AUTONOMOUS ROBOTIC GRASPING USING DEEP LEARNING METHODS FOR RESOURCE EFFICIENT SPACE ROBOTICS APPLICATIONS

Maha Badri *, Moritz Gewehr *, Sabine Klinkner *
* Institute of Space Systems, University of Stuttgart, Pfaffenwaldring 29, 70569 Stuttgart, Germany
maha_badri@yahoo.fr, gewehr@irs.uni-stuttgart.de, klinkner@irs.uni-stuttgart.de

Abstract

Planetary rovers have proven to be invaluable assets within space exploration missions, providing profound insights and expanding scientific knowledge. As the frontiers of space exploration continue to expand, there is a growing demand for rovers with advanced capabilities and a higher degree of autonomy. A key aspect of future rover generations involves their ability to autonomously manipulate objects in extraterrestrial environments, particularly in missions involving sample collection, analysis, or return. However, this task presents significant challenges due to the unknown nature of objects and the complex terrains encountered. To address these challenges, this paper investigates the potential of deep learning techniques to enhance autonomous robotic grasping in extraterrestrial environments. The proposed approach introduces an end-to-end grasp estimation system, enabling rovers equipped with robotic arms to autonomously execute grasping actions solely based on visual information provided by its on-board sensors. Transfer learning is employed, harnessing the power of pre-trained deep learning models from computer vision applications and pre-existing public grasping datasets. These models are then finetuned using a self-generated dataset containing objects relevant to manipulation tasks in space exploration missions. To bridge the gap between space exploration and deep learning research, a pipeline is introduced to automatically generate a large, labelled dataset of objects suitable for autonomous grasping in planetary exploration missions. Additionally, a 3D planetary robot simulation environment is developed as the core platform for generating and automatically labelling synthetic custom data, emulating conditions encountered in extraterrestrial environments. Preliminary results demonstrate the promising ability of the system to successfully grasp novel objects based only on RGB-D visual information, achieving a success rate of 85%. This approach enables future rovers equipped with human-like grasping abilities to operate without prior knowledge of target objects, while maintaining resource efficiency, thereby enhancing autonomous on-board decision-making in space missions.

Keywords

Space Robotics; Deep Learning; Autonomous robotic grasping; RGB-D perception; Transfer Learning; Convolutional neural network; End-to-end grasping; Object manipulation

NOMENCLATURE

Symbols

h	Height of the grasping rectangle
μ	Friction Coefficient
μ_k	Kinetic Friction Coefficient
μ_s	Static Friction Coefficient
Q1	First quartile
Q3	Third quartile
σ	Activation Function
θ	Orientation of the grasping rectangle
w	Width of the grasping rectangle
(x, y)	Centre of the grasping rectangle

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CAD	Computer Aided Design
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DOF	Degrees of Freedom
ESA	European Space Agency
GPU	Graphics Processing Unit
GUI	Graphical User Interface
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
IQR	Interquartile Range
ML	Machine Learning
MRCP	Modular Rover Chassis Platform

doi: 10.25967/610429

MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NUMA	Non-Uniform Memory Access
ReLU	Rectified Linear Unit
RGB-D	Red, Green, Blue and Depth
RGB	Red, Green and Blue
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RMSProp	Root Mean Squared Propagation
RNN	Recurrent Neural Network
ROI	Region Of Interest
SGD	Stochastic Gradient Descent
STN	Spatial Transformer Network
URDF	Unified Robot Description Format
V-HACD	Volumetric Hierarchical Approximate Convex Decomposition

1. INTRODUCTION

The exploration of space has been revolutionised by the deployment of planetary rovers, which have provided invaluable insights into previously inaccessible extraterrestrial environments. Generations of mobile robotic systems (rovers) such as Spirit, Opportunity, and Curiosity have made remarkable achievements by exploring planetary surfaces, leading to groundbreaking discoveries that enhanced knowledge gains in multidisciplinary science fields [1]. Despite their significant technological advancements, current rover operations are still heavily reliant on human operators due to their limited autonomous capabilities [2, 3]. As the frontiers of space exploration continue to expand, future missions face increasingly complex challenges. These include the necessity to cover longer traversal distances, navigate through extreme environments that are scientifically intriguing like crater slopes or lava tubes, employ more sophisticated scientific instruments, and manage large amounts of data. However, the vast distances between Earth and remote exploration sites introduce significant delays in transmitting commands and receiving data, which reduce the time available for crucial mission tasks, thereby restricting the rovers' potential for exploration as well as their ability to adapt to unforeseen situations. Therefore, enhancing the autonomous capabilities of planetary rovers becomes crucial in order to achieve the ambitious goals of future space exploration missions. In particular, autonomous grasping and object manipulation play a central role, especially in the context of analyses and processing, or even in complex scenarios like sample return missions aimed at gathering valuable data from celestial bodies [4, 5]. In such missions, planetary rovers equipped with robotic arms must possess the capability to safely retrieve samples and store them in secure containers for transportation back to Earth. Furthermore, planetary rovers that are able to autonomously grasp diverse objects and tools can perform a wide range of tasks that extend beyond sample collection, such as collecting objects of interest, manipulating complex scientific instruments, executing assembly operations, as well as performing maintenance and servicing tasks. However, developing efficient grasping methods that meet the increasing demands for autonomy while effectively managing onboard resources presents a significant challenge. Traditional techniques for autonomous object manipulation heavily rely on prior knowledge of the target object and its properties, often necessitating the maintenance of a database of offline pre-computed grasps for each object the robot might encounter, which can be resource-intensive, as the size of such a database grows exponentially with each potential object. Moreover, the uncertain nature of extraterrestrial environments introduces another layer of complexity to the grasping task, which traditional approaches fail to handle, as they lack the necessary generalisation capabilities due to their reliance on known objects and specific environmental conditions. As a result, there is a compelling need for novel, resource-efficient object grasping and manipulation methods that can effectively address the uncertainties inherent to space exploration missions.

Deep learning-based approaches have emerged as a promising alternative to traditional grasping methods in the robotics community. These approaches enable robots to learn robust and generalisable behaviours, which holds great promise for enhancing the effectiveness of robotic grasping in extraterrestrial environments. Despite the recent success of deep learning in robotics, its application in space robotics is still in its early stages. As the next phase of the Mars Sample Return Program approaches, involving the retrieval of samples collected by Perseverance from the Martian surface, recent studies [6–11] have demonstrated the potential of deep learning techniques for enhancing the sample retrieval process, leading to a more efficient and precise sample collection. While the focus of these studies has been on learning partial components of the robotic grasping pipeline, such as object localisation or pose estimation, a comprehensive approach to learning the complete manipulation pipeline in an end-to-end manner is yet to be explored. This paper aims to build upon this previous work and presents an end-to-end grasp estimation system by applying deep learning techniques to the task of vision-based robotic grasping in extraterrestrial settings.

Previous work at the Space Robotics Lab at the University of Stuttgart put strong emphasis on investigating space exploration technologies and developing technical solutions for future surface mission applications. Within this context, a Modular Rover Chassis Platform (MRCP) was developed for various robotic technology demonstration scenarios. The MRCP also supports model payloads such as 5-DOF robotic arms and deployable camera masts for object detection [12, 13]. The robotic arms, developed within interdisciplinary research projects, linking industry, research and education, demonstrated their ability to grasp and store sample tubes in a designated sample dispenser on the rover, as well as secondary surface object. However, to effectively enhance the technologies and capabilities required for future surface exploration missions, a higher level of autonomy is essential for the rover to autonomously identify, collect, and store samples. Continual research work in the field of space robotic exploration technologies

at the Space Robotics Lab opened up the research field of deep learning methods for autonomous grasping technologies in space applications. Within this context, this paper shall describe and reflect on five major developments of the ongoing research.

- Development of a resource-efficient end-to-end grasp estimation system: the system is designed to allow rovers equipped with a robotic arm to learn grasping based solely on visual information acquired by the sensors on-board. This is achieved by using deep learning techniques to train the system to recognise objects and determine the best way to grasp them. The computationally intensive training process will be conducted on Earth, and once trained, the resulting models can be used in conjunction with the sensors on-board to accurately identify objects and infer the best grasp configuration.
- Application of transfer learning with off-the-shelf pre-trained deep learning models: due to the lack of publicly available labelled datasets relevant to space exploration missions, the study applies transfer learning. This involves using pre-trained deep learning models, which are then finetuned using a self-generated dataset containing objects relevant to manipulation tasks in space exploration missions.
- Generation of a labelled dataset of objects suitable for autonomous grasping in planetary exploration missions: the study includes the development of a pipeline for automatically generating a large, labelled dataset. This dataset includes objects that are suitable for autonomous grasping in the context of planetary exploration missions.
- Development of a 3D planetary robot simulation environment: the study involves the creation of a 3D planetary robot simulation environment, which serves as the core platform for generating and automatically labelling synthetic custom data.

A visionary motivation for this approach is that future rovers, equipped with human-like grasping abilities would be able to operate without prior knowledge of target objects, while maintaining resource-efficiency. Furthermore, the developed platform lays the fundamentals for further research and development in the field of autonomous robotic grasping and object manipulation in extraterrestrial environments, providing valuable contributions for autonomous on-board decision-making in space missions.

The following sections will provide a detailed examination of the methodology, results, and implications of this study. The paper is structured as follows: Section 2 provides an overview of the theoretical foundations of deep learning and robotic grasping. Section 3 describes the methodology used in this study and discusses its key findings. Finally, Section 4 summarises the results and outlines future research directions.

2. THEORETICAL BACKGROUND

This section introduces the basic concepts of deep learning and provides an overview of the robotic grasping problem, aiming to provide a reference for notations and definitions that were used and adapted for elaboration of this study.

2.1. Deep Learning

2.1.1. Introduction to Artificial Neural Networks

Recent advancements in computational power and data availability have led to the emergence of deep learning, a subfield of machine learning (ML) that has revolutionised the field of artificial intelligence (AI). Deep learning has outperformed traditional ML algorithms in fields such as robotics, computer vision, and natural language processing [14, 15], and in some cases, even surpassed human performance [16]. Deep learning involves using Artificial Neural Networks (ANNs), which are computational models inspired by biological neural networks. ANNs can learn hierarchical representations and complex mappings from large amounts of data, without relying on explicit human instructions. ANNs are powerful tools that can approximate any continuous function to a high degree of accuracy with enough neurons [17]. Therefore, ANNs are widely recognised as universal function approximators.

Feed-forward neural networks are a fundamental type of ANN architectures [16]. This architecture consists of an input layer, one or more hidden layers, and an output layer. As shown in Figure 1, the input vector $(x_i)_{1,\dots,n}$ flows through the hidden layers to produce a predicted output vector $(\hat{y}_i)_{1,\dots,n}$ in the output layer. ANNs can be shallow or deep depending on the number of hidden layers. Deeper ANNs have more representational power, making them better suited for solving complex problems. The fundamental building block of a feed-forward neural network, also referred to as perceptron [16], is illustrated in Figure 1. The perceptron consists of a single computational layer, where the input values are fed into a single artificial neuron that processes them to generate the output. Inspired by the inner workings of human brain neurons, each input value received by the perceptron x_i is first multiplied by a corresponding weight value w_i. To account for the invariant part of the prediction that doesn't depend on the input values, a bias term b is added to the sum of the weighted inputs. The resulting quantity is then passed through an activation function σ, ultimately producing the neuron's output.

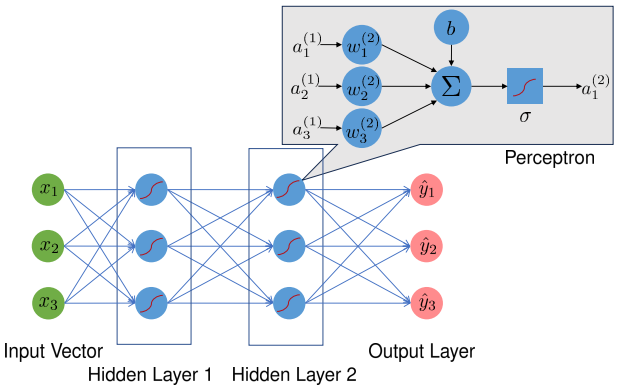


Figure 1. Typical structure of a Feed-forward Neural Network

2.1.2. Training Artificial Neural Networks

The training of Artificial Neural Networks (ANNs) is a complex process that incorporates several key steps. At its core,

training involves using the input data to obtain a prediction from the network and then iteratively adjusting the weights and biases of the neurons to improve the prediction accuracy.

Forward Propagation: in this initial training phase, an input vector is introduced into the network, initiating a series of sequential computations across each layer, where the network computes the output of each layer, using the current weights and biases. Upon reaching the final layer, the output value is computed, representing the ANN's prediction for the given input.

Loss Function: following the forward propagation phase, a predicted output is generated for a specific input vector. In order to evaluate the accuracy of this prediction, a loss function is computed. This function quantifies the discrepancy between the predicted output of the neural network and the actual target value from the training data.

Back-Propagation: this phase primarily focuses on computing the gradient of the loss function with respect to each weight across the network, starting from the output layer and moving backward through the network's layers. Using these gradients, standard optimisation techniques are employed to adjust the weights, aiming to minimise the overall loss.

Optimisation: the process of training neural networks is fundamentally an optimisation task, where the optimisation objective is determined by the loss function. During training, the primary goal is to adjust the network's parameters, mainly its weights and biases, in a way that minimises the loss function. Gradient descent-based optimisation algorithms [18], such as Stochastic Gradient Descent (SGD), Adam, and RMSProp, are the most commonly used optimisation algorithms in neural network training. These algorithms operate by iteratively adjusting the model's parameters in the direction of the steepest descent of the loss function, aiming to find a local or global minimum that ensures optimal network performance.

While the process of training ANNs as outlined above might seem straightforward, it is important to emphasise the inherent complexity of this optimisation task due to the high dimensionality of the parameter space and the potential for complex, non-convex loss functions. Therefore, achieving optimal performance during training presents significant challenges.

Underfitting and Overfitting: underfitting occurs when the model is too simplistic, failing to model and learn the underlying patterns in the data. Such models often have poor performance due to their limited complexity. On the other hand, overfitting arises when the network is too complex and models the training data, including its noise and outliers, too well, resulting in a lack of generalisation. When presented with new data, the network tends to perform poorly. In both cases, the predictive accuracy of the model is compromised. Therefore, it is crucial to have a balanced model that can capture the patterns in the data without sacrificing its ability to generalise. In order to avoid overfitting, regularisation methods [19] are often employed, such as L1 and L2 regularisation, or dropout, to reduce the model's complexity. While L1 and L2 regularisation add a penalty term to the loss function, dropout, on the other hand, randomly drops out neurons during training, forcing the model to be more robust. Additionally, techniques such as early stopping can help to determine the optimal number of epochs needed to train the model without overfitting. This is achieved by evaluating the model's performance on unseen data, using a validation set that is distinct from the training data. By monitoring the model's validation performance during training, overfitting can be prevented by selecting the model that achieves the best generalisation performance on the validation set.

