long as the following condition is satisfied:

$$r_{active-region} \leq r_{sensor-region} \tag{15}$$

This allows the detection of obstacles before the multicopter reacts to them. This characteristic is of utmost importance for the implementation of the suggested modifications in the following subsections. It will become clear why it is important to distinguish between these two regions as we proceed.

3.2 Adding Safety Distance to Range Data

The original VFH* algorithm made use of a safety distance d_{safety} when constructing the primary polar histogram as was shown in Section 2.1. Though this safety distance works well when the multicopter is moving around an object it has no effect on the initial distance between the multicopter and an obstacle. This is because the safety distance d_{safety} is used to calculate only the enlargement angle $\gamma_{i,j}$ in Equation 4 and is not accounted for in any way in terms of the direct distance between the multicopter and the obstacle. Using the original VFH* algorithm would mean that the multicopter would only react to an obstacle once the obstacle is within the multicopter's active region. This means that the safety distance is only considered as the multicopter moves around the obstacle. In general this works well but to ensure that the multicopter starts reacting at the appropriate distance from an obstacle we account for the safety distance d_{safety} when updating the histogram grid C as follows:

$$d_{enlarged} = d_{i,j} - d_{safety} \tag{16}$$

where:

$d_{i,j}$: Distance from active cell $C_{i,j}$ to the RCP.

By using $d_{enlarged}$ when updating the histogram grid instead of $d_{i,j}$ the multicopter will start reacting to an obstacle exactly a distance of d_{safety} sooner than it would when d_{safety} is only included in the enlargement angle $\gamma_{i,j}$. It is important to note that this improvement is only possible because we have a sensor region. This allows the histogram grid to be updated before an obstacle enters the multicopter's active region. The correction made in Equation 16 means that the multicopter will detect the obstacle a distance of d_{safety} sooner than normal. This however means that the effective distance between the multicopter's center point and the obstacle is:

$$d_{effective} = r_{active-region} + d_{safety} \tag{17}$$

3.3 Compensating for Multicopter Position Uncertainties

Most multicopters make use of GPS systems to determine their positions. Accurate GPS and vicon systems are often very expensive and even then one can never be 100% certain of the multicpoter's position. This poses a problem for the VFH* algorithm since it constructs its primary polar histogram from the histogram grid C based on the multicopter's position on the histogram grid.

To account for the uncertainties in a multicopter's position we first acknowledge that the multicopter can be anywhere within a circular uncertainty region around its estimated position as shown in Figure 3. From the figure one can see that the multicopter would fit through the gap between the two obstacles if its position is 100% certain. But since that is unlikely one has to consider the uncertainty region around the multicopter which means that the multicopter might very well be on a collision course with one of the two obstacles. The radius of the uncertainty region may vary depending on the type and quality of sensor used as mentioned earlier. We define the radius of the position uncertainty region as $r_{position}$. The uncertainty region is accounted for in the VFH* algorithm by modifying the range readings from the two-dimensional LIDAR range sensor. This can be done by subtracting $r_{position}$ from the LIDAR range measurement when updating the histogram grid C:

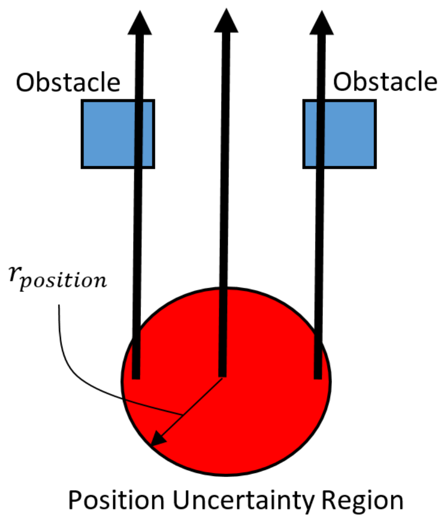

Figure 3: Multicopter position uncertainty.

$$d_{enlarged} = d_{i,j} - d_{safety} - r_{position} \qquad (18)$$

where:

$d_{i,j}$: Distance from active cell $C_{i,j}$ to the RCP.

