Simulation of an Obstacle Avoidance Algorithm in a Dynamic 2D Environment

J.S Coetzee*, Dr. WJ Smit†
Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Stellenbosch, South Africa

http://www.imavs.org/

ABSTRACT

This paper describes the methods followed to more accurately predict how a combination of D* Lite and the Virtual Force Field method would react when sensor as well as pose uncertainties are considered during the map building process. These uncertainties are then simulated using MATLAB where the main focus is the effect that these uncertainties have on the combined algorithms' output and whether the same results can be obtained by using simplified assumptions.

1 INTRODUCTION

Obstacle avoidance algorithms can be broadly classified into two different categories, global and local path planning [1]. Global path planning is mainly done offline, whereas local collision avoidance must be fast, reactive and usually carried out online to ensure the safety of the vehicle by compensating for previously unknown obstacles.

The key distinction between the two obstacle avoidance categories is the amount of information available regarding the robots' immediate environment. If all the information regarding the obstacles is available, global path planning will be the best approach. On the other hand, if the information regarding the environment is incomplete or unreliable, local information based on sensor data will have to be used to navigate [2].

Apart from using one or a combination of the two approaches to provide a basic solution, there are other criteria the algorithms must preferably satisfy. These criteria include the robot's dynamics, implementation time, limited computational resources, robustness as well as the optimality of the code being implemented [1].

Although many techniques for both local and global collision avoidance have been proposed in the recent literature [1, 2, 3], there is still a great interest around the globe to better address obstacle avoidance.

The aim of this paper is to simulate a combination of two different avoidance algorithms with a probability map building technique to see how it will react in a dynamic environment. Therefore, it was decided to use a slower global path planning algorithm (D* Lite) with a fast reacting local obstacle avoidance algorithm (Virtual Force Field) while building a map of the surroundings by taking into account sensor noise as well as pose uncertainty.

In Section 2 a general background regarding the map building process used in this paper is given. This is followed by an overview of the Virtual Force Field method as well as D* Lite. The simulations, as well as their results, will be discussed in Section 3 followed by the conclusion and future work in Section 4.

2 BACKGROUND

2.1 Map Building

In an ideal world, hard assignment regarding the occupancy of a grid cell (one for occupied and zero for empty) can be used since everything about the robot and sensor is known. However, sensor measurements are noisy, which implies that the position of the robot is not accurately known. Therefore, it would be more realistic to assign a probability value to each grid cell, representing the certainty of a cell being occupied.

Since the sensor is mounted on the multicopter, all measurements received will be given relative to its pose at that time [4]. Suppose, m_i is the outcome of cell i being occupied given the current pose and measurement (z) from the multicopter and sensor respectively. Assuming, for now, that the multicopter estimates its pose correctly, the grid update formula is given by [5]:

$$\log\left(\frac{p(m_\mathrm{i}|z_{1:\mathrm{t}})}{p(m_\mathrm{i}^\mathrm{c}|z_{1:\mathrm{t}})}\right) = \log\left(\frac{p(m_\mathrm{i}|z_\mathrm{t})}{p(m_\mathrm{i}^\mathrm{c}|z_\mathrm{t})}\right) + \log\left(\frac{p(m_\mathrm{i}|z_{1:\mathrm{t\text{-}1}})}{p(m_\mathrm{i}^\mathrm{c}|z_{1:\mathrm{t\text{-}1}})}\right) - \log\left(\frac{p(m_\mathrm{i})}{p(m_\mathrm{i}^\mathrm{c})}\right) \quad (1)$$

The last term in Equation 1, also given in Equation 2, is known as the prior probability log-odds ratio of a cell. In the absence of any prior knowledge regarding the environment, this term is set to zero, i.e. $p(m_\mathrm{i}) = 0.5$.

$$\log\left(\frac{p(m_\mathrm{i})}{1-p(m_\mathrm{i})}\right) \quad (2)$$

The second term on the right-hand side of Equation 1 is the probability of cell i being occupied. This value takes into

*Email address: 16484649@sun.ac.za
†Email address: wjsmit@sun.ac.za

http://www.imavs.org/

account all previous measurements regarding cell i whereas the first term on the right-hand side of Equation 1, also given in Equation 3, is known as the inverse sensor model. This term only contains new information given by the sensor at time t. The main reason log-odds is used is to enable the update equation to integrate new measurements into the grid through addition, avoiding truncation errors as well as the multiplication of probabilities.

$$p(m_{\mathrm{i}}|z_{\mathrm{t}}) \tag{3}$$

To convert the log-odds ratio (given on the left hand side of Equation (1)) back to a probability value, the following steps can be taken:

$$\lambda = \log\left(\frac{p(m_{\mathrm{i}}|z_{1:\mathrm{t}})}{1 - p(m_{\mathrm{i}}|z_{1:\mathrm{t}})}\right) \tag{4}$$

$$p(m_{\mathrm{i}}|z_{1:\mathrm{t}}) = \frac{e^{\lambda}}{1 + e^{\lambda}} \tag{5}$$

Therefore cell i will be updated accordingly, taking into account all prior knowledge regarding obstacles as well as new sensor measurements.