Vanishing and Exploding Gradient: the backpropagation algorithm, responsible for updating the weights in a neural network, relies on the chain rule to compute gradients. However, as these gradients propagate backwards through the network, they may diminish to the point where they become very small, almost zero. This phenomenon, referred to as vanishing gradient, causes the earlier layers of the network to receive minimal updates, which can hinder the learning process. On the other hand, exploding gradients occur when gradients amplify as they propagate through the network layers, becoming excessively large. This results in overly aggressive weight updates that may cause the model to oscillate or diverge, which can destabilise the training process. To address these issues, various techniques can be used, such as using an adaptive learning rate to stabilise the gradient descent process. Additionally, using batch normalisation can help stabilise the gradient descent process by normalising the inputs to each layer. The selection of an appropriate activation function is also crucial. For instance, the vanishing gradient issue is often associated with activation functions like Sigmoid, which have bounded derivatives that can become very small for large input values. However, there are alternative activation functions that can be used to mitigate this issue, such as the rectified linear (ReLU) activation function and its variants, which have proven to be more robust against the vanishing gradient problem due to their larger derivatives.

Hyperparameters Tuning: while the model's internal parameters are adjusted during training, hyperparameters are set before training begins and remain constant throughout. Common hyperparameters include the learning rate, number of hidden layers, and batch size. Finding the optimal set of hyperparameters can significantly improve the model's accuracy, but it can be computationally intensive due to the extensive search over a large parameter space. Techniques for hyperparameter tuning include manual tuning, grid search, random search , and Bayesian optimisation. Grid search and random search are automated techniques that evaluate the model's performance on a pre-defined grid of hyperparameters. Bayesian optimisation is a more advanced technique that models the model's performance as a function of the hyperparameters and uses this model to guide the search towards the optimal set of hyperparameters.

2.1.3. Convolutional Neural Networks

By emulating the hierarchical structure of the visual cortex [20], Convolutional Neural Networks (CNNs) were developed to identify objects and patterns in images. CNNs have revolutionised the field of computer vision, enabling machines to perform tasks that were previously considered beyond their capabilities, such as image classification [21], object detection [22], and semantic segmentation [23]. CNNs have achieved unprecedented success in image classification, outperforming transitional computer vision techniques and even surpassing human capabilities [24]. This success is mainly attributed to their ability to process

image data while preserving its spatial structure, unlike feed-forward neural networks that flatten image data into a one-dimensional vector, ignoring any spatial relationships between the pixels.
Figure 2 displays the basic architecture of CNNs, which typically consists of convolution, pooling, and fully connected layers. The convolution and pooling layers are designed to extract hierarchical features from the input image data, starting with the detection of low-level features such as edges, lines, and curves. As these features are passed from one layer to the next, they are combined to form increasingly complex higher-level features, allowing the network to identify more complex patterns and structures in the image data. The extracted feature maps of the final convolution or pooling layer are then flattened into a one-dimensional vector and passed to a set of fully connected layers. These layers are responsible for mapping the extracted features to the application-specific outputs of the network, such as the probabilities of each class in image classification tasks.

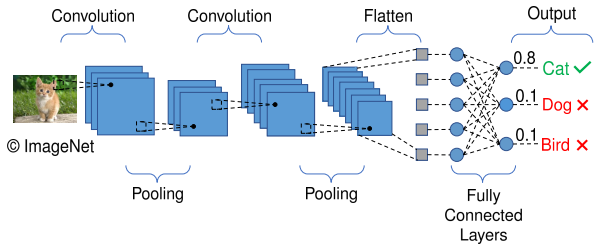


Figure 2. Typical structure of a Convolutional Neural Network

In recent years, several well-known CNN architectures have been developed for image recognition tasks, achieving state-of-the-art results on the ImageNet dataset. AlexNet [25] was the first CNN to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [26]. VGGNet [27] demonstrated that increasing the depth of neural networks can improve their performance, while Inception [28] introduced the inception module, which integrates a network-in-network design to add depth and width while preserving computational efficiency. Introduced in 2015, ResNet [29] distinguished itself by achieving an order of magnitude more layers than prior architectures thanks to the introduction of skip connections to address the problem of vanishing gradients, which had been a major obstacle in training very deep neural networks. Winning the ILSVRC competition in 2015, ResNet was the first deep learning model to achieve human-level performance in image recognition tasks.

2.1.4. Transfer Learning

Transfer learning is a widely used technique in machine learning that modifies a pre-trained model to address new tasks without requiring training from scratch. By leveraging the foundational knowledge obtained during the primary training phase, transfer learning allows the model to adapt to new challenges and tasks with minimal additional training. This approach can save valuable time and resources, especially in cases where a large amount of labelled data may not be accessible for the new task. CNNs pre-trained on extensive datasets like ImageNet [30] are particularly well-suited for transfer learning. Once trained, such networks can process any image dataset, as the early layers are responsible for extracting general features from images, which are useful for a variety of tasks beyond image classification. By replacing or finetuning the last layers of the network, which are more task-specific and responsible for making the final predictions, the pre-trained network can be adapted to the new task with less data and time than training from scratch, while still leveraging the general features learned from the original dataset.

2.2. Robotic Grasping

2.2.1. The Mechanics of Grasping

Grasping is one of the most fundamental capabilities in robotics, as it enables robots to interact with objects in their environment. To design effective grasping strategies, a deep understanding of the underlying mechanics is essential [31], including the role of friction, as it affects a robot's ability to securely hold an object and prevent it from slipping or dropping. The Coulomb friction model is a basic framework for understanding the frictional interactions between two solid surfaces. According to this model, the frictional force, which is the force resisting the motion between two surfaces in contact, is directly proportional to the normal force pressing them together, with the factor of proportionality being the friction coefficient μ and the direction of this frictional force always opposing potential or actual motion. Within this model, friction is categorised into two main types: static and kinetic friction [32]. Static friction is the force that prevents an object from immediately moving when a force is applied to it. This resistance adjusts to match the applied force, up to a certain threshold, which is determined by the product of the coefficient of static friction μ_s and the normal force. Once the applied force exceeds this value, the object begins to move and kinetic friction comes into play. Unlike static friction, kinetic friction has a constant magnitude, given by the product of the coefficient of kinetic friction μ_k and the normal force. Typically, μ_k is lower than μ_s, meaning that less force is required to maintain an object's motion than to initiate it. Another fundamental concept is the frictional cone, which describes the range of forces at a contact point. In a robotic grasping scenario, where a gripper's fingertip comes into contact with an object, the primary force pressing these surfaces together is the normal force. However, friction introduces additional forces that can act tangentially to this contact point. These tangential forces, when visualised in relation to the normal force, form the frictional cone, which essentially represents all permissible forces that can be exerted to an object without causing it to slip.

2.2.2. Robotic Grasping Methods

This section provides an overview of methods used in robotic grasping, with a particular focus on planar grasping techniques using a parallel-jaw grippers, which are relevant to this paper.
Foundational robotic grasping approaches include analytical techniques [33], which rely on mathematical and physical modeling and require prior knowledge of the geometric properties of objects being grasped. Over time, the research community has shifted towards data-driven techniques [34],

focusing on learning from real or simulated data to predict effective grasps.

Deep learning-based methods have revolutionised the field of robotic grasping, enabling the development of more effective grasping systems, by learning directly from raw sensor data. This shift towards deep learning for grasping was enhanced by the availability of large publicly accessible datasets, which can be used to train and evaluate deep learning models. The following provides an overview of some datasets used for parallel-jaw grasping that are particularly relevant to this study.

Cornell Grasp Dataset [35]: developed in 2010 by researchers at Cornell University, this dataset is widely used for robotic grasping, containing a total of 885 real RGB-D images of 240 different objects as well as 8019 hand-labelled grasp rectangles.

Jacquard Dataset [36]: another popular dataset for robotic grasping is the Jacquard grasping dataset, which contains 54,000 RGB-D images of 11,000 objects, along with their objects' segmentation masks and over 1 million annotations of successful grasps from simulated attempts.

GraspNet 1 Billion [37]: introduced in 2020, this dataset is currently the largest dataset available for robotic grasping, providing over 1 billion annotated grasps for more than 95,000 real RGB-D images. Unlike the two other datasets that only cover planar grasps, GraspNet 1 Billion is known for its comprehensive grasp annotations, which include 6-DOF grasps.

Based on their output type, deep learning-based grasping methods can be classified into two main categories: classification-based and regression-based. While classification-based methods aim to classify input data into pre-defined categories, regression-based methods aim to predict a numerical value based on input data.

Classification-based methods: Lenz et al. [38] were among the first to apply deep learning to the problem of robotic grasping. Eliminating the need for manual features engineering, the authors introduce a two-step cascaded system, where the first network generates candidate grasps, and the second network evaluates and ranks them. Pinto et. [39] used a CNN-based classifier to evaluate each image patch, which represents a potential grasp-oriented rectangle, for its likelihood of being a successful grasp location. Using multiple-stage spatial transformer networks (STN), Park and Chun [40] introduced a novel classification-based approach for robotic grasp detection, which allowed for partial observation of intermediate steps in the grasp detection process, making the model more transparent and interpretable.

Regression-based methods: Redmon et al. [41] introduced a real-time approach for robotic grasp detection of novel objects using CNNs, specifically AlexNet for features extraction. Their single-stage regression-based approach enables the prediction of grasping rectangles directly from RGB-D images, eliminating the need for standard sliding window or region proposal techniques. Evaluated on the Cornell Grasp Detection Dataset, this approach achieved an accuracy of 88%. In a related study, Kumra and Kanan [42] used a deeper neural network architecture, namely ResNet50, for features extraction, achieving a higher accuracy of 89.21% on the Cornell Grasp Dataset. Focusing on multi-modal fusion, Zhang et al. [43] propose a CNN-based approach that combines RGB features and depth features to enhance grasp detection accuracy. Instead of predicting the grasp parameters for the entire image, some methods use a Region of Interest (ROI)-based or pixel-wise approach, which focuses on specific regions or pixels within the image. Once the ROIs are identified, the grasp parameters are predicted based on the features extracted from these ROIs. Applying this approach, Zhang et al. [4] achieved impressive success rates of 92.5% and 83.8% in single-object and multi-object scenes, respectively.

Reinforcement learning (RL) techniques are gaining popularity in the field of robotic grasping [44]. Using this approach, robots can learn effective grasping policies through trial and error. By receiving feedback from their interactions with the environment, robots are also able to improve their performance over time.

3. METHODOLOGY

This section outlines the methodology used in this study starting with the problem statement, followed by describing the custom dataset generation pipeline, and concluding with the training procedure.

3.1. Problem Statement

3.1.1. Previous Work at the Space Robotics Lab

This study builds upon prior research conducted by the Space Robotics Lab at the University of Stuttgart. The lab has significantly contributed to the field of space robotics through various projects. These include designing and operating miniaturised robotic systems for space exploration. In a cooperative technology development project, the Nanokhod Microrover is prepared for a long-term Lunar Surface Application [45]. A further focus of the lab is developing sensory components and software to facilitate autonomous operation of such systems. One of the lab's technology development platform is the Modular Rover Chassis Platform (MRCP) [12, 13], which serves as the basis for this study. The MRCP, a reconfigurable platform designed for adaptability and superior locomotion performance, is equipped with modular interfaces that facilitate integration of additional subsystems and payloads.

Within an interdisciplinary education program, several in-house developed robotic subsystems were successfully integrated and operated into the MRCP [46]. Relevant for this study, are a 5-DOF robotic arm and a 2-DOF camera mast, which expand the platform's capabilities to perform complex tasks such as sample collection and instrument handling in space environments. The MRCP's capabilities were demonstrated during a sample return challenge, where the platform showed promising results in terms of its performance in collecting samples and storing them in a dispenser, as depicted in Figure 3. However, as the first technology demonstrator, the current control architecture of the robotic arm is not developed for fully autonomous operation. Therefore, throughout the experiments, the arm was controlled manually.

3.1.2. Proposed Approach

This study aims to improve the capabilities of robotic systems by enabling it to perform grasping tasks autonomously

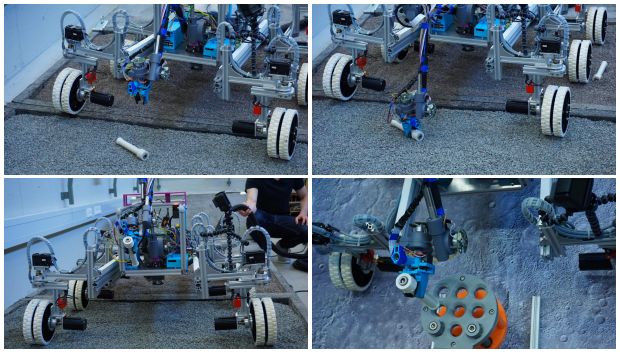
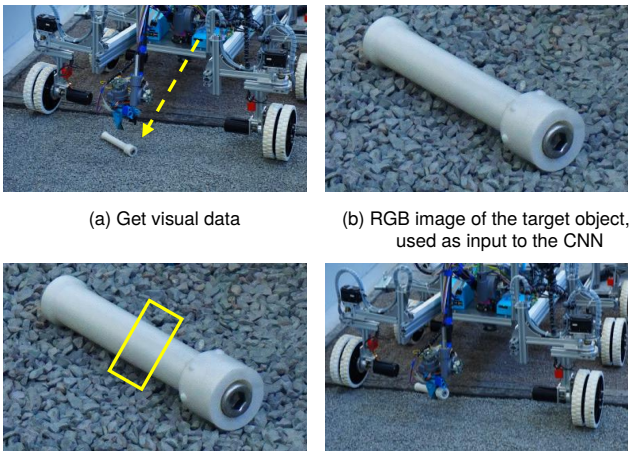


Figure 3. Overview of the sample return challenge conducted at the Space Robotics Lab of the University of Stuttgart using the Modular Rover Chassis Platform (MRCP) [47]



(a) Get visual data

(b) RGB image of the target object, used as input to the CNN

(c) Grasp estimation using the offline trained CNN

(d) Grasp execution

Figure 4. Illustration of the proposed approach for the online sample collection process

using deep learning algorithms while taking the limited resources available for space systems into account. Autonomy is a crucial capability in space robotics as it allows for efficient use of resources, including time, power, and cost savings, while also providing higher scientific returns, particularly in advanced mission scenarios where real-time teleoperation may be inefficient or unreliable. The study introduces a resource-efficient end-to-end grasp estimation system, designed for rovers equipped with a robotic arm with a parallel gripper. By implementing a new framework, this study aims to lay the groundwork for future innovations in autonomous grasping under extraterrestrial conditions.

Existing robotic systems often struggle in uncertain and unknown environments, necessitating an approach that can handle novel objects. To this end, the proposed system is trained using deep learning techniques, enabling the rover to identify and grasp objects autonomously based solely on the visual information provided by its onboard sensors, thereby eliminating the need for any prior information about the objects being grasped. To ensure the system's resource efficiency, the computationally intensive training process is conducted on Earth. Once trained, the resulting models can be used in combination with the sensors on-board to accurately identify objects and infer the best grasp configuration. Once the grasp estimation is completed, conventional inverse kinematics methods can be employed to execute the computed grasps. The entire process is shown in Figure 4.

