Simulating GPS-denied Autonomous UAV Navigation for Detection of Surface Water Bodies

Arnav Deo Singh
Queensland University of Technology
Brisbane, Australia
arnav182@gmail.com

Fernando Vanegas Alvarez
Queensland University of Technology
Brisbane, Australia
f.vanegasalvarez@qut.edu.au

Abstract—The aim to colonize extra-terrestrial planets has been of great interest in recent years. For Mars, or other planets to sustain human life, it is essential that water is present and accessible. Mars may contain or provide signs of the possibility of near surface water, but this work will only focus on surface water for the purposes of simulation. In this paper, we present a method for autonomous navigation and detection of surface water bodies in the GPS denied environments of Mars via a fully autonomous UAV. A combination of existing state-of-the-art tools and techniques have been utilized to enable the development of this system. Additionally, we create a modular framework to simulate the mission using the AirSim simulation environment and the Robot Operating System (ROS). The simulation environment is leveraged by using the Unreal game engine running in Windows OS, interfacing with an open source software for simultaneous localization and mapping (SLAM), running on ROS and Linux OS. A simulated mission was successfully implemented and demonstrated using the framework. Results obtained indicate that the framework enables the navigation of a UAV in the simulated Mars environment and allows the UAV to detect surface water bodies. The developed simulation framework, along with the knowledge and techniques attained in this research, could accelerate the development, testing and deployment of missions for a real-world Mars UAV, for the detection of surface water bodies. Additionally, this research aims to support and build upon prior work to further aid the search for water on other planets, as well as assisting humans in becoming a multi-planet species.

Keywords—Mars UAV, surface water detection, UAV navigation in GPS denied environments, Mars exploration, space exploration, SLAM, autonomous mission, simulation

I. INTRODUCTION

The context of this research project and its application relates to Target Acquisition missions on Mars; more specifically, the problem of searching for water on Mars via a fully autonomous UAV. Mars has been of great interest in the Aerospace industry in the last few years. As an example, private companies such as SpaceX are investing in development and technology with their desire to colonize the red planet. If humans really were to colonize Mars, it would be vital that a viable source of water is available.

According to NASA [1], water is essential for life to exist; hence, a target acquisition system focused on detecting bodies of surface water in a GPS-denied environment such the one on Mars is of great importance.

NASA has been developing a UAV for the exploration of Mars and is aiming to launch it with their new rover in 2020 [2]. For the purposes of this research project, it will be assumed that this UAV is in the form of a quadcopter, is suitable for operating in the conditions of Mars, and has the same control dynamics as the ones used on Earth.

UAV and multispectral cameras have been previously used in plant protection and farming for detecting specific diseases or stress levels [3]–[6] and thus could be used in missions for searching for water on Mars.

The research question for this project was to investigate how to develop a fully autonomous UAV to navigate the GPS denied environments of Mars, in the search for bodies of surface water. This question can be divided into two sub-questions as follows:

1. What techniques will be used to develop a target acquisition system capable of detecting bodies of surface water?
2. What methodologies will be utilised to enable fully autonomous navigation in the GPS denied environments of Mars?

Since it was not feasible to actually build and test this on Mars at the time of writing this paper, the development and testing was conducted using the AirSim simulation environment that was developed by Microsoft. AirSim is a simulator and platform for experimenting with Artificial Intelligence (AI) techniques such as Deep Learning, Computer Vision and Reinforcement Learning algorithms for autonomous vehicles. It supports Hardware-In-The-Loop (HITL) with supported devices and provides physically and visually realistic simulations because it is developed as an Unreal Engine plugin [7].

A summary of the literature review on prior work will be presented and will focus on the following:

- Detection of bodies of surface water – Mars may have near surface water or provide signs of the possibility of near surface water, but for the purposes of developing this fully autonomous UAV in simulation, only surface water was explored.
  - The types of sensors used for detection
Machine Learning techniques for water detection

- Modelling sensors for simulation

- Navigation in GPS denied environments since GPS is not available on Mars

- Visual Odometry and SLAM techniques
  - Feature based and direct SLAM

- Motion and mission planning using Partially Observable Markov Decision Process (POMDP), since the objective of the drone is to maximise its chance of locating a source of water while in a partially observable environment

- UAV motion control using AirSim

II. LITERATURE REVIEW

A. Target Acquisition

a) Detection of Surface Water Bodies

Water on the surface of a planet is referred to as surface water; for instance, rivers, lakes and oceans would be examples of surface water. There are two categories of sensors that can detect surface water: microwave sensors and optical sensors. Microwave sensors can operate under any weather condition and penetrate cloud and vegetation coverage due to its long wavelength. Optical sensors (i.e. multispectral/hyperspectral) on the other hand, are widely used because of the abundance of data and the appropriate spatial and temporal resolutions; that is, the number of pixels utilised in construction of the image [8], and the time duration for acquiring a single frame of a dynamic process [9], [10].

