Post-Print

P. Peñarroya, M. Pugliatti, F. Ferrari, S. Centuori, F. Topputo, M. Vetrisano, M. Sanjurjo-Rivo
Cubesat Landing Simulations on Small Bodies Using Blender
Advances in Space Research, Published online 25/07/2022
doi:10.1016/j.asr.2022.07.044

CubeSat Landing Simulations on Small Bodies using Blender.

Pelayo Peñarroya[a,*], Mattia Pugliatti[b], Fabio Ferrari[c], Simone Centuori[a], Francesco Topputo[b], Massimo Vetrisano[a], Manuel Sanjurjo-Rivo[d]

[a] *Deimos Space S.L.U., Ronda de Poniente, 19, 28760 Tres Cantos Madrid, Spain*
[b] *Politecnico di Milano, Department of Aerospace Science and Technology, Via Privata Giuseppe La Masa, 34, 20156 Milano MI, Italy*
[c] *University of Bern, Space Research and Planetary Sciences, Physics Institute, Gesellschaftsstrasse 6, 3012 Bern, Switzerland*
[d] *Universidad Carlos III de Madrid, C. Madrid, 126, 28903 Getafe, Madrid, Spain*

Abstract

Landing on small-bodies is a very challenging problem that requires high degrees of robustness and autonomy. Being able to perform simulations with great flexibility and accuracy is paramount for the development and design of landing systems. To this end, contact dynamics plays a fundamental role and is often handled by complex tools that require large amount of development and validation efforts and very specific expertise. In the last decade, the Visual Effects (VFX) industry has developed numerous suites that deal with contact dynamics frameworks. In this work, the possibility of leveraging on the work of the VFX industry by using Blender, one of these tools, as the source for the contact dynamics modelling is investigated.

This research focuses on the description of the methodology used for the landing simulations and the validation of the tool developed. A step-by-step guide through the simulation setup is given, discussing how the wrapping GNC simulator and Blender interact. Validation tests for the different parameters and dynamic models involved in the simulations are also presented. The results refer to the landing of a CubeSat in the crater region of an asteroid. In particular, the artificial crater that will be generated on Dimorphos by NASA's DART impact in late September 2022, is considered in the simulations presented in this work. Safety maps are generated by post-processing these results, and are used to assess different landing strategies or site-selection criteria on the Dimorphos crater study case. Finally, the role of the developed tool in optimising the use of space resources and its contribution to landing design strategies is discussed.

Keywords: Landing; Blender; Contact dynamics; Asteroid; Safety map.

1. Introduction

An increasing number of missions in the recent past have attempted to perform landings on small bodies, which poses a very challenging problem in a highly uncertain environment. Designing these landing sequences in a robust manner is paramount for the success of the mission concept, and requires a wide range of tools. Having these tools is an asset that makes this complex process more manageable, reducing of the overall mission risk.

One of the key functionalities among this range of software suites involves impact simulation tools. A great effort has been put forth by the scientific community to study many different contact models and develop software capable of tackling this

*Corresponding author.
Email addresses: pelayo.penarroya@deimos-space.com (Pelayo Peñarroya), mattia.pugliatti@polimi.it (Mattia Pugliatti), fabio.ferrari@unibe.ch (Fabio Ferrari), simone.centuori@deimos-space.com (Simone Centuori), francesco.topputo@polimi.it (Francesco Topputo), massimo.vetrisano@deimos-space.com (Massimo Vetrisano), manuel.sanjurjo@uc3m.es (Manuel Sanjurjo-Rivo)

https://dx.doi.org/10.1016/j.jasr.xxxx.xx.xxx
0273-1177/

complex task. Thuillet et al. (2021) performed a campaign of hundreds of numerical simulations using a soft-sphere discrete element method based on the Mobile Asteroid Surface Scout (MASCOT) lander, carried by the Hayabusa2 mission (Grimm et al., 2013), using the Parallel K-D tree GRAVity code (PKDGRAV) code (Dikaiakos & Stadel, 1996), which is a parallel hierarchical tree-structured code used to conduct cosmological simulations on shared-memory and message-passing multiprocessors. In Sunday, Cecily et al. (2021), a comparative study was performed to assess the performances of that code with respect to Chrono. Van Wal et al. (2020) presented a methodology for fast, parallelized simulation of the bouncing deployment of landers and rover probes onto a small celestial body. All of these works developed their own simulation framework and offer accurate analyses about the specific interactions between the spacecraft and the terrain, for a very detailed study-case. However, the development of such simulation frameworks is a time-consuming and technically challenging part of such projects that require very specific expertise.

On the other hand, and with the huge rise of the video-game and Visual Effects (VFX) industries, many tools have been developed to perform visual rendering of a certain scene, under a set of physical constraints, e.g., impacts, fluid dynamics, or deformations. These industries have invested in improving the quality of such effects and it is precisely in these sectors where some of the most powerful and renowned tools are developed (Senkic, 2010; Ivaldi et al., 2014). These frameworks offer the advantage of being Commercial Off-The-Shelf (COTS) tools that are *ready-to-use*, i.e., the development and validation effort is avoided by the user. They are, however, designed for a certain segment of public, namely, VFX industries, professionals, and enthusiasts. This translates into software primarily conceived to be used for visualization purposes.

As a consequence, parameters and settings used in such tools by the user often do not exhibit a clear link with real-world physical properties, to support engineering purposes. The result is a framework whose scenario-setting parameters include for example, the bounciness of a certain rigid body or the strength of a force field. These, however, are terms which are neither used nor unequivocally defined within the scientific and engineering communities. Therefore, behind this implementation, a great deal of calibration (or translation from visualization to physics) is needed. The interest of this objective is to find out how large this effort is and to analyse how accurate the obtained results can be, i.e., to investigate whether adapting already existing tools to asteroid landing simulation is worth the effort or whether it would be preferred to tailor a software suite for the needs of a certain space project.

To comprehend the reach that such a tool should have, it is key to understand how far traditional methodologies have gone in the past decades. Robotics is the discipline in which contact dynamics has been studied the most and longest (see Bekker (1960), for instance), and has been only relatively recently applied to space-related studies. Krenn & Hirzinger (2008), for instance, developed and studied a contact dynamic model for both satellite docking and rover locomotion on planetary surfaces. There, a detailed description of their contact detection models and included forces is given. To perform contact detection, the coordinates of the meshes involved in the simulation were transformed to a common reference frame. Then, these points are mapped into a two-dimensional radial-axial reference frame. The computational effort of using this technique was mitigated by the authors by choosing to reduce the dimensions (and thus the matrix operations) from three to two dimensions. Parameters like normal penetration or penetration velocity are key for the computation of the resulting forces in the dynamical system, based on the model by Bekker (1960), which has been extensively used in other robotic locomotion research projects like in Gerhart (2004). An updated version of that methodology was introduced by Zhang et al. (2021), where their authors use Polygonal Contact Model (PCM)s to simulate six-degree-of-freedom (6-DoF) motion of arbitrary-shaped landers in the neighborhood of a small body with multi-contact dynamics.

Van Wal et al. (2017) presented a parametric analysis to study the role that different design and physical parameters play in the landing design and their corresponding sensitiv-

ities. Their simulation framework took restitution, Coulomb friction, and rolling resistance into account, and included a persistent rock model, which was generated stochastically and procedurally (similarly to what was developed by Tardivel et al. (2014a)). One of the key conclusions of this work highlights the importance of modelling the full rigid-body dynamics of the spacecraft or lander, in opposition to the traditionally used spherical models. On top of that, it was found that the Coefficient of Restitution (CoR) of the surface is the most significant influence in the simulation results, while the friction has a very low impact in the deployment.

For the Hayabusa mission, (Yano et al., 2006), Takashi Kubota et al. (2008) showed how they modelled their sampler horn and the analysis they run to study interactions with the terrain. Later, for the Hayabusa2 mission, a small lander called MASCOT (Grimm et al., 2013) was carried by the spacecraft to be launched into the asteroid's surface. Similar studies were conducted for OSIRIS-REx, as described by Berry et al. (2013), where their Touch-and-Go (TaG) technology is included in the contact dynamics analysis to understand the interactions with the surface of Bennu. More recently, an interesting method for a three-legged robotic lander was proposed by Caruso et al. (2020). In there, the authors make use of MSC Adams (see Giesbers (2012)) as dynamics engine provider for their analyses, which is a proprietary software used as a multibody dynamics simulation solution.

Currently, there are a number of tools that offer physics functionalities, typically aimed at the generation of animations and renderings. Some of them are commercialised under private licenses and developed by the industry and others have open-use licenses and are built by the community. In an attempt to make use of the great amount of resources available, this work screened some of these tools and considered their application for landing analysis.

Starting from rendering tools developed specifically for space-related purposes, arguably, the most renown and significant contributions are the Planet and Asteroid Natural scene Generation Utility (PANGU) and SurRender. PANGU (Martin et al., 2019) is a tool used to render images of celestial bodies from a space mission perspective. It offers instrument modelling, image rendering, or CAD support and management and is commercialised by STAR-Dundee (Mills et al., 2010). It has been developed by the University of Dundee for the European Space Agency (ESA) and is being used on several ESA studies and development projects aimed at producing precise and robust planetary lander guidance systems. SurRender, which is an in-house tool developed by Airbus, offers a similar set of capabilities described in Brochard et al. (2018). It addresses the ray-tracing challenges in the space environment, including high variability of sizes or specific optical properties, for instance. It has been used in projects for ESA, Centre National D'Etudes Spatiales (CNES), and for Airbus' internal projects.

However, both of these tools lack a contact dynamics engine underneath that could allow for the landing simulations this work aims for. To find those capabilities, one could recur to the video game industry, where physics are a fundamental part of any realistic simulation. Unity is a well-known tool and has been used before integrated into a larger framework dedicated to image rendering , even for space applications, in Craighead et al. (2008) or Wood et al. (2018).

Similarly, Unreal Engine (Sanders, 2016) offers a great alternative to Unity, featuring similar capabilities. According to Dickson et al. (2017), Unreal has a steeper learning curve at the beginning but is capable of generating more professional-looking results with less effort, and gives the users a free-to-use alternative.

While rendering is not a basic requirement for impact simulations, it is a desirable feature that would contribute to the problem understanding by the user, in particular for landing simulations, where visual inspection of the trajectories is a relevant evaluation method. Additionally, and with the scalability of the tool in mind, integrating the software into a wider Guidance, Navigation, and Control (GNC) simulation suite would enable the generation of images for image-based navigation analysis or Image Processing (IP) technique investigations. Some of the options above offer very specialised space-based

image rendering and instrument simulation, and others are designed with their physics engines as a core functionality; but none was deemed to offer a balanced alternative that could do both satisfactory for this project. With that in mind, an effort was made to find a trade-off between both the rendering and the physics worlds that could be easily included in a GNC simulation tool.

Blender (Community, 2018) is an open-use license VFX suite with a strong community support that offers many different features including 3D modelling, UV unwrapping, texturing, raster graphics editing, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, rendering, motion graphics, or compositing, for instance. These functionalities make it very attractive for image-based navigation methods around small bodies, since, given a mesh model of a certain body, datasets of images can be rendered to simulate camera influence in GNC performances or to train, validate, and test IP algorithms, as exploited by Pajusalu et al. (2022), Pugliatti et al. (2021), or Pugliatti et al. (2022). In addition to that, Blender offers a set of dynamic modelling options (called *Force Fields* in the suite) including, wind, magnetic, or harmonic force fields, among many others. Since there is no limitation to the number of force fields in a simulation (other than in terms of processing power), any kind of dynamical environment could be potentially simulated, including the surroundings of a small celestial body.

The contact dynamics engine built within the tool enables rigid-body simulations where collisions and deformations are evaluated, once the defining parameters are established. This engine is based on PyBullet (Coumans & Bai, 2016), and allows for many different settings regarding collision type, enveloping mesh selection, particle interaction, margin definition, or physical properties definition, for instance. On top of that, Blender offers the option to initialise simulations using Python scripts that are executed using the internal distribution that comes with its installation.

All of the above-mentioned characteristics make it very appealing to integrate Blender into a GNC simulation tool to perform landing analyses without going through the effort of having to develop each of the models and modules required.

The ultimate goal of this study, other than the proof-of-concept of the use of Blender as a modelling engine for contact simulations, is the development of a simulation package that can be used to analyse landing sequences onto small celestial bodies and to help designing landing strategies tailored to the different elements involved in the scenario. By performing such analyses, the different design parameters that define a landing scenario (e.g., impact attitude, touchdown velocity, or settling position, for instance) can be optimised with respect to a certain criteria. By post-processing the results obtained from these simulations, safety-maps could be created, to be used as a look-up tables for trajectory and landing feasibility functions within the guidance capabilities of a flight dynamics suite, or even onboard autonomous GNC systems. These maps are envisioned to inform about the different landing design parameters for a certain scenario, acting as a key resource in close-proximity and landing-stage mission design.

Having access to such a resource would increase the robustness of the mission design and would allow for quick adaptation to newly acquired information in later stages of the mission, characterized by limited decision windows or critical turnaround times. The design of disposal strategies could also be aided by this methodology, contributing, thus, to a responsible management of the space environment around small bodies and to a sustainable use of the resources needed to realise such missions.

The rest of the paper is organized as follows: Section 2 introduces and justifies the study-cases and scenarios considered. Section 3 describes the modelling of the elements included in the scenario, such as orbital dynamics models, problem geometry, shape models, or small body morphology, among other parameters. After that, Section 4 explains the methodology used to achieve the objectives discussed above. Section 5 shows the results of the validation and calibration tests performed prior to the analyses, including orbital dynamics and contact physics tests, the two main contributors to the simulated trajectories.

Finally, Section 6 shows the obtained results and poses a discussion about their most important aspects.