Although this algorithm assumes a static environment, it will also update the map sub-optimally for moving obstacles by creating a smeared representation in the evidence grid [4]. For more information about the inverse sensor model and the pose uncertainty of the multicopter, refer to Section 3.

2.2 *Virtual Force Field*

This Virtual Force Field is a global path planning algorithm with local collision avoidance. Historically, this algorithm was mainly used for ground robots and only recently started being used in multicopters. One of the most significant benefits of this method is the fact that no knowledge of the multicopter model is neccesary [6]. The Virtual Force Field function consists of an attraction field that pulls the multicopter to the goal position and a repulsive field eminating from obstacle to ensure that it travels safely [1]. The multicopter can therefore be seen as a particale with a positive electrical charge. If all the obstacles are given a positive charge as well, there will be a repulsive force on the multicopter. On the other hand, if the goal is given a negative charge, there will be an attraction force on the multicopter [7]. This method then calculates the resultant force along with the resultant direction of the combined forces to plan its next position as seen in Figure 1.

An alternative way of implementing this method is by taking the gradient of the resultant force to calculate the next position. This can be compared to a marble on the floor. If the floor is slanted in a certain direction, the marble will start rolling till it reaches the bottom [7], as seen in Figure 2.

By using the gradient method, a path to the goal position can be generated by checking the neighbouring values at each

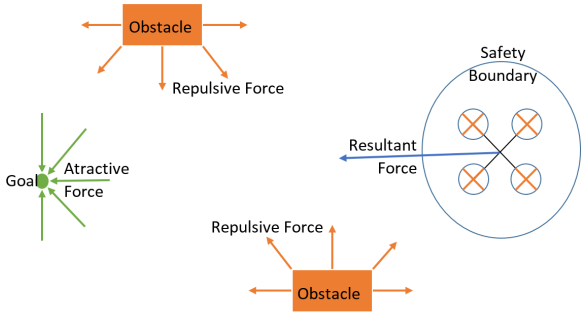

Figure 1: Potential field vector diagram.

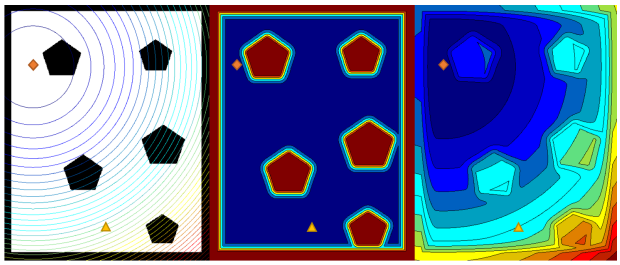

Figure 2: Potential field gradient diagram. The triangle and the diamond respectively represents the start position goal position of the robot. The first image represents the attractive force around the goal position, while the second image represents the repulsive force from each obstacle. The last image represents the sum of these forces.

cell and continuously moving to the cell with the lowest F_{total} value, where F_{total} is given by:

$$F_{\mathrm{total}}[i,j] = F_{\mathrm{att}}[i,j] + F_{\mathrm{rep}}[i,j] \tag{6}$$

$$F_{\mathrm{att}}[i,j] = \sqrt{(x - x_{\mathrm{goal}})^2 + (y - y_{\mathrm{goal}})^2} \tag{7}$$

$$F_{\mathrm{rep}}[i,j] = \begin{cases} 0 & \text{if } r > r_{\mathrm{safe}} \\ 2^{(r_{\mathrm{safe}} - r)} & \text{if } r < r_{\mathrm{safe}} \end{cases} \tag{8}$$

The current distance of the multicopter to the nearest obstacle is given by r where r_{safe} is the safety boundary as seen in Figure 1.

The Virtual Force Field method does not explicitly avoid moving obstacles and has a tendency to get caught in local minima. Therefore, the multicopter can get caught in cluttered environments before reaching its goal [1], but these issues can be addressed with the proper implementation of a global path planning algorithm [1, 8] as addressed in Section 3.3.