Since hyperspectral sensors have a larger number of bands that are narrow and provide more much more detail than multispectral sensors, it would be more suited for applications that measure water quality rather than detection of water [11]. For this reason, many related works have utilised the multispectral sensor.

To detect water from a multispectral image, the reflectance property of water must be utilised. An effective method is to use water indices which are determined from two or more bands; this allows for the differentiation between water and non-water bodies. Various indices for surface water detection exist, however, the mNDWI water index is said to be more reliable and results in more accurate detection of surface water [10]. The mNDWI index is a modified version of the Normalised Difference Water Index (NDWI) and uses the Short-Wave Infrared band (SWIR) because it is less sensitive to optical elements within water (e.g. sediment concentrations) [10]. The equation for mNDWI is as follows:

\[
m_{\text{NDWI}} = \frac{(\text{Green} - \text{SWIR})}{(\text{Green} + \text{SWIR})}
\]

Where Green and SWIR represent the spectral reflectance measurements in the visible green band and short-wave infrared band. If there is an absence of SWIR band in the multispectral image, the NDWI index can be used as it utilises the Near Infrared (NIR) band. An issue with mNDWI however, is the fact that it cannot distinguish between water and snow. To fix this limitation, the mNDWI index should be used in conjunction with a visible/NIR band [10].

b) Machine Learning Techniques

Machine Learning (ML) is the concept of enabling a computer to learn without being explicitly programmed to do so. Machine Learning can be divided into two categories; supervised learning (SL) and unsupervised learning (UL). Supervised learning is when a machine is trained with labelled data, meaning that it learns from data that is already identified. Once the machine has trained with the labelled data, test data is fed into the system and the algorithm produces a result based off the knowledge it has acquired from training [12].

Unsupervised learning is when the machine is provided with unlabelled data, meaning that the data is unidentified, and the unsupervised learning algorithm has to identify trends or similarities in the data to produce an outcome/result without any prior training.

The task of detecting water from multispectral images falls into the category of classification because it can either be detected as surface water or not surface water. Classification algorithms in SL can be used to enable the machine to learn how to detect surface water from multispectral images; or, clustering algorithms in UL can also be used.

A classification algorithm that has been used by [13] and [14] to detect water from multispectral images is the Support Vector Machine (SVM). When labelled training data is provided to the SVM, an optimal hyperplane which classifies the data into two categories is determined. A hyperplane is essentially a line that divides the data into two categories. The dimension of the hyperplane is dependent on the number of input features; for example, if the number of features is two, the hyperplane is just a line. When its three, it becomes a two-dimensional plane [15].

The data points that are in close proximity to the hyperplane are the support vectors. These influence the position and orientation of the hyperplane and are used to maximise the margin between the data points and the hyperplane.

c) Modelling Multispectral Sensor for Simulation

Since the project needs to be developed in a simulation environment, the multispectral sensor will need to be modelled. Bondi et al. [16], have modelled a thermal IR sensor in the Airsim simulation environment for an Africa savanna landscape. Due to the complexity...
of modelling the physical phenomena, Bondi et al. [16] have assumed the following:

1. Assuming the atmosphere is clear, dry and cool, upwelled radiance and down welled radiance are negligible.
2. Due to majority of the terrain being flat in a savanna, background radiance is negligible, meaning that most of the contribution of thermal IR photons would be directly from the object.
3. Due to direct contribution, the effects on thermal IR photons from the atmosphere are not considered.
4. Assume that objects are Lambertian; meaning they emit light uniformly.
5. Camera lens has perfect transmission and no falloff. This was said to be false, however, it can be accounted for in the future.

