AAS 20-583

DEEP REINFORCEMENT LEARNING FOR SAFE LANDING SITE SELECTION WITH CONCURRENT CONSIDERATION OF DIVERT MANEUVERS

Keidai Iiyama[*], Kento Tomita[†], Bhavi A. Jagatia[‡], Tatsuwaki Nakagawa[§], and Koki Ho[¶]

This research proposes a new integrated framework for identifying safe landing locations and planning in-flight divert maneuvers. The state-of-the-art algorithms for landing zone selection use local terrain features such as slopes and roughness to judge the safety and priority of the landing point, and are not able to evaluate the safeness of the decision itself to target the selected landing point with concurrent consideration of control and observations over time. In response to this challenge, we propose a reinforcement learning framework that optimizes a landing site selection strategy concurrently with a guidance and control strategy to the target landing site. The learned model could evaluate and select landing sites with explicit consideration of the terrain features, quality of future observations, and control, in order to achieve a robust landing trajectory in system-level. The method is demonstrated for a lunar landing scenario.

INTRODUCTION

On-board hazard detection and avoidance (HDA) capabilities are essential to enable new mission concepts that involve planetary surface operations. With a quick assessment of the perceived terrain data (e.g., DEM, visible spectrum map, or a combination thereof) from optical and/or Lidar sensors, the HDA technology creates a map of probability of safety for prioritizing candidate landing zones. NASA has been actively developing the technology for Precise landing and Hazard Avoidance (PLHA),[1–6] and Safe and Precise Landing Integrated Capabilities Evolution (SPLICE) project is currently underway to develop next generation PLHA technologies.[7–9] The HDA algorithm being developed for SPLICE directly leverages the HDA algorithms from the previous Autonomous Landing Hazard Avoidance Technology (ALHAT) project. ALHAT is capable of quickly assessing the DEM on-board and in real time during the descent, and assigns a probability of a safe landing to each pixel on the map.[10–20] ALHAT's HDA capability along with the terrain relative navigation and hazard detection functions was successfully demonstrated during the hardware-in-the-loop testing on the Morpheus Vertical Testbed.[16,21] However, a critical caveat of ALHAT is that it only selects a list of discrete landing points based on local static information (e.g., slopes, roughness, etc.). To consider the feasibility of the divert maneuver, landing selection algorithm that calculates approximate landing footprint has been presented.[22,23] To reflect predetermined scientific values of each landing site in the landing site selection process, landing site selection methods that leverages Bayesian networks has been proposed.[6] Cui, et al. (2017) proposed a method that calculates synthetic landing area assessment criterion based on terrain safety, fuel consumption, and touchdown performance during descent.[24]

In these previously proposed landing site assessment methods, the possibility that the target landing site will be changed in the future is ignored. However, considering that the field of view and the quality of the

[*] Master Student, Department of Aeronautics and Astronautics, The University of Tokyo, 7-3-1, Hongo,Bunkyo-ku,Tokyo, Japan.
[†] Ph.D. Student, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332.
[‡] Master Student, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332.
[§] Bachelor Student, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332.
[¶] Assistant Professor, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332.

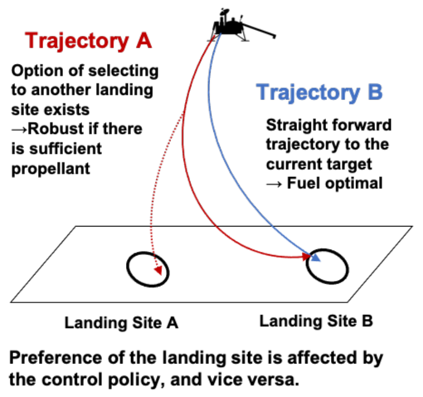

Trajectory A
Option of selecting to another landing site exists →Robust if there is sufficient propellant
Trajectory B
Straight forward trajectory to the current target → Fuel optimal
Landing Site A
Landing Site B
Preference of the landing site is affected by the control policy, and vice versa.

Figure 1. Landing site selection and control

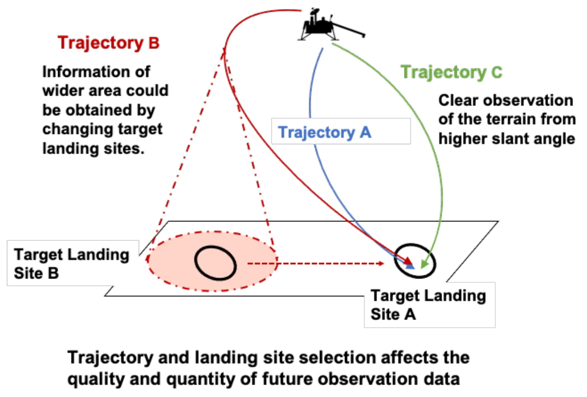

Trajectory B
Information of wider area could be obtained by changing target landing sites.
Trajectory A
Trajectory C
Clear observation of the terrain from higher slant angle
Target Landing Site B
Target Landing Site A
Trajectory and landing site selection affects the quality and quantity of future observation data

Figure 2. Observability and trajectory

observation data changes depending on the spacecraft state, the best landing site changes each time new observation data is obtained. This effect could not be ignored especially when observability is limited or safe landing sites are sparse. When multiple chances of observation and divert maneuver are considered, the following 2 aspects should be considered.

The first aspect is the coupling between landing site selection and control maneuver planning. Existing landing site selection algorithms assume that the lander is guided by a simple predetermined control law, while control law focuses on guiding the lander to a fixed landing target decided outside the controller. However, When there are chances of divert maneuver in middle of the trajectory, if the control policy changes, the best landing site may change. Therefore, landing site and future control plan has to be optimized simultaneously in order to achieve an optimal trajectory. The concept is shown with an example at Fig.1.

The second aspect is that the visibility of the terrain during descent. As illustrated in Figure 2, the change in the control policy and the target landing site has impact on the quality and number of the obtained observation data during descent. Therefore, the controller has to be aware that sufficient information to judge a good landing site could be obtained by following the future trajectory. In general landing trajectory design, glide slope constraints are applied as a path constraint in order to ensure observations from higher elevations, but the actual quality and quantity of observation information could not be explained only by slope angles. To tackle this problem, Crane (2014) developed an on-line information-seeking trajectory modification method which selects a trajectory that minimizes the weighed sum of the estimated entropy and field of view of the image obtained in the future, fuel consumption, and the coverage of the entire field. However, the research focused on modifying the nominal trajectory for a fixed landing site, and updating landing site successively during descent was not considered in the research. Since changing the landing site affects the observation by change in the trajectory and the observation target, the impact of landing site selection to the observation data quality should also be considered.

In order to tackle these problems, this paper proposes an learning-based method that selects landing site and design guidance trajectory successively during HDA phase. The agent in the proposed method seeks to maximize total probability of landing to a safe terrain landing site while minimizing total fuel consumption, landing error to the target, and final velocity. Model-free reinforcement learning techniques are leveraged to learn a policy that concurrently selects landing site and guidance strategy to the target, by interacting with the simulator environment and learning how to maximize total probability of successful landing with minimal fuel consumption. The general concept is shown in Fig.3.

The outline is as follows. First, we explain the assumptions used in modeling the HDA phase and show that the sequence of action choices can be formulated as a partially observable Markov decision process (POMDP). Next, the method to optimize landing site selection policies and controller is introduced. Finally, the performance of the obtained controller is analyzed and qualitative interception is given.

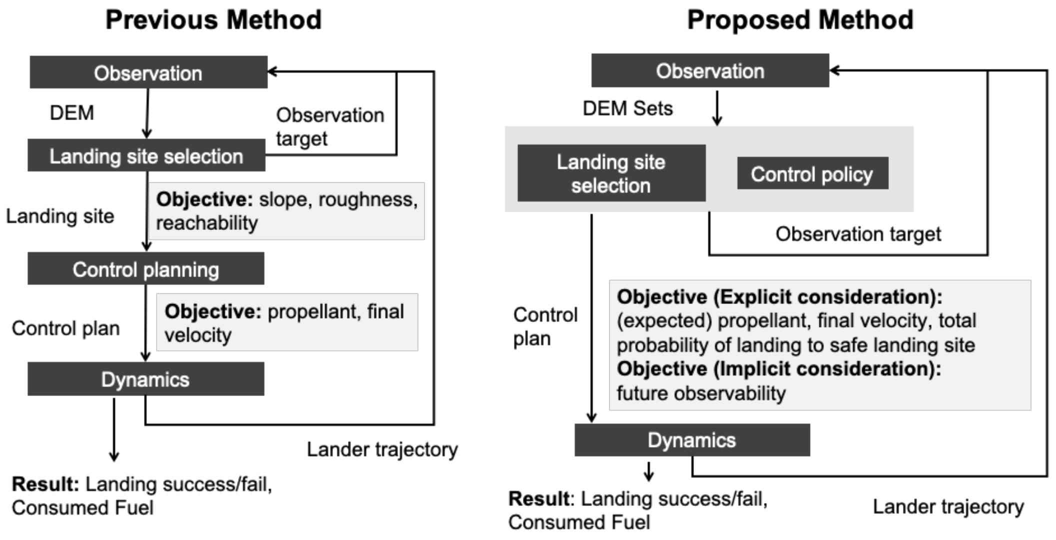

Previous Method
Observation
DEM
Observation target
Landing site selection
Landing site
Objective: slope, roughness, reachability
Control planning
Control plan
Objective: propellant, final velocity
Dynamics
Lander trajectory
Result: Landing success/fail, Consumed Fuel
Proposed Method
Observation
DEM Sets
Landing site selection
Control policy
Observation target
Control plan
Objective (Explicit consideration): (expected) propellant, final velocity, total probability of landing to safe landing site
Objective (Implicit consideration): future observability
Dynamics
Result: Landing success/fail, Consumed Fuel
Lander trajectory

