Terrestrial Unmanned Roving Vertical Take-off and Landing (TURVTOL)

Bennett Bartel, Ryan Bonk, Terelle Cadd, Chad Hite, Hunter Huth, Songcheng Lin, Stewart Nelson, Jamie O'Brien, Hannah Oliver, Zachary Preston, Nicole Schneider, Catie Spivey, Carson Stebbins, Christopher Titus, and Isaac Vliem *
NASA Langley Research Center, Hampton, Virginia, 23666

As the uses for autonomous rovers and drones has increased with their variability and utility, a fundamental gap has emerged - one which melds multi-modal capabilities into a singular unit. This project aims to fill that gap with the introduction of a multi-modal unmanned vehicle concept that can traverse ground quickly, hop or fly over obstacles and hazardous areas, and operate without the use of GPS or a human in the loop. The 2020 NASA Academy at Langley has developed a first iteration conceptual design of such a vehicle. This design has proven to be feasible through analyses confined to simulated environments.

Nomenclature

$4WD$	=	Four Wheel Drive
$COTS$	=	Commercially-Available Off-The-Shelf
CPU	=	Central Processing Unit
$e_d(t)$	=	Positional error
$e_\omega(t)$	=	Angular error
EDF	=	Electric Ducted Fan
ESC	=	Electronic Speed Controller
g	=	Gravitational Acceleration
$G_c(s)$	=	Controller transfer function
$G_{DD}(s)$	=	Differential drive controller transfer function
$G_M(s)$	=	Motor transfer function
GPS	=	Global Positioning System
GPU	=	Graphics Processing Unit
FEA	=	Finite Element Analysis
FSM	=	Finite State Machine
IMU	=	Inertial Measurement Unit

*NASA Academy Team Research Associates, Aeronautics Research Directorate

k_d = Derivative constant

k_i = Integral constant

k_p = Proportional constant

$MPPT$ = Maximum Power Point Tracking

$MTOW$ = Maximum Takeoff Weight

$N(t)$ = Noise signal

$p_d(t)$ = Desired position

$p(t)$ = Vehicle position

$P(s)$ = S-domain integrator

$\omega_d(t)$ = Desired angular velocity

PID = Proportional-Integral-Derivative

PWM = Pulse-Width Modulation

RC = Radio Control

ROS = Robot Operating System

$SITL$ = Software in the Loop

$\theta_d(t)$ = Desired vehicle angle

$\omega(t)$ = Current vehicle angle

UAV = Unmanned Aerial Vehicle

$v_d(t)$ = Desired velocity

$v(t)$ = Vehicle velocity

I. Introduction

The Terrestrial Unmanned Roving Vertical Take-Off and Landing (TURVTOL) rover is a multi-modal vehicle concept designed by the NASA Academy and sponsored by the NASA Langley Research Center. The following paper features ongoing work for a vehicle design that can determine its position in GPS denied environments and autonomously traverse a path by either ground or air. The paper begins with the hardware design and then moves into the software development. The presented work is a full design of a multi-modal vehicle with simulated measurements for future prototype testing.The paper concludes with future work that needs to be completed followed by the desired next steps of the project.

II. Background

Unmanned, autonomous vehicles are increasingly important in geographical, exploratory, biological and other studies. Autonomous rovers have conducted experiments in diverse and extreme terrains including deserts [1], the arctic

[2], and even other planets. Autonomous robotic systems generally endure harsher conditions than humans, so they fulfill a vital task in scientific research conducted within extreme environments.

Autonomous robotics has come a long way in the past few decades. The speed and efficiency of off-road robotics has increased as a result of improvements in environmental sensing, environmental representation, algorithms in navigation, and processors [3]. Despite these advances, autonomous vehicles can be limited by their mechanical capabilities. Wheel slippage, steep terrain, and other environmental challenges all present difficulties that cannot be solved by intelligent software. Obstacles that cannot be traversed by traditional means require a new, innovative approach to how unmanned vehicles are developed and controlled.

Multi-modal vehicles can operate using two or more modes of travel. In particular, multiple patents have been filed for vehicles that can convert between flight and ground based travel [4–6]. An autonomous, multi-modal vehicle would allow for scientific research in terrain that is not conducive to a traditional ground-based vehicle. However, driving still has the benefit of reducing power consumption. Therefore, a multi-modal vehicle would allow for extended missions in environmentally challenging locations.

There are two categories for a system which allows for driving and flying. The first category involves vehicles that use the same method of propulsion for each mode of travel. This allows for a reduction in weight due to fewer components. The Hybrid Terrestrial and Aerial Quadrotor (HyTAQ) is a quad-rotor mounted inside a cage with a single rotational axis of freedom. The quad-rotor can tilt and produce a forward velocity to roll on the ground. The multi-modal version can travel a distance 4 times greater than an aerial-only version [7]. Alternatively, a vehicle called the Quadroller uses caster wheels attached to the bottom of a quad-rotor to allow for omni-directional ground travel [8].

The other category of multi-modal systems has different propulsion methods for both modalities. Having two different propulsion methods allows for better control and maneuverability. The Flying Sprawl-Tuned Autonomous Robot (FSTAR), for example, uses a sprawling chassis to change the angle of the arms holding the rotors and wheels to switch between flying and driving modalities [9]. The multi-modal vehicle design discussed here falls into this category.

III. Requirements

The vehicle was designed from a set of requirements that determined its functionality. The requirements guided the subsystem design, and more specific criteria were chosen for each component. These will be discussed in later sections. The top level vehicle requirements are shown in Table 1.

IV. Hardware Sub-System Design

In order to arrive at the final design for the first iteration of a vehicle concept, a series of trade studies and brainstorming collaborations were conducted. In this section, initial design ideas are explained, the exploration of important design concepts and examples are covered, and the final design along with its validation is presented. The

Table 1 Top level vehicle requirements.

ID	Requirements
1	The system shall operate autonomously and efficiently maneuver in extreme terrain.
2	The system shall operate on a renewable energy source for the duration of its mission profile.
3	The system shall have land traversing and flight capabilities.
4	The system shall be robust to survive extreme environments: dust, water, wind, low temperatures, etc.
5	The system shall have an operational range greater than that of traditional multicopters.
6	The system shall be adaptable to various missions.
7	The system must be low maintenance to increase survivability.

description of the design process is organized into the three following sections: preliminary design, detailed design, and validation.

A. Preliminary Design

The preliminary design process began with several trade studies in which the critical concepts for the design were considered and the various options were analyzed for feasibility and performance. Following the trade studies, there was a brainstorming process in which various vehicle designs were considered and evaluated. Finally, the final selection is discussed at the end of this section.

1. Trade Studies

Several trade studies were conducted during preliminary design. Trade studies were conducted for flight mode, rotor configuration, motor and rotor selection, locomotion, drivetrain design, and power systems. The trade studies and their results are detailed in the following sections.

i. Flight Mode Selection

The first trade study performed determined the flight mode of the multi-modal vehicle. In this context, a flight mode was a method to generate forces to begin and sustain flight. All conventional flight mechanisms were considered and categorized into four main groups: helicopter, multicopter, lifting surfaces and integrated concepts. Figure 1 depicts the flight mode alternatives considered along with their respective high level sub-categories.

The selection criteria favored highly maneuverable flight and the ability to take off and land from rough terrain. Range and endurance were not heavily weighted due to the mission requiring short bursts of flight to bypass difficult terrain. Therefore, the multicopter flight mode was selected due to its superior maneuverability, reduced energy requirements to support flight, and adaptability to missions. Additionally, ducted fans were found capable of producing higher max thrust with smaller diameter propellers due to the aerodynamics of the duct, but came with the disadvantage of a large amount of energy consumed during the process. It was determined that ducted fans could be considered a plausible concept to integrate with the multicopter flight mode in the event greater amounts of thrust or

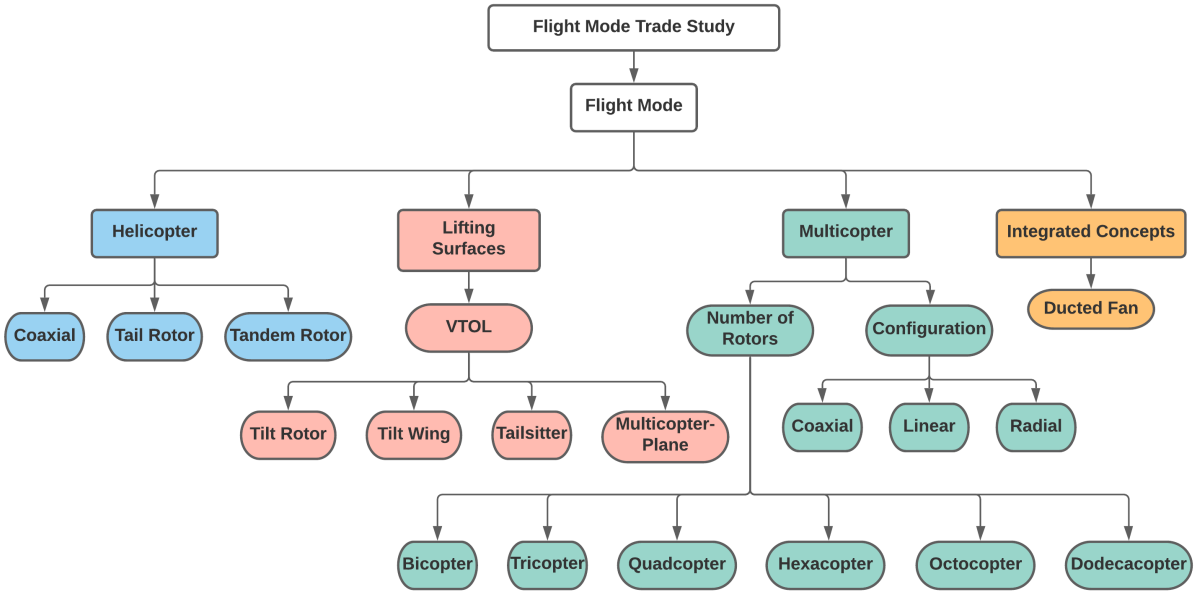


Fig. 1 Flight mode alternatives considered for the initial flight mode trade study.

a smaller vehicle footprint was deemed necessary. As a result, the multicopter and ducted fan flight modes were selected for further research and design application.

ii. Multirotor Configuration

The next step to determining the vehicle design was to choose how many motors the vehicle should employ for flight. The first limiting requirement was the survivability and robustness of the vehicle during mission deployment. It was determined that at least four motors are required to maintain vehicle reliability, as one motor failing on a 2 or 3 motor vehicle configuration would allow for system failure. In addition, the controls of both a tricopter and bicopter add excess complexity to the system with the benefit of increased endurance, which was deemed an undesirable trade off. Therefore, bicopters and tricopters were eliminated. Further research on multicopter control showed that an even number of motors is ideal, as an odd number would create undesirable yawing behavior [10]. A database of off-the-shelf multicopters was created to determine the operating envelope for each motor configuration. Common multicopter types are quadcopters, hexacopters, octocopters, coaxial octocopters and coaxial dodecacopters. This database determined that the maximum number of motors is 12 due to the diminishing returns of lift produced vs. weight, footprint and complexity added. Due to the expected lifting capability required for the vehicle to operate effectively, results of the multirotor configuration trade study determined that hexacopters, octocopters, coaxial octocopters, and coaxial dodecacopters were the best suited configurations.

iii. Motor/Rotor Selection

After the multicopter flight system configuration was chosen, a combination of motors and rotors that best met

the design requirements had to be selected. Due to very limited testing capabilities available during the design process, the team used original equipment manufacturer data alone to make design decisions [11]. A catalog of manufacturer product specification data was compiled that contained performance characteristics of a variety of different motor and rotor combinations at varying throttles percentages. The resulting database contained rotors from 17 to 47 in in diameter and motors with a peak output between 0.6 and 11 kW at 10-90% throttle. Additionally, thrust, power, and range approximations were computed for quadcopter, hexacopter, octocopter, coaxial octocopter, and coaxial dodecacopter rotor configurations using a spreadsheet that took in performance parameters and output vehicle capabilities. A graphical illustration detailing the functionality of the motor and rotor calculator is depicted in Fig.2.

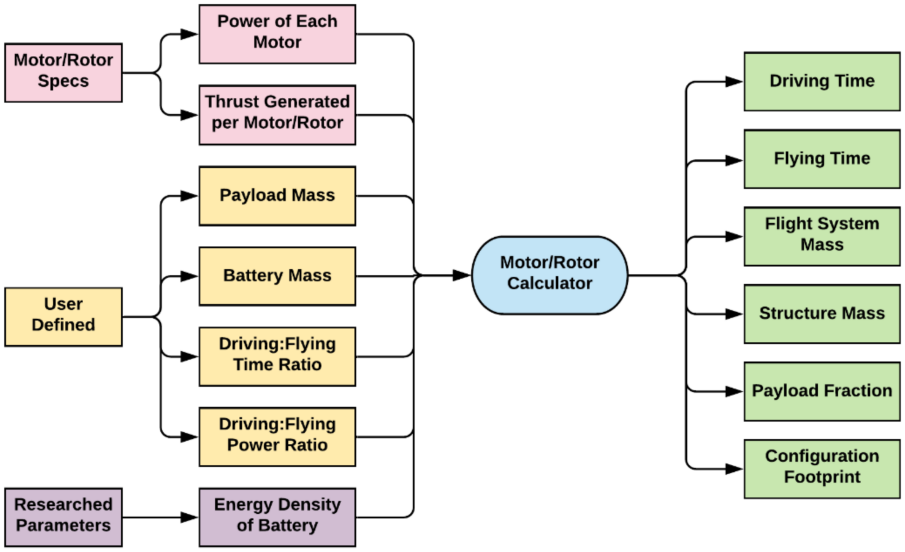


Fig. 2 Graphical representation of spreadsheet calculator used to compare motor/rotor combinations.

