International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 10 Issue 4 ǁ 2022 ǁ PP. 58-74

Simulation of Co-Operative UAV Using Neighborhood Optimization

Mr. G. Pradeep Kumar M.E.., (PhD.,)
Assistant Professor, Department of ECE, Velammal College of Engineering and Technology, Madurai, Tamilnadu

M.Venu Vikaash, B.E.
Department of ECE, Velammal College of Engineering and Technology, Madurai, Tamilnadu

A. Jerome Akash, BE.
Department of ECE, Velammal College of Engineering and Technology, Madurai, Tamilnadu

S. Riyas Ahamed, BE.
Department of ECE, Velammal College of Engineering and Technology, Madurai, Tamilnadu

ABSTRACT
UAVs are going to be assigned different tasks like e.g., rendezvous and space coverage, which require processing and communication capabilities. This work extends the architecture ROS/Gazebo with the likelihood of simulation of co-operative UAVs. We assume UAV with the underlying attitude controller supported the open-source Ardupilot software. the mixture of the coordination algorithm in Gazebo is implemented with software modules extending Ardupilot with the aptitude of sending/receiving messages to/from drones, and executing coordination protocol. As far as it concerns the simulation environment, we have extended the earth in Gazebo to hold quite one drone and to open a particular communication port per drone. within the paper, results on the simulation of a representative coordination algorithm are shown and discussed, during a scenario where a little number of Iris Quadcopters are deployed.

Date of Submission: 02-04-2022 Date of acceptance: 16-04-2022

I. INTRODUCTION

1.1 UAV (UNMANNED AERIAL VEHICLE):

An unmanned aerial vehicle (UAV) (or uncrewed aerial vehicle, commonly called a drone) is an aircraft without a personality's pilot on board. UAVs are a component of an unmanned aircraft system (UAS), which include a UAV, a ground-based controller, and a system of communications between the 2. The flight of UAVs may operate with various degrees of autonomy: either under remote by a personality's operator or autonomously by onboard computers said as an autopilot.

Compared to crewed aircraft, UAVs were originally used for missions too "dull, dirty or dangerous" for humans. While drones originated mostly in military applications, their use is rapidly finding more applications including aerial photography, product deliveries, agriculture, policing and surveillance, infrastructure inspections, science smuggling and drone racing.

Unmanned aerial vehicles (UAV) are a category of aircrafts which will fly without the onboard presence of pilots [WAT 12]. Unmanned aircraft systems incorporates the aircraft component, sensor payloads and a communication system station. they'll be controlled by onboard electronic equipments or via control equipment from the bottom. When it's remotely controlled from ground it's called RPV (Remotely Piloted Vehicle) and requires reliable wireless communication for control. Dedicated control systems is also dedicated to large UAVs, and may be mounted aboard vehicles or in trailers to enable close proximity to UAVs that are limited by range or communication capabilities.

UAVs are used for observation and tactical planning. This technology is now available to be used within the emergency response field to help the crew members. UAVs are classified supported the altitude range, endurance and weight, and support a good range of applications including military and commercial applications. the littlest categories of UAVs are often in the middle of ground-control stations consisting of laptop computers and other components that are sufficiently little to be carried easily with the aircraft in small vehicles, aboard boats or in backpacks. UAVs that are fitted with high precision cameras can navigate round the

area, take pictures and permit the crew members to perform image and structural analysis. As UAV operations require onsite personnel, it'll be helpful for onsite crew members to access the area first before entering the disaster affected area. UAVs that are suitable for outdoor operation and may fly at reasonable altitude are used for disaster impact analysis. The important aspect of such UAVs is that the initial assessment gives a transparent disaster planning direction. After the survivors are detected via image analysis, crew members can then attempt to make contact with the survivors and perform quick rescue operations. Nano UAVs is used in-built and combined with robots capabilities and may be a really useful in detecting structural damages to buildings and detect survivors trapped inside debris.

In recent years, increasing research efforts and developments are improving UAV for various application and reliability. UAV continues to be in experimental stages at the instant. Also, a shortage of skilled onsite crewman may be a bigger problem. [PRA 06] highlights that a minimum of three staff members is required to control a UAV.

Full autonomy is obtainable for specific tasks, like airborne refueling or ground-based battery switching; but higher-level tasks involve greater computing, sensing and actuating capabilities. One approach to quantifying autonomous capabilities is predicated on OODA terminology, as suggested by a 2002 US Air Force research lab, and utilized in the table below:

United States Autonomous control levels chart

Medium levels of autonomy, like reactive autonomy and high levels using cognitive autonomy, have already been achieved to some extent and are very active research fields.

Reactive autonomy, like collective flight, real-time collision avoidance, wall following and corridor centring, relies on telecommunication and situational awareness provided by range sensors: optic flow,[90] lidars (light radars), radars, sonars.

Most range sensors analyze electromagnetic wave, reflected off the environment and coming to the sensor. The cameras (for visual flow) act as simple receivers. Lidars, radars and sonars (with sound mechanical waves) emit and receive waves, measuring the round-trip transit time. UAV cameras don't require emitting power, reducing total consumption. Radars and sonars are mostly used for military applications.

Reactive autonomy has in some forms already reached consumer markets: it should be widely available in but a decade.Cutting-edge (2013) autonomous levels for existing systems Simultaneous localization and mapping

SLAM combines odometry and external data to represent the globe and therefore the position of the UAV in it in three dimensions. High-altitude outdoor navigation doesn't require large vertical fields-of-view and may depend on GPS coordinates (which makes it simple mapping instead of SLAM).

Two related research fields are photogrammetry and LIDAR, especially in low-altitude and indoor 3D environments.

Indoor photogrammetric and stereophotogrammetric SLAM has been demonstrated with quadcopters.

Lidar platforms with heavy, costly and gimbaled traditional laser platforms are proven. Research attempts to handle cost, 2D to 3D expansion, power-to-range ratio, weight and dimensions. LED range-finding applications are commercialized for low-distance sensing capabilities. Research investigates hybridization between light emission and computing power: phased array spatial light modulators, and frequency-modulated-continuous-wave (FMCW) MEMS-tunable vertical-cavity surface-emitting lasers (VCSELs).

Robot swarming refers to networks of agents ready to dynamically reconfigure as elements leave or enter the network. they supply greater flexibility than multi-agent cooperation. Swarming may open the trail to data fusion. Some bio-inspired flight swarms use steering behaviors and flocking.[clarification needed]

Future military potential

In the military sector, American Predators and Reapers are made for counterterrorism operations and in war zones during which the enemy lacks sufficient firepower to shoot them down. they're not designed to face up to antiaircraft defenses or air-to-air combat. In September 2013, the chief of the US Air Combat Command stated that current UAVs were "useless during a contested environment" unless crewed aircraft were there to safeguard them. A 2012 Congressional Research Service (CRS) report speculated that within the future, UAVs is also able to perform tasks beyond intelligence, surveillance, reconnaissance and strikes; the CRS report listed air-to-air combat ("a tougher future task") as possible future undertakings. The Department of Defense's Unmanned Systems Integrated Roadmap FY2013-2038 foresees a more important place for UAVs in combat. Issues include extended capabilities, human-UAV interaction, managing increased information flux, increased autonomy and developing UAV-specific munitions. DARPA's project of systems of systems, or General Atomics work may augur future warfare scenarios, the latter disclosing Avenger swarms equipped with High Energy Liquid Laser Area weapons system (HELLADS).