Figure 3. General Concept

PROBLEM FORMULATION

Dynamics

Soft lunar landing scenario is assumed in this paper. In this paper, the 3 degrees of freedom (3-DOF) problem is considered. The equation of motion governing the dynamics of the problem are given as follows.

$$\begin{aligned} \boldsymbol{v} &= \dot{\boldsymbol{r}} \\ \boldsymbol{a} &= \frac{\boldsymbol{T}}{m} + \boldsymbol{g} \\ \dot{m} &= -\frac{||\boldsymbol{T}||}{g_{ref} I_{sp}} \end{aligned} \tag{1}$$

where $r = [r_x \; r_y \; r_z]$ is the position in the target centered orthonormal frame with the z-axis pointing upward, and $\boldsymbol{g} = [0 \; 0 \; -1.62]^T$ is the gravity. In this paper, the gravity is considered constant during the entire mission, and the effect of planetary rotation is ignored. In addition, limitation in the thrust range is applied as follows.

$$0 \leq |\boldsymbol{T}| \leq T_{max} \tag{2}$$

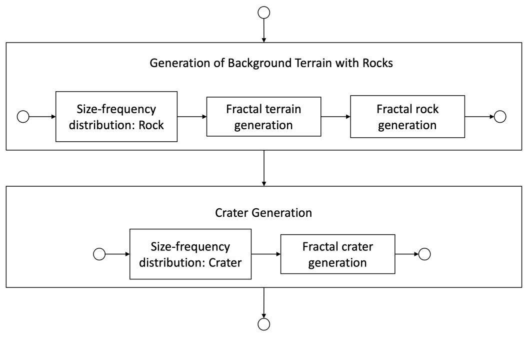
In addition, the following glide slope constraint are applied in most planetary landing problems

$$\theta_g = \arctan\left(\frac{\sqrt{r_x^2 + r_y^2}}{r_z}\right) < \theta_{gmax} \tag{3}$$

This constraint is applied in order to avoid the lander from hitting the terrain at low altitudes, and to ensure that the inclination to the target is kept over a certain amount for navigation and hazard detection purposes.

Terrain Generation

Observation data is consisted of DEM generated by HD LIDAR, and spacecraft state information (position, velocity, mass) with noise. In order to generate Lidar DEMs, we need a true DEM as a reference. Lunar Reconnaissance Orbiter (LROC) database provides Digital Terrain Model (DTM) of the lunar surface.[25] However, the resolution of the DTM ($\approx$ 50m/pixel) in the database is not sufficient for the LIDAR DEM model generation. Therefore, we made an original terrain generator leveraging crater distribution models proposed by Pike, and boulder distribution models proposed by Bernard[26].[27] While the model was originally developed for Mars, studies of rock density on the moon shows that the model can be substituted for the lunar topography by adjusting the parameters. The generated terrain has a size of 1000m x 1000m and resolution of 1m. The generation process of the terrain is described in Fig.4. Example of the generated terrain is shown in Fig.5.



Generation of Background Terrain with Rocks
Size-frequency distribution: Rock
Fractal terrain generation
Fractal rock generation
Crater Generation
Size-frequency distribution: Crater
Fractal crater generation

Figure 4. Terrain generation process

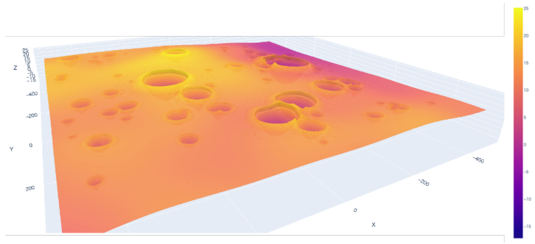


Figure 5. Example of the generated terrain

Safety Map Generation

In order to assess if the landing has succeeded in the simulator, the safety of each landing point in the generated terrain map has to be assessed. We evaluated the safety by applying the safety assessment algorithm developed in the ALHAT project to the entire generated terrain. The algorithm for assessing the deterministic safety value $V_D \in \{0, 1\}$ is shown in Algorithm 1.

Observation Data Generation

The control agent utilizes 2 observation data for control: estimated lander state (position, velocity, and mass) and 2d map of the terrain. For the former lander state, we assumed that we have perfect information without errors. Incorporating navigation errors and hazard relative navigation algorithms in the framework will be an interesting future work. For the latter 2d map of the terrain, we considered 2 options. The first option is to directly pass the Lidar DEM, while the second option is to pass the stochastic safety map obtained by applying Algorithm1 on the obtained Lidar DEM. The first option requires the control agent to assess the safety of each landing site from scratch, while in the second option the control agent has access to more direct information about the safety of each landing point based on roughness and slope values. In this paper, we chose the second approach as we failed to design a controller using the first approach. Modifying the architecture to guide the lander using only DEM information will be our future work.

The simulation of the Lidar DEM during descent requires complex calculations. To simulate a Lidar DEM, detector pattern formulation, ray interception with the terrain, addition of the range bias, transformation to the point clouds, and finally transformation to a digital elevation map has to be conducted even if we assume that the Lidar position is fixed to a known position. In lunar descent cases, errors from vehicle state knowledge, Lidar misalignment, map assembly errors when considering vehicle motions, and system latency should also have to be considered. Since the accurate modeling of the entire procedure is extremely complicated, as an initial concept study, we chose to generate pseudo observation safety map in each observation timing by directly cutting out the portion of the stochastic safetymap of the entire terrain that could be obtained by directly applying Algorithm1 to the entire terrain, add adding errors to them. By this way, we are able to greatly save the computation time required for simulation since we can avoid generating DEMs and running Algorithm1 at each observation timing. Modeling the entire process in high fidelity would be the core of our future work.

Algorithm 1 Deterministic and Stochastic safey map generation

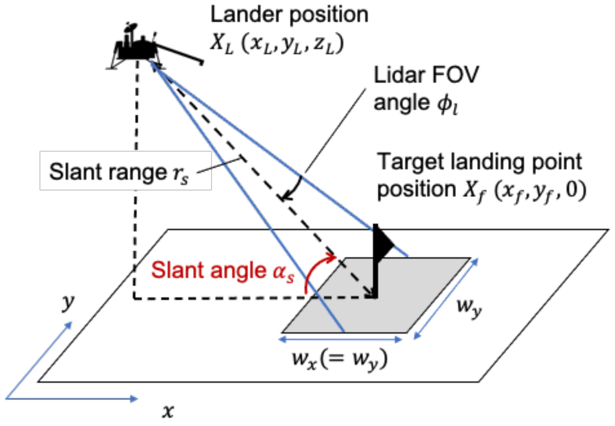
```
Input: DEM $D(m \times n)$
Output: deterministic safety value map $V_D(m \times n) \in \{0,1\}$, stochastic safety value map $V_P(m \times n) \in [0,1]$
  Initialize:
    $S \leftarrow [0, \ldots, 0]$                                ▷ list of slope for each orientation
    $R \leftarrow [0, \ldots, 0]$                                ▷ list of roughness for each orientation
    $P \leftarrow [0, \ldots, 0]$                                ▷ list of safety probability for each orientation
    $O \leftarrow [0, \frac{1}{n_o}\pi, \ldots, \frac{n_o-1}{n_o}\pi]$   ▷ possible lander footpad orientation
  for $row = 1, \cdots, m$ do                                    ▷ for each pixel in the DEM
    for $col = 1, \cdots, n$ do
      Calculate pad placement and contact
      for $o_i = 1, \cdots, n_o$ do                             ▷ calculate worst case slope for all orientations
        $S[o_i] = \text{GET\_SLOPE}(row, col, D)$
      end for
      $s_{max} \leftarrow \text{MAX}(S)$
      if $s_{max} > safe\ slope\ threshold$ then                ▷ slope not safe
        $V[row][col] \leftarrow 0$
      else
        for $o_i = 1, \cdots, n_o$ do
          $R[o_i] = \text{GET\_ROUGHNESS}(row, col, D)$
          $P[o_i] = \text{ROUGHNESS\_TO\_SAFETY\_PROBABILITY}(row, col, D, R)$
        end for
        $V_P[row][col] \leftarrow \text{MIN}(P)$
        $r_{max} \leftarrow \text{MAX}(R)$
        if $r_{max} > safe\ roughness\ threshold$ then          ▷ roughness not safe
          $V_D[row][col] \leftarrow 0$
        else                                                    ▷ slope and roughness safe
          $V_D[row][col] \leftarrow 1$
        end if
      end if
    end for
  end for
  return $V_D.V_P$
```

When generating "observation safety map", 3 main relationship between the lander's state (or trajectory) and the Lidar DEM (and the generated safety map) were modeled. The first effect is the field of view (FOV) of the Lidar DEM. For simplicity, we assumed the obtained DEM is square, and its 2 axis is always aligned with the x, y coordinate of the map. The length of one side of the square $w_x = w_y$ is calculated using the following equation.

$$w_x = r_s \tan \phi_l \tag{4}$$

where r_s is the slant range (distance between the Lander and the target position), and ϕ_l is the field of view of the Lidar which was set to 11.4 [deg].(Fig. 6) In reality, the DEM field of view is not rectangle, and the field of view differs between the horizontal and vertical direction of the lander, but we believe this assumption is fair enough for an initial concept study.