Using the results from the spreadsheet, various configurations were ruled out. Both quadcopters and hexacopters were eliminated because the configurations were unable to produce enough thrust to lift the required payload while also allocating a reasonable amount of mass for the ground system. Octocopters and coaxial dodecacopters were capable of producing sufficient thrust. However, the large footprint of these configurations would hinder the ground performance and restrict the maneuverability of the vehicle. A compromise between both of these extremes was the coaxial octocopter which allowed for a reasonably small footprint, and allocated enough mass for the ground system that the made the design feasible. Additionally, the intrinsic value of the vehicle had to be considered. At a maximum takeoff weight (MTOW) below 20 kg, a relatively small payload could be carried which greatly diminished the utility of the vehicle. This proved that a pure multicopter seemed more capable of any mission than a hybrid vehicle due to the fact that a hybrid vehicle has to carry its ground system in flight, which takes up mass that would otherwise be allocated for additional payload. However, as MTOW increases, the vehicle has enough lift to carry larger payloads. Above 40 kg, the hybrid vehicle is estimated to carry reasonable payloads further than many multicopters. The value

of the hybrid vehicle lies in its ability to carry relatively large payloads long distances, and it was determined that a MTOW of at least 40 kg was required to make the hybrid vehicle design worthwhile. The high lift, small footprint, superior maneuverability, robustness, and VTOL capability of the coaxial octocopter led to it being chosen as the final rotor configuration for the vehicle.

iv. Locomotion Trade Study

The team considered using the rover's propellers to traverse the land by means of thrust vectoring methods. However, a qualitative trade study found that depending solely air propelled ground propulsion would demand additional servo motors for pivoting the propellers, which increased system weight and created mission-ending failure points. Track concepts were considered but were rejected due to the excessive amount of moving parts and failure points. Screw propelled vehicles were used in snowy, swampy, and rocky terrain; however, the team decided to move away from this concept as it demanded a great deal of power.A locomotion trade study was done to determine the best mode of ground travel for the vehicle. Other potential types of locomotion solutions included vehicles that utilized wheels, climber, walker, and a rolling cage concepts. These are shown in Fig. 3. Alternative modes were decided against due to complexity in their designs and robustness needed for travel over sub-optimal terrain. A comparison of the pros and cons between tires and tracks is shown in Table 2.

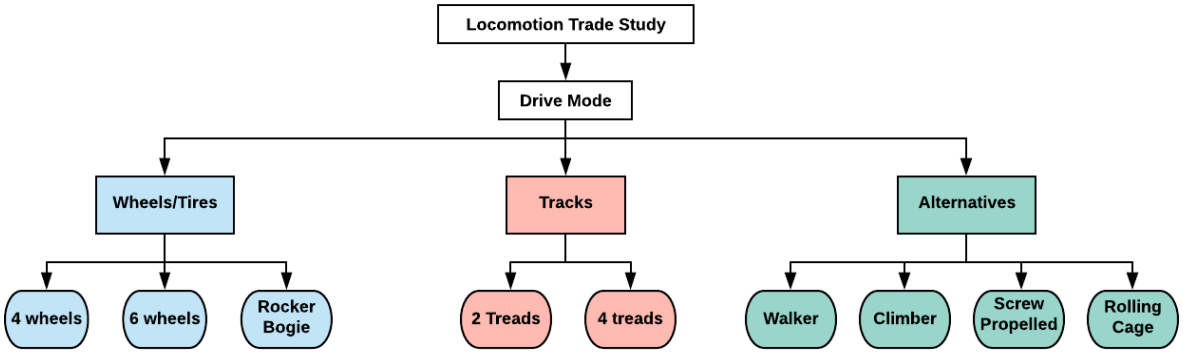


Fig. 3 Ground travel alternatives considered for the locomotion trade study.

v. Drivetrain Design Study

Following the Locomotion Trade Study, methods for integrating the rover's drivetrain were explored. Three options were considered in a comparative analysis: direct drive, two motors to drive the front and back, or a single motor to drive each wheel with a power transmission system. Using simplicity and lack of knowledge about transmissions as a basis in the study, the Academy ultimately settled on the direct drive method for power transmission. This drivetrain was chosen due its simplistic design and required the least amount of components to drive the wheel. Another aspect of the drivetrain addressed in this study was steering. The team decided to stick with skid steering for

Table 2 Pros and cons of tracks vs. tires for ground travel.

	Pros	Cons
Tires	**•Low Production Costs** **•Lower Torque** **•High Maneuverability** **•Lightweight** **•Simplicity** **•Wide Range of Materials**	**•Driving Over Obstacles**
Tracks	**•Power Efficiency**	**•Lower Speed** **•Less Maneuverability** **•Breaks Easily** **•Short Life** **•Difficult to Repair**

the first design iteration, largely influenced by time constraints. However, more steering options are to be explored on the second pass-through of the design in the future.

vi. Design of Solar Power System

One of the unique features and requirements for the rover's design was that it had to be solely powered by a renewable energy source. The team investigated different array chemical composition makeups, types of solar cells, and costs to determine the best fit for our application. Based upon a quantitative analysis between charging time, power output, area, and number of cells required, a silicon based noncrystalline cells were selected. Along with generating power, there must be a means of capturing that energy for storage. A battery management system was selected based upon the voltage and power requirements of the wheel and propeller motor specifications. Since the geared drive motors for ground propulsion required a lower voltage, a converter was used to lower the voltage from 48 V to 24 V. To operate the rover's on board computers, a 48 V to 5 V converter was used. These individual electrical component additions allowed the air motors to utilize 48 V through 100 A rated Electronic Speed Controllers (ESCs).

2. Brainstorming

After initial trade studies were performed by various Academy subteams, the team began designing preliminary integrated rover concepts. After an initial down selection of the preliminary design concepts was completed, the team was left with five concepts to evaluate.The Coaxial Octocopter, Jarvis, Four Wheel Drive (4WD) Wotor, and FSTAR were all completed vehicle concepts. However, the fifth concept included a folding flight system that needed to be developed into a full vehicle concept.

i. Coaxial Octocopter With Overlapping Blades

The Coaxial Octocopter with overlapping blades, shown in Fig. 4, was a concept that featured 8 larger rotors arranged in a coaxial configuration by using two levels of four rotors and motor assemblies. The upper level was

offset by 45° to presumably reduce interference in the inflow of the lower rotors from the upper rotor downwash and improve thrust efficiency. Four wheels would be attached to the aluminum chassis at each corner to allow the rover to drive. Large rotors were required to generate sufficient thrust, which led to a large structure footprint.

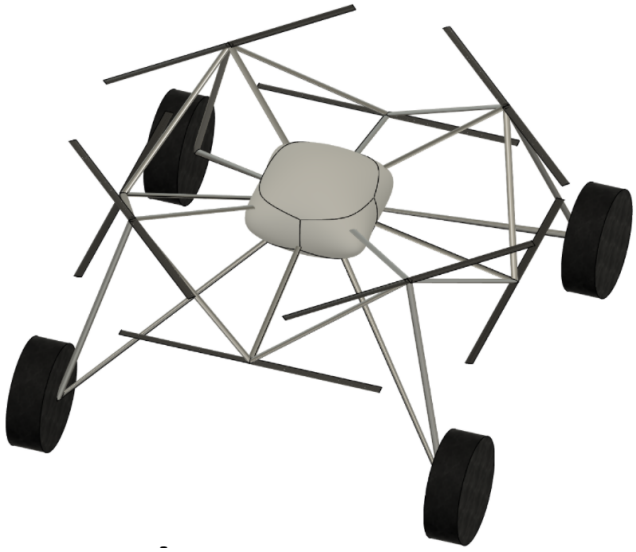


Fig. 4 Model of the coaxial octocopter with overlapping blades preliminary concept.

ii. 4WD Wotor

The 4WD Wotor concept was developed with the purpose of providing a method of protecting the flight system. This coaxial octocopter rover was designed to drive on four wheels and protect two rotors in a coaxial set-up within each wheel. This wheel design, along with added rotor cages, would protect the rotors in the event of a collision during flight or during a hard landing. This also would allow the rover to recover from a crashed position since the rotors would still be able to rotate. Additionally, placing the rotors in the wheels would give the rover a mid to lower center of gravity and provide extra surface area on the top of the rover for solar cells. These characteristics of the 4WD Wotor concept can be seen in Fig. 5.

Fig. 5 Model of the 4WD Wotor preliminary concept.

iii. Jarvis

The Jarvis concept, which is featured in Fig. 6, was developed with the intention of utilizing the high thrust capabilities of an Electric Ducted Fan (EDF). This design incorporates a large EDF through the center of the rover to supply majority of the thrust needed for flight while four smaller rotors or EDFs provide directional control. The system drives using four wheels coupled with a suspension system. For recharging purposes, an innovative solar charging option was designed for the front of the rover to account for the lack of available surface area on the rover's topside. Although the system was estimated to consume a large amount of power, this concept offered a much smaller footprint.

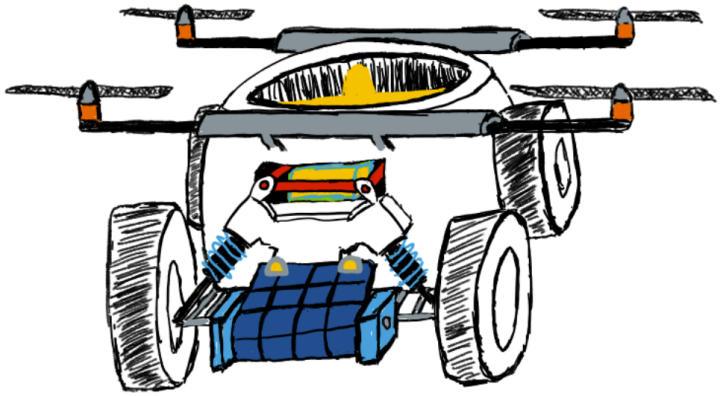


Fig. 6 Sketch of the Jarvis preliminary concept.

iv. Flying STAR

The Flying STAR concept was a design that utilized a fold-able center chassis which allowed for transition between flying and driving modes [9]. By incorporating the rotors into the wheel configuration and reducing motor RPM with a gear reduction wheel assembly, this compact design was attractive by only using a single motor to drive both the rotor and the wheel. This concept had the ability to recover from crashed positions, crawl below low clearances, and transition to flying mode with ease. This design may have been the most flexible and maneuverable that the team considered.The change would have been made to the fold-able chassis so that it would have been allowed to crawl, fly, and drive. The Flying STAR could flip itself out of a crashed position by pushing itself off of the ground. Although the mechanisms of this design were fairly complex, the robustness of the vehicle would allow it to travel across any type of terrain and reliably recover from poor conditions to guarantee the mission's success.

v. Folding Concepts

While driving over rough terrain, two issues arose when integrating a multicopter configuration with a driving system. These were identified as tipping and rotor damage. To prevent these, multiple mechanisms were designed that would allow the rotors to fold or retract. The Umbrella folding concept utilized linear actuators to retract the

rotors in a telescoping manner. The Pancake folding concept consisted of circular cages around the rotors that hinged to fold the rotors down. The Flower Fold concept incorporated rotor arms that hinged at the center to fold the rotors in. Finally, the Butterfly concept featured a pivoting suspension that could fold the rotors in while driving and inversely fold the wheels in when flying. Sketches of the four folding concepts are depicted in Fig. 7.

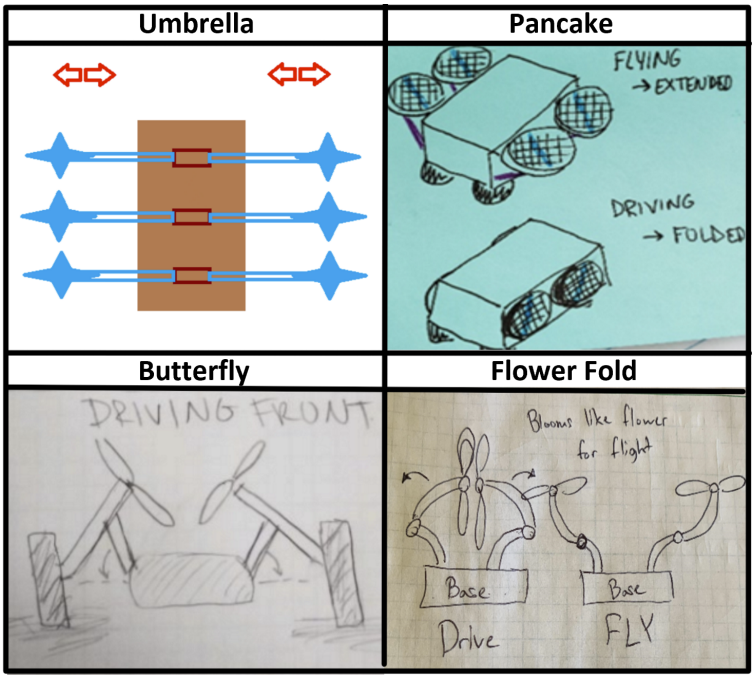


Fig. 7 Sketches of the four preliminary folding concepts, which were named Umbrella (upper left), Pancake (upper right), Butterfly (lower left), and Flower Fold (lower right).

3. Final Selection

After reducing the pool of design concepts to five potential candidates, a comparison between each design was made by specifying commonalities and comparing qualities such as range, operation time, cruise throttle percentage, vehicle mass, payload mass, and other qualities necessary to satisfy recharge ability constraints. Once trade studies were conducted and outcomes were compared, the 4WD Wotor concept was selected as a baseline for the final design. The 4WD Wotor exhibited many appealing qualities that deemed it a worthy finalist. It stood out as the best choice due to its partial rotor protection, large amount of surface area for solar implementation, and its potential to recover from a crashed position. Once the decision was made, design optimization and validation had began by building the rover from the ground up. Using modeling software and testing it within simulated environments, we proved the design to be capable of carrying out our designed mission.

B. Detailed Design

In order to complete the bottom up design for the first iteration of the rover, it was quickly recognized that rotor size would be the first main consideration. With the rotor blades being nested inside of the wheels, the rotor's diameter directly contributed to the size of the wheel configuration needed and influenced the vehicle's overall maximum allowable weight at takeoff. This is where optimization became an important factor. Since MTOW was a function of rotor size, in order to have an effective but efficient vehicle, a balance between structure mass and lifting capability needed to be established.