2. Study-Case and Context

To give the obtained results a significant context, a mission scenario corresponding to a current mission to an asteroid is used. In particular, a case study in the context of the Asteroid Impact and Deflection Assessment (AIDA) collaboration (Michel et al., 2016) was selected. AIDA is a collaboration between National Aeronautics and Space Administration (NASA) and ESA that encompasses two missions: Double Asteroid Redirection Test (DART) (Cheng et al., 2012) and Hera (Michel et al., 2021), respectively. The former will send an impactor spacecraft to crash on the secondary body of the Didymos binary system, called Dimorphos, allegedly creating an artificial crater on the surface of the secondary body (Rivkin et al., 2021). A few years later, the Hera mission will rendezvous with the system and visit it together with two 6U CubeSats: Juventas (Goldberg et al., 2019) and Milani (Ferrari et al., 2021a). The scope of the latter two is to demonstrate CubeSat technologies in deep-space as well as to augment the scientific outcome of the mission. As part of their disposal strategies, one of the options under consideration is to attempt a landing close to the crater region of Dimorphos. The landing and bouncing off the surface of Dimorphos are part of an experimental phase which follows after the successful conclusion of the nominal scientific objectives of the mission, as described by Ferrari et al. (2021a).

Since the trajectory design for this landing attempt is not the focus of this work, the simulation efforts are constrained to the very last stage of the landing sequence. Thus, the study-case used here is based on an "entry-gate" concept: a very traditional landing technique in which a gate is defined close to the landing spot (Bennett, 1970). The spacecraft trajectory then aims to arrive at such a gate within a certain velocity threshold (both in magnitude and direction) to guarantee a touchdown near the selected region.

For the preliminary analyses in this work and to test the methodology developed, the initial velocity at the entry-gate is set to null, starting from a resting condition relative to Dimorphos. The direction of the descent, thus, is solely decided by the accelerations in the simulation. This effect is referred to as "CubeSat rain". Different grid resolutions are tested, starting from a gate at an altitude of around 10 m, uniformly distributed. The width of the grid is chosen so that it covers the higher diameter craters included in the simulation. The grid shape corresponds to the intersection of the sphere at the altitude mentioned above with a pyramid with a half-angle of 10.0° with its vertex at the centre of the mesh, i.e., a spherical patch. A representative example of this scenario is depicted Figure 1.

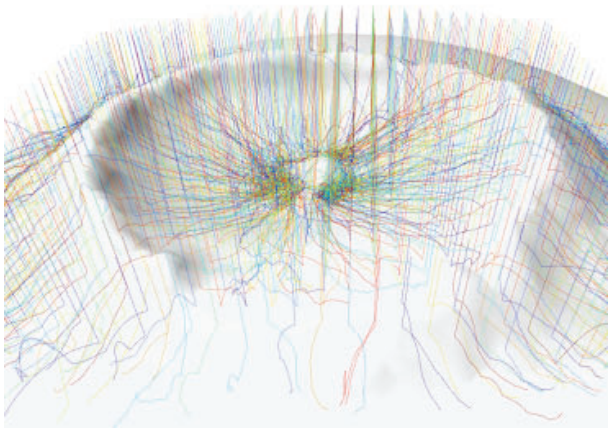


Fig. 1: Example of an entry-gate simulation, where initial states are distributed on a grid at an altitude of 10.0 m above crater model C2, spanning 20.0 m in diameter.Trajectories from an arbitrary entry-gate simulation (CubeSat rain) on a crater.

3. Shape and Dynamical Modelling

The dynamical environment used for this analysis is the Didymos Near-Earth Asteroid (NEA) binary system (although some assumptions are introduced in Section 4). In Naidu et al. (2020), a very thorough description of the system is given, based on delay-Doppler radar imaging of the binary system. There, the observations were used to estimate 3D shape model and spin states for the primary (Didymos) and size and spin state of the secondary (Dimorphos). The aforementioned DART spacecraft will produce a crater on Dimorphos, whose properties are unknown. Studies have been performed in this direction where possible crater morphologies are suggested (Ormö et al., 2020), but the exact shape of the body is

Table 1: Didymos system physical parameters, from Naidu et al. (2020) and Wiegert (2020).

Parameter	**Value**
Aphelion	2.2755 au
Perihelion	1.0141 au
Heliocentric semi-major axis	1.6444 au
Eccentricity	0.383 88
Sidereal spin period	2.26 years
Orbital heliocentric inclination	3.4078°
Known satellites	1
Rotational period (primary)	2.26 h
Distance between primary and moonlet	1.18 km
Orbital period of moonlet	11.92 h (tidally locked)
Diameter of primary	780 m
Diameter of moonlet	160 m
System mass	5.278×10^{11} kg
Density	1.7 g cm^{-3}
Absolute magnitude	18.16
Last close approach to Earth (Nov 2003)	7.18×10^{6} km
Close approach for DART mission (Oct 2022)	11.0×10^{6} km

still unknown and will only be resolved once the DART spacecraft visits the system in October 2022. The most relevant characteristics of the system are gathered in Table 1.

Currently, Dimorphos is estimated to orbit about the barycenter of the system in a circular orbit of 1.18 km radius and an orbital period of 11.92 h. The orbit is coplanar with the equatorial plane of Didymos. Similarly to the Earth-Moon system, Dimorphos is tidally locked, always exhibiting the same face to the primary. The impact of the DART spacecraft on the secondary body will likely change the tidal lock and the rotation of the secondary around the primary. For the scope of this study, that plays no role in the modelling of the system or the results obtained, since illumination conditions are out of the parameters considered and a body-fixed frame is adopted for the simulations. The ellipsoidal shape model for Dimorphos is taken from Ferrari et al. (2021a) and has the dimensions recollected in Table 2.

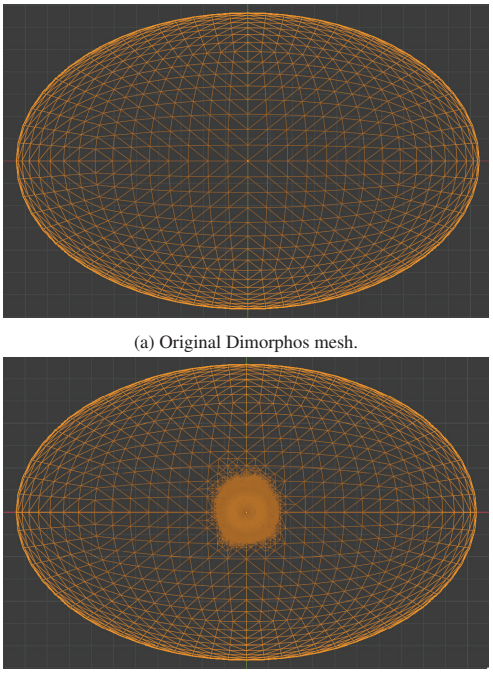
Table 2: Ellipsoidal shape model used as a reference for Dimorphos (Ferrari et al., 2021b).

Axis	**Value**
a	103.0 m
b	79.0 m
c	66.0 m

The key characteristics of the crater, such as shape, size, and depth, have large uncertainties associated with them. For the purpose of this work, a number of craters are created to account for the different possibilities. Based on Marchi et al. (2015), a *depth-to-width* aspect ratio of around 0.2 has been used as a reference to model these craters, which are integrated on the reference shape model shown in Figure 2a using a proportional deformation tool in Blender. As explained in Guevarra (2020), proportional editing affects the nearby geometry of the modified mesh vertex, which makes it a very useful resource when a smooth deformation of the surface of a dense mesh is needed. The mesh's resolution is increased over the crater region to make sure that the impacts registered in the simulations accounted for the very irregular topography of some of the crater models, as it can be observed looking at Figure 2b.

Eleven different crater types are designed using diameter and depth as size-defining parameters and by assuming different possible morphologies for the crater. The five different crater morphologies used are:

- C0: Obtained as a smooth proportional deformation on the mesh of Dimorphos.
- C1: Obtained by merging the Arizona crater (Barringer, 1914).
- C2: Obtained by merging a peak crater, based on C1 with



(a) Original Dimorphos mesh.

(b) Modified Dimorphos mesh.

Fig. 2: Grid of the shape model used for Dimorphos from X-Z plane, crater is placed centred on the Y-axis of the body-fixed reference system. Top: reference shape model. Bottom: shape model with the 20 m C0 crater merged with the original mesh.

the inclusion of a small elevation in the centre of the crater, to include one of the most typical crater morphologies (Allen, 1975).

- C3: Obtained by merging a soft-crater with 1.0 m depth.
- C4: Obtained by merging a soft-crater with 1.5 m depth.

Each shape of the C0, C1, and C2 craters features diameters of roughly 5, 10, and 20 m. C3 and C4 craters both span 10 m in diameter. Such crater size range is consistent with latest simulation-based estimates for the DART mission (Rivkin et al., 2021). Figure 3 shows renderings of some of the modelled craters as well as the mesh used for the CubeSat , which corresponds to a 6U CubeSat, whose dimensions are detailed in Figure 4.

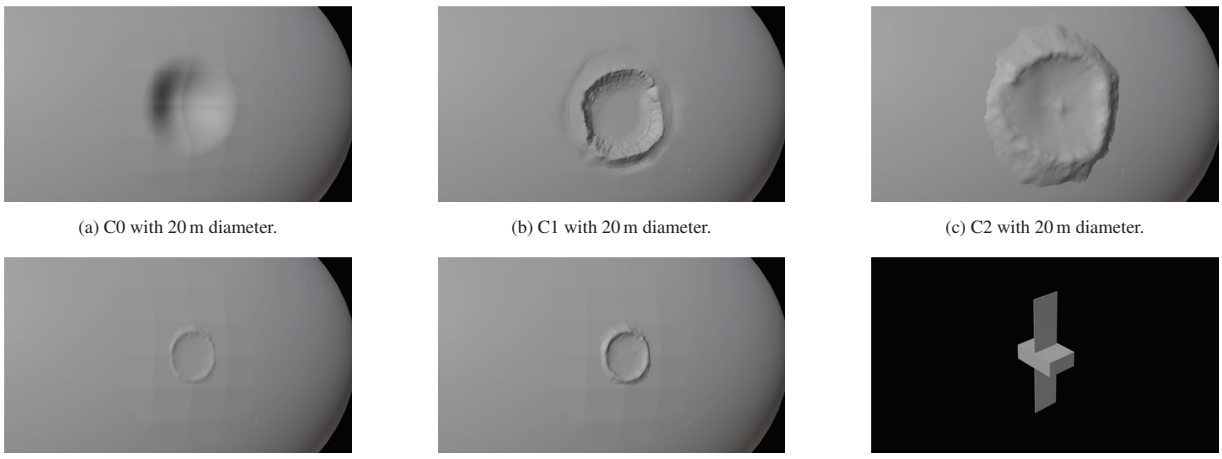
The orbital and suborbital dynamics of the problem is one of the two main drivers for the landing simulations performed in this work, together with the contact dynamics. Orbital perturbations around asteroids are primarily driven by the gravity potentials of the bodies involved and by the Solar Radiation Pressure (SRP) exerted by the Sun. These are the only forces accounted for in the dynamics model used here. To compute these accelerations during the simulation, Astrodynamics Simulator (AstroSim) is used. AstroSim is a GNC simulator that includes an orbital propagator with the relevant orbital accelerations for small body environments (Peñarroya et al., 2022), among other functionalities. All of the algorithms and frameworks developed in this work are included in the AstroSim suite as a module called *astroContact*, which can be used both integrated with a larger GNC simulation or completely detached from the AstroSim environment, since there are no strict internal dependencies.

4. Methodology

Once the different modelling elements have been introduced, the next step is to develop a methodology to produce analyses for landing trajectories systematically. The parametric nature of this study (similarly to what was presented by Van Wal et al. (2017)), makes Monte Carlo approaches desirable, since the different combinations of parameters involved can be analysed separately to understand the effects that each generates on the landing trajectories obtained. The idea is to use the above-mentioned landing gates as a grid to set up the initial positions and velocities of the CubeSat prior to the impact (the latter is always zero in this case). By placing the initial conditions of the trajectory at each of these grid points, it would be possible to create a dense cloud of initial states from which to simulate the descent trajectories and their corresponding rebounds until they reach a settling point, as it was previously shown in Figure 1.

This simulation setup requires a number of steps:

1. set up the simulation scenario,
2. fix initial state for the CubeSat,
3. propagate its trajectory until it reaches the gate,
4. switch to Blender to propagate the trajectory from the final state provided by AstroSim until touch-down is achieved,
5. use Blender's internal dynamics to simulate the rebounds,
6. log the propagated trajectory, and



(a) C0 with 20 m diameter. (b) C1 with 20 m diameter. (c) C2 with 20 m diameter.

(d) C3 with 10 m diameter. (e) C4 with 10 m diameter. (f) Model used for the 6U CubeSat.

Fig. 3: Examples of the different models used for the craters and the CubeSat.

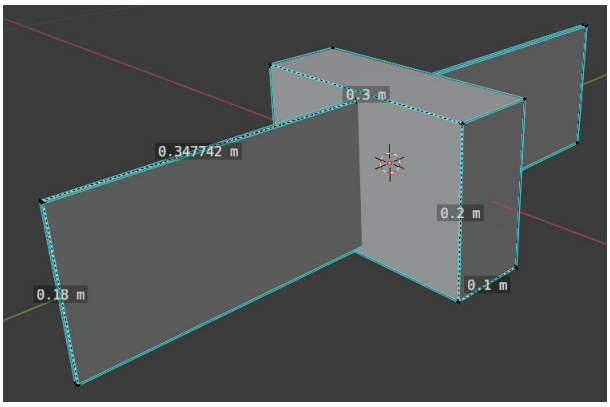


Fig. 4: CubeSat dimensions used in the simulations.

7. process the obtained results and construct a safety map.

Steps 1-3 are carried out using AstroSim, which propagates a certain initial trajectory until it reaches the low-gate. Step 4 requires the final state provided by AstroSim to be fed into Blender's internal dynamics simulation. To do that, and making use of the feature that Blender offers to launch simulation scenarios from Python scripts, *astroContact* launches a Blender instance with the scenario parameters that correspond to step 3, i.e., after the initial propagation in AstroSim.

Doing that is not as trivial as it may seem, and some considerations need to be made. To begin with, even if the low-gate is set at a very low altitude with respect to the surface of Dimorphos, the fact that the spacecraft keeps moving while descending would mean that the different accelerations would also change, so it is not ideal to set a constant value for the gravity vector in Blender and let the simulation run. Hence, the perturbations that need to be included must be analysed. As it was stated before, gravitational potential and SRP are the two most significant dynamical effects during ballistic flight, as discussed in Ferrari et al. (2021b).