The second effect is the effect of the slant range. As the slant range increases, errors in the range measurement in the Lidar DEM increases. In order to model this effect, In addition, as the slant range gets longer the sampling distance increases, small boulders in the terrain will be overlooked. This effect was modeled by fixing the Lidar DEM size to 64x64. The third effect is the effect of slant angle (angle between horizontal plane and beam direction, in our research this is same as the slant angle). As the slant angle decreases, holes



Lander position $X_L\ (x_L, y_L, z_L)$
Lidar FOV angle ϕ_l
Slant range r_s
Target landing point position $X_f\ (x_f, y_f, 0)$
Slant angle α_s
y
w_y
$w_x (= w_y)$
x

Figure 6. Illustration of the field of view of the Lidar DEM

Algorithm 2 Observation Data Generation at Each Timestep

```
Input: true probabilistic safety value map of the entire terrain $V_p(1000 \times 1000)$
  true position of the lander in a surface fixed frame $X_L = [x_l, y_1, z_l]$
Output: noisy probabilistic safety value map $O_P(64 \times 64) \in [0, 1]$

  // calculate the FOV of the DEM and return 2D array of center position of each pixel in the
  // observed DEM
  $X_D, Y_D, w_x, w_y = \text{CALC\_FOV}(X_L, \phi_l)$                ▷ $\phi_l$: Lidar FOV, Eq.4
  for $row = 1, \cdots, m$ do                                          ▷ for each pixel in the DEM
      for $col = 1, \cdots, n$ do
          $x_d, y_d = X_d[row][col], Y_d[row][col]$
          $\theta = \text{SLANT\_ANGLE}(x_d, y_d, X_L)$
          if $\theta > 20°$ then                               ▷ slant angle to the pixel is over thershold
              $O_P[row][col] = 0$
          else
              // calculate the safety probability of the pixel by finding nearest neighbor pixel in the
              // whole $V_P$ map
              $v_p = \text{NEAREST\_NEIGHBOR}(x_d, y_d, V_P)$
              $r = \text{SLANT\_RANGE}(x_d, y_d, X_L)$                                             ▷ [m]
              $O_P[row][col] = \text{CLIP}(v_p + \mathcal{N}(0, \frac{r}{500} * 0.05), 0, 1)$     ▷ Add noise to the safety value
          end if
      end for
  end for
  return $O_P, w_x, w_y$
```

in the DEM appear behind surface hazards. Therefore, regions where safety value could not be calculated also appears in the safety map which leads to fewer safe regions in the map.[28] In this paper, this tendency was modeled by assigning unsafe labels to pixels that has slant angle between the lander smaller than 70 degrees. In our settings, it is assumed that Lidar DEMs could be obtained every 5 seconds, and the DEM size was fixed to 64x64. This is based on the maximum processing time of the ALHAT algorithm to process DEM and generate a safety map. The entire observation data generation process is described in Algorithm 2.

Modeling the Problem as POMDP

The HDA sequence is modeled as a POMDP $P =< \mathcal{S}, \mathcal{A}, T, R, \Omega, O >$ as follows.

- State space $s \in \mathcal{S}$: Spacecraft state (position, velocity, mass), true DEM of the entire field
- Action space $a \in \mathcal{A}$: Lander thrust output, target landing position (determines next target position for Lidar measurement)
- Reward space $r \in R$: Consumed fuel, the safety of the final landing point, final velocity
- State transition function $T : S \times A \to \Pi(S)$: Stochastic dynamics of the spacecraft. $T(s, a, s')$ is the probability of moving to state $s' \in \mathcal{S}$ when the agent at state $s \in \mathcal{S}$ takes action $a \in \mathcal{A}$.
- Observation space Ω: Lidar DEM, predicted spacecraft state
- Observation function $O : S \times A \to \Pi(\Omega)$: Lidar DEM generation models, spacecraft state observation model

Obtaining a optimal policy (a policy that could maximize expected reward) in POMDP is difficult due to the system is partial observability of the problem. In general, a agent requires the entire history of the observation and action pairs $h_t = \{a_0, r_0, o_1, \ldots, a_{t-1}, r_{t-1}, o_t\}$. POMDP could be converted into a belief Markov decision process (belief MDP) by introducing belief state b. Belief state is a conditional probability function for $s \in S$ given the history h_t . When state transition function T and observation fucntion O is known, optimal policy of the POMDP could be obtained by approximating the belief MDP and applying value iteration method.

In this paper, we will seek to obtain the optimal policy of the above POMDP without modeling the T and O function, but by learning from transition data collected by interacting with the simulator. There are 2 approaches in model-free approach for POMDPs. The first approach is the memory-less approach, which learns Markov policy by simply considering the most recent observation o as a state. In POMDP, the bellman equation is not strictly satisfied, so deterministic policies leveraging only current state information is not guaranteed to be optimal. However, when the partial observability of the problem is not strong, memory-less approach may be sufficient. The second approach is the memory-based approach, which learns history-dependent policy that uses the entire history data. This is usually achieved by using Recurrent Neural Networks (RNNs) that could store history information as a single state.In this paper, both memory-less and memory-based approaches are tested.

GUIDANCE AND CONTROL

Observation Data Interpretation

Since 64×64 map data is used as part of an observation, policies that takes in full image data taken as an input has too large dimension to optimize. Therefore, the whole history data is transferred into a low-dimension internal state value by leveraging an auto-encoder. The role of the auto-encoder is to extract an abstract, compressed representation of the LIDAR DEM data as an latent vector z, in order to avoid directly passing down large input data to the reinforcement learning agent. The auto-encoder was trained separately with the reinforcement learning agent, as proposed in the "World Models" paper by Ha (2018). The dataset for training was collected through 5000 random rollout of the environment, and the auto-encoder was trained to minimize the difference between the observed safety map and the reconstructed safety map produced by the decoder. The following Fig.7 shows examples of the training data and the reconstructed image.

General Concept of Control

For control, the most intuitive approach is to train the agent to output 3-dimensional thrust vector that guides the lander along a robust and fuel-efficient trajectory to the selected landing point. However, learning this type of policy from scratch requires large computational cost for learning, because the output thrust

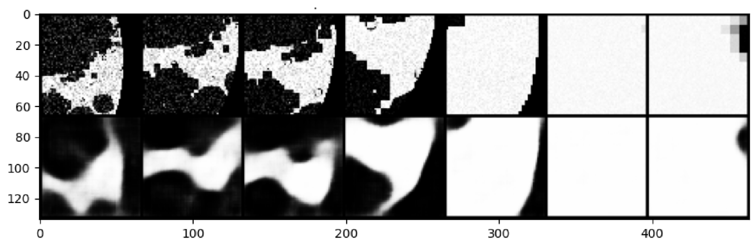


Figure 7. Original observation data (Up) and reconstructed observation data(Down) by the Autoencoder

are not constrained to guide the lander towards the selected target point. Several researches has tacked this problem by shaping the reward in an effective way,[29] but we took a different approach in order to relate the control policy with target landing points. We adopted Zero-Effort-Miss/Zero Effort-Velocity (ZEM-ZEV) feedback guidance algorithm as a baseline guidance law,[30] and designed the agent that controls the target landing point r_f, and adaptive hyper-parameters (K_R, K_V, t_{go}) in the ZEM-ZEV guidance algorithm. The idea of learning adaptive hyper-parameters in ZEM-ZEV feedback guidance algorithm with reinforcement learning has already been proposed in previous research, in order to design an ZEM-ZEV feedback controller that avoids slant angle constraint violation during descent.[31] The difference in our paper is that the target position changes during the descent which requires aggressive changes in the gains to achieve pinpoint and soft landing.

