Autonomous Vision-Based Navigation around Asteroids Using Convolutional Neural Networks

van der Heijden, L.F.J.; Mooij, E.; Woicke, S.

DOI
10.2514/6.2025-1699

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the AIAA SCITECH 2025 Forum

AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-1699

Autonomous Vision-Based Navigation around Asteroids using Convolutional Neural Networks

Lars F.J. van der Heijden*and Erwin Mooij†
Delft University of Technology, Faculty of Aerospace Engineering,
Kluyverweg 1, 2629 HS Delft, The Netherlands

Svenja Woicke‡
OHB SE,
Manfred-Fuchs-Platz 2-4, 28359 Bremen, Germany

This paper investigates the efficacy of Convolutional Neural Network (CNN) based methods to navigate autonomously around asteroids. The main contribution of this work is the successful development of a first-of-a-kind pose estimation pipeline, consisting of a CNN-based feature detector and a Perspective-n-Points (PnP) solver to allow accurate, safe, and autonomous distance estimation with respect to a target asteroid. A top-down CNN-based feature detector is developed, consisting of an object and keypoint detection network in sequence, which detects n pre-defined keypoints, designated on the target's 3D model, within the 2D image. The simulated target asteroid is Bennu, a subkilometer asteroid with a spinning top-shape and pronounced equatorial bulge. The networks have been trained and evaluated on synthetic datasets created in this work, consisting of 32,352 images with a variety of poses from 4.5 to 9 km from the asteroid, for different illumination conditions, asteroid orientations, and image corruptions that emulate real sensor artifacts. The pipeline could achieve a mean and median *line-of-sight* distance estimate of around 42 m and 30 m, respectively, at a confidence level of 90% for the large relative range, while satisfying the accuracy requirement of a maximum of 10% knowledge error for 99.979% of the cases.

Nomenclature

$\mathbf{C}_{B,A}$	=	Transformation matrix from frame A to frame B
c_x, c_y	=	x,y location of the camera's principal point in the pixel reference frame, px
du, dv	=	Horizontal and vertical pixel length, m/px
$\mathbf{E}_{px,i}$	=	Pixel error for a single keypoint detection i, px
$\mathbf{E}_t$	=	Translational error, m
$\mathcal{F}_x$	=	Reference frame x
f	=	Focal length, m
f_x	=	Scaled focal length in the x-direction, px
f_y	=	Scaled focal length in the y-direction, px
$\mathbf{K}$	=	Intrinsic camera parameters matrix
ME, k	=	Mean error of the keypoint detections for an image k, px

*MSc graduate, Section Astrodynamics and Space Missions. Currently: Consultant at McKinsey & Company
†Associate Professor, Section Astrodynamics and Space Missions, e.mooij@tudelft.nl, Associate Fellow AIAA
‡Chief Project Engineer, svenja.woicke@ohb.de

N_u, N_v	=	Number of horizontal and vertical pixels in the image, px
n	=	Order of refinement of the icosphere
n	=	Number
$\mathbf{P}$	=	Pose matrix
$\mathbf{p}$	=	Detected 2D feature in the pixel reference frame
$\mathbf{q}$	=	Quaternion
R	=	Radius of the icosphere
$\mathbf{r}^A$	=	Position vector presented in the A frame
$\mathbf{r}^B_{A/B}$	=	Position vector from the origin of the B frame to the origin of the A frame presented in the B frame
s_x, s_y	=	Scaling factor: The size of one pixel in the x- or y-direction, respectively, 1/m
$\mathbf{t}$	=	Translational vector, m
$\hat{\mathbf{t}}$	=	Predicted translational vector, m
u_i, v_i	=	2D x and y-coordinate of the i-th keypoint presented in the pixel reference frame, px
$\hat{u}_i, \hat{v}_i$	=	Predicted 2D x and y-coordinate of the i-th keypoint presented in the pixel reference frame, px
x^A, y^A, z^A	=	Cartesian x, y, z coordinates presented in frame A
α	=	Bearing angle around Y_B, rad
α	=	Learning rate
β	=	Bearing angle around X_B, rad