1. Rotor Configuration

The final configuration was Xoar 24x9 multicopter rotor blades paired with a Xoar TA8225 120 KV motor based on published data from the manufacturer's website [12]. The selection of the rotor blades was driven by the desire to minimize rotor diameter while providing enough lift to meet the payload and range requirements of the mission. As will be discussed in future work, the added mass of a larger wheel was in fact outweighed by a substantially improved maximum thrust as a result of larger rotors. The coaxial rotor spacing was chosen to be a conservative 20% of rotor diameter based on previous research [13, 14]. However, the 20% separation was based on rotors that were only 16 in in diameter, and research concerning 48 in diameter rotors suggests that rotor separation has a minuscule effect on thrust performance [15]. There is a gap in research at the rotor size of 24 in used in this design; however, the 16 in diameter tests were closest to the dimensions of the chosen rotor so the 20% rotor separation was used. This was also useful because it would allow the rotors to be brought closer if need be, and the results from this first iteration would prove as a useful bound for the second iteration sizing. Having designed the wheel, it was learned that the rotor separation should be minimized, as the increase in thrust from having a large rotor separation dramatically increases the mass of the wheel. In the case that more thrust is needed, the rotor should be made bigger instead of separated further.

2. Wheel Configuration

Several wheel configurations were considered for the design. The following sections describe the decisions made regarding rims, drivetrains, and hub plates, as well as the finalized design.

i. Rim

To accommodate the 24 in Xoar rotor blades, the internal diameter of the rim was sized to 29 in, a common bicycle tire diameter. Picking this rim size meant that a commercially available solution could be purchased that is lightweight and structurally sound. This selection also allowed for matching tires to be purchased, resulting in a wheel with an outer diameter of 34.8 in and a width of 2.3 in.

With a Commercially-Available Off-The-Shelf (COTS) solution chosen for the rim and tire, the Academy had to determine how to drive the wheel without interfering with the propellers mounted at the wheel's center. Utilizing an

internal spur gear, the team planned to epoxy this gear directly to the face of the rim's inner circumference. The large contact area between the two surfaces ensured that the aluminum internal rim gear remained fixed to the inner circumference of the composite bicycle rim. This design, pictured in Fig. 8, serves as the final component of the rover's drivetrain.

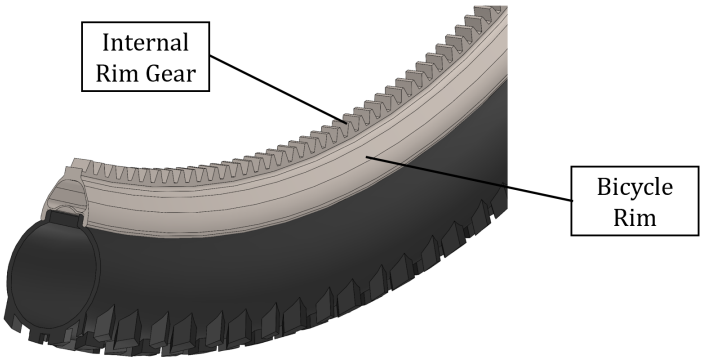


Fig. 8 Section of the rover's wheel, showing the internal rim gear epoxied to the inside of the rim.

ii. Drivetrain

To begin work on the drivetrain, the Academy addressed the following: what rotational speed and torque is required at the wheel, and what type of drivetrain will be implemented? Starting with the outputs of the drivetrain, the team determined that because the wheel's outer diameter was 34.8 in and the vehicle could not navigate autonomously at speeds above 2 m/s, the target rotational speed for the system must be at least 43.2 RPM. Next, the output torque was determined through a series of terramechanics calculations [16]. The resulting torque per wheel was calculated to be 4.23 Nm to traverse most soil types and achieve the maximum velocity from a stopped position within two meters. With the desired outputs known, the team compiled a list of motors, documenting their output speed and torque. A trial-and-error process led to the selection of a 24 V DC motor rated at 3000 RPM and 0.11 Nm of torque. The final task was to create a gear ratio that manipulated the motor's inputs into the desired outputs for the rover.

Through a two-stage series of gear reductions, a compact gear assembly was imagined, comprised of only spurs gears and resulting in an overall gear ratio of 54:1. By adjusting the number of teeth for different gears and maintaining a diametral pitch of 10 $in/teeth$, the gearing was kept within a 9 in region and contained inside a 3D printed custom casing. Shown in Table 3, this gear box could achieve 55.6 RPM and 5.94 Nm of torque, exceeding the target outputs.

Table 3 The drivetrain's target actual vs. the target outputs.

Parameter	Target	Actual
RPM	**43.2**	**55.6**
Torque (N-m)	**4.23**	**5.94**

iii. Hub Plate

To encase the internal rim gear and provide structural support for the wheel, an inner and outer hub plate were designed. Both hub plates attach to the chassis arm to keep the coaxial system stationary. The inner hub plate was designed to be constructed out of 0.5 *in* aluminum core honeycomb panel so it could provide the necessary structural integrity to support the wheel while remaining lightweight [17]. Since the outer hub plate would only experience a fraction of the load, it was designed to be crafted out of 17-gauge 6061 aluminum. The two hub plates were separated by an arrangement of offsetting shafts that incorporated the mating, pinion, and guide gears for the drivetrain. These shafts were held in place by an assembly featuring brass bushings, washers, snap rings, and honeycomb inserts for the inner hub plate. Flex tape was installed across the inner portions of the hub plates to keep outside debris from eroding the drivetrain. Additionally, a thin blade seal made of rubber was designed to seal the gap between the hub plate and the rotating tire by using a low frictional force, and these seals and the hub plates are pictured in Fig. 9 below.

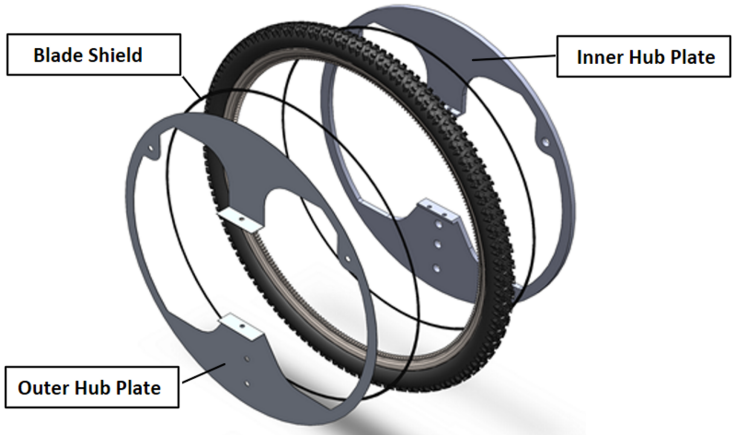


Fig. 9 Hub plates and blade shields that provide support and protection.

iv. Finalized Design

With the rover's wheel, drivetrain, and hub plates designed, a virtual model of the assembly was generated, as shown in Fig. 10. The result was a coaxial rotor assembly nestled between the rim's internal spur gear with hub plates protecting the gearing and providing structural support for the wheel. Additionally, the drivetrain possessed the torque and angular velocity necessary to satisfy the mission requirements and drive the outer rim while the rest of the wheel's components remained stationary. With the wheel fully imagined, the team moved on to the chassis to complete the rover's mechanical design.

3. Chassis

With the integrated coaxial wheel design complete, a chassis was designed to structurally support a quad-wheel configuration. Load cases corresponding to all rotors at maximum thrust, a crash condition of 4.5 *g* on all four wheels,

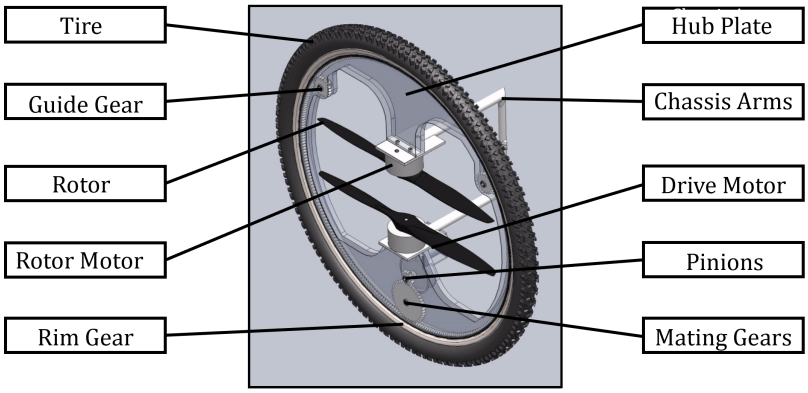


Fig. 10 This schematic documents the main components of the wheel design with transparent hub plates for viewing purposes.

and a crash condition of 4.5 g on all pairs of wheels were defined. It was quickly evident that the limiting case would be the 4.5 g loading on the front and back wheel pairs. This crash condition incorporates a safety factor of 1.5, so the target safety factor of the simulation was 1.0 [18] . Additionally, geometric constraints such as the location of the wheels, the volume the rotors rotate within, and a volume for the payload box were also defined at this point in the design.

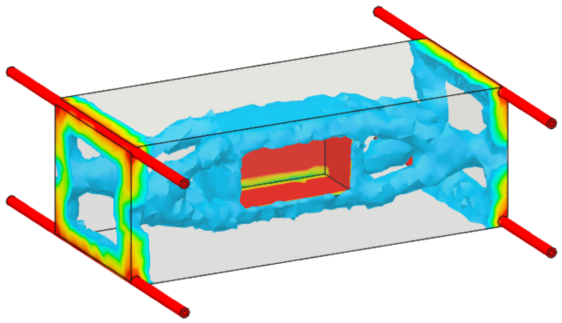


Fig. 11 The shape optimized chassis with defined areas of critical load bearing.

With these constraints in mind, Fusion 360's shape optimization software was used to begin determining the optimal structure for the chassis. The results from the initial study are shown in Fig. 11, where the gray box is the starting volume which was optimized. The colored shape represents the elements in that volume with a high load criticality, essentially a measure of importance of that element to the structural soundness of the part given all of the load conditions. This was the the first iteration outline for the chassis structure.

The next step was to refine the structure and ensure its manufacturability. The arms were replaced with tubes, and gussets were welded to thicken the joints where needed. Through finite element analysis, the design was refined to minimize weight. It was found that a non standard tube thickness was ideal for the tubes, but in order to preserve

manufacturability the thickness was rounded up to the nearest standard size. This resulted in about 2 kg of extra mass and shows that there is still room for improvement. However, due to time constraints of the project, other aspects had to be prioritized, but this will be a focus of further iterations. Another limit of the tubes was that the size was constant along the length of each tube, which led to excess strength along the middle and lack of strength near the joints. This was counteracted by the aforementioned gussets.

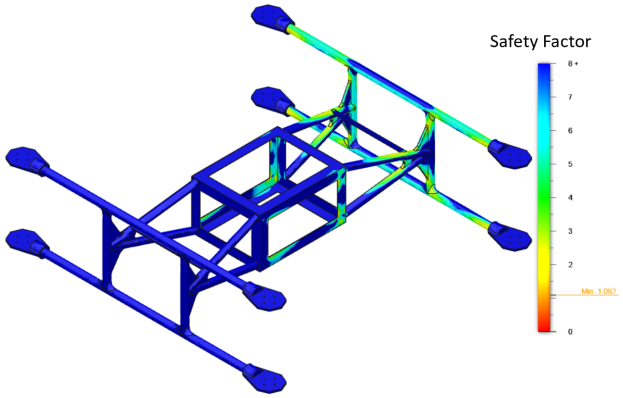


Fig. 12 Finite Element Analysis (FEA) of optimized chassis model.

The final, optimized model resulted in a chassis capable of surviving all of the specified load conditions, and is pictured above in Fig. 12. The results of the final analysis indicate that the chassis survives the 2-wheel hard landing with a 1.09 safety factor, which concludes a sufficiently strong chassis structure was designed, and that it is well optimized.

4. Solar Charging

It is required that the vehicle have a sustainable energy source to recharge itself during missions. Energy sources such as solar, wind, geothermal, radioisotope, and biomass were considered for this application. However, due to weight, time of charge, and vehicle integration, solar charging was selected. The vehicle was equipped with a solar module that included fourteen silicon based mono crystalline cells. The solar cell has an efficiency that is around 24%, which is higher compared to off the shelf solar cells. To optimize the captured solar energy, a solar charger controller, with Maximum Power Point Tracking (MPPT), is added to the system. An MPPT optimizes the conversion of high voltage output from the solar module to a lower voltage that is required to charge the batteries. With the addition of MPPT, the solar module will be able to charge 80% of the batteries in less than 12 hours in an ideal environment.

5. Battery Selection

Once the motors for the flight and ground system, battery requirements could be specified. The vehicle would be flying short hops at low speed, so the average power required was estimated to be 140 W from the manufacturer data. Based on limitations from the autonomous navigation of the vehicle, a speed of 2 m/s was assumed, so in order to

traverse the minimum range of 300 m, the vehicle needed to be capable of sustaining flight for 150 s. An additional 60 s was added to allow adequate time for the autonomous system to takeoff and land. The motors required a voltage of 52 V and demanded a total of 480 A at max throttle, so the battery needed to be capable of discharging at this rate for at least a short burst. Given the power, time, current, and voltage requirements of the system, a minimum battery capacity of 737 Wh was determined. Wrapped into this number are the efficiency losses (5%), safety factor (10%), and usable percentage of battery (80%). From these specifications, it was possible to identify similar batteries and their power density which was used to better refine the mass estimate of the batteries to be 3.57 kg.

6. Complete Model of the TURVTOL Rover

Upon completion of battery selection, the entire rover had been created. A virtual model of the design is shown in Fig. 13. With this first iteration realized, the team brought TURVTOL to life, weighing in at 42.407 kg and satisfying all mission requirements.

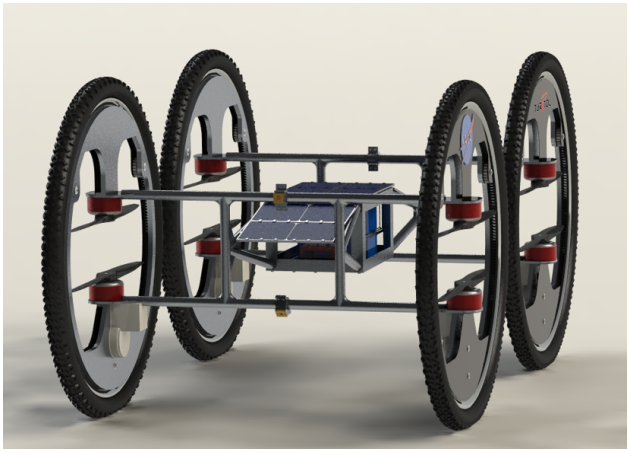


Fig. 13 Model of the first iteration of TURVTOL.

