Cognitive control of quadrocopter using supervisor

Fredrik J. Haugen, Martin Hansen, Rune Schlanbusch and Raymond Kristiansen
Narvik University College, Narvik, Norway

***Abstract*—In this paper we present a mathematical model of a quadrocopter, together with a complete supervisor solution for path planning to cover a designated area with a quadrocopter type UAV. The supervisor is also able to optimize and recalibrate its route in the case of unexpected events during flight, such as sudden changes in wind, or someone physically altering its position in the path. Simulations has been performed using MATLAB/SIMULINK to visualize the results, and show that the supervisor solution does indeed generate a path to cover an area initially and in the case of a sudden event, and that the quadrocopter model is able to follow it properly.**

I. Introduction

Cognitive control in UAV's and generally in any type of system is revolutionizing how we can perform certain tasks without the need of human interaction. This includes path planning, area surveillance, search and rescue operations, target tracking and much more. Cognitive control in engineering is inspired by how cognitive control manifests itself in the human brain, and as we know and experience every day, it does so in a very remarkable way [1]. The ultimate goal is that a quadrocopter should be able to sense that something happened in its path, and with help of its sensors, position tracking, virtual planning, intelligence (to some extent) and learning could do something about it immediately. Starting small, in this paper, a simple form of artificial (man-made) cognitive control for path planning and area coverage with a quadrocopter model is presented.

A. Cognitive Control

From [2] it is known that for the increasing use of autonomous vehicles in air, land and naval roles, it is necessary to know more about how the autonomous vehicles deal with different situations. Similarly, in this paper, reasoning and decision making is a point of focus. Cognitive control trough the eyes of neuroscientists and psychologists has also been examined, and [1] presents more information on this, and how it can be used in engineering. From the latter it is found that the first step used by many cognitive control designers in the past has been to look to nature to find a solution to create sophisticated engineering systems. This form of inspiration has proven to be quite valuable, since we know that not only the human brain, but all livings things, have some form of cognitive control present in their everyday lives. The idea behind this is to be able to give this ability (i.e sense, learn, adapt, think) to technological systems, somewhat in the same sense as the famous Hollywood movies iRobot, Terminator and Transformers. In our case, we aim at enabling a quadrocopter to follow a desired trajectory despite of a sudden event, such as a sudden change in position. In other words we are aiming to include cognitive abilities in the control system of the quadrocopter type UAV. This does not fit entirely into the focus of Cognitive InfoCommunications but it may be considered as a first step in this direction. For a deeper understanding on the scope and goals of cognitive control and cognitive InfoCommunications, which from this article is from an artificial perspective, see [3], [4] and [5].

B. Supervisor Algorithm

Before deriving supervisor algorithms, it is important to know what their purpose is, and how they might be applied to solve our problem. Our problem may be considered as a so-called Traveling salesman problem (TSP), where a "salesman" is to visit multiple cities (Waypoints) in the shortest distance traveled, and then return home. Looking at previous work on supervisor algorithms, we find the work [6]), where implementing an A* algorithm was used to create path planning for an UAV called LocalHawk. The A* algorithm is widely used in path planning, where there are multiple obstacles present. Other solutions often used for TSP problems, are genetic algorithms. A genetic algorithm (GA), as described in [7], is a search heuristic that is designed to mimic the processes of natural evolution. It is very common in optimization problems, and is using techniques such as mutation, selection, inheritance, and crossover. Looking at work with GA's, and more specifically on traveling salesman problems, we find the work in [8] and [9]. For an overview of how a typical GA works, the reader

This work was supported by the Norwegian Research Council and is part of the Arctic Earth Observation project 195143/I60.

All authors are with the Department of Technology, Narvik University College, Narvik, Norway. Corresponding author is Fredrik Haugen: fredrikjhaugen@gmail.com.

