REALISTIC MODELS FOR CHARACTERIZING THE PERFORMANCE OF UNMANNED AERIAL VEHICLES

Joedocei Hill, University of Kentucky
Simone Silvestri

Available at: https://works.bepress.com/joedocei-hill/1/
REALISTIC MODELS FOR CHARACTERIZING THE PERFORMANCE OF UNMANNED AERIAL VEHICLES

Joedocei Hill

April 20th, 2018
Acknowledgements

I would like to thank my advisor Dr. Simone Silvestri, whom without, this project would not have been possible. Dr. Silvestri was always available when I needed him and offered valuable knowledge and experience throughout my duration of working with him on this project.

I would also like to thank my committee Dr. Judith Goldsmith and Dr. Neil Moore. Both of whom have been substantial influences in helping me get to the point to where I am now.

Lastly, I would like to thank Dr. Mirek Truszczynski and Dr. Cleo Price, in the Graduate School. Both of these men have been true fountains of knowledge and help throughout my graduate school journey specifically.
Contents

1. Problem Definition
   a. Introduction
   b. Application
   c. Related Works
2. Installation
3. Implementation
4. Analysis
5. Results
6. Future Work
7. Reference
8. Appendix
Problem

Introduction

Over the last decade as technology continues to become more developed and publicly available, it is us, the consumers that find ourselves with an onslaught of gidgets and gadgets to filter through and play with. One of these “newer” technological advances we find in our grasps today are Unmanned Aerial Vehicles, also called UAVs, or more commonly in society, drones. Drones come in many different shapes, sizes, and price ranges, with dozens of interfaces and produced by a number of different manufacturers. One particular point of interest to pay attention to is the price point of a drone. This single identifier determines several characteristics of the drone, for example all the ones listed previously. The lower a drone is on price the more you expect simple, less reliable and overall a cheaper quality piece of technology and this lack of quality is very evident. These lower priced drones are seen more as for children or those looking to get minimal experience with drones.

On the other end of the spectrum you find the higher quality drones, those which usually come with a programmable interface, all the bells and whistles such as gimbal attachments and autonomous flying modes, as well as dedicated software apps and a plethora of other perks. The model of drone we will be considering in our experiments is on the higher quality end, specifically the Solo 3DR drone.

The drones described in the preceding paragraph are on par with what a consumer could expect to see when shopping for drone. Unmanned Aerial Vehicles have been used by the government dating back to the early 50’s. As the decades have rolled on the government has of course upgraded and made modifications to their original drone concepts to make them sleeker, stealthier, more light weight, and just about every other beneficial attribute one could think of.

The models we think of now when we hear the term “drones” are rather small, manual and autonomous vehicles, usually used in photography and videography such as what anyone could commonly find on a modern youtube vlog (or video-blog). These smaller commercial drones
date back to the early 2000s but have recently gained a lot of popularity in the last decade or so, especially in the last 5 years or so with the giant popularity spike in youtube and more specifically the youtube vlogs just mentioned. Drones have also seen a raise in popularity due to their use in the courier business. A dozen or so notable companies, including Amazon, have fiddled around with the idea of using drones to transport goods to their customers. This new mode of courier transport would allow these companies a competitive edge over their competition as they continue to push the technological envelope and offer their customers experiences they didn’t even know they wanted. This forward thinking model follows the success of many other companies and entrepreneurs such as Henry Ford (creator of the Ford Motor Company) who is credited with the quote “If I had asked my customers what they wanted they would have said a faster horse,” meaning, give the people what they don’t know they want.