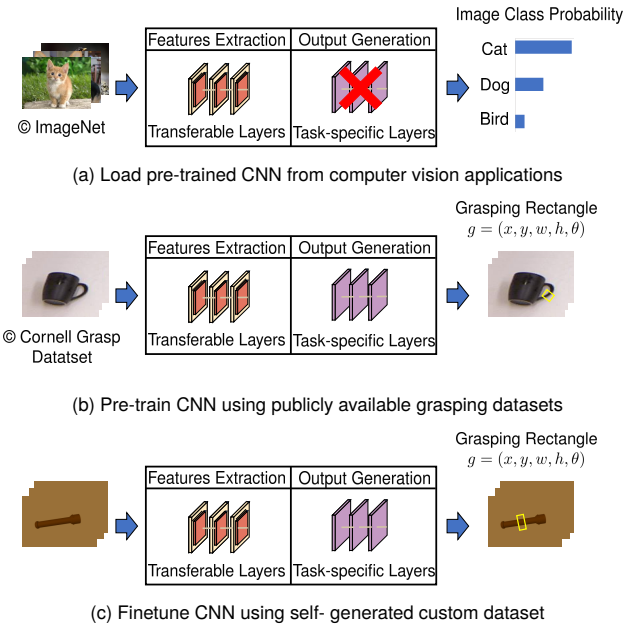
While deep learning techniques have proved their efficiency in terrestrial grasping tasks, applying them to extraterrestrial environments is limited by the lack of publicly available, labelled grasping datasets containing objects relevant to space missions. To address this challenge, this study proposes the use of transfer learning, to exploit off-the-shelf pre-trained CNNs from computer vision applications. These CNNs, having been previously trained on large-scale datasets such as ImageNet [30], have acquired the ability to recognise a wide range of objects with high accuracy, thus developing essential capabilities such as feature extraction and pattern recognition from images. Due to their hierarchical learning process, these models are able to identify lower-level features such as edges, textures, and shapes in the early convolutional layers, and higher-level features like object classes in the deeper layers.

To adapt these models to the task of robotic grasping, the first layers, responsible for learning universal features, are retained, while the last layers are replaced with new ones dedicated to learning the grasping task. These new layers use the low-level features learned by the model's initial convolutional layers to infer the optimal grasp rectangles based on the input RGB image. The next stage involves training the adapted models using publicly available robotic grasping datasets. This process provides the models with a foundational understanding of the grasping task, by allowing them to gain preliminary experience with objects with a wide variety of shapes, sizes, and textures, thereby enhancing their ability to grasp novel objects. These models are then fine-tuned using a self-generated dataset containing objects relevant to manipulation tasks in space exploration missions. This process further refines the grasping abilities of the models, ensuring their efficiency in handling objects likely to be encountered in extraterrestrial environments. This training procedure in shown in Figure 5 and is further described in Section 3.3.

To bridge the gap between space exploration and deep learning research, this study also proposes a pipeline to automatically generate a large, labelled dataset of objects suitable for the task of autonomous grasping in planetary exploration missions. The dataset contains objects that the rover is likely to encounter in an extraterrestrial environment, such as sample tubes critical for sample return missions, and maintenance tools required for maintenance and servicing tasks. A 3D planetary robot simulation environment was developed as a platform for generating and automatically labelling synthetic custom data. This environment, taking extraterrestrial factors into account, including aspects such as lighting, gravity, and terrain, can also be used to evaluate the performance of the trained deep learning models by executing the grasps generated by these models in a simulated planetary setting.

3.1.3. Grasp Representation

This research primarily focuses on planar grasping due to the abundance of public datasets available for this problem. Moreover, restricting the gripper movements to a two-dimensional plane considerably reduces the complexity

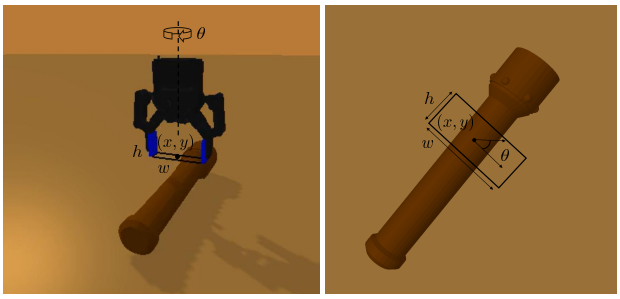


(a) Load pre-trained CNN from computer vision applications

(b) Pre-train CNN using publicly available grasping datasets

(c) Finetune CNN using self- generated custom dataset

Figure 5. Illustration of the training procedure for an end-to-end grasp estimation system



(a) (b)

Figure 6. Illustration of the grasp representation $g = (x, y, w, h, \theta)$ in the 3D simulation environment (a) and in the image plane (b)

of the problem, thereby enabling a more efficient training process. Nevertheless, the proposed dataset generation pipeline is capable of generating data for both planar and 6-DOF grasping scenarios, allowing for the future expansion of this research into more complex grasping settings.

In this paper, the task of robotic grasping is addressed as a regression problem, where the aim is to identify a grasping rectangle within the image plane, representing the optimal grasp pose for a given object, as illustrated in Figure 6. The grasping rectangle is characterised by its centre (x, y), width w and height h in image coordinates, as well as its orientation θ. This 5-dimensional grasp rectangle, introduced by [35] serves as a compact representation of the full 7-dimensional parallel-jaw gripper's configuration used by the robot to perform a grasp and consisting of the 3D position, 3D orientation, as well as the opening width of the gripper. To obtain the 3D position, the centre of the grasping rectangle can be used along with the depth information obtained from the RGB-D data on-board to compute the grasp depth or the z-position of the gripper. The 3D orientation is determined by the three Euler angles, but for planar or 4-DOF grasping, certain gripper movements are restricted. As the gripper approaches objects from above, two orientation angles are set to zero, leaving only the rotation angle around the z-axis θ. Finally, the width of the rectangle is used to compute the gripper's opening width.

3.2. Custom Dataset Generation

3.2.1. Dataset Creation Techniques

Deep learning approaches rely heavily on the concept of data-driven learning, which requires a significant amount of high-quality labelled data to train deep learning models. In recent years, several large labelled grasping datasets have become publicly available [35–37], primarily focusing on terrestrial robotics applications and containing objects such as tools, office supplies, and household items. However, these objects are not representative of those encountered by a rover in an extraterrestrial planetary environment, making them less suitable for space robotics applications. To address this limitation, it is essential to create a more appropriate dataset that includes a diverse range of objects relevant to planetary exploration missions. While generating images of objects of interest is relatively straightforward, the challenge lies in accurately labelling these images, which can be achieved through one of the following three techniques [36].

Manual labelling: this technique relies on human expertise and involves human annotators manually and marking grasping rectangles on images, representing potential ways to grasp the object. While this approach benefits from human intuition about object properties and grasp dynamics, it can be time-consuming. As the variety of objects increases, the manual approach becomes less efficient for creating a comprehensive dataset, which limits the scalability of the process. Furthermore, the lack of a unified standard for determining optimal grasp configurations can lead to inconsistencies among annotators who may have different opinions of optimal grasps, resulting in discrepancies in dataset quality.

Real-robot experiments: this approach involves using a physical robot to attempt grasping real objects in various ways. Successful attempts are added to the dataset, resulting in an accurate representation of the complexities of the real world. The resulting dataset provides precise data that takes into account the challenges and uncertainties inherent in real-world situations. However, creating a comprehensive dataset requires significant resources, including various objects and multiple robots. Additionally, the continuous operation of robots can lead to increased wear, resulting in higher maintenance costs and potential operational downtime, making it time-consuming and expensive. Furthermore, human intervention may still be needed to determine whether a grasp attempt is successful or not, as the robot's sensors might not be able to accurately evaluate the stability and effectiveness of the grasp.

Simulation-based labelling: simulation-based techniques are not only efficient, but also cost-effective for generating large datasets for robotic grasping without human intervention. These techniques use computer models of various objects with different shapes and sizes, which can be integrated into a simulated environment where a virtual

robot attempts to grasp them. Successful grasps can then be recorded and used to train deep learning models. This approach offers great scalability and flexibility, allowing researchers to generate large-scale datasets with a wide range of variables that can be expanded as needed, which would be time-consuming and resource-intensive to achieve through manual labelling or real-robot experiments. The Jacquard dataset [36] provides a great example of the potential of simulation-based techniques, containing over 50,000 images of 11,000 unique objects and over 1 million annotated successful grasps.
Driven by the need for a scalable, efficient, and cost-effective method, a simulation-based approach was adopted. Several physics simulators were considered, but ultimately PyBullet [48] was selected due to its flexibility, performance, and ease of use. PyBullet is an open-source physics engine with a Python API, making it well-suited for robotics and machine learning applications. One of PyBullet's key features is its synthetic camera integration, which can generate RGB-D images, crucial for producing object images within the simulation.

3.2.2. Computational Pipeline

To ensure the relevance and accuracy of the dataset, a custom 3D simulation environment has been developed to simulate extraterrestrial conditions such as reduced gravity and various terrain types. This section introduces the computational pipeline used to generate a large custom dataset for space robotics applications. The dataset generation process consists of multiple steps, which are illustrated in Figure 7.
3D object models: as previously mentioned, the main objective of future planetary exploration missions is to enable rovers to perform complex robotic tasks autonomously, such as sample collection or maintenance and servicing tasks, without any human intervention. To accomplish these tasks, rovers must be capable of handling specific objects that align with these mission objectives, such as sample tubes and maintenance tools. However, accurate interaction with these mission-critical objects in a 3D simulation environment requires precise 3D object models. For sample tubes, the primary source of 3D models is a generic CAD model developed at the Space Robotics labs. Based on this model, a more flexible parametric model was developed in Siemens NX. This allows the manipulation of certain design parameters, such as tube length and diameter, to create a wide range of CAD models with different configurations. By programmatically adjusting these parameters, various sample tubes with different characteristics can be generated, which can be used to simulate many grasping scenarios. On the other hand, to ensure that the simulation environment has a diverse and realistic set of objects for maintenance and servicing tasks, maintenance tools are selected from publicly available datasets, primarily the YCB-Dataset [49] and the 3DNet-dataset [50], which offer a comprehensive collection of 3D models of household objects, including maintenance tools.
Automatic URDF Generation: 3D object models are often represented as STL or OBJ files, which contain all the necessary geometric information about the object. However, physics engines, such as PyBullet, only work with URDF files, which include both the geometric and physical properties of the object. To convert 3D object models to URDF files, an automatic tool has been developed in Python. First, the 3D object model is scaled to the desired size, using a defined scale factor. This factor is computed based on the field of view and far value of the camera to ensure that the object fits comfortably within the view and contributes to a realistic representation of the scene. With the object properly scaled, the next step is to create a visual mesh. This mesh serves as the object's graphical representation within the simulation environment, providing all the surface details

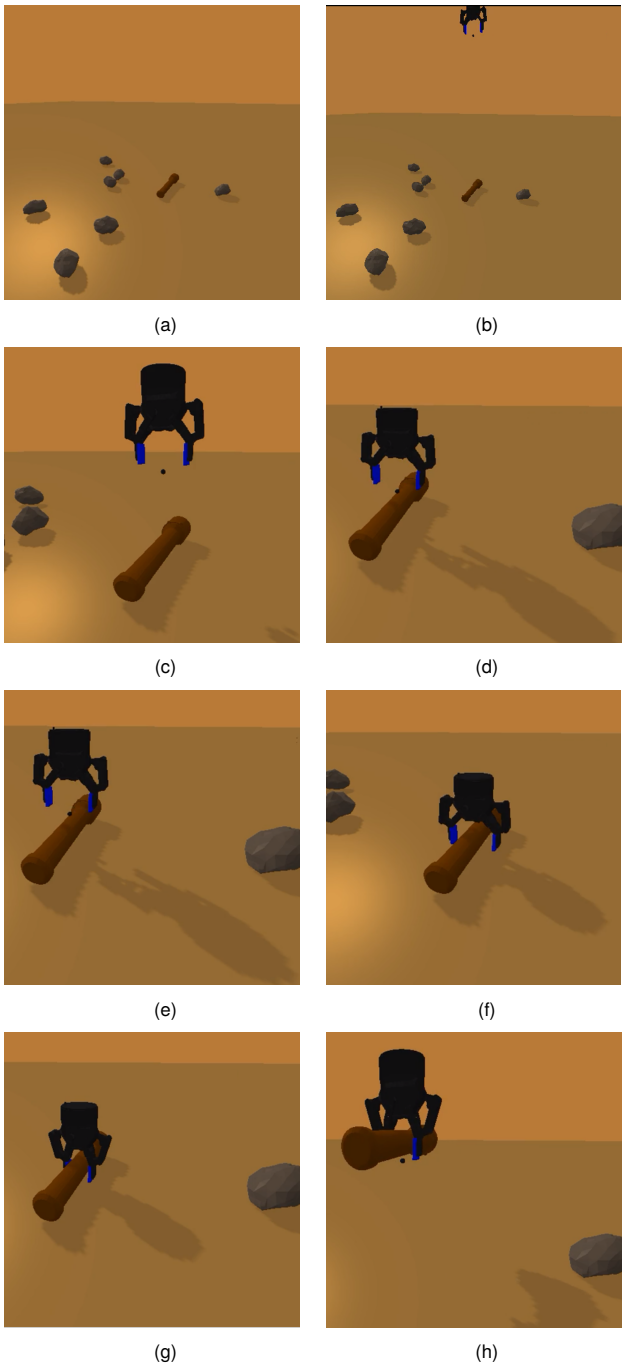


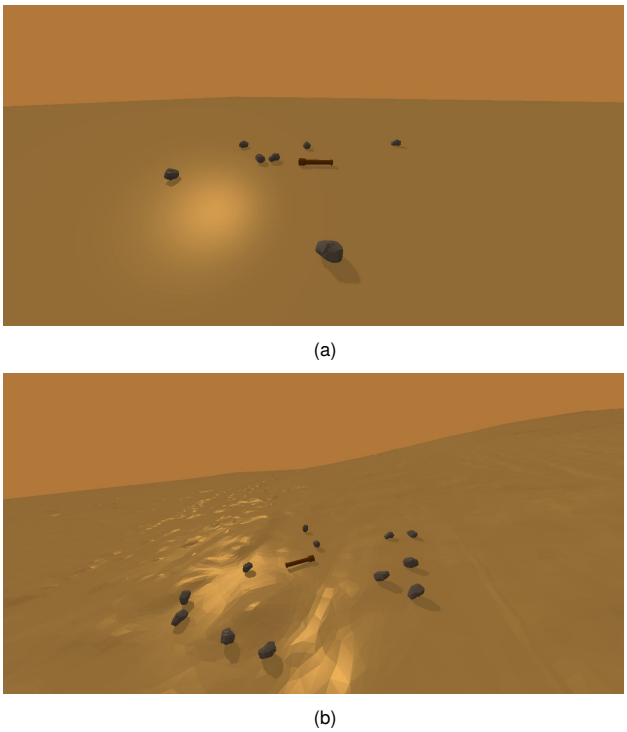
Figure 7. Illustration of the dataset generation pipeline, starting with scene creation (a,b,c), followed by Monte-Carlo grasp labels generation (d,e) and concluding with grasp attempt (f,g) and outcome (f)

necessary for the rendering engine to draw the object on the screen. Simultaneously, a collision mesh is generated to handle the physics-based interactions between the object and the robotic gripper. Collision meshes are simplified versions of the visual mesh, designed to reduce the computational load during physics simulations. However, this simplification often leads to inaccurate collision detection, particularly for objects with complex or concave surfaces. To overcome this issue, the V-HACD algorithm [51] is used to generate an optimised collision mesh that accurately represents the object's surface. The V-HACD algorithm works by dividing the object into smaller convex shapes, which are then reassembled into a simplified mesh, while maintaining the object's original shape as much as possible. The final step involves the computation of mass and inertia using the trimesh library [52], which enables accurate computation of these properties based on the object's mesh and its density.