From these assumptions, the direct contribution of photons in the IR band can be modelled by using Planck’s Law:

\[ L(T, \varepsilon_{avg}, R_A) = \varepsilon_{avg} \int_{\lambda_1}^{\lambda_2} R_A \left( \frac{2hc^2}{3\lambda^5} \frac{1}{\exp\left(\frac{hc}{kT\lambda}\right)-1} \right) d\lambda \]

(2)

Where \( L \) is the radiance, \( T \) is the temperature in Kelvins, \( \varepsilon_{avg} \) is the average emissivity of the band, \( R_A \) is the peak normalized spectral response of the camera, \( h \) is Planck’s constant, \( \lambda \) is the wavelength, \( k \) is the Boltzmann constant and \( c \) is the speed of light.

B. Navigation in GPS Denied Environments

a) Localization and Mapping Techniques

Since Mars is a GPS denied environment, the UAV will have to localize itself. There are multiple ways this can be accomplished. Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) are the two major techniques for localization in GPS denied environments. VO is the process of determining the pose of a robotic vehicle in real-time using only the input of single/multiple cameras attached to it. SLAM on the other hand, is a technique which allows a robotic vehicle to localize itself and simultaneously create a map of its environment with the use of sensors. The benefit of using SLAM is that once a map of the environment is created, it is easier to localize the vehicle. In VO however, no map is created, hence, the vehicle has no way to remember visited locations and therefore has to redo the process. For this reason, the research project will focus on SLAM [17].

SLAM can be divided into two categories; feature based and direct SLAM. Feature based SLAM is a technique derived from the concept of extracting and matching feature points; whereas direct SLAM is a process which involves image alignment [18]. Feature based SLAM is faster than direct SLAM, is robust to inconsistencies/outliers in the system and does not need to be initialised well. However, it can only use and reconstruct feature points, while direct SLAM can use and reconstruct the entire image [18].

Since feature-based SLAM requires less computing power and has these advantages over direct SLAM, the project will focus on feature-based SLAM techniques. Furthermore, the Mars UAV will need to be compact and light weight, plus, the map won’t be used to detect a target. Creating a full map will only waste resources and computing power. [19], [20].

A feature-based SLAM system known as VINS-mono, is a monocular visual-inertial state estimator that is robust and versatile. The algorithm utilises a tightly coupled sliding-window nonlinear optimization method, and fuses pre-integrated IMU measurements and feature observations to attain high accuracy VIO data [21]. The main benefits that VINS-mono provides are:

- Four Degrees of Freedom (DOF) global pose graph optimisation
- Real-time performance capability for drone navigation
- Loop detection

The algorithm’s loop detection module allows re-localisation to be performed by fusing feature observations with VIO data, enabling accurate re-localisation with low computational overhead [21].

The final enhancement phase of any SLAM system is loop closure detection. To attain a globally consistent SLAM solution when localizing and mapping over extended periods of time, loop closure is required. The process of diminishing the accumulated drift in the pose estimate by identifying same scenes in non-adjacent frames and adding a constraint between them, is known as loop closure [17].

b) Motion and Mission Planning Using POMDP

The sequential decision process of detecting water on Mars can be modelled as a Partially Observable Markov Decision Process. The mathematical framework to model sequential decision processes under state, transition and observation uncertainty is known as the POMDP [22], [23]. It consists of a set of elements \{S, A, O, T, Z, R, \gamma, b\}, where S is the set of states in the environment; A is the set of actions an agent (Mars UAV) can execute; O is the set of observations; T is the transition function amongst states after performing an action; Z is the distribution function which defines the probability of observing \( o \) from state \( s \) after performing action \( a \); R is the collection of rewards for every state; \( \gamma \) is the discount factor and \( b \) is the belief state.

The POMDP framework can be utilised to maximise the drone’s chances of locating a source of water in a partially observable environment by defining appropriate reward functions. Due to the time constraints of this project, utilisation of the POMDP will be out of scope. However, it will be required if further research and development were to occur [24].
c) UAV Motion Control Using AirSim

The AirSim simulation environment consists of a default built-in flight controller called simple_flight. It was designed to work on real UAVs as well as in simulation, and comes with the benefit of zero additional setup. Furthermore, it is located within the same codebase, so debugging becomes easier. AirSim also supports the PX4 flight stack for HITL applications, however, it requires additional configuration and expertise to use [7]. For this reason, AirSim’s default flight controller was used in the development of the simulation framework.