I. Introduction

Missions to small bodies are increasingly gaining interest, as they might hold the secrets to our Solar System's origin, while some are also posing a threat to life on Earth. The recently launched DART and HERA missions are the latest examples in a line of successful asteroid missions, such as NASA's NEAR[1] and Dawn missions,[2] and JAXA's Hayabusa I and II missions.[3,4] The small size and irregular shape of asteroids result in complex dynamics and accurate navigation is vital for a mission's success. The use of Earth-based navigation does not allow for accurate navigation during some close-proximity mission phases, either due to too large uncertainty on the position estimate or due to round-trip communication delays up to 20 min. Therefore, recent missions have used some form of vision-based autonomous navigation.[5,6] Vision-based navigation systems solely relying on a monocular camera, i.e., relying on a single image taken by a navigation camera to estimate the position of the camera with respect to the asteroid, are also becoming a more attractive alternative to active sensors, such as lidar, or stereo cameras, as they have lower mass, hardware complexity, costs, and power consumption.[7–9] Developing monocular vision-based autonomous navigation is especially relevant, given the recent increase in research into the use of deep-space CubeSats,[10,11] which have stringent requirements on the available resources.

Currently used vision-based navigation approaches either rely on detecting pre-defined *landmarks* on the target,[2,12] detecting (physical) features and matching them to a database,[1,13,14] tracking craters or unknown features across images (relative navigation),[15] or navigate through the use of artificial markers.[3,4]

The extraction of visual features is a vital part of these vision-based navigation systems, and the accuracy of the system is determined by the Image Processing (IP) algorithm used. The majority of hand-engineered IP algorithms, such as Harris corner detectors, relies on the image gradient to detect rich-textured features on highly visible parts of the target. These detected features are image-specific and, therefore, this does not allow for an offline feature selection step,[16] thereby requiring a computationally expensive online image-to-model mapping step.[13] Moreover, these methods rely heavily on *a-priori* information or depend on the precision of an initial state estimate.[13] Furthermore, traditional IP algorithms lack robustness against adverse illumination conditions and image noise,[16] which is crucial for use in a vision-based navigation system for space-borne problems.[17]

Machine learning methods offer a great alternative to these traditional vision-based autonomous navigation approaches, while mitigating the mentioned drawbacks. Machine learning has, over the years, been successfully applied to tackle terrestrial problems of increasing complexity, and only as of 2019 the research into using deep learning for spacecraft navigation has begun through the Spacecraft Pose Estimation Challenge (SPEC).[18] An in-depth survey of deep-learning techniques for spacecraft relative navigation is given in

Ref.,[19] but the main applications revolve around *pose estimation of uncooperative spacecraft*, *crater identification/classification and subsequent navigation*, and using it to *estimate orbital parameters or the gravitational field*. The research into *pose estimation of uncooperative spacecraft* is the largest. Great progress is being made, boosted by the new SPEC 2021.[20]

Applications of deep-learning techniques to asteroid missions in particular focused on estimations of the gravitational field,[21] small-body shape classification,[22] and landing-site detection for autonomous soft-landing.[23] Mancini et al.[24] developed a deep-learning-based method improving on the OSIRIS-REx Natural Feature Tracking (NFT) navigation used for landing. Pugliatti and Topputo[25] use segmentation maps from which they predict a class, which corresponds to an estimated state of the spacecraft with respect to the asteroid. Each of these applications rely on the use of Convolutional Neural Networks (CNNs).

Yet, to the best of our knowledge, no one developed a learning-based pose estimation pipeline for asteroid navigation. This is a deep-learning landmark-based approach, referred to in machine learning literature as keypoint detection, mimicking methods used on asteroid missions, such as Dawn, and covered in research.[12,13] These methods relied on the detection of *pre-defined* landmarks for which the 3D location was known and the 2D-3D correspondence could be established. This was subsequently used to solve for the pose of the spacecraft with respect to the target. These methods relied on the availability of a 3D shape model.

