THE JOURNAL OF COMPUTER SCIENCE AND ITS APPLICATIONS
Vol. 24, No 1 June 2017

NETQUADCOPTER AUTONOMOUS FLIGHT CONTROL USING FLOOD FILL ALGORITHM

N.V. Blamah[1], E.O. Okorie[2] and B.Y. Baha[3]
[1,2]Department of Computer Science,
University of Jos, Jos, Nigeria

[3]Department of Information Technology,
Mobdibbo Adama University of Technology, Yola

[1]blamahn@yahoo.com, [2]nickson1277@gmail.com[3]bybaha@yahoo.com

ABSTRACT
Netquacopter Autonomous Flight control using flood fill/dynamic programming (Netquadcopter) is centered on the use of Netquadcopter to survey areas which normally would be difficult and dangerous for human beings to ply. It also aids communication between the control station and a base station severing as a medium of communication in areas where there is limited telecommunication network. It used a combination of PID algorithm for flight control, flood fill algorithm for autonomous flight, with X configuration and a new algorithms for obstacle avoidance using four ultra-sonic sensors. It consists also of wireless communication system that transmits sensor values from the Netquadcopter to a computer (control station), showing in real time activities and state of the Netquadcopter. This work tends to use flood fill algorithm to implement autonomous flight, it was designed and constructed by the authors but the path of travel was stimulated due to the unavailability of real environment for travel.

Keywords –Quadcopter, Netquadcopter, flood fill algorithm, Dynamic programming, Autonomous flight, Maze

1.0 INTRODUCTION

A quadcopter is an aircraft lifted and propelled by four motors. It is capable of flying without a pilot or crew on-board, and it is piloted remotely via remote control (RC). Quadcopter maintains the ability to move rapidly and agilely through the air much like a plane while maintaining the helicopter's ability to hover and move at a low air speed. By equipping such a copter with distance sensors and camera, an autonomous aerial vehicle is created which can avoid nearby obstacles regardless of velocity. Information gathered by these sensors will then be used to navigate the quadcopter through spaces too tight for other styles of autonomous aerial vehicles [2].

This type of vehicle fits in the Vertical Take Off and Vertical Landing (VTOL) category as they can successfully perform vertical take offs, landings and hovering despite being heavier than air [5].

2.0 A REVIEW OF QUADCOPTERS

Over the years, research has been published relating to Quadcopter. The following are list of few of them in categories:

1. Geometric: Geometric control is concerned with the development of control systems for dynamic systems evolving on nonlinear manifolds that cannot be globally identified with Euclidean spaces[18], and research on it include Geometric Tracking Control Of A Quadcopter UAV On Se(3)[18], Geometric Control of Multiple Quadrotor UAVs Transporting a Cable Suspended Rigid Body[19], Geometric Stabilization of a Quadrotor UAV with a Payload Connected by Flexible Cable[6], Control of Complex Maneuvers for a Quadrotor UAV Using Geometric Methods on Se(3)[20].

2. Quadcopter control algorithm: To Guide a quadcopter to accomplish the planned mission, a control system needs to be designed. This consists of position control for translational motion and attitude control for rotational motion, the following research has been done under it: Flight PID controller design for a UAV quadrotor[2], Attitude Stabilization Control of a Quadrotor UAV by Using Backstepping Approach[21], Attitude Control of a Quadrotor with Optimized PIDController[8], Different Approaches of PID Control UAV Type Quadrotor[17].

3. Graphical user interface (GUI): under this category the following are few research, Simple GUI Wireless Controller of Quadcopter[4], Quadcopter (UAVs) for Border Security with GUI System[9],GUI Controller Design for QuadcopterControl[5], Assisted Teleoperation of Quadcopters Using Obstacle Avoidance[10].

4. Obstacle avoidance: This category list few different obstacle avoidance methods that has been researched on, such as Aerial collision avoidance System Final Report[3], Use of LIDAR for Obstacle Avoidance by an Autonomous Aerial Vehicle[16], Autonomous Obstacle Avoidance and Maneuvering on a Vision-Guided MAV Using On-Board Processing (MAV)[13], Real Time Obstacle Avoidance And Navigation Of A Quad-Rotor Mav Using Optical Flow Algorithms[15], Design and Implementation of a Real Time Wireless Quadcopter for Rescue Operations [7], Search-and-rescue remote sensing quadcopter[1], Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface[11], Re-design of a quadcopter capable of autonomous flight and collaboration with a UGV[12], Experimental Study of Quadcopter Acoustics and Performance at Static Thrust Conditions[14].

The Netquadcopter presented in this research work uses flood fill algorithm for obstacle avoidance using four Sonar sensors.

Sonar sensors are cheaper compared to cameras which have been used in other research for obstacle avoidance. The prevailing security challenges in Nigeria motivated this research; with autonomous flight the military can send aid to affected victims or information with the quadcopter without any human interference or control.

3.0 COMPONENT SPECIFICATIONS

Quadcopter is controlled by RC and requires the following components: a frame, motors (rotors) with propellers, electronic speed controllers (ESC), a power supply, a radio receiver and transmitter, a flight controller (Arduino mega), a transceiver, ultrasonic sensors, and Inertial measuring unit(IMU). All the measurements, dimensions and standards of the components were selected to comply with required standards.

3.1 Netquadcopter Frame

The frame used in this research was constructed locally; it is an Aluminum pipe of 2cm x 1cm dimension. The length from one end of the frame to the other is 45cm long. Two 45cm long Aluminum pipes were joined to form an

X shape, which serves as the frame of the quadcopter, and 3 plastic plates were used to form a base for the frame. These plates were responsible for carrying the flight controller board, and other components that are needed for optimal flight control. The landing gear was carved out from an aluminum bar to ensure that the quadcopter has a landing stand and for protection of the camera when the quadcopter lands.

3.2 Motors and Propellers

The motors spin the propellers to provide the quadcopter with lifting thrust. Quadcopter uses brushless DC motors, as they provide thrust-to-weight ratios superior to brushed DC motors. However, they require more complex speed controllers. These motors are typically given two ratings: KV ratings and current ratings. The KV rating indicates how fast the motor will spin (RPM) for 1V of applied voltage. The current rating indicates the max current that the motor may safely draw. For this research, we selected 2200Kv, 30A max motors.