Controller

ZEM/ZEV feedback guidance algorithm calculates the optimal acceleration using the ZEM and ZEV, which represents the distance between the final target position and velocity and the projected final position and velocity when no additional control is added from current time t. The optimal acceleration is calculated as

$$\boldsymbol{a} = \frac{K_R}{t_{go}^2}\boldsymbol{ZEM} - \frac{K_V}{t_{go}}\boldsymbol{ZEV} \tag{5}$$

where K_R, K_V are control gains, and t_{go} is time-to-go. When there are no limitation in the thrust magnitude and no constraint in the trajectory, it is proved that energy-optimal trajectory could be obtained by setting the control gain as $K_R = 6, K_V = 2$. Energy optimal time-to-go t_{go} could be obtained by solving the following equation.

$$t_{go}^4\boldsymbol{g^T g} - 2t_{go}^2(\boldsymbol{v^T v} + \boldsymbol{v_f^T v} + \boldsymbol{v_f^T v_f}) + 12t_{go}(\boldsymbol{r_f} - \boldsymbol{r})^T(\boldsymbol{v} + \boldsymbol{v_f}) - 18(\boldsymbol{r_f} - \boldsymbol{r})^T(\boldsymbol{r_f} - \boldsymbol{r}) = 0 \tag{6}$$

When there is constraint $|\boldsymbol{T}| = m|\boldsymbol{a}| \leq T_{max}$ in thrust magnitude, saturated acceleration is used for control as follows.

$$\boldsymbol{a} = \begin{cases} \bar{\boldsymbol{a}} & |\bar{\boldsymbol{a}}| \leq T_{max}/m \\ \bar{\boldsymbol{a}}T_{max}/m|\bar{\boldsymbol{a}}| & |\bar{\boldsymbol{a}}| > T_{max}/m \end{cases} \tag{7}$$

$$\bar{\boldsymbol{a}} = \frac{K_R}{t_{go}^2}\boldsymbol{ZEM} - \frac{K_V}{t_{go}}\boldsymbol{ZEV} \tag{8}$$

The goal of the research is to train an reinforcement learning agent that decides K_R, K_V, t_{go} and goal position r_f at each time step.

TRAINING

Reinforcement Learning

In standard reinforcement learning problems, the agent interacts with the fully-observed environment E. At each timestep t, the agent outputs an action a_t, and the environment returns the next observation $o_{t+1} = s_{t+1}$

and a step reward $r_{t+1} = g(s_t, a_t)$. The agent decides its action based on its policy $\pi(s_t)$, which maps the state to a probability distribution over possible actions. The environment follows the transition dynamics $p(s_{t+1}|s_t, a_t)$ and initial state distribution p_{s0}. The entire process could be modeled as a Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p_{s0}, p, g\}$. The goal of the agent is to learn a policy that maximizes the expectation of discounted future reward $\mathbb{E}[R_t|\mathcal{M}, \pi]$ where $R_t = \Sigma_{i=t}^{T} \gamma^T g(s_i, a_i)$. Reinforcement learning algorithms utilizes the following action-value function (or Q-function), which describes the expected return after taking action a_t in state s_t, then following policy π.

$$Q^\pi(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t] \tag{9}$$

For the Q function, it is known that the following recursive equation called the Bellman equation holds.

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[g(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^\pi(s_{t+1}, a_{t+1})]] \tag{10}$$

Deep Deterministic Policy Gradient (DDPG)

The Deep Deterministic Policy Gradient (DDPG) is an actor-critic, off-policy, model-free reinforcement learning algorithm for continuous action spaces.[32] In actor-critic methods, the critic estimates the action-value function $Q(s, a)$, while the actor produces the action given the current state based on its policy $\pi(a|s)$. As for the actor, we consider a parametarized deterministic policy $a = \pi_\phi(s)$ with parameter ϕ. ϕ could be trained using the following deterministic policy gradient theorem and the estimated Q-values by the critic.[33]

$$\nabla_\phi J(\phi) = \mathbb{E}_{s \sim \rho_\pi}[\nabla_a Q(s, a|\theta)|_{s=s_t, a=\mu(s_t)} \nabla_\phi \mu(s|\phi)|_{s=s_t}] \tag{11}$$

For the critic, in order to handle continuous state and action spaces, $Q(s, a)$ is also approximated using a function approximator parametarized by θ. The critic uses the collected data to learn parameters that better approximate $Q(s, a)$. This could be achieved by minimizing the following loss $L(\theta)$

$$L(\theta) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}[Q(s_t, a_t|\theta) - y_t)^2] \tag{12}$$

where β is an arbitrary stochastic behavior policy, and ρ^β is an state visitation distribution using policy β. y_t is the target value

$$y_t = g(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}|\theta) \tag{13}$$

In DDPG, deterministic policy is considered. For deterministic policy $a = \mu(s)$, the inner expectation in Eq.10 could be eliminated as follows.

$$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[g(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))] \tag{14}$$

Therefore, when calculating the loss in Eq.12, transitions obtained from different stochastic policy could be used. The collected transition data (s_t, a_t, r_t, s_{t+1}) is stored in a replay buffer, and then minibatch is sampled from the replay-buffer for loss calculation in Eq.12. This enables the samples for training to be distributed identically and independently which is the basic assumption for neural network training. When exploring the environment for collecting transition data, noise is added to the actor policy.

$$\bar{\mu}(s_t) = \mu(s_t, \theta_t) + \mathcal{N} \tag{15}$$

In addition, target networks are introduced in order to stabilize the training process and achieve greater convergence. When calculating Eq.13 with collected transition data (s_i, a_i, r_i, s_{i+1}), target critic with parameter θ', and target policy with parameter ϕ' are used.

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\phi')|\theta') \tag{16}$$

Target networks are updated in order to slowly track the learned networks, as follows

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta' \tag{17}$$

$$\phi' \leftarrow \tau\phi + (1 - \tau)\phi' \tag{18}$$

where $\tau << 1$.

The key advantage of the DDPG method is that it enables to train deep reinforcement learning for continuous action space off-policy by assuming a deterministic policy. This greatly improves the sample efficiency because transition data obtained by different policy in the past could be re-used for training current agent.

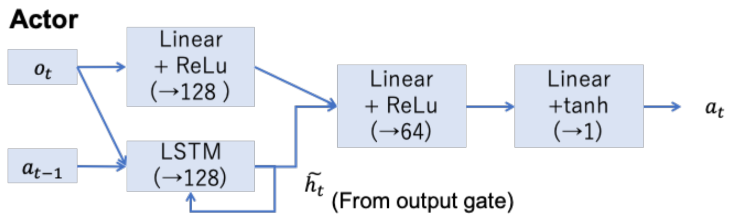


Figure 8. Architecture of the policy network

Twin Delayed Deep Deterministic Policy Gradient (TD3)

While DDPG has achieved significant performance in continuous control problems, the critical drawback is that it was sensitive to hyper parameter settings. This is because of the overestimation of the Q-values which lead the policy to fall into local optimas. The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm made 3 major changes in the DDPG algorithm to overcome this drawback.[34] The first change (clipped double Q-learning) is to use 2 seperate Q value approximators (critics), and use the smaller Q value of the 2 networks to calculate the target value y_t. smaller Q value when calculating target values. The second change is target policy smoothing. For action in the target Q value, clipped noise is added to the actual action. This prevents the Q function to have sharp peaks by returning similar Q values for similar actions. Considering the clipped double Q-learning and target policy smoothing, the target value from transition data (s_i, a_i, r_i, s_{i+1}) could be calculated as follows.

$$y_i = r_i + \gamma \min_{i=1,2} Q'(s_{i+1}, \mu'(s_{i+1}|\phi') + \epsilon|\theta'_i), \quad \epsilon \sim clip(\mathcal{N}(0,\sigma), -c, c) \tag{19}$$

The last change is the delayed updates of the policy and the target networks. In order to stabilize the learning process, updates of the target networks and policy networks are carried out less frequently (example: once per every N steps) than the critic and networks.

TD3 with memory

When the system is partially observed, optimal policy and action-value function both becomes function of the entire observation-action history h_t. Recurrent Deterministic Policy Gradient (RDPG) algorithm uses recurrent neural networks (RNN) in the policy and the action-value networks to preserve (limited) information of the history as part of the state.[35] In the RDPG setting, $\mu(s)$ and $Q(s,a)$ could be replaced with $\mu(h)$ and (h,a). Thus, the policy update could be obtained as follows.

$$\nabla_\phi J(\phi) = \mathbb{E}_{\tau\sim\nu_\mu}\left[\sum_t \gamma^{t-1} \nabla_a Q(h,a|\theta)|_{h=h_t, a=\mu(h_t)} \nabla_\phi \mu(h|\phi)|_{h=h_t}\right] \tag{20}$$

The critic and actor networks are where $\tau = (s_0, o_0, a_0, s_1, o_2, a_2,)$ are drawn from the trajectory distribution ν_μ generated by the current policy μ . In this paper, the RDPG algorithm was implemented by adding LSTM networks in the critic and actor networks. LSTM is a RNN architecture that is composed of a cell, an input gate, an output gate, and a forget gate. The critic and actor agent inputs the observation and action history h_t to their LSTM, and uses the output of the output gate $\tilde{h}_t$ to calculate their outputs. The network structure for the critic and and the actor is shown in Fig.8,9. This architecture is based on the works of Peng (2018) of robotic control. The upper network in the figure uses the current observation (and action for critic), while the bottom network utilizes past information stored in the LSTM. The network was trained with the TD3 algorithm, as shown in Algorithm 3. The difference from the original paper is that in our implementation, we created multiple replay buffers to store transition sequence of different episode length T. In our environment, the episode length varied from 4-10 timesteps. We stored transition data with different length to separate replay-buffers so that we can create mini-batches with same episode length when training.