1.2 ROS (ROBOT OPERATING SYSTEM):

ROS (Robot Operating System) may be a standard actual for robot software development. Analogously, for simulation of UAV aircraft, Gazebo has been widely employed in the scientific community. In Gazebo, Ardupilot is that the open-source software that enables to hold out the control of various unmanned vehicles mainly through its four different components: the Antenna Tracker, the APM Rover, the ArduPlane and also the Arducopter. specifically, Arducopter implements the particular drone control. So far, this ROS/Gazebo architecture only allows for the simulation/emulation of one aircraft.

during this work we present a possible extension of the architecture enabling the simulation of possibly multiple heterogeneous vehicles, adhering to their own individual dynamics, further as interacting with one another per shared co-operation strategies. specifically, we consider UAVs with an underlying attitude controller supported Ardupilot, which uses the MAVLink protocol (Micro Air Vehicle Link) for the communication. the combination of the co-ordination algorithm in Gazebo is implemented with software modules extending Ardupilot with the aptitude of (i) sending/receiving MAVLink messages to/from drones, and (ii) executing the co-ordination protocol. An abstraction of the communication by which drones exchange information is implemented with a co-ordination script, which is executed locally by each drone instance. Every fixed interval a drone sends information (e.g. actual position) to other drones. Each drone uses the information received from the opposite drones via the coordination script to compute a brand new target point, supported the task the drones need to perform. A case study has been developed, where atiny low number of Quadcopters are deployed and perform space-coverage operations by applying a rather modified version of co-ordination algorithm .

Robot software

The definition of the word "robot" tends to differ looking on whom you ask, but the key characteristic of the robots running ROS and being simulated in Gazebo is actuation. So no chatbots or spambots; we're talking about robots that are physically able of interacting with their environments, moving themselves and even other objects. And they're not wind-up toys moving blindly either; they're equipped with sensors that allow them to watch how the planet is changing around them. Tying it all at once, they need logic making sense of those observations to create informed decisions about what movement to form next to finish a particular task. this is often called the sense-think-act cycle, and Dolly's software is organized to reflect these three pieces.

ROS 2 is being developed with the goal of offering a customary software platform to industry and academia that may support them from research and prototyping up to deployment and production. ROS 2 builds on the success of ROS 1, which is already used today in various robotics applications everywhere the planet. a crucial a part of this transition is to take care of the core ROS concepts and tools that have made ROS so successful within the robotics community to this point. one among these well-known concepts is that of a "node," which could be a unit of computation accountable for a awfully specific task. each bit of Dolly's sense-think-act cycle is mapped to a node. The "laser" node senses the globe, the "follow" node processes that data to search out the closest point ahead and generate a command with the direction to maneuver, and therefore the "diff-drive" node moves the wheels as commanded (so called because Dolly may be a differential wheeled robot). As Dolly moves, its laser readings change, and also the cycles starts again. Dolly's software only has three nodes for simplicity, but large robotics applications may have many nodes working together, each answerable for a discrete well-defined task.

The foremost basic method of communication in ROS uses a many-to-many publisher-subscriber mechanism through channels called "topics". The laser node publishes scans on the "scans" topic, which the follow node subscribes to; successively, the follow node publishes movement commands on the "cmds" topic, which the diff-drive node subscribes to. Dolly only uses topics, but additionally to the present one-way sort of communication, ROS also offers a request-response mechanism called "services," yet as "actions," which are used for triggering longer behaviours.

When using these communication patterns, ROS developers tend to use standardized messages whenever possible, which makes it convenient to share nodes among various projects. during this distributed architecture, nodes don't care about which other nodes they're reproof, they only care about which topic, service or action is getting used. this suggests that if in some unspecified time in the future someone decides to get rid of Dolly's wheels and exchange them for propellers to show it into a flying sheep, they won't have to touch the laser or the follow nodes. they're going to only have to swap the diff-drive node for something else that translates the movement commands on the cmds topic in a very way that matches the robot's new body.

The convenience of code reuse is one in every of ROS's greatest strengths because it allows developers to leverage each other's work the maximum amount as possible. By building atop existing software within the ROS ecosystem, developers can target the unique aspects of their particular applications. In fact, implementing Dolly only required writing the follow node, which has but 100 lines of code. The laser and diff-drive nodes are provided by gazebo_ros_pkgs, a regular ROS package that creates the bridge between simulation-specific and non-specific logic. When Dolly is prepared to become a physical robot, those nodes would be substituted by hardware-specific drivers and controllers, but the follow node will be kept the identical because it publishes and

subscribes to plain messages. But as you'll imagine, the follow node isn't the brightest robot logic out there. In fact, Dolly can't tell someone from a tree. during a real application developers would leverage other capabilities provided by the community, like the navigation stack, which might allow Dolly to maneuver autonomously within the world.

All of the communication patterns mentioned above are migrated from ROS 1 to ROS 2 and improved along the way. While ROS 1 uses a custom communication layer, ROS 2 is made on top of DDS. DDS is an industry standard proven in mission-critical applications like aviation and nuclear energy. you'll read more details about the DDS integration during this article on InfoQ.

Additionally to the messaging system, ROS 2 provides powerful developer tools. for instance, RViz may be a visualizer for ROS topics that's invaluable during application development and debugging. it's a 3D scene during which data from any a part of the applying is displayed together, like point clouds and coordinate frames. RViz also provides control interfaces like 3D markers which will be dragged to maneuver a true robot. RQt is another handy graphical tool that lets developers quickly put together widgets to interact with any aspect of their robotics application, be it simulated or not. The image below shows Dolly's laser scans in RViz alongside Gazebo's view showing the visualization within simulation. RViz will display scans the identical way, no matter whether they're simulated or coming from hardware.

ROS, an open-source project, provides a standard framework for robotics applications. ROS is heavily utilized by the research community for service robotics applications, but its technology will be applied to other application areas, including industrial robotics. ROS capabilities, like advanced perception and path/grasp planning, can enable manufacturing robotic applications that were previously technically infeasible or cost prohibitive.

ROS-Industrial

ROS-Industrial is an open-source project that extends the advanced capabilities of ROS to manufacturing automation and robotics. The ROS-Industrial repository includes interfaces for common industrial manipulators, grippers, sensors, and device networks. It also provides software libraries for automatic 2D/3D sensor calibration, process path/motion planning, applications like Scan-N-Plan, developer tools just like the Qt Creator ROS Plugin, and training curriculum that's specific to the wants of manufacturers. ROS-I is supported by a global Consortium of industry and research members.

Provides a one-stop location for manufacturing-related ROS
Software.

Striving towards software robustness and reliability that meets the requirements of business applications.

Combines the relative strengths of ROS and existing technology, combining ROS high-level functionality with the low-level reliability and safety of an industrial robot controller, as opposition replacing anyone technology entirely.

Stimulates the event of hardware-agnostic software by standardizing interfaces.

Provides an "easy" path to use cutting-edge research to industrial applications by employing a common ROS architecture.

Provides simple, easy-to-use, well-documented application programming interfaces.

ROS Industrial Benefits

1.2.1 THE ROS-INDUSTRIAL STACK:

LEVERAGES POWERFUL FUNCTIONALITY WITHIN ROS:

- Custom inverse kinematics for manipulators, including solutions for manipulators with greater than six degrees-of-freedom.
- Advanced 2-D (image) and 3-D (point cloud) perception.
- Rich toolset for development, simulation, and visualization.

ENABLES NEW APPLICATIONS

- Unstructured applications that include advanced perception for identifying robot work pieces as opposed to hard tooling.
- Scan-N-Plan algorithms based upon advanced perception and just-in-time planning as opposed to simply replaying taught paths.
- Model-based approaches that permit automated programming for thousands of unique CAD parts.

SIMPLIFIES ROBOT PROGRAMMING TO THE TASK LEVEL

- Eliminates path planning and teaching. Collision-free, optimal paths are automatically calculated given tool path waypoints.
- Applying abstract programming principles to similar tasks (useful in low-volume applications or with slight variations in work pieces).

1.3 GAZEBO SIMULATOR:

Robot simulation is an important tool in every roboticist's toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the power to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips may be a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. better of all, Gazebo is free with a vibrant community.