C. Validation

X-Plane 11 pro was used to provide a platform to test the stability and airworthiness of the rover despite the absence of a physical model. The data pulled from these simulations, company-provided spec sheets, and in-house thrust stand tests were combined to determine the maximum amount of lift that various rotor configurations could produce. These results were then analyzed to establish a weight budget which acted as a critical parameter that drove the structural aspects of the entire vehicle.

1. Thrust Testing

Small-scale thrust testing was performed in order to validate propeller efficiency assumptions. The testing components used were two RAYCorp 5x3 two-blade bullnose propeller pair and two Multistar Elite 2240 brushless

motors. The motors counter-rotated and were mounted to an adjustable base which allowed a maximum and minimum separation of the motors of 60% and 10% times the propeller diameter, respectively. This base was then mounted to an L-shaped arm which would rotate as a result of the thrust moment produced by the motors. The resulting force of this moment was measured by a tared food scale with and accuracy of $\pm 1g$. The motor throttle was varied using an RC transmitter and receiver to send PWM signals to the motors. These signals were also sent to a flight controller connected via telemetry to a computer running ground station software to log data. As the throttle was increased incrementally, the values for throttle PWM, battery voltage and current were measured and logged at each thrust test point. Three types of thrust tests were performed, all shown in Fig. 14.

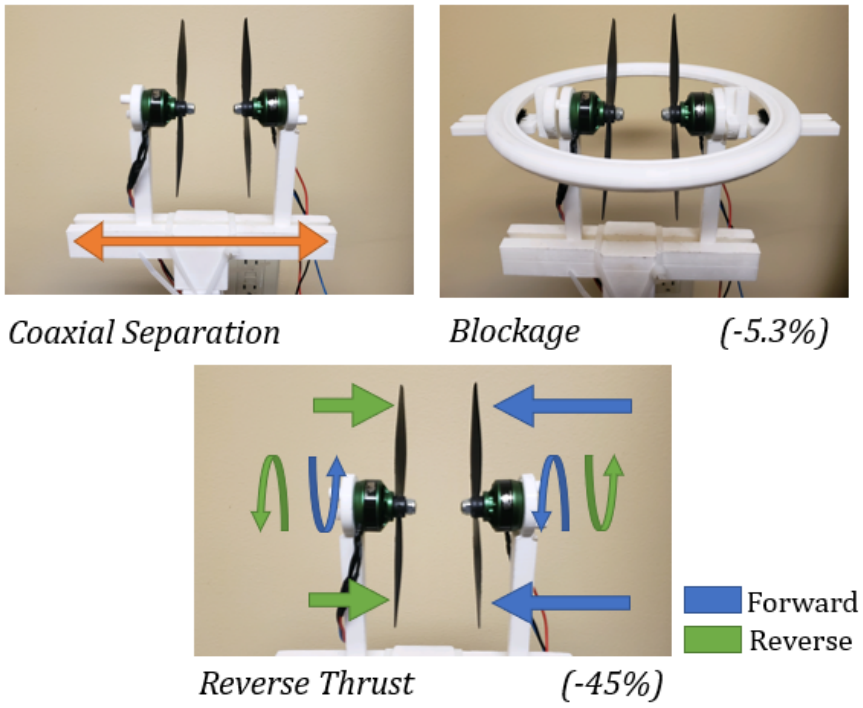


Fig. 14 Depiction of the thrust tests that were performed.

The coaxial separation test varied the spacing between the propellers to determine the efficiency of coaxial rotors at each separation value. The experimental data was compared against the theoretical data of no coaxial losses to determine the value of efficiency. This theoretical data was found measuring the thrust of one motor with no obstructions and multiplying the thrust by 2 to simulate a second idealized motor contribution. It was determined that the ideal separation was 30%, but that the separation would not scale to TURVTOL since coaxial efficiencies do not scale linearly with propeller dimensions as previously discussed in rotor configuration section. The blockage test determined the experimental effects of the tire on the total thrust. This experiment used a 3-D printed tire geometry scaled to the experimental propeller diameter. The experimental data was compared against the theoretical data of the same propeller

separation with no blockage to determine the total losses due to the tire geometry. It was determined that the thrust losses due to the tire geometry was 5.3% of the total thrust. The reverse thrust test determined the experimental effects of reversing the polarity of the motors in a coaxial set-up without also reversing the motors. The purpose of this test was to verify if TURVTOL would be capable of producing enough thrust to flip over if the vehicle found itself upside down. The experimental data was compared against the theoretical data of the same propeller separation with no polarity reversal to determine the total losses due to the motor reversal. It was determined that losses due to the revered thrust was 45%.

2. X-Plane

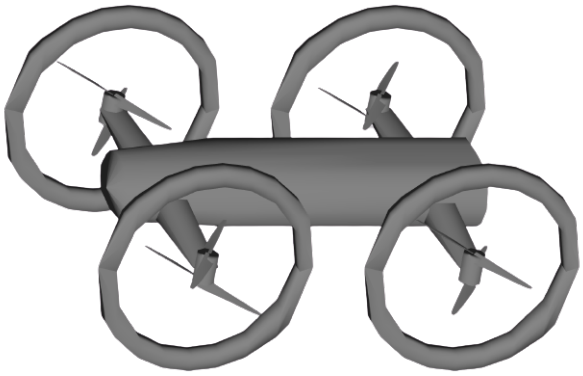


Fig. 15 Picture of the X-Plane model used for flight testing.

An X-Plane model was created to allow aerodynamic testing in parallel with the structural testing being performed by the mechanical subteam. Figure 15 shows the model used for testing. The model was created before the actual design was finalized, and because of this the its fuselage and chassis differs from that of the final design. However, the rotor diameter and performance, wheel size, and overall dimensions of the vehicles match. The detail represented in the X-Plane model was limited by the capabilities of the editor, specifically the complexity of the geometry and the number of parts which the simulator will compute the physics for. While creating a more accurate model is possible by leveraging lower level changes in the code, this is beyond the scope of this project and not the primary focus. The model created allowed for a reasonable estimate of the flight performance of the vehicle, and it allowed a degree of validation of the design's flight-worthiness.

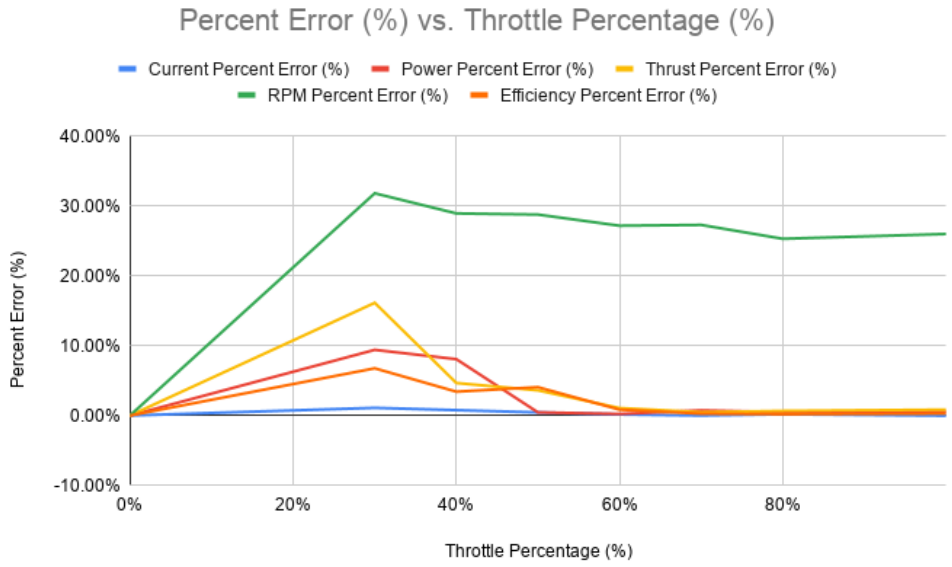


Fig. 16 Comparison of manufacturer data to parameters pulled from the simulated X-Plane model during flight tests and demonstrations.

The most crucial part of the X-Plane model was matching the motor/rotor performance to the manufacturer data. This was a trial and error process focused primarily on matching the thrust and power outputs to the current draw from the battery. The geometry of the model's rotor blades and the performance characteristics were tweaked until the thrust and power at a given amperage matched the manufacturer data. Figure 16 shows the final results of the model's rotors and motors. The average thrust, power, and current values differed by less than 5% from the manufacturer data with even closer agreement at the higher throttle percentages where the vehicle operates. It is worth noting that there are larger disagreements at lower throttles, but these can be neglected because the vehicle is not capable of flight for these throttle percentages. Additionally, RPM vastly differs, but this was unavoidable and the least important of the values to match with manufacturer data. The wheel geometry in the X-Plane model lacks the hub plates in the final model which will deswirl the rotor wake. This causes the model to exhibit better yaw performance than it would in reality. X-Plane is a program designed primarily for larger planes, not relatively small multicopters, and it can only approximate these characteristics to begin with. Because of this fact, the flight tests discussed later will unavoidably differ from reality. The majority of the aerodynamic testing took place within the virtual environment of X-Plane, and simulated results could only be matched to manufacturer specifications. Further verifying the simulated results to real world data will be a primary goal of future work and prototypes.

V. Software Sub-System Design

The following sections discuss and explain the ideas and motivations behind key design choices for software, computer hardware, and autonomous path planning. The software will be implemented using the open-source Robot Operating System (ROS) [19], and it is expected to be able to operate without a human in the loop.

A. On-Board Computer

The vehicle will have a single on-board computer responsible for processing environmental inputs and performing intelligent navigation/control. The software will be built using ROS as a framework for highly parallel computation [19]. The computer will be running programs for sensor management, localization, environmental mapping, path-planning, vehicle controllers, and state determination in real time. The algorithms used in autonomous navigation are limited by the computational power of the on-board computer. Therefore, a powerful on-board computer is critical.

For this design, there is special consideration for an on-board computer with a powerful graphics processing unit (GPU). A GPU is a programmable processor that is optimized for repetitive and highly parallel operations. [20]. GPU acceleration has shown significant reduction in computation time for image processing mapping and path planning for robotics [21–23].

1. Evaluation Criteria

Due to the critical importance of the on-board computer, the following list of criteria were used to evaluate commercially available computers.

1) **Central Processing Unit (CPU) speed:** Emphasis is placed on the CPU because the autonomous nature of the project requires environment and location data to be processed in as near real time as possible. Given this, a powerful CPU is needed to allow the rover to make decisions as fast as possible. The computer will be running multiple programs simultaneously, so a multi-core CPU is required.
2) **Graphical Processing Unit (GPU) speed:** In addition to an efficient CPU, another major consideration is a powerful GPU. In order to navigate a complex environment autonomously, image processing capability is necessary. To process these images efficiently and without overtaxing the CPU, an on-board computer with GPU acceleration will be chosen. This would prove useful in aiding the CPU in processing images in real time. In addition, GPU acceleration could be used in computationally intensive areas of SLAM and path-planning.
3) **Power efficiency:** The rover will be powered from a solar powered battery. The battery needs to have adequate power for the motors, so it is important that the on-board computer consumes minimal power.
4) **Random Access Memory (RAM) available:** Navigation can require large amounts of accessible memory for holding images and large maps produced from SLAM. A large amount of RAM will allow for greater computational efficiency during large memory accesses

5) **Dimensional Constraints:** The computer must be able to fit in the slender design of the rover. Single board computers are desirable for their form-factor designs.
6) **Input and Output (I/O) Connections:** The navigation system will use a variety of sensors, so the on-board computer has to be capable of handling multiple, simultaneous I/O operations. There must be enough physical connections to handle the sensors, and the connections must have the bandwidth to handle simultaneous data transfers from multiple cameras.
7) **Operating Temperature Range:** Due to the adverse conditions that the rover is expected to perform in, a computer capable of handling temperature extremes is required. The computer, with the aid of thermal insulation, needs to continue operation even in the most adverse of conditions.
8) **External Memory Available:** In addition to RAM, the computer needs to have sufficient external memory available, whether that be an on-board storage drive, or an I/O port that allows for an external storage extension.

2. Final Decision

The Jetson Xavier NX (Jetson) [24] was chosen as the on-board computer for the vehicle. The Jetson has a 6 core CPU pared with a 6 MB L2 cache and a 4 MB L3 cache for running multiple programs in parallel. The Jetson has 8 GB of LPDDR4 memory, so it can hold/transfer images and large data sets at a reduced power consumption. This computer has an NVIDA GPU with 384 CUDA cores and 48 tensor cores to aid in image processing and accelerating navigation algorithms. The Jetson also packs this into a small 103 mm x 90.5 mm x 34 mm package. Furthermore, The power consumption of the Jetson can be changed to meet the required computational speed.

B. SLAM Algorithm: VINS-Fusion with OctoMap

Simultaneous localization and mapping (SLAM) provides two major functions for an autonomous navigation system. It builds a map of the surrounding environment and determines the vehicle's location within this map. This is critical for avoiding collisions with obstacles or general rough terrain during navigation. Path-planning is dependent on accurate outputs from SLAM.

SLAM is useful in robotics, specifically robotics which involve any kind of terrain navigation. SLAM at its very core is built to help robots map and self localize in a space using various sensory inputs. Sometimes GPS input is either unreliable or simply not available. In these cases, it does not bode well to have a robot navigate a complex environment on its own. This is where SLAM comes in. SLAM algorithms use sensory inputs from multiple on-board sensors including some form of visual/spatial mapping sensor, such as a camera or LiDAR. The algorithm also commonly includes inertial information from an IMU or equivalent sensor.

Visual odometry tracks features between sequential images from a camera to measure the movement of the camera between frames [25]. In visual inertial odometry (VIO), the slow frame rate of a camera (30 Hz) can be aided by adding

an inertial measurement unit (IMU) to estimate movement between frames [26]. An IMU can only provide relative movement and is subject to an accumulating drift from true position. The camera can correct for this drift for better localization. Often times, wheel odometry is included in the mix to help with the localization. All of these sensory inputs combine and not only allow a robot to build a map of its environment, but also allow a robot to localize itself in the environment. This is incredibly helpful when it comes to building path planning algorithms, as we will discuss in a later section.