To analyse the environment, the work from Ferrari et al. (2021b) is considered, where a perturbation analysis was performed in the vicinities of Dimorphos. In such an analysis, the gravity of the primary, the secondary, the Sun, and the SRP accelerations are considered. Since the distance from the Didymos binary system to the Sun varies widely along its orbit due to its high eccentricity (see Table 1), the SRP is considered as a range from the closest to the furthest positions of the binary systems with respect to the Sun. Other perturbations that could be considered in this scenario include Thermal Radiation Pressure (TRP) or Albedo Pressure (AP), which for Bennu's environment are, respectively, 3 and 2 orders of magnitude below SRP, according to Chesley et al. (2020). Because of such a low weak influence, and given the short time frame of the simulations, these perturbations are not included in the analyses shown in this work.

Because the cases to be analysed for this analysis start from around 10.0 m altitude above Dimorphos' surface, the results shown in Figure 4 from Ferrari et al. (2021b) indicate that the central gravity acceleration of the secondary body is the main driver of the accelerations undergone by the CubeSat. Thus,

two assumptions are posed regarding orbital dynamics.

The CubeSat is assumed to be under the influence of Dimorphos' point-mass gravity only. Gravitational effects of the primary and the SRP are neglected, due to the short time of flight before touch down, and the vicinity to Dimorphos' surface at initial simulation time. This assumption is consistent with Didymos system's acceleration field reported in Ferrari et al. (2021b), and suitable to support the goals of this work.

Thus, only the gravitational potential of the secondary is considered. However, there is still a choice to be made regarding how this potential would be implemented. Blender offers two ways to add dynamic effects to a certain simulation: by setting a gravity vector inertially or to include a property to the bodies involved known as *Force Field*. The former requires the use of a built-in functionality in Blender called *keyframes*, which are way-points that fix any simulation parameter in a temporal scale forcing them to take certain value at a certain frame. The latter, instead, leaves the responsibility of the computation of these values to Blender's internal physics engine and numerical integrator.

To trade both methods off, an assessment is performed based on the following criteria:

- modelling,
- computational effort,
- implementation,
- flexibility, and
- numerical error.

Fixing acceleration keyframes means that, because these acceleration values, which are computed using AstroSim, depend on the position of the spacecraft at each frame, they need to be updated accordingly as the simulation goes. This means a constant back-and-forth communication between Blender and AstroSim during the propagation, which makes for a huge computational burden.

In order to keep Blender's simulation script as detached as possible from the GNC simulator, an interpolator for the acceleration values is created using an homogeneous spherical distribution (Fibonacci distribution (González, 2009)) around the body. Figure 5 shows an example of how the points would be distributed for a setup with three spheres with radii 100.0 m, 200.0 m, and 300.0 m, containing 500, 600, and 700 each, respectively. It is important to make sure that the radius and the number of points are aligned with the desired resolution in the sphere surface. Radial resolution can be controlled via separation between consecutive spheres. With a grid resolution of 10 m, concentric spheres are superimposed to guarantee that the simulation is not outside the interpolated region and that the distance from any position coordinate to its closest sphere does not surpass the chosen resolution. The values for the acceleration are then obtained by querying values to the interpolator inputting a three-dimensional position array. A similar approach, although using the mesh's vertices as points for the interpolation, is used in Van Wal et al. (2020), where a voxelization of a polyhedron model is pre-computed offline and stored for its online interpolation. There, the selected scheme computed the perturbation of the chosen polyhedron model with respect to a central gravity model, since it showed to be the most memory

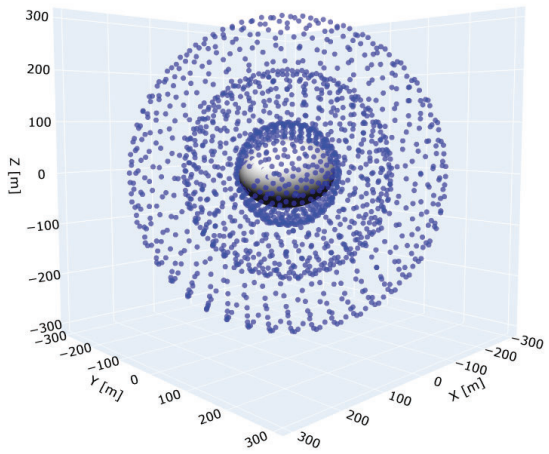


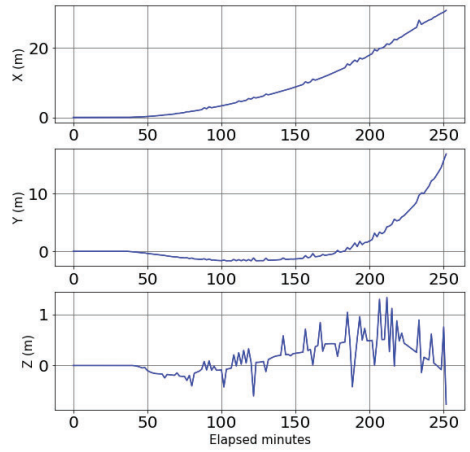
Fig. 5: Example of Fibonacci spheres around Dimorphos. Three spheres with radii of 100.0 m, 200.0 m, and 300.0 m, containing 500, 600, and 700 each, respectively, are laid over the mesh of the asteroid to conform the interpolator cloud.

efficient.

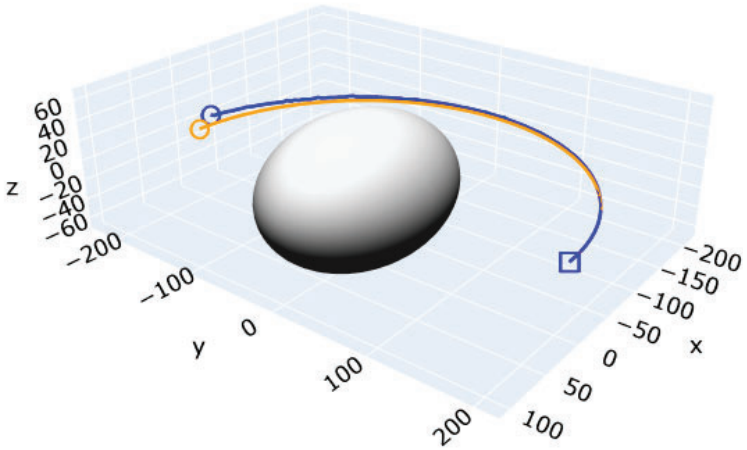
The benefit of such interpolators is that their points can contain information about any kind of acceleration AstroSim could compute, including different types of gravity potentials (point-mass, spherical harmonics, or polyhedron models, for instance). To improve the fidelity of the interpolator to the actual model, the spatial resolution of the cloud of points in the interpolated gravity field can be tailored to the dynamical environment and the right interpolation scheme can be chosen to contribute to mitigating approximation errors in the acceleration values obtained.

However, one drawback they present is their heavy computational weight. Depending on the resolution of this cloud, the computation of the points is typically on the order of tens of minutes. On top of that, if any of the accelerations are time-varying (as it is the case for SRP or Third-Body Gravity (TBG)), the cloud needs to have an extra dimension accounting for time, making the process even slower and heavier. Nevertheless, this does not burden the simulation time since, once the interpolation cloud is created, there is no need to recompute those values for each simulation, i.e., the cloud is stored as a variable that is then loaded into the simulation.

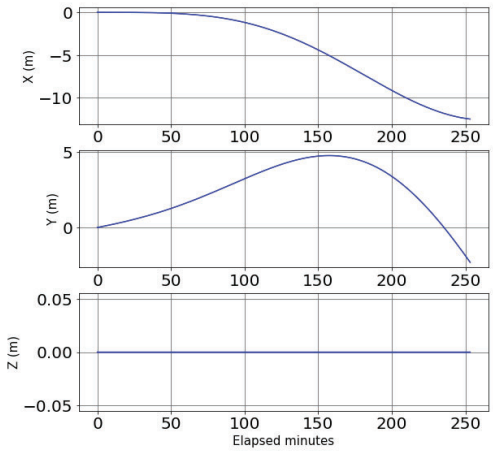
Even if clouds can be computed and stored beforehand, the sample trajectories take several minutes to simulate. This is due to the way Blender interprets the dynamics, which requires the simulation to *bake* (computer-graphics term referring to, for example, pre-computing dynamics in a simulation) all of the simulation frames every time a new acceleration keyframe is



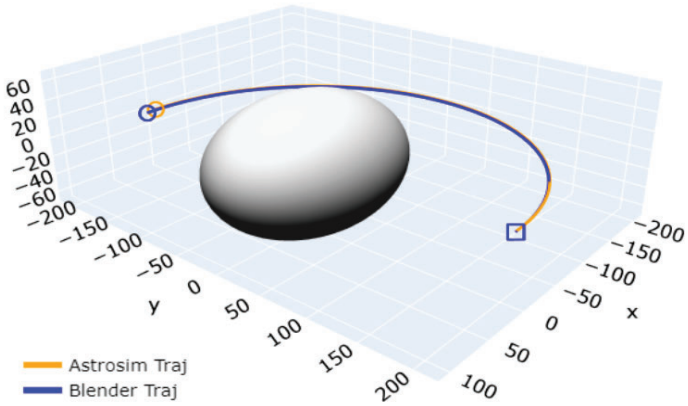
(a) Position error for keyframes.



(b) Trajectory comparison for keyframes.



(c) Position error for force fields.



(d) Trajectory comparison for forcefields.

Fig. 6: Trajectories propagated in AstroSim and in Blender around Dimorphos at 2.0 radii, in body-fixed reference frame.

added, i.e., each time-step. This not only increases the computational time of the analyses, but also limits the maximum timespan a certain analysis can be run for, since each additional frame scales the computational effort quadratically. For example, for a simulation spanning 500 time-steps, the dynamics of the simulation need to be updated at each step for each of the 500 time-steps (or frames), thus, making for a total of 250 000 frames to be baked. Consider that, for the usual time-step sizes used for landing simulations, 500 frames would correspond to around 500 s, which is already a very limiting timespan for landing simulations.

On top of that, the frame-by-frame addition of keyframes to the simulation makes the acceleration values interpreted by Blender become discontinuous, even if the interpolation schemes for the keyframes are set properly. This fact results in the behaviour shown in Figure 6a. There, it can be observed how, in spite of the fact that errors are not growing exceedingly large in time, their behaviour is very much influenced by the interpolation error create by the nature of this method.

Force fields, on the other hand, provide an alternative to keyframes that only needs to be set up at the beginning of the simulation, saving a great deal of the computational effort since dynamics need to be baked only once, resulting in simulations that take few seconds to finish. In this case, accelerations are fully computed within Blender, using the internal dynamics engine. The perturbations to be included in the simulation need to be modelled using the different force field options provided in Blender , adjusting the input parameters so that the produced output has physical sense. A more detailed description on how this is achieved is given in Section 5.

Because of that, and despite Blender being a very flexible tool, the modelling of a richer perturbational environment (including SRP or TBG, for instance) becomes more challenging than with the keyframe method described above, where they can be computed using other tools and techniques and included into the cloud interpolator. This does not mean, however, that more complex perturbations can not be modelled in Blender. For instance, force fields include a tunable parameter called *shape*, which is used to choose the way in which the direction of the effector force (in this case gravity) is calculated. Available options include:

- point: field originates from the centre of the object;
- line: field originates from the local Z axis of the object;
- plane: field originates from the local XY plane of the object;
- surface: field originates from the surface of the object; and
- every point: field originates from all vertices of the object.

From that list, the option *every point* can be chosen to model what is known as polyhedron models or Mass Concentrations (Mascon)s, which are explained in detail in Werner & Scheeres (1997) and Muller & Sjogren (1968), respectively.

Nevertheless, and as discussed in previous sections based on the work from Ferrari et al. (2021a) and Van Wal et al. (2020), once contact begins, the rest of the accelerations involved in the simulation become much less relevant, in relation to the effect impacts have on the trajectory. Thus, the force model implemented for the segment of the simulation computed by Blender for the results shown in this work represents a point-mass gravity potential . Note that this setup does not limit the orbital and sub-orbital regimes of the simulations, which still are simulated in AstroSim, where other accelerations of interest can be included to bring simulation results closer to reality.

On the other hand, time-dependent accelerations are much better handled with this approach , even though, as previously discussed, they might not play a relevant role, given the short simulation timeframe.

In terms of accuracy, the results for the obtained trajectories can be observed in subfigures 6c and 6d, where errors are much smoother than those obtained with the keyframe method. It is also worth noting how the error introduced by the interpolator used in the keyframe method is larger than that introduced by assuming point-mass gravitational model, as it is done in the force field method. More detailed figures for position error evolution are shown in Figure 10 and discussed in Section 5.

Table 3: Trade space for gravity vector keyframes and force fields as Blender's orbital dynamics source. Bolded fields are preferred criteria.

Criteria	Gravity keyframes	Force fields
Modelling	**Validated perturbations**	Blender internal dynamics
Comp. effort	Very high	**Low**
Implementation	Complex	**Simple**
Flexibility	**Any perturbation**	Only Blender perturbations
Numerical error	High	**Low**

After testing both options, the force field approach is chosen because of its much lower computational effort, its lower implementation complexity, and the higher accuracy that is observed during testing. Table 3 summarises the pros and cons of both approaches.

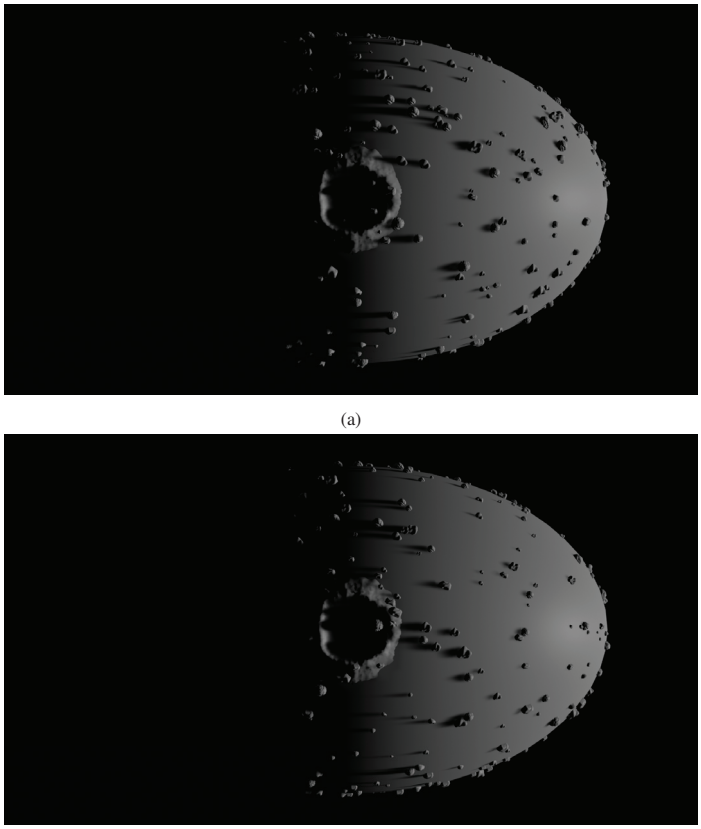
The trade-off criteria where force fields present worse results than the keyframe approach are not as relevant for the decision. The accuracy of Blender's internal force models have been proven to be satisfactory (as it will be shown in Section 5) and, when the numerical noise is taken into account, they actually end up providing more accurate results. The flexibility regarding, for instance, the implementation of different perturbations is not important for this work, considering the assumptions made in Section 3.

