Fuzzy Logic Based Algorithm for Quadrotor Stabilized Point-to-Point Movement

Reiichiro Christian S. Nakano, Argel Bandala, Gerard Ely Faelden, Jose Martin Maningo, Elmer P. Dadios
Electronics and Communications Engineering Department, Manufacturing Engineering and Management Engineering Department
De La Salle University, Manila, Philippines
reiichiro.nakano@gmail.com, elmer.dadios@dlsu.edu.ph

***Abstract*—This paper presents an algorithm based on fuzzy logic in order to successfully move a quadrotor from any two points in air with little deviation from a planned path. The simulation considers the highly nonlinear dynamics of a quadrotor and tests the ability of the algorithm to tolerate the unpredictable effects of random wind and sensor noise on the algorithm**

***Index Terms*—fuzzy logic, quadrotor, stabilization, algorithm, wind, noise**

I. Introduction

Quadrotor unmanned aerial vehicles (QUAV) are an increasingly popular unmanned aerial vehicle. They are defined as a four-rotor helicopter, with each rotor lying in a parallel plane, forming a square whose center converges at the center of mass of the entire quadrotor [1].

Quadrotors have advantages over fixed-wing aircraft like airplanes such as the ability to do a vertical landing and takeoff in limited spaces or simply staying in place and hovering over a particular area [2]. One may also build quadrotors with significantly less expense and materials than other rotor vehicles such as helicopters [3].

One problem with using quadrotors, however, is the inherent difficulty of controlling their flight [4] [5]. With the availability of only four actuators applying a unidirectional linear thrusting force to control the vehicle's three linear and three angular movements, their dynamics in relation to the motor speed are highly complex [6]. In addition, their light weight make outside wind a very significant factor in keeping them stable. A quadrotor controller needs to be robust enough to control six positional variables and their rates of change by manipulating only four input variables, all the while being able to tolerate sensor noise, unpredictable weather, and abrupt changes in flight commands [4] [7] [8].

From this, it is clear that the control of a quadrotor is a multiple input multiple output system. It is extremely difficult, and perhaps impossible, to formulate a mathematical model that will exactly simulate how a quadrotor will fly in real life. There are numerous highly non-linear variables to take into account. Wind speed, sensor noise, rotor blade deformation, and motor winding resistance are just a few of the variables that are impossible to predict in a complete mathematical model of a quadrotor's flight. It is for this reason that in this paper, fuzzy logic was chosen as the control algorithm to be used in guiding and stabilizing a quadrotor's flight from a specific point in space to another.

In this paper, a fuzzy logic system is presented that takes as input the current positional coordinates of a quadrotor, the x, y, and z-coordinates in a Cartesian coordinate plane and the pitch degree, and yaw angles, and their rates of change. Another input in the system is the position of the destination point in 3-dimensional space. Using these input variables, the fuzzy logic will calculate each thrust force that is to be applied by each of the four rotors in a manner that will move the quadrotor to the target point stably and efficiently.

To test the effectiveness of the algorithm, the algorithm is applied to a quadrotor flight simulator. The simulator is designed in C and takes into account a simplified model of the dynamics of quadrotor flight. It also adds random variables to the simulation that represent random sensor noise and wind effects in the system. As discussed above, while this will only be an approximate mathematical model of how a quadrotor will behave in real life, the random noise variables added to the simulation account for the other non-linear factors absent in the equations and are expected to make the final results of the simulation similar to how the system will behave in real life.

Control algorithms for quadrotor flight stabilization have been proposed before, using techniques such as fuzzy sliding mode control, [8] [9] and local interaction with actuator saturation [10] [11] [12]. Swarming algorithms for different behaviors are also discussed in detail for quadrotor unmanned aerial vehicles [6].

The complete discussion of the system is arranged as follows: the first part will give a brief overview of fuzzy logic and a flowchart of the system being proposed. The second part will discuss and present the equations representing the dynamics of the quadrotor used in the simulation. The third part will discuss in detail each component of the fuzzy logic. The fourth part will show the results of the experiments performed with the algorithm. Finally, the last part will provide a conclusion and discussion of the results.

II. Fuzzy Logic

Fuzzy logic is form of soft computing that deals not with straight 1's and 0's, but with values in between. Its value lies in its applications to real life. In real life, one describes objects not only with definite characteristics such as hot or cold, but with in-between characteristics such as warm or somewhat hot. Fuzzy logic attempts to combine real-life solutions and integrate them with digital control systems that do operate on 1's and 0's. This ability to work with variables that are not exact gives fuzzy logic a considerable advantage over other problem-solving methods. Problems with complex mathematical models but are easily solved by a human operator's experience are the problems best suited for a fuzzy logic solution [11] [12].

