UNSYS digital

The Journal of Instrumentation, Automation and Systems

Development of a Coaxial MAV with Real-Time Obstacle Avoidance Capability

Sagar Setu and Abhishek

Department of Aerospace Engineering, Indian Institute of Technology Kanpur, Kanpur, India

Abstract—This paper discusses the development, implementation, and deployment of a computationally light, yet robust real-time collision avoidance system on a hover capable coaxial rotary wing Micro Air Vehicle (MAV). The real-time capability of the algorithm is demonstrated by performing all computations required for processing the depth camera image onboard at 13 Hz using a single core of a 1.6 GHz quad-core processor -based single-board computer. The primary sensor used is Microsoft Kinect which has an operating range of 0.5-3.5 m. The algorithm uses template matching feature of an open source library OpenCV to find a window through which a vehicle of specified dimensions and known speed can pass safely. The actuator control and active yaw stabilization is done using Navstik which is a Micro Navigation & Control hardware. Two test vehicles are built and equipped with this framework for proof of concept.

Keywords—Obstacle avoidance, coaxial Micro Air Vehicle, indoor semi-autonomous navigation.

I. Introduction

Micro Air Vehicles (MAVs) are a breed of small Unmanned Air Vehicles (UAVs). Defense Advanced Research Projects Agency (DARPA) formally defined MAVs as aircrafts with all dimensions smaller than 6 inches, an approximate weight of 100 g with about 20 g payload capacity and an endurance of 60 minutes [1]. Wide variety of MAVs exist today with primary difference in their source of thrust and lift which determine their payload capacity, maneuverability, endurance and hovering capability. So, the term MAV is now used to refer to a much broader range of vehicles which satisfy the criteria of portability in general sense of the word. Some types of MAVs successfully flown around the world today are fixed wing, conventional single rotor, coaxial rotor, multirotors, cycloidal rotor, flapping wing, etc. A review of such vehicles may be found in [2].

MAVs, owing to their small size and high maneuverability are apt for use in constrained urban environment, such as the inside of a building. Operation in cluttered indoor environment would also require assistance from autonomous flight controllers with the initial goals of reducing the human pilot's work load and eventually for fully autonomous operations with minimal human intervention. With the development of smaller and lighter radio units, surveillance equipment, chemical, biological, and radioactivity sensors, and other kinds of useful payload, autonomous MAVs for various applications have become a realistic possibility. Some other desired qualities in a vehicle for indoor flight are hover/slow-flight capability, compactness and maneuverability in all degrees of freedom. Rotary wing MAVs stand out among other designs when we consider these features. Collision avoidance has been a matter of prime concern in the field of mobile robotics. Researchers have used several sensors in past to detect proximity of obstacles in path, some of the popular choices are discussed below:

a) *Ultrasonic range finder* – This sensor consists of an ultrasonic sound wave generator and detector. It works on a principle similar to radar or sonar. This sensor can provide useful data only in a small solid cone. Additionally, it fails when the inclination of the surface on which it is incident is greater than half of the solid angle the receiver is capable of monitoring. Its capabilities and limitations are discussed in [3].

b) *Infrared proximity detector* – This sensor works by emitting an infrared beam and detecting its intensity to determine the proximity of object. This sensor provides data only along a line. Its performance is also heavily affected by ambient lighting conditions and is especially deteriorated by sunlight. Another issue is reflecting off of incident infrared beam from polished surfaces. An example of a MAV using infrared proximity sensors for collision avoidance can be seen in [4].

c) *Laser scanner* – These are very accurate proximity sensors usually based on time-of-flight calculations to obtain distances. However, they are bulky and expensive. Moreover, they have low resolution, i.e., they do provide 2-D or 3-D scans but at very few points in their field of view. Consequently, they are prone to missing slim obstacles. Shim, et al. [5] have used laser scanner on a UAV for collision free flight. It has also been used for detection and tracking of moving objects [6]. It is a popular sensor for implementation of SLAM (Simultaneous Localization And Mapping) for both indoor and outdoor applications [7]-[11].

d) *Optic flow sensors* – Optic flow sensors essentially provide the ratio of velocity parallel to a surface and the distance to it. In the event of a particular surface nearing the vehicle, this ratio will increase due to decrease in distance and a collision can be predicted. Collision avoidance using optic flow has been implemented successfully [12]-[16] but remains particularly useful for avoiding walls in corridors or caves or large obstacles due to its limitations. Optic flow sensors

Corresponding author: Abhishek (e-mail: abhish@iitk.ac.in).
This paper was submitted on January 31, 2014; revised on May 12, 2014; and accepted on June 20, 2014.

appear in nature as the vision system of insects like houseflies and bees. Their limitation can be observed by noticing that these insects are incapable of avoiding obstacle if they are moving perpendicularly into it. This is why we see houseflies hitting a wall repeatedly without being able to move around it.

e) *RGB camera* – RGB cameras are the common digital cameras that we find around us. In structured environments, where the colour and texture of obstacles is distinct from that of rest of the surrounding and can be used to differentiate between them and find an obstacle free course. Unfortunately, we don't find such structure in our everyday life rendering this technique impractical to vehicles that have to navigate in an unknown environment. Some examples of using this sensor for collision avoidance have been discussed below.

f) *Depth camera* – These cameras consist of an infrared projector and a camera. The distance estimation is done by raw disparity calculation on the infrared matrix projected. Examples of such camera are the depth camera unit in the Microsoft Kinect sensor and Asus Xtion Pro. These cameras have good accuracy, high resolution, large field of view and appropriate operating range for slow moving aerial vehicles. Some problems associated with them are failure in sunlight or in presence of other infrared sources and inability to detect transparent obstacles.

Several attempts have been made by past researchers to use monocular vision to detect and avoid obstacles in path of mobile robots. With the advancement in 3-D sensing technology, the idea of using lighter yet more accurate depth sensors for collision avoidance, object tracking and localization attracted the attention of many researchers across the world. Benavidez and Jamshidi [17] devised a technique wherein their ground vehicle could identify traversable area using RGB camera after it had been trained using a depth camera. Biswas and Veloso presented another method suitable for ground robots based on fast sampling plane filtering technique on 3-D point cloud [18]. Their approach, however, uses previously acquired data for matching and prediction. In another work, depth image from Kinect sensor is segmented into few vertical sections effectively using it as a low resolution device [19]. Consequently, it predicts mainly where the vehicle should not go rather than finding the exact nearest safe path. The performance of such collision avoidance system is similar to what can be obtained using an array of line sensors like sonar and infrared proximity sensors. Flacco, et al. used Kinect for a third person view of the robot and the obstacle and analyzed complete 3-D point cloud data to obtain distances and provide feedback [20].

All the systems mentioned here are in general computationally heavy. For the computation of depth data onboard we have relatively less powerful single-board computers and the processing must be simple enough to run in real-time on these computer boards. Some of these methods also needs prior training, familiarization and in some cases full 3-D point cloud of the area being navigated into. This makes them useless for exploration tasks in unknown environments. Therefore, a simple and computationally light method that requires no information of its surroundings beforehand for obstacle detection and avoidance is needed.

In this research, Microsoft Kinect sensor is used as primary sensor for obstacle detection and avoidance due to the fact that it directly provides the depth data without any computational penalty, is cost effective and is very accurate within its range, which is adequate for a slow moving vehicle meant to be operated indoors. The data stream from depth camera of Kinect sensor was acquired using *libfreenect* and processed using *OpenCV* and *Numpy* using *Python* programming language. In the heart of the program, object matching function of OpenCV predicts if a part of frame captured denotes a completely obstacle free space or not. Two vehicles were constructed as test platform for the designed collision avoidance algorithm. The first vehicle was a twin-coaxial rotor vehicle with its rotors in tandem configuration. The other vehicle was a regular coaxial vehicle. The structure of the vehicles was constructed using carbon fiber composite. The actuators on the vehicles were controlled using onboard microprocessor units. The yaw of the vehicle was actively controlled using feedback from an onboard gyroscope. The vehicles relied on stabilizer bar attached to each coaxial rotor system for passive roll and pitch stability.