Once the dynamics have been set up, position keyframes (not to be mistaken with acceleration keyframes) are used to set up the initial position and velocity of the CubeSat at the beginning of the simulation, until the integration is released to Blender's internal dynamics.

The other relevant contributor to the simulation is the contact dynamics engine. As it has been previously justified, all the responsibility of this is given to Blender, which, in turn, uses Bullet to compute the corresponding dynamics. Thus, rigid-object physical properties are also set in Blender, including bounciness coefficients (see Section 5) and frictional properties of the bodies involved in the simulation.

Blender offers a great variety of parameters to configure a simulation scenario with rigid-body objects. In Van Wal et al. (2020), the authors perform sensitivity analysis with respect to several design parameters, such as, the shape of the object, the CoR, and the coefficient of friction. The different analyses performed there, extract their conclusions from two main results: settling time and surface dispersion. The former refers to the time it takes for an impactor to settle down stationary on the surface of the body starting from touchdown. The latter helps understanding how large the dispersion of the settling coordinates can be with respect to the initial touchdown point. In Van Wal et al. (2017), the authors also study the relevance of rock distributions on the surface of the body and the effect they have on the final landing trajectory. Tardivel et al. (2014b) already concluded that the presence of rocks on the surface of a body can not be neglected for landing simulation purposes. However, in this work, the focus of the simulation lies on the presence of craters and their morphology. Additionally, in Allen (1975), it is shown how rocks are not so often present in craters, due to the latter being generated by collisions, which clear the area around the impact point.

The software developed allows to configure the simulation so that the effects of the aforementioned parameters can be studied. Numerical values are straight-forward to set, since they only need to be specified and transmitted to Blender. The feature generation is a more complex process and a pipeline has been developed in AstroSim specifically to address this problem. This pipeline, originally generated to create databases for Convolutional Neural Network (CNN)-based IP algorithms, is able to distribute population of features on the surface of a given mesh, using Blender as well. The user needs to define the features that need to be included in the population, such as rocks or craters, for instance; select the number of features to be distributed, their size, which can also be specified as a normal distribution with mean size and standard deviation. Thus, a population of features can be stochastically and programmatically generated. Figure 7 shows two randomly generated distributions for the mesh model with C2 with 20.0 m diameter.



(a)

(b)

Fig. 7: Arbitrary rendered images for Dimorphos with crater model C2 generated using AstroSim. Notice how features (craters and boulders) have been randomly distributed around the surface of the body.

As mentioned before, Blender offers a very wide variety of adjustable parameters for contact dynamics simulations. These parameters, on which the software developed is focused (but not limited to), include:

- CoR for the object and the body,
- coefficient of friction for the object and the body,
- mesh geometry for object and body (including features such as craters or rocks),
- collision margin and shape,
- spacecraft mass and inertia,
- linear and angular damping,
- simulation time-step and maximum timespan, and
- initial state and velocity (linear and angular).

For the simulations in this work, damping values are always set to null, the collision margin is adjusted to 1.0 cm, and the shape of the collision, which is the enveloping volume that is used to compute contacts, is chosen to be the mesh itself in order to obtain more accurate impact simulations. Other collision shapes, which is the geometry that is used to compute the contact points, available in Blender include convex hulls, spheres, or cubes, for instance.

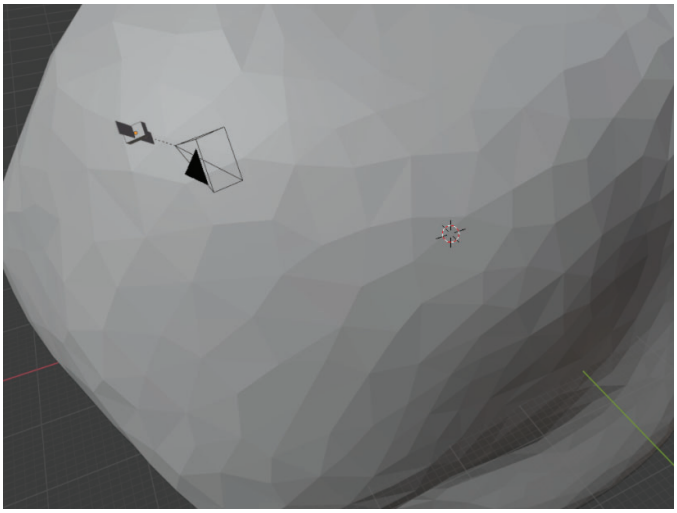


Fig. 8: Screenshot of a simulation in Blender to run a landing trajectory and analyse the conditions at impact using Blender's contact dynamics engine.

The different coefficients, the mesh geometry (including its features), the initial state, and the simulation time-step are part of a more detailed analysis that will be shown in Section 6.

Finally the trajectory and the rebounds are simulated in Blender, i.e., executing Step 5 in the above algorithmic process.

When the CubeSat settles down, the position-and-orientation data are saved to a file, completing step 6. The settling criteria is studied and, at the time of writing, a combination of linear and rotational kinetic energy law is designed to stop the simulation.

Step 7 takes the results and creates visualizations of interest as the ones shown in Section 6. Figure 8 shows a screenshot of a simulation environment in Blender's Graphical User Interface (GUI).

5. Validation and Calibration

Blender is originally designed as a VFX and computer-graphics tool. Thus, the parameters included as properties are not always based on physics. For instance, the CoR (which was

one of the key parameters highlighted by Van Wal et al. (2017)) is referred to as Coefficient of Bounciness (CoB), which may or may not be the same parameter that is usually applied in contact dynamics formulation. The way in which such parameters are utilised in Blender is not always transparent or thoroughly explained in the documentation. To achieve a proper understanding of the physical meaning of such parameters, some validation and calibration tests are performed.

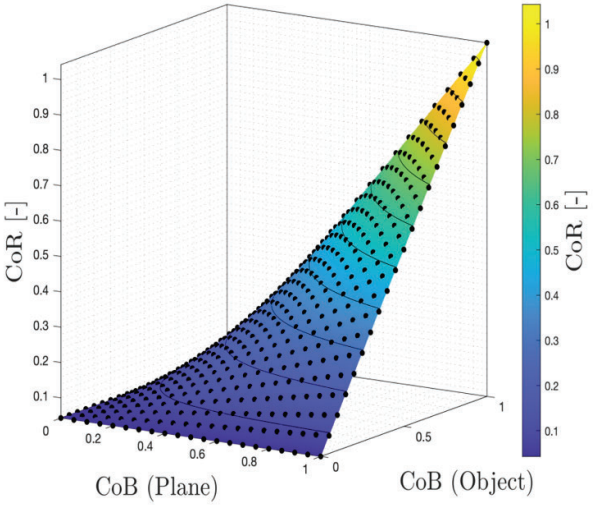
The first test is meant to calibrate the design parameters in Blender that refer to the rigid-body properties (in particular CoR) of the simulated objects and involves a plane and an sphere-like polygon, which is dropped from a fixed height. By changing the CoB of the two objects, the CoR can be computed as the fraction between the magnitude of the velocity after and before the impact. From this simple test it is possible to determine that the CoR is the product of the bodies' CoBs, as it is shown in Figure 9. This experiments shows that, if the CoB of the plane is fixed to 1.0, then the $CoR = CoB_{obj}$.

Other than rigid-body dynamics parameters, it is necessary to validate the implementation of the orbital dynamics as explained in Section 4, i.e., using force fields as attractors to simulate gravitational potential of Dimorphos. In these tests, the ground-truth provided by AstroSim and the trajectories simulated in Blender are compared.

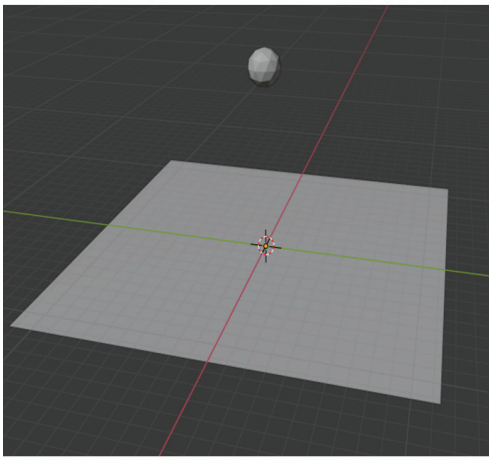
This becomes more of a calibration task, since, as mentioned before, most of the parameters that Blender uses to set up simulations are not strictly physical terms. In this particular case, force fields require a value of *strength*, which, would typically correspond to the gravitational parameter, μ, in astrodynamics. To perform this calibration, and after inspecting Blender's source code , the *strength* parameter in Blender is found to correspond to:

$$F = \chi \cdot \mu \cdot m_{sc} \cdot \Delta t_{frame}^2 \tag{1}$$

where μ is the gravitational parameter of the body exerting the force field, m_{sc} is the mass of the Spacecraft (S/C), Δt_{frame} is the time-step simulated per frame, and χ is a constant parameter that needs to be tuned for each gravitational attractor



(a) Correlation between CoR and CoB.

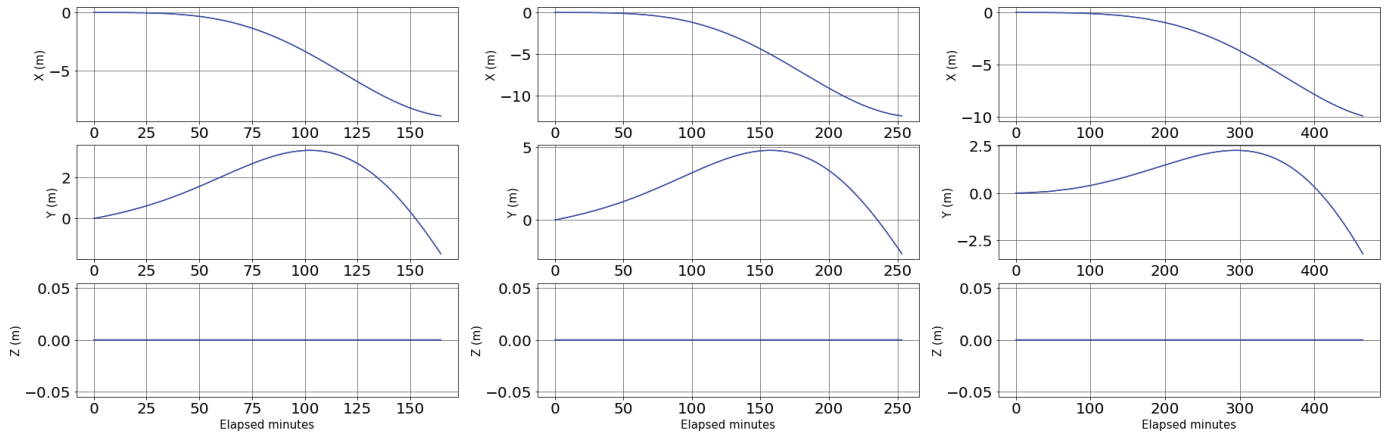


(b) 3D representation of the objects used for calibration, in Blender's GUI

Fig. 9: CoR-CoB validation test in Blender (Peñarroya et al., 2021).

in Blender, acting as a calibration parameter to adapt Blender's internal units to International System (IS)'s units. For the study-case and simulations shown in this work, its value corresponds to 138 922.69 $s^2\,m^{-2}$. The resulting value, F, represents a force and should be expressed in force units (in line with the units of the elements provided in equation 1).

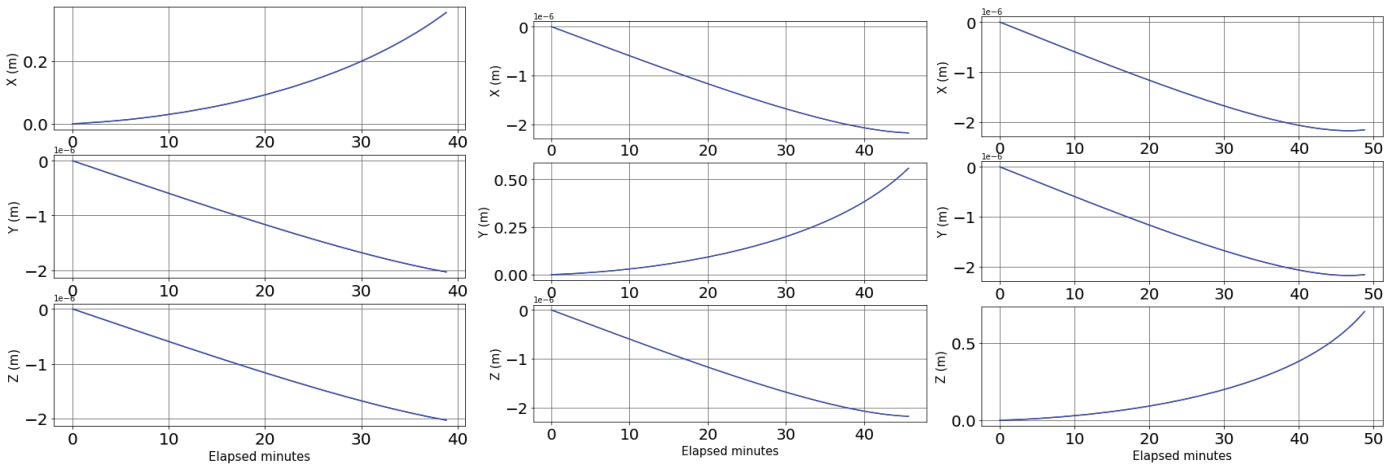
Figure 10 shows the validation results for three orbits around Dimorphos with semi-major axis of 1.5 radii, 2 radii, and 3 radii, in plots 10a, 10b, and 10c, respectively, where radii refers to the reference radius for Dimorphos, corresponding to a in Table 2, i.e., 103.0 m. There, it can be observed that, the closer to the body the orbit is, the faster the dynamics and the higher the errors (for equal propagation time); getting up to 3 m



(a) Trajectory at 1.5 radii distance.