It is for this reason that the problem presented in this paper is solved by fuzzy logic. As discussed above, the dynamics of a quadrotor are extremely difficult to resolve using mathematical equations. However, it is a relatively simple task for one to visualize that if the quadrotor is moving forward faster than appropriate, the motors must be adjusted to apply a weaker thrust force to slow down the vehicle.

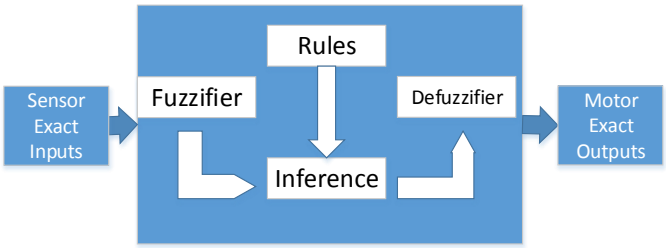

Figure 1. Fuzzy logic block diagram

Figure 1 shows the block diagram of the algorithm used in the paper. There are a total of crisp exact mathematical inputs used in the system. Six of these are the positional variables associated with the quadrotors and include the x, y, and z-coordinates of the quadrotor with respect to a fixed Cartesian coordinate frame and the pitch, roll, and yaw angles of the quadrotor. The other six are the rates of change of these angles. In the fuzzy algorithm, these crisp variables are fuzzified by mapping them into the fuzzy input sets. Using the rules set in the algorithm, the input variables generate an inference that is then mapped into the fuzzy output sets through defuzzification. There are four outputs. These four outputs are the thrusts that are applied by each motor of the quadrotor. To apply the algorithm in real life, there must be available sensors in the system to know the values of the input variables at any given time. Also, the controller must be able to apply the correct input signal to the motor in order to apply the amount of thrust that is specified by the output of the algorithm.

III. Quadrotor Flight Simulation

A quadrotor is a rigid body, most commonly modelled as a cross with a sphere at the middle, containing batteries and circuitry, and four motors to control its motion in three-dimensional space: one on each end on the cross.

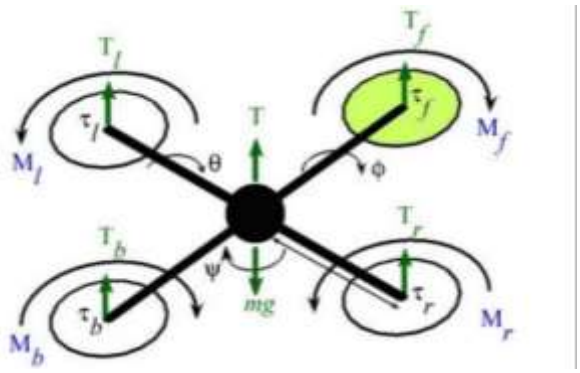

Figure 2. Quadrotor Forces

.

Controlling a quadrotor is a rather difficult and complex problem, as you only have four actuators, the motors, to control six degrees of freedom in a three-dimensional space: the translational x-axis, y-axis, and z-axis movements and the rotational roll, pitch and yaw angle movements. There are also a huge number of nonlinear effects that contribute to the quadrotor's dynamics while in the air. In order to develop a working simulation of the quadrotor, a highly simplified model of quadrotor dynamics based on the Newton-Euler equations for a rigid body is presented [8] [9] [13] [14].

$$\begin{matrix} \ddot{p}x \\ \ddot{p}y \\ \ddot{p}z \end{matrix} = \begin{matrix} \dot{\varphi}\dot{p}y - \dot{\theta}\dot{z} \\ \dot{\emptyset}\dot{p}z - \dot{\varphi}\dot{x} \\ \dot{\theta}\dot{p}x - \dot{\emptyset}\dot{y} \end{matrix} + \begin{matrix} -g\sin\theta \\ g\cos\theta\sin\emptyset \\ g\cos\theta\cos\emptyset \end{matrix} + \begin{matrix} 0 \\ 0 \\ \frac{-fz}{M} \end{matrix} \quad (1)$$

$$\begin{matrix} \ddot{\emptyset} \\ \ddot{\theta} \\ \ddot{\varphi} \end{matrix} = \begin{matrix} \frac{jy-jz}{jz}\dot{\theta}\dot{\varphi} \\ \frac{jz-jx}{jy}\dot{\emptyset}\dot{\varphi} \\ \frac{jx-jy}{jz}\dot{\emptyset}\dot{\theta} \end{matrix} + \begin{matrix} \frac{1}{jx}\tau_{\emptyset} \\ \frac{1}{jy}\tau_{\theta} \\ \frac{1}{jz}\tau_{\varphi} \end{matrix} \quad (2)$$