This enlarges the area an obstacle occupies in the histogram grid and allows the uncertainty region to be shifted from the multicopter to the obstacles as illustrated in Figure 4. This is however only half the solution as this does not enlarge the sides of the obstacle to account for the multicpoter's position uncertainty. To account for that we need to modify the enlargement angle $\gamma_{i,j}$ from Equation 4 used to construct the primary polar histogram:

$$r_{enlarged} = r_{multicopter} + d_{safety} + r_{position} \qquad (19)$$

$$\gamma_{i,j} = \arcsin\left(\frac{r_{enlarged}}{d_{enlarged}}\right) \qquad (20)$$

Thus, the multicopter will start reacting sooner to an obstacle to compensate for inaccuracies in its position estimation. The multicopter's position uncertainty is also compensated for when it wants to move around obstacles since we have included $r_{position}$ term in the enlargement angle $\gamma_{i,j}$ as shown in Figure 4. In so doing the multicopter's position uncertainty is taken into account when VFH* algorithm calculates candidate directions.

Figure 5 shows a simulation of the VFH* algorithm taking into consideration various position uncertainties. It can be observed that the multicopter reacts sooner when the uncertainty is more due to Equation 18 while the increase in the size of the arcs as the robot moves around the obstacle are the consequence of Equation 20. The effective distance between the multicopter and the obstacle where the multicopter starts reacting to the obstacle can be updated as follows:

$$d_{effective} = r_{active-region} + d_{safety} + r_{position} \qquad (21)$$

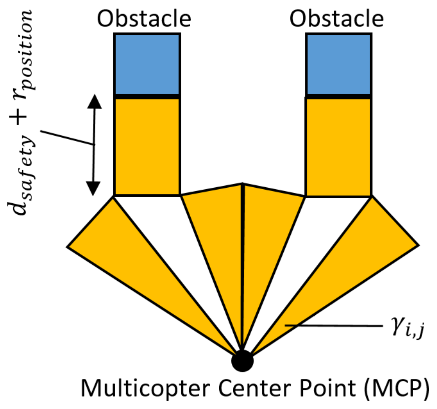

Figure 4: Enlarged obstacles with multicopter as point-like vehicle.

Figure 6 on the other hand shows the effect of three position uncertainties when the multicopter attempts to traverse terrain with clustered obstacles. The multicopter manages the shortest root when no uncertainty is present, $r_{position} = 0$m, while it is still able to move in between two obstacles which are slightly further apart when $r_{position} = 1$m. However when the uncertainty is increased even more to $r_{position} = 4$m the multicopter can no longer be certain that it won't collide with any of the obstacles and thus appropriately chooses to go around the cluster.

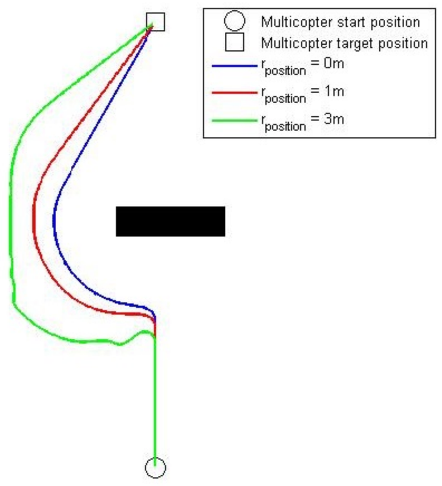

Figure 5: Effect of various position uncertainties on multicopter trajectory.