II. Obstacle Avoidance Algorithm

Kinect is a motion sensing input device developed by Microsoft for the Xbox 360 video game console. It features an RGB camera, depth sensor and multi-array microphone. The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under most ambient light conditions [21]. The Kinect sensor gives video output at a frame rate of 30 Hz. The python wrapper of the libfreenect library is used to obtain the raw 11-bit disparity value in form of a matrix which needs to be converted to the depth information. For this the formula proposed by Stéphane Magnenat [22] is used which is given by:

$$s = 0.1236 \cdot \tan\left(\frac{\text{raw Disparity}}{2842.5} + 1.1863\right) \quad (1)$$

where s is distance measured in meters. The results of the calibration test along with the corresponding percentage error associated with the depth measurements, for the full operational range of Kinect (50-350 cm), are shown in **Table I**. The high accuracy of Kinect depth sensor at measuring distances over a range of 50 cm to 350 cm can be observed from the low mean absolute error of 0.46 % for the data set shown in **Table I**.

The earliest attempts for the identification of the window for safe passage of the vehicle involved two key steps [23]:

a) Identification of region of interest in the depth image

b) Scanning of the region of interest to identify a window for safe passage of the vehicle.

TABLE I Kinect sensor error estimate

Obstacle	Sensor	Using Eq. 2.5	%
50	393	49.1	-1.80
75	627	74.7	-0.40
100	743	99.9	-0.10
125	814	125.5	0.40
150	861	151.1	0.73
175	893	175.4	0.23
200	917	199.4	-0.30
225	937	225.1	0.04
250	953	250.8	0.32
275	965	274.4	-0.22
300	976	300.2	0.07
325	984	322.3	-0.83
350	992	347.9	-0.60

The implementation of the technique of collision avoidance and path finding has been done on python programming language with the aid of libfreenect, OpenCV and Numpy libraries. Combinations of inbuilt functions available with these libraries have been used to substitute running loops over images and matrices wherever possible to make the code more time-efficient. The first step in the process of obstacle avoidance is obtaining data from the Kinect device. This is done by using Python wrapper of libfreenect library. Calling the appropriate functions, we obtain a Numpy array of integers of size 640×480. Each of the 307200 elements is the raw disparity at that pixel and can be converted to depth information in real world using (1).

Figure 1 (a) and **Figure 1 (b)** respectively show the view from RGB and depth camera of the same location. The Numpy array is scaled down to 320×240 to reduce the number of elements that have to be processed in the rest of the steps. The reduction in size is done by choosing odd rows and columns and rejecting the even ones. This method of scaling down helped us preserve the real identity of every pixel which would have been lost on using techniques which operate on a block of neighboring pixels. On the other hand, choosing a connected set of pixels as region of interest would have reduced the field of view. The downside of using this technique is decreased resolution, which was observed to cause no significant effect in practical situation. **Figure 2 (a)** shows scaled down depth image which has the same nature as full-scale image in **Figure 1 (b)**.

The next step involves classifying every element as an obstacle or otherwise and obtaining a binary image accordingly. For our purpose, we have chosen 2 m to be the distance below which we classify an element as obstacle and set its value to true (represented as white pixel in binary image). We expect a slow moving indoor vehicle to move at 1 m/s. Selecting a threshold distance of 2 m allows the pilot a generous time of 2s for moving around the obstacle. This distance translates to the raw disparity of 920 according to (1). The value of rest of the points is set to false (represented as black pixels in binary image shown in **Figure 2 (b)**.

Next, the binary image is matched against a template image which represents a window of safe passage. Based on the imaging information of Kinect, for the scaled down image a 110×110 pixel image consisting of only black pixels represents approximately a 0.8 m×0.8 m window in the real world which is sufficient for the vehicle to safely pass through. The intrinsic parameter for Kinect's depth camera have been found in [24] and presented in **TABLE II**.

TABLE II Important intrinsic characteristics of Kinect IR camera

Parameter	Symbol	Value
Focal Length	f	5.453 ± 0.012 mm
Principal point offset	x_0	0.003 ± 0.039 mm
	y_0	0.008 mm
Pixel Size	p_x	9.3 µm
	p_y	9.3 µm

Using these values, we can verify the size of window chosen in pixels. Here, we have ignored the effect of offset of principal focal point (x_o,y_o) on the intrinsic characteristic of depth camera's lens as it is two orders of magnitude smaller than the focal length of the lens.

The template binary image is slid across the original binary image starting from the top leftmost part and evaluated. The extent of match is stored in a third resultant image. This step is implemented using the template matching function available in OpenCV. The algorithm being used is the sum of squared difference which is one of the simplest and most time efficient methods offered by this function.

If I represents the original image of dimension W × H, T represents the template image of size w × h, and R represents the result in image form of dimension (W – w + 1) × (H – h + 1), the calculation of result is done as

$$R(x,y) = \sum_{x`,y`} \left(T(x,y`) - I(x+x`, y+y`)\right)^2 \qquad (2)$$

Here, x' varies in 0, 1, …, w – 1 and y' varies in 0, 1, ….., h – 1. Since, we have only binary images to deal with, this method works very well and we obtain the value zero in resultant image for perfect matches between template and original image representing top left corner of every window of safe passage. Upon obtaining the result image, it is converted to a Numpy array for convenient and efficient manipulations. This

array is converted to a Boolean array with all the elements with value less than 10 marked as true and the remaining as false. Zero is not chosen as the threshold value as it has been observed that there are minor fluctuations in the Kinect input and a few pixels with false values can cause an eligible window of safe passage to be rejected. Choosing a value of 10 means that we are allowing a tolerance of 10 misreported values in a window of dimension 110 × 110 (12100 points).

(a) **(b)**

Figure 1 Results of algorithm shown with overlayed square in: (a) Binary image and (b) Depth image

From this Boolean array, the indices of all the elements which are true is saved in another tuple P whose first array P_X stores the x-coordinate of a window and the second array P_Y stores the corresponding y-coordinates. For the next step, which is finding the window of safe passage closest to the heading of the vehicle represented by the center of the original image I, the location (160, 120) is taken to be center location C. The sum of squares of the difference of location along x axis of a window from 160 and location along y axis of the window from 120 gives us the distance of the window from the center of the image. If $P_X(i)$ and $P_Y(i)$ represent the x and y coordinates of the top leftmost corner of i^{th} window of passage. The corresponding distance from center is D(i), the i^{th} element of an array D with same size as P_X and P_Y.

$$D(i) = (P_X(i) + 35 - 160)^2 + (P_Y(i) - 35 - 120)^2 \qquad (3)$$

The pixel with minimum value in the array D represents the top leftmost pixel for window of safe passage closest to the current heading. The required maneuver to pass through this window can then be performed using a PID control scheme. A control scheme that involves feedback based closed loop operation becomes necessary to deal with variable factors like air gusts and change in rotor speed (and hence, thrust) due to decay in charge of the power supplying battery pack.

In **Figure 1 (a)** and **(b)**, the detected window has been shown as a square box on binary depth image and RGB image respectively.

A flowchart depicting the algorithm for obstacle avoidance described earlier is shown in **Figure 2**.