Shown above are the complete equations of motion of the quadrotor [14]. The first equation is a matrix mathematically describing the translational movements of the quadrotor in three dimensions. The symbols px, py, and pz represent the quadrotor's displacement in the x-axis, y-axis, and z-axis respectively. The symbols $\emptyset, \theta$, and φ represent the quadrotor's current roll, pitch, and yaw angles respectively. It can be seen that the first matrix on the right hand side of the equation is derived from the Newton-Euler equations for a rigid body. The second matrix on the right hand side of the equation represent the forces that the quadrotor experiences due to gravity g. The last matrix on the right-hand side of the first equation represent the total thrust force, fz, which the propellers exert on the quadrotor. It must be noted that translational components px, py, and pz shown in the equations are taken with respect to the xyz axis that is always in line with the cross of the quadrotor. This allows the thrust force from the propellers to always act parallel to the z-axis regardless of the roll or pitch angles.

The second equation describes the angular acceleration of the quadrotor in the roll, pitch, and yaw angles as a function of the quadrotor's torque around its x, y, and z-axes, and its moment of inertia around the x, y, and z-axes represented by jx, jy, and jz respectively.

We can observe that the model we have two systems of second order non-linear differential equations. However, if we examine more closely, we can see that in the second equation, the time-dependent functions$\emptyset, \theta$, and φ do not actually appear in the equations. Only the first and second derivatives of these functions appear. This is intuitive in the sense that the time rate of change of the roll, pitch, and yaw angles do not depend on their current values. All that matters are the torques that affect the angular accelerations and the moments of inertia that resist these rotations. This fact allows us to reduce the second equation to a system of first-order non-linear differential equations.

We can observe that the model we have two systems of second order non-linear differential equations. However, if we examine more closely, we can see that in the second equation, the time-dependent functions$\emptyset, \theta$, and φ do not actually appear in the equations. Only the first and second derivatives of these functions appear. This is intuitive in the sense that the time rate of change of the roll, pitch, and yaw angles do not depend on their current values. All that matters are the torques that affect the angular accelerations and the moments of inertia that resist these rotations. This fact allows us to reduce the second equation to a system of first-order non-linear differential equations.

Equation 1 almost shows the same case as equation 2. Only the second derivative of the functions px, py, and pz appear in the system. However, there is the existence of the functions$\emptyset, \theta$, and φ in the system that must be accounted for.

This is easily resolved by solving for these functions using the second system, and simply plugging their values into the first system. Finally, just as in the second system, we can reduce equation 1 to a system of first order non-linear differential equations.

While the equations derived are mathematically correct, they are highly simplified models for the dynamics of a quadrotor. One obvious factor absent in the equations is the effect of frictional and drag forces that a quadrotor experiences while moving and rotating in air. This is a rather glaring omission because it is unnatural to assume that there is no air present, when the thrust force itself is due to the propellers of the quadrotor pushing down on the air below it as they spin. Therefore, the effect of air resistance is added in the simulation by adding a motion-opposing force that is proportional and opposite to both the linear and angular velocities of the quadrotor. This allows a more accurate simulation. Also, wind effects are added to the simulation by adding an external time-varying force to the quadrotor.

In addition to air resistance, there are other factors to account for, such as varying wind speeds, and blade flapping, or the deformation of the propeller blades due to the flexibility of the materials used, that contribute to the highly nonlinear dynamics of a quadrotor. It is clear that it would be overly complex to model all of these nonlinear effects and achieve an extremely accurate model of a quadrotor. The model presented in this paper, however, is accurate enough to provide a test bed for simulating a quadrotor running on a fuzzy logic controller.

IV. Fuzzy Logic Components

The fuzzy algorithm has three steps: the fuzzification of the crisp sensor inputs, the application of the fuzzy rules to form an inference, and the defuzzification needed to get the crisp outputs.

