Vision based Obstacle Avoidance System for Autonomous Aerial Systems

Prof. Kanchan Sarmalkar
Instrumentation Dept.
Vidyavardhini's College of Engineering and Technology,
University of Mumbai, Palghar, India

Sanil Jain
Instrumentation Dept.
Vidyavardhini's College of Engineering
and Technology, University of Mumbai Palghar, India

Dr. Swaroop Hangal
Aerospace Dept.
Indian Institute of Technology, Powai, Mumbai, India

Abstract— **Here we present this topic to explore the idea of how autonomous unmanned aerial vehicles are creating significant transformational revolution with the collision avoidance algorithms. And, also, how we could strategically create more advanced ways to improve and improvise the functioning of sensors used in aerial robotics by replacing them with the cost-efficient camera modules and apply AI or even non-AI enabled computer vision. The primary aim of the project is to look for various systems which could help autonomous drones in detecting obstacles with the help of computer vision, and accordingly letting the UAV to maneuver in order to prevent collision.**

Keywords— UAV, Autonomous, Computer Vision, Aerial Robotics

I. INTRODUCTION

There has certainly been an overlooked revolution which has changed our lives dramatically. With every new invention, a new change is being followed. Inventions have been kicking the notch to an extent of achieving higher ends. And most recently, we witnessed the invention of the Autonomous car. But this was all implemented only in 2-dimension, that is, on the ground. The revolution that needs to be addressed is making use of 3rd-dimension, the aerial dimension, which already has been coming with applications with endless possibilities. Autopilots were another start of the aerial robotics era. Now, with the recent multiple innovations, most notably came the idea of unmanned aerial vehicles. the development of UAVs and micro aerial vehicles boomed in the past decade with various innovative thoughts crossing boundaries of all the applications in the public, civil and military sector. The recent developments added to the field were the making of autonomous drones. So, what we aim in this presented paper is to check and see how many possible ways are there with which autonomous drones would function when dealing with obstacles.

II. AIMS AND OBJECTIVES

The drone is fitted with a GPS and programmed to be able to autonomously move swiftly from one location to another using GPS waypoints in a preplanned path. A significant

consideration is given to safety and ruggedness due to the possibility of collision with a variety of objects. The primary aim of the project is to design a system which will help autonomous drones in detecting obstacles and determining at what distance those obstacles are present with the help of computer vision algorithms, and accordingly prevent a collision.

The goal of the project is to act as a proof of concept for small scale autonomous aerial robotics.

III. LITERATURE REVIEW

1. EXISTING SYSTEM

The presented method relies on visual navigation using the on-board camera of the drone employed in the control feedback with vision playing a key role in maneuvering. A paper presented on obstacle avoidance in a MAV uses 2 LRF modules or any proximity sensors in a setup called Hardware-in loop simulation [1]. This is a setup that replaces simulation models of some flight critical hardware like OBC, servos, and communication devices in the loop with actual hardware. The most important feature in the HILS setup is its capability to perform the Real-Time (RT) simulation.

There exist two main categories of obstacle avoidance algorithms. On one hand there are the local or reactive approaches. Those algorithms do not build a map of the environment or save obstacle positions. They calculate the best reaction from the current sensor data. On the other hand there are the global approaches. Those algorithms have access to a map of the environment which is already precomputed or they build the map themselves while encountering obstacles.

Often path planning algorithms, such as dynamic programming of A* algorithms[2] are used to determine the best path from the given map. And even Dijkstra algorithm[2] that provides the shortest distance between any two nodes with a value above a certain metric value.
Those global algorithms are less prone to get stuck in local minima as they consider all the available information about the terrain. In the case where the whole environment is mapped; they might even yield an optimal solution. However, global methods require a lot of computation power. Goerzen et al. [3] give an overview over the most common obstacle avoidance algorithms, which is focused mostly on global approaches. They further distinguish between algorithms which take kinematic and dynamic constraints of the robotic platform into account and those which do not.

2. PROPOSED SYSTEM

In the proposed method, we aim to use a drone with GPS enabled flight controller and creating waypoints using Q ground Controller, making it to autonomously maneuver through the predetermined path. A camera available on the middleware package with a drone would be used inorder to estimate the distance of any obstacle if any is faced. After that, using the algorithm of 3DVFH and with a few more additions in it, the drone autonomously will be able to create a new and estimated new path to avoid that obstacle and reach the final waypoint.