Gazebo Simulation

Gazebo (opens new window)is a strong 3D simulation environment for autonomous robots that's particularly suitable for testing object-avoidance and computer vision. This page describes its use with SITL and one vehicle. Gazebo also can be used with HITL and for multi-vehicle simulation.

Running the Simulation

Run a simulation by starting PX4 SITL and gazebo with the airframe configuration to load (multicopters, planes, VTOL, optical flow and multi-vehicle simulations are supported).

the simplest thanks to do that is to open a terminal within the root directory of the PX4 PX4-Autopilot repository and call wreak the required target. as an example, to begin a quadrotor simulation (the default).

The commands above launch one vehicle with the total UI. Other options include:

Starting PX4 and Gazebo separately in order that you'll keep Gazebo running and only re-launch PX4 when needed (quicker than restarting both).

Run the simulation in Headless Mode, which doesn't start the Gazebo UI (this uses fewer resources and is far faster)

Usage/Configuration Options

Headless Mode

Gazebo is run during a headless mode within which the Gazebo UI isn't launched. This starts up more quickly and uses less system resources (i.e. it's a more "lightweight" thanks to run the simulation).

Set Custom Takeoff Location

The takeoff location in SITL Gazebo are often set using environment variables. this may override both the default takeoff location, and any value set for the globe.

Change Simulation Speed

The simulation speed are often increased or decreased with regard to realtime using the environment variable PX4_SIM_SPEED_FACTOR

Change Wind Speed

To simulate wind speed, add this plugin to your world file and replace SET_YOUR_WIND_SPEED with the required speed

Using a Joystick

Joystick and thumb-joystick support are supported through QGroundControl (setup instructions here).

Improving Distance Sensor Performance

this default world is PX4/sitl_gazebo/worlds/iris.world (opens new window)), which uses a heightmap as ground.This can cause difficulty when employing a distance sensor. If there are unexpected results we recommend you modify the model in iris.model from uneven_ground to asphalt_plane

Simulating GPS Noise

Gazebo can simulate GPS noise that's kind of like that typically found in real systems (otherwise reported GPS values are noise-free/perfect). this is often useful when engaged on applications that may be impacted by GPS noise - e.g. precision positioning.

GPS noise is enabled if the target vehicle's SDF file contains a price for the gpsNoise element (i.e. it's the line: true). it's enabled by default in many vehicle SDF files: solo.sdf, iris.sdf, standard_vtol.sdf, delta_wing.sdf, plane.sdf, typhoon_h480, tailsitter.sdf.

To enable/disable GPS noise:

Build any gazebo target so as to come up with SDF files (for all vehicles). Open the SDF file for your target vehicle

Search for the GPS Noise element:

If it's present, GPS is enabled. you'll disable it by deleting the If it's not preset GPS is disabled. you'll enable it by adding the gpsNoise element to the gps_plugin section (as shown above). the following time you build/restart Gazebo it'll use the new GPS noise setting.

Loading a particular World

PX4 supports variety of Gazebo Worlds, which are stored in PX4/sitl_gazebo/worlds (opens new window)) By default Gazebo displays a flat featureless plane, as defined in empty.world (opens new window).

you'll be able to load any of the worlds by specifying them because the final option within the PX4 configuration target.

Set World Location
The vehicle gets spawned very near the origin of the planet model at some simulated GPS location

II. LITERATURE SURVEY

LITERATURE SURVEY 1:

Hoang, V.T., Phung, M.D., Dinh, T.H. and Ha, Q.P., 2018, October. Angle-encoded swarm optimization for uav formation path planning. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5239-5244). IEEE.

This paper presents a completely unique and feasible path planning technique for a bunch of unmanned aerial vehicles (DAVs) conducting surface inspection of infrastructure. the final word goal is to minimise the travel distance of DAVs while simultaneously avoid obstacles, and maintain altitude constraints also because the shape of the UAV formation. A multiple-objective optimisation algorithm, called the Angle-encoded Particle Swarm Optimization (θ- PSO) algorithm, is proposed to accelerate the swarm convergence with angular velocity and position being employed for the situation of particles. the entire formation is modelled as a virtual rigid body and controlled to keep up a desired geometric shape among the paths created while the centroid of the group follows a pre-determined trajectory. supported the testbed of 3DR Solo drones equipped with a proprietary Mission Planner, and therefore the Internet-of- Things (loT) for multi-directional transmission and reception of knowledge between the DAV s, extensive experiments are conducted for triangular formation maintenance along a monorail bridge. The results obtained confirm the feasibility and effectiveness of the proposed approach.

LITERATURE SURVEY 2:

Bernardeschi, C., Fagiolini, A., Palmieri, M., Scrima, G. and Sofia, F., 2018, October. Ros/gazebo based simulation of co-operative uavs. In International Conference on Modelling and Simulation for Autonomous Systesm (pp. 321-334). Springer, Cham.

UAVs is assigned different tasks like e.g., rendezvous and space coverage, which require processing and communication capabilities. This work extends the architecture ROS/Gazebo with thepossibility of simulation of co-operative UAVs. We assume UAV with the underlying attitude controller supported the open-source Ardupilot software. the combination of the co-ordination algorithm in Gazebo is implemented with software modules extending Ardupilot with the potential of sending/receiving messages to/from drones, and executing the co-ordination protocol. As far because it concerns the simulation environment, we've extended the planet in Gazebo to carry quite one drone and to open a particular communication port per drone. within the paper, results on the simulation of a representative co-ordination algorithm are shown and discussed, in a very scenario where atiny low number of Iris Quadcopters are deployed

LITERATURE SURVEY 3:

Pierre, D.M., Zakaria, N. and Pal, A.J., 2011, December. Master-slave parallel vector-evaluated genetic algorithm for unmanned aerial vehicle's path planning. In 2011 11th International Conference on Hybrid Intelligent Systems (HIS) (pp. 517-521). IEEE.

The demand of Unmanned Aerial Vehicle (UAV) to monitor natural disasters extends its use to multiple civil missions. While the utilization of remotely control UAV reduces the human casualties' rates in hazardous environments, it's reported that almost all of UAV accidents are caused by human factor errors. so as to automate UAVs, several approaches to path planning for UAVs, mainly supported Genetic Algorithm (GA), are proposed. However, none of the proposed paradigms optimally solve the trail planning problem with contrasting objectives. We are proposing a Master-Slave Parallel Vector-Evaluated Genetic Algorithm (MSPVEGA) to unravel the trail planning problem. MSPVEGA takes advantage of the advanced computational capabilities to process multiple GAs concurrently. In our present experimental set-up, the MSPVEGA gives optimal results for UAV

LITERATURE SURVEY 4

Schermer, D., Moeini, M. and Wendt, O., 2018. A variable neighborhood search algorithm for solving the vehicle routing problem with drones. In Technical Report. TU Kaiserslautern.

Drones have began to play an increasing role in logistic systems in both, academic research and practical context. specifically, drones have already been applied in various public and personal service sectors including energy, agriculture, and emergency response. Recently, the Vehicle Routing Problem with Drones (VRPD) has been introduced as a variant of the Vehicle Routing Problem. In

the case of the VRPD, a fleet of vehicles, each of them equipped with a collection of drones, is tasked with delivering parcels to customers. the target consists in designing feasible routes with minimal mission time. The drones could also be launched from and retrieved by the vehicles and move at a velocity which may differ from the vehicle's speed. Furthermore, drones possess a limited flight endurance and tiny carrying capacity. The VRPD will be formulated as a Mixed Integer Linear Program (MILP) and, consequently, be solved by any standard MILP solver. With the aim of improving the performance of solvers, we introduce some sets of valid inequalities. Additionally, thanks to limited performance of the solvers in addressing large-scale instances, we address this issue by proposing an algorithm supported the well-known Variable Neighborhood Search (VNS) approach. so as to judge the performance of the introduced algorithm furthermore because the solver in solving the VRPD instances, we applied extensive computational experiments.