The majority of the research revolves around human and uncooperative spacecraft pose estimation, and can be divided into *end-to-end* and *feature-based* architectures. Within *end-to-end* architectures a single CNN replaces the entire pipeline and directly outputs the pose from an input image.[26–28] Within *feature-based* architectures a hand-engineered IP algorithm is replaced by a CNN-based feature detector, which detects n pre-defined features from the 2D image and these 2D keypoints and their 2D-3D correspondence are then sent to a pose solver, which solves the PnP problem. The *feature-based* architectures are the better option, as they have outperformed the *end-to-end* architectures on the SPEC,[18] with a four-times smaller average position error and seven-times smaller average orientation error. Moreover, *end-to-end* architectures result in a more complex learning problem, which requires more data to train them. The use of a CNN facilitates an offline feature selection step prior to training and, as such, avoids the cumbersome and computationally intensive image-model matching step of the detected 2D feature with its location on the 3D model, plaguing traditional approaches.[16] Furthermore, another advantage is that the CNNs output predictions for all n features, even when they are occluded or not directly visible (e.g., at the back of the target), thereby not relying on the number of landmarks directly in view to make accurate predictions. Moreover, CNNs have been proven to be more robust against adverse illumination conditions and image noise compared to traditional IP approaches.[17] CNN-based approaches also do not rely on an initial pose/state estimate and can be used for pose initialization, i.e., lost-in-space scenarios.

The research mostly focuses on developing CNN-based architectures that perform well on a given dataset, without considering the computational effort required to run these pipelines, staying within the theoretical realm. However, for a realistic application to spacecraft processors, which have limited computational capacity and memory, efficiency is equally important. Howard et al.[29] started a trend within machine learning to develop more lightweight networks that achieve similar performance. Manning et al.[30] demonstrated that such lightweight networks can be used on the space-grade embedded system of a CubeSat.

However, due to the unavailability of publicly available large-scale datasets of asteroid images suitable for deep-learning purposes, these CNNs have to be trained using synthetic images rendered in a 3D rendering software, such as Blender[31–33] or Cinema4D.[34] Unfortunately, no publicly available dataset of synthetic asteroid images exists.

Inspired by promising research into the use of CNNs for pose estimation of uncooperative spacecraft, the main objective of this paper is to investigate the efficacy of a CNN-based pipeline to navigate autonomously around asteroids. The main contributions of this work are:

1. The development of a first-of-a-kind CNN-based pose estimation pipeline suitable for autonomous navigation around asteroids;

2. The use of lightweight networks, having only a fraction of the parameters and Floating Points Operations (FLOPs) compared to other state-of-the-art deep learning networks. Making the application of machine learning to spacecraft more realistic;

3. The development of a model-agnostic synthetic-image generation and annotation pipeline suitable for the creation of deep-learning datasets using Blender and Python.

The target asteroid is Bennu, as a detailed 3D model is publicly available, as well as a lot of literature surrounding the mission and the asteroid. Bennu is a sub-kilometer asteroid with a mean radius of around 246 m and a spinning top-shape and a pronounced equatorial bulge[a], similar to the Didymos asteroid, which is the target of the upcoming HERA mission designed by ESA. The Didymos asteroid has an estimated radius of 390 m, however, no shape model of the Didymos asteroid is publicly available[b]. Therefore, an asteroid that shares similarity with it (Bennu) would allow this work to contribute to the upcoming HERA mission, demonstrating that such a technique could work. The main focus of this pipeline is to estimate the distance of the spacecraft relative to the asteroid. However, the pipeline also outputs an estimate of the orientation of the camera towards the asteroid, which can be used, if desired, alongside traditional instruments, such as star-trackers, in a navigation filter.

This proposed pipeline is purely software-based and does not require any additional hardware besides a navigation camera and processor. This allows the CNNs to be trained on Earth during the mission and the parameters of the networks (weights and biases) can simply be uploaded and updated throughout the mission, if needed.

The paper is organized as follows. Section II lists the reference frames used. Section III discusses the pose estimation framework, after which the dataset created in this work is detailed in Sec. IV. The proposed algorithm/pipeline is detailed in Section V. The results are presented in Sec. VI and the paper is concluded with the conclusions in Sec. VII.