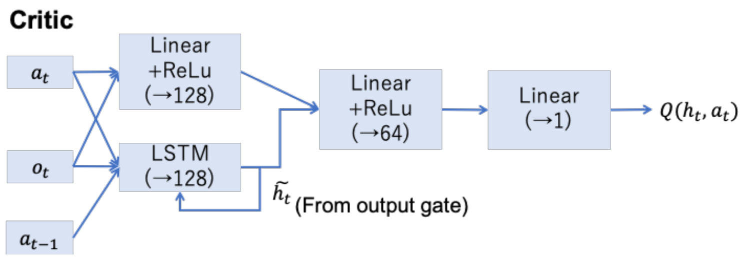


Figure 9. Architecture of the critic network

Algorithm 3 TD3 with LSTM

```
Initialize Critic Network $Q_{\theta_1}, Q_{\theta_2}$, and actor network $\mu_\phi$ with random parameters $\theta'_1, \theta'_2, \mu'$
Initialize target networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize list of replay buffer list $\mathcal{B}[]$
for episodes = 1, M do
    initialize empty history $h_0$
    $a_0 \leftarrow$ RANDOM($\mathcal{A}$),    $t \leftarrow 0$
    while not done do
        $t \leftarrow t + 1$
        receive observation $o_t$, reward $r_{t-1}$
        $h_t \leftarrow h_{t-1}, a_{t-1}, o_t$        ▷ Append observation and previous action to history
        Select action $a_t = \mu(h_t|\phi) + \epsilon$,    $\epsilon \sim \mathcal{N}(0, \sigma)$
        If done $d_t \leftarrow 1$, else $d_t \leftarrow 0$
    end while
    Store the sequence $(a_0, o_1, r_0, d_0, \ldots, a_{t-1}, o_t, r_t, d_t)$ in $\mathcal{B}[t]$
    $T \leftarrow$ RANDOM($t_{max}$)        ▷ Select the sequence length to sample from
    Sample a mini-batch of $N$ episodes with length $T$:
    $(a^i_0, o^i_1, r^i_0, d^i_1, \ldots, a^i_{T-1}, o^i_T, r^i_{T-1}, d^i_T)_{i=1,\ldots,N}$ from $\mathcal{B}[T]$
    Construct $T \times N$ set of sequences $h^i_t = (a^i_0, o^i_1, r^i_0, d^i_1 \ldots, a^i_{t-1}, o^i_t, r^i_{t-1}, d^i_t)$
    Compute target values for each sample episode for $t = 0, \ldots, T-1$
        $y^i_t = r^i_t + (1 - d^i_{t+1})\gamma \min_{k=1,2} Q'(h^i_{t+1}, \mu'(h^i_{t+1}|\phi') + \epsilon|\theta'_k)$,    $\epsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$
    Update critic by minimizing the loss
        $\theta_k \leftarrow argmin_{\theta_k} \frac{1}{NT} \sum_i \sum_t (y^i_t - Q(h^i_t, a^i_t))^2$
    Update actor parameter $\phi$ by the deterministic policy gradient
        $\nabla_\phi J(\phi) = \frac{1}{NT} \sum_i \sum_t \nabla_a Q(h, a|\theta)|_{h=h^i_t, a=\mu(h^i_t)} \nabla_\phi \mu(h|\phi)|_{h=h^i_t}$
end for
```

Overall Framework and Training Process

The observation o_t which is used as an input of the agent is summarized in Table.1 and the output of the agent a_t is summarized in Table.2. For the training agent, we tested 2 architectures: a simple TD3 algorithm for memory-less approach, and a TD3 with LSTM approach for memory-based approach. For the architecture of TD3 agent without memory, we used the same architecture as the upper network of the policy and critic networks shown in Fig.8, 9. In order to stabilize the training, all action parameters were scaled to range [0,1] during training. In the actor, the target point at timestep t ($= x^{(t)}_f, y^{(t)}_f$) is not outputted directly. Instead 2 variables $\alpha_x, \alpha_y \in [-0.25, 0.25]$ is outputted from the actor, and calculated the next target point using the following equation.

$$x^{(t)}_f = x^{(t-1)}_f + \alpha_x w_x, \quad y^{(t)}_f = y^{(t-1)}_f + \alpha_y w_y \tag{21}$$

Table 1. Elements of the observed state o_t

content	symbol	size
(true) lander position	r_t	3
(true) lander velocity	v_t	3
(true) lander mass	m_t	1
width of the observed DEM range in x,y direction	$w_x^{(t)}, w_y^{(t)}$	2
current target position	$x_f^{(t)}, y_f^{(t)}, z_f^{(t)}$	3
latent vector of the generated safety map encoded through autoencoder	z_t	32
total		44

Table 2. Elements of the action a_t

content	symbol	size	range
gain of the ZEM-ZEV control	K_R	1	[5, 7]
gain of the ZEM-ZEV control	K_V	1	[1, 3]
degradation of time-to-go from previous timestep	δt_{go}	1	[4.25,5.75]
target landing point position within the DEM FOV	α_x, α_y	2	[-0.5, 0.5]
total		5	

By this way, the next target landing point is always selected to be within the FOV of the latest observed DEM. This prevents control failures due to large deviations of the target between observations. The range of α_x and α_y was set to $\alpha_x, \alpha_y \in [-0.25, 0.25]$. The calculated new target point will be the center of the FOV of the DEM in the next step. Therefore, α_x and α_y are important output that has impact on both guidance and observation. In addition, the actor does not directly output t_{go} of the next step, but instead output the degrade of the time-to-go until the next observation timing which will be conducted 5 seconds later ($= \delta t_{go}$). The initial t_{go} is calculated using the initial lander position and target landing site and Eq.6.

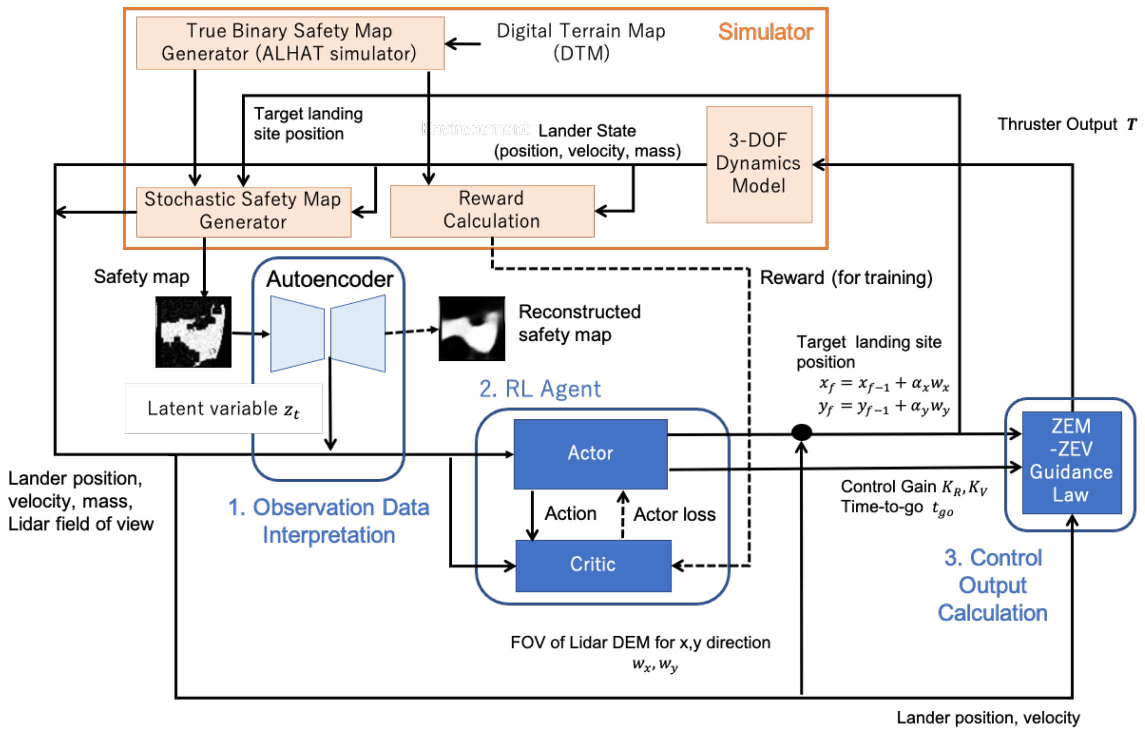
Reward setting is a important key in reinforcement learning algorithm. The reward at timestep t was implemented as follows.

$$\begin{aligned} r_t = \alpha_m \frac{m_{t-1} - m_t}{m_0 - m_{dry}} + \alpha_f (U(z_{max} - z_t) + U(m_{dry} - m_t)) + \alpha_v d_t(|\boldsymbol{v_t}|) \\ + \alpha_r d_t(|\boldsymbol{r_t} - \boldsymbol{r_f^{(t-1)}}|) + \alpha_s d_t V_D(r_t) \end{aligned} \tag{22}$$

where

$$\begin{aligned} U(x) &= \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ d_t &= \begin{cases} 1 & \text{if episode done} \\ 0 & \text{otherwise} \end{cases} \\ V_D(r_t) &= \begin{cases} 1 & \text{if } r_t \text{ is a safe landing point} \\ -1 & \text{if } r_t \text{ is not a safe landing point} \end{cases} \\ z_{max} &= 50 \quad \text{(maximum height of the terrain)} \end{aligned} \tag{23}$$