Propellers come in many sizes and materials. They are measured by their diameter and their pitch in the format (diameter) x (pitch). The pitch is a measurement of how far a propeller will "travel" in one revolution. Prop selection is important to yield appropriate thrust while not overheating the motors. For this research, we selected 9x4.7 carbon fiber props, which yield 1.4lbs of maximum thrust while drawing 10.2A. With four motors, the maximum thrust for the Netquadcopter is approximately 5.5lbs.

3.3 Electronic Speed controller

Every motor needs an individual electronic speed controller (ESC). These speed controllers accept commands in the form of PWM (Pulse Width Modulation) signals and output the appropriate motor speed accordingly. Every ESC has a current rating, which indicates the maximum current that it may provide the motor without overheating. Appropriate ESCs must be chosen to ensure that they can provide enough current for the motors. We selected 30A china ESC for this research, as they are well reviewed for use with quadcopters and have a sufficient current rating.

3.4 Radio Receiver

The radio receiver (Rx) receives radio signals from a Radio Controller (RC) transmitter and converts them into control signals for each control channel (throttle, yaw, roll & pitch). Modern RC receivers operate on a 2.4 GHz radio frequency, while older Rx units often used 72MHzfrequencies. Rx units may have as few as 4 channels, but many have more channels for additional control options. We selected a Flysky 6 Channel Receiver for this research. The RC receiver accepts radio signals from an RC transmitter and translates it into separate channels of control. The receiver in our quadcopter is capable of outputting 6 channels of control, that is throttle, yaw, roll, pitch, and 2 auxiliary channels (controlled by toggle switches on the transmitter). RC signals are a form of specialized pulse width modulation (PWM), in which the length of the HIGH pulse contains the output information. Each HIGH pulse varies from approximately 1 ms to 2 ms, with a period of 20 ms

In order to read this signal into the Arduino mega flight controller, the Arduino mega needs to measure the length in microseconds of the HIGH pulse. The simplest way to achieve that is to use the pulse In function, which measures pulse lengths on a pin. However, this method is not suitable for a flight controller because the function blocks the rest of the program from running while it waits for a pulse.

3.5 Flight Controller

The flight controller is the "brain" of the quadcopter, and performs the necessary operations to keep the quadcopter stable and controllable. It accepts user control commands from the Rx, combines them with readings from the IMU sensor(s), and calculates the necessary motor output. For this research, we selected a purpose-made flight controller board.

These boards often have integrated attitude sensors, and provide well-tested flight control software. For this research however, we used an Arduino Mega as the flight controller, as we have programmed the flight control software ourselves and also customize it for other needs. We Chose the Arduino Mega as the flight controller for our quadcopter, because of its large number of I/O pins - 54 in total – which enabled us to have connection to all the components we used in this research.

3.6 Dynamic System Control

Quadcopters are generally unstable. Anyone attempting to control a quadcopter with manual inputs only and no IMU sensor would quickly find out that balancing the aircraft is very nearly not possible. So, to make the quadcopter a stable system, it is important to integrate an attitude sensor and a set of dynamic system controllers. For this research, we chose PID (Proportional Integral Derivative) controllers due to its relative simplicity and good performance.

3.7 IMU 10 DOF

GY88 is the IMU (Inertial Measuring Unit) used in this work. GY88 is a 10 DOF (degree of freedom) IMU sensor consisting of a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer and 1 axis barometer which are MPU650, HMC5883L, BMP085 respectively. All these sensors communicate with the Arduino Mega using I2C serial communication. When configured correctly, they provide reliable inertial measurements.

The IMU provides the flight controller with the readings of the quadcopter's orientation in space. The minimum requirement is a gyroscope, but most quadcopters also incorporate an accelerometer. For a self-stabilizing of the Netquadcopter, only gyroscope was used.

The schematic diagram in Fig. 1 presents the components configured for the Netquadcopter proposed in this research.

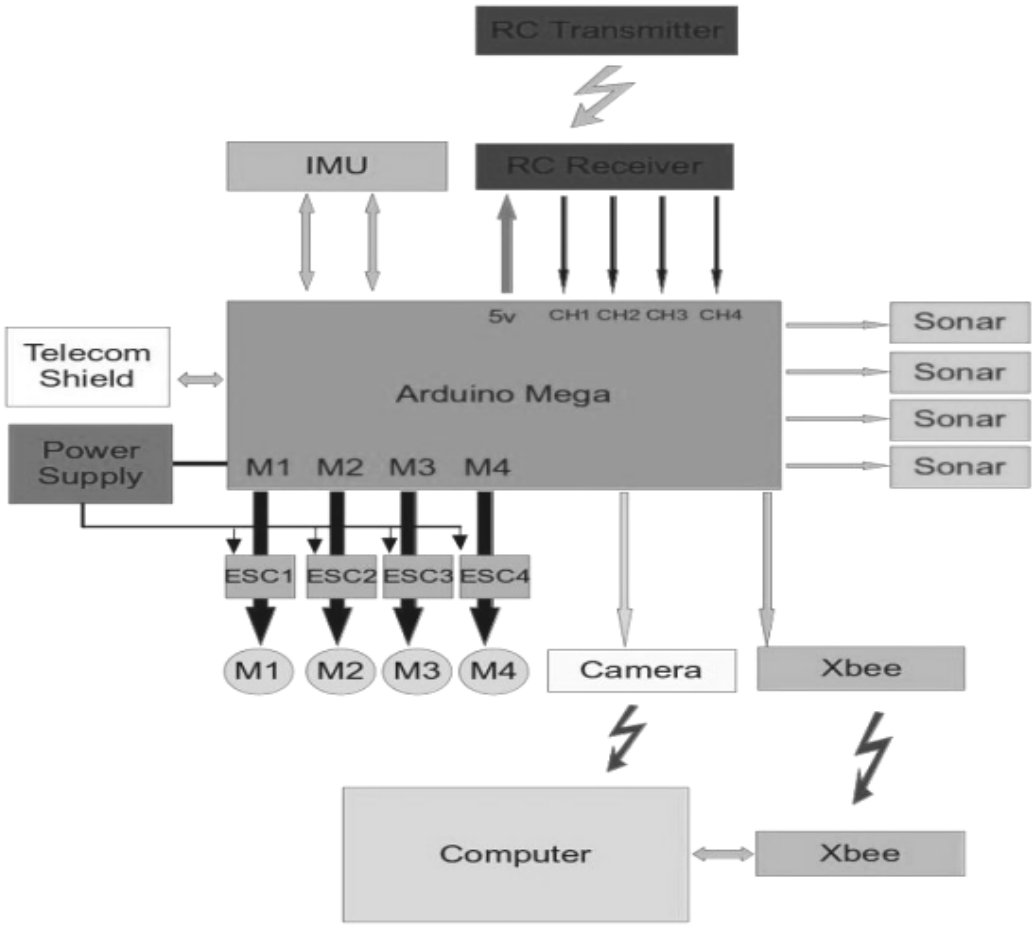


Fig.1.Netquadcopter Schematic diagram (FromHybrid Communication Based Surveillance Quadcopter UAV 2017)

4.0OBSTACLE AVOIDANCE

Ultrasonic sensors are sensors that convert ultrasound waves to electrical signals. These devices work on a principle similar to that of transducers used in radar and sonar systems, which evaluate attributes of a target by interpreting the echoes from radio or sound waves.

Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object.

The HC-SR04 ultrasonic sensor was used in this paper for obstacle avoidance. The ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400cm or 1 inch to 13 feet, its operation is not affected by sunlight or black materials like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes with ultrasonic transmitter and receiver module

For the Netquadcopter to avoid obstacles, four HC-SR04 ultrasonic sensors were used. Each of these sensors was placed in-between the Netquadcopter frame. An algorithm was developed to handle obstacle avoidance and these targets 16 possibilities which are listed below:

Table.1. Sensor Detection and Netquadcopter Decisions to avoid obstacle

State Number	Sensor Detectors	Movement Decision of Netquadcopter
0	None	Continue flight
1	Front	Left or Right or Backward
2	Right	Left or forward or Backward
3	Back	Right or Left or Right
4	Left	Right or Forward or Backward
5	Front and Right	Left or Backward
6	Front and Back	Left or Right
7	Front and Left	Right or Backward
8	Right and Back	Forward or Left
9	Right and Left	Forward or Backward
10	Back and Left	Forward or Right
11	Front, Left, and Back	Right
12	Front , Right, and Back	Left
13	Front, Right , and Left	Backwards
14	Left, Right, and Back	Forward
15	Front, Left, Back, and Right	Move Up (yaw)

Once the Netquadcopter senses an obstacle within 35cm range, one of the above movement decisions is triggered and the Netquadcopter is prompted to take action as to how to move away from the obstacle to avoid colliding with it.

Algorithm 1: Obstacle avoidance formal algorithm

```
1QuadSensors<FrtSenTrig,FrtSenEc,RgtSenTrig,RgtSenEc,BckSensTrig,BckSenEc,
LftSensTrigr,LftSenEc> sensors;
2       threshold = 35;
3       event = 0;
4       sensors.sense();
5       if(sensors.FrontDistance<=threshold )event += 1;
6       if(sensors.RightDistance<= threshold) event += 2;
7       if(sensors.BackDistance<= threshold) event += 4;
8       if(sensors.LeftDistance<= threshold) event += 8;
9       return event;
10      avoid(byte event){
11      Coditions(event)
12      {
13      State 1:Move Left or Right or Backward; Stop;
14      State 2:Move Left or forward or backward; Stop;
15      State 3:Move Right or Left or Right; Stop;
16      State 4:Move Right or Forward or Backward; Stop;
17      State 5:Move Left or Backward; Stop;
18      State 6: Move Left or Right; Stop;
19      State7:Move Right or Backward; Stop;
20      State 8:Move Forward or Left; Stop;
21      State 9:Move Forward or Backward; Stop;
22      State 10: Move Forward or Right; Stop;
```

```
23      State 11:Move Right; Stop;
24      State 12:MoveLeft;Stop;
25      State 13:MoveBackwards;Stop;
26      State 14:Move Forward; Stop;
27      State 15:Move Up(yaw);Stop;
28      default:
29      Continue flight;
30       }
31         }
```

Algorithm 1 is for obstacle avoidance, line 1 initialize the sensor trigger and echo pins, while thresholds for the four direction is set using the FrontDistance, RightDistance, BackDistance, LeftDistance. State are conditions of the sensors, followed by the respective actions performed, and a stop after each action is executed.

5.0PATH FINDING

Dynamic programming/flood fill method can be used for path planning. It has a number of advantages as well as disadvantages. In this paper, Dynamic programming was used to achieve autonomous flight of the Netquadcopter; given a map and one or more goal position(s), it will output the best path from any possible starting location. This planning technique is not just limited to one starting location, but from any starting location. Besides A*, dynamic programming is an alternative method for path planning. Much like A * it will find the shortest path.

In this paper we considered 6X6 grid map in which the Netquadcopter is expected to fly to and from, and avoiding obstacles along its way. This shows the stochastic nature of the world and that there is need to plan not only for the most likely position, but for other positions as well. Dynamic programming is helpful because it does give a solution for every location. Dynamic programming will give us an optimum action (called the policy) to perform for every navigable grid cell. The policy is defined as the label for each cell; it's a function that maps the grid cell into actions, and action in this case is the direction of the arrow from the start(S) to the goal (G) on the gird. The function is mathematically represented as shown below, where X represents the X-axis and the Y-axis

Policy X,Y → Action

For the Netquadcopter to follow this plan (to fly from the starting point to the goal point) we will use the grid or what is popularly known as Maze, as shown Fig2.