A large challenge for SLAM with a multi-modal vehicle is using a sensor suite that will be effective in both modes of travel. For instance, wheel encoders will have no purpose during flight. Somehow, the SLAM algorithm would need to switch sensory inputs from one suite to another in order to make sure it still has the data it needs to fly. This poses a number of problems. Not only is it complicated to code such a change, but by doing so, one is introducing a great deal of ambiguity around the measurements the SLAM algorithm receives during the movements. No matter how short the time interval during which the switch happens, if the robot is moving, some amount of error will be introduced into the localization. Furthermore, different sensor fusion algorithms could be effective during the two different modes of travel.

VINS-Fusion is an algorithm designed by the Hong Kong University Aerial Robotics Group, and it is an extension from the previous VINS-Mono [27]. The algorithm is a visual-inertial odometry (VIO) platform that fuses information from cameras and an IMU for localization. It is capable of online temporal and spatial calibration between the cameras and IMU and has been effective in aerial and ground robotics [28–30].

The other component of SLAM is building a map of the surrounding environment. 3D sensors typically output point cloud data that must be converted to a form the path-planner can use for obstacle avoidance. Two common methods are surface reconstruction and volume estimation. Surface reconstruction will use triangulation to create meshes and estimate surfaces [31]. Volume estimation will segment a point cloud into voxels that occupy a volume in 3D space [32].

An OctoMap is an octree-based voxelization of a point cloud that creates a probabilistic representation of free or occupied 3D space [33]. The 3D environment is divided into nodes of determined resolution, and each nodes occupancy will be determined from point cloud data from a sensor. These nodes are what will be searched during path-planning. OctoMap is an open-source software tool.

1. Design Criteria

Given the constraints that the SLAM Algorithm needed to operate under, the following criteria were considered before coming to any final conclusions as to which algorithm would be most effective for the rover project:

1) **Robustness:** The planned mission sets the minimum standard for SLAM robustness. Uneven, jagged, or generally misshapen terrain is likely to cause occasional errors in the sensor readings. The SLAM algorithm needs to be aware of these errors and adjust accordingly to avoid improper path planning due to a poorly constructed map. Since the vehicle does not have access to other sources for localization (GPS, compass, etc.), it

needs to be proactive with the sensors it has.

2) **Efficiency:** The SLAM algorithm needs to be as efficient as possible while localizing the rover. Essentially, the algorithm cannot overtax the main computer's resources. Additionally, the algorithm needs to be optimized for as close to real time mapping as possible so that when the rover is actively moving, it can make well founded decisions about where to continue. Another reason for efficiency is because of the flight aspect of the rover. While driving on the ground similarly requires real time mapping, the consequences of short delays are not as all-encompassing as the consequences of delays during flight.
3) **Memory Usage:** While the concept of an incredibly detailed and comprehensive mapping of the rover's environment is enticing, it was impractical, especially given the limitations of the on-board hardware. The SLAM algorithm is not supposed to take up an overly large amount of memory to allow the computer to run other processes. Otherwise, the computer would be overtaxed, leading to bottlenecks elsewhere in the operating system.

2. Final Design

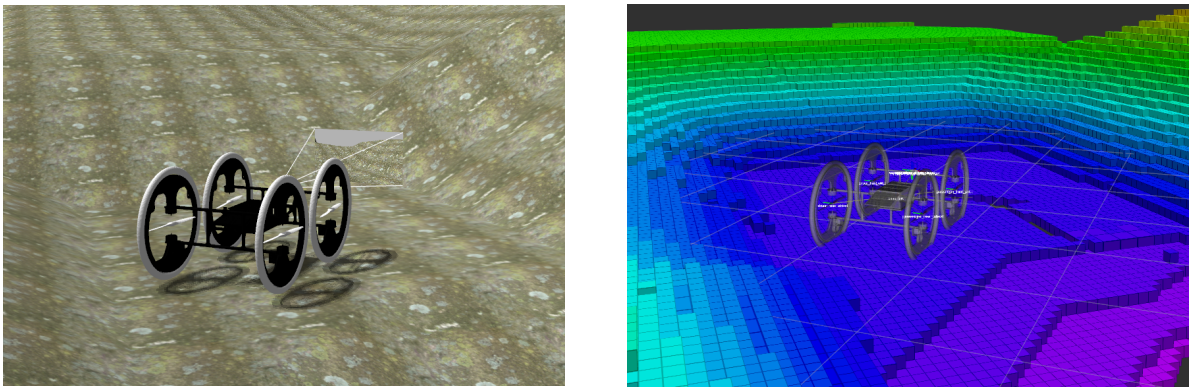
The primary sensors for localization will be the RGB-D cameras and an IMU. In particular, an Intel RealSense D435i will be chosen [34]. The RGB-D camera will take stereo images which will be fed into the VIO algorithm along with IMU data. RGB-D cameras also output dense point cloud data which will be used for building our environment map.

Visual inertial odometry will be performed by VINS-Fusion, which has been chosen because of its robustness in flying and driving [27]. This algorithm also allows for online spatial and temporal calibration of the IMU and camera [29]. Specifically, the vehicle will use a GPU accelerated version to take advantage of the on-board computer's powerful GPU. VINS-Fusion offers support for both monocular and stereo cameras for localization.

The vehicle will use the octree-based approach for point-cloud voxelization, OctoMap [33]. OctoMap will use the localization from VINS-Fusion and the point-cloud from the RGB-D camera to create an environment representation for the path-planner. The output can be seen with simulated terrain in Fig. 17. OctoMap allows for path-planning in three dimensions by providing nodes with $[xyz]$ coordinates for search elements. By using octrees, OctoMap also reduces memory consumption. A resolution of .1 m was chosen for the OctoMap to speed up computation while allowing for an occupied node to still fit under the chassis.

C. Path-Planning Algorithm: Modified A*

An efficient path planner is critical to navigating extraneous, off-road terrain. The non-uniformity of the environment presents challenges because the path planner must be efficient. With a large number of obstacles present, the path planner must choose the optimal route out of a large set of possible routes. Furthermore, the path planner must be fast to contribute to the rover's speed in maneuverability but robust to navigate a dangerous environment. Mistakes in path



a) **b)**

Fig. 17 a) Simulated terrain b) Visual representation of occupied space determined from OctoMap

planning, especially in flight, could cause collisions that could compromise the vehicle. The path-planner must also consider the different vehicle kinematics for each mode of travel, so it can plan paths that the vehicle is capable of following.

The path-planner is closely related to the perception algorithms used to map surroundings. In 3D space, the map size can grow at a cubic rate from the dimensions of the map. With using an octree implementation of the environment, the total nodes will grow at a cubic rate to the resolution of the octree. For example, doubling the resolution of the octree will octuple the amount of nodes to expand during a search [33]. Therefore, the search algorithm must drastically reduce the amount of nodes searched in the 3D environment.

The A* search algorithm [35] is similar to djiskta's algorithm [36], but it uses a heuristic function to prioritize expanding nodes in the direction of the goal node. The heuristic function scores each node by the distance to reach that node and the distance to the goal node. The node with the lowest score is expanded next. This can drastically reduce the search time of the algorithm by always searching the node that has the highest probability of being in the optimal path.

A* algorithms have been used in path-planners for UAVs because of their speed[37]. Sharif et. al showed the A* can plan energy efficient paths for multi-modal vehicles, and the path-planner designed in this paper follows a similar approach [38]. The algorithm requires accurate localization, environment mapping, and a destination.

1. Design Criteria

The following criteria were considered when designing a search algorithm for the multi-modal vehicle platform.

1) **Path Efficiency:** The paths-generated must be optimal for the vehicle kinematics. The path's must be optimized for low distance, time, and energy consumption.
2) **Computational Speed:** The path-planner must be able to generate paths quickly to prevent the vehicle from waiting for path generation. The vehicle must also be able to re-plan when the map is updated. This is especially

important during flight to prevent energy consumption due to hovering.

3) **Robustness:** The paths generated must not contain errors that would cause collisions. The path-planner must be able to plan efficient and safe paths in a variety of possible terrain environments.

2. Modified A Search Algorithm*

The pseudo code for the A* implementation can be seen in Algorithm 1.

Algorithm 1: Multi-modal A*

```
Result: Set of way points in 3D space
next = startNode;
openList = [];
closedList = [];
while next != NULL do
    closedList.insert(next);
    if isDestination(next) == true then
        path = buildPath(next);
        return path;
    else
        for each neighbor of next do
            if isCollision(neighbor) == false then
                neighbor.parent = next;
                neighbor.score = getHeuristic(neighbor, next);
                openList.insert(neighbor);
            else
        end
    end
    next = openList.pop;
end
```

The vehicle's path-planning algorithm begins at the current position of the vehicle. As seen in Fig. 18 below, the neighboring 26 nodes are individually checked for collisions using the collision checker described in the following section. The distance between nodes is .1 *m* dependent on the resolution of the OctoMap. Green nodes (lateral) will cost the least, and blue nodes (diagonal up/down) will cost the most. The yellow nodes (up/down) can only be used during flight.

When a node is discovered and does not result in collision, it is assigned a score based on the energy consumed to reach that node and the euclidean distance to the destination. The energy to reach a node is calculated by adding the

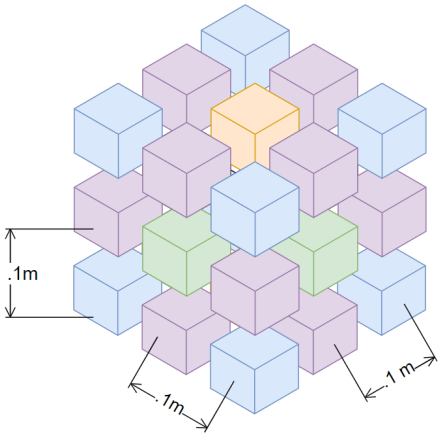


Fig. 18 Graphical representation of the 26 neighboring nodes that can be added to the OPEN list during expansion.

energy required to reach the parent node to the energy difference between the nodes. The nodes are added to a sorted linked list called the OPEN list, so nodes with the lowest scores are at the top of the list. This heuristic function ensures that the algorithm finds the optimal path for energy consumption and distance travelled [39]. The next node to expand is popped from the top of the OPEN list and added to the CLOSED list. The CLOSED list holds all nodes that have been expanded to ensure no node is expanded twice. The process of discovering, checking, and expanding nodes is repeated until one of two conditions is met.

1) The destination node is reached and expanded.
2) The edge of the mapped environment is reached, so the path should be executed until sufficient information has been added to the map for re-planning.

3. Collision Checking

Collision checking uses a set of coordinates relative to the position of the vehicle to determine if a part of the rover collides with the environment. The set of coordinates is a 3D array of locations with a spacing of .1 *m* to match the OctoMap resolution. The collision checker uses different collision boxes for flying and driving. Flying uses a simple 1.8 m x 2.0 m x 3.0 m box to provide a .5 m buffer on each side of the rover. Driving uses a shape similar to the rover to allow for ground clearance under the chassis. Furthermore, the collision checker will check to make sure the nodes below the tires are occupied while driving. Figure 19 shows the collision checker used while driving. The center of the physical rover will coincide with the center of the coordinates being checked. The green nodes must be free for this to be a valid position. The blue nodes represent the nodes that must be occupied by the ground for this to be a valid position. In each column of blue nodes, only one of the two must be occupied. The collision boxes will be oriented in the direction of travel. To reduce the amount of Nodes to be checked, internal boxes are not checked. Only the outer shell of nodes is checked for collisions.

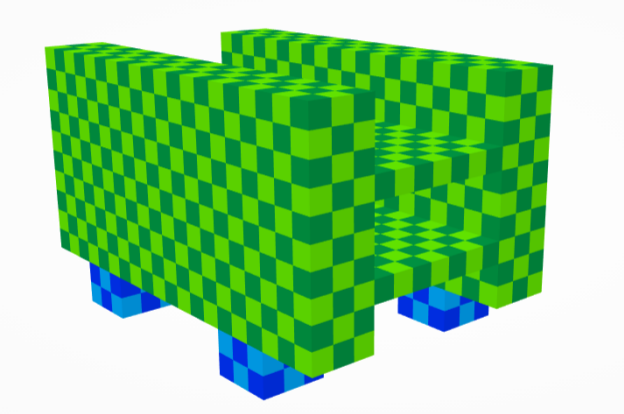


Fig. 19 Visual representation of the nodes checked for collision while driving.

D. Control Loop

Control loops play a critical role in autonomous robotics and navigation. A control loop accepts an inputted signal or value and gives a desired output, with some amount of error. In a closed control loop, the output signal is fed back to the input, and the error between the desired output and actual output adjusts the input signal in a continuous loop, with the ultimate goal of converging on the desired value.

An effective control loop for this vehicle is necessary, especially when navigating rough terrain. The automatic controls will determine the required velocity for the vehicle as it approaches the next way point. If the loop is unstable, this can lead to overshoot or oscillations, which could be dangerous for the vehicle with several obstacles in the vicinity. The control loop is designed to take in a series of way points from the path planner and translate them into commands for the controllers of the vehicle. The vehicle uses its current location and the global position of the next way point to methodically traverse along the given path. There are essentially two loops; one being for the driving controller, and the other is for the flight controller.

1. Design Criteria

The following criteria were considered when designing the control loop for this vehicle.

1) **Accuracy:** The control loop needs to accurately direct the vehicle from its current position to the desired way point. This requires reliable controls and consistent communication with the path planner and the FSM.
2) **Stability:** The controls must be well tuned such that the vehicle's position converges upon the way point with minimal overshoot or oscillation.
3) **Vehicle Safety:** The control loop is designed with vehicle safety in mind. The final implementation will include methods of becoming unstuck or regaining traction if the vehicle is sliding. Integration of wheel encoders will make this possible with future work.

2. Control Loop Design

Due to the multi-modal nature of the vehicle, the control loop needs to consist of two separate control loops. The design of the control loops is shown in Fig. 20. The outermost loop is the navigation loop, which will consist of the SLAM and path planner nodes. The inner loops are split into parallel loops, one set to handle flying, and the other set to handle driving. The navigation loop takes the current position and destination, and outputs a path. The middle loops are the flying and driving controllers that translate commands from the outer loop to the inner loops by taking in the route and producing relevant velocities for the inner loops, which contain the on-board flight controller and the differential drive controller. The inner loops communicate with the motors based on the velocities they receive and publish an error that is reported back to the navigation loop.