Gripper Model: inspired by the European Space Agency's (ESA) Analog-1 project [53], the Robotiq85 gripper, displayed in Figure 6, is used in this paper as an exemplary end-effector model. Focusing only on the interaction between the gripper and the object, dummy links and joints are used in the gripper URDF file to represent the rest of the robotic arm. This approach allows the gripper to "float" in the simulation, eliminating the need for additional inverse kinematic computations, thereby reducing the computational complexity of the simulation.

Scene Creation: the scene creation process involves several steps, starting with setting up the simulation environment in which the gripper will interact with various objects. To create a realistic simulation environment, the gravity is set to match that of Mars, which is approximately 38% of Earth's gravity. After that, a plane is loaded as the base surface for the simulation. To increase the realism of the landscape for the gripper to interact with, textured planes or terrain models generated based on Mars topography data can be used instead of basic flat surfaces. The user can choose between using a textured plane or the path of the Curiosity rover, which can be obtained from NASA's 3D resources [54]. Next, the selected object's 3D model is loaded in STL or OBJ format, converted to URDF files using the conversion tool described and added to the scene. The object is then dropped from a random position and orientation, and additional elements such as rocks can be loaded into the simulation for a more challenging and realistic environment for the robotic grasping algorithm, as shown in Figure 7a. Once the scene is set up, RGB-D images of the object are captured, which will then serve as input to the neural network that powers the robotic grasping algorithm. Furthermore, a hash function is used to create a unique ID for each scene. This ID is then used to name the image files, simplifying their association with the specific object and simulation they represent. Finally, the robotic gripper is loaded into the simulation, moved to a user-defined start position (see Figures 7b and 7c) and the simulation loop is executed. During each iteration, the robotic gripper attempts to grasp the object and records the outcome of each attempt.

RGB-D Perception: during the simulation, the camera is positioned above the object to capture the entire object within its field of view, providing comprehensive visual data for the subsequent grasping operation. The synthetic camera emulates the same camera system which is currently being used on the robotic camera mast at the space robotics lab. The camera model, which is implemented using OpenGL [55], involves transformations between multiple coordinate systems. These transformations are used to convert the gripper pose between camera coordinates and world coordinates, and vice versa. Additionally, the depth buffer values provided by the synthetic camera are transformed into meaningful depth information. Once completed, RGB and depth images are saved along with segmentation masks that separate the object from the background.



(a)

(b)

Figure 8. Examples of scene creation in the 3D simulation environment illustrating a flat surface (a) and a terrain (b) that replicates the path of the Curiosity rover, obtained from NASA's 3D resources [54]

Grasp Labels Generation: the robotic grasping process can be executed in two distinct modes: manual and automatic. In manual mode, the user has direct control over the gripper's movements and actions, which is particularly useful for testing and debugging purposes. In automatic mode, the gripper's movements are controlled automatically to generate random grasps by using a Monte Carlo approach. The robotic gripper attempts multiple grasps on each object from different angles and orientations. Each grasp is evaluated based on pre-defined success criteria. Successful grasps are then automatically added to the dataset, while unsuccessful ones are discarded.

Monte-Carlo grasps generation: first, the gripper's width w is sampled from a uniform distribution ranging from its minimum to maximum opening width. In order to generate grasp candidates in the most promising areas, a pixel (x, y) within the object's bounding box is sampled from a normal distribution. Using the depth value at the selected pixel, the target gripper position in world coordinates can be computed, and the gripper is moved to that position, as illustrated in Figure 7d. Next, the yaw rotation angle of the gripper θ is sampled from a uniform distribution ranging from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ and the gripper is rotated accordingly (see Figure 7e).

Finally, the grasp is executed by moving the gripper downwards until it either makes contact with the object or reaches a pre-defined z-position, as displayed in Figure 7f. The gripper then closes and returns to its start position (see Figures 7g and 7h). The grasp is considered successful, if the object is still held within the gripper's finger pads. The gripper jaw size h, which remains constant across each grasp attempt, is then mapped to image coordinates. The resulting grasp label represented by (x, y, w, h, θ), as well as the object's images captured at the beginning of the simulation, are then stored in the corresponding folder based on the outcome of the grasp attempt.

Automatic Dataset Generation: for the automatic dataset generation, the simulation is executed in the direct mode, which is one of two modes offered by the PyBullet physics engine. The other mode is called GUI Mode, which provides a graphical interface to visualise the simulation in real-time. In direct mode, commands are executed in the background without the need for a graphical interface, enabling parallel computation on a cluster and reducing computational demands by redirecting resources from rendering graphics towards the simulation. To further accelerate the dataset generation process, parallelisation is used in direct mode. This involves replicating the simulation across all available CPU cores. In each replication, the same object is dropped from a different position and orientation at the start of the simulation. Alternatively, the user can choose different objects for each replication. The simulations are run on an available in-house cluster, which has 128GB RAM and is powered by 2 NUMA nodes, where each node has 8 Intel Xeon CPUs, with 2 threads per core. In order to ensure the smooth operation of the simulation, a comprehensive logging system is implemented, documenting each step, along with any errors or warnings that arise during the dataset generation process. Additionally, a YAML configuration file is included, which allows the user to adjust various parameters without having to delve into the underlying code, making the simulation even more user-friendly while providing precise control over the dataset generation process.

3.2.3. Results & Discussion

The custom simulation environment has been successfully implemented to replicate extraterrestrial conditions. Initial runs of the simulation demonstrated its ability to emulate the physics and dynamics of grasping tasks in such environments. However, during the dataset generation process, computational challenges were encountered despite parallelisation being implemented and accelerating the process. Generating a large dataset within a reasonable time frame requires significant computational power, which was not fully provided by the available cluster, thereby limiting the size of the dataset that could be generated.

Despite these computational limitations, a preliminary dataset was generated that serves as a proof of concept, evaluating the pipeline's efficacy. A simple textured plane served as the base surface for the simulation environment in this dataset. Future datasets can benefit from integrating more complex terrains, particularly those derived from Mars topography data. While a diverse range of objects was available for grasping, this preliminary dataset focused on sample tubes due to their inherent presence and significance in recent planetary exploration missions with grasping applications. A broader range of objects can be included in subsequent datasets to increase the diversity and robustness of the training data.

Figure 9 illustrates a portion of the preliminary dataset. The simulation-based approach used to generate grasp labels proved efficient, with the grasp labels consistently aligning with the visual data. Grasp attempts were randomly generated, as described above, resulting in multiple successful grasps for each object instance, as shown in Figure 9. Each image is accompanied by comprehensive data detailing the object's characteristics, grasp parameters, and the simulated environmental conditions during the grasp attempts. This dataset creation pipeline demonstrated its ability to generate a rich, scalable, large dataset suitable for training deep learning models to address the challenges of robotic grasping in space exploration missions.

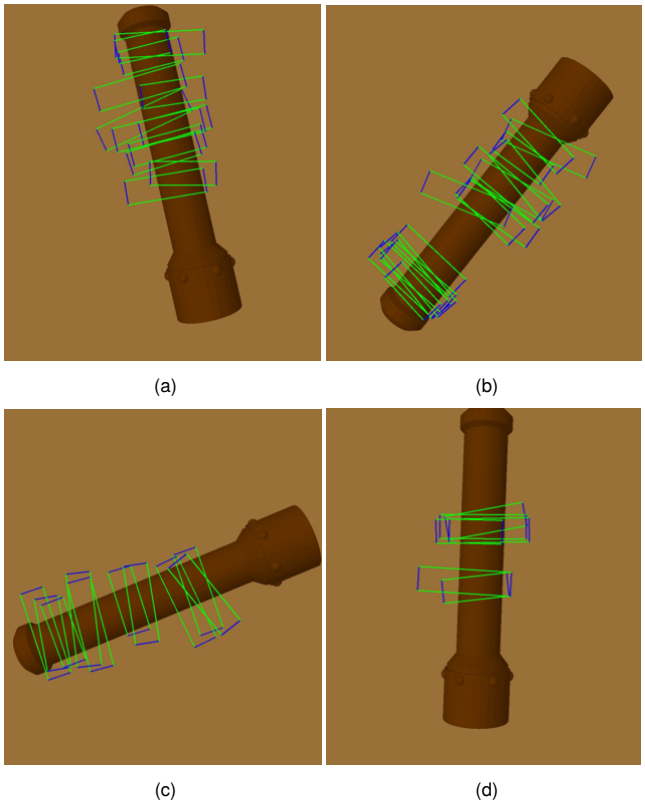


Figure 9. Examples of RGB images from the preliminary dataset along with their successful grasp labels

3.3. Training Procedure

3.3.1. Data Pre-processing

The aim of this study is to harness the power of publicly available robotic grasping datasets for pre-training the deep learning models before proceeding with the training phase on the custom-generated dataset. This study focuses on public datasets compatible with parallel-jaw grippers, including the Cornell Grasp dataset [35], the Jacquard dataset [36], and the GraspNet 1 Billion dataset [37]. These datasets come in various formats, requiring a comprehensive data pre-processing pipeline to standardise them into a unified format for effective model training. In the case of

the Cornell Grasp dataset, each image is accompanied by a separate background file. Similarly, the Jacquard dataset includes an object mask for each image, which separates the object from its background. The custom-generated dataset, which was created using a similar method as the Jacquard dataset, includes these objects masks as well. These features were used to allow the user to either include or exclude the background in the pre-processing pipeline for these datasets. However, the GraspNet 1 Billion dataset is different, as it includes multiple objects within each scene. Focusing on single-object grasping, the segmentation masks provided within the dataset were used to generate new segmented images for each object. Multiple images are generated from each original image, with each new image containing a single object with the background subtracted. Figure 10 provides illustrative examples from all four datasets used in this study.

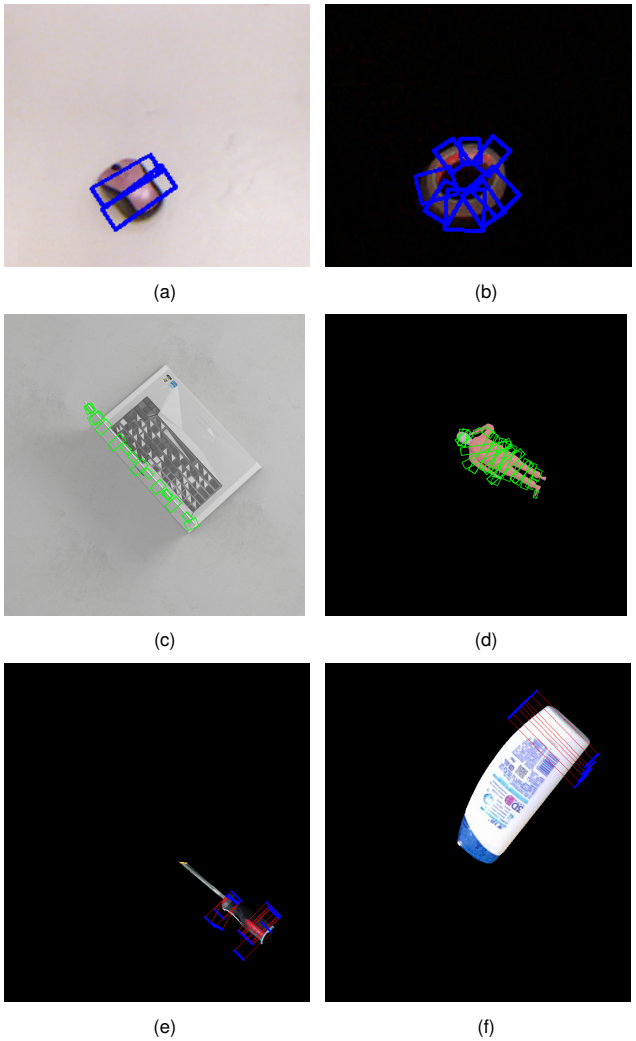


Figure 10. Examples of images from various datasets: Cornell Grasp Dataset [35] (a,b), Jacquard Dataset [36] (c,d), and GraspNet 1 Billion [37] (e,f). Some images are displayed with their original background, while others are shown after background substraction. Each image is accompanied by its labelled grasps

To ensure a consistent annotation across the different datasets, the five-parameter grasp representation introduced in Section 3.1.3 was implemented, standardising the representation of grasp rectangles throughout the datasets and ensuring interoperability and comparability between them. Furthermore, the pre-processing pipeline is designed to be versatile, offering the possibility to convert original RGB images into different colour spaces, such as YUV and HSV, thereby enabling a comprehensive evaluation of different input modalities. To enhance the robustness of the deep learning models to input data variations, a wide range of data augmentation techniques was employed across all datasets. These techniques include, but are not limited to, geometric transformations like rotation, scaling, and translation, as well as colour transformations such as brightness and contrast variation, as shown in Figure 11. For the synthetic data, Gaussian noise was added to the depth images in order to emulate real-world conditions. This increased the diversity and complexity of the data, resulting in more effective models with better generalisation capabilities.