The term with α_m is a penalty for fuel consumption. The term with α_f is a penalty term for breaking the 2 critical path constraints: hitting the ground and running out of fuel. The term with α_v is a penalty for



True Binary Safety Map Generator (ALHAT simulator)
Digital Terrain Map (DTM)
Simulator
Target landing site position
Lander State (position, velocity, mass)
3-DOF Dynamics Model
Thruster Output T
Stochastic Safety Map Generator
Reward Calculation
Safety map
Autoencoder
Reconstructed safety map
Reward (for training)
Target landing site position
$x_f = x_{f-1} + \alpha_x w_x$
$y_f = y_{f-1} + \alpha_y w_y$
Latent variable z_t
2. RL Agent
Actor
ZEM -ZEV Guidance Law
Lander position, velocity, mass, Lidar field of view
1. Observation Data Interpretation
Action
Actor loss
Control Gain K_R, K_V
Time-to-go t_{go}
Critic
3. Control Output Calculation
FOV of Lidar DEM for x,y direction w_x, w_y
Lander position, velocity

Figure 10. Overall framework

velocity norm at final episode to ensure soft landing, and the term with α_r is a penalty for not landing to the final target landing point. These 2 terms are required because ZEM-ZEV control with adaptive gain and t_{go} parameters are not guaranteed to achieve a pinpoint soft landing to the target. While settings the weights $\alpha_m, \alpha_v, \alpha_r, \alpha_s$ is a difficult problem which depends not only on what the mission designer focuses on but the stability of the training process. It is natural to assume that α_f, α_v and α_s should take relatively large values because hitting the ground at a high rate of speed and landing in a danger zone can both lead to the immediate loss of the lander. We used $\alpha_m = 1, \alpha_f = -10, \alpha_v = 0.1, \alpha_r = 0.01, \alpha_s = 1$ in our research. Note that slope angle constraints are not considered as penalty in the reward settings. This is because we believe slope angle constraints are not directly related to the primary objective of the planetary landing, but rather heuristic constraints. By interacting with the simulator that simulates the observation process and the dynamics, we expect the agent to ”learn” to take a trajectory with adequate slope angle in order to achieve safe landing. The overall framework is summarized in Fig 10.

SIMULATION

Initial Condition and Hyper-parameter Settings

The initial state and other specs of the lander is summarized in Table.3. Parameters that have range are randomly initialized with the range every time the episode starts. The true DTM is picked randomly from list of DTMs generated before the training.

Training Process

The following figure Fig.11 illustrates the transition of the training loss of the critic and actor network, reward, and the ratio of landing to safe landing site during training, for both the agent without LSTM and with LSTM. For the figures for the losses, the value on the x-axis is multiplied by 10 in order to match the

Table 3. Simulation condition

content	symbol	value
Initial altitude (downrange) [m]	z_0	[900, 1000]
Initial crossrange distance [m]	$\sqrt{x_0^2 + y_0^2}$	[200 250]
Initial downrange velocity [m/s]	$\lvert v_z \rvert$	[20, 35]
Initial crossrange velocity [m/s]	$\lvert v_z \rvert$	[5, 10]
Initial mass of the lander [kg]	m_0	1200
Altitude of the final target point [m]	z_f	50
Dry mass of the lander [kg]	m_{dry}	1150
Specific Impulse of the lander [s]	I_{sp}	325
Maximum thrust of the lander [N]	T_{max}	12000
Thrust magnitude error ratio	$\lvert T - T_{plan} \rvert / T_{plan}$	0.05
Number of different DTMs used for training	$N_{D-training}$	70
Number of different DTMs used for variation	$N_{D-variation}$	18

scales. Values in the graph shows the moving average of 5000 episodes. When validating the performance during training, DTM datasets that are different from training datasets were used. It required around 250k iteration of training for both agent to reach around the maximum performance.

Contrary to our expectations, the agent without LSTM showed better performance compared with the agent with LSTM. The training of agent with LSTM was much more sensitive to its hyper-parameters, and we did not manage to achieve higher performance than TD3 without memory.In addition, as shown in Fig.11 agent without memory achieved much more faster convergence. It should be noted that even when training iteration numbers are the same, larger number of training episodes are required to train the memory-based agent because training update could be conducted only once per episode in the memory-based approach. The reason why agent without LSTM could achieve high performance is likely because the partial observability of our problem is not strong, thanks to the access to the stochastic safety map with relatively low errors. Adding larger errors into the observation data such as navigation errors might require the use of memory to cope with strong partial observability.

Comparison with other strategies

In order to assess the performance of the trained agent, performance of the 2 agents were compared with 2 other strategies. The first strategy is the "Fixed Control Policy". This policy ignores the K_R, K_V, t_{go} output from the agent, and uses the agent only to select the target point. The gains are fixed to $K_R = 6, K_V = 2$, and t_{go} was calculated by Eq.6. This policy was introduced to see how the agent's adjustment of gain and flight time affected the trajectory, final velocity, and landing errors to the target. The second strategy is the "Single Divert Policy". This strategy makes no use of trained agents. This policy targets the initial target point until the altitude reaches 500m, and then selects the pixel with maximum safety value in the stochastic safety map obtained at 500m altitude as the final target point. The gains are fixed to $K_R = 6, K_V = 2$, and t_{go} was calculated by Eq.6. This policy was introduced to represent a simplified version of existing landing techniques.

The performance of the four policy were tested by landing simulation of 1000 episodes with random initial conditions. Representing the trained agent, agent without memory was selected for comparison. In order to assess the performance in cases that agent needs to change its initial landing site, initial target position was chosen so that at least 80% of the area within 100 meter radius is hazardous. Fig.12 shows the histogram of the reward, landing error to the final target point[m], final velocity magnitude[m/s], maximum thrust[N], and minimum slant angle [deg]. The mean value is summarized in Table.4.

The trained memory-less agent achieved maximum ratio (94.8%) of landing to safe landing site. The fixed control policy had a lower safe landing ratio to the agent policy, due to its limitation in the changing the thrust

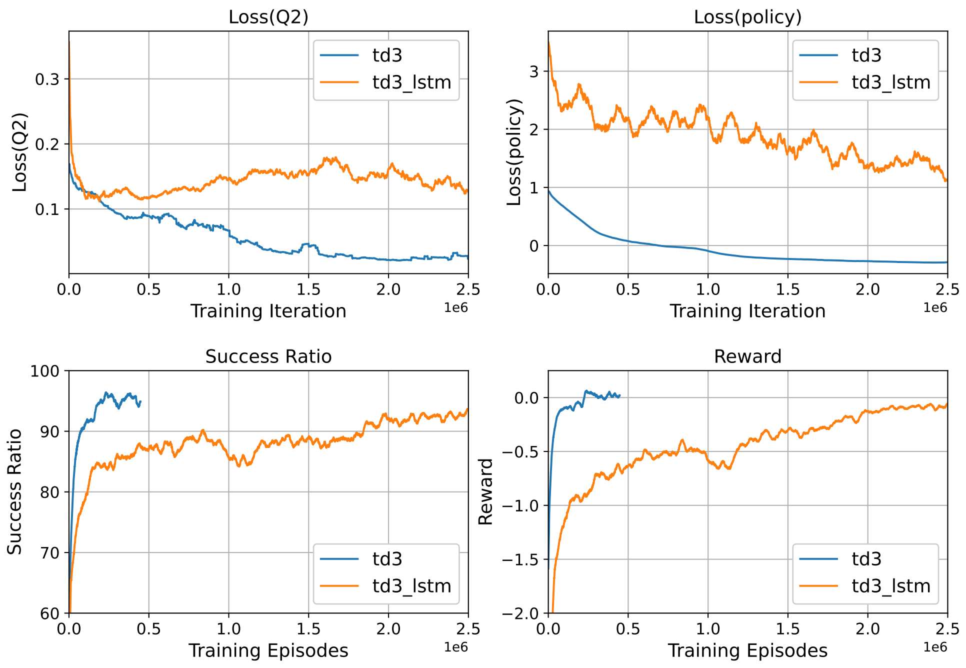


Figure 11. Training log. The agent without LSTM achieved better obtained reward.

magnitude flexibly when long divert maneuver is required. Fig.13 shows an example of such a situation. In this case, the initial target point (the center of the red square) is inside the hazardous region (black area of the figure), and the target landing point has to be shifted toward upside of the figure in order to achieve a safe landing. While the trained agent reduces the z axis velocity immediately as shown in Fig.13-(b) in order to maintain the altitude and wide field of view for searching safe landing sites, the fixed controller does not aggressively thrust the propellant as shown in Fig.13-(d). Therefore the field of view of the fixed gain controller is kept limited, and the fixed controller fails in finding a wide safe area to land.

The trained agent with memory had the lowest probability of landing to safe landing site, and largest landing position error and velocity error among the 4 policies. On the contrary, the average fuel consumption was the lowest among the 4 policies. This implies that the agent was trapped into a local optima policy that increases reward by reducing fuel consumption than improving other metrics.