IV. SCOPE

We are witnessing the advent of a new era of robotic drones that can autonomously fly in natural and man-made environments[4]. Recent developments in the field of artificial intelligence have also pushed the limits of autonomous computer vision-enabled systems that if any obstacle is to be faced in the path, using a camera module and computer vision algorithms based avoidance systems in the GPU machines, we would be using radar and lidar sensor data as ground-truth information. Through this, a DNN is trained to predict the distance to objects and simultaneously the drone can be controlled in order to prevent any collision. In the future, unstructured environments with remote surroundings would be within the reach of Global Positioning Systems, since with emerging technologies in the exploring domain, no place could be claimed as remote in local territory. So, navigation using GPS would be rather more viable.

V. FLOWCHART

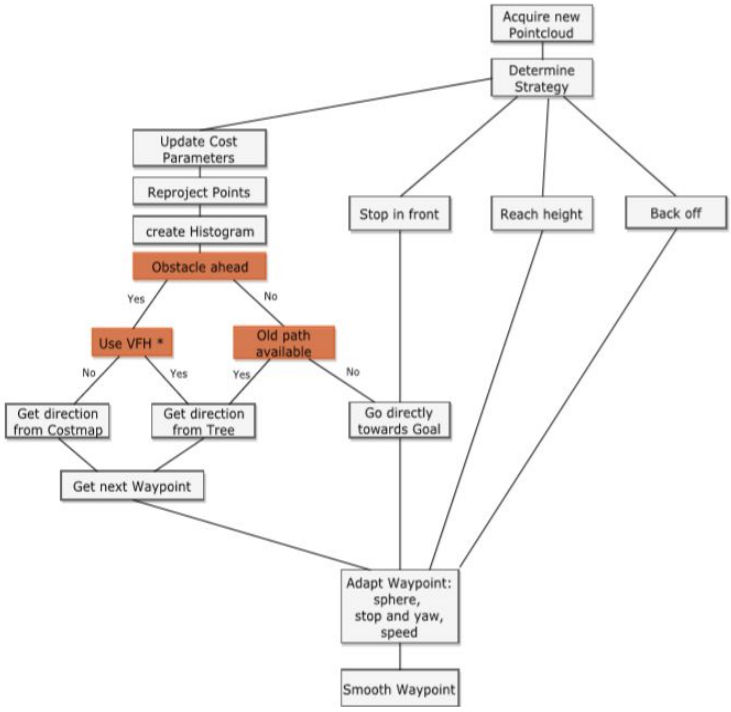

Figure 1: Logic of the algorithm including all the discussed features.

VI. IMPLEMENTATION

Module 1 : Robot Operating System

a. Robot Operating System

Writing software and programming for robots is difficult, particularly as the scale and scope of robotics continue to grow. Different types of robots can have extremely varying hardware, making code reuse nontrivial and difficult to manipulate. Moreover, the sheer size of the required code can be daunting, since it needs to contain a deep stack starting from driver-level software and continuing up across various parameters as described in [5] of perception, abstract reasoning, and beyond. Since the required breadth of expertise is well beyond the capabilities of any single researcher, robotics software architectures must also support large-scale software integration efforts.

For working on drones in Software In The Loop(SITL), the mavros ROS package enables MAVLink extendable communication among computers running MAVLink enabled autopilots and MAVLink enabled GCS with ROS. MAVROS is mainly a supported bridge between ROS and the MAVLink protocol. Currently, it is being extended to enable fast-RTPS messaging, which includes a layer that translates PX4 uORB messages to common ROS idioms.

b. Nomenclature of ROS

The fundamental concepts of the ROS implementation are nodes, messages, topics, and services.

Nodes are processes that perform computation. Nodes communicate with each other by passing messages. A message is a strictly typed data structure. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types and constants.
A node sends a message by publishing it to a given topic, which is simply a string such as "odometry" or "map." A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence.

Module 2 : Gazebo and RVIZ

The design and implementation of new algorithms can be a difficult task that becomes particularly acute with the lack of convenient test environments. In situations such as these, this application with its sensory realism can play a time-saving role. Conventionally, the development of new algorithms either required custom simulators or direct testing on the hardware; Gazebo's realistic environments and simple interface can drastically reduce the turnaround time from a conceptual stage to functional system. The development of Gazebo has been driven by the increasing use of robotic vehicles for outdoor applications [6]. The 3D visualization for ROS applications is done using a tool of RVIZ. It provides a view of your robot model, captures sensor information from robot sensors, and works upon those captured data for further commanding of the robot. The difference between Gazebo and RVIZ is that Gazebo is the actual real-world physics simulator that helps set up a world and simulate a robot moving around. Rviz is the visualization software that allows viewing that gazebo data, if simulating, or real-world data if the gazebo is not used, but a real robot.