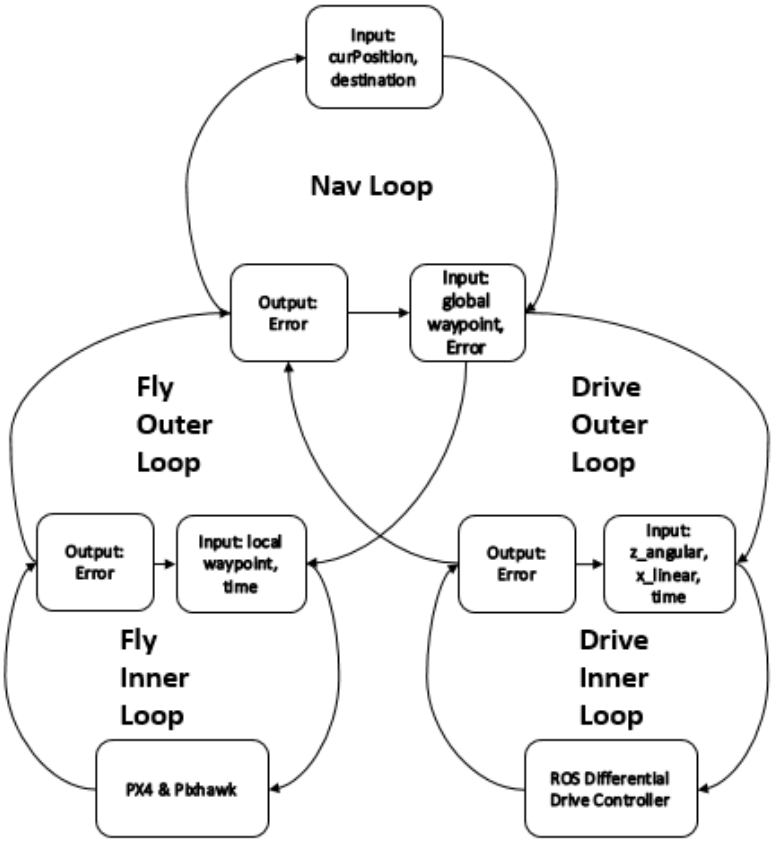


Fig. 20 Control loop architecture.

3. Driving Control

When the vehicle is on the ground, it will be controlled by the driving controller. When given a new way point, the vehicle first turns in the direction of the way point until it is within the defined angular threshold. Then, the vehicle begins driving forward and turning to approach the way point and improve the accuracy of the angle at which it is

approaching. When the vehicle reaches the way point, it stops and begins to turn in the direction of the next way point. This process is depicted in Fig. 21.

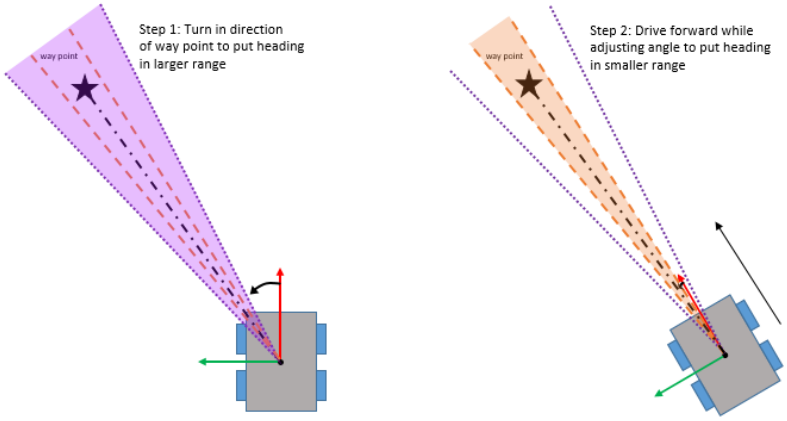


Fig. 21 Driving way point following operations.

The driving controller uses a positional closed control loop, which automatically adjusts for the linear and angular error between the way point and the vehicles current position and orientation. This control loop uses nested proportional-integral-derivative (PID) controllers for both heading and distance, meaning the velocity of the vehicle will change based on its orientation and position relative to the current way point. The proportional component slows the vehicle down as it approaches the angular threshold or the way point itself. The integral component takes a time summation of angular error and distance to the next way point, which consequentially increases the speed as time goes on. Finally, the derivative component adjusts the speed of the vehicle based on the time derivative of angular error/distance to the way point, which ultimately provides stability as the vehicle is approaching the next way point. The three components create a stable system that converges on the way point.

The distance and heading control loops can be seen in Fig. 23. For the distance controller, $p_d(t)$ is the desired position, and $e_p(t)$ is the positional error. $v_d(t)$ is the desired velocity, and pwm is the pulse-width modulation signal that gets sent to the motors. $v(t)$ is the vehicle's velocity, $N(t)$ is a noise signal that represents any disturbances to the system, and $p(t)$ is the vehicle's position. For the heading controller, $\theta_d(t)$ represents the vehicle's desired angle, and $e_\theta(t)$ is the angular error. $\omega_d(t)$ is the desired angular velocity, and pwm is the signal that gets sent to the motors. $\omega(t)$ is the vehicle's current angular velocity, $N(t)$ is the noise signal that represents disturbances to the system, and $\theta(t)$ is the vehicle's current angle, or heading. For both controllers, $G_c(S)$ is the transfer function for the controller, $G_{DD}(s)$ is the transfer function for the ROS Differential Drive controller, and $G_M(s)$ represents the motor transfer function. $P(S)$ represents the integral from velocity to position, or angular velocity to heading, which is $1/s$. The ROS Differential Drive Controller [40] is used, which has its own proportional controller and acts as the nested velocity controllers that send commands to the motors. The PID tuning constants for the distance controller are $k_p = 0.11, k_i = 0.01, k_d = 0.35$,

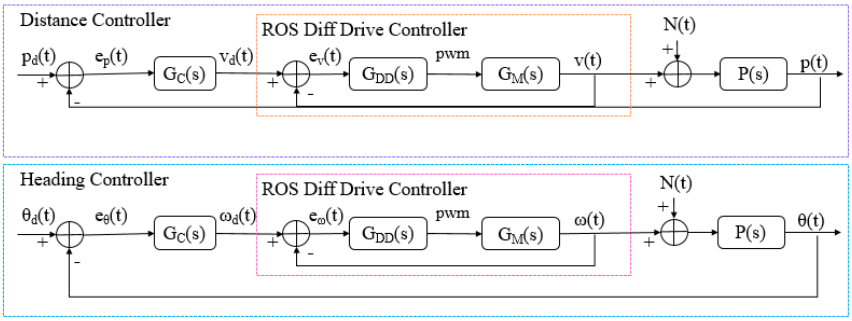
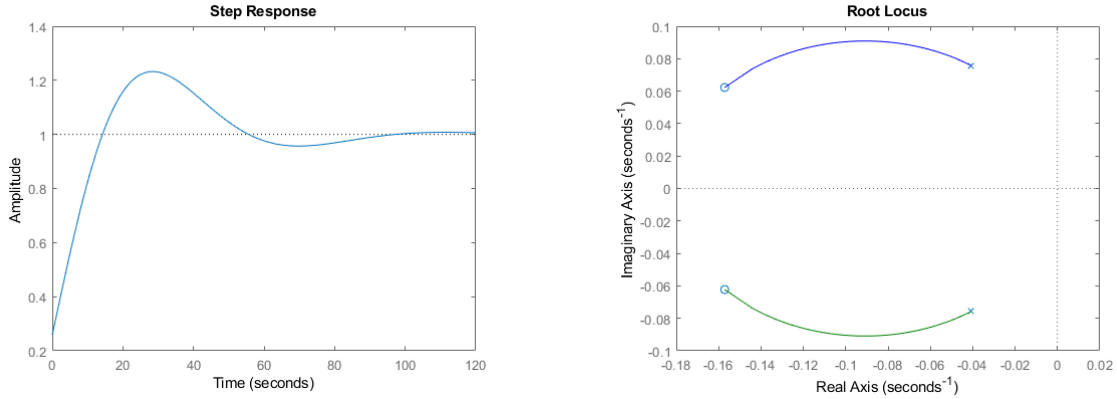


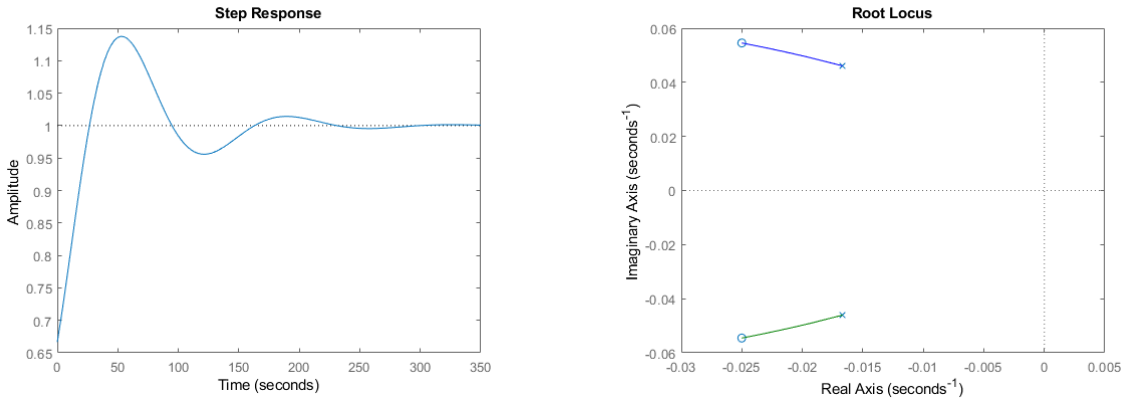
Fig. 22 The controllers for distance and heading.

and the constants for the heading controller are $k_p = 0.1$, $k_i = 0.0072$, $k_d = 2.0$. The behavior of the distance controller can be seen in Fig. 23. The root locus shows that the poles are to the left of the imaginary axis, which gives the system stable behavior. The step response does display some overshoot, but due to how the control loop operates, this does not present a problem. When the distance error is less than 5 centimeters, the control loop switches way points and sets the linear velocity to zero such that the vehicle can turn in the direction of the next way point, as described previously. This control structure suppresses the overshoot of the step response. The behavior of the heading controller can be seen in Fig. 24. As with the distance controller, the poles are placed such that there will be a stable system while avoiding marginal stability, which occurs when poles reside on the imaginary axis. The step response plot shows a small amount of overshoot, but similarly to the distance controller, when the heading error is within a small threshold, the angular velocity is set to zero, and the vehicle drives straight toward the way point. This prevents the overshoot shown in the step response, which is further mitigated as the vehicle then adjusts its orientation while driving to the way point. The controllers allow the vehicle to drive along the given path accurately.



a) **b)**

Fig. 23 Distance controller a) step response and b) root locus.



a) **b)**

Fig. 24 Heading controller a) step response and b) root locus.

4. Flying Control

The flying control loop is the connection between the path planner and the on-board flight controller. However, the on-board flight controller had not been integrated into the simulation before the need for demonstration. For the initial demonstration, the ROS library, Hector Quadrotor [41], was used for flying in simulation. The library contains the necessary controllers (position, heading, attitude, etc.) for flight, as well as the required Gazebo plugins for flight physics for the simulation. Controllers from Hector Quadrotor were utilized in control loops similar to those used with the ROS Differential Drive Controller. This worked for the initial demonstration, but the simulation is being developed to use the simulated on-board flight controller instead of Hector Quadrotor. Since the on-board flight controller can accept way points on its own, the control loop is a translator between the path planner and said controller. The path planner will publish global $[xyz]$ coordinates to a ROS topic, which the control loop then converts to the vehicle's local reference frame before forwarding the location information through the MAVLink [42] protocol to the autopilot firmware using MAVROS [43] commands. Additional messages are sent to initialize flight controller mode changes, arming/disarming and takeoff/landing commands. The autopilot interprets these messages and sends corresponding signals to the flying motors to enable travel to the way points. The proposed flight controller is the mRo Pixhawk [44]. The Pixhawk was chosen due to its popularity in autonomous unmanned aerial vehicles (UAVs), compatibility with the PX4 [45] autopilot, and team member experience with the flight controller. The autopilot chosen for the flight controller was PX4 due to it's comprehensive documentation and SITL simulation support. The primary goal of using a Pixhawk board with PX4 autopilot are to prove feasibility through simulation while expediting migration from SITL testing to testing with a TURVTOL prototype.

5. Vehicle Safety

The control loop is responsible for vehicle safety. This responsibility includes responding to the states determined by the FSM as well as reacting to relevant environmental sensing. A few examples of safety measures required in the control loop include sensing if the vehicle is slipping or stuck on the terrain. While driving, the control loop will run continuous comparisons of the vehicle's pose as obtained through VIO and the wheel encoders. If the VIO determines that the vehicle is moving more than is calculated with the wheel encoders, the FSM will assume the vehicle is slipping and go into the SLIPPING state. If the opposite situation occurs, the FSM will go into the STUCK state. In either case, the control loop will execute protocols to regain traction. Traction regaining protocols will be designed as a part of future development.

E. Finite State Machine (FSM)

For the vehicle to operate with multiple modes of travel, a state control machine is necessary. Finite state control is a valuable design such that the vehicle's behavior can be cleanly organized into a finite number of discrete states. Transition signals, along with the vehicle's current state, will determine the next state of the vehicle. This type of state control will allow for coordination across all of the various ROS nodes, including the SLAM node, path planner node, and control loop node.

1. Design Criteria

The following criteria were emphasized when desigining this FSM.

1) **Robustness:** The finite state machine needs to be as robust as possible in terms of logic flow and environmental response. To meet this requirement, states were defined for the vehicle and fine tuned to consider all possibilities of operating conditions. The FSM was then further fine tuned over several weeks of consideration of vehicle operations.
2) **Efficiency:** The FSM is designed to make other nodes as efficient as possible. The states will inform other nodes, such as the path planner and the control loop, of the most optimal way to operate. For example, when the vehicle is driving across terrain that would make it difficult for the vehicle to takeoff for flight, the FSM will switch from the NORM_DRIVE state to the DRIVE_NO_TAKEOFF state, which will inform the path planner to not even consider flight paths until the vehicle is back on level terrain. The same thing will occur if the wind is too strong for the vehicle to fly safely. This will increase the efficiency of the path planner, and subsequently, the vehicle's computation.