(b) Trajectory at 2 radii distance.

(c) Trajectory at 3 radii distance.



(d) Vertical drop along X-axis.

(e) Vertical drop along Y-axis.

(f) Vertical drop along Z-axis.

Fig. 10: Position error for the trajectory validation tests.

after 100 min of propagation for the closest cases.

The behaviour for the position components in the upper row of Figure 10 is similar for all three cases, only modified by the magnitude of the error at the end of the timespan of the propagation, which is fixed to half an orbital period, to obtain equivalent dynamical behaviours. Errors are always under 15 m, even for the longest timespans that go over 400 min. Considering that the timespans to be typically simulated for touchdown simulations range from seconds to few minutes, the errors observed in the validation tests are deemed satisfactory.

However, there is one issue still to be analysed. As mentioned above, the closer the S/C gets to the asteroid, the more intense the gravitational pull becomes, and, with it, the larger the propagation error. Since the intention of *astroContact* is to simulate landing sequences, i.e., trajectories that reach the body surface, an extra set of tests are performed, where the trajectories are vertical drops onto the surface of the asteroid. For such tests, the error is plotted until the moment of impact, in which AstroSim stops the propagation (flagging a collision event), while Blender keeps simulating until the settling criteria, corresponding to linear velocities under 0.001 $\mathrm{m\,s^{-1}}$ and rotational rates under 0.001 deg/s, is met. The results shown in the lower row of Figure 10 are obtained by comparing both trajectories only until the moment of impact.

Thus, a drop is simulated along each of the body axes of Dimorphos. From Table 2, the reference radius can be estimated as 103.0 m (the longest semi-major axis). The drop-trajectories depart from an altitude of 50.0 m from that reference radius, i.e., a total of 153.0 m from the centre of the body. Notice how, due to the non-spherical shape of the body, the simulation times-

pans, which are governed by the touchdown time, vary slightly from 38 min to 49 min. For reference, the altitude used for the results that will be shown in Section 6 is 10.0 m, so simulation timespans will go even lower, supporting the argument posed above.

The errors for these drops increase only along the velocity direction and are always in the order of cm. With that, it can be concluded that, for the nature of the simulations intended for this work, the orbital dynamics can be considered validated.

Finally, the the contact-dynamics engined needs to be validated. The aim of this batch of tests is not to validate the underlying engine (PyBullet, as mentioned in Section 1), but to validate whether the parameters used to set up the simulation maintain rigor from a scientific perspective.

Following energy conservation principles, as the descent takes place, the energy of the S/C shall remain constant, distributed among linear kinetic, rotational, and potential energies. Then, at the moment of the impact, a momentum transfer takes place between the CubeSat and the colliding object, i.e., Dimorphos. In this impact, a fraction of the energy of the CubeSat is transferred to the asteroid as kinetic energy, another fraction is dissipated if the colliding objects are not perfectly rigid and the impact is not perfectly elastic, and the remaining fraction remains to the CubeSat.

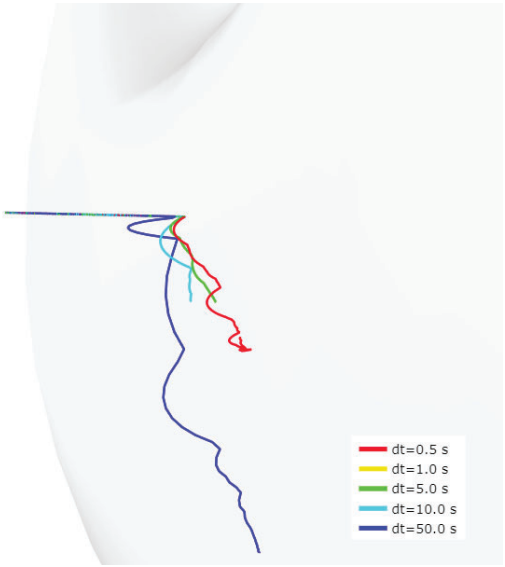


Fig. 11: Sample drop-point simulated with various Δt values.

An interesting fact is found while running the simulations for this test. As for any kind of numerical integration task, the chosen time-step plays a fundamental role in the accuracy of the results obtained. Same holds true for contact dynamics. Figure 11 shows how varying the time-step used for the simulations, makes the trajectories diverge due to numerical propagation errors.

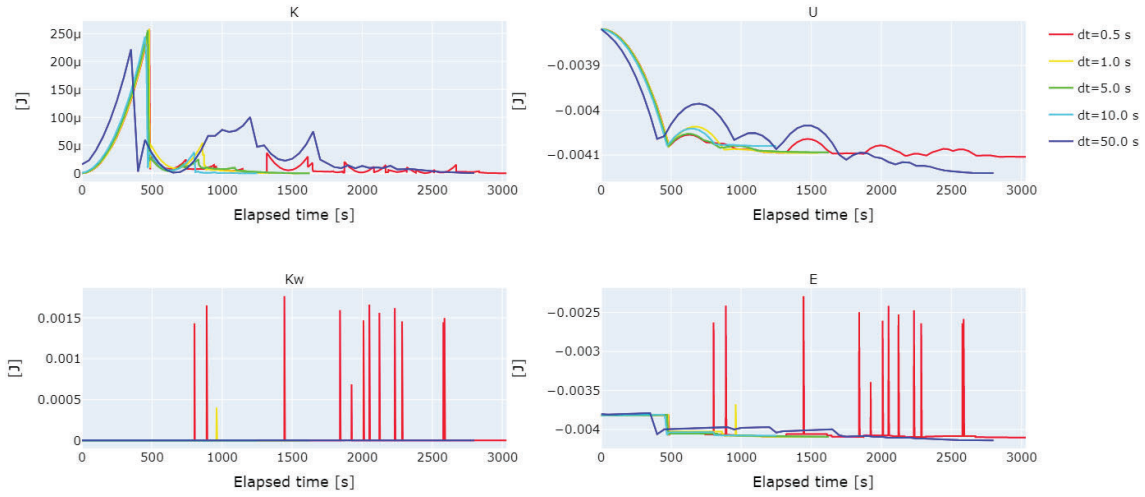
When checking the energy summary of the CubeSat, for the different time-steps used, singularities appear for certain curves. Figure 12a shows how the trajectory corresponding to $\Delta t = 0.5$ s presents peaks that are well out of the ranges of energy observed for the rest of the components. In that figure, U stands for potential energy, K for kinetic linear energy, Kw for kinetic rotational energy, and E for mechanical energy (the sum of all three former ones). The inertia matrix used to compute Kw is a diagonal matrix obtained by internal communication with the authors of (Ferrari et al., 2021a) and is represented by:

$I = 10^{-4}[0.4782, 0.2566, 0.2566]$ kg m^{-2}.

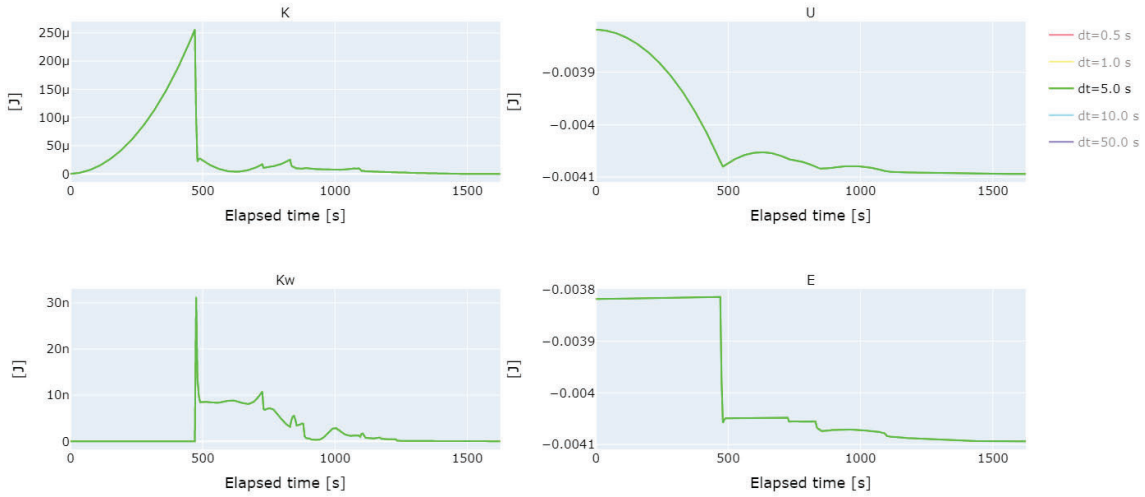
These peaks are due to collision tolerances and margins not being in line with the simulation time-steps used. When two rigid-body meshes intersect each other, the dynamics would generate a collision and bounce effect that would cause the two bodies to reject each other, affecting their trajectories and introducing a rotational momentum to each of them. However, if the time-step is too small, then next dynamical evaluation might not be far enough for the intersection to have disappeared and then numerical errors appear due to two consecutive steps evaluations detecting contact. This is the case for Δt of 0.5 s and 1.0 s in Figure 12a.

Thus, it becomes clear that the selection of the time-step to be used in the simulations is crucial to the quality of the output generated by *astroContact*. Two criteria need to be followed: avoid excessively long time-steps to prevent inaccuracies in the orbital dynamics (trajectory), and avoid excessively short time-steps to prevent numerical errors in the contact dynamics (impacts and bounces).

Filtering the results using these two criteria, rules the two smaller time-steps out due to numerical errors and the two



(a) Energy summary for all the Δt values tried.



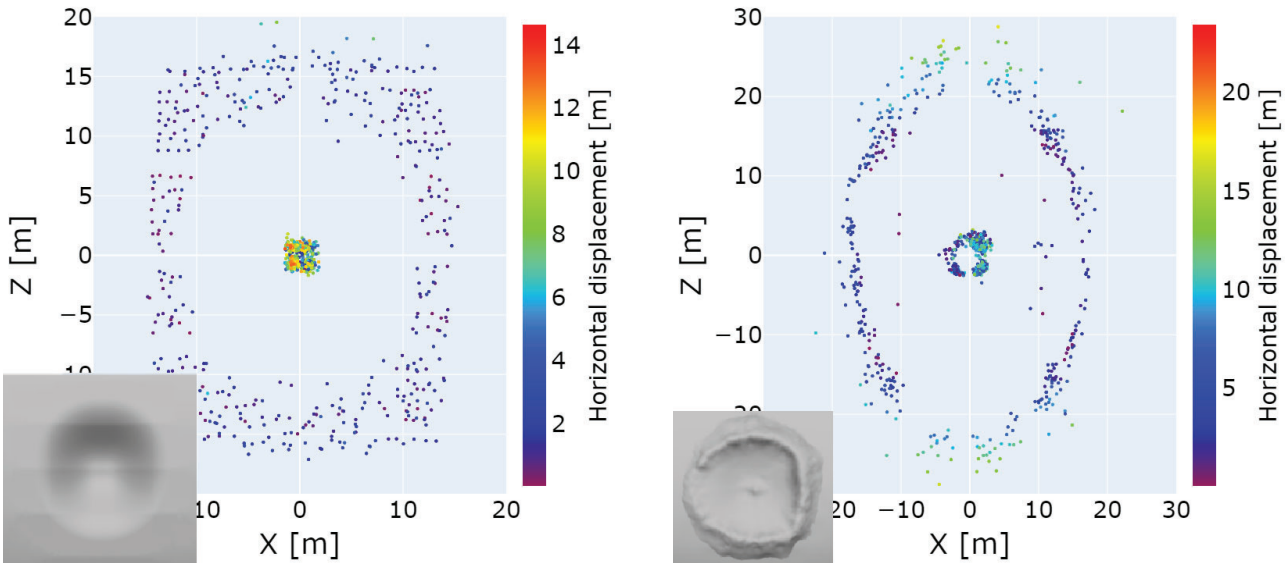
(b) Energy summary for the selected Δt value.

Fig. 12: Energy summaries for a set of candidates Δt values tried before and after filtering. Lower values are discarded due to numerical errors while higher values are discarded due to trajectory divergences.

longer ones due to trajectory divergence (as shown in Figure 11); resulting in the energy summary observed in Figure 12b. There, it can be seen how the sample with a Δt of 5.0 s yields satisfactory results.

Finally, after studying the effect that the time-step has on the simulations, the energy plots can be analysed to check whether energy is in fact conserved. To do that, this work will only focus on the energy states of the CubeSat, i.e., the energy transferred from the CubeSat to the asteroid will not be analysed. As expected, the kinetic energy levels increase as the spacecraft gains velocity and drop abruptly at touchdown. This results in the first drop in mechanical energy that can be observed around the 500.0 s mark in Figure 12b. Then, a parabolic curve can be observed with inverted aspects in kinetic and potential energy, respectively. This can be attributed to the first bounce after touchdown. After that, another impact happens, indicated by the kinetic rotational energy, around the 700.0 s mark (this time much smaller). The probe then keeps bouncing and rolling until it settles onto the surface of Dimorphos.

The results shown in this section conclude that the validation and calibration test for the simulations run by *astroContact* are satisfactory and that the validity of the latter as a framework for



(a) Crater C0 with 20.0 m diameter.

(b) Crater C2 with 20.0 m diameter.

Fig. 13: Horizontal displacement plots for a 900 samples batch. Dots in the figure represent the final settling points for each of the simulations run, and their color code corresponds to the horizontal displacement relative to their initial position in the grid.

landing simulations including contact dynamics is confirmed.

6. Results

The ultimate idea, as briefly exposed in Section 1, is to generate safety maps that can be used to design the landing strategy for a certain mission scenario. To do that, a parametric analysis is performed, considering the different values that the design parameters could take. Thus, for each combination of design parameters, a batch of simulations needs to be performed to draw conclusions from it, using a Monte Carlo approach.