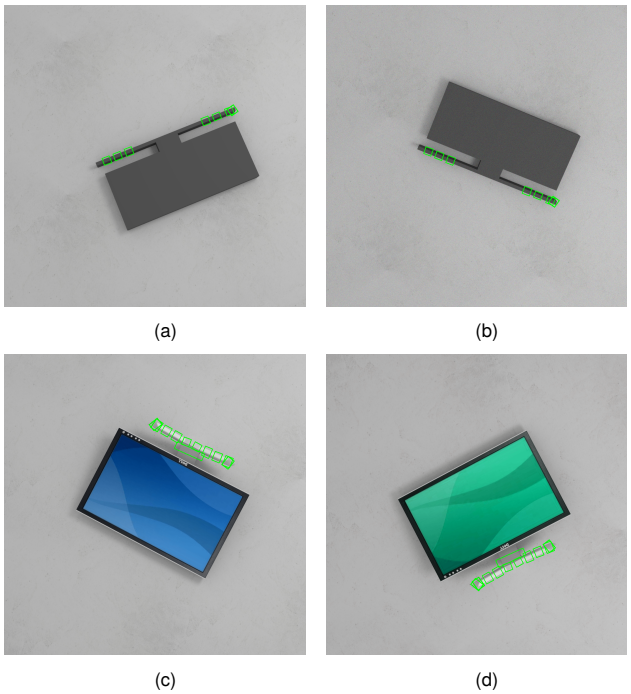


Figure 11. Examples of data augmentation techniques applied to images from Jacquard Dataset [36], displaying pairs of both the original (a,c) and augmented versions (b,d)

3.3.2. Training and Validation

To address the challenges associated with the robotic grasping tasks in planetary exploration missions, this paper proposes a two-phase training strategy that combines pre-training on public grasping datasets with finetuning on a custom-generated dataset. By leveraging the power of transfer learning through this hybrid training approach, the model is able to learn to grasp a wide range of objects, while adapting to the specific conditions of extraterrestrial environments. The training procedure starts by splitting the dataset into training and validation subsets to allow for best model selection based on the validation performance, thereby enhancing the network's generalisation

performance and preventing overfitting. In the first training phase, the model is pre-trained on publicly available grasping datasets, which include a wide range of objects with various shapes, sizes, and textures. This phase provides the model with a robust foundational understanding of the grasping task, which helps minimise the amount of labelled data required for subsequent finetuning. The pre-training stage is further divided into two subphases to optimise both model performance and computational efficiency. During the first subphase, only the final layer responsible for grasp prediction is trained, while the convolutional layers of the pre-trained ResNet, responsible for feature extraction from RGB images, are frozen. By freezing these layers, the pre-trained weights are preserved along with the generalisable features that these layers have learned from extensive training on the ImageNet dataset. Next, the entire CNN is trained end-to-end, starting from the best-performing model obtained in the previous step, in order to further enhance its performance. Finally, the model is finetuned on the custom-generated dataset during the second training phase, resulting in a model that is able to adapt to the specific challenges of robotic grasping in space exploration missions. In the following, details regarding the training process and the different components of the neural network will be provided.

Network architecture: the neural network used in this study is based on the ResNet-50 model, which is well-known for its state-of-the-art performance in image classification tasks, as discussed in Section 2.1.3. With fewer trainable parameters than other CNNs with similar performance, ResNet provides a balance between accuracy and computational efficiency, making it an ideal choice for this study. As shown in Figure 13, the ResNet-50 model in the architecture used in this paper serves as the backbone for feature extraction. Initially trained on the extensive ImageNet dataset, ResNet offers robust feature extraction capabilities that are highly transferable across various tasks. To adapt ResNet-50 for the specific use-case of robotic grasping, the last fully-connected layers, originally designed for object classification tasks, were replaced with new layers designed for grasp prediction, which take the extracted features as input and output optimal grasping parameters.

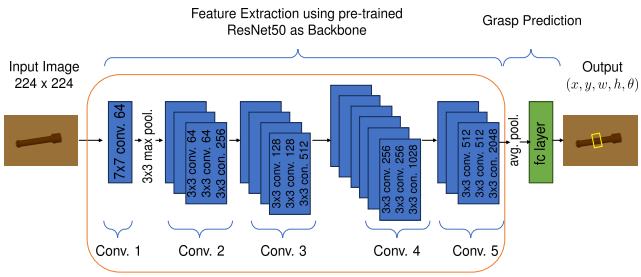


Figure 12. Illustration of the modified ResNet architecture

Loss function: due to the complex nature of the grasp prediction task, the design of the loss function is crucial to the effective training of the neural network. In this work, the Euclidean loss function is used to measure the difference between predicted and actual values for each component of the grasping output. Rather than computing the loss on entire vectors, each component of the grasp is considered separately, enabling the model to learn to predict each variable independently. This component-wise approach is particularly important due to the inherently different nature of the variables involved. While x, y, w, and h correspond to spatial dimensions, θ represents angular orientation. The Euclidean loss functions ensures that larger deviations from the target values have a large impact on the total loss. However, this approach presents a significant challenge for the treatment of the angle component θ, which exhibits small absolute variations. The model may neglect its importance while trying to minimise errors in other parameters with larger absolute deviations. To address this issue, a scaling factor is introduced within the loss function to adjust the contribution of the angular component to the total loss, ensuring that the model pays sufficient attention to accurately predicting the grasp angle, despite the small absolute values.

Activation function: in this study, the architecture of a standard ResNet model was modified by replacing its last layer, which required a reevaluation of the activation function choice for this layer. Various activation functions were considered for this last layer, including bounded ones like Tanh and Softmax. Despite their popularity, these functions proved to be suboptimal, as they led to vanishing gradient issues during backpropagation, which severely limited the model's learning ability. Based on these findings, a linear activation function was selected for the last layer, which helped maintain a stable gradient flow during backpropagation, thereby improving the model's learning capacity. Additionally, the linear activation function is inherently well-suited for regression tasks, as it allows for a continuous range of output values. However, using a linear activation function results in unbounded output values, presenting a significant challenge for the robotic grasping task, as the output values are inherently limited by physical and sensor constraints. In particular, the centre point coordinates (x, y) and dimensions (w, h) of the grasping rectangle must be constrained to pixel values that lie within a specific range determined by the camera's resolution. Furthermore, the orientation angle θ should be limited to the range of $-\pi/2$ to $\pi/2$ in order to ensure a physically realistic and interpretable output. To address this issue, a post-processing step in introduced, consisting in a clipping operation on the network's output. This clipping step constrains the network's output to a pre-defined acceptable range, thereby guaranteeing its relevance applicability for the robotic grasping task.

Learning rate schedule: achieving optimal model performance is a challenging task that depends on various factors, including the learning rate, which controls the magnitude of updates applied to the model's weights during training. Setting the learning rate too high can cause the model to overshoot the optimal solution, leading to oscillations and potential divergence. On the other hand, a learning rate set too low could cause the model to converge slowly, which would compromise efficiency and lead to suboptimal solutions. Rather than maintaining a fixed learning rate throughout the training process, a more advanced approach involves using dynamic learning rate scheduling. This strategy adjusts the learning rate in response to the model's performance metrics during training, thereby providing a more flexible and effective training process. This research uses an adaptive learning rate scheduling strategy known as "ReduceLROnPlateau" to enhance the

performance of the deep learning model. This strategy continually monitors a pre-defined metric, in this case, the training loss, over multiple training epochs. If the algorithm detects no improvement in the monitored training loss for a specified number of epochs, it automatically reduces the learning rate. This allows the model to adjust its learning rate based on the training progress by taking smaller and more refined steps toward an optimal solution. This adaptability is particularly crucial for complex tasks such as robotic grasping, where the loss function landscape may contain local minima. A fixed learning rate can cause the model to become trapped in these local minima, potentially missing better solutions. The "ReduceLROnPlateau" strategy addresses this issue by dynamically lowering the learning rate, offering a mechanism to escape from these local minima and steer the model towards a more globally optimal solution.

Regularisation techniques: the training process was initiated with a focus on optimising the model's complexity to ensure that the network architecture was deep enough to accurately capture the patterns in the training data. A model that is too simplistic may not have the necessary capabilities to achieve efficient and reliable grasping, while a model with excessive complexity can become prone to overfitting. As discussed in Section 2.1.2, overfitting occurs when the model performs exceptionally well on the training data but poorly on new, unseen data. To address these challenges, the model was first allowed to intentionally overfit the training data as a diagnostic measure to ensure that it was deep enough to learn the inherent complexities of the robotic grasping problem. Once the optimal depth was selected, various regularisation techniques, including dropout, weight decay, and early stopping, were evaluated to mitigate overfitting and improve the model's generalisation performance. After rigorous testing, the best generalisation performance was achieved through a combination of dropout and early stopping. While dropout deactivates a subset of neurons randomly during each training iteration to prevent the model from relying too heavily on specific features, early stopping, on the other hand, operates by continuously monitoring the model's performance on the validation dataset during training. Once the performance on the validation set stops improving or starts to decline, training is stopped and the model state with the best validation performance is retained, regardless of its performance on the training set. When combined, these two techniques provide a powerful regularisation strategy, resulting in a generalisable model that can perform robustly across a diverse set of data.

Evaluation metrics: to evaluate the performance of a neural network, various metrics are used to assess the accuracy of the model in predicting the five-parameter grasping rectangle. Commonly used in object detection tasks, the Intersection over Union (IoU) metric measures the similarity between the predicted grasping rectangle and its corresponding ground truth. The IoU is computed by dividing the area of the overlap between the two rectangles by the area of their union. An IoU value of 1 indicates a perfect overlap, where the predicted rectangle completely matches the ground truth, while an IoU value of 0 means that there is no overlap between the predicted rectangle and the ground truth, indicating that the prediction is completely inaccurate. In the computer vision community, an IoU value above 0.5 is generally considered a successful detection. However, the robotic grasping community sets a more relaxed threshold of 0.25 due to the challenges posed by the inherent complexity of the grasping task, as well as the limited data availability in the field, which allows for a more realistic evaluation of model performance [38, 41, 42]. In standard object detection tasks, the bounding boxes are axis-aligned and not rotated. However, the robotic grasping task requires predicting an oriented rectangle, which involves predicting a rotation angle represented by θ. To evaluate the model's accuracy in predicting this rotation angle, the orientation error metric is used, which is calculated as the absolute difference between the predicted rotation angle and its corresponding ground truth value. For a prediction to be considered accurate, the orientation error must fall within a 30-degree margin [38, 41, 42]. To assess the model's ability to accurately predict the dimensions and orientation of the grasping rectangle, the Intersection over Union (IoU) and orientation error metrics are used as continuous performance indicators during both the training and validation phases. Additionally, Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are used to evaluate the overall model's performance during the training phase, providing a quantitative assessment of the prediction error. While the Mean Squared Error (MSE) is calculated by averaging the squared differences between predicted and actual values, Root Mean Squared Error (RMSE), which is the square root of MSE, provides an average measure of the deviation between the predicted and actual values, presented in the same unit as the original data, making it more interpretable.

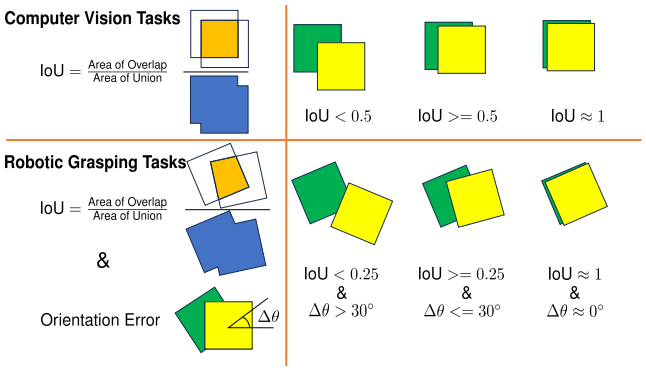


Figure 13. Illustration of the IoU and Orientation Error evaluation metrics

Workflow automation: in the field of machine learning, managing complex workflows can often be tedious, error-prone, and time-consuming due to their repetitive tasks, ranging from data pre-processing to post-processing analysis. To address these issues, a comprehensive framework that automates these tasks has been developed, leveraging a wide range of tools and techniques, with the aim of accelerating the research process and minimising potential errors. A key component of this framework is the use of user-friendly YAML files for configuration management. These files consolidate essential model configuration parameters, such as the choice of optimiser, dataset, and learning rate schedules, into a unified location. Bash scripts are used to further automate the workflow, managing various stages of the machine learning pipeline, including data pre-processing, model training, and post-processing

analysis. This centralised approach eliminates the need for manual configuration in each training session, reducing the associated error risks. In addition, PyTorch is used for training parallelisation. This built-in feature automatically distributes the computational workload across available resources, such as multiple cores on CPUs or GPUs, resulting in significant computational efficiency, especially for complex models. Throughout the training process, key performance metrics such as the loss function and accuracy values are recorded and saved in CSV format for post-processing analysis. For real-time monitoring, TensorBoard was integrated into the framework to provide live updates on the various key performance metrics, thereby facilitating the comparison of the network performance across multiple training runs.

3.3.3. Hyperparameter Tuning

As mentioned in Section 2.1.2, hyperparameters refer to a wide range of parameters that influence the learning process and must be specified by the user prior to training. Choosing the right configuration has a significant impact on factors such as the network's convergence rate, generalisation ability, and predictive accuracy. However, due to the complex interactions among the various hyperparameters, manual exploration to identify the optimal combination is both impractical and resource-intensive. To address this challenge, a variety of techniques have been developed to automate the hyperparameter tuning process by efficiently exploring the configuration space. Grid search exhaustively searches the space by evaluating the network's performance across a pre-defined set of hyperparameter values, while random search adapts a probabilistic approach by randomly selecting hyperparameter sets and evaluating the network's performance based on these selections. Relying on trial-and-error, both methods are still time-consuming and resource-intensive. A more sophisticated approach is Bayesian optimisation, which enhances the tuning process through probabilistic modelling. This method employs a Gaussian process to model the network's performance, which is then used to direct the search towards promising regions within the configuration space. By iteratively updating the model based on evaluations, Bayesian optimisation effectively uses the posterior distribution to suggest the next set of hyperparameters, thereby making efficient use of computational resources. In this study, a Bayesian approach for hyperparameter optimisation is implemented using the Optuna framework [56], which offers a variety of sampling algorithms and pruning strategies that can be combined to accelerate the optimisation process. While the sampler is responsible for suggesting hyperparameter sets, the pruner assesses the trial's progress and decides whether to continue or terminate the trial. Optuna's pruning strategy aligns with the concept of early stopping, allowing unpromising trials to end early, thereby directing computational efforts towards more promising configurations. This strategic allocation of resources significantly reduces the number of trials needed to achieve convergence towards high-performing configurations, saving both time and computational resources.

3.3.4. Results & Discussion

During training, significant computational challenges were encountered. In particular, the pre-training process using large-scale public grasping datasets required a powerful and flexible computational infrastructure capable of handling large amounts of data. To overcome the limitations of the in-house cluster several cloud computing service providers were evaluated, leading to the selection of Linode [57], which offered the computational resources and flexibility required for conducting multiple training experiments simultaneously, thereby accelerating the model selection process. Although financial constraints prevented the full utilisation of all Linode features, resource allocation was optimised through strategic decision-making. Prioritising key hyperparameters and configurations that were believed to significantly impact the model's performance allowed for more targeted experiments, thereby enabling the most effective use of the limited resources available.

After evaluating various pre-training experiments, the highest validation accuracy was achieved by incorporating background information into the training dataset. This finding aligns with the use of the ResNet model. Originally pre-trained on the ImageNet dataset, the ResNet model appeared to be inherently equipped to handle and extract relevant features from RGB images containing background information, making it perform less well on images without background. Among the datasets with background information, the Cornell Grasp Dataset consistently outperformed the Jacquard dataset, making it the most appropriate dataset for pre-training the model. In terms of architecture, several fully connected layer configurations were tested for grasp prediction. A single layer with 1024 units emerged as the most effective configuration, achieving a balance between the model's capacity to learn the complexities of the training data and its ability to generalise to new and unseen data. To prevent overfitting, dropout was integrated into the model. After testing several dropout probabilities, a rate of 0.2 was found to be the most effective, preventing overfitting while allowing the model to learn adequately from the training data. After identifying the optimal configuration for pre-training, the remainder of this section will provide a detailed overview of the model's performance.