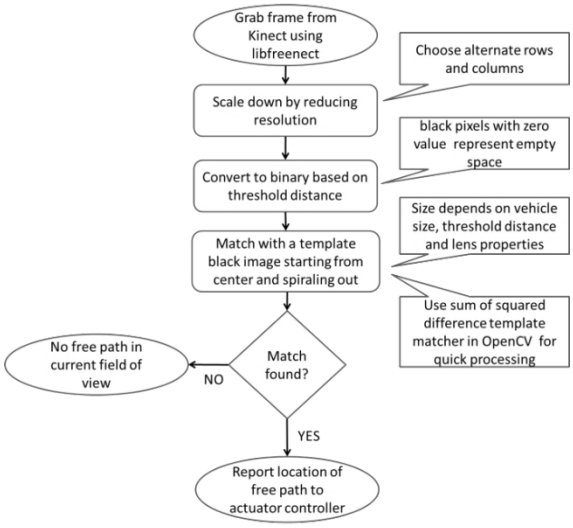


Figure 2 Flowchart depicting obstacle avoidance algorithm

III. VEHICLES

The collision avoidance algorithm developed with the objective of achieving a semi-autonomous MAV needs to be tested on a slow moving flight vehicle. The features desired in the test platforms were

a) Hovering/slow flying capability
b) Enough payload capacity to carry Kinect and single-board computer
c) Small size to suit indoor flying
d) Good inherent stability preferably achieved through passive means.

For the purpose of building our vehicle, coaxial configuration was chosen keeping the above requirements in view. The baseline model taken was off-the-shelf LAMA 400D by Walkera. This model has a stabilizer bar attached with the top rotor by the virtue of which it exhibits good stability in roll and pitch. It weighs 580 g and has rotor diameter of 497 mm. The model has been experimentally determined to be able to fly for approximately 8 min with an extra load of 120 g using the stock rotor blades. Initially, a twin coaxial vehicle was constructed to ensure enough payload capacity. However, the vehicle designed posed some problems for the pilot, mainly due to yaw-pitch coupling, which made it difficult to use it as test vehicle in its current state. So, an easier to fly conventional coaxial vehicle using the same baseline model was built which satisfied the payload requirements while being easily pilotable.

A. Tandem Twin Coaxial Vehicle

To meet the required payload capacity of about 200 g including a stripped down Kinect and a single board computer, two LAMA 400D models were joined back-to-back in tandem configuration. Tandem configuration allows the vehicle to pass through a window of same size as a single coaxial vehicle as the fuselage frontal area remains unchanged. The details of the final vehicle have been discussed in the following sections.

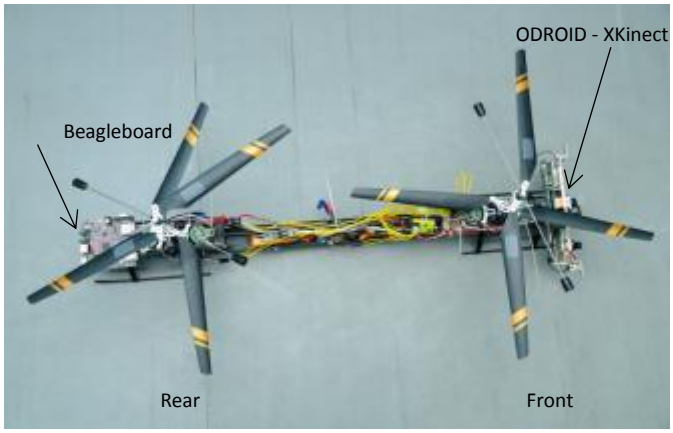
To join the two helicopters, carbon-fiber-reinforced polymer angles with 5 layers of carbon fiber and L-shaped cross-section were used as main load bearing members. Their tail booms were joined together to increase the torsional rigidity. Finally, trusses of carbon fiber strips were used to connect tail boom and angles together providing additional strength. Carbon fiber has high strength to weight ratio in bending making it suitable for use on a MAV. A top and side view of the final vehicle is shown in **Figure 3(a)** and **(b)** respectively.

The dimensions of the vehicle are:

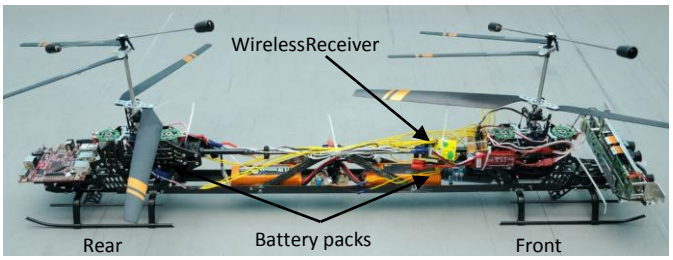
- Height: 26 cm
- Width: 50 cm
- Length: 102 cm

The total weight of the vehicle was 1250 g and the approximate moment of inertia in yaw was calculated to be 0.2 kg m^2.

The vehicle has 4 stock brushed motors, 2 at each end and each one driving one rotor, with the corresponding motors on both ends receiving the same signal. The roll and pitch are controlled by actuating the two swashplates in a symmetric manner. To tilt the swashplate similarly at both ends, the signal for swashplate actuator at rear end is the complementary signal of the one at front end. The yaw control is done using differential RPM of upper and lower rotors at both ends. Due to high inertia in yaw the pilot has decreased yaw authority which makes the vehicle less maneuverable. This tandem coaxial vehicle with swashplate cyclic control is a unique design and has not been built earlier. It has exhibited excellent stability and warrants research on its dynamics to remove its limitations as it has shown the potential of being a complete solution for high payload indoor flying MAV.



(a)



(b)

Figure 3 Tandem twin-coaxial vehicle: (a) Top view and (b) Side view

The main electronic components of the vehicle are ESCs for brushed motors, wireless receiver, a gyroscope and two microcontroller boards (one motor control and another for roll and pitch control). The brushed ESCs needs a PWM signal at 2 KHz while the roll and pitch servo work with a 50 Hz PWM signal. Hence, there is a requirement of using two separate boards with their timers clocked at appropriate frequencies. The four-channel signal, consisting of roll, pitch, throttle and yaw values, from wireless receiver is sent to the corresponding microcontroller boards (Arduino UNO). The board controlling the motors receives angular velocity feedback from a gyroscope and adjusts motor RPM of upper and lower rotors to stabilize the vehicle in yaw. A PID controller is used for this purpose. The tuning parameters and the calculated gains using Zeigler-Nichols tuning rule (second method) have been given in **TABLE III**. The gain values obtained were fine-tuned experimentally through flight testing, except for yaw degree of freedom, which was tuned using a custom built yaw test stand, which restricts all other degrees of freedom except yaw. The feedback available from gyroscope in this case was in the unit rad/s.

Since, the helicopters were united back-to-back; there was a need to modify the control of swashplate servos. In the present configuration, if the position of servo on front end of the vehicle is represented by θ (in degrees), the position of the corresponding servo on the rear end of the vehicle should be its supplementary angle ($180^o - \theta$) to ensure that the tilt of both the swashplates with respect to a common frame of reference is identical. To address this need, the Arduino UNO board controlling the swashplates was programmed to generate output

signals for four servo motors arranged symmetrically at either end of the vehicle.

