AAS 20-455

A MACHINE LEARNING SOLUTION TO OPTIMAL LANDING SITE SELECTION AND LANDER CONTROL

Omkar S. Mulekar* **and Riccardo Bevilacqua†**

Previous investigations have shown that Artificial Neural Networks (ANNs) can be trained to drive closed-loop controllers to yield near optimal trajectories. The problem of selecting an optimal landing site near an objective is defined and investigated for a 3 Degree of Freedom point-mass lunar lander. A trajectory optimization software is used to generate optimal state-action pairs to train one ANN for use in an optimal controller. It is also used to generate data to train a site-selecting ANN from lander initial states, surface geometry, and objective position. The ANNs demonstrate a dynamics-based method of selecting an optimal landing site.

Keywords: Neural networks, optimal control, lunar lander, landing site selection

INTRODUCTION

Landing technologies play a central role in lunar and planetary exploration, as they are used in one of the most dynamic, hazardous, and critical phases of exploration missions. Surface hazards like rocks, slopes, and craters have been a key factor in decisions made by both pilots and mission planners for selecting landing sites. Large-scale hazards can be observed from *a priori* reconnaissance by lunar orbiters, but small and local hazards may not always be accounted for beforehand, as demonstrated in the Apollo 11 mission when Neil Armstrong performed a downrange divert to avoid a boulder field.[1] Although the earliest Hazard Detection and Avoidance (HDA) systems consisted of the pilot's vision and control authority on the spacecraft,[1] recent advances in computer vision have allowed for the development of novel algorithms that perform HDA autonomously using camera-based and LIDAR-based terrain sensing techniques.[2–4]

Background

The research discussed made use of Artificial Neural Networks (ANNs) to approximate solutions to optimal control problems.

Neural Networks Machine learning techniques such as ANNs have been employed for a number of problems requiring accurate approximation of functions for which equations are not easily defined, but for which extensive data is either available or easily produced. ANNs originated in attempts to mathematically represent the brain's information processing and learning systems.[5] They

*PhD Student, Mechanical and Aerospace Engineering, University of Florida, 939 Sweetwater Dr., MAE-A 211, Gainesville, FL 32611.

†Associate Professor, Mechanical and Aerospace Engineering, University of Florida, 939 Sweetwater Dr., MAE-A 211, Gainesville, FL 32611.

are built by "neurons," which map many inputs to a single output. Neurons are defined by weights, a bias, and an activation function. The output of a neuron y given an input vector $\mathbf{x}$ is given by

$$y = \phi(\mathbf{w}^T\mathbf{x} + b)$$

where $\mathbf{w}$ is a vector of the associated weights, b is the bias, and ϕ is the activation function. The classical activation function used is the sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

While this activation function can be powerful, problems such as its vanishing gradient have arisen.[6] Learning techinques are typically based on the gradient of the activation function, and the sigmoid has near-zero gradient for very large and very small inputs. Other activation functions such as the hyperbolic tangent $\phi(z) = \tanh(z)$ and the Rectified Linear Unit (ReLU), $\phi(z) = \max(0, z)$ have been employed more in recent years, especially in regression problems.

One layer of an ANN, or specifically a densely connected layer, maps the input vector to several outputs through several neurons. For simple architectures with densely connected layers, the ANN architecture is defined by the number of layers between the input and output vectors (i.e. hidden layers), the number of neurons in each layer, and the activation functions used on each neuron. If an ANN has one hidden layer, it is a shallow neural network; but if it has multiple, it a deep neural network (DNN).[5]

The process of training an ANN is a supervised learning technique, meaning each input in a set of training data $\mathbf{x}_i$ has an associated training output $\mathbf{t}_i$ to evaluate the ANN's performance for predictions $\mathbf{y}_i$ made during training. The process of training an ANN involves iterating through training data, evaluating some loss function $E(\mathbf{w})$ that indicates the quality of the ANN's fit to the data (e.g. mean squared error or mean absolute error), and updating weights and biases $\mathbf{w}$ accordingly through the process of back propagation. The classical training algorithm is gradient decent which given a learning rate η updates the weights in the ANN by evaluating the gradient of the loss function

$$\Delta\mathbf{w} = -\eta\nabla_{\mathbf{w}}E$$

where $\Delta\mathbf{w}$ are the changes in the weights defining the ANN.

Another training algorithm is the Levenberg-Marquardt algorithm which is a simple yet robust alternative to gradient decent.[7] The weight update equation is given by

$$\Delta\mathbf{w} = -[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}\mathbf{e}$$

where $\mathbf{J}$ is the Jacobian matrix approximating the Hessian as $\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}$, the matrix $\mathbf{I}$ is the identity matrix, μ is a damping scalar chosen to make the hessian approximation positive definite, and $\mathbf{e}$ is the error vector.[8] Both of the mentioned training methods for ANNs are iterative in that they cycle through training inputs (either individually or in "batches"), and they update the ANN parameters so that the output error decreases.[5]

ANNs in Optimal Controls One recent use of ANNs has been in the field of Optimal Controls, where the goal is to design a control law that minimizes (or maximizes) some cost function given some system dynamics and defined constraints on the controls and system states.[9] In aerospace applications, a common objective is to optimize fuel usage by minimizing the cost function

$$J = \int_{t_0}^{t_f} |\mathbf{T}| d\tau$$

where $\mathbf{T}$ is a thrust vector. The cost function is said to be constrained to system dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ where $\mathbf{x}$ is the system state variable and $\mathbf{u}$ is the control strategy. Some optimal control problems have analytical solutions, however those subject to nonlinear dynamics (as is the case in astrodynamics) often require numerical methods to solve.

There are several techniques to transcribe optimal control problems to Nonlinear Programming (NLP) problems. One example is direct collocation, which discretizes the optimal control problem's differential and integral equations using techniques like the trapezoidal or Hermite-Simpson methods.[10,11] Several packages in Python and MATLAB like OptimTraj, GPOPS-II, and OpenOCL allow for optimal control problem formulation for automatically handled NLP transcription and generation of a solution.[12–14] OpenOCL specifically uses direct collocation, and it provides the solution in the form of a state and control history, or state-action pairs. These state-action pairs can then be used to train an ANN for use in a closed loop controller. It has been demonstrated that ANNs can learn optimal state-feedback for several aerospace applications including thrust-vectoring rockets and drones.[15] Previous investigations from Sánchez and Izzo have made use of DNNs specifically, and they used on the order of 10 million state-action pairs for training.[16] Other implementations of DNNs from Furfaro have directly used surface images to calculate optimal thrust actions of a lunar lander.[17]

Connection to a Larger Problem The work discussed serves as a preliminary investigation for a larger problem of providing fuel optimal landings in virtually any terrain on lunar and planetary surfaces. Traditional landing sites are limited to geometrically simple regions of the lunar surface. Even with the advances in HDA that allow landing near locally avoidable hazards, the landing site itself is usually not rugged. A graphic depicting this larger problem is shown in Figure 1.