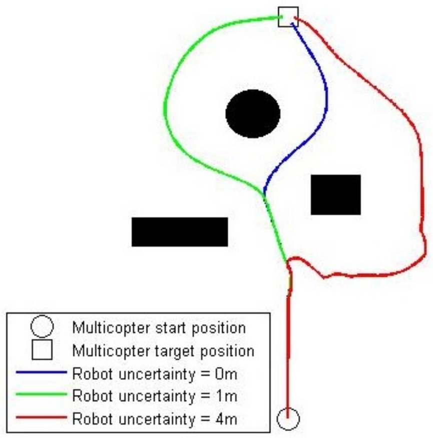

Figure 6: Effect of position uncertainty on multicopter trajectory in a cluttered environment.

3.4 *Compensating for Multicopter Tilt Angle*

One way in which multicopters are entirely different from mobile ground robots is that they use pitch and roll manoeuvres to move forward or to the sides. This also means that unlike mobile ground robots they do not have to change their orientations (yaw) to change direction as they can easily move in any direction through a combination of pitch and roll manoeuvres.

A potential problem that arises with implementing VFH* on a multicopter using a two-dimensional LIDAR range sensor is that if a gimbal is not used the tilt angle of the drone will affect the range measurements made by the sensor. Figure 7 shows a multicopter as seen from the side. If the multicopter is stationary while taking a range measurement the true distance to the object would be the distance R_1, ignoring sensor inaccuracies of a few centimetres. However, as the multicopter starts to move either towards the target, or away from it, a tilt angle θ_{tilt} is introduced and the sensor range measurement s would indicate that the object is further away from the multicopter than it actually is. This error is dependant on the obstacle distance to the multicopter and may even be insignificant if the multicopter travels at low speeds, causing small tilt angles (θ_{tilt}), or if the obstacle is very close to the multicopter, causing short range readings (s).

The effective range of an obstacle R_1 is easily calculated as:

$$R_1 = s \cdot \cos(\theta_{tilt}) \tag{22}$$

The concept is exactly the same for pitch and roll manoeuvres. Thus, the error caused by a multicopter's tilt angle during either pitch or roll manoeuvres can easily be corrected by calculating the effective range R_1 and using the effective range instead of the actual measured range s to update the histogram grid C. It is important to note that while the one end of the multicopter might be lifted by an angle θ_{tilt} the other end will be pointing down by an angle θ_{tilt} and every other range reading will be at an angle in the range $0^\circ \leq \theta < \theta_{tilt}$. Figure 8 shows a simulation of the VFH* where the tilt angle

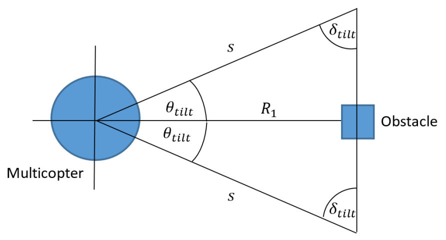

Figure 7: Effect of pitch angle on range readings.

was varied from 0° to 60° in increments of 15°. It is important to note that the obstacle is shown in the position where the two-dimensional LIDAR sensor initially measures it to be before correcting the measured distance. Thus the obstacle would make up a region much closer to the multicopter on the histogram grid for each of the cases where the tilt angle is greater than zero. In all the cases the multicopter only reacts to the obstacle once it is detected in its active region. The effect of the tilt angle correction can clearly be seen as the multicopter starts reacting sooner on the figure as the tilt angle increases. As the tilt angle is increased the measured distance is corrected and the object is detected closer to the multicopter's starting position than what was initially measured.

3.5 *Compensating for Multicopter Yaw Uncertainties*

Uncertainties in a multicopter's estimated yaw angle effect the VFH* algorithm in a similar fashion as pitch and roll do. Figure 9 shows a top view of a multicopter with the yaw uncertainty, $\theta_{\Delta yaw}$, defined as the angle by by which the multicopter may differ from it's estimated yaw angle. It is impor-