LITERATURE SURVEY 5

Tahir, A., Böling, J.M., Haghbayan, M.H. and Plosila, J., 2020. Comparison of linear and nonlinear methods for distributed control of a hierarchical formation of UAVs. IEEE Access, 8, pp.95667-95680.

A key problem in cooperative robotics is that the maintenance of a geometrical configuration during movement. As an answer for this, a multi-layered and distributed system is proposed for the swarm of drones within the formation of hierarchical levels supported the leader-follower approach. The complexity of developing an oversized system is reduced during this way. to confirm the tracking performance and interval of the ensemble system, nonlinear and linear control designs are presented; (a) Sliding Mode Control connected with Proportional-Derivative controller and (b) Linear Quadratic Regular with integral action respectively. The safe travel distance strategy for collision avoidance is introduced and integrated into the control designs for maintaining the hierarchical states within the formation. Both designs provide a rapid adoption with regard to their settling time without introducing oscillations for the dynamic flight movement of vehicles within the cases of (a) nominal, (b) plant-model mismatch, and (c) external disturbance inputs. Also, the nominal settling time of the swarm is improved by 44% on the average when using the nonlinear method as compared to the linear method. Furthermore, the proposed methods are fully distributed in order that each UAV autonomously performs the feedback laws so as to attain better modularity and scalability

LITERATURE SURVEY 6

Yasin, J.N., Haghbayan, M.H., Heikkonen, J., Tenhunen, H. and Plosila, J., 2019, September. Formation maintenance and collision avoidance in a swarm of drones. In Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control (pp. 1-6)..

Distributed formation control and obstacle avoidance are two important challenges in autonomous navigation of a swarm of drones and may negatively affect one another thanks to possible competition that arises between them. In such a platform, a multi-priority control strategy is required to be implemented in every node so as to dynamically optimise the tradeoffs between collision avoidance and formation control w.r.t. given system constraints, e.g. on energy and latency, by reordering priorities in run-time and selecting the appropriate formation and collision avoidance approach supported the state of the swarm, i.e., the kinematic parameters and geographical distribution of the nodes also because the location of the observed obstacles. during this paper, we propose a technique for formation/collision co-awareness with the goal of energy consumption and latency minimisation. The algorithm consists of two partial nested feedback-based control loops and supported three observations: 1) for formation maintenance the relative location of the neighbour nodes; 2) observation of an obstacle by a neighborhood sensor, represented by a boolean value, used for both formation control and collision avoidance; and 3) in critical situations for avoiding collisions, the gap of an obstacle to the node. The obtained comprehensive experimental results show that the proposed approach appropriately keeps the formation during the swarm's travel within the presence of various obstacles.

LITERATURE SURVEY 7

Bürkle, A., Segor, F. and Kollmann, M., 2011. Towards autonomous micro uav swarms. Journal of intelligent & robotic systems, 61(1), pp.339-353.

Micro Unmanned Aerial Vehicles (UAVs) like quadrocopters have gained great popularity over the last years, both as a groundwork platform and in various application fields. However, some complex application scenarios imply the formation of swarms consisting of multiple drones. during this paper a platform for the creation of such swarms is presented. it's supported commercially available quadrocopters enhanced with on-

board processing and communication units enabling full autonomy of individual drones. Furthermore, a generic communication system station is presented that is integration platform. It allows the seamless coordination of various sorts of sensor platforms.

LITERATURE SURVEY 8

Duan, H., Tong, B., Wang, Y. and Wei, C., 2019, July. Mixed game pigeon-inspired optimization for unmanned aircraft system swarm formation. In International Conference on Swarm Intelligence (pp. 429-438). Springer, Cham.

This paper proposes a unique mixed game pigeon-inspired optimization (MGPIO) algorithm for unmanned aircraft system (UAS) swarm formation control. The outer loop controller supported artificial potential field method is intended to rework the UAS swarm formation into abstract movements within the potential field. The inner loop controller supported PIO is meant to unravel the optimal UAS position. a unique pigeon-inspired optimization integrated with mixed theory of games is proposed to reinforce its capacity and convergence speed to resolve complex problem while reducing the computational load. This method maintains the potential of the PIO to diversify the pigeons' exploration within the solution space. Moreover, the proposed method improves the standard of the pigeons supported things. A series of simulation experiments are conducted compared with basic PIO and Particle Swarm Optimization (PSO) approach. The experimental results verify the feasibility and effectiveness of the proposed method.

LITERATURE SURVEY 9

Mirzaeinia, A., Hassanalian, M., Lee, K. and Mirzaeinia, M., 2019. Energy conservation of V-shaped swarming fixed-wing drones through position reconfiguration. Aerospace Science and Technology, 94, p.105398.

There is currently a growing interest within the area of drag reduction of unmanned aerial vehicles. during this paper, the swarming flight of the fixed-wing drones and a load balancing mechanism during the swarm is investigated. As an example, the swarm flight of EBee Sensfly flying wings is analyzed through the proposed methodology. The aerodynamic drag forces of every individual drone and therefore the swarm are modeled theoretically. it's shown that drones through the swarming flight can lay aside to 70% of their energy and consequently improve their performance. As swarming drones have different loads and consume a unique level of energy reckoning on their positions, there's a desire to exchange them during the flight so as to reinforce their efficiency. to the current end, regarding the quantity of drones, a replacement algorithm is defined for them in order that they'll be ready to save more energy during their mission. it's shown that there's quite 21 percent improvement on the wing time and distance of swarming drones after replacement. This method of replacement and formation is considered mutually of the effective factors in an exceedingly drag reduction of swarming aerial vehicles.

LITERATURE SURVEY 10

Kim, H.J. and Ahn, H.S., 2016, December. Realization of swarm formation flying and optimal trajectory generation for multi-drone performance show. In 2016 IEEE/SICE International Symposium on System Integration (SII) (pp. 850-855). IEEE.

in this paper, we introduce the multi-drone platform which is realized by swarm algorithm. one in all the foremost challenging problems when maneuvering multiple drones is guaranteeing collision avoidance. Since our swarm algorithm is predicated on the bogus potential function(APF), collision avoidance between quadrotors is guaranteed. First, the constructive thanks to generate swarm controller are going to be given very well. Then, swarm-based platform may be applied to useful application like visual performance show using multiple drones. The optimal trajectory generation method is additionally provided by using optimization technique.

III. EXISTING METHODOLOGIES

In this section, we describe the current algorithm for a swarm of drones. the strategy is to combine swarm formation control and collision avoidance mechanism to facilitate the strategy of autonomous swarm navigation. To accomplish this, a very unique top-level algorithm is developed, composed of two partial feedback-based algorithms: one for formation control and one for collision avoidance. The feedback for each drone's controller comprises both collision radius and formation distance, and also the goal is to cut back their errors, i.e., differences between the observed values and thus the reference values. The angular error is that the difference of the required angle from the observed angle, indicating what quantity the node should communicate maintain its position w.r.t. its neighbour. Correspondingly, the gap error is that the difference of the measured

distance from the reference distance, indicating what proportion the node should meet with or farther to or from its neighbor.
If there is no feedback for an object detected by the on-board sensor system, indicating there isn't any external object within the vicinity, the algorithm maintains the formation by dynamically checking and adjusting the gap of the drone to its neighbours. The goal is to remain the space greater than the collision radius and shut to the pre-specified formation distance.
Upon detection of an obstacle, the algorithm raises the priority of the collision avoidance part. The collision avoidance part of the algorithm gets the simplest priority once the UAV approaches the minimum safe distance from the obstacle.
After bypassing the obstacle(s), a Failsafe/Fault-Tolerance check is executed to see if the UAV has lost its connection or if it still encompasses a relation to its respective leader.