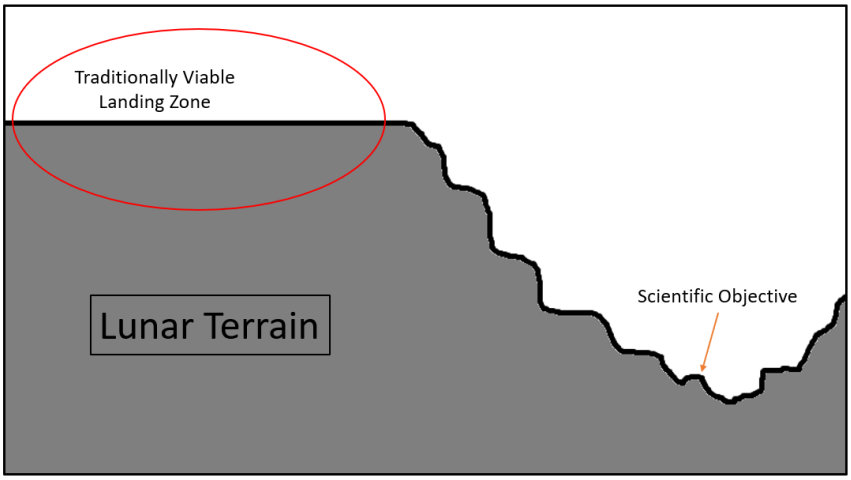

Figure 1. Graphic of proposed problem

Problem Formulation

This paper aims to address the problem of landing close to an objective within a complex terrain while balancing the minimization of fuel usage with the minimization of the landing site distance to an objective $\mathbf{r}_T$, thus achieving an optimal landing site. The problem is investigated in the context of a point mass, fuel-consuming Three Degree of Freedom (3DOF) lunar lander. The goal is to

minimize the cost function

$$J = \alpha|\mathbf{r}(t_f) - \mathbf{r}_T| + \beta \int_{t_0}^{t_f} \sqrt{u_x(\tau)^2 + u_y(\tau)^2 + u_z(\tau)^2} d\tau \tag{1}$$

where $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$ is the system state variable, $\mathbf{u} = [u_x, u_y, u_z]^T$ is the control input, and $\mathbf{r} = [x, y, z]^T$ is a position vector. Any state variable can be expressed in terms of the position vector as $\mathbf{x} = [\mathbf{r}^T, \dot{x}, \dot{y}, \dot{z}]^T$. Although the state variable does not contain the system mass m, it is still included in the system dynamics. The scalars α and β are weights chosen to balance the terminal cost (distance to the target) with the path cost (which optimizes fuel consumption). In this investigation, a value of 0.5 is used for both. The cost function is considered constrained to system dynamics of a fuel-consuming point-mass with mass m. The system dynamics are defined as

$$\begin{cases} \ddot{x} = a_{g,x}(x, y, z) + \frac{1}{m} u_x \\ \ddot{y} = a_{g,y}(x, y, z) + \frac{1}{m} u_y \\ \ddot{z} = a_{g,z}(x, y, z) + \frac{1}{m} u_z \\ \dot{m} = -\frac{\sqrt{u_x^2 + u_y^2 + u_z^2}}{I_{sp} g_0} \end{cases}$$

where $\mathbf{a}_g(x, y, z) = [a_{g,x}(x, y, z), a_{g,y}(x, y, z), a_{g,z}(x, y, z)]^T$ is the acceleration of gravity at a lander position (x, y, z) as calculated from some gravity model, I_{sp} is the specific impulse of the simulated model, and $g_0 = 9.81\text{m/s}^2$. The initial constraint $\mathbf{x}(t_0)$ is defined for $t_0 = 0$, and the terminal constraint $\mathbf{x}(t_f) = [\mathbf{r}(t_f)^T, 0, 0, 0]^T$ is defined for $\mathbf{r}(t_f) \in X$ where X is a set of surface points, or grid points, representing a terrain map produced by a surface mapping system near the scientific objective position $\mathbf{r}_T$.

In this study, three-dimensional terrains are represented by a series of sine functions with randomly generated magnitudes A_i, frequencies ω_i, and phases ϕ_i. The equation for a terrain surface given these randomized parameters is

$$z_{surf}(x_{surf}, y_{surf}) = \sum_{i=1}^{M} A_i \sin(\omega_i x_{surf} + \phi_i) + \sum_{i=M+1}^{N} A_i \sin(\omega_i y_{surf} + \phi_i) \tag{2}$$

The grid points X are extracted from the generated surface, and Gaussian noise is added. The boundary constraint $\mathbf{r}(t_f)$ is limited to a surface grid point in X. The objective position is randomly generated at a point near the surface. An example of a randomly generated surface, grid, and objective can be seen in Figure 2.

Newton's gravitational acceleration is a standard choice for the gravitational model. In this problem is implemented as $\mathbf{a}_g(x, y, z) = -(\mu/\rho^3)\boldsymbol{\rho}$ where μ is the gravitational parameter for the moon, and $\mathbf{r}$ is the distance vector pointing from the center of the moon to the spacecraft. Here, $\boldsymbol{\rho} = x\hat{\mathbf{e}}_x + y\hat{\mathbf{e}}_y + (z + R_{moon})\hat{\mathbf{e}}_z$ where $\{\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y, \hat{\mathbf{e}}_z\}$ is the orthonormal basis fixed to the target site with $\hat{\mathbf{e}}_z$ pointing up. It should be noted that this use of Newtonian gravity should be limited to inertial reference frames. However, the simulation time is on the order of 50 seconds, and the moon rotates once every 28 days giving reason to approximate the target-site fixed frame as inertial. In future simulations, a Moon Centered Inertial frame rather than the target-site fixed frame, as well as a more complicated gravitational model like the GRGM1200A Lunar Gravity Field will be used if needed.[18,19]