III. METHODOLOGY

A. Target Acquisition System

a) Modelling of Multispectral Sensor

As mentioned in the literature review, multispectral sensors are used to detect bodies of water. To enable the detection of surface water in the simulated Mars environment, a multispectral sensor needs to be simulated. The modelling was based off the work done by Bondi et al. [16] and is rudimentary. Similar assumptions were also made to be able to use Planck’s law. Most of these assumptions are invalid and should be catered for once the foundation of the project has been developed. From these assumptions, the direct contribution of photons in multiple bands of the electromagnetic spectrum can be modelled by using Planck’s Law (eq. 2).

To calculate the radiance of an object; that is, the amount of light the sensor "sees" from the object being observed [25], the temperature of the object is required. To make the project more realistic, it was decided to find real Mars weather data and feed it into the simulation. This was done by scraping data from Insight lander’s weather site (https://mars.nasa.gov/insight/weather/). The retrieved data included average atmospheric temperature readings as well as many other values which could be used in future development.

The code for the modelled sensor was based off the work done by Bondi et al. [16] and was modified to calculate the radiance in multiple bands (blue, green, red, NIR and SWIR) for various objects in the Mars environment. The temperature of the surface water was assumed to be equal to the average atmospheric temperature retrieved from Insight landers weather site. This is not correct, but it at least provides some reference as to what it may be if it existed.

Once this was done, the water index was calculated. For this, a custom water index was utilised; the formula for this is presented in Eq. 3:

\[
customWaterIndex = \frac{(NIR-SWIR)}{(NIR+SWIR)}
\]

The reason for using a custom water index was because the mNDWI index did not work; spectral reflectance values are required instead of radiance values. However, a method to convert radiance to spectral reflectance was not found and simulating spectral reflectance was not plausible in the time frame available. This will be further discussed in later sections.

The process for using the modelled sensor is to start the simulation and then run the script. After the water indices are calculated, the script assigns a digital count to each object in the environment and this is then set as the stencil ID in Unreal Engine. All other objects except for water and rocks are set to 0 since they are not of interest. These IDs are used to determine the colour in the RGB bands to represent the multispectral image as an RGB image. To make it easier to view the feed from the sensor, a custom camera view was developed for the AirSim plugin, based on the other views.

b) Detection of Surface Water Bodies from Simulated Multispectral Data

To enable automatic detection of surface water from the simulated data, the use of machine learning algorithms such the support vector machine would be used. However, due to this being a rudimentary model of the multispectral sensor and other limitations being present, a workaround solution was used. The workaround was to basically detect when the water texture was recently rendered, and then return a Boolean value. However, this method was not used in the final solution as Unreal Engine would render the water texture even if it wasn’t in direct sight. Hence, further work will need to be done for this subsystem in the future.

B. Navigation System

a) System Architecture

VINS-mono requires ROS and Linux to operate, which introduced some challenges for integrating VINS-mono with AirSim. This is because AirSim requires Unreal Engine and is most stable on the Windows OS; however, VINS-mono requires Linux and ROS.

Microsoft has released a tool called the Windows-Subsystem-For-Linux (WSL); basically, Windows now has a built-in Linux kernel thanks to Microsoft’s Hyper-V technology [26]. From the Microsoft store, one can download many distributions of Linux, including Ubuntu 16.04, and run it natively on Windows. For GUI tools such as RVIZ, a software known as MobaXTerm can be used; basically, it’s a remote desktop application which uses X11 technology to show GUI applications from a terminal interface. A flow diagram describing the methodology used for simulating a full autonomous mission for surface water detection and navigation in GPS-denied environments is presented in Figure 1. A diagram describing the modular framework and the complete system architecture is shown in Figure 2.
b) VINS-mono Setup

To test the VINS-mono algorithm, the demo image publisher from AirSim’s ROS package was used. Along with this, an IMU publisher was created in Python. The image and IMU topics were then referenced in VINS-mono’s configuration file. Camera calibration parameters were also needed in the configuration file, but this was left to the default values because this information would need to be obtained via a virtual camera calibration in Unreal Engine. Even though this would result in inaccurate pose information, this was done just to test if the system was working.

c) Virtual Camera Calibration

In order to use VINS-mono correctly, virtual camera calibration needed to be performed. The first step was to obtain the intrinsic parameters for the AirSim camera. Following the formulas provided in [27] and [28], the intrinsic matrix for a resolution of 1280 x 720p turned out to be:

\[
K = \begin{bmatrix}
640 & 0 & 640 \\
0 & 1 & 0 \\
360 & 360 & 1
\end{bmatrix}
\] (4)