978-1-4799-1546-0/13/$31.00 ©2013 IEEE

is referred to [10] and [11]. A third set of algorithms, formerly known as artificial neutral networks (ANN), is also widely used as supervisory control. ANN's are models inspired by the central nervous systems of the brain, capable of pattern recognition and machine learning, where the system learns from experience. In [12], we can see how this started with a set of algorithms. From [13], we see a ANN used to solve a TSP.

C. Mathematical Model: Quadrocopter

From [14] we obtain a successful implementation of the mathematical model represented in [15] in MATLAB/SIMULINK. The latter proves the mathematical soundness of the model, and the former presents a SIMULINK implementation of the model. We can also see from [16] a successful attempt at modeling and control of an actual micro quadrocopter, based on Newtons law of motion. From previous work we find that a quadrocopter can consist of a small circuit board with a micro-controller, an inertial measurement unit (IMU), some form of power source, and four motors, propellers, and electronic speed controllers. The vehicle also needs some form of communication for controlling start/stop (the rest is autonomous), and possibly also telemetry. For a more complete overview of a possible design look at [17].

D. Contribution

This paper presents the main results of [11], and contains a method to model, compute and simulate a desired trajectory for a quadrocopter to cover an area, initially, and in the case of a sudden change in position, without the use of a GPS (typically indoor). The user defines the area to be covered, and the Field-of-view (FOV) angles of the camera, and then the solution generates a list of waypoints, sorted by a cognitive genetic algorithm to make sure that the mathematical model gets an near-optimal route. Since we at all times know the distance between the position vector and the desired position vector we have so called perception, and can determine exactly where we are. The results have been simulated in MATLAB and SIMULINK to give a visual demonstration of the solution.

II. Modeling Approach

A. Mathematical Model: Quadrocopter

For the basis of our mathematical model of a quadrocopter, the solutions in [14] and [15] has been used, since they are proven to work for simple trajectories. Since our model is a quadrocopter which is supposed to fly/hover in midair, the absolute linear position is defined in the inertial frame x,y,z -axes with vector $\boldsymbol{P}$, and the rotation on the different axes are defined as vector $\boldsymbol{N}$. These vectors are denoted as

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{N} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{1}$$

where ϕ is the roll angle and determines the rotation around the x-axis, θ is the pitch angle which determines the rotation around the y-axis, and ψ is the yaw angle which determines the rotation around the z-axis. We can now derive the linear and rotational velocity by differentiating the position vector as

$$\dot{\mathbf{P}} = \mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{2}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix} = \mathbf{W}\dot{\mathbf{N}} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & -C_\phi & S_\phi C_\theta \\ 0 & -S_\phi & C_\phi C_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3}$$

where $S.$, $C.$ and $T.$ denotes $\sin(\cdot)$, $\cos(\cdot)$ and $\tan(\cdot)$, respectively. Differentiating one more time, we obtain translational and rotational acceleration as

$$\ddot{\mathbf{P}} = \mathbf{a} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \qquad \dot{\boldsymbol{\omega}} = \begin{bmatrix} \dot{\omega}_\phi \\ \dot{\omega}_\theta \\ \dot{\omega}_\psi \end{bmatrix}. \tag{4}$$

Since the matrices are indeed orthogonal the transformation matrix for angular velocities from inertial to body frame, is also the transpose of $\boldsymbol{W}$ which is $\boldsymbol{W}^T$ as shown in [15],

$$\mathbf{W^{-1}} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix}. \tag{5}$$