The single observation policy fails in selecting safe landing sites when safe landing sites could not be obtained in the single observation due to limitation in the field of view or slant angles. The rate of failure may be reduced by selecting optimal altitude for the divert decision or implementing better landing site selection algorithms from safety maps. However, in reality, the risk of single observation policy is likely to increase than our simulator due to additional error sources (navigation errors, attitude constraints, etc) which are not simulated in our research. These errors degrade the quality of the DEM (or safety map) and the accuracy of lander guidance to the target point.

While the trained agent has high potential in adeptly changing its target position and control outputs to find safe landing sites, they also have limitations in reducing velocities and landing error to the final target point decided by the agent. Since convergence is not guaranteed when control gain in ZEM-ZEV control are changed, this disadvantage is inevitable without further refinements. In addition, fuel consumption of the memory-less trained agent was larger compared with other 2 comparison agents, due to large thrust magnitude during descent and frequent changes of the target point. Addressing the trade-off between safety

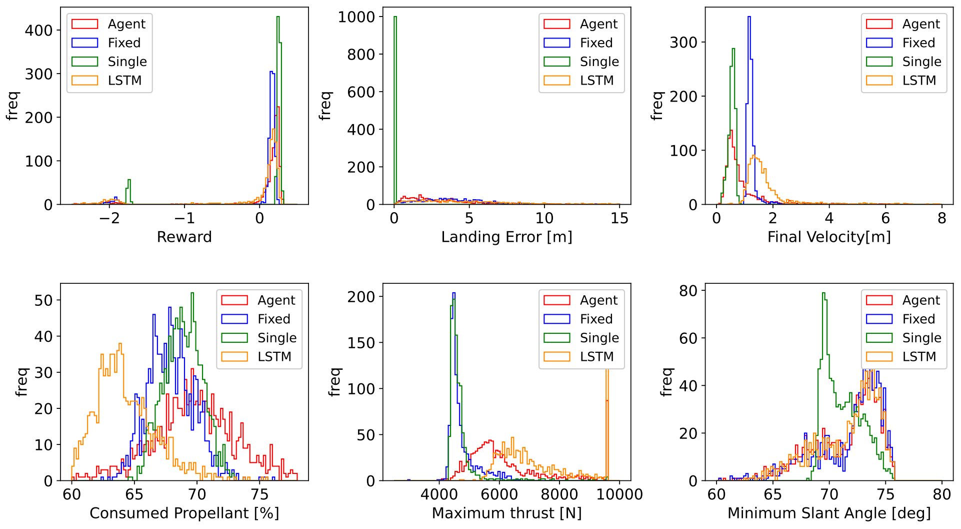


Figure 12. Evaluation of performance. 'Agent' is the trained memory-less agent, 'LSTM' is the trained memory-based agent, 'Fixed' is the policy with fixed ZEM-ZEV parameters, and 'Single' is policy with single divert maneuver and final landing site selection by directly using the stochastic safety map.

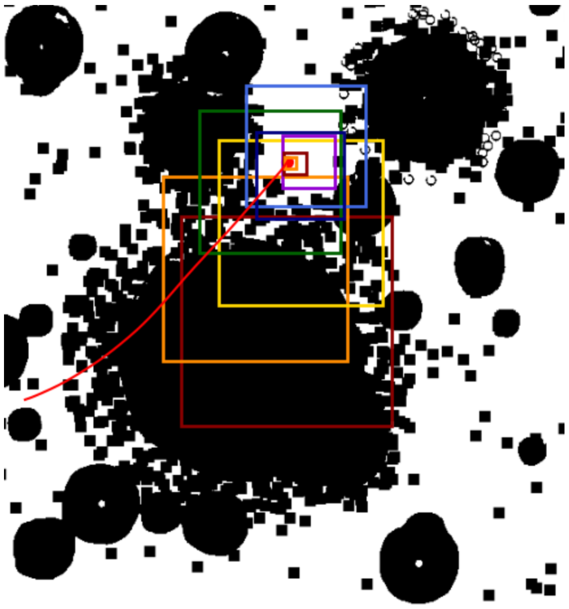
Table 4. Comparison of performance between methods (1000 episodes. Mean value)

Criteria	Memory-Less Agent	Memory-Based	Fixed Control	Single Divert
Safe Landing Ratio [%]	94.8 %	88.3 %	93.2 %	89.7 %
Distance to final target [m]	2.492	4.547	3.786	0.005
Final Velocity [m/s]	0.637	1.520	1.189	0.557
Propellant Consumption [%]	69.88	63.70	67.84	68.89
Minimum Slant Angle [deg]	72.429	72.570	72.795	70.85
Maximum Thrust [N]	5818	6863	4553	4556

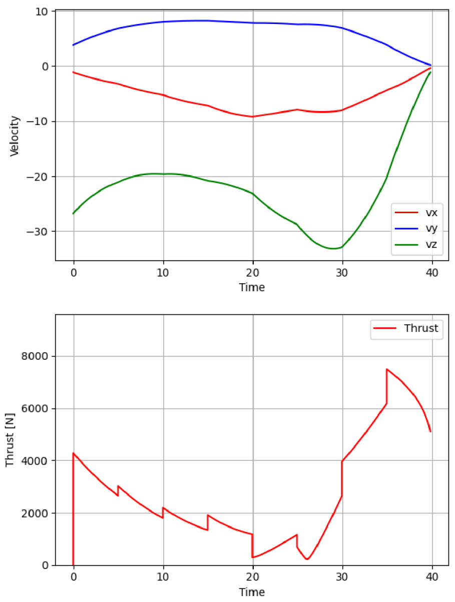
and fuel consumption from the point of the frequency of the target point changes is an interesting future work.

CONCLUSION

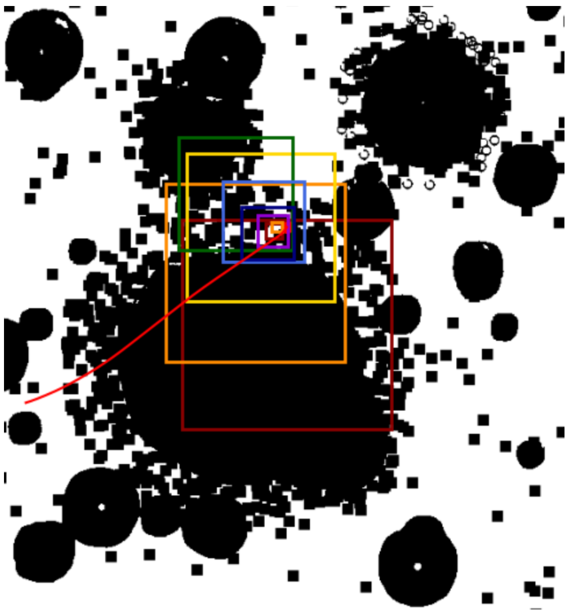
In this paper, we propose a new learning-based framework for Hazard Detection and Avoidance (HDA) phase which successively updates the target landing site and control parameters simultaneously after each observation, in order to cope with the coupling between observation, guidance, and control. We modeled the HDA sequence as a POMDP, and a reinforcement learning agent that interprets the the obtained map using autoencoder and outputs control parameters for ZEM-ZEV controll was developed to find an optimal policy. The agent was trained by a interacting with the simulator, and the trained agent was able to achieve over 90% probability of successful landing at difficult landing sites where over 80% of the terrain was hazardous around the initial target point, by gradually updating the target landing point towards safe regions. However, even when frequent update of the target point is not necessary, the trained agent tends to waist fuel and degrade landing accuracy by updating the target point every time new observation is obtained. Modifying the agent to decide optimal observation timing and divert maneuver timing as well may contribute in enhancing the performance.



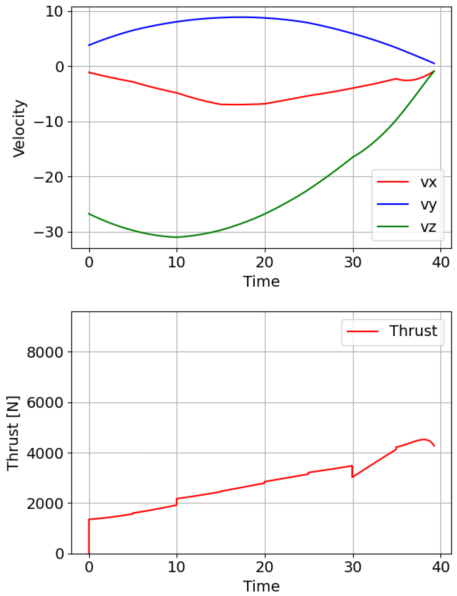
(a) Trajectory by agent controller (White:Safe Black:Unsafe)



(b) Control history by the agent controller (Up:Velocity Down:Thrust)



(c) Trajectory by fixed controller (White:Safe Black:Unsafe)



(d) Control history by the fixed controller ((Up:Velocity Down:Thrust)

Figure 13. Comparison of the obtained agent and a ZEM-ZEV controller with fixed gain. Red line in figure(a),(c) shows the lander trajectory, while the square shows the field of view (FOV) of the Lidar DEMs.