The next step was to develop a virtual checkerboard environment to obtain all intrinsic parameters for the virtual camera. This was difficult because the physics engine limits the amount of rotation possible on a UAV; however, a solution was eventually found. A static chessboard mesh was created in Unreal Engine; with each square size being 50cm and the computer vision mode in AirSim was used to rotate the camera without the physics engine for correct calibration. An image of the checkerboard environment is shown in Figure 3:

```
Figure 3 - Virtual checkerboard environment
```

Once this was done, the camera calibration can be performed using the ROS camera calibration package as shown in Figure 4. Please refer to Appendix A for output data from camera calibration.

```
Figure 4 - Virtual camera calibration
```

### 1796
This was able to get VINS-mono to automatically calibrate and estimate the extrinsic parameters for the IMU + camera.

C. Mars Simulation Environment

Originally it was planned to use the Mars 2030 environment developed by FMG Labs and NASA [29]. This environment was based on real Mars terrain data captured from the Mars Reconnaissance Orbiter. Unfortunately, the environment was not able to be ported to the project because it was an asset from a custom version of Unreal Engine. The only other alternative was to purchase a suitable environment from the Unreal marketplace and modify it to suit the project needs (i.e. add surface water etc.). An image of the finished environment is shown in Figure 5.

IV. RESULTS & DISCUSSION

A. Detection of Surface Water Bodies

After running the multispectral sensor script as mentioned earlier in the methodology section, the following data was written to a text file. Please refer to Appendix B for the simulated multispectral data.

From Appendix B, it can be seen that the custom water index for the water texture is 1.6. During testing, it was found that other non-water textures were negative values; however, it wasn’t consistent enough. If the simulated multispectral sensor can be fixed in the future to provide spectral reflectance instead of radiance values, then this data can be passed into the SVM algorithm to automatically detect surface water bodies.

The environment used to test the target acquisition system was an Earth like environment to replicate the testing procedures done in the real world. The goal of the detection system is to detect surface liquid water bodies on Earth. It is not known if liquid water bodies exist on Mars or other planets, but by testing the system on Earth, it will ensure that it is able to detect it if it exists. Figure 6 shows the test environment used; once all systems were finished and tested, then the Mars environment was used as a final demonstration.

Figure 6 - Test simulation environment with water and rock textures

Figure 7 and Figure 8 exhibit the sensor detecting water and boulder rocks. Note, the first sub-view in the figures is a depth segmentation view for testing only. The second sub-view is the live feed from the simulated multispectral sensor.

Figure 7 - Detection of surface water

Figure 8 - Water and boulder rock detection

B. SLAM Results Using VINS-mono

Using the extrinsic parameters estimated by VINS-mono, the following estimated path was generated:
Note, these paths were generated using a resolution of 1920 x 1080p, and a different maneuver for the initial IMU + camera calibration.

From Figure 9, it can be seen that estimated path (blue line) follows the trajectory of the ground truth path (red line), however, it has quite a bit of offset from the ground truth. The calculated RMS error values in the x, y and z coordinates were 28.4726, 28.4202 and 1.8761 meters respectively. There is quite a large error in the x and y coordinates, however the z coordinate is reasonable.

After analyzing the results, it was decided that the IMU + camera calibration was not performed correctly, and that the path should have some loop closures in it. Unfortunately, there was a roadblock in creating the new path. The machine on which the simulation was running began to lag due to the large amount of data being processed (30 Gb). The machine runs Unreal Engine, all the ROS publishers, the mission planning script, WSL, VS Code, ROS bag recorder node, and AirSim RPC server. Running all these processes simultaneously limits the performance of the simulation on a computer with the following specifications (MSI GS65 9SE): Intel core i7-9750H; 16GB RAM; NVIDIA GeForce RTX 2060 and 512GB SSD.

For this reason, the resolution was decreased to 1080 x 720p. This change meant that the intrinsic parameters and extrinsic calibration for IMU + camera had to be redone. This is where the orbit path was used, as mentioned in the methodology section. After the new calibration, the following estimated path was generated:

Note, the UAV was meant to return to origin, adding a final loop closure; however, the VINS estimator crashed, due to the large amount of processing as discussed earlier. Therefore, the path was stopped when it detects surface water.