As shown in [17] the idea behind a 4-rotor vehicle which we choose to call a quadrocopter, is to remain stable in mid air by the use of 4 rotors. Rotors 1 and 3 are spinning in one direction, while rotors 2 and 4 are spinning in the opposite direction. While all these rotors are spinning with the same angular velocity, they produce equal amount of torque, and thrust towards the center of their rotation, at the same time, and therefore the quadrocopter will remain stable. Each of the sets of motors (1,3 or 2,4) controls one of the axes, either roll or pitch. By increasing the speed in one of these rotors, it will maintain the torque to keep the yaw stable, but at the same time produce a net torque about the roll/pitch axis. Movement in yaw is acquired by increasing the angular velocity in two opposite rotors, and decreasing the angular velocity in the others. Since the quadrocopter is assumed to be, and should be a symmetric structure with four arms aligned in the body x, y-axes, the inertia

matrix is a diagonal matrix $\boldsymbol{I}$ where $I_{xx} = I_{yy}$, such that

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{6}$$

1) Thrust and Torque: As discussed earlier the combined forces of all the rotors gives us thrust $\boldsymbol{T}$ in the bodys z-axis, and the torque $\boldsymbol{\tau}_b$ consisting of the torques in the corresponding (body frame) angles

$$\mathbf{T} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \qquad \boldsymbol{\tau}_b = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{7}$$

where $\boldsymbol{\tau}_\phi, \boldsymbol{\tau}_\theta$ and$\boldsymbol{\tau}_\psi$ are the torques in the roll, pitch and yaw angular directions, respectively. Since the quadrocopter has four rotors that all rotate independently, the angular velocity of rotor n denoted by ω_n creates a force f_n in the direction of that rotors axis. The thrust and torque given by each motor is obtained as

$$f_n = k\omega_{rot,n}^2 \tag{8}$$

$$\tau_{rot,n} = b\omega_{rot,n}^2 + \boldsymbol{I}_{rot}\dot{\omega}_{rot,n} \tag{9}$$

where k is the lift constant, b is the drag constant, and $\boldsymbol{I}_{rot}$ is the rotors inertia. The acceleration $\dot{\omega}$ is usually not taken into account since its value is typically very small. All these rotors combined, form a total thrust T and torque $\boldsymbol{\tau}$ in the body axis, that can be computed as

$$T = \sum_{n=1}^{4} f_n = k\sum_{n=1}^{4} \omega_{rot,n}^2 \tag{10}$$

$$\boldsymbol{\tau}_b = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(\omega_4^2 - \omega_2^2) \\ lk(\omega_3^2 - \omega_1^2) \\ \sum_{n=1}^{4} \tau_{rot,n} \end{bmatrix} \tag{11}$$

where l is the arm length, which is the distance between the rotor and the center of the quadrocopter (center of mass). Equations 10 and 11 can now be used to create the matrix for thrust and torque, given the angular velocity ω of each rotor

$$\begin{bmatrix} T \\ \boldsymbol{\tau}_b \end{bmatrix} = \boldsymbol{\xi}\boldsymbol{\omega}_{rot}^2 \tag{12}$$

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -lk & 0 & lk \\ -lk & 0 & lk & 0 \\ b & -b & b & -b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}. \tag{13}$$

2) Newton-Euler: The dynamic capabilities of the quadrocopter is derived from Newton's second law of motion, and can be acquired as

$$\mathbf{f} = m\ddot{\boldsymbol{P}} = m\boldsymbol{a} \tag{14}$$

$$\mathbf{a}_i = -\boldsymbol{g}_i + \frac{\boldsymbol{T}_i}{m} \tag{15}$$

$$\mathbf{a}_i = -\boldsymbol{g}_i + \frac{1}{m}\boldsymbol{R}_b^i\boldsymbol{T}_b \tag{16}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m}\boldsymbol{R}_b^i \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \tag{17}$$

where here $\mathbf{f}$ is the force, $\mathbf{a}_i$ is the acceleration in the inertial frame, m is the mass of the quadrocopter, and $\mathbf{g}_i$ represents gravity. Equations (14-17) can now be written in terms of torque produced by the quadrocopter according to

$$\boldsymbol{\tau} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \tag{18}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{I}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}). \tag{19}$$

3) Aerodynamics: Since the quadrocopter is moving through the air, aerodynamic effects on the vehicle must be taken into consideration. The presented model includes the linear resistance, but other effect such as blade flapping and airflow disturbance could also be included. The linear velocities slowing the quadrocopter motion can be written in an inertial frame as

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{1}{m}\boldsymbol{R}_b^i \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \frac{1}{m}\begin{bmatrix} A_r & 0 & 0 \\ 0 & A_r & 0 \\ 0 & 0 & A_r \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{20}$$

where A_r is the air resistance, also called the inertial frame drag force coefficients for the velocities in the x, y, z-directions.