3.1 OPTIMAL SWARM RECONFIGURATION

After observing an obstacle in its flight path, a UAV has got to maneuver around it in line with rules set by the collision avoidance algorithm. Such maneuvers generally distort the shape of the swarm's formation from the originally planned shape which will, sometimes, be crucial to the success of its mission. it is the intent of our submission to repeatedly guard the collision avoidance maneuvers specified the disturbance from the planned, i.e. optimal, formation is kept at the minimum during the course of the maneuver(s) which, after navigating past the obstacle(s), the swarm is returned back to its initial formation. This process raises a formation construction problem that's widely covered within the literature. However, in our case, the formation algorithm, or in other words the disturbance rejection of a swarm, must be compatible with our obstacle avoidance algorithm whose main target is to reduce the final settling time and energy of the system. it's worth mentioning that we deploy a non-rigid mapping function for efficiency reasons. that's to say that the tactic of returning the swarm formation to its original shape isn't required to re-establish initial neighbouring states among the drones since all the drones are considered to be identical. as an example, within the initial state drone 2 has two neighbours drones 1 and three, after reconstructing the formation its new neighbours could even be drones 4 and 5, it should even become the new global leader. within the subsequent text, we refer to the primary i.e. the desired formation shape because the model formation, while the shape at any instant during the flight is remarked because the scene. within the method of getting back from the scene to the model, there are two main inquiries to be addressed. Firstly, what is the optimal alignment of nodes within the scene to node positions within the model? We name this because the mapping problem. Secondly, what is the optimal trajectory of each node within the scene so as that it's mapped into the required node position within the model. For the first issue, we apply the well-know concept of point set registration , which relies on thin-plate splines formulation (TPS) that's commonly accustomed solve data interpolation and smoothing problems . After determining the mapping strategy, for the second problem, the proposed collision avoidance algorithm utilizes the shortest path scheme for deciding trajectories of individual nodes. Though a more efficient solution for the second part could even be possible, our current focus is on designing an optimal mapping strategy, thus, it suffices to point, here, that search for an efficient trajectory of each node is one avenue for future work
.

3.2 MASTER-SLAVE PARALLEL VECTOR EVALUATED GENETIC ALGORITHM (MSPVEGA)

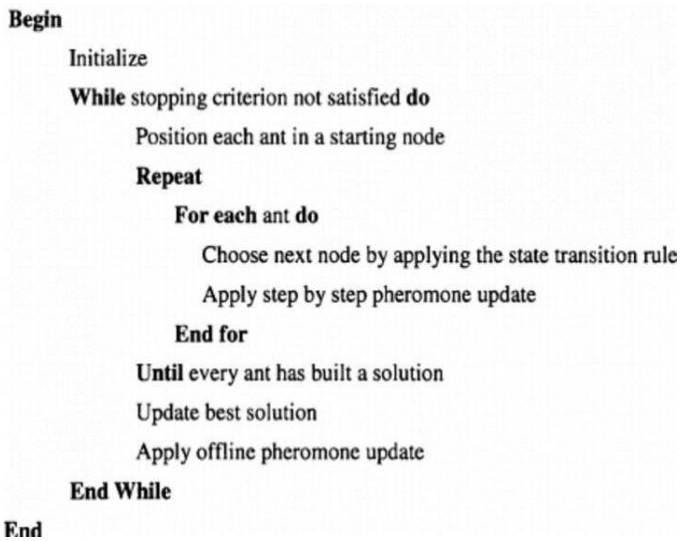
MSPVEGA relies on master-slave strategy. In MSPVEGA, the master coordinates the migration while the slaves run the parallel GAs. All GAs execute concurrently. Each GA operates with its own population which is evaluated with one objective. Fig.1 depicts the building blocks of MSPVEGA

MSPVEGA Building Blocks

After a predefined number of iterations, a specific subset of the local population migrates from the local GA to the master node. The master node reshuffles all the received individuals, segregates them in keeping with the target with which they perform better. Finally, the segregated groups are sent back to the suitable slaves. Within each GA, the fitness function with which each individual is evaluated could be a mathematical representation of a specific objective. Given variety of objectives, α, MSPVEGA dynamically configures b+1 slaves indexed from 0 to b; the slave indexed by b runs a selected GA that mixes all the objectives in Objective Weighting Genetic Algorithm. The fitness function such as such objective.

3.3 ANT COLONY ALGORITHM

Ant colony algorithm is inspired by the method of finding food and transferring it to the nest by ants. Ants secrete pheromone on the way to the food. This pheromone goes up for better paths, though pheromones also have evaporation properties. This method is a population-based met heuristic that can be used to find approximate solutions to difficult optimization problems.
Figure 4: Pseudo code of ACO optimization algorithm.



The choice of a solution component from Ni k is done probabilistically at each construction step. The exact rules for the probabilistic choice of solution components vary across different ACO variants. The best known rule is equation 3.

Selection of the next sensor is done by the roulette wheel method, which is chosen randomly with a uniform distribution from one of the sensors not previously selected. So the next sensors are selected until all the sensors are considered. After repeating the algorithm, the shortest path is found that is the optimal path.

IV. PROPOSED METHODOLOGIES

4.1 VARIABLE NEIGHBORHOOD SEARCH: BASIC CONCEPTS

Variable Neighborhood Search (VNS) could be a metaheuristic framework that was for solving complex optimization problems. The authors propose a scientific change of increasingly distant neighborhoods of the present incumbent solution. The new neighborhoods are then explored by an area search method so as to spot the local optima. a replacement solution is accepted if and on condition that an improvement was made.

Let y be a feasible solution to an optimization problem min f(y). Algorithm 1 details the abstract structure of the fundamental Variable Neighborhood Search (BVNS). More precisely, BVNS consists of several components and requires an initial solution y, moreover as two integer parameters kmax and imax that impose the utmost depth of the neighborhood and maximum number of iterations, respectively. within the following, we offer a brief description of the components.

y' ← Shake(y, k) A shake could be a random move that's wont to generate a replacement solution y0 ∈ Nk(y). We visit Nk(y) because the k-th neighborhood of y. Given an answer y, the neighborhood Nk(y) are often reached from y through exactly k shakes.

y'' ← LocalSearch(y0) a neighborhood search procedure is employed to aim improving the answer y0 that was found in Nk(y) (i.e., the k-th neighborhood of y), which could cause a brand new solution y''

.

y, k ← NeighborhoodChange(y, y00, k) Here, the operator determines if the new solution y'' replaces y because the incumbent solution. Algorithm 2 details the structure of the neighborhood change operator for a minimization problem. That is, if f(y'') < f(y), then y'' will become the new incumbent solution y. Furthermore, we reset the parameter k to If f(y'') ≥ f(y) we increase the worth of the parameter k by 1.