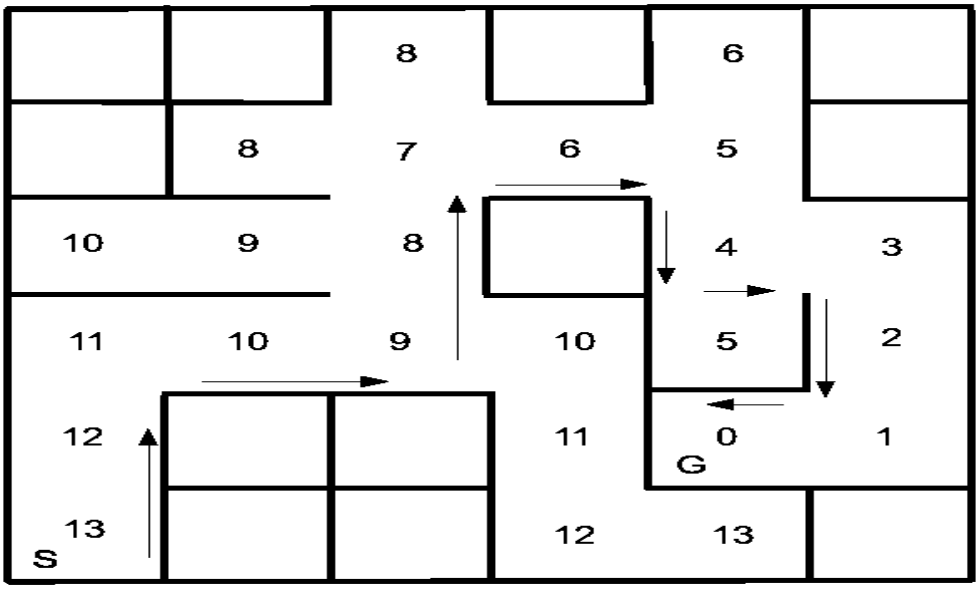


Fig. 2. Traveling Path Grid for Netquadcoter travel

The 6X6 grid or map further showcases the stochastic nature of the world. Using the grid, the value function $f(x,y)$ associated to each cell is the length of the shortest path to the goal. With this, we can recursively calculate the value for each cell by summing the optimum neighbor value and a movement cost of one. The value function associated to each grid cell and the length to the shortest path to the goal is $f(X, Y)$, which is recursively calculated by taking the optimal neighbor $\min f(X', Y')$, and considering its value and the cost it takes for instance time. Let's say 1, thus we have:

Value function to the nearest cell as, $f(X, Y) = \min f(X', Y') + 1$

Where X', Y' are optimal neighbors recursively calculated.

From the Grid, S stands for starting point of the Netquadcopter and G for the goal. The arrows show the direction of travel. The ultimate goal is to allow the Netquadcopter to figure the path to the target cell G from the starting cell S. In order to do this, we used dynamic programming to allow the Netquadcopter to make intelligent decisions.

Fig.2. shows how this is implemented; the idea is to put values into each cell depending on how far it is from the target cell (G). We begin with Goal point G (target cell) which is marked 0, then all neighboring cells that are not separated by a wall are given higher value(s),

The neighboring cell of G that is not separated by a wall is given the value 1s
The neighboring cell containing 1 that is not separated by wall is given 2s
The neighboring cell containing 2 that is not separated by wall is given 3s
The neighboring cell containing 3 that is not separated by wall is given 4s
The neighboring cell containing 4 that is not separated by wall is given 5s
The neighboring cell containing 5 that is not separated by wall is given 6s
The neighboring cell containing 6 that is not separated by wall is given 7s
The neighboring cell containing 7 that is not separated by wall is given 8s
The neighboring cell containing 8s that is not separated by wall is given 9s
The neighboring cell containing 9s that is not separated by wall is given 10s

The neighboring cell containing 10s that is not separated by wall is given 11s
The neighboring cell containing 11s that is not separated by wall is given 12s
The neighboring cell containing 12s that is not separated by wall is given 13s.

The direction of travel is indicated by the arrow. This is not based on assumption; rather, it follows the dynamic programming algorithm. The Netquadcopter starts at Cell S and moves forward; the direction it takes depends on the value placed at each cell. In other words, it follows the path or cell with a number less than its present position. In our case, the starting cell is 13, it moves to cell 12, then proceeds to cell 11, makes a turn and move to cell 10, then proceeds to cell 9 makes a turn and proceed to cell 8, then to 7, 6, 5, 4, 3, 2, 1 and finally to cell G (0) which is the goal or target cell. In all, we can say it took the Netquadcopter 13 steps to arrive at its target cell.

For the Netquadcopter to return to its starting point we developed another algorithm, still using the dynamic programming approach but this time, the Goal or target cell is the original position where it first took off from while the starting point is its current position. We will use Fig.3 to demonstrate this

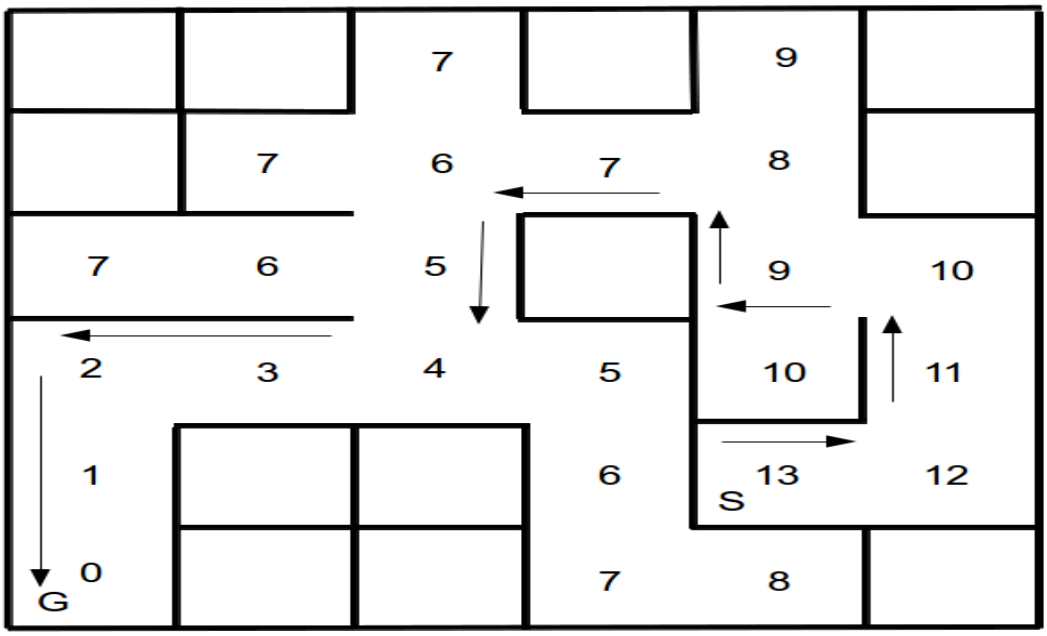


Fig. 3. Traveling Path Grid for Netquadcoter return

Just as explained above, the Netquadcopter follows the cell with values less than its present position or cell till it gets to its final destination that is from S cell with number 13 till it gets to the G cell with value 0. Implementing this algorithm in our grid, we did not encounter a tie, that is a situation where by the NetQuadcopter meet two cells that are of the same value and is left to make decisions as to which path to follow.

Fig.4is an example of tie situation, When the Netquadcopter is on the cell with values 4, its neighboring cells are 5 at the left, 3 at the right, and 3 at the front, the Netquadcopter can't obviously take left because 5 is higher than 4. So it is left with the decision to follow any of the cells with value 3. Herein lies the problems; which direction will the Netquadcopter take? Based on our algorithm the Netquacopter will head in the forward direction and will not consider taking the right direction; the

reason being that it is less expensive to move forward than to turn right and also it's easier for the Netquadcopter to move forward than waiting to turn to the right before proceeding with its navigation.