In order to do that, a script is generated, where the design parameters are specified, that calls various instances of Blender (depending on the machine's processing capabilities) to execute the simulations in parallel. Some of the design parameters used include grid resolution, CoR, attitude profile, initial position and velocity, or crater model, for instance. The design parameters used in this work are just a sample of what could be analysed. These parameters can potentially include anything that could influence the simulations, so it is possible to perform analysis with respect to any other aspects the mission designers deem relevant.

For the simulations presented in this work, the default parameters are collected in Table 4. If not otherwise specified, the results obtained would correspond to these numbers.

Table 4: Default input parameters for the contact dynamics simulations.

Parameter	**Value**
CoR spacecraft	0.5
CoR body	0.5
Coefficient of friction	0.5
Collision margin	1.0 cm
Grid resolution	1.0 m
Initial position	10.0 m altitude (on the grid)
Initial velocity	0.0 m s^{-1}
Initial attitude	Smaller CubeSat side to nadir

Simulations provide some information, such as the horizontal displacement analysis, i.e., the horizontal variation in position from the start of the trajectory (at the gate) to the final settling point (on the surface). Figure 13 shows the horizontal displacement for a simulation batch of 900 samples run using craters C0 and C2, both in their 20.0 m diameter versions. Trajectories that started above a highly-sloped point of the surface exhibit the greatest displacement, which is direct consequence of the slide/rebounds they suffer until they stop. Figure 13 includes a miniature of the crater used for the simulation and,

there, it can be observed how the crater morphology plays a major role in the distribution of the settling points. This is a powerful tool to study how different morphologies or features can affect the touchdown selection or settling probabilities in different spots of interest. For instance, the small hill in the centre of crater C2 clearly affects the distribution of the scattered points around it.

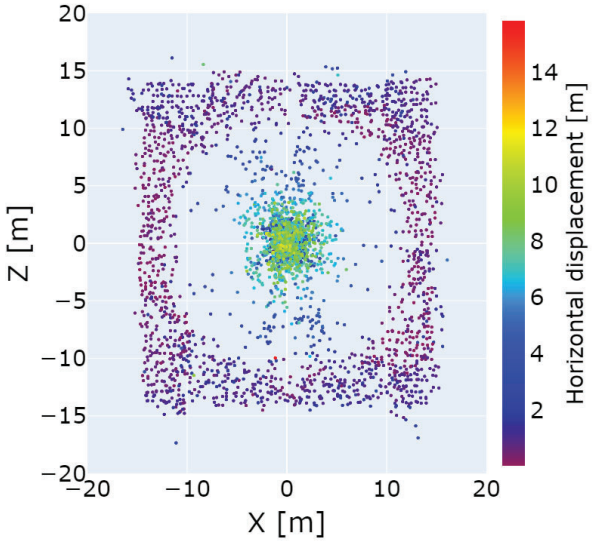


Fig. 14: Higher resolution (0.5 m at the gate, i.e. 3600 samples) batch horizontal displacement results on crater C0. Dots in the figure represent the final settling points for each of the simulations run, and their color code corresponds to the horizontal displacement relative to their initial position in the grid.

When increasing the resolution of the simulation (raising, thus, the number of samples), more detailed information on the nature of the landing can be extracted. Figure 14, for instance, shows the results of a 0.5 m resolution batch on crater C0 with 20.0 m diameter, featuring 3600 samples. There, as is for all the simulations shown in this work, the initial velocity is set to null, and the attitude profile pointed the smaller side of the CubeSat to nadir, while aligning the solar panels to the Z-axis. This crater is generated by smoothly and symmetrically deforming the surface of Dimorphos, so no irregularities or features are present in its morphology.

The most obvious result is that the trajectories whose first bounce happens within the crater diameter tend to end up lying on the centre of the concavity. On top of that, an interesting

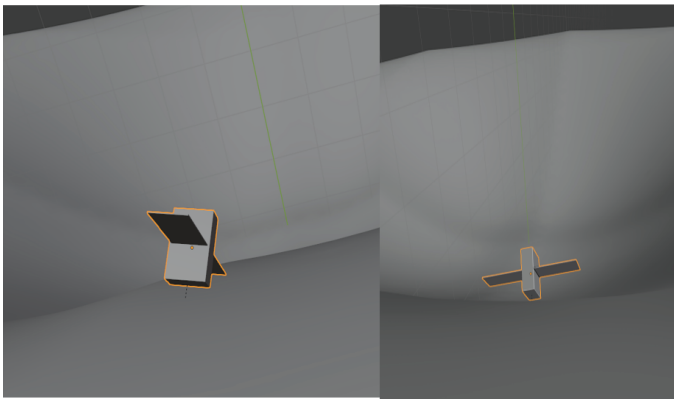


Fig. 15: Examples of landings where the panels generate a “braking effect” (left) and where they don’t (right) (Peñarroya et al., 2021).

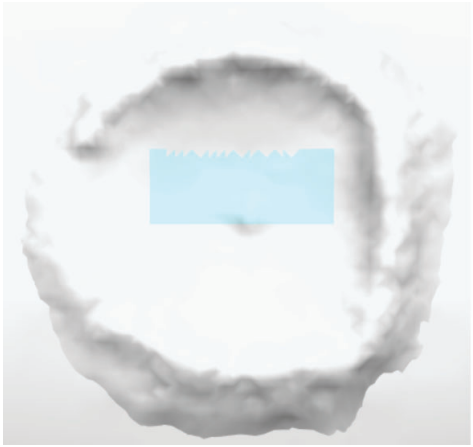


Fig. 16: Crater C2 with the target settling area highlighted in light-blue.

consequence of the selected attitude is a denser group of settling points along the Z-axis. This is explained by the orientation of the solar panels, which are parallel to this axis, and prevent the CubeSat from rolling towards the centre (see Figure 15). Because the CubeSat will slide over the crater’s interior instead of rolling, the settling point for a considerable number of trajectories ends up further away from the crater’s center. These results demonstrate how easily the attitude can affect the bouncing of the CubeSat on the surface of the asteroid.

With the data from the simulations, the next and final step is to create the safety maps for a certain landing strategy. First, a set of constraints that define the condition of ”safety” needs to be defined, e.g., areas of interest for the settling point, maximum number of bounces, duration of the landing until settling, or even specific criteria such as preventing the panels from suffering any impact. The way in which the creation of such maps

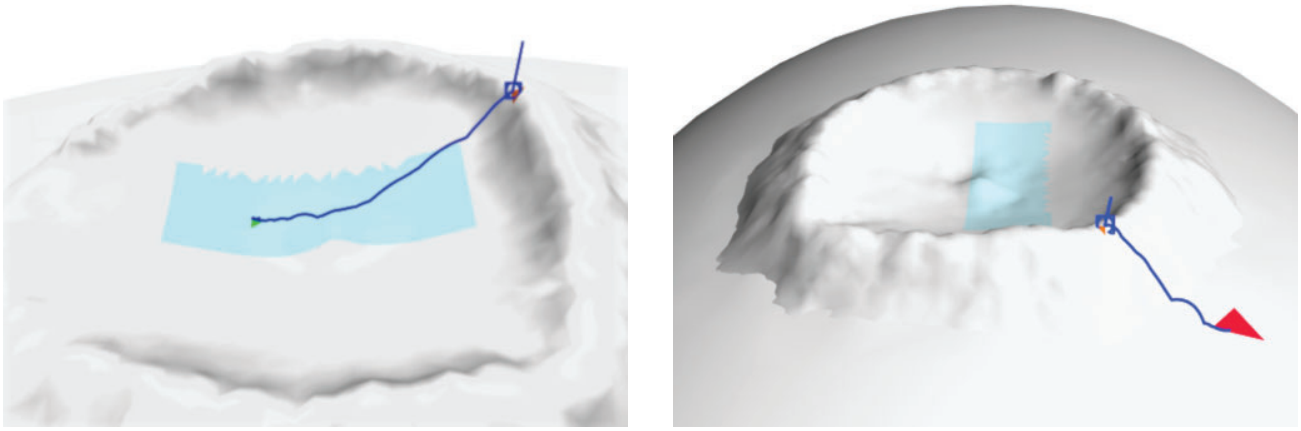
are conceived is to overlay a number of logic masks to each of the samples run on the Monte Carlo and deem successful those that are compliant with all criteria. Of course, the more detailed the success criteria, the fewer samples will be successful, and the less ”safe” the generated map will be.

For demonstration, a relatively simple safety criteria will be applied for the results that will be shown throughout this section, which will consist only on constraining the final settling area. Figure 16 shows an example of a target settling area for crater C2, spanning 20.0 m in diameter. In this example, crater C2 has been chosen because its features and irregularities make for richer and more complex simulation results. The reason why the upper bound of the light-blue area in the figure is not straight is due to the way in which the area is designated, which checks for each of the triangular facelets of the mesh, resulting on that saw-tooh shape visible on the upper limit.

Figure 1 in Section 3 represents the batch of simulations run for the creation of this map, where a resolution of 1.0 m for the initial grid, and a nadir pointing attitude are chosen; the same conditions used for the higher resolution test shown in Figure 14.

The generation of the maps require an individual evaluation of each of the samples on a simulation. For each of these samples, the touchdown point is retrieved by analysing the simulated velocities and accelerations. Then, the CubeSat trajectory is simulated until the settling point is reached. Once the trajectory ends, the safety criteria defined are evaluated to label the sample as ”safe” or ”unsafe”. In this case, the only criteria to be checked is whether the settling point is inside the target area or not. Figure 17 shows two samples with successful and failed results, respectively. Even though the touchdown points are close to each other, the irregularity of the terrain (in this case the rim of the crater) makes them diverge from each other, after impacting on the surface.

By repeating this process for each sample in the simulation, all of the touchdown points can be evaluated as ”safe” or ”unsafe”, creating a map that, for a given set of landing initial conditions (at the gate), serves as look-up table for touchdown point selection that can be used by the guidance algorithms for the mission to set the end-constraints for the trajectory optimisation. Figure 18 shows the results for the test under consideration. There, it can be observed how most of the touchdown points in the upper interior part of the crater are considered safe because the CubeSat ends up settling inside the target area. It is worth commenting on the isolated red points inside the mostly green area, and the green points inside the mostly red area. The fact that the surface of the asteroid (or the crater in this case) is not a smooth curve justifies the outliers inside confidence areas



(a) Success example.

(b) Failure example.

Fig. 17: Landing simulation examples used for safety map generation. Light blue represents the target area, the blue box represents the first touch-down, green represents settling point inside target area, and red represents settling point outside target area.

and shows how challenging and sensitive the problem is.

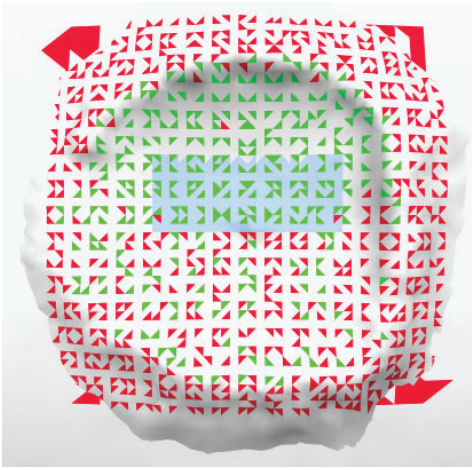


Fig. 18: Safety map for settling area constraints. Light blue represents the target area, green represents touchdown points that end up settling inside the target area, and red represents touchdown points settling outside the target area. Success ratio is 32.96%.

Post-processing the results on the map in Figure 18 shows a success ratio of 32.96%, understood as the samples deemed safe with respect to the total number of samples. This number has no practical meaning by itself, since it depends greatly on the simulation resolution or the restrictiveness of the safety criteria selected. However, it becomes useful when compared to other maps where some input parameters have been modified, and provides insight into how the different landing design parameters affect the safety maps. Figure 19 shows two examples of modifications that affect the generation of safety maps.

The map shown in Figure 19a corresponds to a case where the safety criteria has been modified: the target area has been shifted to the lower half of the crater. Due to crater morphology, the success ratio drops to 18.09%, which is a consequence from the fact that this lower half of the crater is richer in irregularities that prevent the CubeSat from settling in the target area and that the slope of that side of the crater is higher and makes it more challeging for the CubeSat to settle close to the centre of the crater.

One of the problems observed in Figure 18 is that there are some inconsistencies inside the safe areas, corresponding to red outlying points. Van Wal et al. (2017) show in their work how the angle of impact plays an important role in the settling time for the simulation, increasing it and letting the lander traverse longer. Following that idea, and in an attempt to improve the obtained results, a new simulation batch is run in which the attitude at impact is not nadir pointing but a tilt of 45.0° is introduced so that the solar panels would hit the surface first, introducing some rotational momentum onto the CubeSat. This case might not be feasible for missions that need to preserve the integrity of the panels to perform tasks on the surface of the body. However, using the structure of the panels as a damping system for the impact energy is something that could be utilised in missions where the landing trajectory follows a ballistic descent without braking mechanisms and that need to preserve the integrity of the scientific payload. By doing so, the success ratio raised to a 34.62% (1.66% higher than the first attempt), as depicted by subfigure 19b. Even though the increase in success ratio might seem small, the reader needs to recall that this ratio is computed with respect to the totality of the samples simulated, not the specific area that is intended to improve. Upon visual inspection of the map, it can be clearly observed how the mostly green area is much more consistent now, and few outliers are present, which supports the design strategy of tilting the CubeSat before impact to favour rotation towards the centre.

According to Van Wal et al. (2017), the coefficient of restitution of the surface is found to have a stronger effect on the deployment statistics than the coefficient of friction, which appears to have no effect in their results. Performing similar sensitivity analyses, Figure 20 shows the results obtained from sensitivity analysis for both cases.