The problem that we will specifically focus on is developing a better understanding of the capabilities of the drones to accomplish a mission. It is a well known issue that consumer market drones are constrained in their resources, especially in terms of available energy. Manufacturers of these consumer drones give very limited information and insight of the true capabilities of their machines, such as only giving approximate flight times based on flight components that give the largest marketable available flight time such as hovering within a consistent bubble. Therefore we have taken on the challenge of creating two different models in order to better understand how the performance characteristics such as flight time are related to the given parameters of our drone system. The first model that we create and analyze looks at basic operations of the drone in flight ie. takeoff, land, and vertical and horizontal movements at constant speeds, while the second model is a bit more in depth. In the second model we look at not only the physical characteristics of the drone, such as whether or not it is carrying a payload, but also mission specifications in order to get a much more realistic and accurate prediction of the drone flight time and the number of waypoints it can safely traverse in a given mission. Our research here will help maintain both the physical systems operational reliability and safety, as
well as lead to the design of more effective and accurate optimization problems, mission planners, simulations, and algorithms.

**Application**

Now with a solid introduction about what drones are and what forms you can expect to find them, we can begin to dive deeper and look into applicable situations where we might use this technology for research purposes with ties to the real world.

As mentioned briefly in the last section, the government/military has been using this technology for decades, coming up on almost half a century. Although much of their use is classified information to a public audience we do know that they have used drones for unmanned aerial missions such as surveillance in hostile and unfamiliar areas. There are also documented works on the government using drones for national border security [4],[5]. In which the authors gathered imaging data on land using UAVs as a systematic way of capturing their data. We find this application to be quite amazing and maybe even revolutionary. This work of land surveying and border security directly ties with one of the biggest campaign promises of our current President, Donald Trump. Regardless of political stance and placing no emphasis on a side, this UAV technology could be a great addition or substitute to “building the wall” and save the American people hundreds of thousands, if not millions of dollars with further research and development by replacing expensive labor costs with autonomous drones that relay information back to a central office once every hour or so.

There has also been published articles [1] that state the Navy to be using drones to fight for our freedom and protect our rights. The referenced article goes to say how the Navy uses swarms of UAVs [as many as 30] that “utilize information-sharing [between one another], enabling autonomous collaborative behavior in either defensive or offensive missions.” This is another great military use of drone technology as we can now save human lives by sending in drones to play the offensive or defensive part we need them to in a combat situation. Which in turn frees up another person that can be used for a much greater purpose in our military. This use of technology will also saves millions of dollars in aircrafts, as we now do not have to send
multi-million dollar aircrafts into these hostile situations and instead can send dozens of low-cost drones manufactured for a few hundreds all together.

Also mentioned briefly before was the use of drone technology for courier purposes such as for the delivery of customer products through services like Amazon Air Prime [2]. Although this is still a trial service, Amazon assures that this technology will be utilized for package delivery. The first trials began in the UK in December of 2016. While the service was not extremely robust or even close to being market ready, it still shows the company is very serious and expects drone delivery services to be a favorable contender for customer deliveries. There has also been recent (2018) articles of a company named Zipline which claims to be the “world's first national and fastest drone delivery service.” This Silicon Valley firm uses drones to transport blood to medical clinics in Rwanda and soon other regions such as the United States. This just goes to further show the solid future that drone delivery of products will play over the course of our lifetime and even further shows the drastic impact it can have. Not only can drones be used to deliver consumer products such as books, electronics, and clothing but even medical supplies such as blood that can save someone’s life who might die, require extremely expensive transport services such as a helicopter, or be in such a rural areas, like Africa, that transportation of said medical supplies would be otherwise be nearly impossible.

We have documented proof that drone technology has been utilized in several different applications of industry surveying such as agriculture [4]. In this paper ([4]), drones were tested for their effectiveness in conservation efforts of forest and nature like forest fire detection, operations inspections, and location services. The drones were also tested for how effective they could be in terms of agriculture such as precision farming, quality of land characteristics, and harvest monitoring. This paper goes to show the power that technology can serve in industries that haven’t had much technological advancement in centuries other than equipment. Farmers and Conservation Officers now have the ability to monitor acres of land and production daily, hourly, or even every minute if choose to implement a drone fleet large enough that is capable of
performing their desired tasks by offering a tool that was once un-implemented or at least fairly expensive to acquire and utilize.