2.3 *D* Lite*

The D* Lite algorithm used in this paper is based on the second version of the D* Lite pseudo code given in [9]. According to [10], D* Lite is easy to understand and extend,

http://www.imavs.org/

while being at least as efficient as Focused Dynamic A* (D*) itself. Even though they are algorithmically different, they still exhibit the same behavior. Both algorithms search from the goal vertex to the current vertex of the multicopter by using a heuristic function to focus their search, whilst also using similar ways to minimize reordering of the priority queue [10].

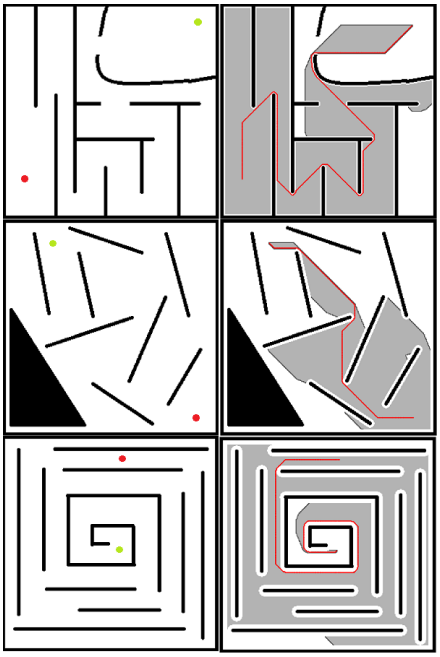

Figure 3: D* Lite global path planning (static environment). The red indicates the shortest path, while the light gray indicates all the vertices evaluated to generate the path. Dark gray indicates the vertices that will be expanded next, if needed. These paths were calculated with a 2, 4 and 6 m boundary around the obstacles respectively(1 pixel $=$ 1 m).

D* Lite is classified as a global path planning algorithm because the multicopter will have to maintain an internal map of its surroundings to calculate a path, as can be seen in Figure 3. When the map is updated with the relevant sensor data, the D* Lite algorithm will determine if the new entries into the map will have an effect on the current path and recalculate if necessary, as seen in Figure 4.

The algorithm is able to re-plan faster than planning from scratch each time as it modifies its previous search results when needed. In other words, it efficiently recalculates the shortest path from the current cell of the multicopter to the goal cell by recalculating only the relevant cells that have changed or have not been calculated before.

The algorithm has to obey two conditions [9, 10]. The first is that the heuristic function $h(s, s')$ has to be non-negative and forward-backward consistent. That is, it has to adhere to $h(s, s'') \leq h(s, s') + h(s', s'')$ for all vertices $s, s', s'' \in S$[1]. The second is that the cost of the shortest path from any vertex $s \in S$ to $s' \in S$ has to be more than or equal to the value of the heuristic function for the same vertices i.e. $h(s, s') \leq c^*(s, s')$. Both the heuristic and cost function used in the end can be described by Figure 5.

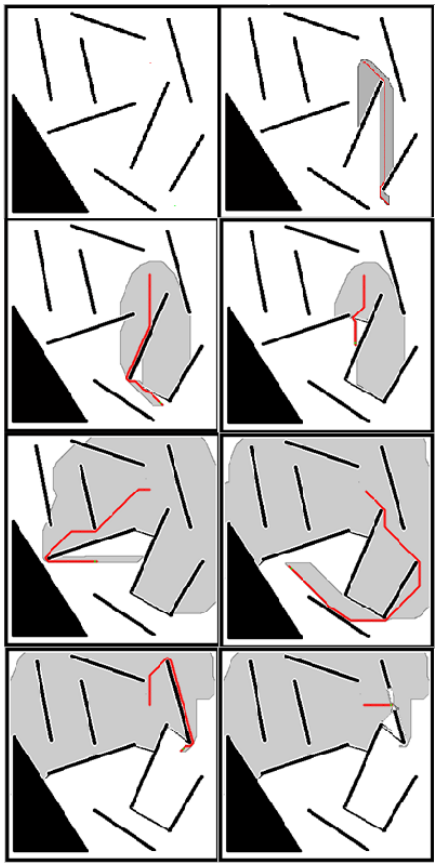

Figure 4: D* Lite global path planning (dynamic environment). The red indicates the shortest path, while the light gray indicates all the vertices evaluated to generate the path. Dark gray indicates the vertices that will be expanded next, if needed. The path was calculated with a 2 m boundary around the obstacles (1 pixel $=$ 1 m).

3 SIMULATIONS AND RESULTS

To accurately simulate how the multicopter will react when using D* Lite and the Virtual Force Field methods described in Section 2, the map of the surrounding area must be built as if the measurements were taken from an actual multicopter. To do this, the amount of measurements being taken by the lidar[2] in one revolution has to be taken into account

[1] S denotes the finite set of vertices of the graph.