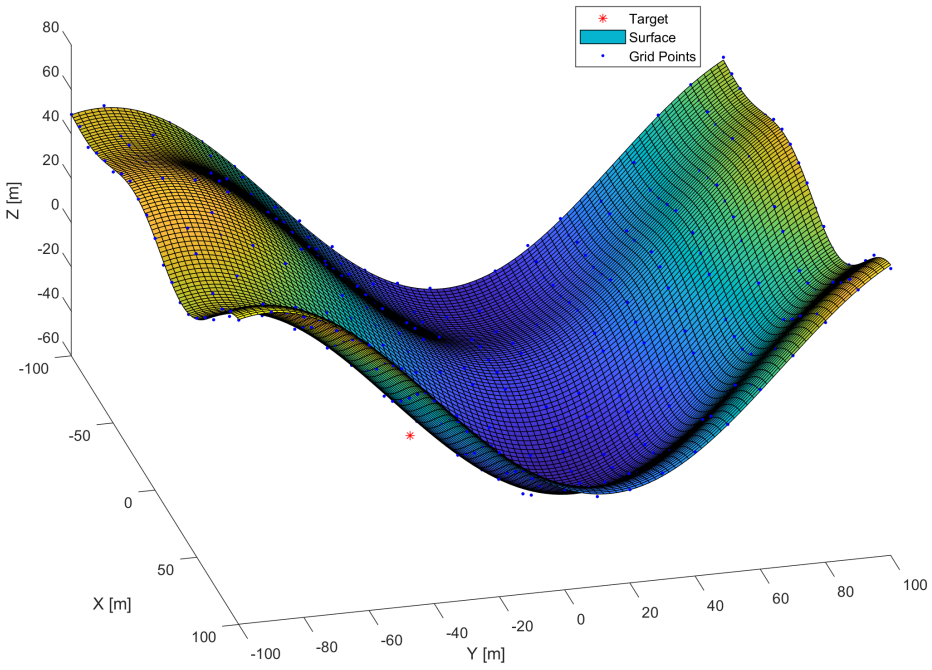

Figure 2. Examples of random surface for training data generation

MODEL DESCRIPTION

A set of two neural networks are trained and implemented: one to select the optimal landing site (ANN 1), and one to be used in a closed loop controller to provide near-optimal control to the landing site (ANN 2). The trajectory optimization tool OpenOCL is used to generate training data in the form of defined "Scenarios" for ANN 1 and state error-action pairs for ANN 2. The performance of the trained ANNs is evaluated through Monte Carlo 3DOF simulations.

ANN Function and Training

The site selecting ANN (ANN 1) is trained to map scenario parameters to a predicted optimal landing site. A diagram of ANN 1 is shown in Figure 3. A scenario is defined by the lander's initial state $\mathbf{x}(t_0)$, grid points on the surface X, and an objective position $\mathbf{r}_T$. In this study, surfaces defined from Eq. (2) are in the domain where x_{surf} and y_{surf} are between -100 m and 100 m. The grid points X are extracted in a $21 \times 21 \times 3$ grid. The ranges of each randomized scenario parameter are given in Table 1. The controller ANN (ANN 2) is trained to map state errors $\mathbf{e}(t) = \mathbf{x}(t_f) - \mathbf{x}(t)$ and system mass m to a control action $\mathbf{u}(t)$. A diagram of ANN 2 is shown in Figure 4. The training data for both ANNs are produced from the MATLAB trajectory optimization package OpenOCL.[14] A scenario is randomly generated, and the optimization problem is posed in the trajectory optimizer as described, except the cost function Eq. (1) in will only include the path cost $J_{path} = \int_{t_0}^{t_f} \sqrt{u_x(\tau)^2 + u_y(\tau)^2 + u_z(\tau)^2} d\tau$.

The trajectory optimizer loops $\mathbf{r}(t_f)$ through the four grid points in X closest to the objective $\mathbf{r}_T$,

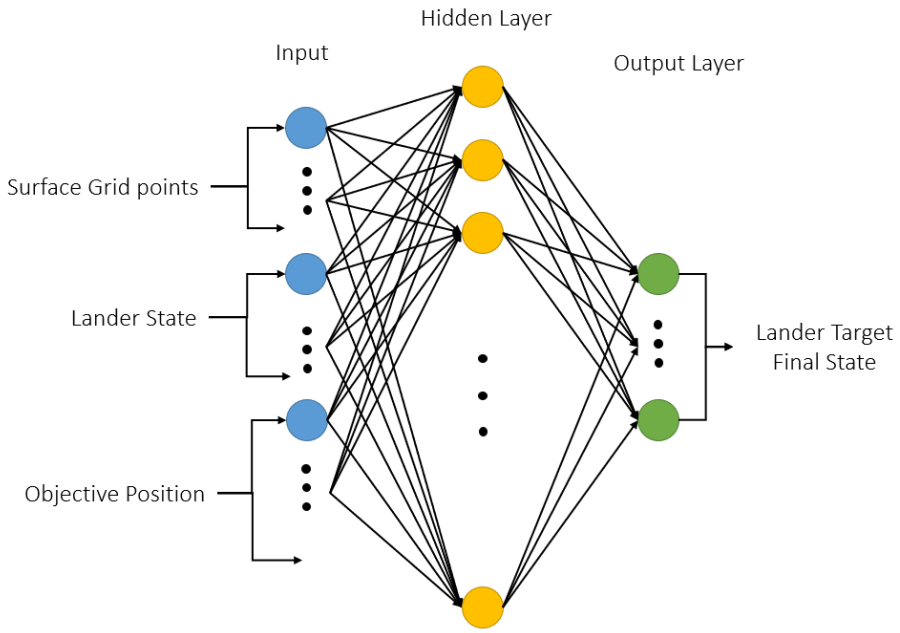

Figure 3. Diagram of inputs and outputs for ANN 1

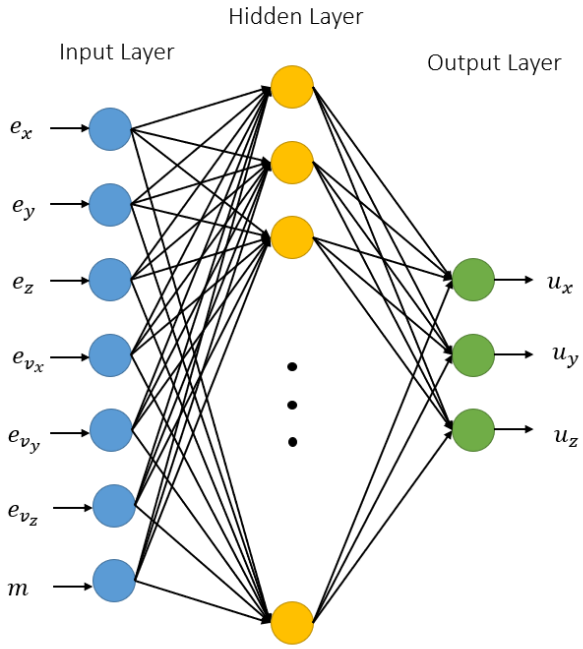

Figure 4. Diagram of inputs and outputs for ANN 2

Table 1. Ranges and values for randomized parameters defining a scenario

Parameter Group	Parameter	Range/Value	Unit
Surface Parameters used in Eq. (2)	A_i	$[-20, 20]$	m
	ϕ_i	$[-3, 3]$	rad
	ω_i	$[-0.1, 0.1]$	rad/m
	M	3	-
	N	6	-
Initial Conditions[17]	x	$[-2000, 2000]$	m
	y	$[-2000, 2000]$	m
	z	$[1000, 1500]$	m
	v_x	$[-15, 15]$	m/s
	v_y	$[-15, 15]$	m/s
	v_z	$[-10, 0]$	m/s
	m	1300	kg
Objective Position	x	$[-100, 100]$	m
	y	$[-100, 100]$	m
	z	$z_{surf} + [-10, 0]$	m

producing a trajectory for each. At present, this restriction is preferred over looping $\mathbf{r}(t_f)$ through every point in X, which would require optimization of 441 trajectories to yield one training data point for ANN 1. Instead, just four trajectories must be optimized to yield one training data point for ANN 1. This restriction is no doubt a limitation on the overall optimization of the landing site selection since the optimal landing site is not necessarily limited to the four grid points close to the trajectory. A resolution to this problem will be investigated in future research.