Additionally we have drone technology that creeps into many verticals of business such as construction, energy, and even insurance [3]. The PrecisionHawk company offers a one-stop shop for implementing drones into your company by offering drones, add-ons, and software packages all specialized for your platform. We feel this company is amazing and worth mentioning because it truly shows the broad nature and real time analytics that can come from implementing this technology into your business, whatever that may be. This company truly does show the true commercial value available to an organization that chooses to implement drone technology into their operations.

And lastly, a topic briefly mentioned in the introduction is consumer use such as for videography or photography purposes. This application is something that is heavily used in pop culture such as youtube videos and music videos. As acknowledged previously, youtube has become quite a popular and flooded video platform for vlogs, which simply means a video blog. A common trait you will find in a lot of popular vlogger’s videos such as Logan Paul, Casey Neistat, and Christian Guzman is that they often use drones to capture amazing landscapes or action shots of themselves. As pop culture usually goes, once popular icons in the community start a trend, it is quickly duplicated by others in hopes of similar success and the vlogs are no exception. You can now find even quite “modest” vloggers using drone technology to get more exciting footage in hopes of blowing up big. Modern music videos also quite frequently use drones to capture action shots of the artists, crowds of people, or beautiful landscapes, all depending on the theme of the video.

But drones do not stop their use only in the youtube world and have even crept into the recording of sports and movies. High-paced sports such as motocross have used drones to track bikers through their races and movies such as 2012 James Bond film “Skyfall” used a drone to record certain parts of the movie. Furthermore, we can easily find the utilization of drones in modern photography as well as they offer a range of new possibilities and viewpoints not before achievable by your everyday or even professional photographer. And should a consumer want
those dazzling sky photos you can believe the cost was magnitudes higher than what you can expect now. Many of the higher-end drones such as 3DR solo used in the experiments of this paper come capable of a wide range of video and photo options and plenty of accessories and add-ons that even the most avid videographer/photographer might feel is a bit excessive, although cool nonetheless.

The possible applications of work for drone technology listed above are just a few of the areas where the work we conducted could be utilized. Any consumer, business, or government entity that utilizes this technology will need accurate and effective ways to fully get the best performance available out of their UAV, given its capabilities. The work we are conducting would indeed offer a solution to that problem, which is that an effective and accurate system does not currently exist. The models we develop will help to give all users a better understanding of the constraints of their UAVs and will hopefully stem further research in this field as drone technology begins to rapidly seep into our everyday lives and even starts to become a life saving technology such as in the case of the military and medical supply transport.

Related works

The application of sensing technologies to remote monitoring, and in particular to UAVs, has been widely investigated in the recent years [12-18]. These works show that remote sensing can provide relevant and accurate information in several application domains. Other works focus on the sensing hardware of UAVs and develop techniques to infer the features of interest given the sensor data acquisition [4], [18], [17], while in [13], [16] a complete UAV based monitoring system is proposed. The coordination of multiple autonomous UAVs in the context of remote monitoring has also been recently considered [15], [19], [20-22]. These works provide autonomous coordination and path planning algorithms for a team of UAVs patrolling an area. Although UAVs have attracted significant attention from the research community, little effort has been given to understand how system parameters such as payload and path characteristics
may affect the performance of the UAV, and its capability of completing a monitoring mission. The main constraint in UAV systems is power consumption. As discussed in [7] and [8], the thrust required to keep even a small UAV aloft has a disproportionately large requirement for power when compared to a number other system components. Only few works address the problem of modeling UAV operations [12], [23]. However, these works are strongly based on a detailed knowledge of the geographical landscape where the UAV is required to operate.

In this paper we provide thorough characterization of how physical, mechanical, and electrical hardware aspects of a UAV affect its performance and ultimately the capability to accomplish a mission. This is critical to maintaining both physical systems operational reliability and safety, as well as to design optimization algorithms for UAV operation and validate them through realistic simulations.
Installation

In order to develop code for the solo 3DR drone you must install the solo CLI (command line interface) tool [26]. This tool provides several important tasks for the solo including: enabling simultaneous WiFi access to Solo and the Internet, updating the firmware on Solo and the Controller, downloading logs, etc… The main purpose of this application being to control the Solo and Controller when connected to their WiFi (the Solo drone produces its own WiFi network that devices can connect to in order to interact with one another).