II. Reference frames

There are five reference frames referenced throughout this paper: world, asteroid, Blender camera, camera, and pixel, denoted by $\mathcal{F}_W$, $\mathcal{F}_A$, $\mathcal{F}_B$, $\mathcal{F}_C$, and $\mathcal{F}_P$, respectively. They are all right-handed systems, but do have different origins and characteristics. $\mathcal{F}_W$ is used in Blender to place the object, camera, and light source, and has the $+Z_W$-axis pointing up, and the X and Y axes follow the right-handed triad.

$\mathcal{F}_A$ has its origin in the asteroid's center of mass and the 3D model of the asteroid is loaded into Blender with $+Z$ as up and $+Y$ as forward. This ensures that the axes of the asteroid correspond with the world axes at the initial orientation. $\mathcal{F}_B$ and $\mathcal{F}_C$ both have their origin in the center of projection and the axes in $\mathcal{F}_C$ are defined according to usual convention. $\mathcal{F}_P$ has its origin in the top left corner of the image plane. The definition of the axes of $\mathcal{F}_B$, $\mathcal{F}_C$, and $\mathcal{F}_P$ can be observed in Fig. 1.

III. Pose estimation framework

This section discusses the pose estimation framework, elaborating on the navigation system and presenting the Perspective-n-Points (PnP) equations, which are used to solve for the pose of the spacecraft with respect to the asteroid.

A. Pose estimation

This work considers a spacecraft flying in close-proximity around an asteroid. The navigation system is used to navigate at distances ($\approx$ ¡ 10 km) in which the asteroid does not fully cover the Field of View (FOV) of the camera. Moreover, the navigation system is used to (re-)initialize the navigation filter or another relative navigation approach at the start of the descent and landing phase. The relative attitude of the spacecraft with respect to the asteroid can be defined as the rotation of $\mathcal{F}_A$ with respect to $\mathcal{F}_C$. The relative position can be described by the vector originating from the origin of $\mathcal{F}_C$ to the origin of $\mathcal{F}_A$. These two quantities define the pose of the spacecraft with respect to the asteroid, which can accordingly be transformed from $\mathcal{F}_C$ to a different frame, such as the NED reference frame, to describe the motion of the spacecraft with respect to the target. However, this is not performed within this work, as the focus lies merely on the development of the pose estimation pipeline, and not on the incorporation within a navigation architecture.

It is assumed that the spacecraft is equipped with a single monocular camera where no variable pointing is possible, i.e., the camera direction with respect to the spacecraft body frame is fixed. Therefore, $\mathcal{F}_C$ can be

[a] `https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/101955-bennu/in-depth/`, Date accessed: 31-01-2022

[b] `https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/didymos/in-depth/`, Date accessed: 31-01-2022