From Figure 10, it can be seen that estimated path follows the trajectory of the ground truth closer than the first path. Here, the estimated path falls short of the ground truth, but in the first, it overshoes. The calculated RMS error values in the x, y and z coordinates with no loop closure optimisation were 21.1160, 27.9689 and 2.2144 meters. The new calibration and path were more accurate than the first, by 7.3566 meters and 0.4513 meters in the x and y coordinates. However, the z coordinate was worse by 0.3383 meters.

With loop closure optimisation, the RMS error values in the x, y and z coordinates were 30.3348, 44.9588 and 2.9887 meters respectively. This performed worse than no loop closure optimisation and the original path & calibration. The loop closure optimisation is supposed to diminish uncertainty and increase the accuracy of the estimated position; however, it seems to have had an inverse affect in this case.

After thorough analysis of results, it was concluded that the extrinsic parameters for the IMU + camera were not correct; meaning that the calibration of the IMU + camera was not accurate. This issue will be further discussed in the future recommendations section.

A video of a complete mission can be seen at [https://youtu.be/e3d4nUNjXag](https://youtu.be/e3d4nUNjXag)

**C. Limitations**

The current model of the multispectral sensor has some severe limitations. The major limitation of this model is that the standard water index (mNDWI) does not give the expected value (positive value usually above 0.5 for water). It returns a value of -1 for water and lower values for non-water objects. This is the reason why a custom water index was needed. A possible reason for this limitation is that this model does not consider the reflection of light on the object and uses radiance instead. According to [25], the use of the reflectance values will provide more reliable vegetation/water indices. Unfortunately, determining the reflectance value in Unreal Engine seems to be very complex; however, the latest version of Unreal Engine (4.2.2) has support for real time ray tracing. This could be used to calculate reflectance values; this technique has been used before by the researchers who created the SENSOR environment [30]. Currently, AirSim officially supports Unreal Engine 4.1.8, so this will have to be experimented with in the future.

Another limitation to this model, is that it does not account for sensor noise, and uses emissivity values that were calculated between 8 to 14 microns of the IR band. This is another reason why the mNDWI index does not work, as the green band has values smaller than the SWIR band; resulting in negative values for water. Unfortunately, it has been difficult to find the emissivity for water and additional objects in other bands. Solutions/recommendations to these limitations will be further discussed in the future recommendations section.
V. CONCLUSIONS & RECOMMENDATIONS FOR FUTURE RESEARCH

The purpose of this paper was to present the research and development of the Mars UAV for the detection of surface water bodies. The research question for this project was to investigate how to develop a fully autonomous UAV to navigate the GPS denied environments of Mars, in the search for bodies of surface water. Since it was not feasible to build and test this on Mars at the time of writing this paper, the development and testing was conducted using the AirSim simulation environment that was developed by Microsoft.

The developed solution consisted of two main subsystems; target acquisition, which enables the automatic detection of surface water bodies, and navigation, which allows the vehicle to autonomously navigate in the GPS denied environments of Mars. The proposed methodologies/techniques to simulate this mission in AirSim comprised of the following: modelling and usage of a multispectral sensor in AirSim for detection of surface water bodies; utilisation of SVM supervised learning algorithm for detection of surface water bodies within multispectral data; the implementation of the VINS-mono algorithm for navigation in GPS denied environments; usage of the POMDP framework to maximise the drone’s chances of locating a source of water in a partially observable environment, by defining appropriate reward functions (only if sufficient time remained for the project), and utilisation of the simple_flight flight controller in AirSim, to control the Mars UAV.

The methods/techniques that formed the final solution for this research project consisted of the following; a rudimentary model of a multispectral sensor for detection of surface water bodies; the implementation of the VINS-mono algorithm for navigation in GPS denied environments, and utilisation of the simple_flight flight controller in AirSim, to control the Mars UAV.

Simulating spectral reflectance in Unreal Engine to achieve realistic simulation of a multispectral sensor is very challenging and would require further research. Due to this, the SVM learning algorithm was not used because it would not be able to distinguish data from objects in the environment, as some objects had the same water index as others.