One may ask: why can't we just roll right, that is, move to the right with the Netquadcopter's front facing forward? In this case, we will end up having the camera not capturing what is at the front of the Netquadcopter as it navigates, rather, it will stream videos containing walls (obstacles).

This obviously cannot be implemented with a robotic car because it will require a total of 3 turns assuming it is going to follow the right direction but only one 1 if it moves forward, therefore, costing more and taking more time if it takes the right direction instead of the forward direction.

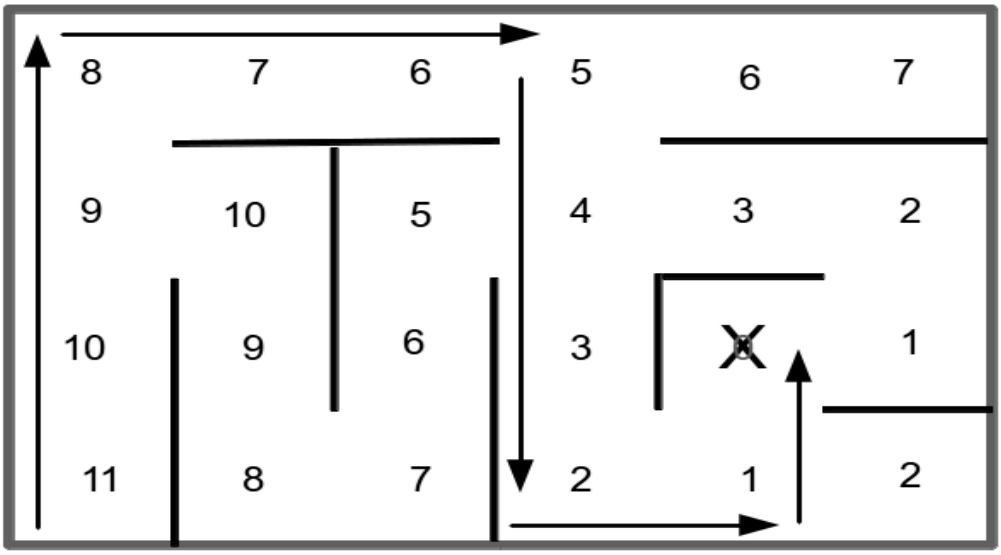


Fig.4. Path planning with a tie

6.0 IMPLEMENTATION AND RESULTS

The implementation of Netquadcopter was achieved using two programming languages, namely C/C++ and python. C/C++ was used to develop the source code that runs on the Arduino Mega (Netquadcopter) while python programming language was used to develop the GUI. PyQt python desktop development framework was used because of its robustness, availability of libraries as well large support and availability of resources online.

6.1 Obstacle avoidance

There are four directional sensors mounted on each side of the Netquadcopter used for detecting obstacles on each sides of the Netquadcopter, the sensors configuration provided sufficient coverage up to60% success rate for obstacles detection. The distance to the obstacle is calculated using the following formula supplied with the ultrasonic sensors:

Distance (cm) = pulseIntValue sensor Echo/58

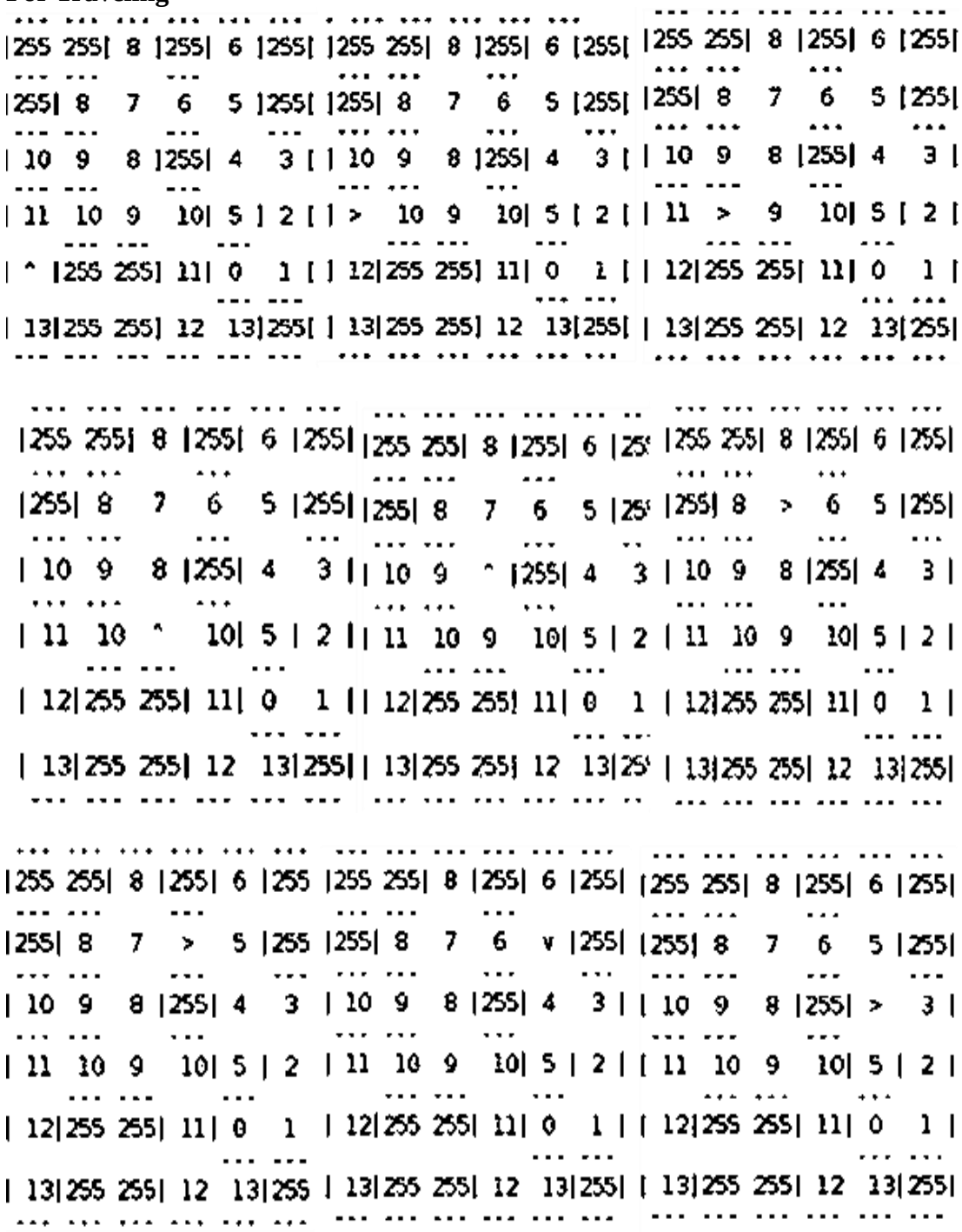
If an obstacle is detected and if the distance is within the 35cm range, then pre-determined roll, pitch and yaw command will be sent to the flight controller to maneuver away from the obstacle.