During the pre-training phase, the deep learning model's performance was evaluated using various evaluation metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Intersection over Union (IoU), and orientation error, to assess the model's capacity for accurately learning grasp configurations from a dataset containing a wide range of objects. For a comprehensive understanding of the model's learning progress, these metrics were plotted over multiple epochs and smoothed for better visualisation.

Figures 14 (a) and (b) show the average values of Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) over training epochs, respectively. Both metrics demonstrate a consistent decreasing trend as the training progresses, indicating that the model's grasp predictions are increasingly aligning with the ground truth values. This trend also underlines that the model is able to learn effectively and improve its performance over time during training.

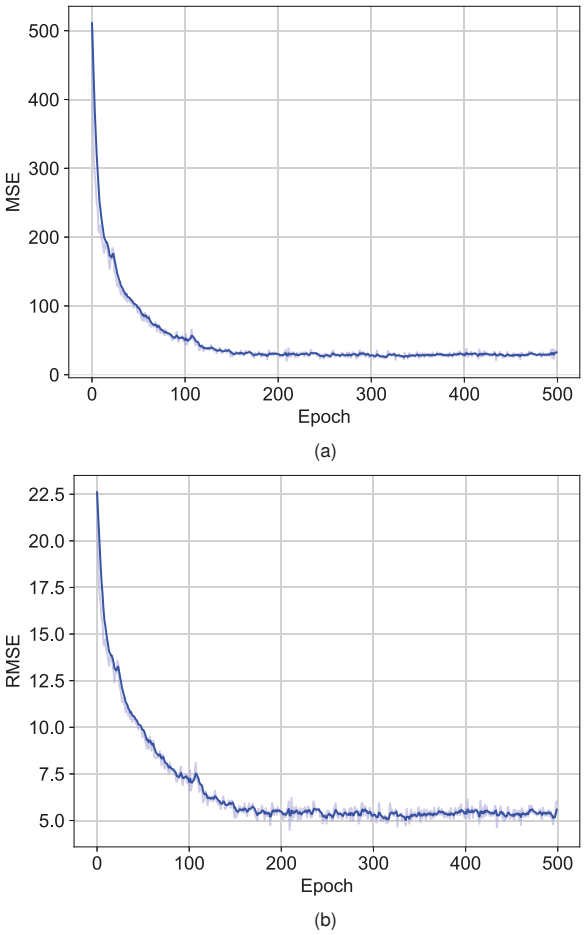


Figure 14. Illustration of the average Mean Squared Error (MSE) (a) and Root Mean Squared Error (RMSE) (b) values during the pre-training phase, plotted over training epochs

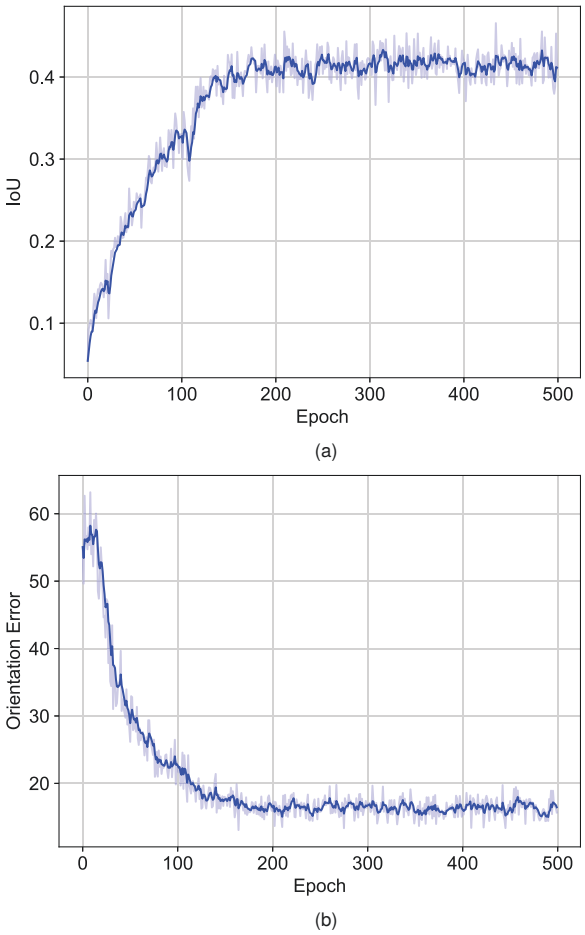


Figure 15. Illustration of the average Intersection over Union (IoU) (a) and Orientation Error (b) values during the pre-training phase, plotted over training epochs

Figures 15 (a) and (b) display the average values of Intersection over Union (IoU) and orientation error across epochs, respectively. Figure 15a demonstrates an increasing trend in the average IoU values, indicating a growing overlap between the predicted and the ground truth grasping rectangles. It is worth noting that the average IoU value surpasses the critical threshold of 0.25 relatively early in the training phase and continues to increase until it steadily converges towards a promising 0.4 by the end of training. Similarly, Figure 15b illustrates a decreasing trend in the orientation error metric, highlighting the model's improving ability to accurately predict the orientation angles of the grasp rectangles. Notably, the model surpasses the orientation error threshold of 30 degrees early in the training process and converges to an impressively low orientation error of less than 20 degrees by the end of training.

These observed trends in the selected evaluation metrics demonstrate that the model is able to improve its performance in predicting grasping rectangles over time and confirm its ability to effectively learn complex grasp configurations from RGB images. Although average values provide a general overview of a model's performance, examining the distribution of the evaluation metrics such as IoU and orientation error across images at different epochs provides a richer and more comprehensive understanding of the model's learning dynamics.

Figure 16 illustrates the distribution of orientation errors for images across various IoU thresholds, with the aim to evaluate the model's ability to learn the orientation θ, with a focus on instances where the model demonstrates high IoU accuracy. Consisting of three subfigures, Figure 16 show the evolution of the model's performance over three different training epochs: at the beginning, middle, and end of training. Each subfigure contains two sections. The top section displays violin plots, which are an effective way to visualise the distribution of orientation errors at various Intersection over Union (IoU) thresholds. These violin plots contain rich statistical information. The width of the violin at different points indicates data density at those error levels. The black line within each plot outlines the interquartile range (IQR), providing a concise summary of the data's variability. Markers on this line represent the first (Q1), second (median), and third quartiles (Q3), which are essential for understanding the data's central tendency and spread around the median. The lower section displays bar plots that show the proportion of images that satisfy or exceed different IoU thresholds. At the beginning of training, Figure 16a indicates that only a small proportion of images achieved high IoU thresholds. Furthermore, the distribution of orientation errors was significantly wider at this stage, particularly for lower IoU thresholds, indicating a higher variability

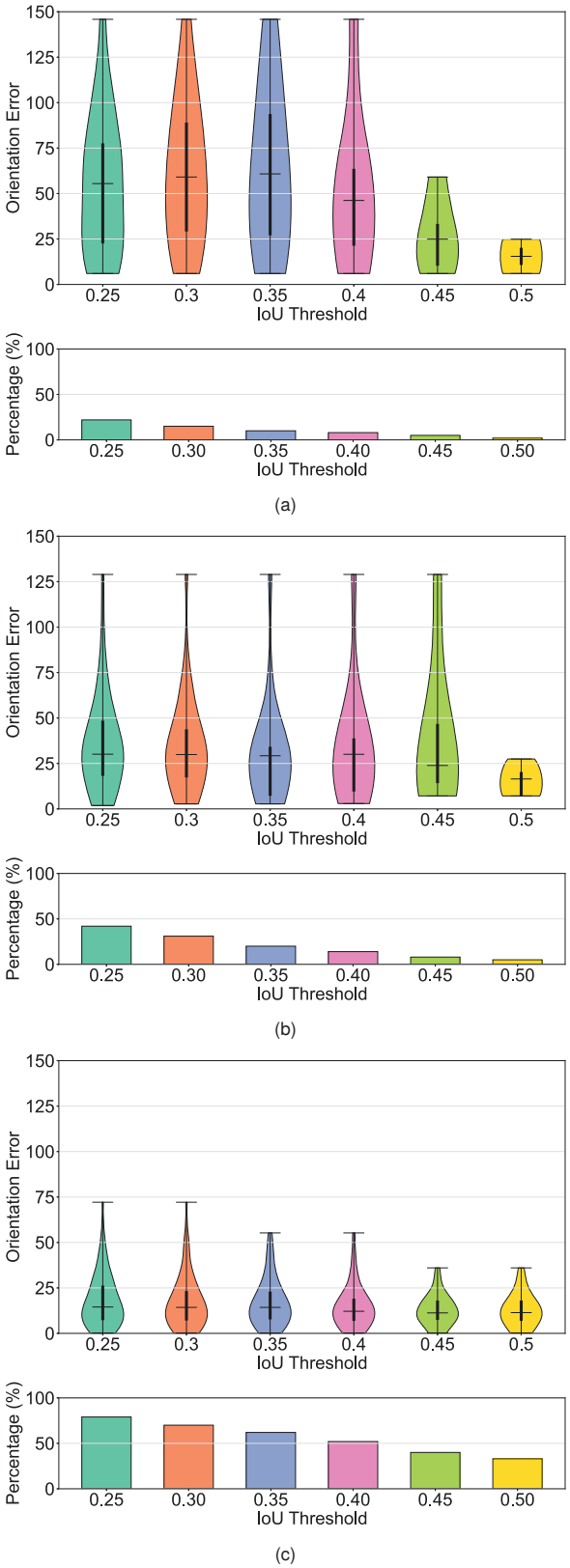


Figure 16. Illustration of the distribution of Orientation Error across various IoU thresholds during the pre-training phase, captured at the beginning (a), middle (b), and end (c) of training. Each subfigure is divided into two sections: the upper part features violin plots that visualise the distribution of the orientation errors at different IoU thresholds, while the lower part presents bar plots illustrating the proportion of images that meet or exceed these IoU thresholds

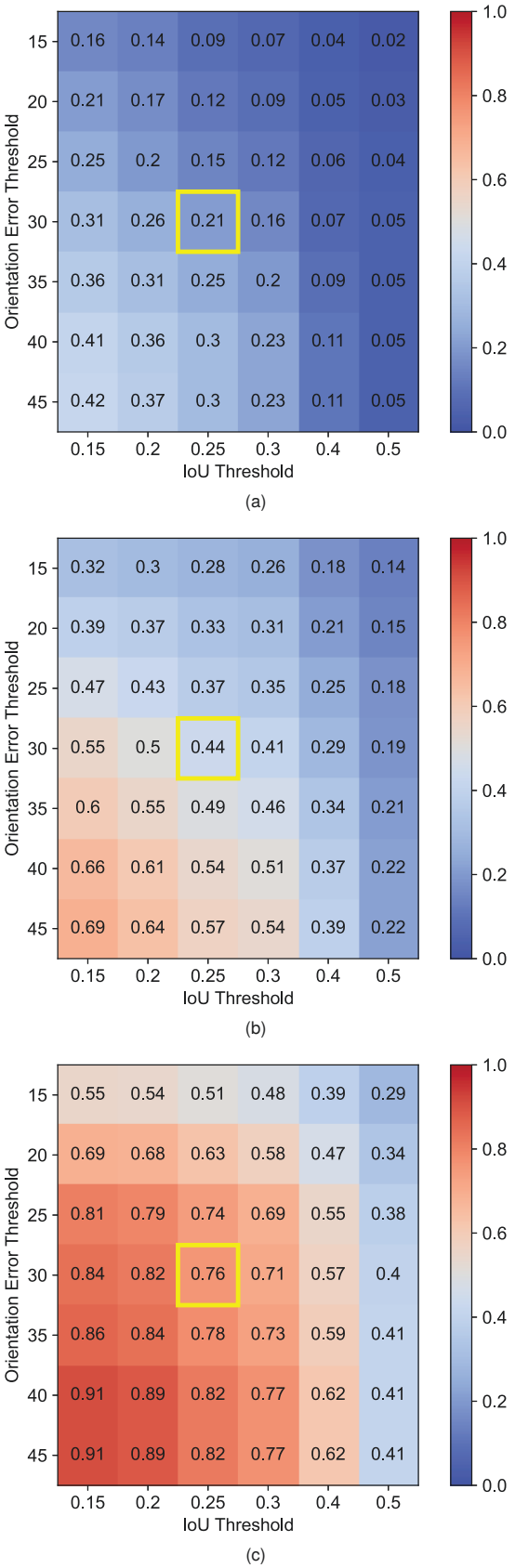


Figure 17. Illustration of the training accuracy across various combinations of Intersection over Union (IoU) and orientation error thresholds during the pre-training phase, captured at the beginning (a), middle (b), and end (c) of training

in the model's orientation predictions when the overlap with the ground truth was not significant. In other words, the model struggled to accurately predict the orientation of the grasping rectangles that were not well aligned with the ground truth. As the training progressed, a clear improvement in the model's performance was observed. Notably, Figure 16b shows that the proportion of images meeting various IoU thresholds increased significantly, with over 40% of the images achieving an IoU greater than 0.25. Additionally, the distribution of orientation errors became more concentrated across all IoU thresholds, with a distinct clustering around lower error values, particularly for higher IoU thresholds, indicating an increased precision in the model's orientation predictions. Towards the end of the training, Figure 16c demonstrates further improvements of the model's performance, with the orientation error distribution maintaining its narrowing trend, with even denser clustering around minimal errors, particularly for higher IoU thresholds. Additionally, the bar plots revealed a further increase in the percentage of images satisfying high IoU thresholds, with approximately 80% exceeding an IoU of 0.25, and nearly 40% exceeding an IoU of 0.5. At this stage, the third quartile (Q3) for orientation errors across all categories was smaller than 30 degrees, implying that 75% of all data had an orientation error less than 30 degrees. Furthermore, the median orientation error was less than 20 degrees across all IoU thresholds, highlighting the model's improved accuracy in predicting the orientation θ of the grasping rectangles.

In order to further evaluate the model's performance, its training accuracy is examined across different IoU and orientation error thresholds. Figure 17 displays three heatmaps representing different stages of the training process. Each heatmap provides a visual representation of the model's accuracy for various threshold combinations of the two primary evaluation metrics: Intersection over Union (IoU) and orientation error. The heatmap's colour intensity represents the percentage of predictions that satisfy both the IoU and orientation error thresholds. Warmer colours indicate a higher proportion of predictions meeting the criteria, reflecting superior model performance, while cooler shades indicate the opposite. A specific cell is highlighted in each heatmap, corresponding to predictions that have an IoU greater than 0.25 and an orientation error less than 30 degree, which are the threshold values commonly used in the robotic grasping community to identify successful grasp predictions. Comparing the different heatmaps in Figure 17 reveals that the model gradually improves its performance in terms of IoU accuracy and orientation prediction as the training progresses. At the end of the training, the model achieves an accuracy of 76% under the standard thresholds of an IoU greater than 0.25 and an orientation error less than 30 degrees. It's worth noting that the model's performance is even better when evaluated under more relaxed thresholds, with an accuracy of 91% for an IoU greater than 0.15 and an orientation error below 45 degrees.