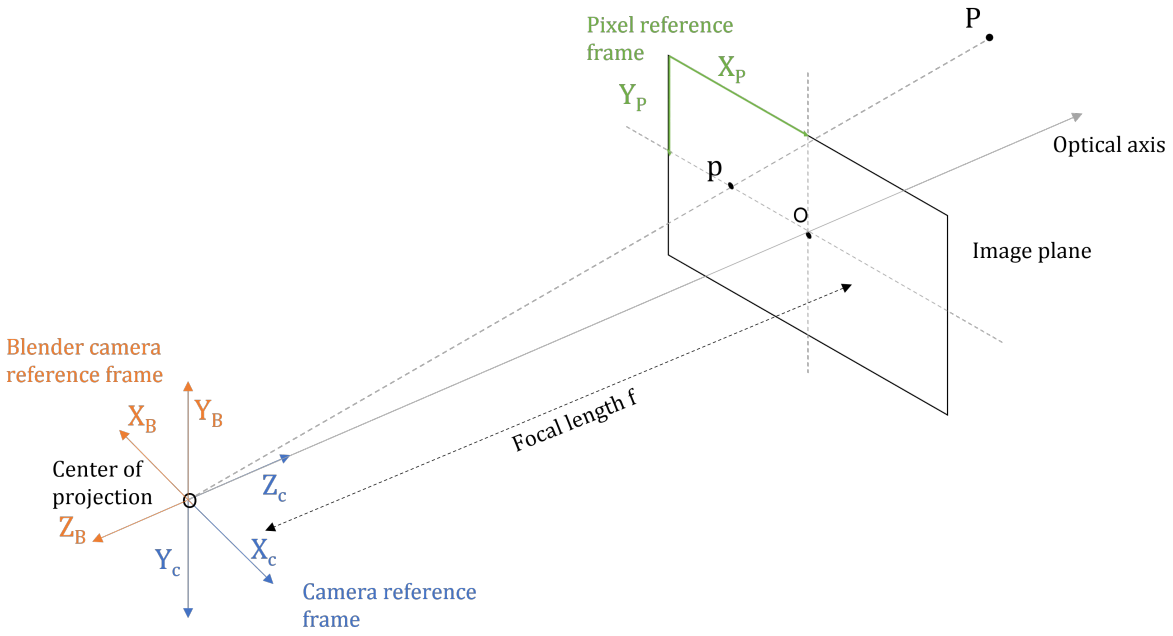


Figure 1: The virtual pinhole camera model containing the different reference frames and other relevant definitions

used to describe the orientation of the spacecraft, removing the need to use a separate spacecraft body-fixed reference frame for the development of the pipeline.

A model-based monocular pose estimation pipeline receives a 2D image and provides an estimate of the pose of the camera with respect to the target. The goal of the Perspective-n-Points (PnP) problem, as illustrated in Fig. 2, is to determine the distance from the target's center of mass ($\mathbf{t}^C$) and its orientation ($\mathbf{C}_{C,A}$) with respect to the camera, given the camera's intrinsic parameters and the set of n 2D-3D correspondences between the points' 3D body coordinates and their 2D projections. The 3D model of the target has to be available for the estimation.

The PnP equations are given below and relate the unknown pose to a detected 2D feature $\mathbf{p}$, using the position of that feature with respect to camera frame, $\mathbf{r}^C$. This is graphically illustrated in Fig. 2.

$$\mathbf{r^C} = \begin{pmatrix} x^C & y^C & z^C \end{pmatrix}^T = \mathbf{t}^C + \mathbf{C}_{C,A}\mathbf{r}^A \tag{1}$$

$$\mathbf{p} = (u_i, v_i) = \left(\frac{x^C}{z^C} f_x + c_x, \frac{y^C}{z^C} f_y + c_y \right) \tag{2}$$

where $\mathbf{r}^A = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^T$ is the i^{th} feature's location presented in the asteroid reference frame, where i = 1,2, ... , k, and k represents the number of designated 3D features on the 3D model. The same applies for $\mathbf{p} = \begin{pmatrix} u_i & v_i \end{pmatrix}^T$ in which i = 1,2 ... , k, represents the k corresponding 2D image points. f_x and f_y refer to the scaled focal lengths ($f_x = s_x f, f_y = s_y f$) in the respective principal directions, and (c_x, c_y) refer to the coordinates of the principal point in the pixel reference frame. Furthermore, an ideal camera is assumed meaning no skewness and distortion are modeled and the intrinsic camera parameters are known. However, when using a real camera, these parameters are estimated through a calibration procedure, which introduces small errors on the ground-truth pose labels, i.e., ground-truth label does not perfectly represent the true location. This does not influence the performance of the algorithm itself, and only effects the resulting pose error. However, the use of a real camera in combination with a mock-up of the asteroid is a different research problem altogether[20]

These equations can be rewritten in matrix form, which illustrates how a point in 3D can be linked to a point in 2D. These equations demonstrate that an important aspect of generating accurate pose estimates is the capability of the IP algorithm to extract the features $\mathbf{p}$ from the 2D image of the target.