A. Fuzzification

Since there are a total of twelve inputs to convert, there are also twelve graphs that map each input to the appropriate fuzzy set. However, since the twelve inputs are mostly similar, the fuzzification process for each input will also be similar. For example, since the x-direction velocity and y-direction velocity are exactly the same except for the direction, they share similar fuzzy sets.

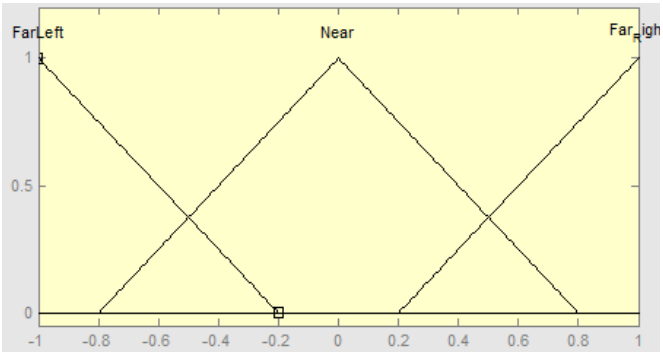

Figure 3. X-direction fuzzy sets

Figure 3 shows the graph mapping the fuzzy sets used for the variable x-direction. This variable states how far the x-coordinate of the current position of the quadrotor is from the x-coordinate of the destination. The fuzzy sets defined for this variable are far to the left, near, and far to the right. For the purposes of this paper, the left direction is in the direction of the negative x-axis while the right direction is in the direction of the positive x-axis. The fuzzy sets used are simple triangular fuzzy sets.

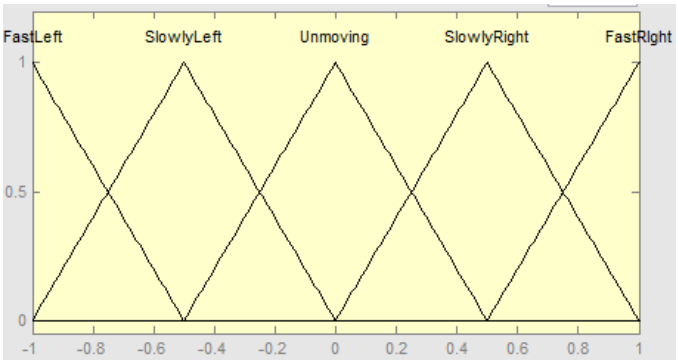

Figure 4. X-direction velocity fuzzy sets

Figure 4 shows the graph mapping the fuzzy sets for the x-direction velocity. This variable states how fast or how slow the quadrotor is moving towards or away from the target location. The fuzzy sets are fast left, slowly left, unmoving, slowly right, and fast right. As with the x-direction sets, the fuzzy sets used here are simple triangular fuzzy sets.

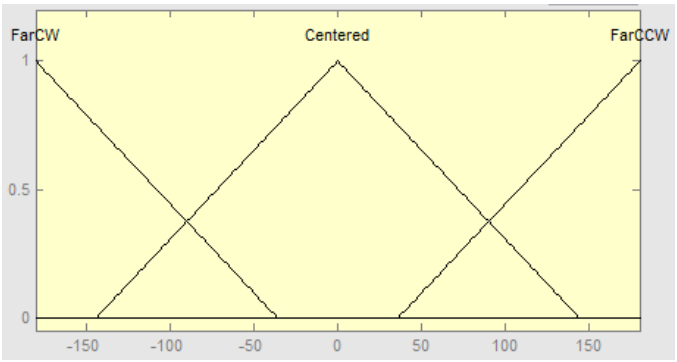

Figure 5. Pitch Angle fuzzy sets

Figure 5 shows the graph mapping the fuzzy sets for the input pitch angle. The fuzzy sets defined for this variable are far counter-clockwise, centered, and far clockwise. The fuzzy sets used here are simple triangular fuzzy sets.

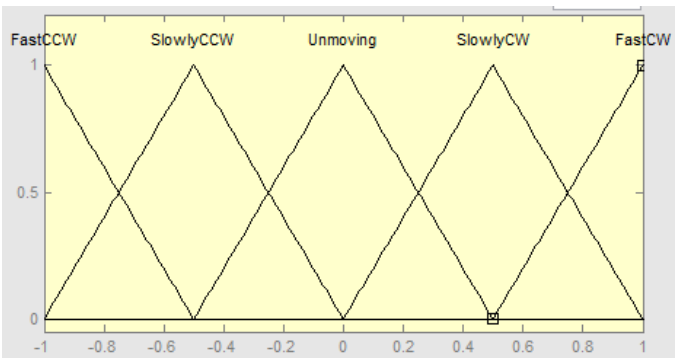

Figure 6. Pitch Angular Velocity fuzzy sets

Figure 6 shows the graph mapping the fuzzy sets for the rate of change of the input pitch angle. The fuzzy sets defined

are fast counter-clockwise, slowly counter-clockwise, unmoving, slowly clockwise, and fast clockwise. Again, the fuzzy sets use triangular fuzzy sets.