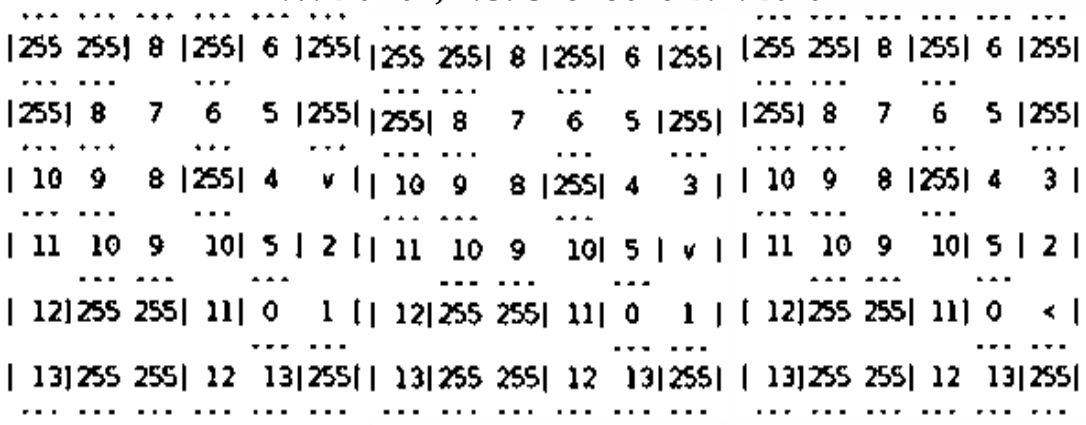
As explained in the methodology, the algorithm developed helps the Netquadcopter to avoid objects on its way by taking actions based on the possibilities stated in Table.1.

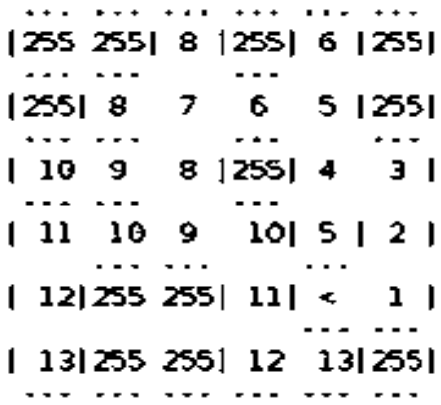
6.2 Dynamic programming/Flood Fill implementation

For this work, since we don't have a physical plan or Grid to fly the Netquadcopter, we used Arduino Serial monitor to demonstrate how the NetQuadcopter navigates in the Grid, this will be for both traveling and returning of the Netquadcopter.