4) Stabilization of the quadrocopter: To stabilize the quadrocopter a simple PD controller is used, which has proven to be simple to implement and very effective. A general PD controller can be formulated as follows

$$u(t) = K_p e(t) + K_d \dot{e}(t) \tag{21}$$

$$e(t) = X_d(t) - X(t) \tag{22}$$

where $e(t)$ is the difference between the desired value X_d and the actual value X. The constants K_p and K_d are the proportional and derivative gains of the PD controller. Including the thrust and torque of the quadrocopter gives the PD controller for our vehicle and can be derived as

$$T = (g + K_{z,d}(\dot{z}_d - \dot{z}) + K_{z,p}(z_d - z))\frac{m}{C_\theta C_\phi} \tag{23}$$

$$\tau_\phi = (K_{\phi,d}(\dot{\phi}_d - \dot{\phi}) + K_{\phi,p}(\phi_d - \phi))I_{xx} \tag{24}$$

$$\tau_\theta = (K_{\theta,d}(\dot{\theta}_d - \dot{\theta}) + K_{\theta,p}(\theta_d - \theta))I_{yy} \tag{25}$$

$$\tau_\psi = (K_{\psi,d}(\dot{\psi}_d - \dot{\psi}) + K_{\psi,p}(\theta_d - \psi))I_{zz}. \tag{26}$$

B. Supervisor Algorithms

1) FOV algorithm: Waypoints initialization: When looking at the system as a total, the goal is to achieve some sort of path planning for our quadrocopter, which is solved by implementation of a supervisor to generate a set of waypoints. This enables the vehicle to fly to these waypoints, do whatever action it is supposed to do, and then move on to the next waypoint. When the mission is complete, it can return home or move to another desired location. The algorithm is based on a FOV subalgorithm, which calculates the necessary number of waypoints based on the FOV. This is included to make the change of camera as easy as possible, so the user only has to define the size of the area (x, y, z), and the field of view angles. In our example we used an Area of 60x100x10 m^3 (x, y, z) and FOV angles of 127 and 120 degrees (GoPro HERO HD3 Camera). The waypoints are the calculated as follows (input values are in degrees):

$$FOV1_{rad} = FOV1 \cdot \frac{2\pi}{360} \tag{27}$$

$$FOV2_{rad} = FOV2 \cdot \frac{2\pi}{360} \tag{28}$$

$$a = 2 \cdot \tan\left(\frac{FOV1_{rad}}{2}\right) \cdot z_d \tag{29}$$

$$b = 2 \cdot \tan\left(\frac{FOV2_{rad}}{2}\right) \cdot z_d \tag{30}$$

which then serves as input values to calculate the waypoint(s) $\boldsymbol{W_{xyz}}$:

$$\boldsymbol{W_{xyz}} = [x \cdot a - (a/2), y \cdot b - (b/2), z_d]\,. \tag{31}$$

The algorithm is implemented in SIMULINK and Figure 1 shows a plot of the results. However, testing with smaller areas, such as 20x20x5 m^3, it is also important to set the desired altitude low. This is because with the altitude at 5 m, and a small area, one waypoint might be enough to cover the whole area, and then it is of little meaning to use a supervisor algorithm. However placing the waypoint lower, means a smaller field of view, and thus more waypoints. This provides better resolution in the total overview, and thus gives us better mapping capabilities.