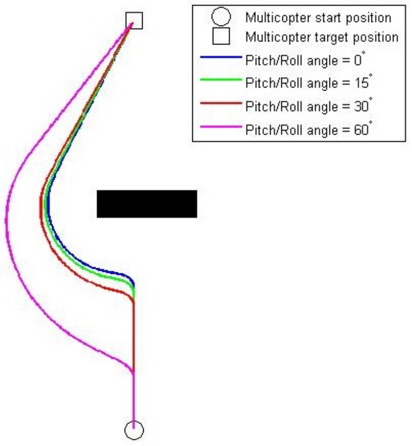

Figure 8: Effect of tilt angle on multicopter trajectory.

tant to note that we are referring to the uncertainty of the yaw angle and not the yaw angle itself. This is different from the roll and pitch angles in that errors in yaw angle estimation occur at random and have an influence on not only the perceived distance of an obstacle but also its perceived width.

To account for the effect yaw uncertainty has on the perceived distance of an obstacle to the multicopter we repeat the process followed in Section 3.4 as shown:

$$R_2 = s \cdot \cos(\theta_{\Delta yaw}) \tag{23}$$

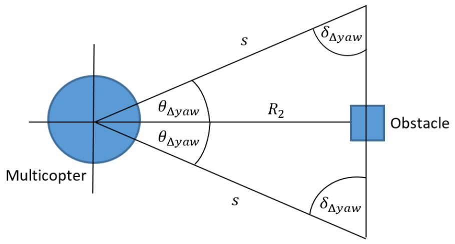

Figure 9: Effect of yaw angle on range readings and obstacle uncertainty.

R_2 is then the corrected distance from the multicopter to an obstacle. However if the multicopter has a tilt angle as shown in Figure 10 then we also need to account for θ_{tilt} when calculating the corrected distance, R_2:

$$R_2 = s \cdot \cos(\theta_{tilt}) \cdot \cos(\theta_{\Delta yaw}) \tag{24}$$

To account for the effect yaw uncertainty has on the width of an obstacle we include Δx, as shown in Figure 10, in the enlargement angle $\gamma_{i,j}$ as shown:

$$\Delta x = R_1 \cdot \sin(\theta_{\Delta yaw}) \tag{25}$$

then

$$\Delta x = s \cdot \cos(\theta_{tilt}) \cdot \sin(\theta_{\Delta yaw}) \tag{26}$$

$$r_{enlarged} = r_{multicopter} + d_{safety} + r_{position} + \Delta x \tag{27}$$

$$\gamma_{i,j} = \arcsin\left(\frac{r_{enlarged}}{d_{enlarged}}\right) \tag{28}$$

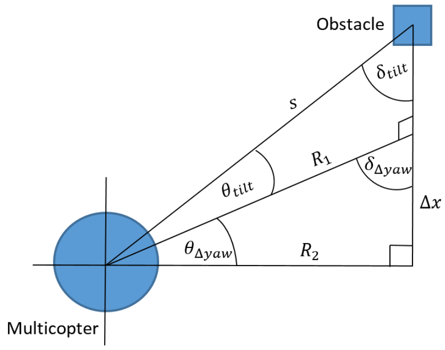

Figure 10: Combined effect of tilt and yaw angles on range readings and obstacle uncertainty.

3.6 *Effects of Steering Control and Search Depth*

Since multicopters do not have to change their orientation (yaw angle) to change their direction of motion the need for steering control in our simulations was questioned. Figure 11 shows the effect of different proportional steering control values on the VFH* algorithm. In this case having no steering control, $K_s = 0$, still produces the correct multicopter trajectory. However, this would only be a valid trajectory if the multicopter is travelling at low speeds. At higher speeds dynamic effects such as the multicopter's momentum would make it impossible to change its direction of motion instantaneously. Thus, if steering control is ignored while the multicopter is moving at moderate to high speeds the multicopter will behave differently than what the VFH* algorithm expects and can cause the multicopter to crash or hit an obstacle. In another observation it was found that the VFH* algorithm sometimes behaves in an unsatisfactory fashion when we neglect steering control from our simulations. The problem is illustrated in Figure 12 where the multicopter, without steering control, first moves to the right. As it moves to the right it does not immediately find a clear path to the target position and the path cost becomes increasingly expensive until the multicopter eventually turns around and moves left. In the end it does successfully move around the obstacle. The initial choice to go right is because we used a circular sensor and active region. This means that only a small part of the obstacle is initially detected. The detected part of the obstacle was inline with the multicopter and the target position as can be seen from the figure. Thus, the choice to go right was most likely made arbitrarily since not enough information was available.

