Spacecraft maneuvering near small bodies using reinforcement learning (Project Proposal)

Aleksandrina Kirkova,[*] Goran Trayanov,[†] and Simona Grigorova[‡]

(Dated: December 5, 2020)

I. MOTIVATION

Spacecraft maneuvering is a difficult task. In the case of a controlled spacecraft we have two instances - either it is manned or controlled from afar. In the former there is danger for the man aboard and the latter is limited because the signal from the spacecraft has to travel to the control room and back, therefore the spacecraft's reaction time is at least $2c * \text{distance} - \text{to} - \text{control} - \text{room}$. So, in a dynamic environment, the idea of a spacecraft that can test its surroundings and then act accordingly with a relatively short reaction time, comes natural. In this project we humor the idea of a spacecraft hovering above a small dynamic celestial body, closely following the approach in [1].

II. DESCRIPTION OF THE PROBLEM

A. Building the environment

1. **Description of the spacecraft and the asteroid**
We model the small celestial body by an ellipsoid with uniform density ρ and smooth surface. We assume the asteroid is tumbling with principal moments of inertia I_x, I_y, I_z. The ellipsoid has dimensions a, b, c along a non-inertial reference frame $\mathcal{F}_S$ with an origin at its center of mass.
The equations of motion EoM of a spacecraft around a small body are taken from [1] and are as follows:

$$\ddot{\mathbf{r}} = -2\omega \times \dot{\mathbf{r}} - \dot{\omega} \times \mathbf{r} - \omega \times (\omega \times \mathbf{r}) + \mathbf{a}_\text{g} + \mathbf{a}_\text{SPR} + \frac{\mathbf{T}}{m} \tag{1}$$

$$\dot{m} = -\frac{T}{I_\text{SP} g_0} \tag{2}$$

Here, with boldfaced characters we denote vector quantities; the non-boldfaced characters are the quantities' magnitudes. $2\omega \times \dot{\mathbf{r}}$ is the Coriolis acceleration, $\dot{\omega} \times \mathbf{r}$ is the Euler acceleration, $\omega \times (\omega \times \mathbf{r})$ - the centrifugal acceleration, $\mathbf{a}_\text{g}$ is the gravitational acceleration due to the small body's gravitational pull, $\mathbf{a}_\text{SPR}$ is the acceleration due to solar pressure radiation and other unknown perturbations, T is the thrust of the engines and m is the mass of the spacecraft. Said spacecraft's mass changes with time due to the expelled propellant with a derivative of $\dot{m}$ proportional to the thrust and inversely proportional to the propulsion system specific impulse I_SP and $g_0 = 9.8 m/s^2$. We evaluate $\mathbf{r}$ at each time step by using a Runge-Kutta method.

The angular velocity of the rotating asteroid ω is overall unknown, as are all of its other parameters, but we can set an upper limit to it. Given that asteroids are mostly piles of boulder with no other force but gravity keeping them together, we set the condition that the centrifugal force should not exceed the gravitational pull. Therefore:

$$\omega^2 r \leq \frac{GM}{r^2} = \frac{G\frac{4}{3}abc\rho}{r^2} \Rightarrow \omega \leq \sqrt{\frac{G\frac{4}{3}abc\rho}{r^3}}\Bigg|_{r=a} = \sqrt{\frac{G\frac{4}{3}bc\rho}{a^2}} \tag{3}$$

The acceleration due to solar pressure radiation and other unknown perturbations $\mathbf{a}_\text{SPR}$ is a purely environmental parameter which will be kept constant for all constructions (different initial conditions for the spacecraft and

[*]Electronic address: avkirkova@uni-sofia.bg
[†]Electronic address: gtrayanov@uni-sofia.bg
[‡]Electronic address: simonakg@uni-sofia.bg

different asteroids). Given we have enough time, we can try to make its magnitude in each direction a random variable sampled from a Gaussian.
The spacecraft we shall describe by its mass m, its empty mass $m/2$, the engine's specific impulse I_{SP} and its maximum thrust T. We assume that the spacecraft knows its exact location relative to the landmark at all times via optical sensors. If we have enough time, we intend to introduce random noise σ_{SN} into the agent's sensory input as well as some other random noise related to the engine's specific impulse $\sigma_{I_{\text{SP}}}$.

2. **Positioning of the spacecraft**
The spacecraft's initial position and initial velocity are randomly sampled from a uniform distribution, similarly to [2].

The hovering position $\mathbf{p}$ is obtained by an algorithm similar to sampling on a sphere [1] given in Algorithm 1. We randomly choose a point on the surface of the ellipsoid, then move that point by some scaling factor s distributed uniformly between $s_{\min}$ and $s_{\max}$. That point is our hovering position $\mathbf{p}$ above our landmark.