2) Traveling Salesman Problem using Genetic Algorithm: In this section we continue with waypoint computations, and the next goal is for the vehicle to be able to find the near optimal route, first initially, and then perform a recalculation if something unexpected happens during flight. To achieve this we settled with solving the traveling salesman problem using a genetic algorithm. The reason for this is that there is a lot of previous research performed on this topic, and A* is more of a collision avoidance type of planner. Neural networks, was also considered inappropriate due to the large diversity of training required for the NN to solve the problem, since it is an experience based solution. TSP has existed all the way back to the 1930's, first as a human planning problem, for the traveling salesmen. Similarly in our project, we need to visit a set of "cities", given as (x, y, z) - coordinates, and the TSP is a perfect problem for exactly this. From [9], as mentioned earlier, we see how we may achieve this in MATLAB with a genetic algorithm, and the favorable results. The algorithm has two requirements:

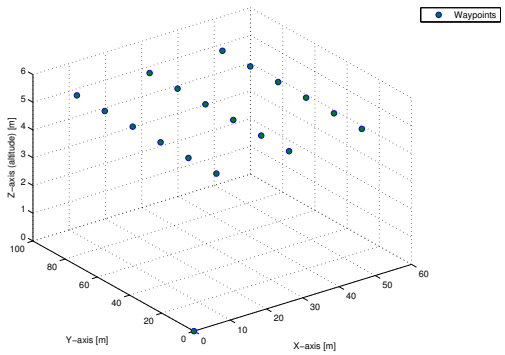

Fig. 1. 3D Waypoints Scatterplot in Simulink, achieved with the FOV algorithm

- A single traveling salesman (our vehicle) has to travel to all of the cities (waypoints) and then return to the city he started from (or another specified city).
- Each City can only be visited exactly once (apart from the initial city).

A possible first approach is to always track the waypoints, and the vehicles position. Then if something happens, e.g. the quadrocopter moves unexpectedly out of the path, the tracker could detect that we now are closer to an unvisited waypoint, other than the desired waypoint, and then recalculate the path with the genetic algorithm to a path where the closer waypoint gets visited first. Achieving this we have pretty much achieved a simplified form of intelligence, or cognitive control. The vehicle makes a decision, distant from the original one, and this is what we wanted to achieve. Another possible solution for this "sudden event"-tracker could be to monitor the accelerometer values, and look for unexplainable values, but even though this is also a good way, we choose the former method because it solves the problem and gives us continuous, real time information about our location. As

stated in II-B1 we have a set of waypoints we want to sort so the desired trajectory becomes as short as possible. Continuing on to the genetic algorithm, we need to look at how this waypoint sorting is achieved. In this particular GA; selection, elitism, and mutation is used. The term crossover, however is not used seeing that it proved to be more destructive than gaining in this problem, but we still use most of the essential components of a traditional GA, such as representation of possible solutions, fitness value evaluation of each individual member, a population of potential solutions to the problem, and a method of propagating and forming new, (and possibly) better solutions to the problem. Starting out with the evaluation of each of the population members, the fitness factor described earlier is here the total distance the member travels for its specific solution. The algorithm then selects the best member of the population based on the fitness value (minDist), and then sets this distance as globalMin. Based on the globalMin, if the distance (fitness value) a member got assigned is smaller than the globalMin, it will become the new globalMin. We call this the Elite member. Moving on, we also use the term Mutation. As seen in the code [11] we randomly group four members of the population at the time, and then send the best of these four on to the next generation. Then three different mutations will be performed on these "best-out-of-four" members, and then pass the mutated version on to the next generation.

After 1000 iterations/generations, we should have a good solution to the problem, and the GA's work is over. The list of waypoints, then gets sorted after the data from the GA, and the final list of sorted waypoints becomes available for the Position Updater, discussed in II-B3. However, should a sudden event occur, another GA similar to the first one, then runs for the remaining unvisited waypoints, but this time the initial starting point is the actual position of the quadrocopter, and the endpoint is home. This change in GA is important so the quadrocopter doesn't return to the point of the sudden event which is in mid-air rather than home.