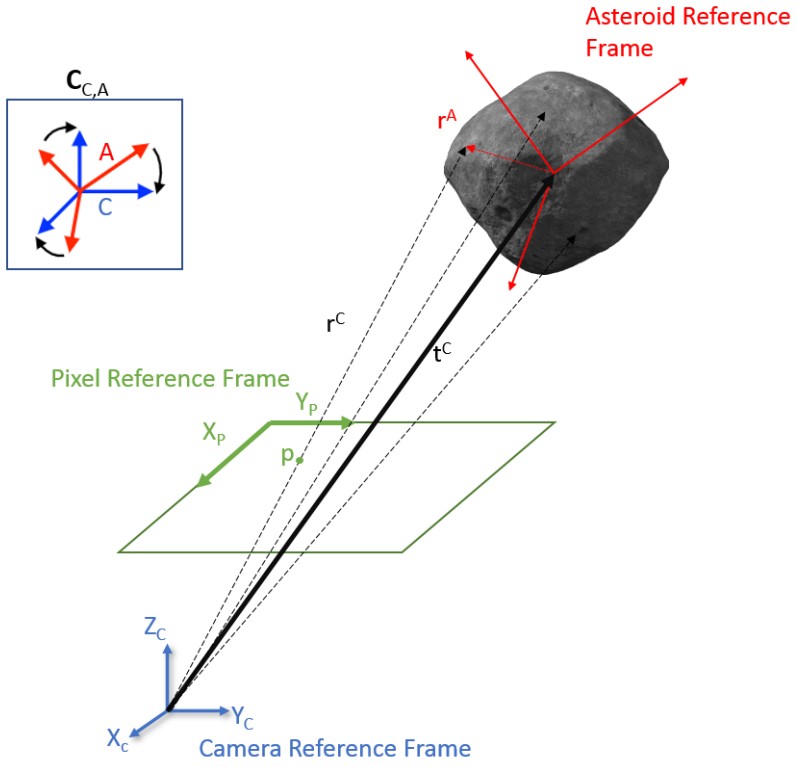


Figure 2: The geometric depiction of the PnP problem

$$\mathbf{p} = \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = [\mathbf{K}][\mathbf{P}] \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{3}$$

where $\mathbf{K}$ represents the camera intrinsic parameter matrix and $\mathbf{P} \in \mathbb{R}^{3\times4}$ represents the pose matrix, where $\mathbf{t}^C$ can also be written as $\mathbf{r}^C_{A/C}$.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} \mathbf{C}_{C,A} & | & \mathbf{t}^C \end{bmatrix} \tag{4}$$

B. Pose estimation solver

The Efficient Perspective-n-Points (EPnP) method[35] available through the OpenCV [c] library, is used in this work. However, for an actual mission the EPnP solver, available through OpenCV, can be replaced with an own implementation following the equations set forth in Ref.[35] This solver has been successfully applied for pose estimation in recent works.[17,18,36–38] The EPnP method solves the PnP problem, as outlined in Eqs. (1) and (2), in a non-iterative way, producing a closed-form solution and removing the need for an initial pose estimate. The main idea of the EPnP solver is to rewrite the 3D keypoints as a weighted sum of four different non-coplanar virtual *control points*, c_j, where $j = 1, 2, 3, 4$, presented in the asteroid reference frame (Fig. 2). These 3D keypoints, r_i^A, are expressed in barycentric coordinates, α_{ij}, of these virtual control points: $r_i^A = \sum_{j=1}^{4} \alpha_{ij} c_j$. The coordinates of these virtual control points are arbitrary. Thereby, rewriting Eq. (3) as a function of a 12-dimensional vector $\hat{\mathbf{x}}$ containing the 12 unknown control points coordinates in

[c] `https://opencv.org/`, Date accessed: 7-10-2021

Table 1: The intrinsic camera parameters used in generating the synthetic images

Parameter	Description	Value
N_u	Number of horizontal pixels	1024
N_v	Number of vertical pixels	1024
f_x	Horizontal focal length	0.15 m
f_y	Vertical focal length	0.15 m
du	Horizontal pixel length	$2.34375{\cdot}10^{-7}$ m/px
dv	Vertical pixel length	$2.34375{\cdot}10^{-7}$ m/px
FOV	Field of View	13.686°

the camera frame, resulting in a linear system, $\mathbf{M}\hat{\mathbf{x}} = 0$, where $\mathbf{M} \in \mathbb{R}^{2n\times 12}$ is a known matrix for n 2D-3D correspondences. The EPnP solver only has $O(n)$ computational complexity, compared to other state-of-the-art PnP solvers that have $O(n^5)$ or higher,[35] making it suitable for usage on resource constrained devices. The EPnP solver needs a minimum of four 2D detections ($n \geq 4$) to output a solution, however, more are desirable.