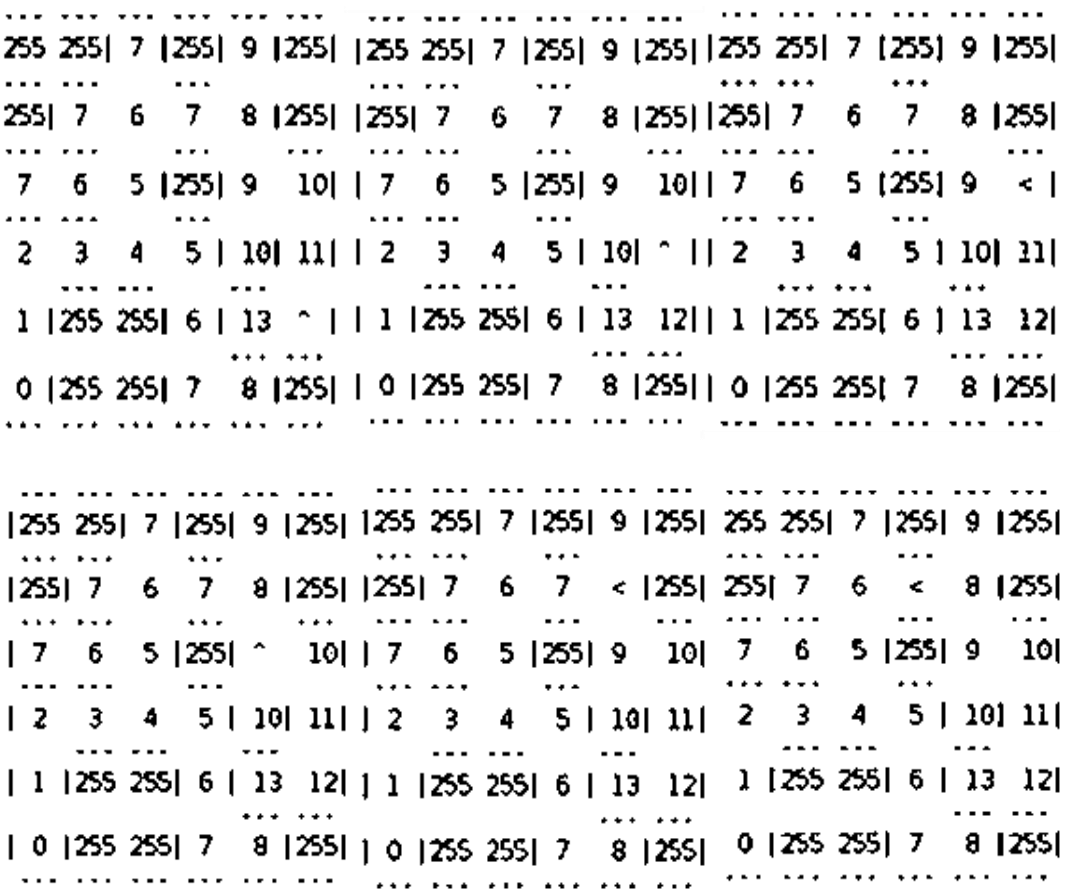
For Traveling

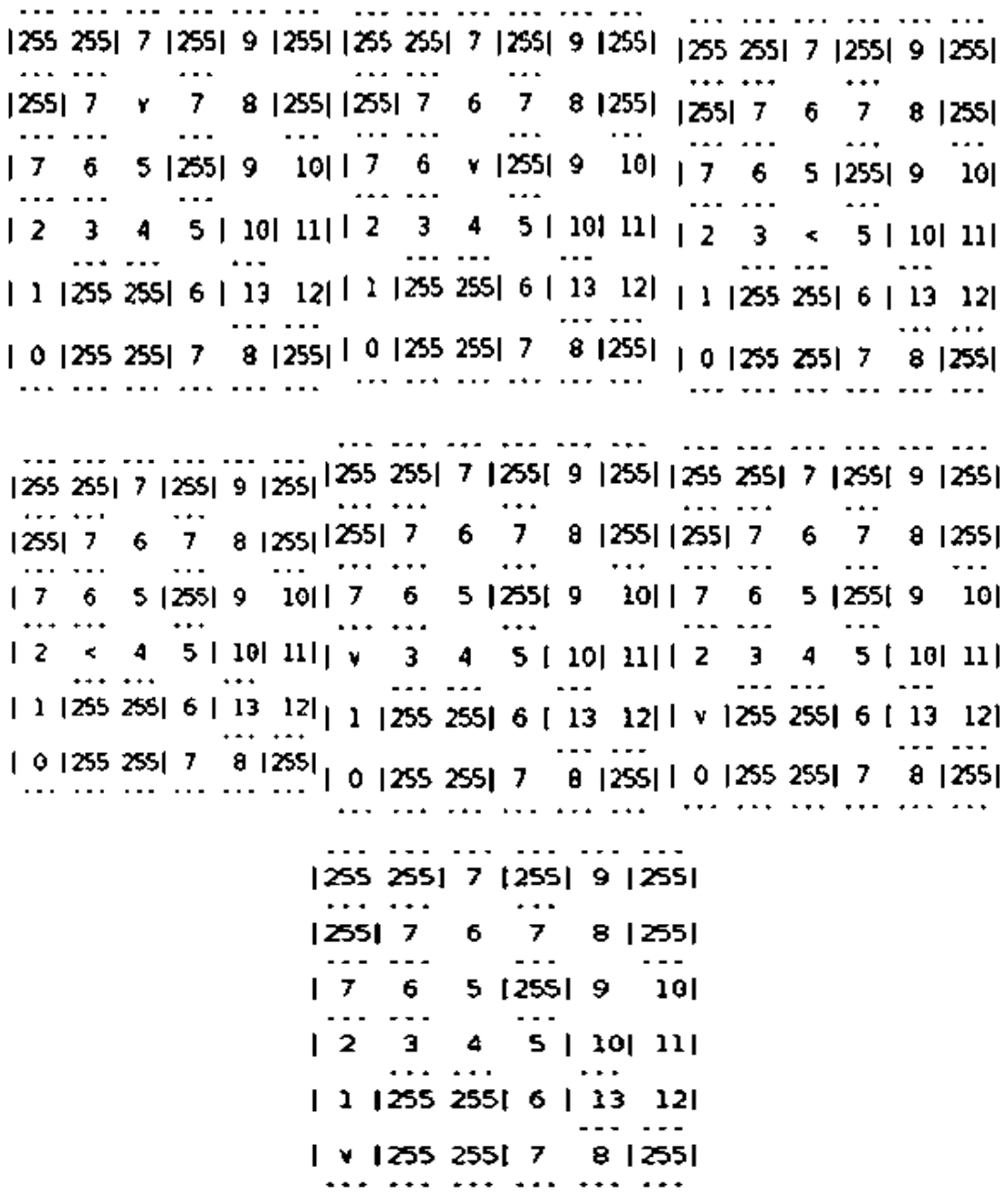






Now for Returning





For all the images the >, <, ^, V shows the direction of travel, moving from one cell to another following the cell with values less than the value of its present cell. The simulation results for both travel and return show the paths threaded by the Netquadcopter; with all obstacles successfully avoided.

6.3 Sending IMU values to the Control station

The IMU sensor gives out the accelerometer, gyroscopes, Barometer and magnetometer values, these values are sent via I2C to the Arduino Mega and this is serially transmitted to the control station wirelessly by the Xbee Shield attached on the Netquadcopter.