However, before installing the solo CLI, python and pip need to be installed before hand. To install python navigate to the python website (“https://www.python.org/downloads/”) and download python 2.7.*. There is a newer version available, 3.6.*, which has had a number of stable releases over several years but is still under active development. Therefore this project was developed in 2.7 in order to be compatible with all necessary modules and libraries. After the python file has been downloaded it should be fairly straight forward as to how to install the files in the desired directories.

As for installing pip for python, there are several ways that one could go about doing this but the method I found to be the most useful was the following [27].

2. Run python get-pip.py. This will install or upgrade pip. Additionally, it will install setuptools and wheel if they’re not installed already.

After python and pip are installed on your machine, now the CLI tool can be installed. Simply connect to a Wifi network with Internet access and run the following command

    pip install git+https://github.com/3drobotics/solo-cli
For Linux based environments you may need to run as root which would be the following command

```bash
sudo -H pip install git+https://github.com/3drobotics/solo-cli
```

After the setup is finished, the CLI tool should now be available for use. We can now use this to run our scripts to the Solo. The first step we must follow is to package our script with the following command

```bash
solo script pack
```

This command will package all the files in the current directory in a format readable for our drone and produce a .tar file once finished. The pack command also packages any dependencies listed in the current directory’s requirements.txt file, a required file for a successful upload. A minimal requirements.txt for a dronekit app will look similar to the following one line

```text
dronekit>=2.0.0
```

After the .tar file has been produced, we can now upload and run our scripts onto the Solo with the following command

```bash
solo script run yoursurname.py
```

The desired script should now be running.
My portion of work in this research was solely implementation, as opposed to theorizing and coming up with ideas. I will detail my implementation steps in detail below in a chronological order of the things I did.

My work on this project originally started after Dr. Simone Silvestri agreed to let me work on extending one of his papers entitled “Realistic Models for Characterizing the Performance of Unmanned Aerial Vehicles.” He had me start by reading the paper for comprehension and then reaching out to the paper’s co-authors in order to get a better understanding of the project and to ask any questions I had developed over our few weeks of correspondence.

My initial contact with first co-author, Ken Goss, was that he was the main person behind the gathering of the data in the paper and therefore responsible for a majority of the figures and tables included in the paper. Which at first glance, I did not assume to be a crucial part of my work, but I would soon learn otherwise. The second co-author on the paper, Riccardo Musmeci, was the person behind much of the code and configurations of the drone. This left Dr. Simone Silvestri to be the man behind everything else, as well as the project supervisor, offering his expertise throughout the entire process. My role in this team would be to extend the paper and add several more tests that could better define the characteristics of the drone such as how altitude levels or temperature affected the performance of the drone. A task I was more than happy to take on because I really wanted my masters project to be about getting hands on and writing code, rather than theorizing or working on abstract ideas.

My first few months of working with Dr. Silvestri and his team [around the beginning of Fall semester] was all about getting up to speed, climbing a steep learning curve of interacting with the drone interface and brushing off my python skills. I started off by working the with drone simulator and hacking together scripts Riccardo had made to get the drone to do simple things such as take off, land, fly in a pattern, and traverse a set of waypoints. These scripts took quite a bit longer than I expected to successfully run on the drone as the interface called for very specific parameters to be set into place and threw error messages that offered me very little insight into
what my actual problem was. With the help of Riccardo and some online documentation I was able to get my scripts running on the solo (the drone used in this paper) simulator. After getting the scripts running on the simulator I was able to change minor parts of the code to get them to run on the physical drone, which I tested several times at a local park. Once I got some initial testing done on the drone it became time for me to finally start doing the extensions to the paper I was assigned to do.