TABLE III Yaw stability controller parameter for twin coaxial vehicle

Parameter	Symbol	Value
Critical gain for sustained oscillation	K_{cr}	2
Period of oscillation	P_{cr}	2.6 s
Proportional gain coefficient	K_p	1.2
Integral gain coefficient	K_i	0.8
Differential gain coefficient	K_d	0.4

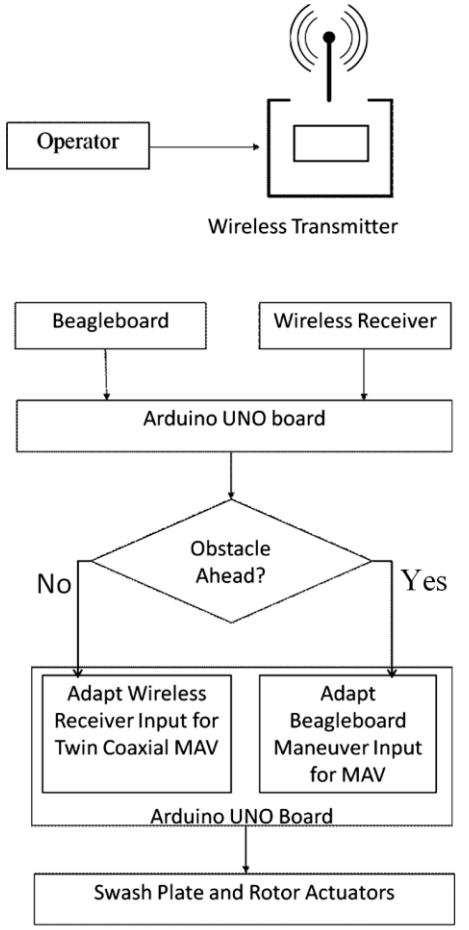


Figure 4 Control flow of the vehicle

The wireless receiver in turn receives signal from a transmitter controlled by a human operator. In the event of detection of an obstacle, the onboard computer instructs the microcontroller to override human operator input with those generated by the obstacle detection algorithm. The control loop involving user, Beagleboard and UNO board has been depicted in **Figure 4**. The Beagleboard communicates with Arduino through a single-bit GPIO pin, which becomes high when an obstacle is detected. In such event the vehicle is pitched back to avoid collision, this either retards the vehicle to stand still or move backwards till it is sufficiently away from obstacle to allow human pilot sufficient time for taking evasive action.

B. Single Coaxial Vehicle

Due to some limitations, foremost of which was pitch-yaw coupling, a new test platform based on single coaxial design was built. Using the same baseline coaxial model, the stock blades were replaced with larger high-lift blades, stock brushed motors were replaced with brushless motors and the body was modified to accommodate the payload.

The structure of this vehicle is again made out of carbon-fiber composite to keep weight down while maintaining strength. The vehicle has been given a layered design for carrying extra electronics to minimize the rotor downwash interference. Tail boom originally present in the model is removed as it serves no purpose. The main characteristics of the vehicle are

a) Weight: 990 g
b) Height: 33 cm
c) Length = Width = Rotor Diameter = 49 cm
d) Motors: 1000 Kv brushless outrunner

To meet the payload requirements, larger high-lift blades were used. The main geometrical characteristics of new and the old blade are compared in **TABLE IV**.

(a)

(b)

Figure 5 Instrumented coaxial MAV and its blades: (a) Single coaxial vehicle; (b) new and old rotor blades

TABLE IV Comparison of blade geometric properties

Characteristic	New	Old
Length	235 mm	225 mm
Maximum chord	34 mm	29.2 mm
Position of maximum chord from the root	45 mm	38 mm
Tip chord	15.5 mm	16 mm
Surface length at the maximum chord	35.6 mm	30.5 mm
Camber at maximum chord	4.1 mm	3.1 mm
Approximate percent camber	9 %	8 %
Radius of curvature	0.091 mm	0.073 mm
Taper ratio	2.2	1.8

Figure 6 shows the geometry of the carbon fiber made layer which carries Kinect and Odroid-X. Assuming that the whole mass of the vehicle is uniformly distributed over this area, the mass moment of inertia about the axis of rotor shaft is calculated to be 6.75×10^{-3} kg m^2. This is significantly lower than that for tandem vehicle and predicts that this vehicle will have better yaw maneuverability and low total thrust change when the vehicle is yawing.

The intercommunication between different electronic components for this vehicle is similar in structure to the twin coaxial vehicle shown in **Figure 4**. However, instead of Beagleboard, a quad-core single-board computer Odroid-X has been used. Arduino UNO has also been replaced by Navstik.

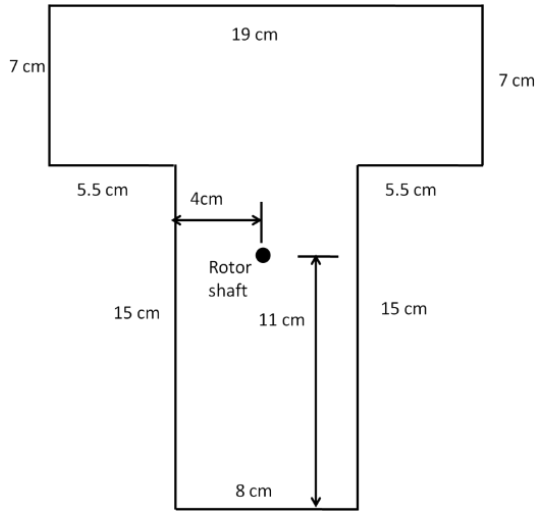


Figure 6 Payload carrying layer of MAV made using Carbon Fibre Composite

The yaw stabilization is done in a similar way as mentioned earlier for the tandem vehicle. Once again, a yaw test bed was used to tune gains of the PID controller for yaw stabilization. The parameters obtained are given in **TABLE V**. The angular velocity feedback available in this case was in the unit deg/s.

TABLE V Yaw stability controller parameter for single coaxial vehicle

Parameter	Symbol	Value
Critical gain for sustained oscillation	K_{cr}	50
Period of oscillation	P_{cr}	4 s
Proportional gain coefficient	K_p	30
Integral gain coefficient	K_i	15
Differential gain coefficient	K_d	0.5

IV. INTEGRATION OF SUBSYSTEMS

Different electronic components have different requirements for power and signal. Since, we have a single power source, a 12 V, 1500 mA·h, 30 C battery pack, there was a need to condition its output for a 5 V regulated supply. For this purpose, IC 7805 was chosen as it is a small, light and reliable voltage converter. It drops down input voltage from 8-12 V to 5 V. Although, the IC is designed to supply 5 V at maximum 1 A, its efficiency drops as it heats up and the cut-off current shifts to 0.7 A. Small capacitance of 33 μF and 100 μF have been used across output and input of IC 7805 to filter out any momentary disruptions in power supply. A diagram showing power supply to different components of the vehicle is shown in **Figure 7**.

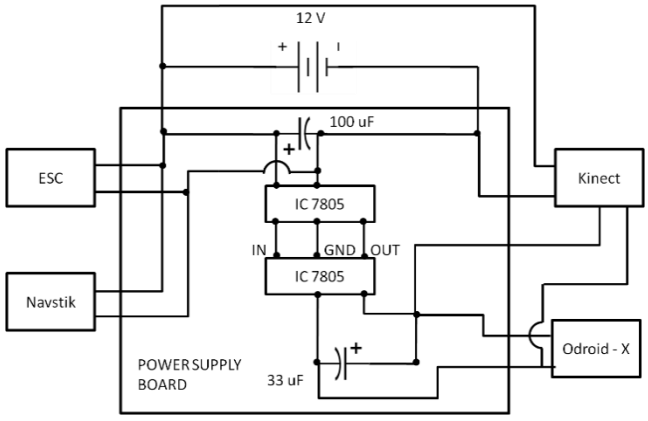


Figure 7 Power supply circuit for MAV

Data flow among all the components used has been depicted in **Figure 8**. The communication between Odroid-X has been done using UART protocol. Python allows the creation of a virtual serial port over USB which can be used like the conventional serial port. However, this requires a Serial – USB converter at the end of Odroid-X. Additionally, a voltage level

shifting is required as the USB port at Odroid-X operates at 5 V while the UART port on Navstik runs at 3.3 V. Both these functions are available in a Navstik accessory named IvyGS which has been used along with Navstik. Gyroscope data is acquired using pre-built functions made available by Navstik manufacturers. PWM signal coming from wireless receiver is also read using inbuilt functions based on digital interrupts.