3) Position Updater: In order to give the quadrocopter the desired list of waypoints in a suitable way, a function is used to check the distance between the actual position $\boldsymbol{P}$, and the desired position $\boldsymbol{P}_d$. When $\boldsymbol{P}$ is the same as (or close to) $\boldsymbol{P}_d$, the next waypoint gets distributed to the quadrocopter. This way we give the quadrocopter time to reach its goal, before we distribute the next waypoint. The essence of this function are the following equation

$$\boldsymbol{D}_{goal} = \boldsymbol{P} - \boldsymbol{P}_d \tag{32}$$

where $|\boldsymbol{D}_{goal}|$ is the distance between the two.

4) Sudden Event: In order to test our system for a sudden event such as if the quadrocopter position changes during the flight, we choose to statically alter the position at a desired point in time. Then the tracker, which constantly measures the distance between the actual position, and all the waypoints in the original list of waypoints, will indicate that another waypoint, different than the desired position, is closer than the desired position. By the means of a reset button, the quadrocopter will then recalculate its trajectory for the remaining waypoints, change input to the new waypoints, and continue on its new path to finalize and return home. The requirements currently in use for he sudden event trigger to trigger a recalculation are as follows:

- Assume a point is interesting if $|\boldsymbol{P}-\boldsymbol{P}_d|$ is larger than $|\boldsymbol{P} - \boldsymbol{POI}|$.
- The point of Interest (POI) can not be a previously visited point.
- The POI can not be 0.

III. Simulation Results

In this section we present the simulation results showing the system performance. For a more detailed results, see [11].

A. List of Waypoints

From Figure 1 we can see the waypoints participant in the system test. These are created by the FOV algorithm (section II-B1), and change based on different types of camera, and area-size. The problem to be solved, is to visit each and every one of these, exactly once, and then return to the initial point. Trying to simulate and achieve this we started out with calibrating the mathematical model for altitude, roll, pitch and yaw -stabilization, and as shown in [11] and [14] the mathematical model behaves like an actual quadrocopter. Onward, the cognitive control by the means of the genetic algorithm, was tested on a fully actuated model (until satisfaction), and then finally everything was implemented together.

B. 3D-Plot

Figure 2 shows a plot of the waypoints, together with the trajectory of the quadrocopter. Taking this into account, we can look at Figure 3, where we can see the plot of the waypoints together with the trajectory, where a sudden event occur at 50s into our flight. We can easily see that the quadrocopter is choosing a new route different from its initial because of the sudden event. As

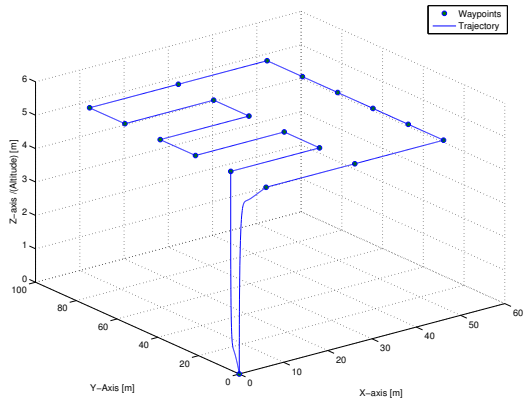

Fig. 2. Cognitive control of quadrocopter using supervisor - Trajectory

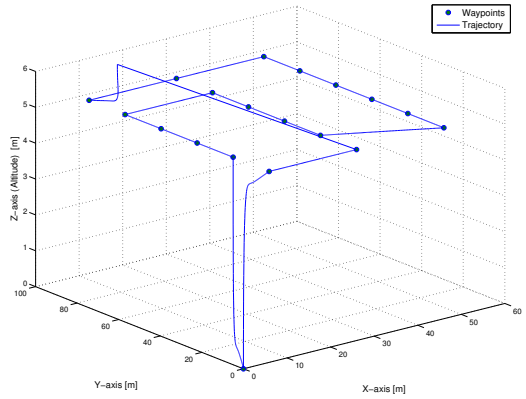

Fig. 3. Cognitive control of quadrocopter using supervisor - Trajectory with sudden event.