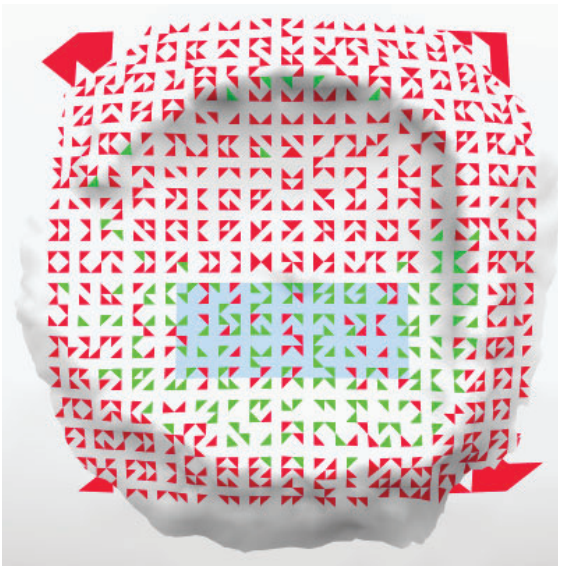
For this particular scenario, results show that the success ratio drops over 1% when the CoR is increased from 0.5 to 1.0. On the other hand, changing the friction coefficient from 0.5 to 1.0 considerably reduces the success ratio, dropping it to 24.22%. To better understand these plots, the time-of-settling plots are also generated for the four cases presented so far with the target area on the upper half of the crater. Figure 21 shows the obtained results. There, it can be observed how changing the initial attitude of the CubeSat shifts the distribution to the

right, i.e., increases the settling times, allowing the spacecraft to traverse longer and reach the target area, as it was expected.

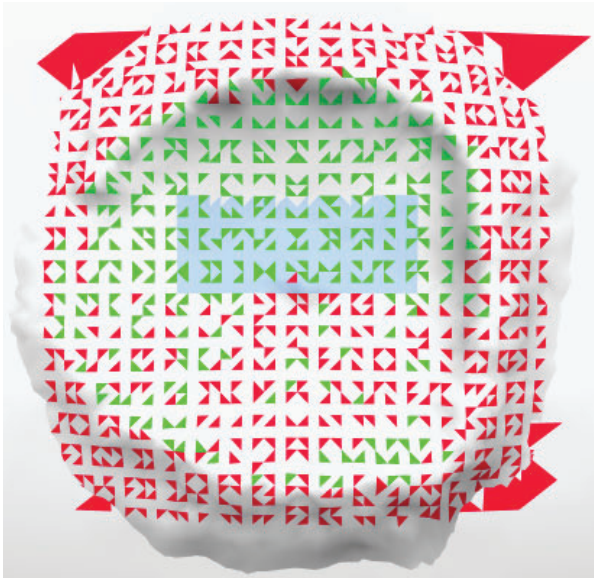
Results for the increased CoR present a similar effect in the settling time to those of the change in the initial attitude. However, the success ratio drops and the outlier touchdown points do not seem to disappear, contrarily to the modified attitude example. On the other hand, changing the coefficient of friction makes a larger impact in the results, both from a success ratio and a time of settling perspective. Simulation times were capped to 5000.0 s, which explains the last bar in the three first histograms of Figure 21, but in the fourth scenario (increased friction), the distribution moves to the left so much that the time-cap can no longer be appreciated. This result contrasts with the conclusions obtained by Van Wal et al. (2017), where the authors deemed the effect of the CoR to be much stronger than that of the coefficient of friction.

Upon further investigation, the difference is found to be due to the nature of the simulation scenario. In the results shown in this work, the trajectories present mainly two types of behaviour: bouncing and sliding. The bouncing (including impacts and rolling) is led primarily by the CoR, while the sliding is determined by the coefficient of friction. The simulation scenario used for the analyses shown here consists on a crater, which features relatively large slopes (with respect to the surrounding asteroid's surface, for instance). If the spacecraft's geometry is such that rolling is not favoured by it the spacecraft trajectory slides downhill due to the slopes of crater until, by means of frictional forces, it settles down in a certain location. It has already been described how the solar panels prevented the CubeSat from rolling in in this chapter (see Figure 14), and Van Wal et al. (2020) already showed how the geometry of the probes used in their simulations affected the settling time, mostly due to "the shapes' geometry, which determines the range of aspect angles at which surface impacts can take place". There, it was found that a tetrahedron was the shape that shortened the most the settling time, followed by the octahedron, the cube, the icosahedron, and the dodecahedron. Looking at Figure 4, it can be argued that the geometry of the CubeSat used in this work resembles that of a cube, or even a octahedron, if the solar panels are considered.

The result, as it can be observed in Figure 20b is that most of the successful touchdown points on the lower half of the crater turn into failures because the higher friction prevents them from sliding longer and reaching the target area. Even some touch-

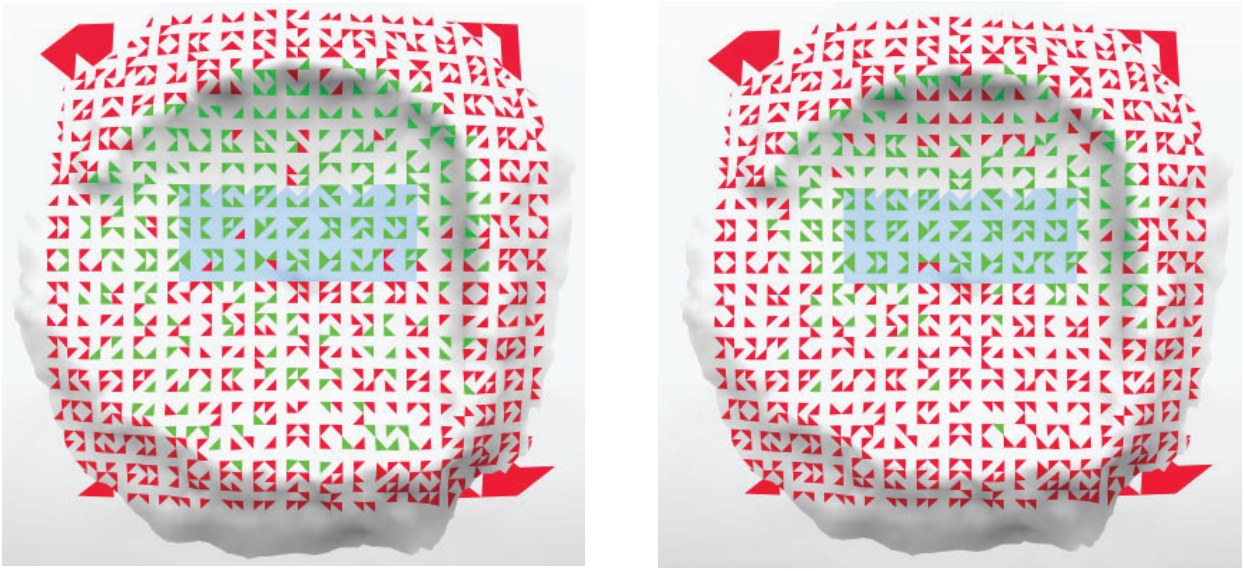


(a) Shift in the target area. Success ratio is 18.09%.



(b) Change of impact attitude. Success ratio is 34.62%.

Fig. 19: Landing simulation examples used for safety map generation with modified criteria and landing design parameters. Light blue represents the target area, orange represents the first touch-down, green represents settling point inside target area, and red represents settling point outside target area.



(a) CoR for the body changed to 1.0. Success ratio is 31.88%.

(b) Coefficient of friction for the body changed to 1.0. Success ratio is 24.22%.

Fig. 20: Landing simulation examples used for safety map generation with modified criteria and landing design parameters. Light blue represents the target area, orange represents the first touch-down, green represents settling point inside target area, and red represents settling point outside target area.

down points on the upper half of the crater turn into failure cases, constraining the landing spot selection further.

The generated safety maps provide a straight-forward and clear way to understand which regions can be targeted for a given landing scenario and a set of configuration parameters. They are a valuable tool for landing spot selection and trajectory design, and could be even used to iterate the geometry of the spacecraft to fit the morphology of a certain landing region, such as a crater, rocky surfaces, or flat areas.

One of the most challenging aspects of landing on asteroids is the uncertainty about the physical properties of the body of interest. Very often, detailed information can only be obtained when the spacecraft already orbits close to the body, or even at the contact itself. Ideally, safety maps could be computed on board for any update on the scenario parameters for the landing. However, the generation of such maps requires relatively high computation capabilities, so doing so on-board is, at least with the current implementation, not possible. A proposed alternative would be to perform a sensibility analysis for a set of parameters of interest. Thus, a number of maps can be computed and stored to use as look-up tables by mission designers and operators.

Even uploading these pre-computed maps to the spacecraft is useful for the final approach trajectory. For the study-case under consideration in this work, the CubeSat has no control over its trajectory but it does over its attitude. Therefore, and provided a sufficiently high navigation loop frequency, these maps could be used to extract the touchdown point from the current trajectory estimated by the navigation filter and steer the spacecraft to an attitude profile that satisfies the imposed conditions for successful landing, e.g., settling inside the light-blue area in the example provided in this work.

7. Conclusions

In this work, Blender has been adopted as a simulation tool for contact dynamics applied to CubeSat landing within the crater region of asteroid Dimorphos. The system dynamics have been replaced by a force field attached to a rigid-body and the trajectories have been propagated using Blender's own numerical methods.

Furthermore, the developed software has been included into the *astroContact* module to generate safety maps for landing strategy design. This module allows for parametric analysis of relevant landing design parameters including (but not lim-

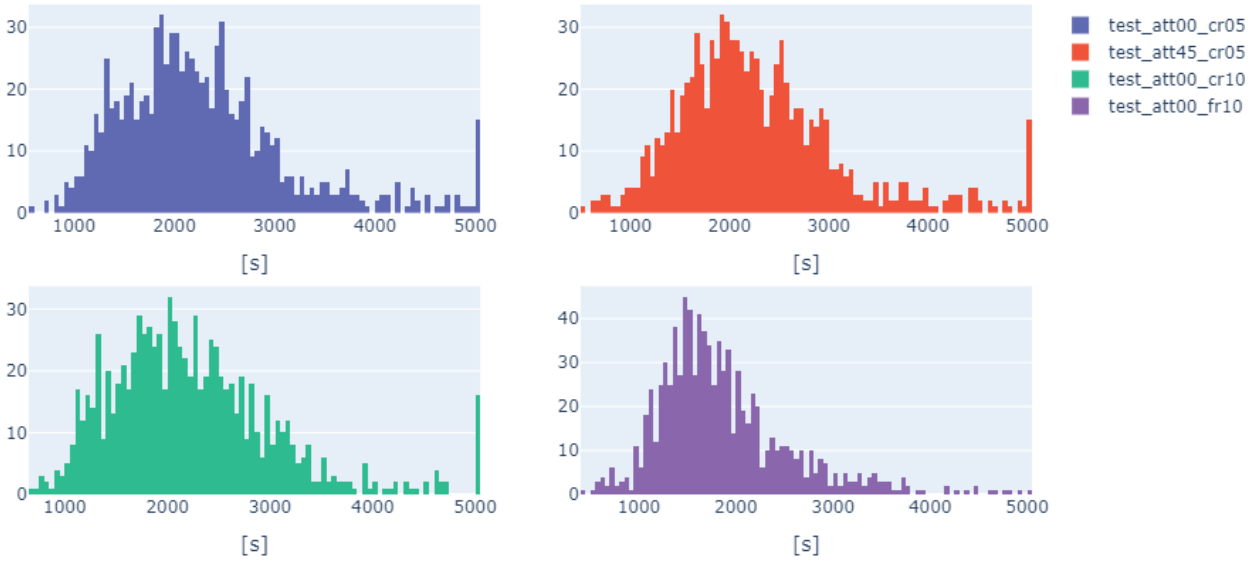


Fig. 21: Histogram for the settling times for different scenarios. Upper-left is the nominal case, upper-right case was generated by changing the initial attitude of the spacecraft, lower-left case was generated by increasing the CoR of the body to 1.0, and lower-right case was generated increasing the coefficient of friction to 1.0.

ited to) spacecraft initial state and pose, CoR for the spacecraft and the body, coefficient of friction, mesh geometry and features, collision margins and shape, spacecraft mass and inertia properties, or simulation time-step, for instance. It provides a useful, quick, and reliable framework for missions attempting landings, giving mission designers an excellent option to tackle these very challenging tasks.

Even though a validation and calibration tests have been performed, comparing *astroContact* to other state-of-the-art models and tools for contact dynamics would help to further benchmark its performance. Both an energy-based validation and a qualitative validation have been performed. The former using a sample trajectory and analysing the evolution in time of the energy of the system and the latter against results available in the literature showing how the results obtained in this work follow the same behaviour, only affected by the presence of crater slopes in the landing simulations presented in this work. Future work will be directed towards performing a validation campaign against a peer-reviewed state-of-the-art method to further support the results obtained by *astroContact*.

Safety maps generated by this tool can be used as a complement to other hazard detection modules to determine safe landing spots. Training a model with a set of maps accounting for possible ranges of landing design parameters could be used on-board as look-up tables to increase mission success, by changing attitude profiles or re-directing the trajectory to a safer area. These maps could then even be over overlaid with classical Hazard Detection and Avoidance (HDA) maps such as shadow detection, feature detection, or slope estimation ones, contributing to the final safe area estimation.

Using these maps for the study-case of a CubeSat landing on an asteroid crater, it is found that, pitching the impact attitude of the spacecraft has a beneficial impact when the target settling area is in the centre of the crater, since it increases the horizontal displacement of the touchdown points that are further apart from it. It is also observed how the coefficient of friction appears to have a stronger influence in the settling time when it comes to sloped terrain (such as craters) than the CoR, for spacecraft geometries that do not favour rolling.

Future steps are directed towards how to make use of such maps for navigation purposes, correlating them with on-board images. The use of these maps for autonomous landing site selection on-board could be an interesting research path, even including Machine Learning (ML) methods (instead of traditional look-up table approaches) to mitigate the computational effort.

8. Acknowledgments

P.P., M.P., S.C., F.T., and M.S. would like to acknowledge the funding received from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813644.

Also, the authors thank all the reviewers (internal and external) for their important contributions to the final quality of this paper.

References

Allen, C. C. (1975). Central peaks in lunar craters. *The Moon*, *12*(4), 463–474. URL: http://link.springer.com/10.1007/BF00577935. doi:10.1007/BF00577935.

Barringer, D. M. (1914). Further Notes on Meteor Crater, Arizona. *Proceedings of the Academy of Natural Sciences of Philadelphia*, *66*(3), 556–565. URL: https://www.jstor.org/stable/4063595.

Bekker, M. G. (1960). *Off-the-road locomotion; research and development in terramechanics.*. Ann Arbor: University of Michigan Press. OCLC: 1940356.