Each trajectory produces 100 state error-action pairs. The state error-action pairs for each trajectory can be used as training data for ANN 2. The full cost (i.e. including the terminal cost) in Eq. (1) of each trajectory is evaluated. The final state of the trajectory with the minimum evaluation of Eq. (1) is saved as training data for the output of ANN 1. The grid X, initial conditions $\mathbf{x}_0$, and objective position $\mathbf{r}_T$ are saved as the corresponding training data for the input of ANN 1.

The initial state is a 6×1 vector, the objective position is a 3×1 vector 3×1 vector, the input grid points come from a 4×3 array (three position coordinates for four grid points), and the output landing selected landing site is a 3×1 vector. The initial mass is not included as an input to ANN 1 because each simulation starts with 1300 kg. ANN 1 therefore maps 21 input values defining the scenario to the 3 output values defining the selected landing site. The controlling ANN (ANN 2) is used in a feedback controller, and therefore maps state error and system mass to a control input $\mathbf{u}$. The state error with respect to the target landing site $\mathbf{e} = [e_x, e_y, e_z, e_{v_x}, e_{v_y}, e_{v_z}]^T$ is a 6×1 vector, the system mass m is a scalar, and the control input $\mathbf{u}$ is a 3×1 vector. ANN 2 therefore maps 7 inputs to 3 outputs.

For a given scenario, ANN 1 is used to select a landing site, which is used as the reference in a feedback loop. This feedback loop uses ANN 2 to drive the lander in a near-optimal trajectory to the landing site. A block diagram of ANN 2 implemented as a closed loop controller is shown in Figure 5.

Effectively, to produce one training data point for ANN 1, the optimization package must produce several purely fuel-optimal trajectories from the randomized initial condition to several landing sites near the objective point. The trajectory that yields the lowest full cost, corresponds to the optimal

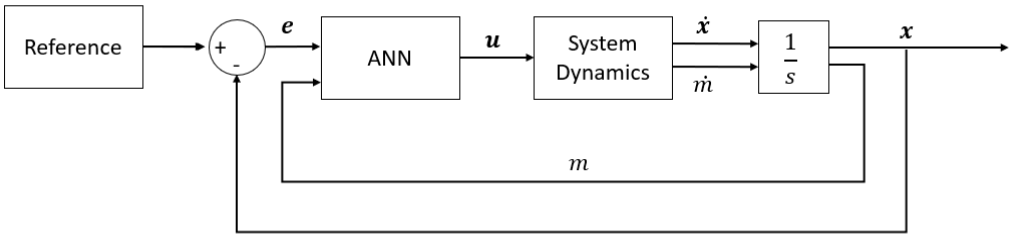

Figure 5. ANN used in closed loop controller for optimal control. Reference refers to the target landing site expressed as a system state.

landing site which ANN 1 must learn to predict. For one trajectory that provides a training data point for ANN 1, several trajectories each provide 100 state error-action pairs to train ANN 2.

In producing the training data for both ANNs, there is abundantly more training data available for ANN 2. However, the data from fewer trajectories are used to train ANN 2 than ANN 1. A total of 1,000 scenarios are used to train ANN 1, and the state error-action pairs from 500 trajectories (50,000 state error-action pairs) are used to train ANN 2.

A shallow architecture was chosen for both ANNs due to the limited size of the training data compared to previous studies.[15,17] ANN 1 has 15 neurons in its hidden layer, and the sigmoid is the chosen activation function. ANN 2 has 90 neurons in its hidden layer, and sigmoid is the chosen activation function. A form of gradient decent, Scaled Conjugate Gradient,[20] is used to train ANN 1, and the Levenberg-Marquardt algorithm is used to train ANN 2.

Monte Carlo Simulations

The performance of the ANNs in selecting landing sites and driving the lander in an optimal trajectory to its final state is evaluated via Monte Carlo Simulations. Scenarios are randomly generated independently from those generated for ANN training data. A 3DOF simulation is then run for each scenario, producing an ANN-selected landing site and an ANN-driven trajectory to that landing site. Optimal landing sites and trajectories are also generated in OpenOCL for each scenario. These trajectories serve as a reference against which to compare the ANN-selected landing sites and ANN-driven trajectories. The costs of the trajectories produced by the trajectory optimizer are compared to the costs of the closed-loop ANN controlled trajectories to assess the capability of the ANNs to learn the optimal solution.

RESULTS AND DISCUSSION

The Monte Carlo simulations are run on 200 scenarios randomly generated independently from the ANN training data. The resulting selected landing sites, trajectories, and control histories are compared to those produced by the trajectory optimizer. First, the characteristics of landing sites selected by ANN 1 are assessed. Then the capability of ANN 2 to drive a near fuel-optimal trajectory is assessed. Then, the overall performance of the entire implemented system is discussed.

Site Selection

An evaluation of how close the lander gets to the scientific objective is needed, though the proximity to the scientific objective is only part of the overall goal. Evaluation and comparison of the terminal costs of both the ANN-driven and optimal trajectories is performed. The difference between the terminal costs for each Monte Carlo simulation is computed, and a histogram is used to visualize the distribution of cost differences. The difference is taken as $J_{term,OCL} - J_{term,ANN}$ where $J_{term,OCL}$ is the terminal cost of the optimal trajectory from OpenOCL, and $J_{term,ANN}$ is the terminal cost of the ANN-driven trajectory. This histogram is shown in Figure 6. It should be