Algorithm 1 BVNS(y, k_{max}, i_{max})

1: $i \leftarrow 1$;
2: **repeat**
3: **repeat**
4: $y^{'} \leftarrow Shake(y, k)$;
5: $y^{''} \leftarrow LocalSearch(y^{'})$;
6: $y, k \leftarrow NeighborhoodChange(y, y^{''}, k)$;
7: **until** $k = k_{max}$
8: **until** $i = i_{max}$
9: **return** y;

Algorithm 2 NeighborhoodChange(y, y^{JJ}, k)

1: **if** $f(y^{JJ}) < f(y)$ **then**
2: $y \leftarrow y^{JJ}$;
3: $k \leftarrow 1$;
4: **else**
5: $k \leftarrow k + 1$;
6: **end if**
7: **return** y, k;

Several extensions to BVNS are proposed and an outline of the proposed methods is given These extensions include Variable Neighborhood Descent (VND) that searches the neighborhoods in a very determenistic way, Reduced VNS (RVNS) that skips the local search procedure, and Skewed VNS (SVNS) that permits for the exploration of solutions that are in distant neighborhoods of the incumbent solution. during this context, Variable Neighborhood Decomposition Search (VNDS) is another extensions of the VNS method that was introduced. It extends the fundamental VNS into a nested two-level VNS scheme, supported a decomposition of the optimization problem. Through our preliminary computational experiments, we were determined that VNDS could be a well-suited approach for solving the VRPD. so as to explain VNDS for the VRPD, we introduce the variables x and y that denote feasible solutions to a VRPD and VRP, respectively.
The VRPD will be embedded as follows into the VNDS framework:
• We try to find an answer y to a VRP, where no drones are used, through VNS. it would be assumed that this decomposed sub problem is less complicated to unravel than the initial VRPD. Furthermore, existing optimization techniques, that are successfully applied to the VRP, may be wont to solve the decomposed sub-problem.
• Once we've got found a VRP solution y, we transform it to a VRPD solution x by inserting drones into the present route of every vehicle k ∈ K.
We design VNDS in a very thanks to to manage the interaction of the sub-problems in an exceedingly way, such an honest heuristic solution will be achieved. Algorithm 3 details the abstract structure of the VNDS approach that has been adapted for the VRPD. The parameter k defines the depth of the neighborhood that may be explored and is initially set to 1. The parameter tmax determines the utmost run time.
The structure of VNDS (Algorithm 3) consists of several components:
• y ← Shake(y(x), k) : A Shake may be a random move that's wont to generate an answer y of a VRP sub-problem that favors characteristics of the answer x of the VRPD problem. • y 0 ← BV NS(y, k) : A BVNS heuristic is employed to get a brand new solution y 0 to the VRP subproblem by improving the answer y (see Algorithm 1). The operators that we use during BVNS are introduced in Section 4.3.
• x 0 ← LocalSearch(y 0 , D) : Once we've found an answer y 0 to the VRP sub-problem, we generate an answer x 0 to the VRPD problem by applying a neighborhood search operator. In our case, the local search operator could be a drone insertion operator that improves a VRP solution by adding feasible drone sorties. The drone insertion operators that we use are introduced thoroughly
. • x, y(x), k ←NeighborhoodChange(x, x0 , y0 , k) : This operator determines if the new solution x 0 replaces x because the incumbent solution. Algorithm 4 details the structure of the neighborhood change operator. If the target value is improved, i.e., f(x 0) < f(x), then x 0 will become the new incumbent solution. Furthermore, we reset the parameter k to 1 and keep track of y 0 . If f(x 0) ≥ f(x) we increment the worth of the parameter k by 1.

Algorithm 3 VNDS(G, K, D, k_{max}, t_{max})

1: **repeat**
2: $k \leftarrow 1$;
3: **repeat**
4: $y \leftarrow$ Shake($y(x)$, k);
5: $y^{J} \leftarrow BV\,NS(y, k)$;
6: $x^{J} \leftarrow LocalSearch(y^{J}, D)$;
7: $x, y(x), k \leftarrow NeighborhoodChange(x, x^{J}, y^{J}, k)$;
8: **until** $k = k_{max}$
9: **until** $t > t_{max}$
10: **return** x;

Algorithm 4 NeighborhoodChange(x, x^{J}, y^{J}, k)

1: **if** $f(x^{J}) < f(x)$ **then**
2: $x \leftarrow x^{J}$;
3: $y(x) \leftarrow y^{J}$;
4: $k \leftarrow 1$;
5: **else**
6: $k \leftarrow k + 1$;
7: **end if**
8: **return** x, k;

4.1.1 Initialization

For generating an initial VRP solution, we use two-phase heuristic, called route-first, cluster second (RFCS) and references therein. The RFCS approach 12 works as follows: First, we build an initial tour using the Nearest-Neighbor-Heuristic (NNH) . More precisely, ranging from the depot, we gradually construct a route by adding the closest unvisited vertex to the tour before returning to the depot. Afterwards, the tour is split equally, supported the amount of accessible vehicles within the set Shake and native Search Operators for BVNS This section is devoted to explaining the operators utilized in the VNDS and BVNS algorithms. within the case of VNDS, the shake operator will take the answer y(x) to the VRP sub-problem that was either generated by RFCS during initialization or stored after performing a part Change Before introducing the BVNS-related operators, we discuss about some classical approaches in solving the VRP. Indeed, the classical heuristics that are utilized in improving VRP solutions may well be classified in two categories

• Single-Route improvement heuristics (also referred to as Intraroute),
• Multi-Route improvement heuristics (also referred to as Interroute).

Single-Route improvements concentrate on improving the routing of one vehicle. Since the general objective consists in minimizing the most recent point, that's determined by the last vehicle to return to the depot, single-route improvements are valuable. Perhaps, the foremost well-known heuristic for improving one route is that the k-opt, which could be a generalization of the 2-opt operator . Multi-Route improvements on the opposite hand try and improve the target function by considering the routing of multiple vehicles at the identical time. The multiroute improvements as follows, all of which might be considered as special cases of a 2-cyclic exchange

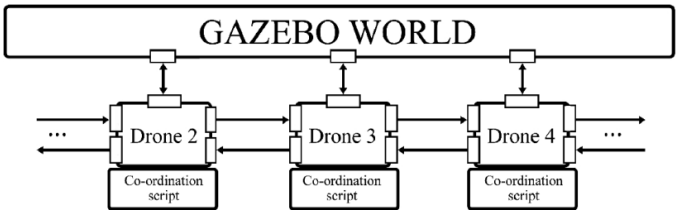
• String Cross (SC), where two edges are exchanged by crossing two edges of two different routes.
• String Relocation (SR), where one vertex is relocated from one route and added to a different route.
• String Exchange (SE), where two vertices are exchanged between two different routes.
• String Mix (SM), which may be a combination of SR and SE that selects the most effective move among the 2.

For applying shake and native search moves during BVNS, we use the 2-opt, SR, and SE moves. particularly, all moves work by generating a random number to work out which vertices (and which routes within the case of SE and SR) may be tormented by a 2-opt, SE, and SR.

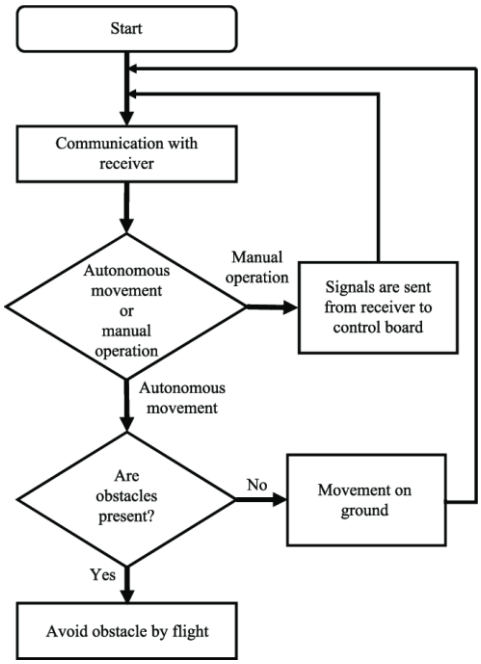
• y' ← Shake(y, k) : within the case of a shake, we take the present routes (belonging to every vehicle in K) and apply a random sequence of k operations (consisting of 2-opt, SR, and SE) to the routes. so as to produce a deep exploration of the answer space and to stir out of local minima, we always accept changes from shake operators whether or not the target value is worse after applying a shake.

• y'' ← LocalSearch(y 0) : within the case of local search, we apply the identical operations (i.e. 2-opt, SR, and SE) to the new routes. However, we apply a greedy search, that consists in applying a move providing it improves the present incumbent objective value.