It should be clear from this that the fuzzification of the other eight inputs to the system are similar to the fuzzification of the four inputs shown above.

B. Defuzzification

After applying the fuzzy rules that were set to the fuzzified input variables, the results must be defuzzified in order to get a crisp output for the quadrotor's actuators. All in all, there are four outputs corresponding to the direction of spin and amount of thrust that each actuator must apply in order for the quadrotor to complete a stabilized point-to-point movement. In order to defuzzify the output, the centroid method was used.

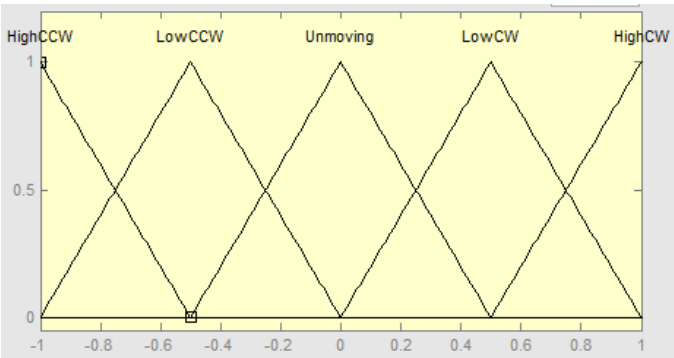

Figure 7. Output Fuzzy Sets

Figure 7 shows the graph that maps the fuzzy output from the fuzzy rule set into an exact value for the thrust applied by each motor. There are five fuzzy sets used for defuzzification: high speed counter-clockwise, low-speed counter-clockwise, unmoving, low-speed clockwise, and high speed clockwise.

V. Experiment Results

The parameters relating to the fuzzy logic algorithm such as centroid tracking accuracy and program run time, are listed here and tested in order to assess the performance and efficiency of the algorithm in solving the problem.

A. Effect of Wind Speed

To quantify the point-to-point movement performance, trials were conducted to test the average error of the algorithm. One trial consisted of one hundred separate runs of the algorithm for a random starting and stopping point in three-dimensional space and the error for each run was calculated as the difference between the actual distance travelled by the center of mass of the quadrotor and the linear distance between the starting and stopping points. The distance between the starting and stopping points for all trials was kept constant. More trials were then conducted for differing wind speeds.

Table 1. Varying Wind Speed Test

Wind Force (Newtons)	Average Error of 100 Runs of the Fuzzy logic (Meters)
10	0.026173
20	0.041306
30	0.055336
40	0.054169
50	0.062468
60	0.067190
70	0.082854
80	0.078457
90	0.074995
100	0.080620

Table 1 shows the results of the trials conducted to test the accuracy of point-to-point movement of the quadrotor with fuzzy logic controller. It can be seen that for a higher wind speed, the average error grows bigger as expected. However, the average error for a wind speed applying 100 Newtons still remains well below 0.1 meters.

B. Effect of Sensor Noise

The effect that sensor noise has on the algorithm was also tested. Holding the wind force constant at 10 Newtons, the algorithm was tested once again for 100 times for each percentage of sensor noise.

Table 2. Sensor Noise Test

Sensor Noise (Percentage)	Average Error of 100 Runs of the Fuzzy logic (Meters)
1	0.02233
2	0.05534
3	0.0834
4	0.09002
5	0.10424
6	0.1412
7	0.2344
8	0.4432
9	0.5512
10	0.6645

Table 2 shows the result of the experiment testing the algorithm's ability to tolerate sensor noise and error.

. It can be seen that the error of the quadrotor's flight increases as the sensor error becomes higher: another expected result. However, it can also be seen that for a sensor error of ten percent, the error in the flight of the quadrotor remains below 1 meter.

C. Graphs of Results

This paper will be incomplete without at least a few examples of the simulation results plotted and visually represented.

The following figure shows the simulated quadrotor animated with OpenGL.