[2] PulsedLight lidar sensor combined with a continuously rotating servo motor, giving up to 160 readings per revolution with a maximum distance of 40m when not rotating.

http://www.imavs.org/

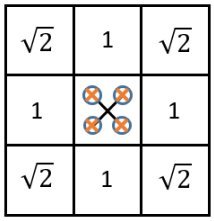

Figure 5: Heuristic and cost function used for D* Lite algorithm.

along with measurement and pose uncertainty as discussed in Section 3.1 and 3.2 respectively.

With a more realistic map of the environment being built, it will be possible to more accurately simulate how D* Lite combined with the Virtual Force Field method will work as can be seen in Section 3.3.

3.1 Incorporating Measurement Uncertainty into Inverse Sensor Model

When using Equation (1) to update the probability of a cell, an ideal inverse sensor model should return $p(m_i|z_t) = 0$ for all the cells in front of the obstacles, $p(m_i|z_t) = 1$ for the cell containing the obstacles and $p(m_i|z_t) = 0.5$ for all the cells behind the obstacles as seen in Table 1. Therefore, the function representing an ideal sensor model can be seen in Figure 6 and is defined as [5]:

$$g(r) = \begin{cases} 0 & \text{if } r < Z - \frac{L}{2} \\ 1 & \text{if } Z - \frac{L}{2} \leq r < Z + \frac{L}{2} \\ 0.5 & \text{otherwise} \end{cases} \tag{9}$$

Where r represents the distance from the current position of the multicopter to the cell that is being updated and Z represents the actual measurement from the sensor. Since the centre of the cells are used to calculate the distance to the sensor, the peak at Z may be completely missed if the centre of the cell has a slight offset from Z. To compensate for this, another parameter, L, is introduced, signifying a band of r values that receive a probability corresponding to definitely occupied ($p(m_i|z_t) = 1$). A natural choice for the value of this parameter is the diagonal distance between two corners of a grid cell.

$p(m_i\|z_t)$	$\log\left(\frac{p(m_i\|z_t)}{1-p(m_i\|z_t)}\right)$	Interpretation
0	$-\infty$	definitely free
0.5	0	unknown
1	∞	definitely occupied

Table 1: Correspondences between the probabilities and the log odds ratios along with their respected interpretations [11].

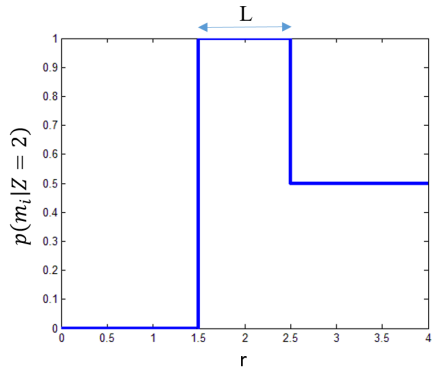

Figure 6: Ideal sensor model.

When collecting measurement data, it may be noisy. Assuming that the noise is normally distributed around the measured value Z, with a standard deviation of σ. A probability density function (PDF) of such a Gaussian distribution can then be generated, as seen in Figure 7, and is defined as:

$$f(r; Z, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r-Z)^2}{2\sigma^2}} \tag{10}$$

To incorporate the noise into the ideal sensor model, a convolution between Equation 9 and 10 had to be performed giving the following piecewise defined function [11, 5]:

$$\begin{gathered} \text{For } r \in \left(-\infty, Z - \frac{L}{2}\right) \\ (f * g)(r) = 0 \\ \text{For } r \in \left[Z - \frac{L}{2}, Z + \frac{L}{2}\right) \\ (f * g)(r) = -\frac{1}{2}\text{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r - 2Z + \frac{L}{2}}{\sqrt{2}\sigma}\right) \\ \text{For } r \in \left[Z + \frac{L}{2}, Z + \infty\right) \\ (f * g)(r) = -\frac{1}{4}\text{erf}\left(\frac{r - 2Z - \frac{L}{2}}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r - 2Z + \frac{L}{2}}{\sqrt{2}\sigma}\right) \\ -\frac{1}{4}\text{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right) \end{gathered} \tag{11}$$

Where erf represents the error function, r represents the distance from the current position of the multicopter in the map to the cell being updated and Z represents the measured distance given by the sensor. To get a better understanding of how the sensor model in Equation 11 works alongside Equation 1, refer to Figure 8 and 9. The cells in front the obstacle got their probability decreased while the cell containing the obstacle got an increase in probability whereas the cells behind the obstacle stayed unaffected as would be expected. The inverse sensor model, however, is not used as shown in

Figure 8. This is because a value of 0 will be mapped to $-\infty$, implying that new measurements will not be able to change the value of the cell any more as seen in Table 1. To avoid this problem, the sensor model was given both an upper and a lower bound value [11].