Finally, we provide several recommendations for future research that researchers might be interested in, which include:

1. Exploring the concept of combining VINS-mono with star sensors for high precision autonomous navigation on Mars [31].
2. Using the Kalibr [32] calibration tool to obtain accurate extrinsic parameters for IMU + camera. If a method of performing IMU + camera calibration using Kalibr in the AirSim simulation environment can be found, then the RMS error for the estimated path can be reduced down to a few meters or even less. This is because VINS-mono is highly accurate if the correct extrinsic parameters are provided [17].
3. Exploring a solution to use reflectance values instead of radiance values for simulated multispectral sensor.

This is so the standard mNDWI water index can be used instead of a custom water index which does not hold in the real world.

a. Conducting further research to use ray tracing and the following methods the researchers in [26] have.

b. Manually updating Airsim plugin to support Unreal Engine 4.2.2 which has real-time ray tracing.

4. Modelling the sensor noise for the multispectral sensor to make simulation more realistic.
5. Finding or collecting emissivity data for water and other objects in the environment for other bands. This could help with the issue of not being able to use the mNDWI water index.
6. Modelling the actual dynamics of the Mars drone for further realism.
7. Integrating other data that was captured with the get_current_Mars_weather function; such as, atmospheric pressure, wind speed etc.

ACKNOWLEDGMENT

The authors would like to thank the Microsoft research team for creating the AirSim simulation environment; the VINS-mono team for creating VINS-mono and providing the open source code, and Epic Games for creating the Unreal Engine.

REFERENCES

[1] NASA, “NASA - Water: The Molecule of Life An Interview with Philip Ball.” Brian Dunbar, 2007.
[2] K. Northon, “Mars Helicopter to Fly on NASA’s Next Red Planet Rover Mission,” 2018.
[3] F. Vanegas, D. Bratanov, J. Weiss, K. Powell, and F. Gonzalez, “Multi and hyperspectral UAV remote sensing: Grapesy phylloxera detection in vineyards,” IEEE Aerosp. Conf. Proc., vol. 2018-March, pp. 1–9, 2018, doi: 10.1109/AERO.2018.8396450.
[4] F. Vanegas, D. Bratanov, K. Powell, J. Weiss, and F. Gonzalez, “A Novel Methodology for Improving Plant Pest Surveillance in Vineyards and Crops Using UAV-Based Hyperspectral and Spatial Data,” Sensors, vol. 18, no. 1, p. 260, Jan. 2018, doi: 10.3390/s18010260.
[5] M. Gatti et al., “MECS-VINE: A new proximal sensor for segmented mapping of vigor and yield parameters on vineyard rows,” Sensors (Switzerland), vol. 16, no. 12, 2016, doi: 10.3390/s16122009.
[6] D. Popescu, L. Ichim, and T. Caramihale, “Flood areas detection based on UAV surveillance system,” in 2015 19th International Conference on System Theory, Control and Computing, ICSTCC 2015 - Joint Conference SINTES 19, SACCS 15, SIMSIS 19, 2015, doi: 10.1109/ICSTCC.2015.7321384.
[7] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” 2018.
[8] K. R. Sping, B. O. Flynn, J. C. Long, and M. W. Davidson, “Spatial Resolution in Digital Imaging (tutorial),” Nikon - microscopu.com, 2015.
[9] F. W. Kneumau, “Temporal Resolution,” Journal of Diagnostic Medical Sonography, vol. 7, no. 6. pp. 354–356, 1991, doi: 10.1177/0785647939100700608.
[10] C. Huang, Y. Chen, S. Zhang, and J. Wu, “Detecting, Extracting, and Monitoring Surface Water From Space Using Optical Sensors: A
Review,” Rev. Geophys., vol. 56, no. 2, pp. 333–360, Jun. 2018, doi: 10.1029/2018RG000598.

[11] H. Imagery, “Multispectral vs Hyperspectral Imagery,” pp. 1–7, 2015.

[12] N. Cristianini, “Supervised and Unsupervised Learning,” Dictionary of Bioinformatics and Computational Biology. 2004, doi: 10.1002/0471650129.dobs0716.

[13] N. Li, A. Martin, and R. Estival, “An automatic water detection approach based on Dempster-Shafer theory for multi-spectral images,” 2017.

[14] S. Patel, “Chapter 2: SVM (Support Vector Machine) — Theory – Machine Learning 101 – Medium.” 2017.

[15] G. Rohith, “Support Vector Machine — Introduction to Machine Learning Algorithms,” 2018, p. 1, 2018.