2. FSM Design

A finite state machine (FSM) has been designed and the general structure has been implemented using the ROS library SMACH [46]. SMACH is used because it allows for relatively simple implementation of complex state machines. The hierarchical design is shown in Fig. 25. At a high level, the states are FLY_OPERATE and DRIVE_OPERATE. Both of these states are sub-machines in the hierarchical state machine design. The FLY OPERATE state handles operations when the vehicle is in the air, and the DRIVE_OPERATE state handles operations when the vehicle is on the ground. The FLY OPERATE sub-machine contains a LANDING sub-machine, which contains states for executing search, return, and landing protocols. The DRIVE OPERATE sub-machine contains a TRACTIO_ LOSS sub-machine that executes protocols when the vehicle is stuck, slipping, or flipped upside down. DRIVE_OPERATE also contains the DORMANT sub-machine, which handles charging and sleeping states for the vehicle. The lowest level states and their respective descriptions are detailed in Table 4.

Table 4 Descriptions of the lowest level FSM states.

State	Description
CHARGING	Low battery
DRIVE_NO_FLY	Driving operations in circumstances where takeoff would be unsafe
FLIPPED	Vehicle seems to be flipped over
FLY	Normal flying operations
HOVER	Waiting for update from path planner
LAND	Path planner has confirmed landing
NORM_DRIVE	Normal driving operations
RETURN_TO_LAUNCH	Cannot find safe landing location; return to known safe location (launch site)
SEARCH_FOR_LANDING	Assesses terrain below for safe landing location
SLEEPING	No current destination; awaiting new mission
SLIPPING	Vehicle seems to be sliding or slipping
STAND_STILL	Waiting for update from path planner
STUCK	Vehicle seems to be stuck
TAKEOFF	Path planner has confirmed takeoff

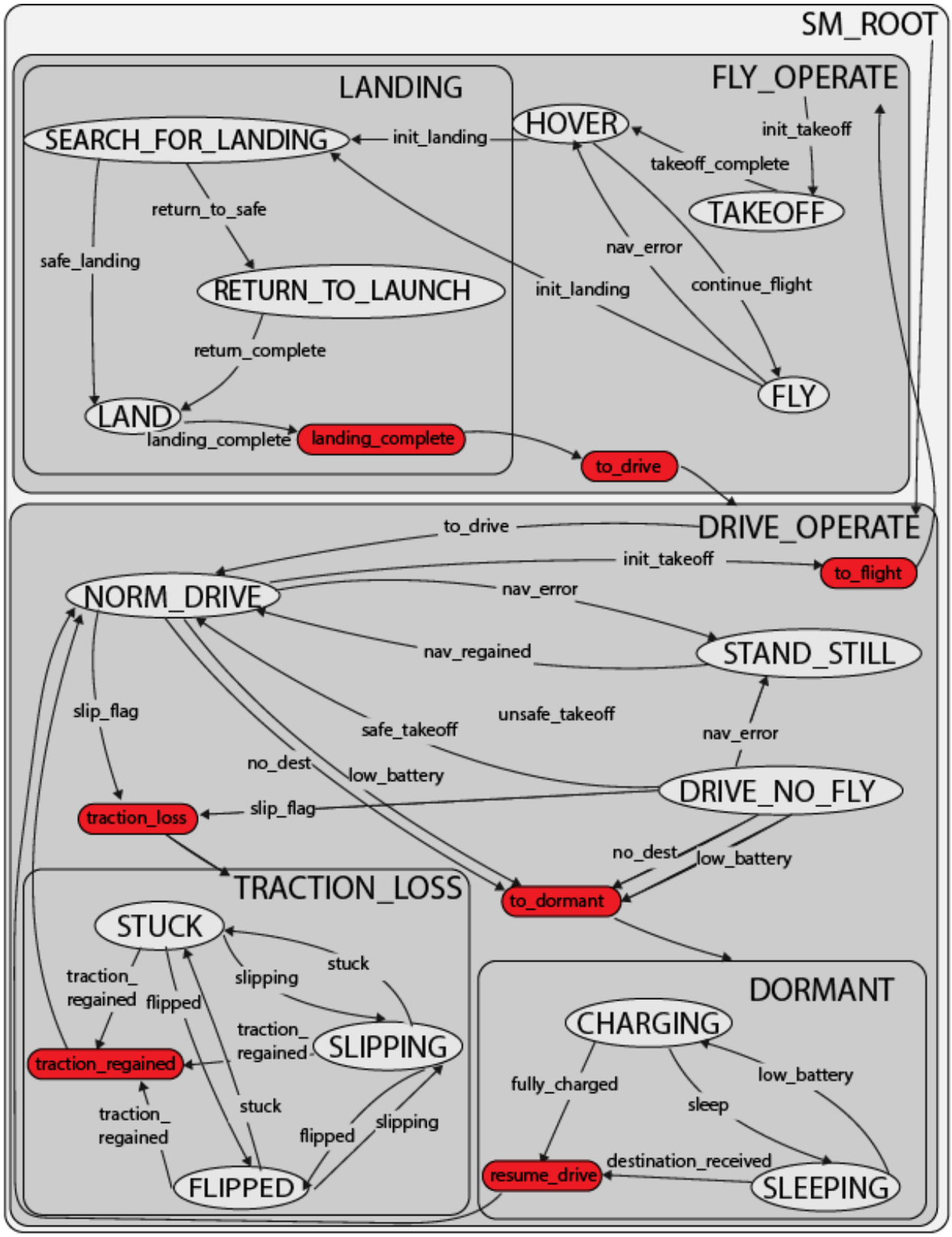


Fig. 25 Visualization of the FSM node, designed as a hierarchical state machine using ROS library SMACH.

Additionally, the FSM includes transition signals that are assigned depending on what the SLAM algorithm, path planner, and control loop are detecting or calculating, as well as certain environmental factors, such as the smoothness of the terrain. The transition signals are designed to keep the vehicle functioning safely by monitoring various parameters, such as battery level, safety of potential landings/takeoffs, and flying conditions, among other important factors. The transition signals are described in Table 5. Currently, the FSM structure has been implemented, and future work includes development of methods for each state and integration of the FSM node with the various ROS nodes.

Table 5 Descriptions of the FSM transition signals.

Transition Signal	True when...
continue_flight	Path planner sends confirmation to continue flying (instead of landing)
destination_received	A New mission has been received
fully_charged	Charging protocols are complete
flipped	VIO indicates that the vehicle has flipped upside down
init_takeoff	Path planner confirms takeoff
land	Either path planner confirms landing or battery is low or there is no current destination
landing_complete	Landing protocols have been completed successfully
low_battery	Battery level is below threshold determined by current state
nav_error	Expected way points are not being published
nav_regained	Way points are once again published as expected
no_dest	The vehicle doesn't have a destination and must await a new mission
resume_drive	The vehicle has both an adequately charged battery and a destination
return_complete	Vehicle has returned to the site of the most recent safe landing location
return_to_safe	Path planner confirms landing but a safe landing cannot be made within a specified search radius
safe_landing	Path planner confirms landing and safe landing has been ensured
safe_takeoff	The current terrain is suitable for takeoff and vehicle's battery is high enough to facilitate safe flight
slip_flag	Vehicle is not moving as expected
slipping	VIO indicates that vehicle is moving more than encoders indicate
stuck	VIO indicates that vehicle is moving less than encoders indicate
takeoff completed	Takeoff protocols have been completed successfully

Table 5 Descriptions of the FSM transition signals.

Transition Signal	True when...
to_drive	Path planner confirms switch from flying to driving
unsafe_takeoff	The current terrain is unsuitable for takeoff or the vehicle's battery is too low to facilitate safe flight

F. Simulation Testing

An effective simulation environment allows for the testing of software while the vehicle itself is still being designed or built. Implementing robotic behaviors in simulation before testing with hardware both allows for safe development as well as facilitates working remotely. The goal of the simulation environment was to be directly transferable to our hardware prototype and to be able to test the vehicle in realistic conditions.

1. Simulation Software

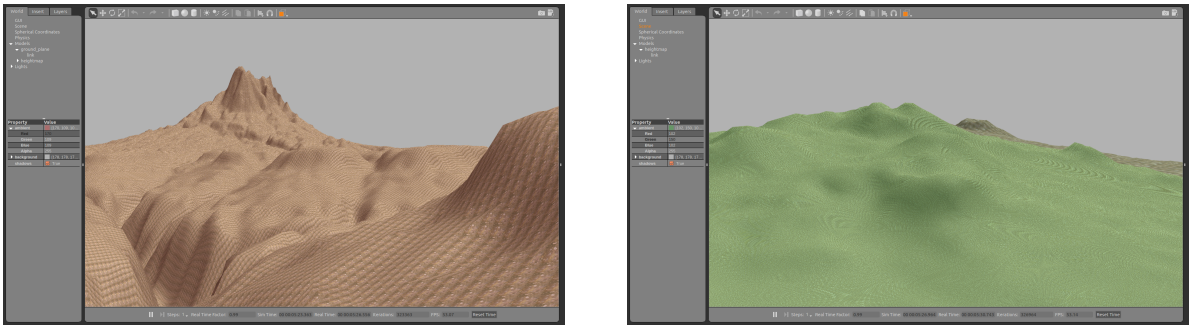
The preferred software of choice in this endeavor was Gazebo due to its common use for simulating robots and autonomous systems and its open-source nature. ROS was used because of its many libraries and compatibility with the hardware that was being used. Three dimensional visual models were created for the project using a combination of SolidWorks and Blender. These models were used to test the path planning and visual-inertial odometry functionalities of the rover.

2. Sensing

The sensors currently being simulated are Intel RealSense D435 cameras, an inertial measurement unit (IMU), and wheel encoders. The RealSense cameras are being simulated using the provided plugins from RealSense. The IMU is currently simulated using the Gazebo IMU plugin, but the Pixhawk has a built in IMU. Once Pixhawk integration is completed, the IMU data will be exported and used in place of the Gazbeo plugin. The wheel encoders are simulated through ROS and Gazebo.

3. Terrain

The terrain for the simulation was created using Mathew Verbryke's Gazebo Terrain repository [47]. The repository allowed us to create terrain from height maps of any area on Earth, which provided us with the opportunity to integrate realistic topography from potential mission sites into our simulation. Terrain generated from height maps of areas within Arizona and Hawaii are shown in Fig. 26. These terrain models were used for the testing of OctoMap and VINS-Fusion in simulation to assess the accuracy of feature tracking, localization, and mapping.

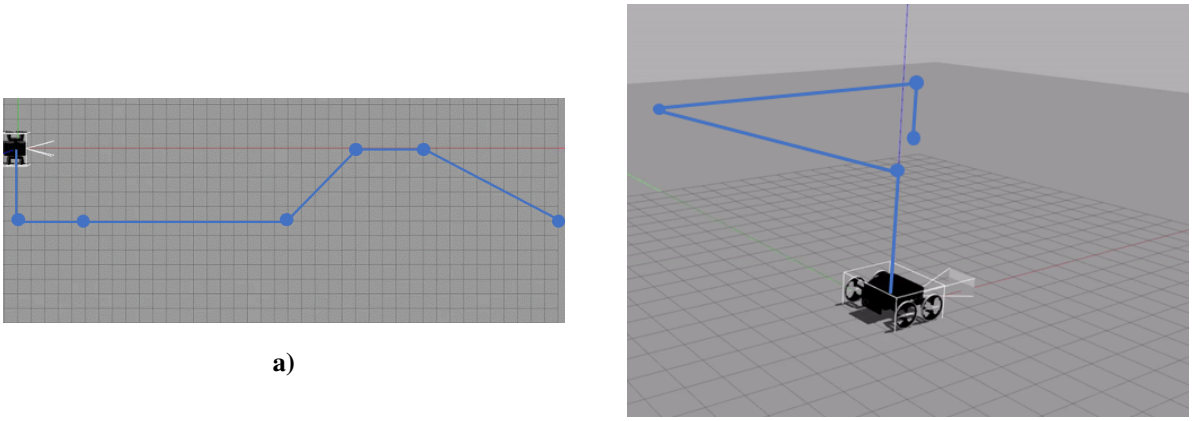


a) **b)**

Fig. 26 Terrain generated from height maps of an areas in a) Arizona Terrain and b) Hawaii.

4. Results

By the end of the summer, the vehicle could autonomously navigate both driving and flying paths, as shown in Fig. 27. Continued development since the end of the summer has included development of the path planning algorithm, ability to follow multi-modal paths, and integration of VINS-Fusion with the path planning and control loop nodes. Integration is an ongoing task, and progress is discussed more thoroughly in the discussion section.



a) **b)**

Fig. 27 a) Vehicle autonomously navigates driving path of 6 way points. b) Vehicle autonomously navigates flying path of 4 way points. Note: The rover depicted here is an older model and does not accurately reflect the rover's current design.

VI. Discussion

A. Hardware

The flight and ground systems described previously serve as the first step in developing a multi-modal autonomous vehicle capable of tackling challenges insurmountable by either a purely driving or purely flying vehicle. Namely, it seeks to allow for long range, long duration missions which take place primarily on rough ground terrain. A problem

faced by many ground systems with similar missions is dangerous or impassible terrain, such as rock fields, cliffs, or bodies of water. This design aims to bypass these problems by simply flying over them, allowing for unparalleled mission freedom. This work adds to the relatively undeveloped concept, and proposes a feasible first iteration conceptual design capable of surmounting these challenges which could be used as a springboard for future research and iterations.

One area which dramatically benefited from the first iteration was the wheel and rotor sizing. Due to the unique wheel configuration which protects the rotors in the case of a tumble, the weight of the wheel and lift of the rotors are deeply coupled. Initially, rotor size was minimized to ensure the wheel would be feasible to manufacture, but having completed the design, it is clear that there is room for improvement. Specifically, a rotor of 27" maximizes the effective thrust of the system.

B. Software

The software described in the previous sections has proposed a framework and implementation for a multi-modal vehicle platform. The software design of this vehicle has focused on using algorithms and sensors that create a continuous software flow between flying and driving. The individual algorithms themselves can be tuned to better achieve this goal.

Visual-inertial odometry, specifically VINS-Fusion, was chosen because it is robust and can be GPU accelerated. Furthermore, The required sensor suite is adept for both modalities of the vehicle, so no major changes in localization would take place during transitions. VINS-Fusion will be paired with OctoMap to provide environmental sensing. Due to the size of the vehicle, multiple RGB-D cameras will be used to provide a full picture of the environment. Approaches for SLAM with multiple cameras will need to be explored, and they have been shown to be more robust than monocular SLAM [48].