There is still a lot to be done for this research. Our main future work will be the refinement of the simulator model including incorporation of accurate Lidar DEM generation models, expansion of dynamics to 6-DOF, and incorporation of lander navigation algorithm. As the partial observability of the environment increases by incorporating various error sources, a memory-based agent might be required for sufficient performance.

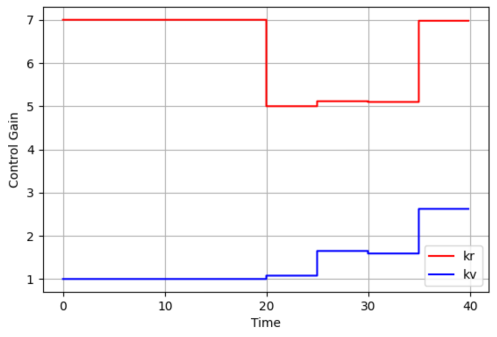


Figure 14. Changes in the control gain K_R, K_V while control in Fig.13-(a),(b)

We are also planning to test other baseline control policies and observation data interpretation architectures in order to improve the agent's performance and stability.

ACKNOWLEDGEMENTS

This material is partially based upon work supported by the National Aeronautics and Space Administration under Grant No.80NSSC20K0064 through the NASA Early Career Faculty Program.

REFERENCES

[1] A. E. Johnson, A. R. Klumpp, J. B. Collier, and A. A. Wolf, “Lidar-based hazard avoidance for safe landing on Mars,” *Journal of guidance, control, and dynamics*, Vol. 25, No. 6, 2002, pp. 1091–1099.

[2] N. Serrano and H. Seraji, “Landing site selection using fuzzy rule-based reasoning,” *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 4899–4904.

[3] A. Huertas, Y. Cheng, and L. H. Matthies, “Real-time hazard detection for landers,,” *NASA Science Technology Conference*, 2007.

[4] L. Matthies, A. Huertas, Y. Cheng, and A. Johnson, “Stereo vision and shadow analysis for landing hazard detection,” *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 2735–2742.

[5] Y. Cheng, A. E. Johnson, L. H. Mattheis, and A. A. Wolf, “Passive imaging based hazard avoidance for spacecraft safe landing,” *I-SIRAS 2001: 6th International Symposium on Artifical Intelligence, Robotics and Automation in Space*, 2001, pp. 1–14.

[6] N. Serrano, “A bayesian framework for landing site selection during autonomous spacecraft descent,” *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5112–5117.

[7] C. I. Restrepo, R. Lovelace, R. R. Sostaric, J. M. Carson, and N. G. Spaceflight, “NASA SPLICE Project : Developing the Next Generation Hazard Detection System,” *2nd RPI Space Imaging Workshop*, 2019, pp. 2–3.

[8] A. D. Cianciolo, S. Striepe, J. Carson, R. Sostaric, D. Woffinden, C. Karlgaard, R. Lugo, R. Powell, and J. Tynis, “Defining navigation requirements for future precision lander missions,” *AIAA Scitech 2019 Forum*, No. January, 2019, pp. 1–18, 10.2514/6.2019-0661.

[9] J. M. Carson, M. M. Munk, R. R. Sostaric, J. N. Estes, F. Amzajerdian, J. Bryan Blair, D. K. Rutishauser, C. I. Restrepo, A. D. Cianciolo, G. T. Chen, and T. Tse, “The splice project: Continuing nasa development of gn&c technologies for safe and precise landing,” *AIAA Scitech 2019 Forum*, No. January, 2019, pp. 1–9, 10.2514/6.2019-0660.

[10] T. Brady, E. Robertson, S. P. C. App, and D. Zimpfer, “Hazard detection methods for lunar landing,” *2009 IEE Aerospace Conference*, 2009, pp. 1–18.

[11] C. D. App and T. B. Smith, “Autonomous precision landing and hazard detection and avoidance technology (ALHAT),” *2007 IEEE Aerospace Conference*, 2007, pp. 1–7.

[12] T. Ivanov, A. Huertas, and J. M. Carson, “Probabilistic Hazard Detection for Autonomous Safe Landing,” *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 5019.

[13] T. Brady and J. Schwartz, “ALHAT system architecture and operational concept,” *2007 IEEE Aerospace Conference*, 2007, pp. 1–13.

[14] S. A. Striepe, C. D. Epp, and E. A. Robertson, “Autonomous precision landing and hazard avoidance technology (ALHAT) project status as of May 2010,” *International Planetary Probe Workshop 2010 (IPPW-7)*, 2010.

[15] D. Rutishauser, C. Epp, and E. Robertson, “Free-Flight Terrestrial Rocket Lander Demonstration for NASA’s Autonomous Landing and Hazard Avoidance Technology (ALHAT) System,” *AIAA SPACE 2012 Conference Exposition*, 2012, p. 5239.

[16] C. Epp, E. Robertson, and J. M. Carson, “Real-time hazard detection and avoidance demonstration for a planetary lander,” *AIAA SPACE 2014 Conference and Exposition*, 2014, p. 4312.

[17] A. E. Johnson, A. Huertas, R. A. Werner, and J. F. Montgomery, “Analysis of on-board hazard detection and avoidance for safe lunar landing,” *2008 IEEE Aerospace Conference*, 2008, pp. 1–9.

[18] A. Huertas, A. E. Johnson, R. A. Werner, and R. A. Maddock, “Performance evaluation of hazard detection and avoidance algorithms for safe Lunar landings,” *2010 IEEE Aerospace Conference*, 2010, pp. 1–20.

[19] B. E. Cohanim and B. K. Collins, “Landing point designation algorithm for lunar landing,” *Journal of Spacecraft and Rockets*, Vol. 46, No. 4, 2009, pp. 858–864.

[20] S. Paschall and T. Brady, “Demonstration of a safe precise planetary landing system on-board a terrestrial rocket,” *2012 IEEE Aerospace Conference*, 2012, pp. 1–8.

[21] N. Trawny, A. Huertas, M. E. Luna, C. Y. Villalpando, K. Martin, J. M. Carson, A. E. Johnson, C. Restrepo, and V. E. Roback, “Flight testing a Real-Time Hazard Detection System for Safe Lunar Landing on the Rocket-Powered Morpheus Vehicle,” *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2015.

[22] N. Serrano, “A Bayesian framework for landing site selection during autonomous spacecraft descent,” *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 5112–5117, 10.1109/IROS.2006.282603.

[23] S. R. Ploen, H. Seraji, and C. E. Kinney, “Determination of spacecraft landing footprint for safe planetary landing,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 45, No. 1, 2009, pp. 3–16, 10.1109/TAES.2009.4805259.

[24] P. Cui, D. Ge, and A. Gao, “Optimal landing site selection based on safety index during planetary descent,” *Acta Astronautica*, Vol. 132, No. July 2016, 2017, pp. 326–336, 10.1016/j.actaastro.2016.10.040.

[25] M. Barker, E. Mazarico, G. Neumann, M. Zuber, J. Haruyama, and D. Smith, “A new lunar digital elevation model from the Lunar Orbiter Laser Altimeter and SELENE Terrain Camera,” *Icarus*, Vol. 273, 2016, pp. 346 – 355, https://doi.org/10.1016/j.icarus.2015.07.039.

[26] R. J. Pike, “Size-dependence in the shape of fresh impact craters on the moon.,” *Impact and Explosion Cratering: Planetary and Terrestrial Implications* (D. J. Roddy, R. O. Pepin, and R. B. Merrill, eds.), Jan. 1977, pp. 489–509.

[27] D. E. Bernard and M. P. Golombek, “Crater and rock hazard modeling for Mars landing,” *AIAA Space 2001 Conference and Exposition*, Albuqurque, NM, 2001, 10.2514/6.2001-4697.

[28] C. I. Restrepo, P.-T. Chen, R. R. Sostaric, and J. M. Carson, “Next-Generation NASA Hazard Detection System Development,” 2020, pp. 1–10, 10.2514/6.2020-0368.

[29] B. Gaudet, R. Linares, and R. Furfaro, “Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Landing,” *Advances in Space Research*, 01 2020, 10.1016/j.asr.2019.12.030.

[30] B. Ebrahimi, M. Bahrami, and J. Roshanian, “Optimal sliding-mode guidance with terminal velocity constraint for fixed-interval propulsive maneuvers,” *Acta Astronautica*, Vol. 62, No. 10, 2008, pp. 556 – 562, https://doi.org/10.1016/j.actaastro.2008.02.002.

[31] R. Furfaro, A. Scorsoglio, R. Linares, and M. Massari, “Adaptive Generalized ZEM-ZEV Feedback Guidance for Planetary Landing via a Deep Reinforcement Learning Approach,” 2020, 10.1016/j.actaastro.2020.02.051.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning.,” *ICLR* (Y. Bengio and Y. LeCun, eds.), 2016.

[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic Policy Gradient Algorithms,” *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML’14, JMLR.org, 2014, p. I–387–I–395.

[34] S. Fujimoto, H. Hoof, and D. Meger, “Addressing Function Approximation Error in Actor-Critic Methods,” *International Conference on Machine Learning*, 2018, pp. 1582–1591.

[35] N. M. O. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, “Memory-based control with recurrent neural networks,” *ArXiv*, Vol. abs/1512.04455, 2015.