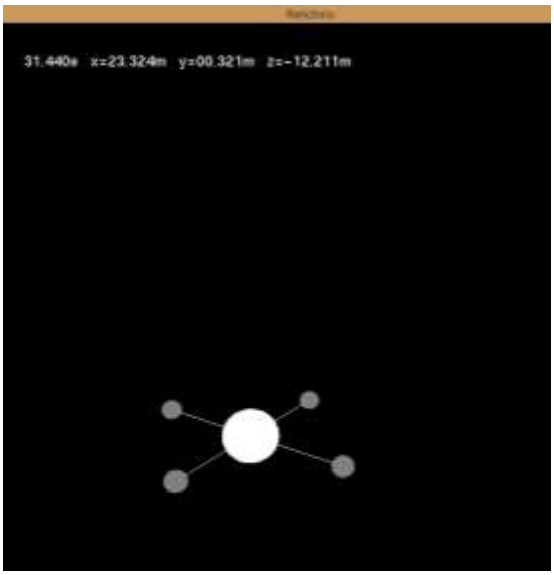

Figure 8. Simulation Example

Figure 8 shows a screenshot of the quadrotor simulation animation. Various data are shown, such as the time the quadrotor has been running and its current x, y, and z-coordinates.

VI. Conclusion

This paper presented the multiple-input multiple-output fuzzy logic algorithm for stabilized point-to-point movement of a quadrotor unmanned aerial vehicle. It was seen that the algorithm was able to successfully move a simulated quadrotor from one point to another with a high degree of efficiency in terms of distance travelled.

The wind error test showed that even for winds applying a 100 Newton force on the quadrotor, the average error remained well below 0.1 meters. The sensor noise test, on the other hand, was tested to have an error less than 1 meter even at 10 percent sensor noise. These results show how well the algorithm adapts to less-than-ideal conditions, as it will have to in order to perform adequately in real life.

For future works, this algorithm may be used in conjunction with quadrotor swarming algorithms as a method of providing a reliable means of flight for each quadrotor in the swarm. Also, research may be done on efficient ways to apply the algorithm on a low-powered and cheap processor for use in actual flights.

References

[1] G. Beni, "From Swarm Intelligence to Swarm Robotics," in *Swarm Robotics*, Springer, 2004, pp. 1-9.

[2] T. Balch, "Communication, Diversity and Learning: Cornerstones of Swarm Behavior," in *Swarm Robotics*, Springer, 2004, pp. 21-30.

[3] L. Wang, H. Shi, T. Chu, W. Zhang and L. Zhang, "Aggregation of Foraging Swarms," in *AI 2004: Advances in Artificial intelligence*, Springer, 2005, pp. 766-777.

[4] T. Schmickl, C. Moslinger and K. Crailsheim, "Collective Perception in a Robot Swarm," in *Swarm Robotics*, Springer, 2007, pp. 144-157.

[5] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*, Springer, 2004, pp. 10-19.

[6] A. A. Bandala, R. R. P. Vicerra and E. P. Dadios, "Swarming Algorithm for Unmanned Aerial Vehicle Quadrotors," *Journal of Advanced Computational Intelligence and Informatics*, vol. 8, no. 5, 2014.

[7] V. Gazi and K. Passino, "A Class of Attraction/Repulsion Functions for Stable Swarm Aggregations," *IEEE Conference on Decision and Control*, 2002.

[8] V. Gazi, "Swarm Aggregations Using Artificial Potentials and Sliding Mode Control," *IEEE Conference on Decision and Control*, 2003.

[9] R. R. Nair and L. Behera, "Swarm Aggregation using Artificial Potential Field and Fuzzy Sliding Mode," *American Control Conference*, 2012.

[10] A. Gasparri, G. Oriolo, A. Priolo and G. Ulivi, "A Swarm Aggregation Algorithm based on Local Interaction for Multi-Robot Systems with Actuator Saturations," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[11] A. Gasparri, A. Priolo and G. Ulivi, "A swarm aggregation algorithm for multi-robot systems based on local interaction," *IEEE International Conference on Control Applications*, 2012.

[12] A. Leccese, A. Gasparri, A. Priolo, G. Oriolo and G. Ulivi, "A Swarm Aggregation Algorithm based on Local Interaction with Actuator Saturations and Integrated Obstacle Avoidance," *IEEE International Conference on Robotics and Automation*, 2013.

[13] M. A. Ferreira and H. K. Lee, Multiscale Modeling: A Bayesian Perspective, Springer, 2007.

[14] M. Cepin, Assessment of Power System Reliability: Methods and Applications, Springer, 2011.