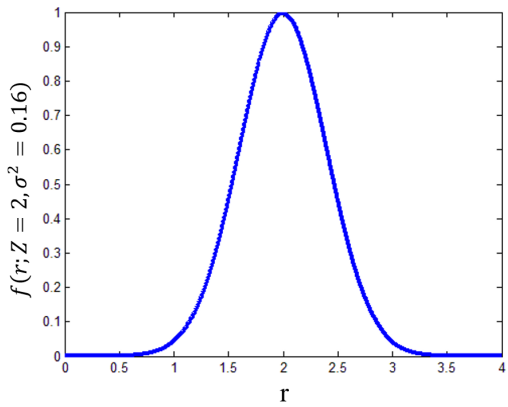

Figure 7: Probability density function.

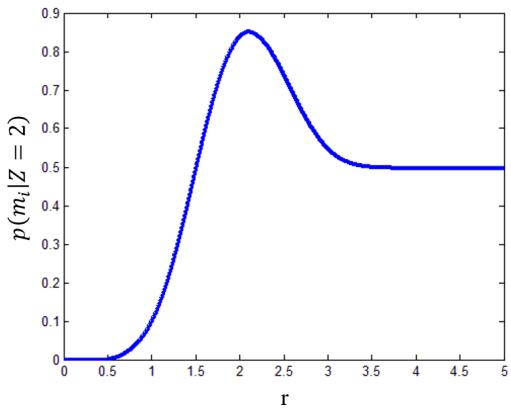

Figure 8: Gaussian inverse sensor model $L = 1$ and $\sigma^2 = 0.16$.

To see what effect the inverse sensor model has on the readings, refer to Figure 10. As the map gets darker in some places, the certainty that no obstacles are present grows while, as the map gets lighter, the certainty that an obstacle is indeed present, increases. Using a lidar as the measurement instrument, it was found that the ideal sensor model would give similar results as the Gaussian inverse sensor model if the same upper and lower bounds were implemented on it. Therefore, the Gaussian inverse sensor model derived would be more suitable for sensors with bigger measurement uncertainty like sonar or stereo vision and an ideal-like inverse sensor model can be used alongside the lidar for all intents and purposes.

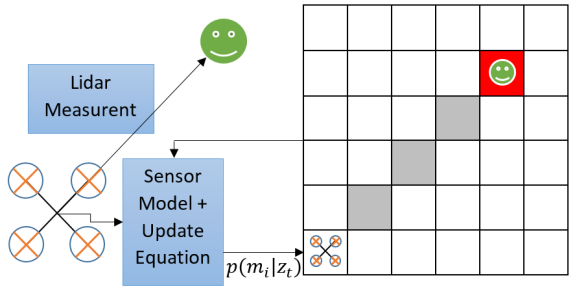

Figure 9: Map being updated by using previous map information along with new measurement information.

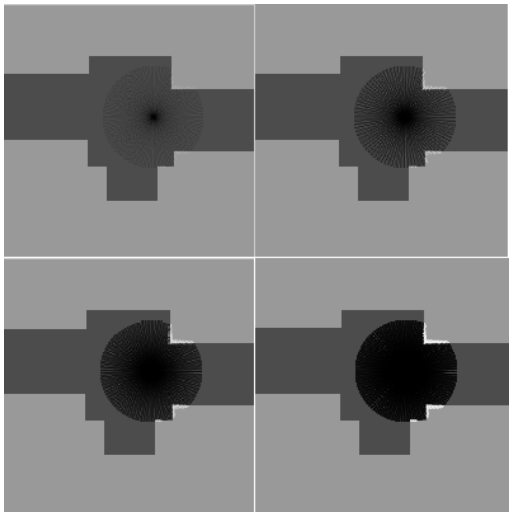

Figure 10: Map being updated using the Gaussian inverse sensor model. $L = 1.4$ and $\sigma = 0.3$. First image (top) $=$ 1 scan. Second image (top) $=$ 5 scans. First image (bottom) $=$ 10 scans. Second image (bottom) $=$ 20 scans.