I started by creating a script that calculated predicted flight models given the basic model in the paper and the power consumption data table found on page two. This script contains several functions of interest but the main ones I focus on are “flightTime”, which calculates the total approximate flight time for a given mission and “numOfTraversableWayPoints”, which calculates how many points are reachable in a mission given data about power consumption of the drone. The code for these functions are available in the appendix.

The function, “flightTime(flightPlan,speed,numPoints)”, takes in three parameters, the flight plan (or mission), the speed of the drone, and the number of points that can be reached in the flight plan [therefore it is assumed that you call this function after you have called the function numOfTraversableWayPoints in order to figure out how many points can be reached for the given flight plan]. It then calculates the delta’s in the X, Y, and Z planes between pairs of coordinates in the drone, divides by the speed, and finds the time it should take the drone to travel in the horizontal and vertical directions separately and then adds them together. The function does this for every pair of reachable coordinates in the flight plan. A brief latency for turning of a quarter second is added to each calculation. The total time to traverse all of the reachable points in the flight plan is then returned to the caller.

The function, “numOfTraversableWayPoints(flightPlan,speed,batteryUseHorizontal,batteryUseVertical,batteryLevel)”, takes in 5 parameters. The current flight plan, the speed of travel for the drone, the battery consumption for horizontal movement at said speed, the battery consumption for vertical movement at said speed, and the battery level once the mission starts [since depending on the
variability in take off this will be different for different altitudes]. This function determines an approximate maximum number of waypoints that can be reached in a given flight plan given data about power consumption. It works by calculating the delta’s in the X, Y, and Z planes multiplying by their respective directional consumption stat, and then adding the horizontal and vertical consumptions together to get a total consumption to travel between a pair of coordinates. Next, we check if the battery level can sustain traveling to that next point (ie having a battery level greater than 10 after traveling because once at 10% the drone will automatically return to its original take off position). If we can travel to the next coordinate safely then we do so, subtract the total calculated consumption from the battery level, and add one to a count of points we can reach, otherwise we determine we cannot safely reach the coordinate and return our current number of waypoints we could reach.

Since these functions assume that the traveling speed of the drone is consistent and does not account for speed up, speed down, or the drone never reaching that speed during the entire test we notice the results are not accurate but this follows our expectations with a consistent velocity model.

**Analysis**

After the scripts were created and I showed the results to Dr. Silvestri, we noticed a big problem. It seemed that the power consumption data I had been using, the one from the original paper was not giving up the results we expected and made us question the accuracy of the data. I was using a brand new 3DR solo drone, with 2 fresh new battery, which we suspect to be the reason why my results were not coming out as expected. It was decided by Dr. Silvestri at that point that I should rerun the power consumption tests to see what data I would gather from the new drone and compare my results to from those gathered by Mr. Goss. This was my new assignment and turned out to be a little bit more challenging than I anticipated. Creating the scripts for the new assignment was not that difficult given the initial work I had already done with the drone. I sent the new scripts through the simulator and they worked
exactly how I expected them to after a few minor debugging sessions. The real problem arose once I began testing the scripts on the physical drone. I had a variety of scripts to test all the scenarios I thought I would need, so I created scripts for the speeds in the original experiment of 8 m/s, 9 m/s, and 13 m/s and then for each speed a variety of distances including 50 meters, 75 meters, 100 meters, 125 meters, and 150 meters and then did this for both the vertical and horizontal directions. When testing in the field I ran into problem of either the scripts would only traverse one point or not traverse any. I spent weeks trying to figure out why this was an issue and scoured the internet for help. Unfortunately, there is very little help, forums, and documentation for troubleshooting errors with the drone, especially errors which produce no error codes.