Algorithm 1 Sampling a point on an ellipsoid with semi-principal axes a,b and c

```
1: function Sample Position($a, b, c, s_{\min}, s_{\max}$)
2:     $u \leftarrow 2\pi.\mathcal{U}(0, 1-\epsilon)$
3:     $v \leftarrow \cos^{-1}(2\mathcal{U}(0,1) - 1)$
4:     $s \leftarrow \mathcal{U}(s_{\min}, s_{\max})$
5:     $\mathbf{p} \leftarrow \begin{pmatrix} s.a.\cos u.\sin v \\ s.b.\sin u.\sin b \\ s.c.\cos v \end{pmatrix}$
6:     return $p$
7: end function
```

3. **States, Actions and Reward functions**
The state space is six dimensional $S = [\mathbf{r} - \mathbf{p}, \dot{\mathbf{r}}]$. Here $\mathbf{r}$ is the current position and $\dot{\mathbf{r}}$ is the velocity, both obtained by a Runge-Kutta algorithm; $\mathbf{p}$ is obtained by sampling (see Algorithm 1).
Each of the spacecraft's controllers corresponds to a thruster along the same axis and is associated with a one dimensional action space, with the action being the the thrust along the axis. Detailed description of how actions are chosen, through a certain policy, is given in the next subsection.
The reward function is a negative quadratic function of the relative offset of the spacecraft to the desired hovering position:

$$\mathcal{R}(S) = -|\mathbf{r} - \mathbf{p}|^2 \tag{4}$$

B. Evolutionary Direct Policy Search

- **The Feed Forward Neural Network**
We are to approach the task of constructing a controller that is able to maneuver near small bodies as a reinforcement learning problem. We introduce the utility function $U(\pi(\theta))$ for each state-action pair that depends on the policy's parameters θ. Our aim is to extract the optimal policy which would generate the highest utility through a generational particle swamp optimiser algorithm. Following the authors of [2], we plan on implementing a direct policy search algorithm whose architecture would be represented by a Feed Forward Neural Network (FFNN), where the policy's parameters θ define the weights between the neurons of the network. The FFNN would consist of an input layer given by the state dimensions and a bias neuron, an output layer given by the action space containing the thrust vectors, and one hidden layer with six hidden neurons (the choice of these hyper-parameters is nontrivial and in our case unique to the paper [1], which is following [2], [3],[4]).

- **Generational Particle Swamp Optimizer (GPSO)**
The policy parameters are encoded into a genome that is to be used in the genetic algorithm through the reshaping of the weight matrices in the FFNN into a vector of weights. Since the goal is to evolve policies with high utility, the utility function of the policy that an individual's genome encodes is a natural choice for the parameter of fitness of an individual in the genetic algorithm.

We define a simulation $sim(\chi, \theta)$ as the evolution of the equations of motions with actions chosen following the policy $\pi(\theta)$ and a single seed χ which defines the initial conditions of the spacecraft controller and all properties of the targeted small body. Therefore we can rewrite the utility $U_{\chi_i}(\pi(\theta))$ as function dependent the simulation for a given seed and the policy defined by θ. The output of this function is deterministic for a given seed and depends solely on the policy.

With this we can proceed by giving a brief explanation of the pseudo-code for the GPSO direct policy search algorithm:

The first loop outputs the champion particle with the highest fitness which encodes the current policy with highest utility (randomly initialized). Secondly, as we let the genetic algorithm evolve, we loop through N generations, after which the champion particle is ensured to encode the best policy.

Algorithm 2 Direct policy search with a generational particle swarm optimizer

```
1: Initialize a population of particles $P = p_0, \dots, p_n$ with random speed and random positions in the search space
2: Initialize a random number generator $\tau$
3: Initialize the champion $c \leftarrow p_0$
4: Sample the first $k$ random seeds $\Xi = \chi_1, \dots, \chi_k$ from $\tau$
5: for Each particle $p_i$ do
6:     $p_i$ encodes a policy $\pi_i$
7:     Evaluate the fitness $f(\Xi, p_i) = -\frac{1}{k}\sum_{j=1}^{k} U_{\chi_i}(\pi_i)$
8:     if $f(\Xi, p_i) < f(\Xi, c)$ then
9:         $c \leftarrow p_i$
10:    end if
11: end for
12: for $N$ generations do
13:     Sample the next $k$ random seeds $\Xi = \chi_1, \dots, \chi_k$ from $\tau$
14:     for Each particle $p_i$ do
15:         Adapt the velocity of the particle
16:         Update the position of the particle
17:         $p_i'$ is the updated particle
18:         if $f(\Xi, p_i') < f(\Xi, p_i)$ then
19:             $p_i \leftarrow p_i'$
20:         end if
21:         if $f(\Xi, p_i') < f(\Xi, c)$ then
22:             $c \leftarrow p_i'$
23:         end if
24:     end for
25: end for
26: Particle $c$ now encodes the best policy
```