3.2 *Pose Uncertainty*

To model the pose uncertainty of the quadcopter, measurements were taken with both a Piksi real time kinematic (RTK) GPS and the quadcopter controller. Since the RTK GPS is accurate up to 2 cm, horizontally, it was used as a ground truth measurement to determine the controllers' position estimation error. This data was then processed and a normal distribution was fit to it. The results of these distributions can be seen in Table 2. To incorporate this data into the model, the cumulative distribution function (CDF) first has to be calculated. Therefore, given a PDF with mean μ and variance σ^2, the CDF, as seen in Figure 11, can be generated with:

$$F_x(x; \mu, \sigma^2) = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right) \tag{12}$$

http://www.imavs.org/

	North		East	
	μ [m]	σ [m]	μ [m]	σ [m]
Test 1	-0.3813	0.2571	-0.1819	0.1864
Test 2	0.5838	0.2951	-0.0847	0.0913
Test 3	-0.6745	0.3395	0.5166	0.2779
Test 4	-1.2974	0.3270	0.7304	0.3291

Table 2: PDF distribution values obtained through testing for both the Northern and Eastern error (Given in North, East and Down (NED) coordinate system). Each test was done over a 15 min time frame.

To sample from a known one-dimensional PDF, a uniform random number, u, between 0 and 1 first has to be generated. This number can then be used to calculate the inverse of the CDF. This method is also known as the inverse transform method [5] and can be found by using:

$$x = \sqrt{2}\sigma \operatorname{erf}^{-1}(2u - 1) + \mu \tag{13}$$

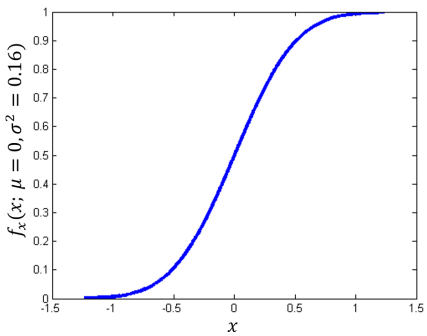

Figure 11: Cumulative distribution function.

These values are then used to update the map. It will be assumed that the same measurement could have been made from any of the positions generated by Equation 13. Since the values were randomly generated from a normal distribution, it makes sense to give an equal weight to each of them, summing to 1. To find the total value a cell will be updated with, each of the weighted probabilities affecting that cell have to be added together, refer to Figure 12 for a visual representation. Equation 1 can therefore now be re-written as follows to take into account pose uncertainty [11, 5]:

$$\log\left(\frac{p(m_\mathrm{i}|z_{1:\mathrm{t}})}{p(m_\mathrm{i}^\mathrm{c}|z_{1:\mathrm{t}})}\right) = \sum_{j=1}^{M} w_\mathrm{t}^{[j]} \log\left(\frac{p(m_\mathrm{i}|z_\mathrm{t}^{[j]})}{p(m_\mathrm{i}^\mathrm{c}|z_\mathrm{t}^{[j]})}\right) + \log\left(\frac{p(m_\mathrm{i}|z_{1:\mathrm{t}\text{-}1})}{p(m_\mathrm{i}^\mathrm{c}|z_{1:\mathrm{t}\text{-}1})}\right) - \log\left(\frac{p(m_\mathrm{i})}{p(m_\mathrm{i}^\mathrm{c})}\right) \tag{14}$$

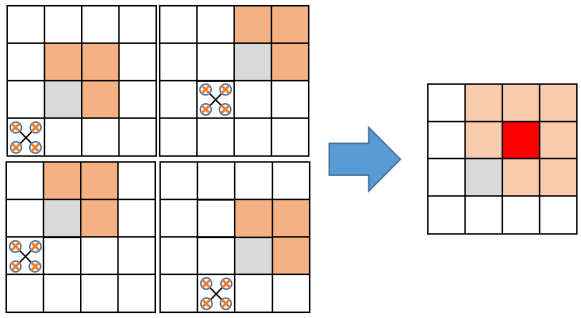

Figure 12: Illustration of how a map will be updated with pose uncertainty.

Since the maximum μ value from Table 2 is -1.2974 m and the maps are generated as images where each pixel represents 1 m, the effect of the pose uncertainty is difficult to see. One noticeable difference in the results, however, is the blurred effect that comes from Equation 14 as illustrated in Figure 12. Due to the pose uncertainty, the map generated from sensor measurements of the environment will also move in accordance with this uncertainty. To illustrate this, an error of 7 m in both the Northern as well as Eastern direction was simulated as seen in Figure 13. Since the pose uncertainty always changes when flying (as shown in Table 2), the map the multicopter builds of the environment will always be different from the previous flight as the map is updated relative the multicopters current pose.