While on the brink of mental fatigue and wanting to give up, I decided to check my scripts with those Riccardo had wrote for the drone when he was testing it. At this point our scripts were vastly different and I had already looked over them a half of dozen times before, but on this last check I noticed something different. My frustration had led me to be extremely nit-picky and scrutinous which caused me to notice a difference between the scripts which I originally thought was insignificant, the mode of the drone. I was following examples from online documentation and had long since deviated from what Riccardo had done, as it wasn’t very relative to what I was now doing. In these documentation examples, they used the “GUIDED” mode on the drone, which allowed me to query about the drone in-flight and gave me the most control, which I thought would be useful for getting speed averages and establishing breakpoints when it was time to quit the script early. However in the scripts Riccardo had developed, he used the “AUTO” mode on the drone. At this point in time, I was not aware of the differences between the two and decided to investigate as thoroughly as I could. I then figured out that in order to run my scripts in the proper fashion I would need to use the AUTO mode which pre-uploaded the flight plan to the drone and went through each point continuously without interruption, even though this would take away my ability to query and control the drone in-flight but this would be my solution and allow me to gather the results I needed.
After, this round of testing I now had to do the exact same thing but with a payload of approximately 300g (precisely 322g), which was a GoPro Hero 4 Silver attached to the solo gimbal and installed on the undercarriage of the drone. I expected this portion of testing to go smoothly since the only thing I was changing was stalling the gimbal, not even using it, I quickly found out I was mistaken. Attaching the gimbal to the drone created a new problem that I could not fix and still to this day do not know a proper solution. What happened was that when the gimbal was installed on the drone I repeatedly got an error claiming that there was no detectable heartbeat from the drone in the last 30 seconds and my script would not run. With an error like this I expected the drone to have shut down or it had lost signal but those were not the problems. I tried several different ways of solving the problem but nothing worked. I tried searching the internet for help or at least a clue but nothing there helped either. I even tried resetting the drone, which also did not work. This error quickly became my most challenging one. Not only for the lack of resources of help, but also because it was an inconsistent error. It only happened sometimes and I was never able to predict when it would or would not happen. I managed to find an unreliable work around to this problem which was that I manually start the drone, then start the script and immediately return the drone to the ground before the actual script started and while the drone was still loading in modules. I did this in hopes that the script would detect a heartbeat from the drone just being active and in a way, manually force a heartbeat into the system. As I mentioned, this was a very unreliable work around and just as unpredictable as the error that I was trying to avoid. This single error would made gathering data for the 300g payload table almost nearly impossible and very time expensive. But after a week or two of back and forth with the drone I was able to gather enough data for this portion of testing.

The final piece of work I have been able to complete up until the writing of this paper was rerunning the test from figure 9 on page 6 of the original paper. Dr. Silvestri advised me that since we gathered all new data from the drone it would be necessary to rerun these a variety of tests from the paper. The first being the test conducted for figure 9. In this figure we have the following graph:
This graph shows the relation between the mass and the flight time, showing a direct correlation between increasing weight and decreasing flight time.
Results

As previously stated I had created variety of scripts to test all the scenarios I thought I would need, for the speeds in the original experiment of 8 m/s, 9 m/s, and 13 m/s and then for each speed a variety of distances including 50 meters, 75 meters, 100 meters, 125 meters, and 150 meters and then did this for both the vertical and horizontal directions. I was able to collect all the data I needed to get the following tables for no payload power consumption in the horizontal and vertical directions.

### Horizontal

<table>
<thead>
<tr>
<th>Speed</th>
<th>Battery use (%/m)</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast (13 m/s)</td>
<td>0.009761904762</td>
<td>0.00007174603175</td>
</tr>
<tr>
<td>Medium (9 m/s)</td>
<td>0.01155555556</td>
<td>0.00008503703704</td>
</tr>
<tr>
<td>Slow (8m/s)</td>
<td>0.013</td>
<td>0.000097333333333</td>
</tr>
</tbody>
</table>

### Vertical

<table>
<thead>
<tr>
<th>Speed</th>
<th>Battery use (%/m)</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast (13 m/s)</td>
<td>0.0366666666667</td>
<td>0.01154700538</td>
</tr>
<tr>
<td>Medium (9 m/s)</td>
<td>0.03333333333</td>
<td>0.005773502692</td>
</tr>
<tr>
<td>Slow (8m/s)</td>
<td>0.0366666666667</td>
<td>0.005773502692</td>
</tr>
</tbody>
</table>