III. IMPLEMENTATION

A. Milestones and approximate time ranges to reach them

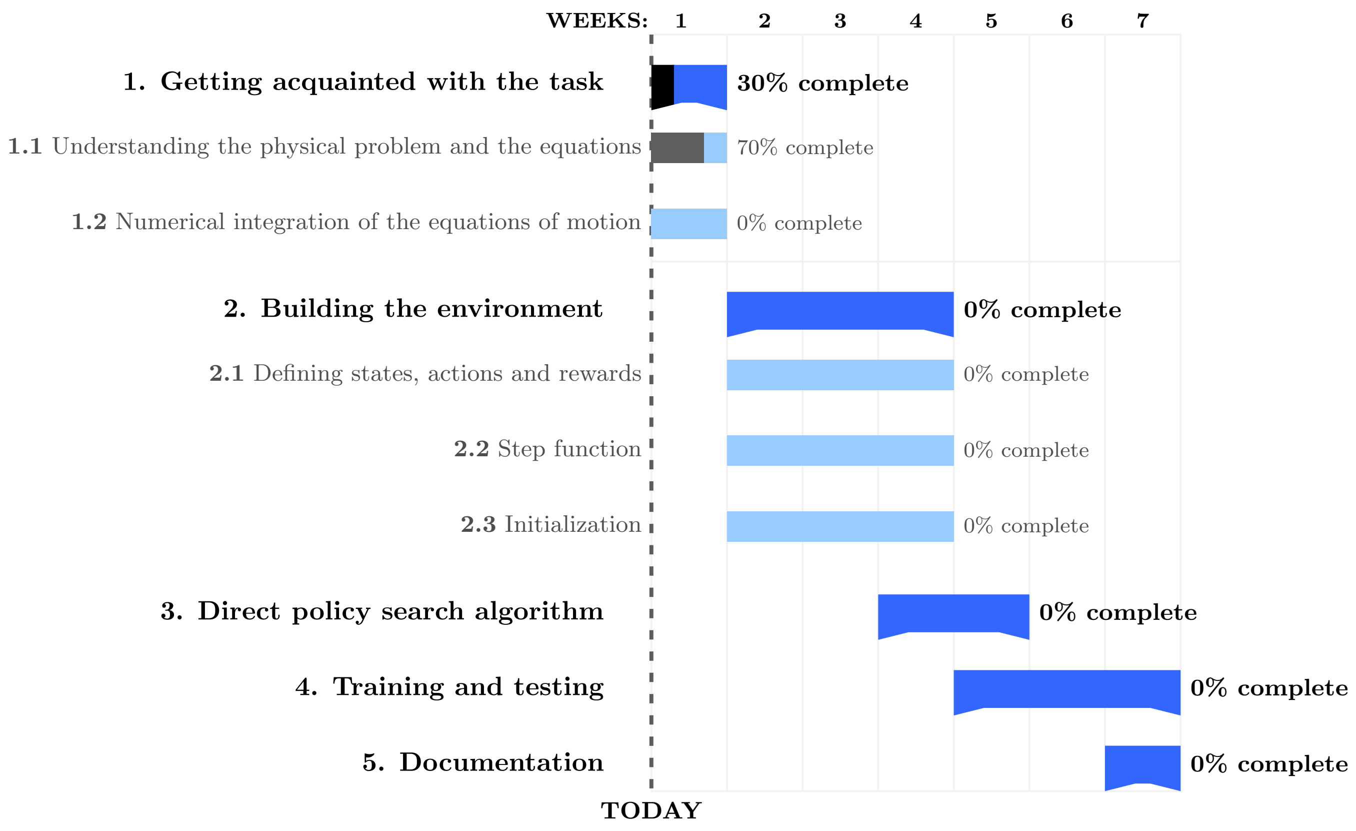
1. **Getting acquainted with the task**

   The completion of part 1 of this milestone (see Gantt chart below for reference) should be done separately by each of the students. Part 2 which consists of defining a function that would numerically integrate the equations of motions would be completed as teamwork by all students. We plan to simultaneously collaborate using the Jupyter Notebook environment. Since we have already become acquainted with the problem, the rest of this milestone should be done by December 13th.