Module3 : PX4

ROS (Robot Operating System) is a general-purpose robotics library that can have applications in a way that can be used with PX4 for offboard control. It uses the MAVROS node to communicate with PX4 running on hardware or by using a simulator like Gazebo. PX4 is an open-source flight control software for Unmanned Aerial Vehicles, including drones. PX4 provides optimized APIs and SDKs for developers working with various interfaces and integrations. It is designed to be deeply coupled with embedded computer vision for autonomous capabilities. The framework lowers the barrier of entry for developers working on localization and obstacle detection algorithms.

VII. ALGORITHM

The 3DVFH Algorithm- The obstacle avoidance algorithm implemented is based on the 3DVFH algorithm introduced by Vanneste et al. Vanneste et al. to build a global map of the environment in the form of an Octomap. From this global map, local information in a bounding box around the UAV is extracted to perform the histogram-based obstacle avoidance. Therefore, this obstacle avoidance strategy lies somewhere in between global and local approaches. Having access to a global map has the main advantage that the algorithm remembers previously seen obstacles that might not be in the FOV anymore. But building a global map also introduces a lot of computational overhead. Therefore, this 3DVFH method is implemented as a purely local algorithm without building a global map. The global map is replaced by direct usage of the 3D point-cloud provided by the stereo camera and a computationally less expensive memory strategy is developed to mitigate the inherent issues of a local approach.

From the cropped point-cloud information, a 2D polar histogram gets constructed. For every 3D point in the clipped point-cloud the azimuth and elevation angle with respect to the UAV position are then calculated. The point is basically then placed in the corresponding histogram bin. In the primary polar histogram, each cell will hold the number of 3D points that fall into its sector. In this process of carrying out, the primary polar histogram is not masked for slightest turn radii as the slow flight is inferred and the drone is able to turn on the spot. The primary polar histogram is converted into a binary polar histogram by comparing the point count in each cell with a threshold. To consider the UAV size as well as a minimum distance to the obstacles, histogram cells inside a safety margin around occupied cells are also considered as blocked. The size of this safety margin is dependent on the obstacle distance.

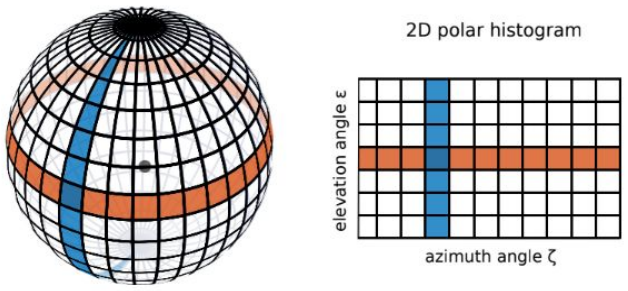

Figure 2: The angles, elevation and azimuth, with respect to the UAV position are calculated for all 3D points. The points are then mapped to the corresponding 2D histogram bin.

For all free cells, cost function is evaluated in the resulting histogram. The cost function contains a goal orientation term and a smoothing term. The goal orientation term compares the evaluated direction to the goal direction. The smoothing term compares the evaluated direction to the direction chosen in the last time-step. The histogram cell with the lowest cost parameter value will then be chosen as the direction for movement.

In case, when no obstacle is present, the histogram will consist of only unoccupied cells and the UAV is allowed to go straight. In this case the waypoint is chosen to lead directly towards the goal using the ground controller

software instead of determining it from the histogram. This reduces the computation time, as there is no demand to evaluate the cost function for every free cell, and it helps avoid the effects of discrete ways of doing it.

The VFH* algorithm- It was introduced by Ulrich and Borenstein for navigation of a ground robot in two dimensions[8]. It allows us to look ahead into the future and evaluate attainable movements not only for the current time-step, but also for the subsequent ones. This algorithm contains approach of combination of both the histogram method and an A* search algorithm.