In order to assess the generalisation ability of the deep learning model, the model's performance on the validation set was evaluated using the same set of evaluation metrics. The validation performance of the deep learning model is visually represented by the heatmap in Figure 18, which displays the proportion of successful grasps across a range of IoU and orientation error thresholds. The validation results show that the model achieves an accuracy of 70% under the conventional IoU and orientation error thresholds of 0.25 and 30 degrees, respectively. This performance improves up to 80% under more relaxed thresholds, highlighting its ability to generalise well across novel objects.

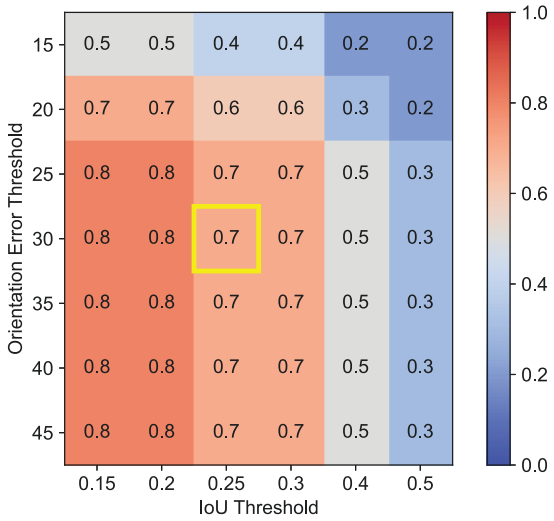


Figure 18. Illustration of the validation accuracy across various combinations of Intersection over Union (IoU) and orientation error thresholds during the pre-training phase

For a better visualisation of the model's performance, visual comparisons between predicted and ground truth grasping rectangles throughout the training and validation phases are shown in Figure 19. Ground truth grasps are shown in green, while predicted grasps are shown in yellow. The blue lines indicate the projection of the gripper finger pads onto the image plane.

After pre-training the deep learning model to establish a foundational understanding of robotic grasping for novel objects, the model was finetuned on a targeted, custom-generated dataset, which is described in Section 3.2. The aim of this additional training phase was to optimise the model's performance for specific objects and conditions relevant to space exploration missions. Due to the limited size of the custom dataset used for finetuning, plotting distributions of evaluation metrics, as was done for pre-training, may not provide enough statistical power to draw conclusions about model performance. Instead, the following will focus on visualisation methods that are better suited to smaller datasets.

Figure 20 present the average values of MSE, RMSE, IoU, and orientation error over the training epochs of the finetuning process. The results indicate that the model achieved similar performance on the custom dataset, showing consistent improvement over time, thereby validating the model's learning efficacy. Notably, the evaluation metrics showed quicker convergence during the finetuning phase, reaching better values in fewer epochs. This outcome suggests that the pre-training was effective and successfully provided a favourable initialisation for the finetuning stage, allowing the model to effectively build upon its pre-trained grasp prediction skills. However, an increase in variability of performance metrics across epochs was observed. This is due to smaller batch sizes resulting from the smaller dataset. While this introduced more fluctuation in the learning curves, the overall

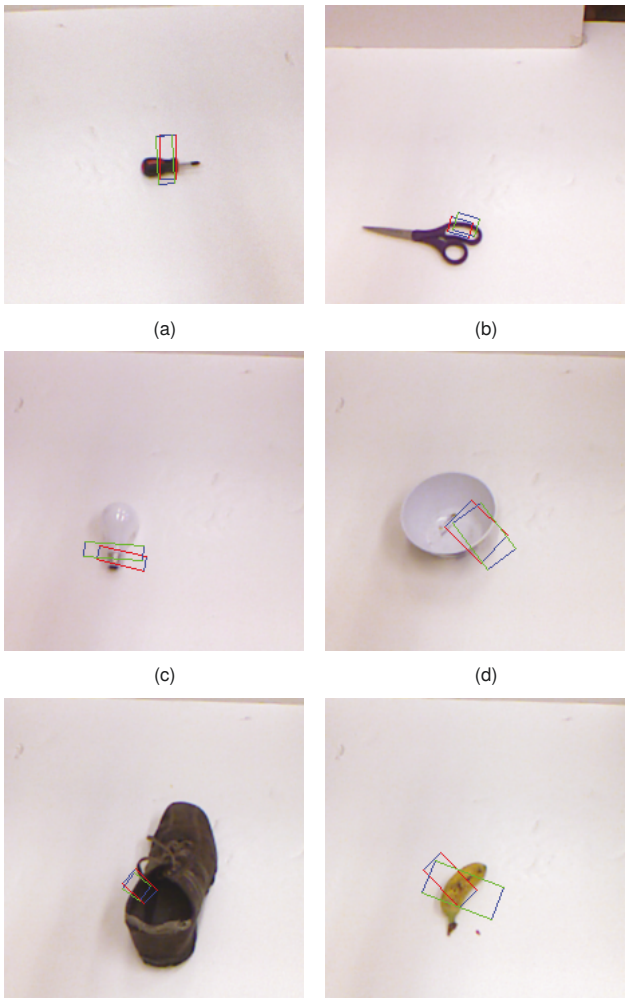


Figure 19. Examples of predicted grasps (in red), along with ground-truth grasps (in green), during the pre-training phase

trend in all key metrics remained positive. This stochastic nature also appears to be beneficial for escaping local minima, thereby aiding the model in achieving a more generalised solution for robotic grasping tasks, as can be seen in the following.

Figure 21 illustrate heatmaps representing the model's performance at various training stages after finetuning. Similar to the pre-training phase, the results show that the performance of the finetuned model improves as training progresses. By the end of the training, the model reaches an accuracy level of 80% under standard thresholds. Under more relaxed criteria, the model performs even better, achieving an accuracy level of 90%.

Figure 22 displays the validation performance of the finetuned model, illustrating its accuracy across a wide range of IoU and orientation error thresholds. The results in Figure 22 indicate an enhanced performance of the finetuned model during the validation stage, achieving an impressive accuracy level of 86% under the community-standard IoU and orientation error thresholds. Furthermore, the model maintains an accuracy level of over 50% even under stricter

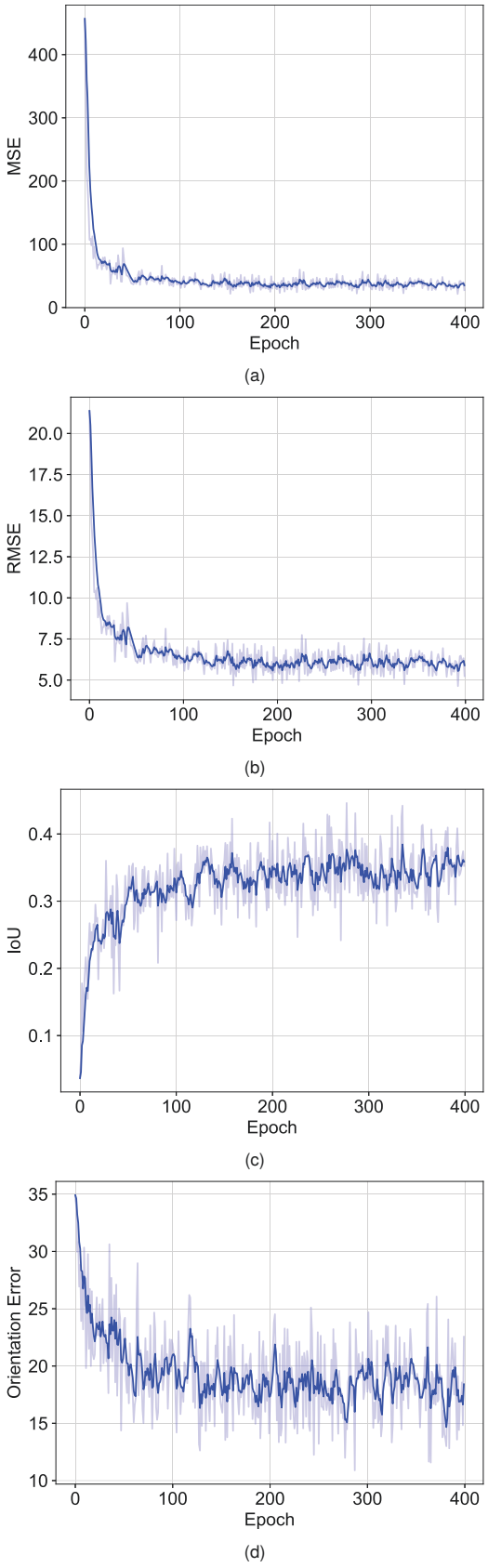


Figure 20. Illustration of the average Mean Squared Error (MSE) (a), Root Mean Squared Error (RMSE) (b), Intersection over Union (IoU) (c) and Orientation Error (d) values during the finetuning phase, plotted over training epochs

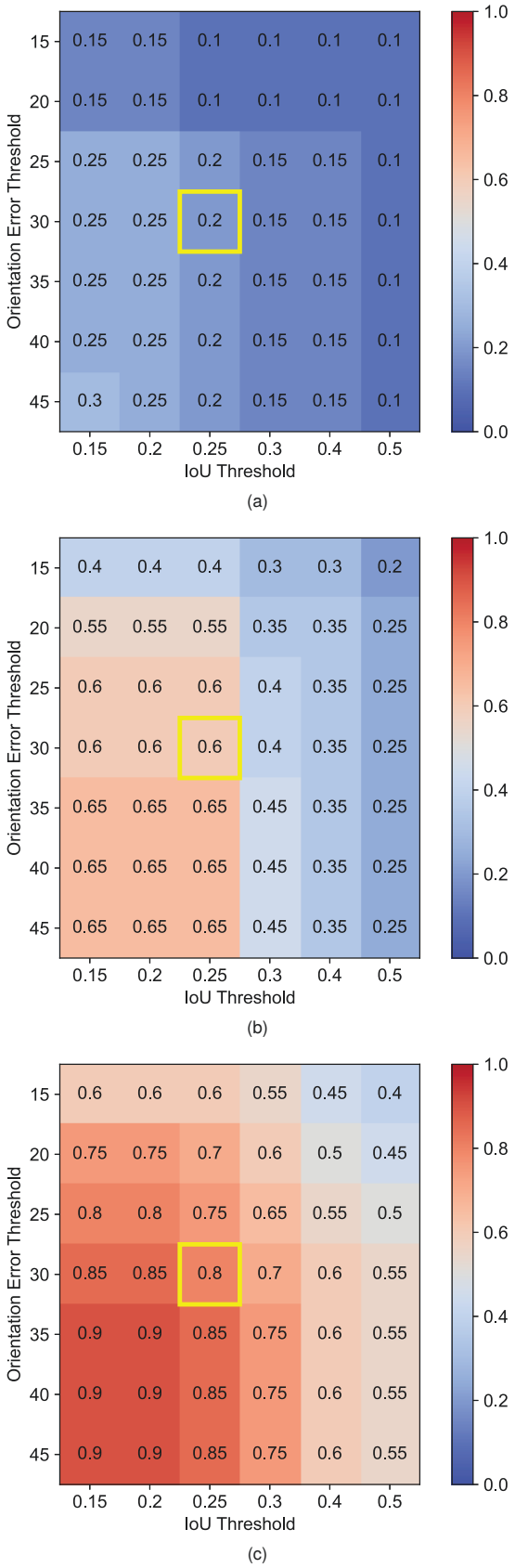


Figure 21. Illustration of the training accuracy across various combinations of Intersection over Union (IoU) and Orientation Error thresholds during the finetuning phase, captured at the beginning (a), middle (b), and end (c) of training

criteria, such as an IoU threshold of up to 0.4 and an orientation error below 15 degrees. These results suggest that the model can generalise effectively to new data, thereby confirming the effectiveness of the finetuning process, which benefits from the advantageous starting point provided by pre-training, leading to improved overall performance in grasp prediction.

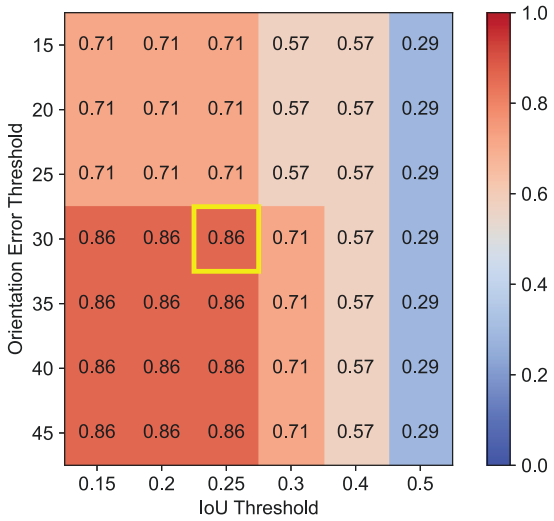


Figure 22. Illustration of the validation accuracy across various combinations of Intersection over Union (IoU) and Orientation Error thresholds during the finetuning phase

For a better visualisation of the model's performance, Figure 23 presents visual comparisons between predicted and ground truth grasping rectangles throughout the training and validation phases.

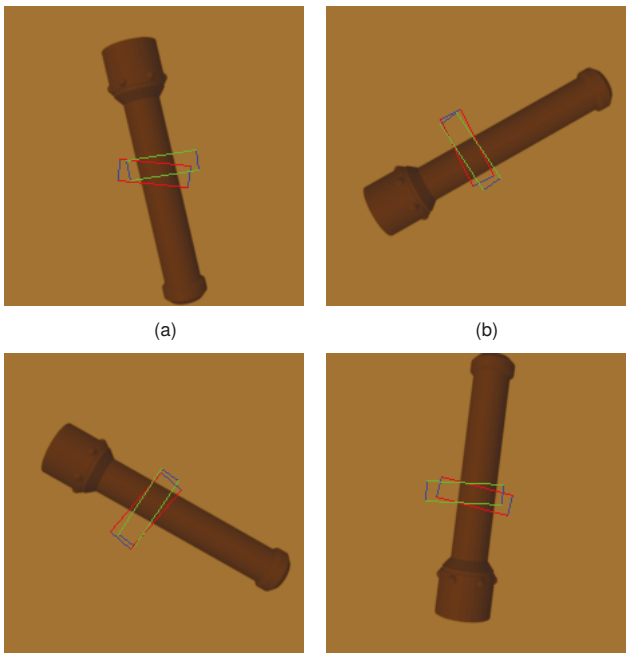


Figure 23. Examples of predicted grasps (in red), along with ground-truth grasps (in green), during the finetuning phase

4. CONCLUSION AND FUTURE WORK

This study aimed to enhance the autonomous capabilities of robotic systems for future space exploration missions, with an emphasis on robotic applications for planetary rovers. Focusing on autonomous robotic grasping, an end-to-end grasp estimation system using state-of-the-art deep learning techniques was developed.

The developed system highlighted the potential of deep learning methods in enabling planetary rovers to autonomously manipulate previously unknown objects based solely on visual information, minimising the need for pre-programmed instructions or real-time human interaction. This enhanced autonomy could not only improve the operational efficiency of planetary rovers but also enables more advanced missions, involving complex tasks such as sample retrieval, maintenance, and servicing activities.