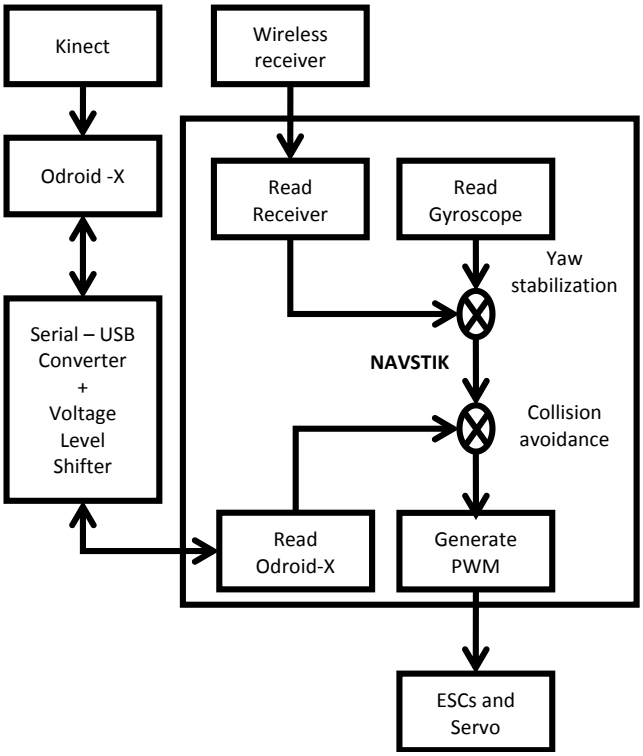


Figure 8 Full data flow diagram among different components

As mentioned in section II, the approximate intrinsic matrix of the depth camera of Kinect sensor is given as:

$$K = \begin{pmatrix} 586 & 0 & 0 \\ 0 & 586 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The threshold distance for distinction between obstacle and otherwise is Z = 2 m. So, the relative position of obstacle free window with respect to the current pose of the Kinect camera is given by:

$$X = \frac{uZ}{f_x}, \qquad Y = \frac{vZ}{f_y} \tag{4}$$

where, X and Y are coordinates in real world coordinates, u and v are the position of safe window of passage in pixels with respect to image center and f_x and f_y are the scaled focal length of depth camera lens.

Given the restrictions due to frame size (640 × 480) and size of template being matched (220 × 220), using (4) and (5), we can see that X varies from -0.72 m to 0.72 m and Y varies from -0.44 m to 0.44 m. To design a controller that would lead our vehicle to execute an accurate maneuver to pass through the detected window required an elaborate process of complete system identification of the vehicle, which was not possible in the given time frame. Instead, simple rules for collision avoidance were implemented which involved slow pitch back in presence of an obstacle and rolling/yawing towards the window of safe passage. This gives the pilot extra time to move around the obstacle and some assistance in navigating in the right direction, which is an appropriate approach for a semi-autonomous vehicle.

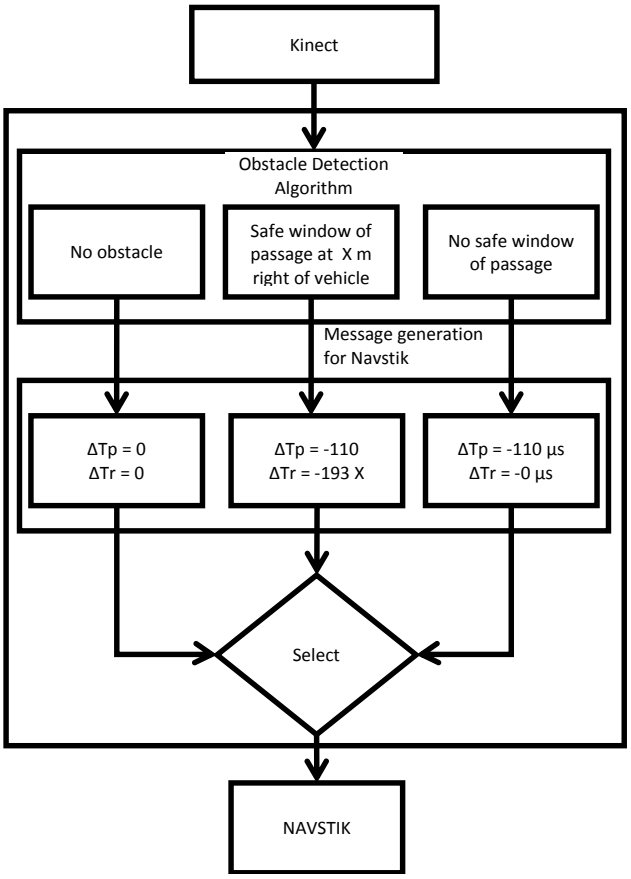


Figure 9 Summary of fully integrated collision avoidance system

In the event of detection of an obstacle, the first and the consistent step taken is to pitch the vehicle back (nose up) by rotating the pitch servo by 20°. Assuming a linear relation between the servo angle (θ) and the pulse-duration (T_p) of PWM signal, we get:

$$\theta = (T_p - 1500) \cdot \frac{9}{50} \tag{5}$$

where, θ is in degrees, T_p is in µs and clockwise is the positive direction of rotation. Using (5), we get the required change in pitch servo pulse width to be approximately 110 µs.

As far as roll motion is concerned, the value of change in servo arm position depends on the location of safe window of passage in the real world coordinates. A moderate value of 25° was selected as the maximum servo arm displacement and a linear relationship between window distance and servo arm was

followed. So, with varying location of window X (in m), the roll servo arm displacement $\Delta\theta_r$ (in degrees) is given as:

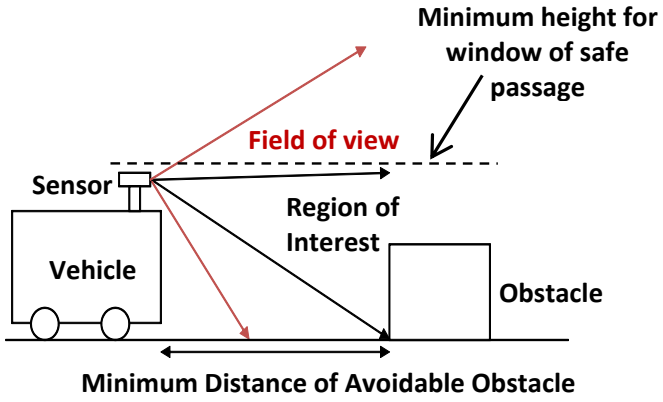
$$\Delta\theta_r = 25\frac{X}{0.72} \quad (6)$$

Using this relation and equation 5, the change in pulse-duration of PWM signal is given as:

$$\Delta T_r = 1250\frac{X}{6.48} \quad (7)$$

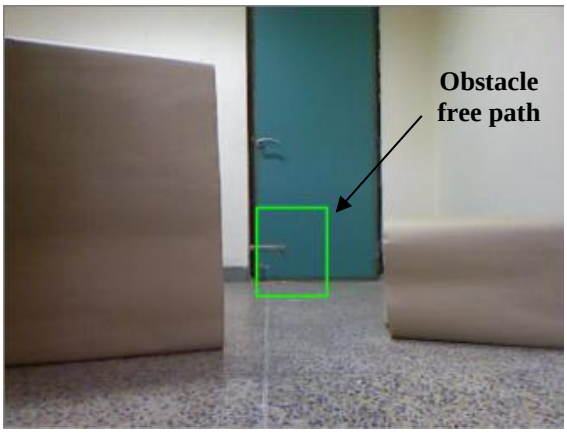
In the event that an obstacle is detected and no safe window of passage can be found in the field of view of Kinect's depth camera, the vehicle only pitches back according to the rule discussed above. No change in roll servo arm position is done in this case. Also, if the window coordinates consist of Y component too, it is ignored as our platform does not have enough extra power to make a climb maneuver. Instead, the vehicle pitches back and rolls according to the X components of location of window of safe passage. Figure 9 shows the actions taken by the vehicle according to the location of obstacles in the form of a flowchart.