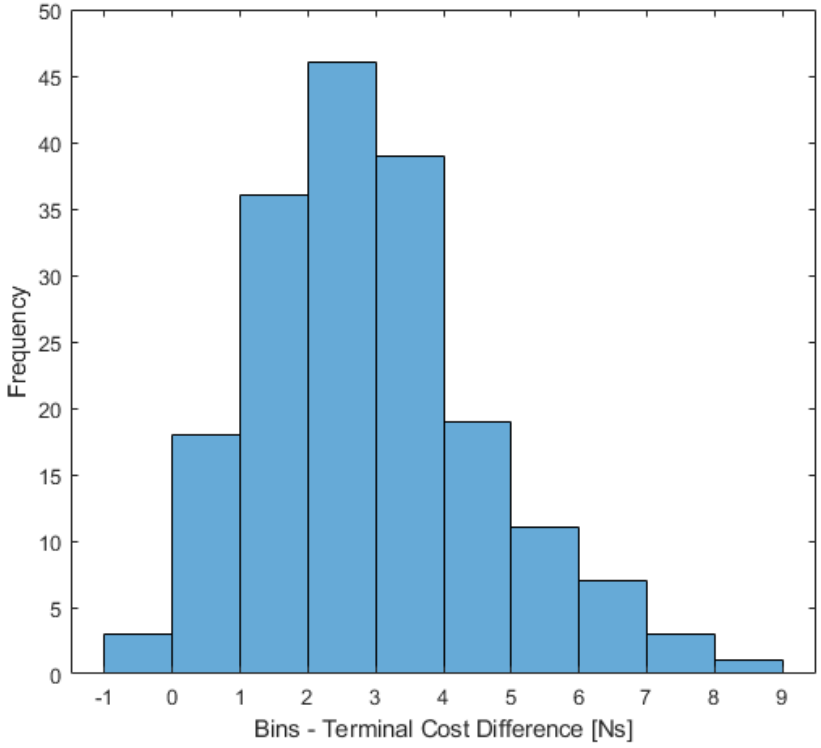

Figure 6. Histogram of terminal cost differences between ANN-driven and optimal trajectories

noted that the terminal cost is an evaluation of distance from the scientific objective to the final lander position (i.e. selected landing site). While such an evaluation would typically have units of distance (meters), the α weighting term in Eq. (1) has units of Ns/m, and the β weighting term is dimensionless so that the terminal and path costs can both have units of Ns and can be added together. The mean cost difference of the data provided in Figure 6 is 2.91 Ns. A positive terminal cost difference implies the optimal trajectory from OpenOCL lands further from the scientific objective, so the positive mean terminal cost difference indicates that the ANN learns to select landing sites closer to the objective than is necessarily optimal.

In addition, the assessment of the viability of the chosen landing site is also needed. The implemented site selection method in OpenOCL requires the final position of the lander to be one of the grid points, which are approximations of points on the surface defined by Eq. (2). There is no such

limitation on the sites chosen by the ANN (though future investigations can include one). ANN chosen landing are not directly on surface grid points, and they are also not necessarily at points between them. One example of an ANN-selected landing site that is neither at nor between surface grid points is in Figure 7. It was found however, that the ANN generally selected a point that was

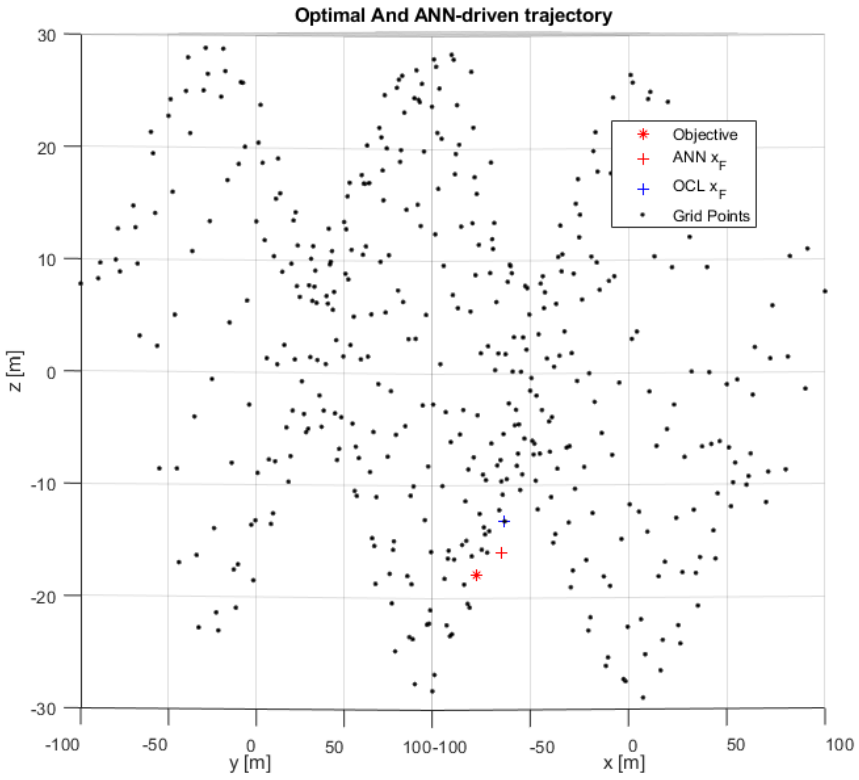

Figure 7. ANN-selected landing site not between grid points, and therefore not viable.

between grid points and the selected landing sites appeared on the surface approximated by the grid points. An example of an ANN-selected landing site that is between grid points is shown in Figure 8.

One option for modifying the ANN-selected landing site is to implement a grid point limitation on the ANN output. The landing site used as a reference for the feedback controller can be limited to a grid point that is closest to the ANN-selected landing site. A histogram of the terminal cost differences from a the Monte Carlo simulation that implements this grid point limitation on the ANN-selected landing sites is shown in Figure 9. The mean cost difference is 2.17 Ns, which is only a 25% decrease from mean terminal cost difference from the Monte Carlo without the implemented grid point limitation.

At present, the ANN has very roughly approximated selection of an optimal landing site. Of course, a larger training data set would likely improve the ANNs ability to approximate selection of an optimal landing site. One concern when training on a larger data set is that rather than learning an underlying dynamics based method for determining the optimal landing site, the ANN may simply learn to match its output to one of the grid point inputs. This would be a subtle indication of