Unfortunately, the pose uncertainty update formula is computationally intensive as a large number of samples has to be generated from the CDF to accurately represent the PDF of the multicopter's pose uncertainty each time new sensor data has to be incorporated into the map. However, it works well when simulating the uncertainty in a robots' pose. Luckily, when actually flying, this equation does not need to be used as the map will be updated according to the multicopters' pose estimation. Therefore, the map relative to the multicopter will be correct and can be used to navigate safely.

If a global map of a static environment is available and no sensor measurements can be taken, then an extra safety boundary can be added to the obstacles. In other words, this will basically enlarge the obstacles to compensate for the uncertainty in the multicopter's pose, ensuring its safety when flying without extra sensor information from the environment.

3.3 Combining Virtual Force Field with D Lite*

To be able to fly autonomously in a partially known or unknown environment, it was decided to use a combination of the three different techniques discussed in Section 2 to ensure the safety of the multicopter. As indicated previously, the idea is to let the multicopter plan a global path while still being able to react locally. D* Lite was therefore used as

http://www.imavs.org/

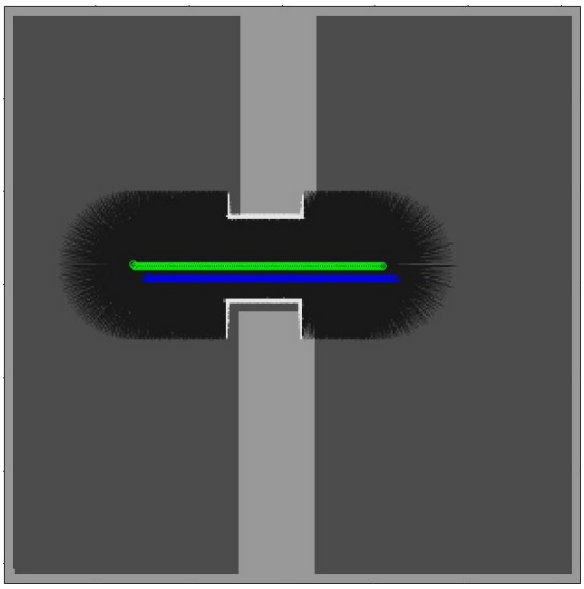

Figure 13: Illustration of how a map built by the multicopter will look over a map of the actual environment. A pose error of 7 m in both the Northern and Eastern direction was chosen for the illustration. Blue line: actual position. Green line: position estimation of the multicopter. Light gray illustrates the actual environment, whereas black and white represents the map the multicopter built. Black being no obstacles whereas white represents obstacles.

a global path planner by calculating a path from the multicopter's current position to the goal position. In a static environment, it will only have to compute the path once, but in a dynamic or unknown environment, the algorithm will be executed continuously until the goal position is reached. Even though the path may stay the same, the algorithm still has to check whether the addition or removal of obstacles affects the path after every scan. This continuous re-checking/planning can therefore get computationally intensive. One way of limiting this is by only re-checking/planning every few steps. The only problem with this is that if an obstacle is scanned, blocking the path to the goal while D* Lite is not active, the multicopter will fly straight into it.

For this reason it was decided to combine the global path planner with the Virtual Force Field method. By combining these two methods, they can help compensate for each others shortcomings i.e. D* Lite won't have to continuously recalculate, while the Virtual Force Field won't get stuck in local minima as the multicopter is being "pulled" to the goal position by making temporary attraction points on the calculated global path.

With the algorithm as described above, the amount of steps that can be taken before D* Lite has to recalculate the path, can be specified. If these steps are more than the sensor range, the multicopter will still move along the previously calculated path and update the map as usual. If an unexpected obstacle appears on this path, the map will be updated accordingly while the Virtual Force Field will assure the safety of the multicoper by maintaining the necessary safety distance.

The safety boundary used by D* Lite has to be the same as the boundary used by the Virtual Force Field, otherwise D* Lite might calculate a path that goes through obstacles that are too close to each other, causing the multicopter to still get trapped in a local minima.

Both D* lite and the Virtual Force Field does not specifically avoid moving obstacles, but since the map building process will generate a smeared representation for a moving obstacle, they will be avoided sub-optimally until the map is updated correctly again.