(a)

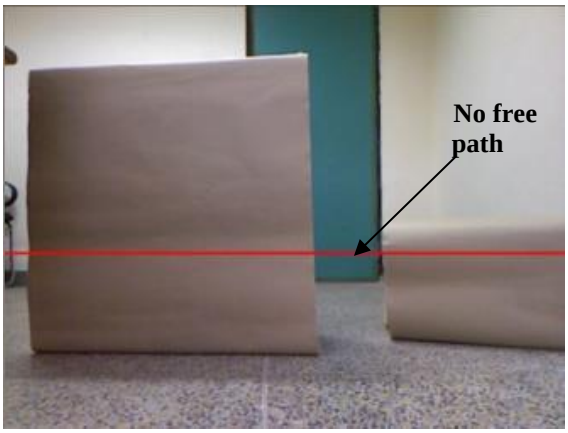


(b)

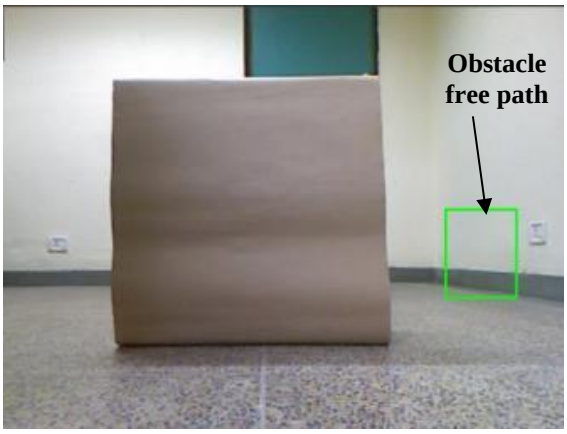
Figure 10 (a) Platform for initial testing, (b) Outline of region of interest selection



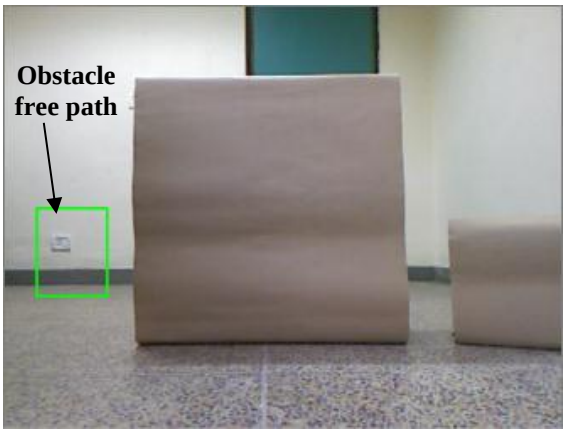
(a)



(b)



(c)



(d)

Figure 11 Free path finding algorithm for terrestrial vehicle

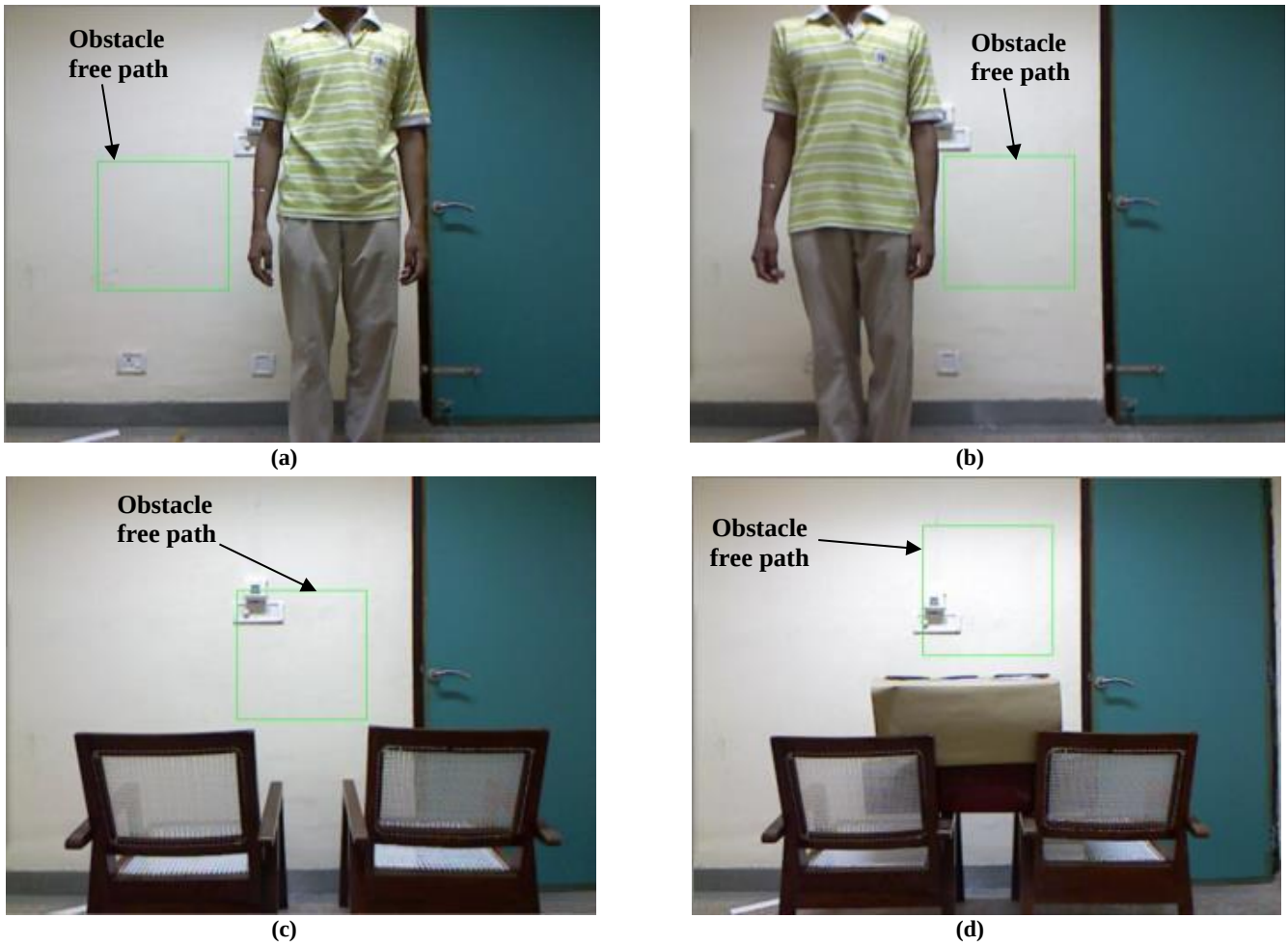
V. RESULTS

Before deploying the system on the vehicle, the collision avoidance system was rigorously tested in various scenarios. To begin with, the system was tested on a terrestrial vehicle so that its robustness can be tested before setting it up on an aerial vehicle which requires more reliability. The first algorithm tested instructed the ground vehicle to stop in case an obstacle is detected right in front of it. For this, an off-the-shelf terrestrial vehicle named *Streak Auto* was used. The vehicle was equipped with a 2 GHz laptop which communicated to its motor controlling microcontroller board *Rhino control board* via a USB to RS232 interface. Since, we were dealing with a ground vehicle only a small portion in vertical dimension of the complete image, named region of interest, was needed to be processed. The vehicle has been shown in **Figure 10 (a)** while **Figure 10 (b)** depicts the selection of region of interest in the depth image over which proximity threshold for obstacle identification was applied. Upon complete integration, this vehicle was able to avoid colliding with any obstacle right in front of it by commanding its motors to stop.