BLOCK DIAGRAM



FLOW DIAGRAM



V. EXPERIMENATATION AND RESULTS

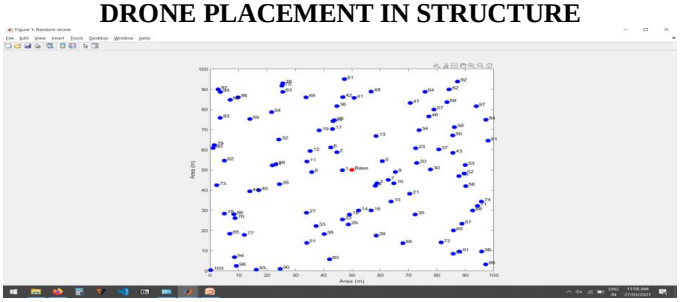
Experimental Parameters

Parameter	Value
Drone Speed 50 m/s	50 m/s
Processing Time 20 seconds	20 seconds
Request Rate	uniform(1, 5) minutes
Battery Life	100 minutes (with no recharging at the station)
Number of clients	5
Number of drones	From 1 to 10
Serviced Area	1000 x 1000m
Max Run Time	Unlimited

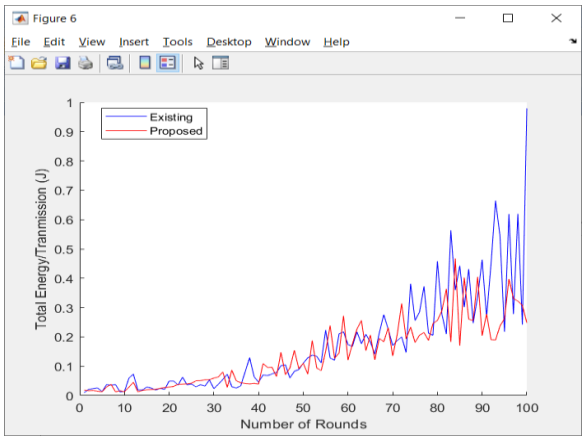
Parameters of QUAD COPTER machine

Parameter	Value	Description
W(m)	0.1050	Width of hub
D(m)	0.1050	Depth of hub
h(m)	0.0390	Height of hub
Q[kg]	0.0400	Mass of motor
P[kg]	0.1000	Mass of hub
R[m]	0.0130	Radius of motor-body
Hm[m]	0.0150	Height of motor-body
R[m]	0.2475	Motor to hub distance
CT	0.5000	Thrust Coefficient
M[kg]	1.0000	Total mass

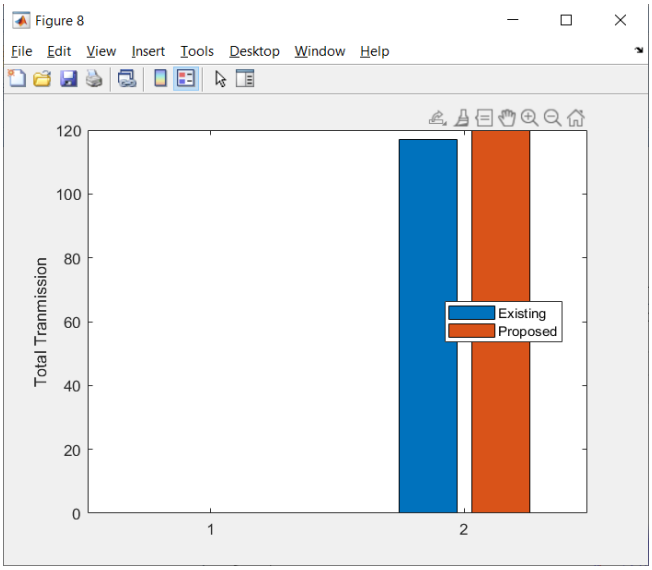
DRONE PLACEMENT IN STRUCTURE



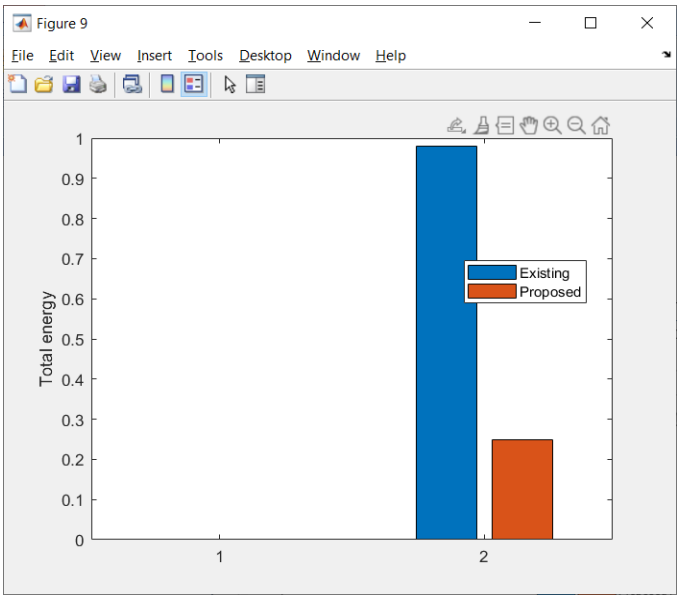
Total Energy Transmission Vs Number Of Rounds



Total Energy Transmission Vs Number Of Rounds

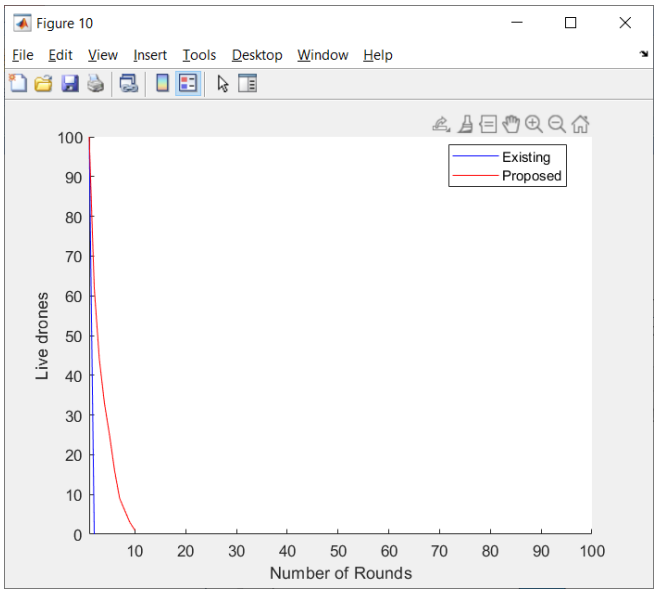


Comparison Results For Existing And Proposed Methods

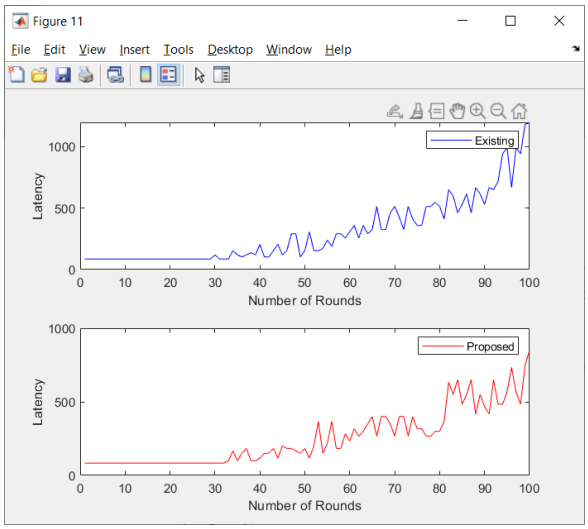


Total Energy Vs Number Of Rounds

Comparison Results For Existing And Proposed Methods
Live Nodes Vs Number Of Rounds