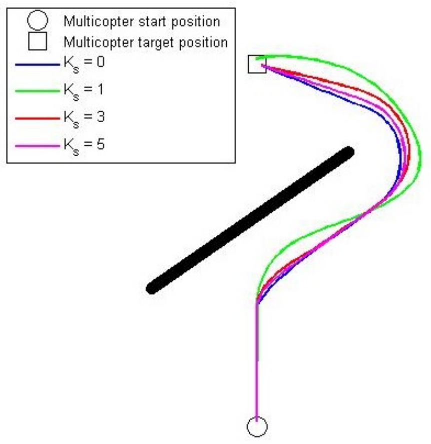

Figure 11: Effect of steering control proportional constant K_s on multicopter trajectory.

When steering control is included in the algorithm the multicopter cannot change its path so easily and commits to moving right until it can move around the obstacle. The reason for this is that if we do not consider steering control the multicopter can change its direction of motion instantaneously even though the cost functions used by the original VFH* does make it expensive for the multicopter to make large changes in its direction of motion. In the above de-

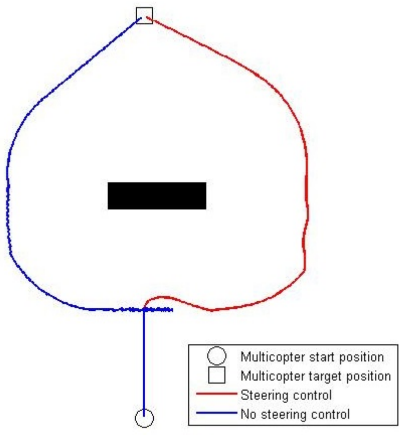

Figure 12: Effect of steering control on multicopter trajectory.

scribed scenario the look-ahead verification search depth was chosen to be $n_g = 5$. If this is increased to $n_g = 10$ the VFH* algorithm determines that the least expensive path would in fact be to the left and makes the correct decision from the start as shown in Figure 13 since it has more information to make an informed decision.

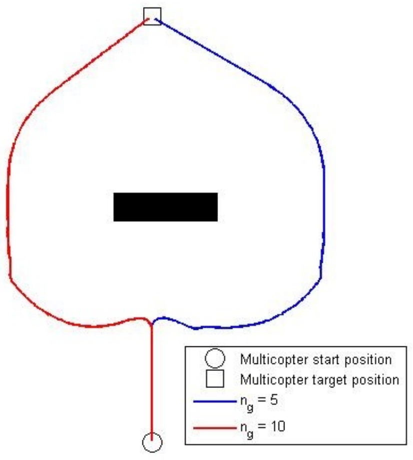

Figure 13: Effect of various search depths on multicopter trajectory.

4 Conclusion

This paper presented an overview of the VFH* algorithm and discussed the effects of position and yaw uncertainties. It also addressed the influence of tilt angles on the range measurement of obstacles when a two-dimensional LIDAR system is used. It was found that the VFH* algorithm is indeed applicable to multirotors and can be modified to account for position and yaw uncertainties. The algorithm can also be modified to consider the tilt angle of a multirotor as it builds its histogram grid with range measurements for a two-dimensional LIDAR system. It was also shown that the VFH* can still benefit from steering control and that the algorithm's search depth n_g is still an important parameter when the best path around obstacles needs to be determined.

References

[1] I. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 3:2505–2511, 2000.

[2] Johann Borenstein and Yorem Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

[3] Johann Borenstein and Yoram Koren. The Vector Field Histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[4] Iwan Ulrich and Johann Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. pages 1572–1577, 1998.