The 3DVFH* algorithm- uses the new complete point-cloud from the camera. Later, it would update the cost parameters according to the progress made. This enables the drone to rise if no way can be found around the obstacle. Then the old histogram is used to re-project the occupied cells into the 3D space and generate a set of 3D points. The first histogram is calculated as a combination of the re-projected points and the data from the new point-cloud. This histogram is saved as the old histogram for the next time-step. In the 3DVFH algorithm, supposedly calculation of the next waypoint from this histogram is done. But the 3DVFH* algorithm instead builds a search tree to propose different movement possibilities. This tree has its root at the current UAV position. This position is entered in the tree as the first origin and the tree cost function as well as the heuristic for the first node are set. From there, new nodes are entered into the tree structure until a specified number N of nodes has been expanded.

VIII. FORMULATION

The implemented 3DVFH algorithm is purely local and reactive, which means that the algorithm has only access to the data of the current time-step and in proximity to the drone. But, the major drawback of local avoidance strategies is that they do not consider any data or action from previous time-steps. So, this system is built in accordance with the polar histogram principle of the 3DVFH algorithm, which allows us to build a memory for obstacles.

a. Reprojection of Histogram into 3D points

For building a memory using 3DVFH, the memory is converted into a histogram at the current drone location. This can be done by re-projecting the occupied cells in the old histogram into 3D points and with that establishing a memory histogram from those 3D points. The elevation ∈ and azimuth ζ angles of the four corner points can be calculated by adding/subtracting half the angular resolution α of the histogram from the elevation and azimuth angles of the occupied cell as shown in equation :

$$\varepsilon_n + \frac{\alpha}{2}, \text{ where n = 1,2,3,4}$$
$$\zeta_n + \frac{\alpha}{2}, \text{ where n = 1,2,3,4}$$
$$p_{ix} = pos_{old\,x} + d \cdot cos\,(\varepsilon_i * \tfrac{\pi}{180})\, sin(\zeta_i * \tfrac{\pi}{180})$$
$$p_{iy} = pos_{old\,y} + d \cdot cos\,(\varepsilon_i * \tfrac{\pi}{180})\, cos(\zeta_i * \tfrac{\pi}{180})$$
$$p_{iz} = pos_{old\,z} + d \cdot sin\,(\varepsilon_i * \tfrac{\pi}{180})$$

b. Building the histogram from the those points

When the angles of individual 3D points are determined, then they are needed to be converted into histogram indices.

For every 3D point (pi), the elevation angle (ε) and azimuth angle (ζ), observed from the current position (pos) of the drone, is calculated with conversion from negative ranges (-180,180) to positive ranges (0, 360).

$$\zeta = \frac{180}{\pi} \cdot atan2\,(p_x - pos_x,\, p_y - pos_y)$$

$$\varepsilon = \frac{180}{\pi} \cdot atan\,\left(\frac{p\,pz - posz}{(px - pos\,x)^2 + (py - pos\,y)^2}\right)$$

From those positive angles, the histogram indices can be calculated :

$$\beta_{temp} = \beta + (\alpha - \beta\%\alpha)$$
$$\beta_{index} = \frac{\beta_{temp}}{\alpha} - 1$$

This procedure is used twice in the process. First, to get the memory histogram and next, to calculate the histogram from the point-cloud provided by the stereo camera. Then, two of those histograms are combined. This combination is used in navigation.

c. Adaptation of Cost-Parameters

When the histogram is built, all directions which are neither blocked nor inside the safety margin are considered as potentially viable directions. The cost of every direction is estimated and the best direction for movement is chosen. The first criterion of goal orientation is split into three different parts: Yaw difference Δ_{yaw}, pitch difference upwards $\Delta_{pitch\text{-}up}$ and pitch difference downwards $\Delta_{pitch\text{-}down}$.

$$c_{goal} = \Delta_{yaw}(g,\, p) + k_{up} \cdot \Delta_{pitch-up}(g,\, p) + k_{down} \cdot \Delta_{pitch-down}(g,\, p)$$
$$c_{smooth} = \Delta_{yaw}(Pold,\, p) + \Delta_{pitch}(Pold,\, p)$$
$$c_{tot} = k_{goal} \cdot c_{goal} + k_{smooth} \cdot c_{smooth}$$

where, c_{goal} is Goal Cost, c_{smooth} is the Smoothing Cost, g is the goal position, p is the projected potentially viable direction, *Pold* is projected old potential direction k is the weight factor.