part of the simulation we also tested with a small sudden event that results in a new route not being initiated. This is because the sudden event did not alter the position of the quadrocopter to a point worth recalibration. The result can be seen in [11]. This demonstrates that the system only need to recalibrate if there is a better alternative than the current trajectory. If such an event happens, the quadrocopter simply returns to the ***POI*** and continues on with the current trajectory. Overall, the result was a working cognitive control of a quadrocopter using supervisor, and though improvements can be done, we can see from Figure 2 and 3 that the presented system solves the initial stated problem. Based on these simulations and the more detailed simulation results in [11], we also see that the system does indeed work, and that (some sort of) cognitive control of a quadrocopter using supervisor algorithms has been achieved.

IV. Conclusion

In this paper we have present a mathematical model of a quadrocopter, together with a complete supervisor solution for path planning to cover a designated area with a quadrocopter type UAV. The supervisor is shown to handle unexpected events during flight, such as sudden changes in wind, or someone physically altering its position in the path. Simulations has been performed using MATLAB/SIMULINK to visualize the results, and show that the supervisor solution does indeed generate a path to cover an area initially and in the case of a sudden event, and that the quadrocopter model is able to follow it properly.

References

[1] S. Haykin, M. Fatemi, P. Setoodeh, and Y. Cue, "Cognitive control," *IEEE*, vol. 100, p. 14, 2012.

[2] R. M. Taylor, "Human automation integration for supervisory control of UAV's," *Virtual Media for Military Applications, Seine, France*, 2006.

[3] J. K. Tar, I. J. Rudas, K. Kósi, A. Csapó, and P. Baranyui, "Cognitive control initiative," *3rd IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2012)*, 2012.

[4] P. Baranyl and A. Asapo, "Definition and synergies of cognitive infocommunications," *Acta Polytechnica Hungarica*, vol. 9, pp. 67–83, 2012.

[5] G. Sallai, "The cradle of cognitive infocommunications," *Acta Polytechnica Hungarica*, vol. 9, pp. 171–181, 2012.

[6] F. J. Haugen and M. Svandal, "Localhawk: Navigation GUI," Kongsberg Defence Systems, Narvik University College, 2011.

[7] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.

[8] M. LaLena. (2013) Traveling salesman problem using genetic algorithms. [Online]. Available: http://www.lalena.com/AI/Tsp/

[9] J. Kirk. (2007, January) Traveling salesman problem - genetic algorithm. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm

[10] M. Buckland, *AI Techniques for Game Programming*. Premier Press, Inc., 2002.

[11] F. J. Haugen, "Cognitive control of quadrocopter using supervisor," Master's thesis, Narvik University College, July 2013.

[12] W. P. Warren McCulloch, *"A Logical Calculus of Ideas Immanent in Nervous Activity" Bulletin of Mathematical Biophysics 5 :115-133*. Kluwer Academic Publishers, 1943.

[13] B. R. Marco Budinich, "A neural network for the travelling salesman problem with a well behaved energy function," *Dipartimento di Fisica & INFN, Italy*, p. 8, 1995.

[14] M. Hansen, "Quadcopter modeling and control," Narvik University College, 2011.

[15] T. Luukkonen, "Modelling and control of quadrocopter," *Aalto University*, p. 26, 2011.

[16] F. Solc, "Modelling and control of a quadrocopter," *Advances in Military Technology*, vol. 5, 2010.

[17] F. J. Haugen, "Mini quadrocopter design," *Narvik University College*, February 2012.