2. **Building the Environment**

   Part 1 of this milestone (Defining the environment via states, actions and rewards) is a task assigned to Simona, while parts 2 and 3 would be mostly completed by Goran. However, we do plan on implementing teamwork whereever needed and discussing each step of all milestones together, so these are merely rough guidelines on how we have divided the task between all students. Once again, we plan on using the environment of Jupyter Notebook so that all of the students have access to the notebooks and can contribute. We aim to complete the whole task within the time frame of December 14th to January 3th.

We all plan on working together on the implementation of the Neural Network in the problem, and will divide the task into subtasks if needed as we go.



3. **Direct policy search algorithm**

   The whole task will be mainly completed by Aleksandrina, once again with help and advice from the other students when needed. The milestone should be completed by January 10th.

4. **Training, testing and Visualization**

   Training will be done using Google Colab if we require less than 12 computing hours. If more time is needed we will do the training in sessions using checkpoints to save our progress. If that goes wrong for whatever reason, we plan to use Physon - the Physics faculty's computational cluster which we have access to.

   Our goal is to produce similar images to FIG. 1 which are taken directly form [1]. The first figure represents the altitude of spacecraft over small body's surface against the period of the small body, while the second one illustrates the trajectory of the spacecraft near the small body.

   We plan for this part to take up to the last three weeks.

5. **Putting together the documentation for the project**

   Every team member will do their part in putting the project together using Overleaf online editor. This should be completed roughly in the last week before the due date.

B. Risk assessment

1. **Not managing to initialize the environment correctly**

   Possible solution: In case we are unable to define the states and actions correctly as a 6-dimensional parameter $(\mathbf{r}, \dot{\mathbf{r}})$ we could decouple it into three 2-dimensional parameters resulting in three different phase spaces for each of the degrees of freedom, which is the approach followed in [2].

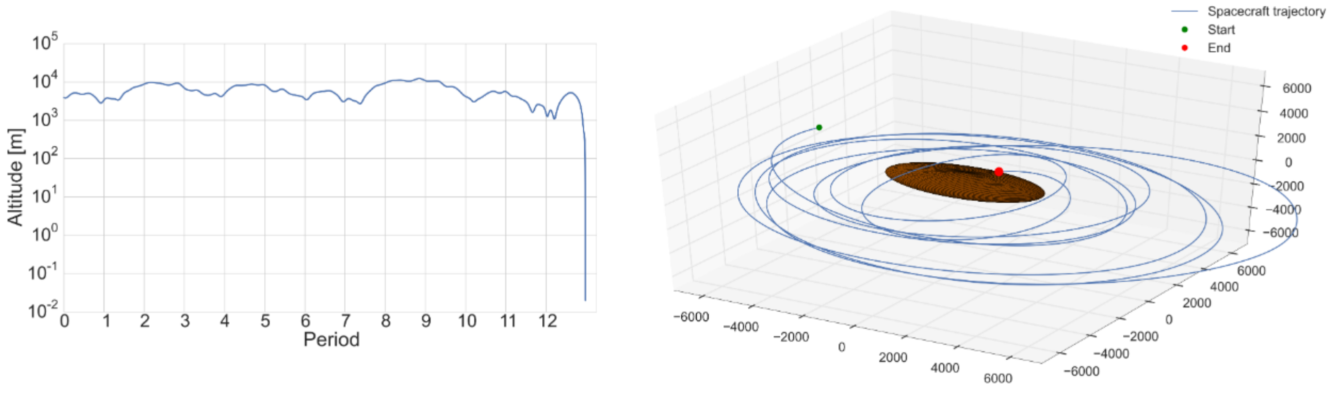


FIG. 1: Altitude of the spacecraft against the period (left), and the trajectory of the spacecraft (right).

2. **Not being able to implement the Neural Network**
   Possible solution: We could remove the hidden layer, resulting in a simplified NN with only an output and an input layer. If this still causes unsolvable difficulties, we could give up the idea of using a NN completely, and use regular Reinforcement Learning.

[1] S. Willis, D. Izzo, and D. Hennes, AAS/AIAA Space Flight Mechanics Meeting (2016).
[2] B. Gaudet and R. Furfaro, AIAA/AAS Astrodynamics Specialist Conference , 5072 (2012).
[3] B. Gaudet and R. Furfaro, IEEE/CAA Journal of Automatica Sinica **1**, 397 (2014).
[4] D. Izzo, L. Simões, and G. de Croon, Evolutionary Intelligence **7**, 107 (2014).