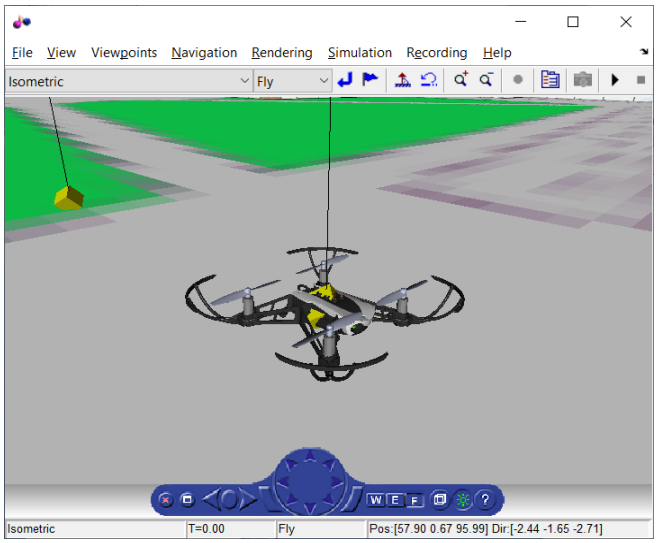


Comparison Results For Existing And Proposed Methods

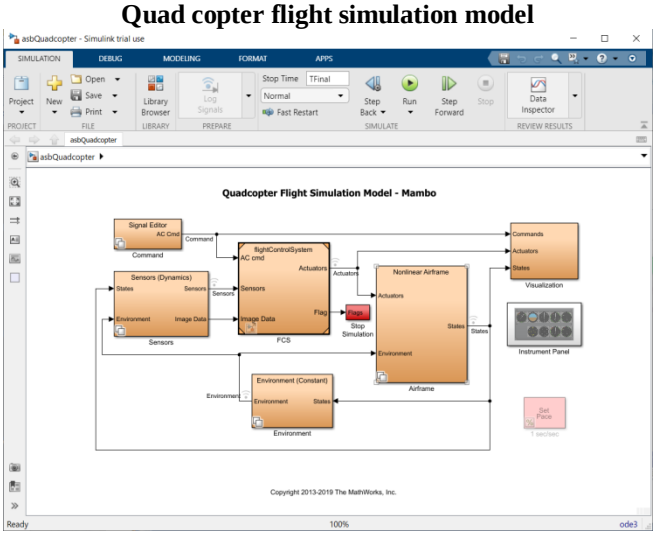


Latency Vs Number Of Rounds

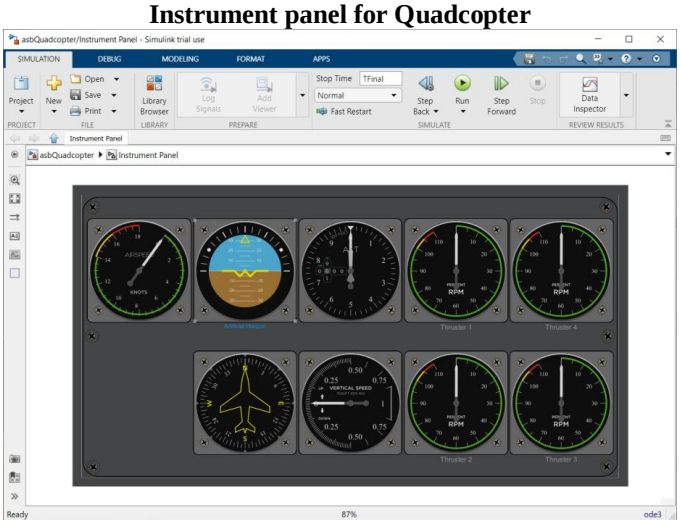
Drone Simulation



Quad copter flight simulation model



Instrument panel for Quadcopter



REFERENCES

[1]. Hoang, V.T., Phung, M.D., Dinh, T.H. and Ha, Q.P., 2018, October. Angle-encoded swarm optimization for uav formation path planning. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5239-5244). IEEE.

[2]. Bernardeschi, C., Fagiolini, A., Palmieri, M., Scrima, G. and Sofia, F., 2018, October. Ros/gazebo based simulation of co-operative uavs. In International Conference on Modelling and Simulation for Autonomous Systesm (pp. 321-334). Springer, Cham.

[3]. Pierre, D.M., Zakaria, N. and Pal, A.J., 2011, December. Master-slave parallel vector-evaluated genetic algorithm for unmanned aerial vehicle's path planning. In 2011 11th International Conference on Hybrid Intelligent Systems (HIS) (pp. 517-521). IEEE.

[4]. Schermer, D., Moeini, M. and Wendt, O., 2018. A variable neighborhood search algorithm for solving the vehicle routing problem with drones. In Technical Report. TU Kaiserslautern.

[5]. Tahir, A., Böling, J.M., Haghbayan, M.H. and Plosila, J., 2020. Comparison of linear and nonlinear methods for distributed control of a hierarchical formation of UAVs. IEEE Access, 8, pp.95667-95680.

[6]. Yasin, J.N., Haghbayan, M.H., Heikkonen, J., Tenhunen, H. and Plosila, J., 2019, September. Formation maintenance and collision avoidance in a swarm of drones. In Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control (pp. 1-6)..

[7]. Bürkle, A., Segor, F. and Kollmann, M., 2011. Towards autonomous micro uav swarms. Journal of intelligent & robotic systems, 61(1), pp.339-353.

[8]. Duan, H., Tong, B., Wang, Y. and Wei, C., 2019, July. Mixed game pigeon-inspired optimization for unmanned aircraft system swarm formation. In International Conference on Swarm Intelligence (pp. 429-438). Springer, Cham.

[9]. Mirzaeinia, A., Hassanalian, M., Lee, K. and Mirzaeinia, M., 2019. Energy conservation of V-shaped swarming fixed-wing drones through position reconfiguration. Aerospace Science and Technology, 94, p.105398.

[10]. Kim, H.J. and Ahn, H.S., 2016, December. Realization of swarm formation flying and optimal trajectory generation for multi-drone performance show. In 2016 IEEE/SICE International Symposium on System Integration (SII) (pp. 850-855). IEEE.

[11]. 3DRobotics: Dronekit-python's documentation (2016).http://python.dronekit.io/

[12]. Adams, S.M., Friedland, C.J.: A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management. In: 9th International Workshop on Remote Sensing for Disaster Response, p. 8 (2011)

[13]. ArduPilot-DevTeam: ArduPilot documentation (2016). http://ardupilot.org/ ardupilot/

[14]. Bernardeschi, C., Domenici, A., Masci, P.: A PVS-simulink integrated environment for model-based analysis of cyber-physical systems. IEEE Trans. Softw. Eng.44(6), 512–533 (2018)

[15]. Chandler, P.R., et al.: Complexity in UAV cooperative control. In: Proceedings of the 2002 American Control Conference (IEEE

Cat. No. CH37301), vol. 3, pp. 1831–1836. IEEE (2002)
[16]. Dronecode-Project: MAVlink developer guide (2018).https://mavlink.io/en/
[17]. Ham, Y., Han, K.K., Lin, J.J., Golparvar-Fard, M.: Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (UAVs): a review of related works. Vis. Eng.4(1) (2016)
[18]. Koenig, N.P., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: IROS, vol. 4, pp. 2149–2154. Citeseer (2004)
[19]. Larsen, P.G., et al.: Integrated tool chain for model-based design of cyber-physical systems: the INTO-CPS project. In: 2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data), pp. 1–6, April 2016
[20]. Lu, P., Geng, Q.: Real-time simulation system for UAV based on Matlab/Simulink. In: 2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering (CCIE), vol. 1, pp. 399–404. IEEE (2011)