The path-planner uses a modified A* that will optimize for power consumption when planning paths that include flying and driving. The weights used for the search will need to be tuned to produce the desired hopping motion of the vehicle. The most problematic aspect of planning is choosing when to land. The weight for landing must be tuned such that the rover will choose to land after it has passed the obstacle, but it will not constantly check for landing after each lateral movement. Another challenge is that this map will be dynamically built as the rover visualizes more terrain, so the path-planner will not typically be able to plan a path that will take off and land in one iteration. The vehicle must plan to take off, and decide where to land in a following iteration.

A solution of having adapted weights during the search execution to prevent wasting computation time will be explored. For example, the weights during flight will grow as the rover has been in the air longer. This will prevent the search algorithm from immediately exploring landing after take off.

To make the vehicle travel more efficiently, the path will be smoothed to account for different vehicle kinematics in both modes of travel. A multi-modal smoothing technique was not proposed in this research, so this will be a subject of future investigation.

The control loop currently executes driving paths and flying paths separately. The driving control loop takes in global coordinates from the path planner, and by converting these to relative coordinates, traverses way point to way point using a PID controller. The driving control loop utilizes PID values that allow the vehicle to traverse along the given path efficiently, accurately and in a stable manner. For the purpose of demonstration, the flying control loop currently utilizes the ROS library, Hector Quadrotor [41], to convert a determined command velocity into commands for the vehicle's hardware. Ideally, Hector Quadrotor will be replaced with Pixhawk and PX4. This ultimately allows the vehicle to traverse airborne paths. Future development will include the traversal of multi-modal paths, design of vehicle safety protocols, and fine tuning the efficiency of both control loops.

The FSM architecture is currently implemented using SMACH. Future work will include developing methods for each state in the FSM and integrating the FSM with the other ROS nodes. The other nodes will need to be modified to publish the signals that the FSM requires to determine the vehicle's state, as well as react to the vehicle's current state. Additionally, the FSM will need to be tested in simulation to ensure that it reacts as expected to changes in the transition signals.

A Gazebo simulation is currently being developed to test and fine tune the software systems. The simulation in development will provide the means to test and optimize the software for the terrain the vehicle will likely encounter. The simulation is currently being used to test individual components of the software structure, and will soon be used to test the integration of all components. Due to the discrepancies between the simulation and physical models, the software will need to be loaded onto a physical prototype for official testing to determine the efficacy of the proposed software.

C. Future Work

A second iteration is currently being designed with the understanding gained from TURVTOL V1. It has been resized to minimize the extra weight of a larger wheel while taking maximum advantage of a larger rotor. The rotors have been moved closer together following experimental testing. Additionally, the payload box has been redesigned to further reduce its weight, and it is being designed such that virtually every component can either be purchased off the shelf or easily manufactured. The second iteration is still a work in progress, but it is estimated to weigh around 60 kg with a payload of about 15 kg while still being capable of the same missions as TURVTOL V1 showing the promise of the system.

The software flow between TURVTOL V1 and V2 will be the same. The software design outlined in this publication will continue to be developed. After the software is developed, it will be fine-tuned using gazebo gazebo simulations and implemented in the physical prototype. Additional designing for communications and charging capabilities will occur once the primary algorithms and nodes that are currently in development have been successfully integrated.

VII. Conclusion

This design paper proposes a multi-modal vehicle system capable of flying and driving in extraneous terrain. A multi-modal vehicle would allow autonomous travel in exceptionally hazardous terrain, which could be valuable for scientific data collection. The vehicle is propelled by 4 differential drive wheels with 2 coaxial rotors inside each wheel hub. The vehicle is still capable of efficient flight with the reduced lift from the rotors being located inside the wheel hub. The design allows for minimal footprint for effective ground traversal. A full software design for multi-modal, autonomous travel has also been discussed. The software includes environmental sensing, localization, mapping, path-planning, control loops, and state determination for a vehicle that drives and flies.

Funding Sources

Individual stipends were provided by the Minnesota Space Grant Consortium, NASA Kentucky Space Grant Consortium, North Carolina Space Grant, Oregon Space Grant Consortium, Pennsylvania Space Grant Consortium, Virginia Space Grant Consortium, West Virginia Space Grant Consortium, and Wisconsin Space Grant Consortium. Christopher Newport University and NASA Langley Research Center provided funding for the project.

Acknowledgments

This research team wishes to acknowledge and thank Elizabeth Ward for her assistance and support. Additionally, we would like to thank Zack Bassett, Ryan Bowers, Annabelle Durand, Mary Dijoseph, Thomas Jordan, James Neilan, Dave North, Patrick Quach, Garry Qualls, Aaron Shepard, Kyle Smalling, and Loc Tran for participating on our advisory committee. We would also like to thank the following:

NASA Langley Research Center for providing necessary hardware and software access, as well as access to research resources.

Christopher Newport University for providing funding.

ViGYAN for performing propeller efficiency tests.

References

[1] Nathalie A. Cabrol, K. W. E. A. G. J. M. G. C. D. C. S. C. P. C. C. D. J. M. D. L. E. G. F. J. G. C. H. A. N. H. D. J. L. M. E. M. G. G. O. J. P. E. P. T. S. K. S. G. T. D. T. A. W. M. W. S. W. M. W., David Wettergreen, "Life in the Atacama: Searching for life with rovers," *Journal of Geophusical Research: Biogeosciences*, Vol. 112, 2007.

[2] Eric Trautmann, J. L., Laura Ray, "Development of an Autonomous Robot for Ground Penetrating Radar Surveys of Polar Ice," *International Conference on Intelligent Robots and Systems*, October, 2009.

[3] Karsten Berns, C. A., Klaus-Dieter Kuhnert, "Off-road Robotics-An Overview," *KI - Künstliche Intelligenz volume*, Vol. 25, 2011, pp. 109–116.

[4] Bakker, J. W. D., "Personal land and air vehicle," , 2011-04-26. URL `https://patents.google.com/patent/US7931230B2/en`.

[5] Radu, B., "Flying car or drone," , 2018-09-25. URL `https://patents.google.com/patent/US10081424B2/en`.

[6] Stekelenburg, M. A. W., Joore, H., Klok, C. C., Rijn, L. P. V. M. V., Beek, K. R. V. D., and Ruit, E. V. D., "Fly/Drive Vehicle That Is Convertible Between A Road Riding Condition And A Flying Condition," , 2020-02-20. URL `https://patents.google.com/patent/US20200055358A1/en`.

[7] Arash Kalantari, M. S., "Design and Experimental Validation of HyTAQ, a Hybrid Terrestrial and Aerial Quadrotor," *IEEE International Conference on Robotics and Automation*, 2013.

[8] Jared R. Page, P. E. I. P., "The Quadroller: Modeling of a UAV/UGV hybrid quadrotor," *International Conference on intelligent Robotics and Systems*, 2014.

[9] Meiri, N., and Zarrouk, D., "Flying STAR, a Hybrid Crawling and Flying Sprawl Tuned Robot," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5302–5308. https://doi.org/10.1109/ICRA.2019.8794260, ISSN: 2577-087X.

[10] Du, T., Schulz, A., Zhu, B., Bickel, B., and Matusik, W., "Computational Multicopter Design," *ACM Trans. Graph.*, Vol. 35, No. 6, 2016. https://doi.org/10.1145/2980179.2982427, URL https://doi.org/10.1145/2980179.2982427.

[11] Xoar, "Xoar Brushless Electric Motors," , 2020. URL https://www.xoarintl.com/brushless-electric-motors/.

[12] Xoar, "Titan Air TA82," , 2020. URL https://www.xoarintl.com/brushless-electric-motors/titan-air/titan-air-TA82-light-weight-pro-series/.

[13] Lei, Y., Bai, Y., Xu, Z., Gao, Q., and Zhao, C., "An experimental investigation on aerodynamic performance of a coaxial rotor system with different rotor spacing and wind speed," *Experimental thermal and fluid science*, Vol. 44, 2013, pp. 779–785.

[14] Lei, Y., Ji, Y., and Wang, C., "Optimization of aerodynamic performance for co-axial rotors with different rotor spacings," *International journal of micro air vehicles*, Vol. 10, No. 4, 2018, pp. 362–369.

[15] Lim, J., McAlister, K., and Johnson, W., "Hover Performance Correlation for Full-Scale and Model-Scale Coaxial Rotors," *Journal of the American Helicopter Society*, Vol. 54, 2009, pp. 32005–1. https://doi.org/10.4050/JAHS.54.032005.

[16] Apostolopoulos, D. S., "Analytical configuration of wheeled robotic locomotion," , 2001.

[17] PLASCORE, "Standard Honeycomb Panels - Specifications for Standard Honeycomb Panels," , 2008. URL https://www.plascore.com/honeycomb/honeycomb-panels/standard-honeycomb-panels/.

[18] Fisher, B. D., "NASA Langley Aircraft Structural Requirements," , Mar. 2016.

[19] "Documentation - ROS Wiki," , 2020-10-20. URL http://wiki.ros.org/.

[20] Nickolls, J., and Dally, W. J., "The GPU Computing Era," *IEEE Micro*, Vol. 30, No. 2, 2010-03, pp. 56–69. https://doi.org/10.1109/MM.2010.41, conference Name: IEEE Micro.

[21] Hangün, B., and Önder Eyecioğlu, "Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations," *INTERNATIONAL JOURNAL of ENGINEERING SCIENCE AND APPLICATION*, Vol. 1, No. 2, 2017.

[22] Rodriguez-Losada, D., Segundo, P. S., Hernando, M., Puente, P. d. l., and Valero-Gomez, A., "GPU-Mapping: Robotic Map Building with Graphical Multiprocessors," *IEEE Robotics Automation Magazine*, Vol. 20, No. 2, 2013-06, pp. 40–51. https://doi.org/10.1109/MRA.2012.2220503, conference Name: IEEE Robotics Automation Magazine.

[23] Bleiweiss, A., "GPU Accelerated Pathfinding," *Graphics Hardware*, 2008.

[24] "NVIDIA Jetson Xavier NX for Embedded & Edge Systems," , 2020-10-19. URL https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/.

[25] Nister, D., Naroditsky, O., and Bergen, J., "Visual odometry," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 1, 2004-06, pp. I–I. https://doi.org/10.1109/CVPR.2004.1315094, ISSN: 1063-6919.

[26] Jones, E. S., and Soatto, S., "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," Vol. 30, No. 4, 2011-04-01, pp. 407–430. https://doi.org/10.1177/0278364910388963, URL https://doi.org/10.1177/0278364910388963.

[27] Qin, T., Li, P., and Shen, S., "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, Vol. 34, No. 4, 2018, pp. 1004–1020.

[28] Qin, T., and Shen, S., "Online Temporal Calibration for Monocular Visual-Inertial Systems," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3662–3669.

[29] Qin, T., Pan, J., Cao, S., and Shen, S., "A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors," , 2019.

[30] Qin, T., Cao, S., Pan, J., and Shen, S., "A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors," , 2019.

[31] Wiemann, T., Nüchter, A., Lingemann, K., Stiene, S., and Hertzberg, J., "Automatic construction of polygonal maps from point cloud data," *2010 IEEE Safety Security and Rescue Robotics*, 2010-07, pp. 1–6. https://doi.org/10.1109/SSRR.2010.5981571.

[32] Julian Ryde, H. H., "3D mapping with multi-resolution occupied voxel lists," 2010. https://doi.org/10.1007/s10514-009-9158-3.

[33] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W., "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, 2013. https://doi.org/10.1007/s10514-012-9321-0, URL http://octomap.github.com, software available at http://octomap.github.com.

[34] "Depth Camera D435i," , 2020-10-23. URL https://www.intelrealsense.com/depth-camera-d435i/.

[35] Peter E. Hart, B. R., Nils J. Nilsson, "A Fromal Basis for the Heuristic of Minimum Cost Paths," *IEEE Transactions of Systems Science And Cybernetics*, Vol. 4, July, 1968.

[36] Dijkstra, E., "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, Vol. 1, 1959, pp. 169–271.

[37] Jose Luis Sanchez-Lopez, M. A. O.-M. M. M. H. V., Min Wang, "A Real-Time 3D Path Planning Solution for Collision-Free Navigation of Multirotor Aerial Robots in Dynamic Environments," *Journal of Intelligent Robotic Systems*, Vol. 93, 2018, pp. 33–53.

[38] Amir Sharif, S. H. H. R., H.M. Lahiru, "Energy Efficient Path Planning of Hybrid Fly-Drive Robot (HyFDR) using A* Algorithm," *Proceedings of the 15th International Conference on Informatics in Control, Automation, and Robotics*, Vol. 2, 2018, pp. 201–210.

[39] Ferguson, D., Likhachev, M., and Stentz, A., "A Guide to Heuristic-based Path Planning," 2005, p. 10.

[40] Magyar, B., "Diff Drive Controller," , 2019-06-06. URL http://wiki.ros.org/diff_drive_controller.

[41] Meyer, J., and Kohlbrecher, S., "Hector Quadrotor," , 2014-01-07. URL http://wiki.ros.org/hector_quadrotor.

[42] Project, D., "MAVLink Developer Guide," , 2020. URL https://mavlink.io/en/.

[43] Team, P. D., "MAVROS," , 2020. URL http://wiki.ros.org/mavros.

[44] Team, P. D., "mRo Pixhawk Flight Controller (Pixhawk 1)," , 2020. URL https://docs.px4.io/master/en/flight_controller/mro_pixhawk.html.

[45] Team, P. D., "PX4," , 2020. URL https://px4.io/.

[46] Bohren, J., "SMach," , 2018. URL http://wiki.ros.org/smach.

[47] Verbryke, M., "Gazebo Terrain Gen," , 2019. URL https://github.com/MatthewVerbryke/gazebo_terrain.

[48] Yang, S., Scherer, S. A., Yi, X., and Zell, A., "Multi-camera visual SLAM for autonomous navigation of micro aerial vehicles," Vol. 93, 2017-07-01, pp. 116–134. https://doi.org/10.1016/j.robot.2017.03.018, URL http://www.sciencedirect.com/science/article/pii/S0921889015302177.