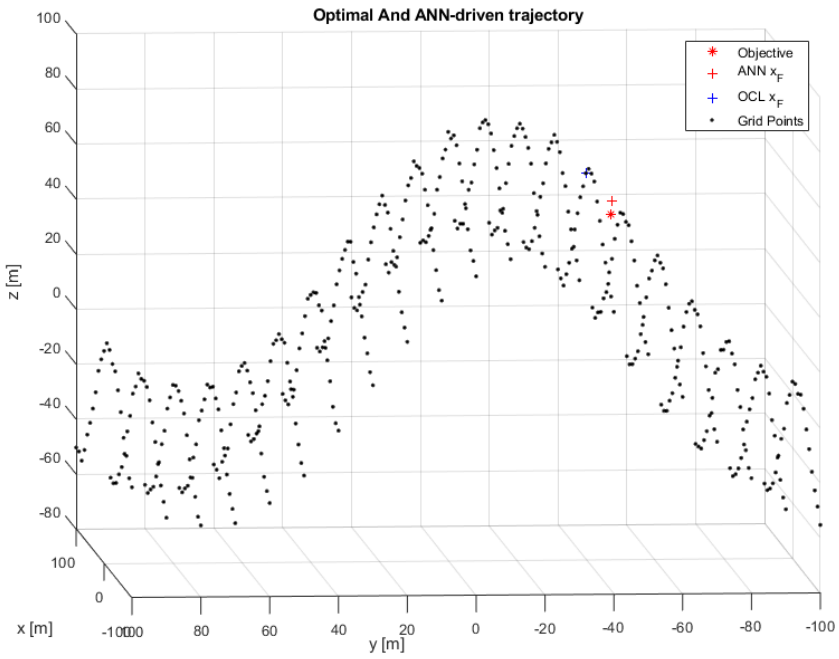

Figure 8. Viable ANN-selected landing site between grid points

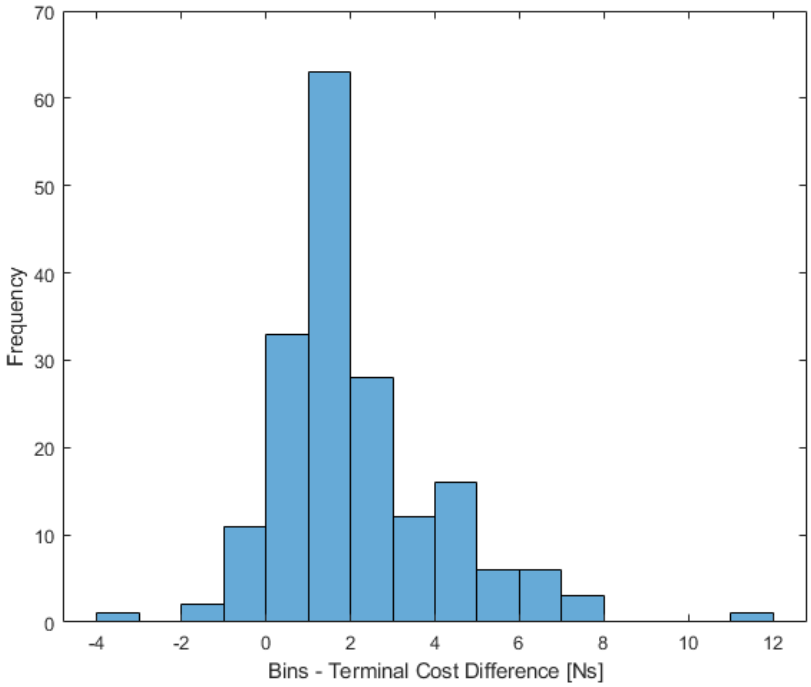

Figure 9. Histogram of terminal cost differences between ANN-driven and optimal trajectories, landing site limited to grid points

overfitting, since the optimal landing site is not necessarily one of the grid points.

Feedback Control

Assessing performance of ANN 2 in providing near-optimal control of the lander to the landing site selected by ANN 1 is done by inspection of the controller behavior near the landing site, inspection of the thrust profiles, and evaluation of the full trajectory cost from Eq. (1).

There are two types of observed behavior of the controller near the landing site: one that drives the trajectory to converge and settle on the selected landing site, and one that drives the trajectory near the landing site but boosts the lander away in an extreme trajectory. Of the 200 Monte Carlo simulations, 189 properly converged onto the landing site, while 11 exhibited the mentioned boost into an extreme trajectory. An example of a properly converging trajectory is shown in Figure 10.

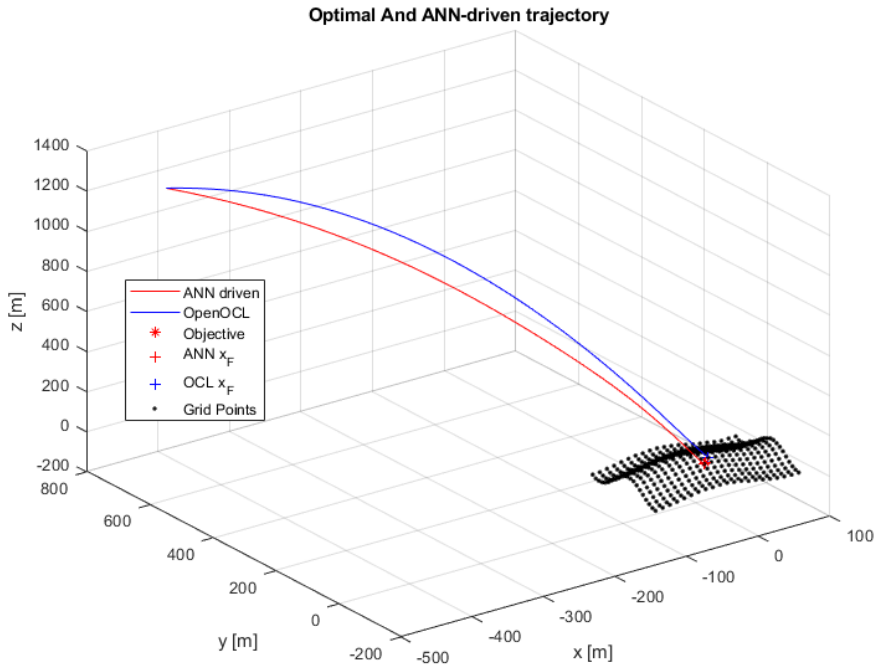

Figure 10. ANN-driven and optimal trajectories

One possible explanation for the Monte Carlo runs with extreme trajectory is the associations provided from the training data. One characteristic of the optimal thrust profiles is a high thrust maneuver at the beginning of the trajectory followed by a coast, and then a high thrust maneuver at the end of the trajectory. Because of this high thrust maneuver at the end of optimal trajectories, the ANNs learn to associate low state errors with high thrusts for some trajectories, causing the lander to boost away from the landing site after approaching close to it. After this boost away, although the position error is in a region the ANN learns to associate with a coasting maneuver, the velocity vector is pointed away from the landing site. Such an error configuration does not exist in the training data for the ANN, so there is no reasonable expectation for what control action the ANN would command. A possible correction to this issue is to expand the region of starting positions in

randomly generated scenarios to include regions closer to the landing site.

Thrust profiles of the optimal solution and the ANN-driven trajectory are compared to evaluate the ability of the ANN to learn the optimal solution. An example of an ANN-driven thrust profile that follows closely to the optimal thrust profile is shown in Figure 11. An example of an ANN-driven thrust profile that controls the lander successfully to the landing site, but does not follow the optimal thrust profile closely is shown in Figure 12.