The weight factors kgoal and ksmooth define how smooth the path is with respect to the oriented behaviour. If the parameters Kup and Kdown are chosen to be higher, the drone will favor flying around obstacles instead of flying over or underneath. That way, other directions help in navigation for the drone according to the pitch of the parameters.

d. Look Ahead: 3DVFH* with memory

If the old path can be reused, the angle δ to the next node ni can be calculated:

$$cos(\delta) = \frac{L_{nodes}^2 + d - d_i^2 - d_{i-1}^2}{2\, L_{nodes}\, d_i}$$

The distance li can be calculate from the angle δ through:

$$l_i = L_{nodes} * cos(\delta)$$

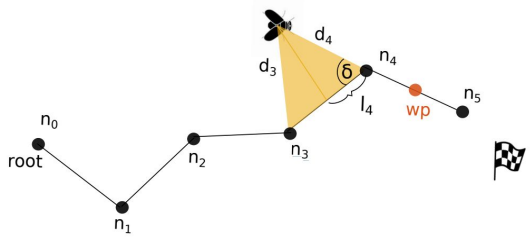

Figure 3: Extraction of the next waypoint from a previously constructed tree path.

From this length the fraction p of the path, which the drone has traveled between node i and i − 1 can be calculated. Using this value the new waypoint can be calculated from the node positions i and i + 1 as described:

$$p = li / L_{nodes}$$
$$wp = (1-p)n_i + pn_i + 1$$

The different cost terms are weighed with factors k to determine the importance of each criterion. The cost function can be written as :

$$c_n = \lambda^{depth\text{-}n} \cdot (k_{target} * c_{target} + k_{yaw} * c_{yaw} + k_{path} * c_{path} + k_{tree} * c_{tree})$$

IX. RESULT

The proposed 3DVFH* obstacle avoidance algorithm is suitable for real time application on UAVs The introduced memory strategy is shown to be effective for outdoor flight as well as complex simulation scenarios. Therefore, in contrast to the 3DVFH algorithm, the 3DVFH* algorithm is not dependent on a global map of the environment which reduces the computational cost of the algorithm. The algorithm combines the ideas behind the 3DVFH and the VFH* algorithm with a novel memory approach. The VFH* algorithm is extended to a 3D environment and the advantages of the look-ahead functionality can be produced again using the 3DVFH* algorithm. Due to this look-ahead capability, the 3DVFH* algorithm did better than the 3DVFH in more complex scenarios.

X. CONCLUSION

The 3DVFH* offers various possibilities for further improvement by consideration of the drone dynamics. Those limitations could be added to the cost function of the look-ahead tree by extending it to include not only positions, but also velocities. The nodes could be interconnected by curves instead of straight lines, such that the maximum velocity for each segment can be calculated.

The 3DVFH* algorithm is a real-time three-dimensional obstacle avoidance algorithm that uses an octomap to determine the obstacles locations. The algorithm can determine the location of these obstacles in real-time because the algorithm will only take obstacles into account that are located close to the drone in an aerial dimension. From the location of the obstacles the algorithm will make a 2D primary polar histogram based on the pose of the robot and location of the obstacles. Next, the algorithm will take the physical capability into account. In this 2D binary polar histogram the algorithm will find multiple paths, give a path weight and determine the path with the lowest path weight. This path will be used to calculate a motion.

XI. REFERENCES

[1] Gade, M. M., Hangal, S., Krishnan, D., & Arya, H. (2016). *Development of Obstacle Avoidance Controller for MAVs: Testing in Hardware-In-Loop Simulation. This work was sponsored by NP-MICAV Project No: DARO/081102072/M/I/CN-04. IFAC-PapersOnLine, 49(1), 413–418.* doi:10.1016/j.ifacol.2016.03.089 K. Elissa.

[2] Elaf Jirjees Dhulkefl, Akif Durdu, "*Path Planning Algorithms for Unmanned Aerial Vehicles*"

[3] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," Journal of Intelligent and Robotic Systems, vol. 57, no. 1-4, pp. 65–100, 2010.

[4] D Floreano ,RJ Wood, "*Science, technology and the future of small autonomous drones"*

[5] Morgan Quigley, Brian Gerkey , Ken Conley , Josh Faust , Tully Foote , Jeremy Leibs , Eric Berger , Rob Wheeler , Andrew Ng,"*ROS: an open-source Robot Operating System* ".

[6] Koenig, N., & Howard, A. (n.d.). Design and use paradigms for gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). doi:10.1109/iros.2004.1389727

[7]] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," Autonomous Robots, vol. 22, no. 2, pp. 101–132, 2007

[8] I. Ulrich and I. Borenstein, "VFH*: Local obstacle avoidance with lookahead verification," IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1572–1577, 2000.