4 CONCLUSION AND FUTURE WORK

In Section 3 a map building technique was successfully developed and implemented by taking into account both sensor and pose uncertainties. For the sensor model it was found that the Gaussian inverse sensor model gave similar results to the ideal sensor model if the same upper and lower bounds were applied. This is only the case because a lidar with small measurement uncertainty was simulated. The Gaussian sensor model, however, could still prove to be useful when used alongside sensors with bigger measurement uncertainties like sonar or stereo vision. With the multicopter's pose uncertainty, it was found that the update model created a blurred representation of the obstacles. As each lidar reading update required a significant amount of samples from the CDF to accurately represent the PDF, it became computationally intensive. Because the map is constantly updated according to the new lidar information, the uncertainty in the pose (and therefore the update equation) is not needed for the map building process when flying. Therefore, the map relative to the multicopter can be assumed correct and safe to travel. If, however, a multicopter has to be flown autonomously without a distance sensor, a map of the environment can be used if an additional uncertainty is added around the obstacles. Thus, moving the uncertainty from the multicopters' position to the obstacle and ensuring the safety of the vehicle.

With a constantly updating map in place, a combination of both D* Lite and the Virtual Force Field method was successfully implemented. Due to the fact that the D* Lite algorithm can get computationally intensive when flying in a constantly changing or unknown environment, it was decided to only recalculate the global path every few steps, when needed. The multicopter therefore follows the global path by making a temporary attraction point on it as it moves. When an unforeseen obstacle appears, the Virtual Force Field will ensure that the multicopter stays at a safe distance until a new global path can be calculated. Therefore, these two algorithms compliment each other since D* Lite is now less computationally intensive and the Virtual Force Field method does not get

http://www.imavs.org/

stuck in local minima any more. The code is, however, not without its flaws. If the safety boundary generated by D* Lite is smaller than the safety boundary generated by the Virtual Force Field, the algorithm can still fall into a local minima as the global path will be able to go through denser packed obstacles. This, however, can easily be fixed by always ensuring they are the same value.

In future work, the algorithm will be integrated with the flight controller and other components as depicted in Figure 14. Extensive tests will then be done on the multicopter simulator, jMAVSim, to see how the multicopter will react in various situations. Lastly, actual flight tests will also be completed to verify the simulated results as well as the assumptions made.

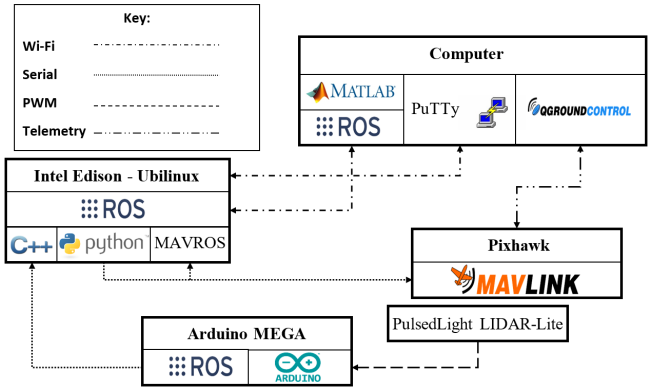

Figure 14: Hardware and software setup along with communication protocols used.

REFERENCES

[1] Anusha Mujumdar and Radhakant Padhi. Evolving Philosophies on Autonomous Obstacle/Collision Avoidance of Unmanned Aerial Vehicles. *Journal of Aerospace Computing, Information, and Communication*, 8(2):17–41, 2011.

[2] Michael Hoy, Alexey S. Matveev, and Andrey V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03):463–497, 2015.

[3] C. Goerzen, Z. Kong, and B. Mettler. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65–100, 2010.

[4] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Srikanth Saripalli. Flying Fast and Low Among Obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2023–2029. IEEE, 2007.

[5] Daniek Joubert, Willie Brink, and Ben Herbst. Pose Uncertainty in Occupancy Grids through Monte Carlo Integration. *J Intell Robot Syst Journal of Intelligent & Robotic Systems : with a special section on Unmanned Systems*, 77(1):5–16, 2015.

[6] Tobias Paul, Thomas R. Krogstad, and Jan Tommy Gravdahl. Modelling of UAV formation flight using 3D potential field. *Simulation Modelling Practice and Theory*, 16(9):1453–1462, 2008.

[7] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path Planning Strategies for UAVS in 3D Environments. *Journal of Intelligent & Robotic Systems*, 65(1-4):247–264, 2012.

[8] David Droeschel, Matthias Nieuwenhuisen, Marius Beul, Dirk Holz, Jörg Stückler, and Sven Behnke. Multilayered Mapping and Navigation for Autonomous Micro Aerial Vehicles. *Journal of Field Robotics*, 33(4):451–475, 2016.

[9] Sven Koenig and Maxim Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 968–975. IEEE, 2002.

[10] S Koenig and M Likhachev. D* Lite. *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 476–483, 2002.

[11] Daniek Joubert. Adaptive occupancy grid mapping with measurement and pose uncertainty. 2012.

http://www.imavs.org/