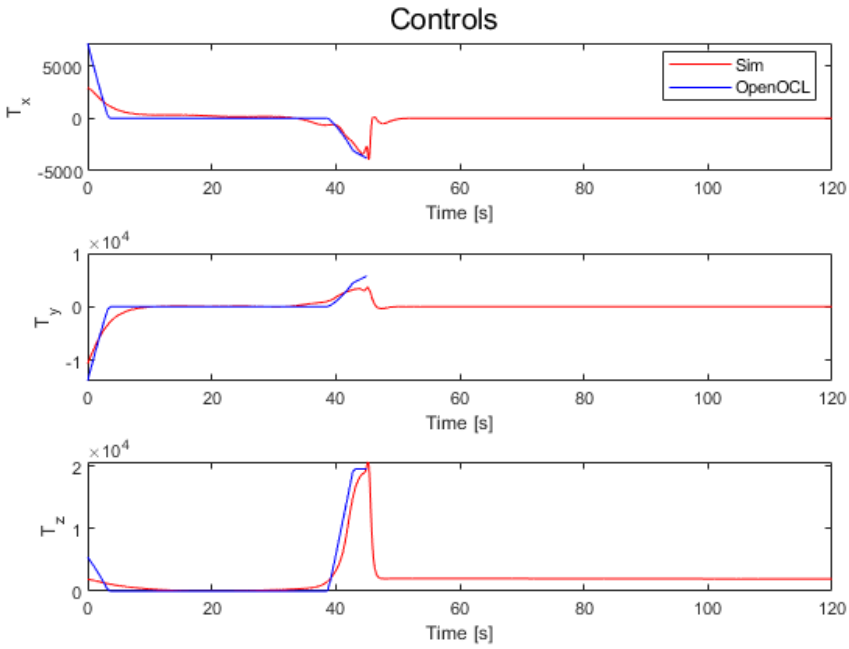

Figure 11. ANN-driven thrust profile that closely matches optimal solution

In addition to inspection of the thrust profiles produced by the ANN, a comparison of trajectory cost evaluations for both the ANN-driven trajectories and the optimal trajectories can be done to assess the ability of the ANN to associate optimal control actions to lander state errors. The difference between the full costs for each Monte Carlo simulation is computed, and a histogram is used to visualize the distribution of cost differences. The difference is taken as $J_{OCL} - J_{ANN}$ where J_{OCL} is the full cost of the optimal trajectory from OpenOCL, and J_{ANN} is the full cost of the ANN-driven trajectory. A histogram of the percent differences is shown in Figure 13. The mean percent difference between the costs of the ANN-driven and optimal trajectories is -32.3%. A negative cost difference implies a higher cost of the ANN-driven trajectory, which is expected given that the ANN is meant to *approximate* the optimal solution. However, there are a several cases that yielded a positive cost difference as seen in Figure 13. In these cases the full cost of the ANN-driven trajectory is lower than the trajectory generated by OpenOCL. It is not the case that the ANN is capable of yielding a trajectory that is "more" fuel optimal, as this would be impossible. The lower full costs are characteristic of scenarios in which the ANN chose a landing site that allowed for a lower trajectory cost. In these cases, the landing site selected by the ANN and the landing site selected in OpenOCL are not the same, so the boundary constraints on the trajectory being optimized via NLP are not the same as those used by the ANN to drive a trajectory. In these specific cases, the model

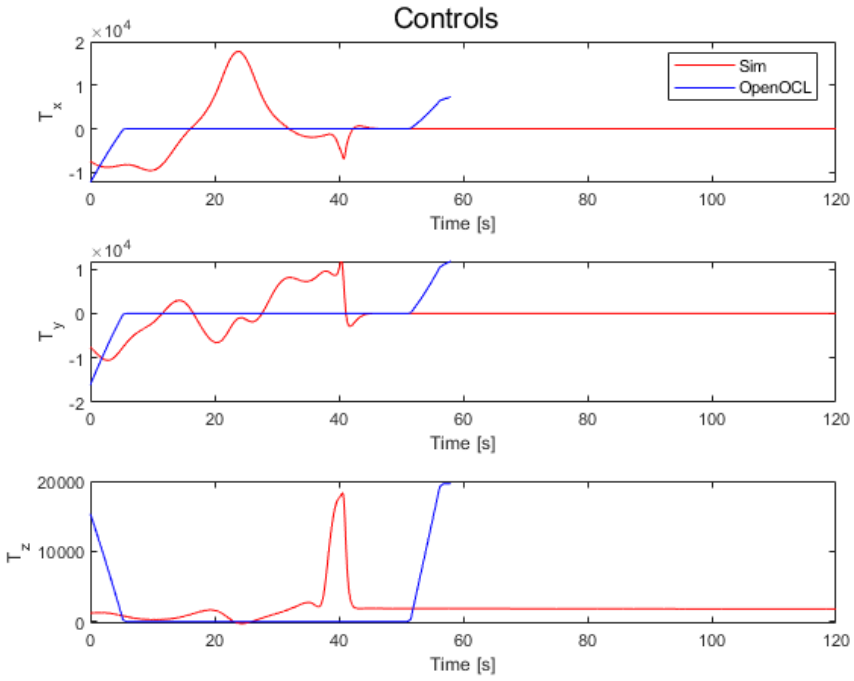

Figure 12. ANN-driven thrust profile that does not closely match optimal solution

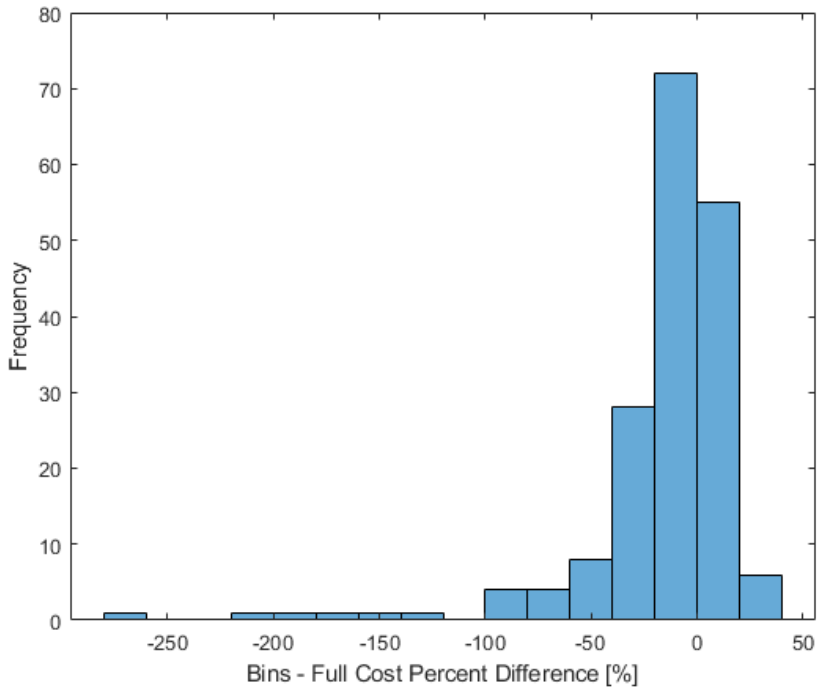

Figure 13. Histogram of full cost percent differences between ANN-driven and optimal trajectories

successfully selected a landing site that allowed for less fuel usage.