To generate a dataset with objects relevant to space exploration missions, a custom 3D simulation environment was designed to emulate extraterrestrial terrains and conditions. The dataset generation framework demonstrated its potential for creating large, scalable, and mission-relevant datasets. Although limited in size due to computational limitations, a preliminary dataset was created using the developed framework, which serves as a proof of concept and foundation for future, more comprehensive datasets containing a wider range of objects and more challenging terrains.

Training deep learning models requires significant computational resources for efficient training and hyperparameter tuning. While initially faced with computational limitations, cloud computing solutions were used to accelerate the model selection process by providing scalable, on-demand computational resources. Leveraging the power of transfer learning, publicly available datasets were used to pre-train existing state-of-the-art deep learning models, which were adapted to the task of robotic grasping. The pre-trained model with the best performance was then finetuned using a custom-generated dataset. Within used reduced datasets for both pre-training and finetuning, a promising success rate of 85% could still be achieved in grasping novel objects based only on RGB-D visual information. This result not only represents a significant step toward enabling future rovers to operate with a high degree of autonomy, without the need for pre-existing knowledge of target objects but also lays a solid foundation for future research, where larger datasets can further enhance the model's accuracy.

This study highlights the importance of computational resources in conducting large-scale research in robotic grasping. Future work can therefore explore cost-effective computational solutions that can offer the required resources to fully unlock the potential of the developed framework by generating more comprehensive datasets and training more complex deep learning models to potentially improve the model's performance.

Our research suggests several avenues for future work. One immediate extension is to incorporate more complex terrains, which will enhance the realism of the training environment, making the model more adaptable to actual space missions. Another direction is to expand the dataset to include a broader range of objects relevant to space exploration, which will not only diversify the training data, but also make the model more robust to different types of grasping scenarios.

The simulation environment developed in this study for dataset generation has implications beyond its intended purpose, as it can also be used for testing and evaluating previously trained deep learning models. Moreover, it enables the use of more sophisticated learning algorithms, such as reinforcement learning methods, which have shown promising results in terrestrial robotic grasping tasks. Since reinforcement learning models require an environment to interact with during training, the developed simulation environment is an invaluable asset for future research in this direction.

While the simulation environment provides a valuable controlled experimental setup, it is essential to recognise the potential challenges associated with transferring the developed algorithms to real-world scenarios. Future work will therefore involve empirical validation of the deep learning models in more realistic settings by incorporating hardware-in-the-loop testing to bridge the gap between simulated and real conditions.

Contact address:

maha_badri@yahoo.fr
gewehr@irs.uni-stuttgart.de
klinkner@irs.uni-stuttgart.de

References

[1] Philip J. Stooke. *The International Atlas of Mars Exploration: Volume 2, 2004 To 2014: From Spirit to Curiosity*. Cambridge University Press, New York, Mar. 2016. ISBN: 978-1-107-03093-0.

[2] John P. Grotzinger, Joy Crisp, Ashwin R. Vasavada, Robert C. Anderson, Charles J. Baker, Robert Barry, David F. Blake, Pamela Conrad, Kenneth S. Edgett, Bobak Ferdowski, Ralf Gellert, John B. Gilbert, Matt Golombek, Javier Gómez-Elvira, Donald M. Hassler, Louise Jandura, Maxim Litvak, Paul Mahaffy, Justin Maki, Michael Meyer, Michael C. Malin, Igor Mitrofanov, John J. Simmonds, David Vaniman, Richard V. Welch, and Roger C. Wiens. Mars Science Laboratory Mission and Science Investigation. *Space Sci Rev*, 170(1):5–56, Sept. 2012. ISSN: 1572-9672. DOI: 10.1007/s11214-012-9892-2.

[3] Kevin Nickels, Matthew DiCicco, Max Bajracharya, and Paul Backes. Vision guided manipulation for planetary robotics — position control. *Robotics and Autonomous Systems*, 58(1):121–129, Jan. 2010. ISSN: 0921-8890. DOI: 10.1016/j.robot.2009.07.029.

[4] Tao Zhang, Kun Xu, Zhixiao Yao, Xilun Ding, Zeng Zhao, Xuyan Hou, Yong Pang, Xiaoming Lai, Wenming Zhang, Shuting Liu, and Jianfeng Deng. The progress of extraterrestrial regolith-sampling robots. *Nat Astron*, 3(6):487–497, June 2019. ISSN: 2397-3366. DOI: 10.1038/s41550-019-0804-1.

[5] Brian K. Muirhead, Austin Nicholas, and Jeff Umland. Mars Sample Return Mission Concept Status. In *Proceedings of the IEEE Aerospace*

Conference, pages 1–8, Big Sky, MT, USA, Mar. 2020. IEEE. ISBN: 978-1-72812-734-7. DOI: 10.1109/AERO47225.2020.9172609.

[6] Gianluca Cerilli and Martin Zwick. Visual servoing for sample tube detection and pick-up on Mars. In *Proceedings of the Advanced Space Technologies in Robotics and Automation (ASTRA) Conference*, Noordwijk, The Netherlands, May 2019.

[7] Raul Castilla-Arquillo, Carlos Perez-del-Pulgar, Gonzalo Jesus Paz-Delgado, and Levin Gerdes. Hardware-Accelerated Mars Sample Localization Via Deep Transfer Learning From Photorealistic Simulations. *IEEE Robot. Autom. Lett.*, 7(4):12555–12561, Oct. 2022. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3219306.

[8] Israel Raul Tiñini Alvarez, Ignacio Amat Pérez, Tim Wiese, Laura Bielenberg, and Renaud Detry. Sample-tube pose estimation for fetching on Mars based on two-stage inference. In *Proceedings of the Advanced Space Technologies in Robotics and Automation (ASTRA) Conference*, Noordwijk, The Netherlands, June 2022.

[9] L. M. Mantoani, R. Castilla-Arquillo, G. J. Paz-Delgado, C. J. Perez-del-Pulgar, and M. Azkarate. Samples detection and retrieval for a Sample Fetching Rover. In *Proceedings of the Advanced Space Technologies in Robotics and Automation (ASTRA) Conference*, Noordwijk, the Netherlands, June 2022.

[10] Jeremie Papon, Renaud Detry, Peter Vieira, Sawyer Brooks, Thirupathi Srinivasan, Ariel Peterson, and Eric Kulczycki. Martian Fetch: Finding and retrieving sample-tubes on the surface of mars. In *Proceedings of the IEEE Aerospace Conference*, pages 1–9, Big Sky, MT, USA, Mar. 2017. DOI: 10.1109/AERO.2017.7943649.

[11] Shreyansh Daftry, Barry Ridge, William Seto, Tu-Hoa Pham, Peter Ilhardt, Gerard Maggiolino, Mark van der Merwe, Alex Brinkman, John Mayo, Eric Kulczyski, and Renaud Detry. Machine Vision based Sample-Tube Localization for Mars Sample Return. In *Proceedings of the IEEE Aerospace Conference*, pages 1–12, Big Sky, MT, USA, Mar. 2021. IEEE. ISBN: 978-1-72817-436-5. DOI: 10.1109/AERO50100.2021.9438364.

[12] Moritz Gewehr, Jan Gläser, and Sabine Klinkner. Optimisation of the Locomotion Performance for Exploration Rover Systems Using a Highly Variable Chassis Platform. In *Proceedings of the Advanced Space Technologies in Robotics and Automation (ASTRA) Conference*, Leiden, The Netherlands, June 2017.

[13] Moritz Gewehr, Tristan Meyer, and Sabine Klinkner. Development and operative performance analyses of the modular rover chassis platform (MRCP). In *Proceedings of the Advanced Space Technologies in Robotics and Automation (ASTRA) Conference*, Noordwijk, The Netherlands, May 2019.

[14] M. Alam, M.D. Samad, L. Vidyaratne, A. Glandon, and K.M. Iftekharuddin. Survey on Deep Neural Networks in Speech and Vision Systems. *Neurocomputing*, 417:302–321, Dec. 2020. ISSN: 09252312. DOI: 10.1016/j.neucom.2020.07.053.

[15] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Comput. Intell. Mag.*, 13(3):55–75, Aug. 2018. ISSN: 1556-603X, 1556-6048. DOI: 10.1109/MCI.2018.2840738.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14539.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, Jan. 1989. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms, June 2017. DOI: 10.48550/arXiv.1609.04747.

[19] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for Deep Learning: A Taxonomy, Oct. 2017. DOI: 10.48550/arXiv.1710.10686.

[20] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148(3):574–591, 1959. ISSN: 1469-7793. DOI: 10.1113/jphysiol.1959.sp006308.

[21] Mehrunnissa. Application of Deep Convolution Neural Network for Image Classification: A Review. *IJATCSE*, 11(2):42–46, Nov. 2022. ISSN: 22783091. DOI: 10.30534/ijatcse/2022/011122022.

[22] Wang Zhiqiang and Liu Jun. A review of object detection based on convolutional neural network. In *Proceedings of the Chinese Control Conference (CCC)*, pages 11104–11109, Dalian, China, July 2017. IEEE. ISBN: 978-988-15639-3-4. DOI: 10.23919/ChiCC.2017.8029130.

[23] Mingqi Gao, Feng Zheng, James J. Q. Yu, Caifeng Shan, Guiguang Ding, and Jungong Han. Deep learning for video object segmentation: A review. *Artif. Intell. Rev.*, 56(1):457–531, Jan. 2023. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10176-7.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile, Dec. 2015. IEEE. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.123.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, Jan. 2015. DOI: 10.48550/arXiv.1409.0575.

[27] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, Apr. 2015. DOI: 10.48550/arXiv.1409.1556.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions, Sept. 2014. DOI: 10.48550/arXiv.1409.4842.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, Dec. 2015. DOI: 10.48550/arXiv.1512.03385.

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. DOI: 10.1109/CVPR.2009.5206848.

[31] Elon Rimon and Joel Burdick. *The Mechanics of Robot Grasping*. Cambridge University Press, Cambridge, 2019. ISBN: 978-1-108-42790-6. DOI: 10.1017/9781108552011.

[32] D. Halliday, R. Resnick, and J. Walker. *Fundamentals of Physics*. John Wiley & Sons Canada, Limited, 2010. ISBN: 978-0-470-54793-9.

[33] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 348–353 vol.1, Apr. 2000. DOI: 10.1109/ROBOT.2000.844081.

[34] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-Driven Grasp Synthesis - A Survey. *IEEE Trans. Robot.*, 30(2):289–309, Apr. 2014. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2013.2289018.

[35] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from RGBD images: Learning using a new rectangle representation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3304–3311, May 2011. DOI: 10.1109/ICRA.2011.5980145.

[36] Amaury Depierre, Emmanuel Dellandrea, and Liming Chen. Jacquard: A Large Scale Dataset for Robotic Grasp Detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3516, Madrid, Spain, Oct. 2018. IEEE. ISBN: 978-1-5386-8094-0. DOI: 10.1109/IROS.2018.8593950.

[37] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11441–11450, June 2020. DOI: 10.1109/CVPR42600.2020.01146.

[38] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, Apr. 2015. ISSN: 0278-3649. DOI: 10.1177/0278364914549607.

[39] Lerrel Pinto and Abhinav Gupta. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours, Sept. 2015. DOI: 10.48550/arXiv.1509.06825.

[40] Dongwon Park and Se Young Chun. Classification based Grasp Detection using Spatial Transformer Network, Mar. 2018. DOI: 10.48550/arXiv.1803.01356.

[41] Joseph Redmon and Anelia Angelova. Real-Time Grasp Detection Using Convolutional Neural Networks, Feb. 2015. DOI: 10.48550/arXiv.1412.3128.

[42] Sulabh Kumra and Christopher Kanan. Robotic Grasp Detection using Deep Convolutional Neural Networks, July 2017. DOI: 10.48550/arXiv.1611.08036.

[43] Qiang Zhang, Daokui Qu, Fang Xu, and Fengshan Zou. Robust Robot Grasp Detection in Multimodal Fusion. *MATEC Web Conf.*, 139:00060, 2017. ISSN: 2261-236X. DOI: 10.1051/matecconf/201713900060.

[44] Yuxi Li. Deep Reinforcement Learning: An Overview, Nov. 2018. DOI: 10.48550/arXiv.1701.07274.

[45] Moritz Gewehr, Andreas Schneider, Josef Dalcolmo, and Sabine Klinkner. Mission Concepts and New Technologies for Lunar Surface Exploration using the Nanokhod Microrover. In *Proceedings of the International Astronautical Congress (IAC)*, Paris, France, Sept. 2022.

[46] Moritz Gewehr and Sabine Klinkner. Evaluation of an Interdisciplinary Postgraduate Education Program on Space Robotics and Planetary Exploration Technologies within the Institute of Space Systems at the University of Stuttgart. In *Proceedings of the International Astronautical Congress (IAC)*, Washington D.C., United States, Oct. 2019.

[47] Weltraumrobotik - Institut für Raumfahrtsysteme - Universität Stuttgart. https://www.irs.uni-stuttgart.de/forschung/satellitentechnik/rover/.

[48] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning, 2016.

[49] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols. *IEEE Robot. Automat. Mag.*, 22(3):36–52, Sept. 2015. ISSN: 1070-9932. DOI: 10.1109/MRA.2015.2448951.

[50] Walter Wohlkinger, Aitor Aldoma, Radu B. Rusu, and Markus Vincze. 3DNet: Large-scale object class recognition from CAD models. In *2012 IEEE International Conference on Robotics and Automation*, pages 5384–5391, May 2012. DOI: 10.1109/ICRA.2012.6225116.

[51] Khaled Mamou, E Lengyel, and A Peters. Volumetric hierarchical approximate convex decomposition. In *Game Engine Gems 3*, pages 141–158. AK Peters, 2016.

[52] Dawson-Haggerty et al. Trimesh. https://trimsh.org/, Dec. 2019.

[53] Kjetil Wormnes, William Carey, Thomas Krueger, Leonardo Cencetti, Emiel den Exter, Stephen Ennis, Edmundo Ferreira, Antonio Fortunato, Levin Gerdes, Lukas Hann, Chiara Lombardi, Erica Luzzi, Sebastian Martin, Matteo Massironi, Samuel Payler, Aaron Pereira, Angelo Pio Rossi, Riccardo Pozzobon, Francesco Sauro, Philippe Schoonejans, Frank van der Hulst, and Jessica Grenouilleau. ANALOG-1 ISS – The first part of an analogue mission to guide ESA's robotic moon exploration efforts. *Open Astronomy*, 31(1):5–14, Jan. 2022. ISSN: 2543-6376. DOI: 10.1515/astro-2022-0002.

[54] NASA-3D-Resources. https://github.com/nasa/NASA-3D-Resources, Aug. 2023.

[55] Dave Shreiner, Graham Sellers, John M. Kessenich, Bill Licea-Kane, and Khronos OpenGL ARB Working Group, editors. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley, Upper Saddle River, NJ, eighth edition edition, 2013. ISBN: 978-0-321-77303-6.

[56] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[57] Cloud Computing & Linux Servers | Alternative to AWS | Linode. https://www.linode.com.