TABLE VI Specification of different computing platforms

	Laptop	Beagleboard	Odroid-X
Processor speed	2 GHz	1 GHz	1.4 GHz
RAM	3 GB	512 MB	1 GB
No. of cores	1 (core 2 duo)	1	4
Operating system	Ubuntu 9.10	Angstrom	Ubuntu 12.1

This step verified the reliability of Kinect as a proximity sensor when used with the chosen libraries and the communication between the computing platform and the motor controlling microcontroller over a virtual serial port in Python. The inability of Kinect sensor to detect slender wire like objects was also observed.



(a) **(b)** **(c)** **(d)**

Figure 12 Free path finding algorithm for aerial vehicle

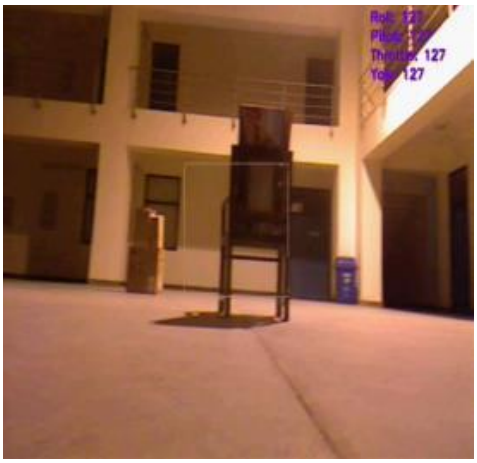
In the next step, the algorithm was expanded to include finding window of safe passage for a terrestrial vehicle. Once, again the whole field of view in vertical direction was not required as it was designed for a ground robot. The algorithm found the window closest to the center of the image and its results are shown in **Figure 11**. The detected window is represented by a green box while a red line represents unavailability of a safe path. Comparing **Figure 11 (c)** and **(d)**, it can be observed how the chosen window shifts its side depending upon whichever is closer to the center of the image. This strategy was chosen so that the vehicle can pass safely by doing minimum maneuver.

The capability of the algorithm was expanded further to adapt it to be used on an aerial vehicle. The step involving extraction of region of interest was removed and the whole image was now processed to find a window of safe passage closest to the center of the image. **Figure 12** shows some cases of implementation of this algorithm. Once again, the detected window is represented by a green box. It can be noticed from the images that the obstacle and the wall, which is not detected as an obstacle, are quite close. When the test was done, the obstacles were kept at nearly the threshold distance, chosen to differentiate between obstacles and free path, away from the Kinect sensor. This also established the accuracy of Kinect and the present algorithm.

After the complete integration of different sub-systems, the vehicles were first flight tested without the collision avoidance system running. The twin coaxial vehicle performed satisfactorily in terms of stability, endurance and extra power available. However, the problem of yaw-pitch coupling made it difficult to maneuver. Hence, only a simple version of collision avoidance system could be implemented on it wherein the vehicle stopped moving ahead in case of presence of an obstacle. Few test flights with complete collision avoidance system were tried but could not be successfully completed due to difficulty in piloting it.

The single coaxial vehicle also exhibited desired level of stability. Although it exhibited poor endurance and climb, due to its increased weight, the vehicle was easy to pilot and complete collision avoidance system could be tested on it. After having tested and verified the algorithm in open loop mode, the algorithm is flight tested in closed loop mode by testing the algorithm onboard the single coaxial vehicle. For simplicity, only the roll degree of freedom was commanded by the obstacle avoidance algorithm in this flight test. Yaw degree of freedom was controlled in Stability Augmentation mode using PID controller and the vehicle was flown by a human pilot in semi-autonomous mode. The tests yielded satisfactory result as the vehicle automatically slowed down in presence of an obstacle ahead of it and safely exhibited roll maneuver toward the path of safe passage identified by the algorithm.

The results shown (**Figure 13**) are a sequence of screenshots of the onboard view from the Kinect RGB camera obtained during one of the test-flights. While the data from the depth camera of Kinect sensor was used for collision avoidance, the video from RGB camera was captured simultaneously and the results of the computations from the collision avoidance algorithm were overlaid in the form of a box representing the nearest window of safe passage for the vehicle and the desired raw values of roll, pitch and yaw angles shown in the top-right corner of every frame.

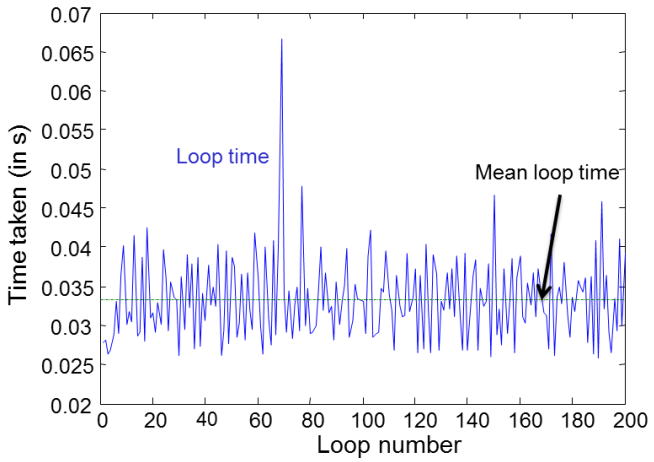


(a)

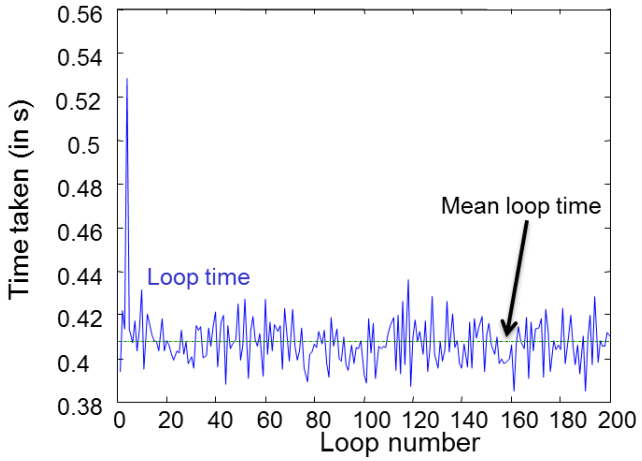
(b)

(c)

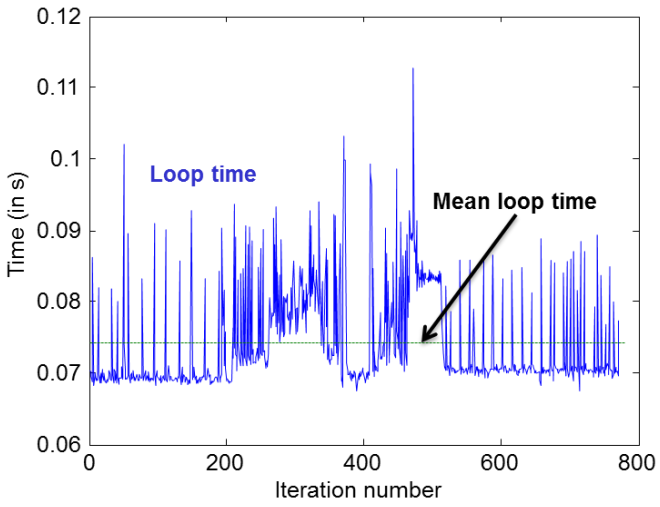
Figure 13 Onboard view from Kinect RGB camera during flight test depicting realtime obstacle detection and avoidance



(a)



(b)



(c)

Figure 14 Loop-time for processing one frame from Kinect's depth camera: (a) Laptop, (b) Beagleboard-xM and (c) Odroid-X

It should be noted that the neutral value of roll input is 127 which decreases when the vehicle rolls left and increased when the vehicle rolls right. The flight test experiment is initiated with the human pilot trying to fly the vehicle directly into an obstacle as shown in **Figure 13 (a)**, at this point the auto-pilot intervenes and finds the safe passage for the vehicle towards the left of the obstacle (**Figure 13 (b)**) and overrides human pilot input to roll the vehicle enough to change the course of the vehicle as shown by reduction in roll value from 127 to 59. **Figure 13 (c)** shows the vehicle with changed attitude flying away from the obstacle autonomously.