Bennett, F. V. (1970). Apollo Lunar Descent and Ascent Trajectories. In *AIAA 8th Aerospace Sciences Meeting* (p. 35). New York, NY.

Berry, K., Sutter, B., May, A. et al. (2013). OSIRIS-REx touch-and-go (TaG) mission design and analysis. *Advances in the Astronautical Sciences*, *149*, 667–678.

Brochard, R., Lebreton, J., Robin, C. et al. (2018). Scientific image rendering for space scenes with the SurRender software. In *69th International Astronautical Congress (IAC)*. Bremen, Germany.

Caruso, M., Scalera, L., Gallina, P. et al. (2020). Dynamic Modeling and Simulation of a Robotic Lander Based on Variable Radius Drums. *Applied Sciences*, *10*(24), 8862. URL: https://www.mdpi.com/2076-3417/10/24/8862. doi:10.3390/app10248862.

Cheng, A., Michel, P., Reed, C. et al. (2012). Dart: Double asteroid redirection test. In *European Planetary Science Congress* (pp. 23–28). volume 7.

Chesley, S. R., French, A. S., Davis, A. B. et al. (2020). Trajectory Estimation for Particles Observed in the Vicinity of (101955) Bennu. *Journal of Geophysical Research: Planets*, *125*(9), e2019JE006363. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2019JE006363. doi:10.1029/2019JE006363. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2019JE006363.

Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation. URL: http://www.blender.org.

Coumans, E., & Bai, Y. (2016). PyBullet, a Python module for physics simulation for games, robotics and machine learning. URL: http://pybullet.org.

Craighead, J., Burke, J., & Murphy, R. (2008). Using the unity game engine to develop sarge: a case study. In *Proceedings of the 2008 Simulation Workshop at the International Conference on Intelligent Robots and Systems (IROS 2008)*. volume 4552.

Dickson, P. E., Block, J. E., Echevarria, G. N. et al. (2017). An Experience-based Comparison of Unity and Unreal for a Stand-alone 3D Game Development Course. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 70–75). Bologna Italy: ACM. URL: https://dl.acm.org/doi/10.1145/3059009.3059013. doi:10.1145/3059009.3059013.

Dikaiakos, M. D., & Stadel, J. (1996). A performance study of cosmological simulations on message-passing and shared-memory multiprocessors. *Proceedings of the 10th international conference on Supercomputing - ICS*, (pp. 94–101). URL: https://www.academia.edu/21623534/A_performance_study_of_cosmological_simulations_on_message_passing_and_shared_memory_multiprocessors.

Ferrari, F., Franzese, V., Pugliatti, M. et al. (2021a). Preliminary mission profile of Hera's Milani CubeSat. *Advances in Space Research*, *67*(6), 2010–2029. URL: https://www.sciencedirect.com/science/article/pii/S0273117720309078. doi:10.1016/j.asr.2020.12.034.

Ferrari, F., Franzese, V., Pugliatti, M. et al. (2021b). Trajectory Options for Hera's Milani CubeSat Around (65803) Didymos. *J Astronaut Sci*, *68*(4), 973–994. URL: https://link.springer.com/10.1007/s40295-021-00282-z. doi:10.1007/s40295-021-00282-z.

Gerhart, G. R. (2004). The Bekker Model Analysis for Small Robotic Vehicles. *SAE Transactions*, *113*, 317–324. URL: https://www.jstor.org/stable/44718826.

Giesbers, J. (2012). Contact mechanics in MSC Adams - A technical evaluation of the contact models in multibody dynamics software MSC Adams. URL: http://essay.utwente.nl/62109/.

Goldberg, H., Karatekin, , Ritter, B. et al. (2019). The Juventas CubeSat in Support of ESA's Hera Mission to the Asteroid Didymos. *Small Satellite Conference*, . URL: https://digitalcommons.usu.edu/smallsat/2019/all2019/73.

González, Á. (2009). Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, *42*(1), 49–64. URL: https://doi.org/10.1007%2Fs11004-009-9257-x. doi:10.1007/s11004-009-9257-x.

Grimm, C., Hendrikse, J., Lange, C. et al. (2013). DLR MASCOT on HAYABUSA-II, A Mission That May Change Your Idea of Life: AIV Challenges in a Fast Paced and High Performance Deep Space Project. In *29th International Symposium on Space Technology and Science (ISSN: 0549-3811)*. Nagoya, Aichi Prefecture, Japan. URL: http://archive.ists.or.jp/upload_pdf/2013-k-51.pdf. doi:10/1/2013-k-51.pdf.

Guevarra, E. T. M. (2020). Blending with Blender: Getting Started. In E. T. M. Guevarra (Ed.), *Modeling and Animation Using Blender: Blender 2.80: The Rise of Eevee* (pp. 27–85). Berkeley, CA: Apress. URL: https://doi.org/10.1007/978-1-4842-5340-3_2. doi:10.1007/978-1-4842-5340-3_2.

Ivaldi, S., Peters, J., Padois, V. et al. (2014). Tools for simulating humanoid robot dynamics: A survey based on user feedback. In *2014 IEEE-RAS International Conference on Humanoid Robots* (pp. 842–849). doi:10.1109/HUMANOIDS.2014.7041462 iSSN: 2164-0580.

Krenn, R., & Hirzinger, G. (2008). Contact Dynamics Simulation for Space Robotics Applications. *Proceedings of IROS*, (p. 8).

Marchi, S., Chapman, C. R., Barnouin, O. S. et al. (2015). Cratering on Asteroids. In P. Michel, F. E. DeMeo, & W. F. Bottke (Eds.), *Asteroids IV* (pp. 725–744). University of Arizona Press. URL: http://muse.jhu.edu/chapter/1705194. doi:10.2458/azu_uapress_9780816532131-ch037.

Martin, I., Dunstan, M., & Gestido, M. S. (2019). Planetary surface image generation for testing future space missions with pangu. In *2nd RPI Space Imaging Workshop*. Sensing, Estimation, and Automation Laboratory.

Michel, P., Cheng, A., Küppers, M. et al. (2016). Science case for the Asteroid Impact Mission (AIM): A component of the Asteroid Impact & Deflection Assessment (AIDA) mission. *Advances in Space Research*, *57*(12), 2529–2547. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117716300692. doi:10.1016/j.asr.2016.03.031.

Michel, P., Küppers, M., Fitzsimmons, A. et al. (2021). The ESA Hera mission to the near-Earth asteroid binary Didymos: planetary defense and science return. In *Global Space Exploration Conference 2021 (GLEX 2021)*. Online. URL: https://elib.dlr.de/142762/.

Mills, S., Mason, A., & Parkes, S. (2010). STAR-Dundee Virtual Devices and System Simulation. In *International SpaceWire Conference*.

Muller, P. M., & Sjogren, W. L. (1968). Mascons: Lunar mass concentrations. *Science*, *161*(3842), 680–684. doi:10.1126/science.161.3842.680.

Naidu, S. P., Benner, L. A. M., Brozovic, M. et al. (2020). Radar observations and a physical model of binary near-Earth asteroid 65803 Didymos, target of the DART mission. *Icarus*, *348*, 113777. URL: https://www.sciencedirect.com/science/article/pii/S0019103520301640.

doi:10.1016/j.icarus.2020.113777.

Ormö, J., Raducan, S. D., Luther, R. et al. (2020). *Effects of target heterogeneity on impact cratering processes in the light of the Hera mission: combined experimental and numerical approach*. Technical Report EPSC2020-922 Copernicus Meetings. URL: https://meetingorganizer.copernicus.org/EPSC2020/EPSC2020-922.html. doi:10.5194/epsc2020-922.

Pajusalu, M., Iakubivskyi, I., Schwarzkopf, G. J. et al. (2022). SISPO: Space Imaging Simulator for Proximity Operations. *PLoS ONE*, *17*(3). URL: http://arxiv.org/abs/2105.06771. ArXiv: 2105.06771.

Peñarroya, P., Centuori, S., & Hermosín, P. (2022). AstroSim: A GNC simulation tool for small body environments. In *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2022-2355. doi:10.2514/6.2022-2355.

Peñarroya, P., Pugliatti, M., Centuori, S. et al. (2021). Using Blender As Contact Dynamics Engine For Cubesat Landing Simulations Within Impact Crater On Dimorphos. In *7th IAA Planetary Defense Conference*. doi:10.13140/RG.2.2.22662.91206.

Pugliatti, M., Franzese, V., & Topputo, F. (2022). Data-Driven Image Processing for Onboard Optical Navigation Around a Binary Asteroid. *Journal of Spacecraft and Rockets*, *59*(3), 1–17. URL: https://doi.org/10.2514/1.A35213. doi:10.2514/1.A35213.

Pugliatti, M., Maestrini, M., Di Lizia, P. et al. (2021). On-Board Small-Body Semantic Segmentation Based on Morphological Features with U-Net. In *31st AAS/AIAA Space Flight Mechanics Meeting* (pp. 1–20).

Rivkin, A. S., Chabot, N. L., Stickle, A. M. et al. (2021). The double asteroid redirection test (DART): Planetary defense investigations and requirements, . *2*(5), 173. doi:10.3847/psj/ac063e.

Sanders, A. (2016). *An introduction to Unreal engine 4*. AK Peters/CRC Press.

Senkic, D. (2010). *Dynamic simulation in a 3D-environment : A comparison between Maya and Blender*. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-7871.

Sunday, Cecily, Zhang, Yun, Thuillet, Florian et al. (2021). The influence of gravity on granular impacts - I. A DEM code performance comparison. *A&A*, *656*, A97. URL: https://doi.org/10.1051/0004-6361/202141412. doi:10.1051/0004-6361/202141412.

Takashi Kubota, Masatsugu Otsuki, & Tatsuaki Hashimoto (2008). Touchdown dynamics for sample collection in Hayabusa mission. In *2008 IEEE International Conference on Robotics and Automation* (pp. 158–163). Pasadena, CA, USA: IEEE. URL: http://ieeexplore.ieee.org/document/4543202/. doi:10.1109/ROBOT.2008.4543202.

Tardivel, S., Scheeres, D. J., Michel, P. et al. (2014a). Modelling of Asteroid Surfaces to Understand Landing Operations. *11th International Planetary Probe Workshop*, *1795*, 8104. URL: https://ui.adsabs.harvard.edu/abs/2014LPICo1795.8104T. ADS Bibcode: 2014LPICo1795.8104T.

Tardivel, S., Scheeres, D. J., Michel, P. et al. (2014b). Contact Motion on Surface of Asteroid. *Journal of Spacecraft and Rockets*, *51*(6), 1857–1871. URL: https://doi.org/10.2514/1.A32939. doi:10.2514/1.A32939. Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.A32939.

Thuillet, F., Zhang, Y., Michel, P. et al. (2021). Numerical modeling of lander interaction with a low-gravity asteroid regolith surface - II. Interpreting the successful landing of Hayabusa2 MASCOT. *A&A*, *648*, A56. URL: https://www.aanda.org/articles/aa/abs/2021/04/aa36128-19/aa36128-19.html. doi:10.1051/0004-6361/201936128. Publisher: EDP Sciences.

Van Wal, S., Reid, R. G., & Scheeres, D. J. (2020). Simulation of nonspherical asteroid landers: Contact modeling and shape effects on bouncing. *Journal of Spacecraft and Rockets*, *57*(1), 109–130. doi:10.2514/1.a34573.

Van Wal, S., Tardivel, S., & Scheeres, D. (2017). Parametric Study of Ballistic Lander Deployment to Small Bodies. *Journal of Spacecraft and Rockets*, *54*(6). URL: https://arc.aiaa.org/doi/abs/10.2514/1.A33832. doi:10.2514/1.A33832.

Werner, R., & Scheeres, D. (1997). Exterior gravitation of a polyhedron derived and compared with harmonic and mascon gravitation representations of asteroid 4769 Castalia. *Celestial Mech Dyn Astr*, *65*(3). URL: http://link.springer.com/10.1007/BF00053511. doi:10.1007/BF00053511.

Wiegert, P. (2020). On the Delivery of DART-ejected Material from Asteroid (65803) Didymos to Earth. *Planet. Sci. J.*, *1*(1), 3. URL: https://iopscience.iop.org/article/10.3847/PSJ/ab75bf/meta. doi:10.3847/PSJ/ab75bf. Publisher: IOP Publishing.

Wood, J., Margenet, M. C., Kenneally, P. et al. (2018). Flexible Basilisk astrodynamics visualization software using the Unity rendering engine. In *AAS Guidance and Control Conference, Breckenridge, CO*.

Yano, H., Kubota, T., Miyamoto, H. et al. (2006). Touchdown of the Hayabusa Spacecraft at the Muses Sea on Itokawa. *Science*, *312*(5778), 1350–1353. URL: https://science.sciencemag.org/content/312/5778/1350. doi:10.1126/science.1126164.

Zhang, Y., Li, J., Zeng, X. et al. (2021). High-fidelity landing simulation of small body landers: Modeling and mass distribution effects on bouncing motion. *Aerospace Science and Technology*, *119*, 107149. URL: https://www.sciencedirect.com/science/article/pii/S1270963821006593. doi:10.1016/j.ast.2021.107149.

Appendix A. Acronyms

List of Acronyms

AIDA	Asteroid Impact and Deflection Assessment
AP	Albedo Pressure
AstroSim	Astrodynamics Simulator
CNES	Centre National D'Etudes Spatiales
CNN	Convolutional Neural Network
CoB	Coefficient of Bounciness
CoR	Coefficient of Restitution
COTS	Commercial Off-The-Shelf
DART	Double Asteroid Redirection Test
ESA	European Space Agency
GNC	Guidance, Navigation, and Control
GUI	Graphical User Interface
HDA	Hazard Detection and Avoidance
IP	Image Processing
IS	International System
Mascon	Mass Concentrations
MASCOT	Mobile Asteroid Surface Scout
ML	Machine Learning
NASA	National Aeronautics and Space Administration
NEA	Near-Earth Asteroid
OSIRIS-REx	Origins, Spectral Interpretation, Resource Identification, and Security–Regolith Explorer
PANGU	Planet and Asteroid Natural scene Generation Utility
PCM	Polygonal Contact Model
PKDGRAV	Parallel K-D tree GRAVity code
S/C	Spacecraft
SRP	Solar Radiation Pressure
TaG	Touch-and-Go
TBG	Third-Body Gravity
TRP	Thermal Radiation Pressure
VFX	Visual Effects