[16] E. Bondi et al., “AirSim-W: A simulation environment for wildlife conservation with UAVs,” Proc. 1st ACM SIGCAS Conf. Comput. Sustain. Soc. COMPASS 2018, vol. 18, 2018, doi: 10.1145/3209811.3209880.

[17] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics,” Intell. Ind. Syst., vol. 1, no. 4, pp. 289–311, Dec. 2015, doi: 10.1007/s40903-015-0032-7.

[18] X. Xu and H. Fan, “Feature based simultaneous localization and semi-dense mapping with monocular camera,” in 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2016, pp. 17–22, doi: 10.1109/CISP-BMEI.2016.7852675.

[19] C. Kerl, J. Sturm, and D. Cremers, “Dense visual SLAM for RGB-D cameras,” in IEEE International Conference on Intelligent Robots and Systems, 2013, doi: 10.1109/IROS.2013.6696650.

[20] C. Kerl, J. Stucker, and D. Cremers, “Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, doi: 10.1109/ICCV.2015.261.

[21] T. Qin, P. Li, and S. Shen, “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator,” 2018.

[22] F. Vanegas, K. J. Gaston, J. Roberts, and F. Gonzalez, “A Framework for UAV Navigation and Exploration in GPS-Denied Environments,” in IEEE Aerospace Conference Proceedings, 2019, doi: 10.1109/AERO.2019.8741612.

[23] H. Kurniawati and V. Yadav, “An Online POMDP Solver for Uncertainty Planning in Dynamic Environment,” Int. Symp. Robot. Res., pp. 1–16, 2014.

[24] F. Vanegas, J. Roberts, and F. Gonzalez, “UAV tracking of mobiletarget in occluded, cluttered and GPS-denied environments,” IEEE Aerosp. Conf. Proc., vol. 2018-March, pp. 1–7, 2018, doi: 10.1109/AERO.2018.8364449.

[25] Harris Geospatial Solutions, “Radiance vs. Reflectance.” 2013.

[26] Microsoft, “Windows Subsystem for Linux Documentation,” 2016. [Online]. Available: https://docs.microsoft.com/en-us/windows/wsl/about.

[27] Marcelino Almeida, “Camera intrinsic parameters corresponding to the Scene image? · Issue #269 · microsoft/AirSim · GitHub.” 2017.

[28] J. Capitan, M. T. J. Spaan, L. Merino, and A. Ollero, “Decentralized multi-robot cooperation with auctioned POMDPs,” in Proceedings - IEEE International Conference on Robotics and Automation, 2012, pp. 3323–3328, doi: 10.1109/ICRA.2012.6224917.

[29] “Fusion Releases Mars 2030 Editor as Part of Larger Collaboration with NASA.”

[30] A. Börner et al., “SENSOR: A tool for the simulation of hyperspectral remote sensing systems,” ISPRS J. Photogram. Remote Sens., vol. 55, no. 5–6, pp. 299–312, Mar. 2001, doi: 10.1016/S0924-2716(01)00022-3.

[31] X. Guan, X. Wang, J. Fang, and S. Feng, “An innovative high accuracy autonomous navigation method for the Mars rovers,” Acta Astronaut., vol. 104, no. 1, pp. 266–275, Nov. 2014, doi: 10.1016/j.actaastro.2014.08.001.

[32] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, “Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes,” in Proceedings - IEEE International Conference on Robotics and Automation, 2016, doi: 10.1109/ICRA.2016.7487628.

**APPENDICES**

**A. Virtual Camera Calibration Output**

```
image_width: 1200
image_height: 740

camera_matrix:
rows: 3
cols: 3
data: [640, 645332, 0.00000, 641, 176910, 0.00000, 640, 547956, 356, 526793, 0.00000, 0.00000, 1.000000]
distortion_model: plumb_bob
distortion_coefficients:
rows: 1
cols: 5
data: [-0.00191, 0.000356, 0.000000, 0.00000, 0.000000]
rectification_matrix:
rows: 3
cols: 3
data: [1.00000, 0.00000, 0.00000, 0.00000, 1.00000, 0.00000, 0.00000, 0.00000, 1.000000]
projection_matrix:
rows: 3
cols: 4
data: [642, 635668, 0.00000, 642, 240565, 0.000000, 642, 300442, 3
59.719225, 0.000000, 0.00000, 1.000000]
```