Which we can see is very different from the original tables in the paper shown below:

### Horizontal

<table>
<thead>
<tr>
<th>Speed</th>
<th>Battery use (%/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast (13 m/s)</td>
<td>0.015</td>
</tr>
<tr>
<td>Medium (9 m/s)</td>
<td>0.028</td>
</tr>
<tr>
<td>Slow (8m/s)</td>
<td>0.031</td>
</tr>
</tbody>
</table>
One of the most notable differences between the data I collected and the original data from the paper is the vertical direction table. In the table I produced from the data I collected, you will notice that all of my values are roughly the same for all speeds. This is because when I ran the experiments I noticed from the 3DR solo app, which I had connected to a Samsung Galaxy S7 for in flight monitoring purposes, that regardless of what I programmed the speed to be in the vertical direction, it always traveled at consistent speed of 1.5 m/s to approximately 2 m/s. I consulted with the previous authors of the paper to see if they had a solution to this problem. After some emails back and forth and testing, it turns out that the firmware on the drone actually needs to configured to allow a vertical velocity change, and even then system parameters would still override this sometimes. I was not able to manipulate the firmware in time for the data gathering process.

After, this round of testing I now had to do the exact same thing but with a payload of approximately 300g. The results I collected are below:

<table>
<thead>
<tr>
<th>Vertical</th>
<th>No Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Battery use (%/m)</td>
</tr>
<tr>
<td>Fast (13 m/s)</td>
<td>0.047</td>
</tr>
<tr>
<td>Medium (9 m/s)</td>
<td>0.068</td>
</tr>
<tr>
<td>Slow (8m/s)</td>
<td>0.083</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>300g Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Battery use (%/m)</td>
</tr>
<tr>
<td>Fast (13 m/s)</td>
<td>0.015</td>
</tr>
<tr>
<td>Medium (9 m/s)</td>
<td>0.015</td>
</tr>
<tr>
<td>Slow (8m/s)</td>
<td>0.01777777778</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vertical</th>
<th>300g Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Battery use (%/m)</td>
</tr>
<tr>
<td>Fast (13 m/s)</td>
<td>0.06904761905</td>
</tr>
</tbody>
</table>
Which we can see is also different from the original tables in the paper, shown below:

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>300g Payload</th>
<th>Battery use (%/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Fast (13 m/s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Medium (9 m/s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slow (8m/s)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vertical</th>
<th>300g Payload</th>
<th>Battery use (%/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Fast (13 m/s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Medium (9 m/s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slow (8m/s)</td>
</tr>
</tbody>
</table>

You will notice that in the vertical table I collected, that this time there is only one row of data for “13 m/s.” This is done for two reasons, one because of the sheer difficulty I was experiencing with that error in trying to collect data with the gimbal attached and two because as of the reason pointed out in the last table. Programming the speed into the script and therefore into the UAV did not affect the speed in the vertical direction whatsoever, therefore I did not find it beneficial to continue trying to test the two other speeds with the gimbal attached.

The final piece of work I had to do was rerunning the test from figure 9 on page 6 of the original paper. When I duplicated the testing for this figure I consistently got similar results as shown by the black line on the figure, therefore there were no changes that needed to be made to the original figure since my results turned out to be the same.
My next tasks were to rerun the tests for figures 10, 11, and 12, and then I could finally start on the original task of expanding the paper. This will be covered more in detail in the next section.
Future Work

This section will detail the things I still need to do in order to feel “done” and accomplished with this assignment. As mentioned very briefly at the end of the last section I still need to rerun the test from figures 10, 11, and 12.