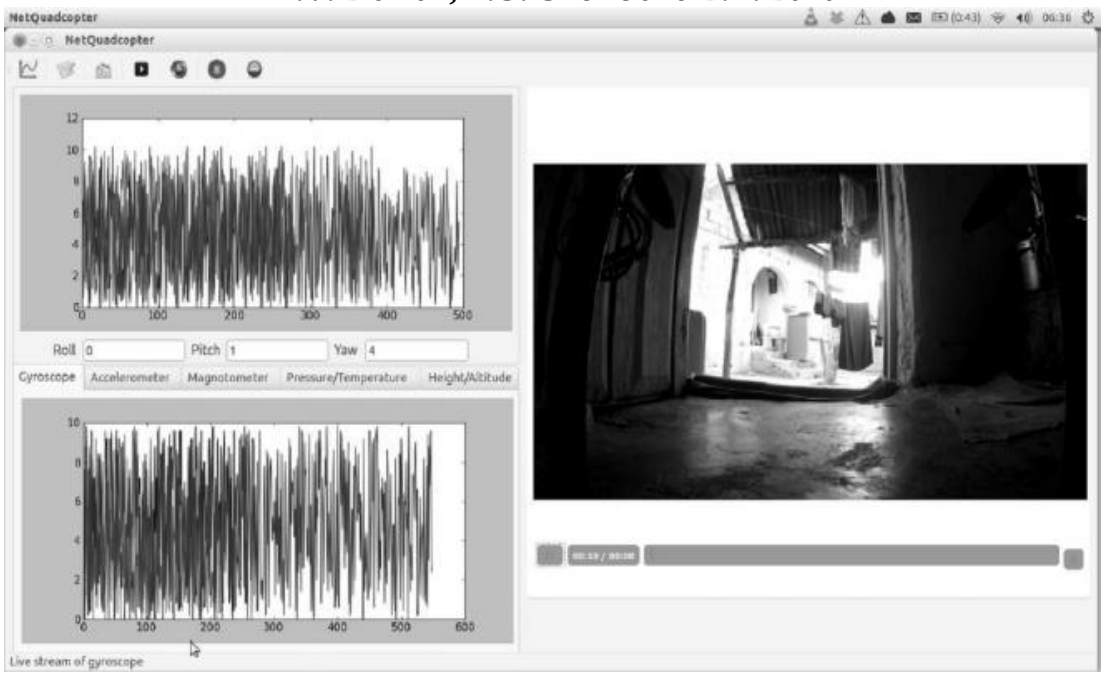


Fig. 5.NetquadcopterGUI with Live Stream of IMU values and Video

7.0 CONCLUSION

Netquadcopter can be used for several purposes, and a lot of scientific research can still be done in this field despite the ongoing research. There are several ways to achieve autonomous flight but most of them are expensive to implement and also restricted for indoor use only. An autonomous Quadcopter can be used for several purposes such as Medical delivery, fumigation of a farmland without needed human control. In this research we used flood fill algorithm to implement autonomous flight, it was designed and constructed by the authors but the path of travel was stimulated due to the unavailability of real environment for travel. It use include delivery of packages without human interference and also delivery of support aid materials to victims of insurgence. Future work include using a better sensor, reducing the size of the quadcopter, deep learning for autonomous flight implementation

8.0REFERENCES

[1] Abdallah, Z.A., Saifallah, Q. (2010). Search-and-rescue remote sensing quadcopter, 10-12.

[2] Atheer, L.S., M M. Haider, A.F.M., & Khalaf, S.G. (2010). Flight PID controller design for a UAV quadrotor, 1.

[3] Bradley, B., Austin, W., & Nelson, G. (2012). Aerial collision avoidance system final report, 2,4.

[4] Dirman, H., Mongkhun, M.Q., Rozaimi, G., Mohd, N.M.T., Wahyu, M.U., Rosil, O. (2012). Simple GUI Wireless Controller of Quadcopter, 1.

[5] Dirman, H., & Mongkhun, Q. (2013). GUI Controller Design For Quadcopter Control, 1.

[6] Farhad, A.G., Daewon, L., Taeyoung, L. (2014).Geometric Stabilization of a QuadrotorUAV with a Payload Connected by Flexible Cable, 1.

[7] Gordon, O., Arinze, O., James, O., Nnaemeka O. (2016). Design and Implementation of a Real Time Wireless Quadcopter for Rescue Operations, 1.

[8] Hossein, B., Mohammad, R., Reza, M., Hossein, N., Seed, M.S. (2012). Attitude Control of a Quadrotor with Optimized PID Controller.

[9] Jinay, S.G., & Rajaram, D.P. (2013). Quadcopter (UAVS) For Border Security With GUI System,

[10] Joao, M., & Rodrigo, V. (2012). Assisted teleoperation of quadcopter using obstacle avoidance,

[1] Abdallah, Z.A., Saifallah, Q. (2010). Search-and-rescue remote sensing quadcopter, 10-12.

[2] Atheer, L.S., M M.Haider, A.F.M., &Khalaf, S.G. (2010). Flight PID controller design for aUAV quadrotor, 1.

[3] Bradley, B., Austin, W., & Nelson, G. (2012). Aerial collision avoidance system final report, 2,4.

[4] Dirman, H., Mongkhun, M.Q., Rozaimi, G., Mohd, N.M.T., Wahyu, M.U., Rosil, O. (2012).Simple GUI Wireless Controller of Quadcopter, 1.

[5] Dirman, H., &Mongkhun, Q. (2013). GUI Controller Design For Quadcopter Control, 1.

[6] Farhad, A.G., Daewon, L., Taeyoung, L. (2014). Geometric Stabilization of a QuadrotorUAV with a Payload Connected by Flexible Cable, 1.

[7] Gordon, O., Arinze, O., James, O., Nnaemeka O. (2016). Design and Implementation of a Real Time Wireless Quadcopter for Rescue Operations, 1.

[8] Hossein, B., Mohammad, R., Reza, M., Hossein, N., Seed, M.S. (2012). Attitude Control of a Quadrotor with Optimized PID Controller.

[9] Jinay, S.G., & Rajaram, D.P. (2013). Quadcopter (UAVS) For Border Security With GUI System,

[10] Joao, M., & Rodrigo, V. (2012). Assisted teleoperation of quadcopter using obstacle avoidance,

[11] Karl, F., Kaitlin, C., Alexander, D., Kaleb, S., Eitan, R., Bin, H.(2013). Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface

[12] Kyle, J.M., Ian, C.C., James, H.D., Jose, G.M., Eric, D.R., Ashley, K.S., John, J.S. (n.d). Re-design of a quadcopter capable of autonomous flight and collaboration with a UGV, 2-3 .

[13] Lionel, H., Lorenz, M., Petri, T., Friedrich, F., & Marc, P. (nd). Autonomous Obstacle Avoidance and Maneuvering on a Vision-GUIded MAV Using On-Board Processing, 1.

[14] Nanyaporn, I., W. Nathan Alexander., William J.D., Sheryl, M.G., Amanda D, (2016). Experimental Study of Quadcopter Acoustics and Performance at Static Thrust Conditions, 1

[15] Parashanth, K.R., Preetham, S., Nagaraja, B., Govind, R.K., & Shankapal, S.R. (2013). RealTime Obstacle Avoidance And Navigation Of A Quad-Rotor Mav Using Optical Flow Algorithms, 1.

[16] Saurabh, L., Deepan, K.K., Pavitra, B., Aditya, J., & Mittal, R.K.(nd). Use of LIDAR for Obstacle Avoidance by an Autonomous Aerial Vehicle, 1.

[17] Szafranski, G., & Czyba, R. (2011). Different Approaches of PID Control UAV Type Quadrotor,

[18] Taeyoung L., Melvin L., & Harris M. (2010). Geometric tracking control of a quadcopterUAV on SE(3), 1.

[19] Taeyoung, L. (2014). Geometric Control of Multiple Quadrotor UAVs Transporting a Cable-Suspended Rigid Body, 1.

[20] Taeyoung, L., Melvin, L., & Harris, M. (2011). Control of Complex Maneuvers for a Quadrotor.UAV using Geometric Methods on SE(3).

[21] Xing, H., Mingyi, H., & Hamid, R.K. (2013). Attitude Stabilization Control of a Quadrotor UAV by Using Backstepping Approach, 1