After successful implementation and testing of obstacle avoidance algorithm, the efficiency of the different computational platforms used is also compared. This is done by comparing the loop refresh rate for all the platforms. It is of immense value due to the weight constraints imposed due to limited payload available on a MAV. It can be observed from **Figure 14**, the loop execution rate for Beagleboard is nearly an order of magnitude greater than that for laptop and Odroid-X. The loop refresh rates for laptop, Beagleboard and Odroid-X are nearly 29 Hz, 3 Hz, and 13 Hz. In the case of laptop, the algorithm runs at maximum possible speed, which is the maximum frame rate provided by Kinect sensor. In case of Odroid-X, a refresh rate of over 13 Hz is achieved which is satisfactory considering the computation is done onboard. However, the performance of Beagleboard is poor and is unacceptable making it unsuitable for a real-time obstacle avoidance system.

VI. Concluding Remarks

Development, implementation and deployment of a real-time collision avoidance system on a hover capable coaxial rotary wing Micro Air Vehicle (MAV) was undertaken. The key contributions of this work and the conclusions drawn from it are the following:

- The utility of Kinect and Odroid-X combination as a cost effective yet robust obstacle detection and avoidance system was established.
- A reliable framework of hardware, software and sensors was built and successfully tested for obstacle detection and avoidance for a rotor based MAV.
- Two different single board computing platforms were tested and systematically compared for their possible use as an onboard computations unit for a hover capable MAV for collision avoidance. Odroid-X was found suitable for the current real-time obstacle avoidance algorithm, as it could process the data coming from Kinect at 13 Hz.
- Two different configurations of coaxial vehicles were built and tested. Coaxial helicopter was found to be easier to pilot and navigate compared to the tandem coaxial vehicle which exhibited pitch-yaw coupling.

The ongoing research aims to build a fully-autonomous rotary wing MAV capable of indoor navigation. The work presented in this paper establishes a standalone platform equipped with collision avoidance system with capability to perform onboard computations. Work is under progress for future implementation of localization/mapping and related control algorithms.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support provided under National Program on Micro Air Vehicles (NP-MICAV) by the Aeronautics Research and Development Board of India for this research study.

REFERENCES

[1] Mueller, T. and DeLaurier, J., "Aerodynamics of Small Vehicles", Annual Rev. of Fluid Mech., Vol. 35, 2003, pp. 89-111. CrossRef

[2] Setu, S., "Development of Micro Air Vehicle with Real Time Obstacle Avoidance", M. Tech. Thesis, Department of Aerospace Engineering, Indian Institute of Technology Kanpur, India, 2013.

[3] Roberts, J., Stirling, T., Zufferey, J. C., and Floreano, D., "Quadrotor using minimal sensing for autonomous indoor flight", Proceedings of the European Micro Air Vehicle Conference and Flight Competition (EMAV2007), Toulouse, France, September 17-21, 2007.

[4] Borenstein, J., and Koren, Y., "Obstacle avoidance with ultrasonic sensors", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, 1988, pp. 213-218. CrossRef

[5] Shim, D.H., Chung, H., and Sastry, S.S., "Conflict-free navigation in unknown urban environments", Robotics & Automation Magazine, IEEE, Vol. 13.3, 2006, pp. 27-33. CrossRef

[6] Mendes, A.; Bento, L.C.; Nunes, U., "Multi-target detection and tracking with a laser scanner," Intelligent Vehicles Symposium, 2004 IEEE , pp.796,801, 14-17 June 2004. CrossRef

[7] Newman, P., Cole, D., & Ho, K., "Outdoor SLAM using visual appearance and laser ranging," *IEEE International Conference on Robotics and Automation*, May 2006, pp. 1180-1187.

[8] Biber, P., Andreasson, H., Duckett, T., and Schilling, A., "3D modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera", In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Vol. 4, pp. 3430-3435). IEEE.

[9] Brenneke, C., Wulf, O., and Wagner, B., "Using 3d laser range data for slam in outdoor environments", In *Intelligent Robots and Systems, October 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* (Vol. 1, pp. 188-193). IEEE.

[10] Guivant, J., Nebot, E., and Baiker, S., "Autonomous navigation and map building using laser range sensors in outdoor applications", *Journal of robotic systems*, Vol. 17, No.10, 2000, pp. 565-583. CrossRef

[11] Kim, S., and Oh, S. Y., "SLAM in indoor environments using omni-directional vertical and horizontal line features", *Journal of Intelligent and Robotic Systems*, Vol. 51, No.1, 2008, pp. 31-43. CrossRef

[12] Nelson, R.C.; Aloimonos, J., "Obstacle avoidance using flow field divergence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.11, No.10, Oct 1989, pp.1102 -1106. CrossRef

[13] Green, W.E., and Oh, P.Y., "Optic-flow-based collision avoidance", Robotics and Automation Magazine, IEEE, Vol. 15.1, 2008, pp. 96-103. CrossRef

[14] Green, W.E and Oh, P.Y. and Barrows, G., "Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments," Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on , Vol.3, pp.2347,2352 Vol.3, 26 April-1 May 2004. CrossRef

[15] Hyslop, Andrew M., and J. Sean Humbert. "Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow." *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 1, 2010, pp. 147-159. CrossRef

[16] Conroy, J., Gremillion, G., Ranganathan, B., and Humbert, J. S., "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation", *Autonomous Robots*, Vol. 27, No. 3, 2009, pp. 189-198. CrossRef

[17] Benavidez, P., and Jamshidi, M., "Mobile robot navigation and target tracking system", *6th International Conference on System of Systems Engineering (SoSE)*, Albuquerque, NM, June 27-30, 2011.

[18] Biswas, J., and Veloso, M., "Depth camera based indoor mobile robot localization and navigation", *IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, May 14-18, 2012.

[19] Correa, D.S.O., Sciotti, D.F., Prado, M.G., Sales, D.O., Wolf, D.F., and Osório, F.S., "Mobile robots navigation in indoor environments using kinect sensor", *Second Brazilian Conference on Critical Embedded Systems (CBSEC)*, Campinas, May 20-25, 2012.

[20] Flacco, F., Kroger, T., De Luca, A., and Khatib, O., "A depth space approach to human-robot collision avoidance", *IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, May 14-18, 2012.

[21] "PrimeSense Supplies 3-D-Sensing Technology to Project Natal for Xbox 360", Microsoft Press Release, retrieved August 31, 2011. VIEW ITEM

[22] URL: openkinect.org/wiki/Imaging_Information retrieved on June 21, 2013.

[23] Abhishek and Setu, S., "Preliminary design of collision avoidance system for micro air vehicle", 7th International conference on Intelligent Unmanned Systems, Chiba, Japan, Oct. 31 – Nov. 2, 2011.

[24] Khoshelham, K. and Elberink, S.O., "Accuracy and resolution of kinect depth data for indoor mapping applications." *Sensors*, Vol. 12.2, 2012, pp. 1437-1454. CrossRef