Final Comments on Orders of Magnitude

The orders of magnitude of different parameters in the model discussed have an impact on its performance. Future investigations should take care to assess the impact of varying these parameters. Two particular parameter sets include the α and β weighting terms in Eq. (1) and the volume of training data used to train both ANNs.

It has been discussed that the full cost in Eq. (1) is comprised of a path cost which optimizes fuel usage and a terminal cost which optimizes distance to a scientific objective on the lunar surface. It was mentioned that a value of 0.5 was chosen for the weighting terms α and β that balance the effect of these costs on optimization. Due to the large difference in orders of magnitude of the terminal and path costs, this value for the weighting terms may not be appropriate. The terminal costs are on the order of 10^1 Ns while the path costs are on the order of 10^8 Ns. Future investigations should make use of a large value for α and a small value for β to assess the effect of more appropriately balancing the terminal and final costs.

It was mentioned that previous investigations from Sánchez and Izzo trained DNNs from approximately 10 million data points. In this investigation 50,000 data points were used to train the ANN used in a feedback controller. While the ANN trained in this investigation is capable of driving the lander to the desired state, and in many cases in near-optimal trajectories, inspection of the thrust profiles emphasizes a need for more training data. Future investigations will include a much larger training data set that would allow the ANN to more properly learn to produce the thrust profiles associated with the optimal solution.

CONCLUSION

This paper discusses an early investigation of the larger problem of finding dynamics-based solutions to the landing site selection of variable leg-length landers in geometrically complex terrain. The research effort developed a dynamics-based method that uses a terrain data to determine a target state of a landing system near a scientific objective. It was demonstrated that an ANN can learn to determine optimal landing sites given surface points and system state information, a concept that can be extended and used in the introduced larger problem. One primary limitation on the model developed is the light amount of training data used for its two ANNs. Future investigations will take care to assess the effects of tuning the cost function weighting parameters and increasing training data. Refinement of the described system would be extended to the described larger problem. A highly robust landing system would reduce constraints on landing site selection during space mission planning, providing more opportunities for scientific exploration and discovery on other terrestrial bodies than were previously available.

ACKNOWLEDGMENT

This investigation was supported by the NASA Space Technology Graduate Research Opportunity (Grant Number 80NSSC20K1188) and the Graduate Student Preeminence Award from University of Florida Department of Mechanical and Aerospace Engineering.

REFERENCES

[1] T. Brady and S. Paschall, "The challenge of safe lunar landing," *2010 IEEE Aerospace Conference*, 2010, pp. 1–14.

[2] C. D. Epp, E. A. Robertson, and T. Brady, "Autonomous Landing and Hazard Avoidance Technology (ALHAT)," *2008 IEEE Aerospace Conference*, 2008, pp. 1–7.

[3] J. M. Carson, M. M. Munk, R. R. Sostaric, J. N. Estes, F. Amzajerdian, J. B. Blair, D. K. Rutishauser, C. I. Restrepo, A. M. Dwyer-Cianciolo, G. Chen, and T. Tse, *The SPLICE Project: Continuing NASA Development of GN&C Technologies for Safe and Precise Landing*, 10.2514/6.2019-0660.

[4] J. J. Hart and J. D. Mitchell, "Morpheus lander testing campaign," *2012 IEEE Aerospace Conference*, 2012, pp. 1–12.

[5] C. M. Bishop, "Neural Networks," *Pattern Recognition and Machine Learning*, Springer, 2006, p. 225–290.

[6] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 6, 04 1998, pp. 107–116, 10.1142/S0218488598000094.

[7] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, Vol. 11, No. 2, 1963, pp. 431–441, 10.1137/0111030.

[8] H. Liu, "On the Levenberg-Marquardt training method for feed-forward neural networks," *2010 Sixth International Conference on Natural Computation*, Vol. 1, 2010, pp. 456–460.

[9] R. Stengel, "Optimal Trejectories and Neighboring-Optimal Solutions," *Optimal Control and Estimation*, Dover Publications, 2006, p. 225–290.

[10] O. Von Stryk, "Numerical Solution of Optimal Control Problems by Direct Collocation," *Optimal Control Theory and Numerical Methods*, Vol. 111, 04 1998, 10.1007/978-3-0348-7539-4_10.

[11] V. M. Becerra, *Practical Direct Collocation Methods for Computational Optimal Control*. New York, NY: Springer New York, 2013, 10.1007/978-1-4614-4469-5_2.

[12] M. P. Kelly, "Transicription Methods for Trajectory Optimization: A beginners tutorial," 2015.

[13] M. Patterson and A. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, Vol. 41, 10 2014, pp. 1–37, 10.1145/2558904.

[14] J. Koenemann, G. Licitra, M. Alp, and M. Diehl, "OpenOCL - Open Optimal Control Library," 06 2019.

[15] C. Sánchez-Sánchez and D. Izzo, "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 41, 10 2016, 10.2514/1.G002357.

[16] C. Sánchez-Sánchez, D. Izzo, and D. Hennes, "Learning the optimal state-feedback using deep networks," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[17] R. Furfaro, I. Bloise, M. Orlandelli, P. Di, Lizia, F. Topputo, and R. Linares, "AAS 18-363 DEEP LEARNING FOR AUTONOMOUS LUNAR LANDING," 2018.

[18] F. G. Lemoine, S. Goossens, T. J. Sabaka, J. B. Nicholas, E. Mazarico, D. D. Rowlands, B. D. Loomis, D. S. Chinn, G. A. Neumann, D. E. Smith, and M. T. Zuber, "GRGM900C: A degree 900 lunar gravity model from GRAIL primary and extended mission data," *Geophysical Research Letters*, Vol. 41, No. 10, 2014, pp. 3382–3389, 10.1002/2014GL060027.

[19] F. Lemoine, S. Goossens, T. Sabaka, J. Nicholas, E. Mazarico, D. Rowlands, B. Loomis, D. Chinn, G. Neumann, D. Smith, and M. Zuber, "GRGM900C: A degree-900 lunar gravity model from GRAIL primary and extended mission data," *Geophysical Research Letters*, Vol. 41, 05 2014, 10.1002/2014GL060027.

[20] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, Vol. 6, No. 4, 1993, pp. 525 – 533, https://doi.org/10.1016/S0893-6080(05)80056-5.