Figure 10 is depicted below:

![Figure 10](image)

This figures shows the correlation between mass and horizontal flight time. As you can see, the model approximation follows what one would logically expect for a test of this nature which is as mass increases the horizontal flight time decreases. But in reality we see there is a slight increase in around the 800g mark before dipping off again. To rerun this test I assume I will follow a similar fashion as when I reran the tests for figure 9.

The next tests to rerun will be figures 11 and 12, which are shown below:
These figures show the relationship between the vertical and horizontal oscillation amplitudes and number of way points reached. As with the previous figures, one would expect a direct correlation which is as the amplitude increases the number of way points decrease. This is what our model predicts and it is what actually happens although the dips are not completely linear. Performing the tests for these figures will be very similar to the tests ran for figures 9 and 10 except I will basically count the number of successful points traversed over a particular amplitude set at a weight of 300g.
After these tests I can complete my task of editing the bar plot, figure 15 on page 8 (this plot is shown below).

![Bar plot](image)

Figure 15. Empirical Flight Time comparison with Model Lower Bound (LB)

For this plot I will run a script I have created that will take a flight plan with similar statistics as the original (since I do not have the original flight plan), an average displacement 8.47m and an average horizontal displacement of 13.27m, and run until 90% of the battery has been depleted. I can then record the flight time and add another bar for the data that I collected.

And finally, after all that, I could finally get back to the original assignment which was extending the paper. For this I would create two graphs, similar to figures 9 and 10 that would look like the following:
Notice I am still using the flight time as the dependent variable but now I have altitude and temperature as the independent variables, respectively.
References


[27] Python. Available at: https://packaging.python.org/tutorials/installing-packages/
# Appendix

A

# calculates total flight duration for a given mission

def flightTime(flightPlan, speed, numPoints):
    currentTime = 0
    # print "currentTime: %s" % currentTime

    for i in range(0, numPoints):
        currentPoint = flightPlan[i]
        nextPoint = flightPlan[i+1]
        # print "currentPoint: (%s,%s,%s)" % (currentPoint.x, currentPoint.y, currentPoint.z)
        # print "nextPoint: (%s,%s,%s)" % (nextPoint.x, nextPoint.y, nextPoint.z)

        deltaX = abs(nextPoint.x - currentPoint.x)
        deltaY = abs(nextPoint.y - currentPoint.y)
        deltaZ = abs(nextPoint.z - currentPoint.z)
        # print "deltaX,deltaY,deltaZ: %s,%s,%s" % (deltaX, deltaY, deltaZ)

        timeHorizontal = (deltaX + deltaY) / speed
        timeVertical = (deltaZ) / speed
        # adding 1 second for latency and turning
        timeTotal = timeHorizontal + timeVertical + .25
        currentTime += timeTotal
        # print "timeHorizontal,timeVertical,totalTime: %s,%s,%s" % (timeHorizontal, timeVertical, timeTotal)

    takeOffAndLandTime = 35
    return currentTime
#calculate the number of traversable waypoints for a given mission ("flightplan")
#function runs until battery reaches approximately 10%. This is when solo would automatically
# begin returning home due to low battery power

def numOfTraversableWayPoints(flightPlan, speed, batteryUseHorizontal, batteryUseVertical, batteryLevel):
    pointsTraversed = 0
    batteryLife = batteryLevel
    for i in range(0, (len(flightPlan)-1) ):
        currentPoint = flightPlan[i]
        nextPoint = flightPlan[i+1]
        deltaX = abs(nextPoint.x - currentPoint.x)
        deltaY = abs(nextPoint.y - currentPoint.y)
        deltaZ = abs(nextPoint.z - currentPoint.z)
        batteryConsumptionHorizontal = (deltaX + deltaY) * batteryUseHorizontal
        batteryConsumptionVertical = (deltaZ) * batteryUseVertical
        totalConsumption = batteryConsumptionHorizontal + batteryConsumptionVertical
        if(batteryLife - totalConsumption <= 10):
            return pointsTraversed
else:
    pointsTraversed = pointsTraversed + 1
    batteryLife -= totalConsumption

return pointsTraversed