IV. Dataset

Due to the unavailability of large-scale datasets of real asteroid images, the CNNs are trained and evaluated using synthetic images generated within this work using the Blender software and its Cycles rendering engine. The advantage of a synthetic dataset is that a uniform sampling of different viewpoints can be guaranteed, mitigating the problem of viewpoint bias plaguing real image datasets.[39] The methodology and code base is model-agnostic, and although the current dataset consists of images of the Bennu asteroid, any 3D model can be seamlessly interchanged in future works. This dataset is made publicly available to foster research within this field and to serve as a benchmark for various state-of-the-art deep learning-based navigation techniques[d]. The camera settings used for rendering are shown in Table 1. The dataset consists of 32,352 images of 1024×1024 px, which is the size often used for navigation cameras, such as the Asteroid Framing Camera used for the Dawn and HERA mission.[13,40] The 'virtual' camera used has a FOV of 13.686°. These images are randomly sampled and split into training, validation, and test sets in a 70%/15%/15% fashion. The Sun is assumed to be located in the equatorial plane and the Sun-asteroid vector is fixed. Examples of rendered images from the dataset are shown in Fig. 3. The entire dataset is annotated in the COCO format[e], allowing for ease of use with a range of keypoint detection networks. The main settings of the dataset can be found in Table 2.

Apart from the rendering itself, three major aspects, which will be discussed in more detail within this section, cover the dataset generation: 1) Determining the camera pose distribution (Subsec. A): This creates the different camera poses, distance and orientation, with respect to the target asteroid. This is used to place the camera within the 3D rendering software Blender and as such create a diverse dataset consisting of a variety of viewpoints. 2) Creating a textured 3D model (Subsec. B): This focuses on creating the textured 3D model by using a 3D shape model of the asteroid. 3) Annotating the dataset (Subsec. C): The designated keypoints and the bounding boxes need to be annotated for each image to allow for the training of the keypoint and object detection networks, respectively. Apart from the creation of a 'clean' dataset using the aforementioned steps, additionally an augmented dataset is created, which is discussed in more detail in Subsec. D.

A. Pose distribution

This subsection discusses the camera pose distribution, which is used to place the camera within the 3D rendering software Blender and as such create a diverse dataset consisting of a variety of viewpoints.

[d] `https://www.kaggle.com/datasets/larsvanderheijden/asteroid-pose-estimation-dataset`

[e] `https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch#:~:text=The\%20COCO\%20dataset\%20is\%20formatted,\%E2\%80\%9D\%20(in\%20one\%20case)`, Date accessed: 4-09-2022

Figure 3: A selection of synthetic images from the dataset

The pose of the camera with respect to the asteroid is required to set up the pose matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$, $\left[\begin{array}{c|c} \mathbf{C}_{C,A} & \mathbf{r}^{C}_{A/C} \end{array}\right]$ as shown in Eq. (4) and to allow for the training and evaluation of the CNNs. This pose is determined through three different factors, namely the distance of the camera to the asteroid ($\mathbf{t}^{C} = \mathbf{r}^{C}_{A/C}$), the camera's orientation with respect to the asteroid ($\mathbf{C}_{C,A}$), and the asteroid's orientation with respect to the world axes, ψ. The camera pose allows for a variety of different viewpoints and illumination conditions.

The different poses can generally be generated using two methods, which represent the exact same pose in relative geometry, but the only thing that changes are the illumination conditions, i.e., shadows and reflectance of the target: 1) Keeping the camera fixed and moving and rotating the target object, and 2) keeping the target object fixed and rotating and moving the camera.

However, the target in this work is an asteroid that rotates around its rotational axis in a predictable fashion. Furthermore, the location of the Sun with respect to the asteroid is fixed for a certain point in its orbit. As a result, the camera-fixed approach cannot be used, as this would result in unrealistic illumination conditions. Therefore, a new approach was devised to generate different camera viewpoints (attitude and

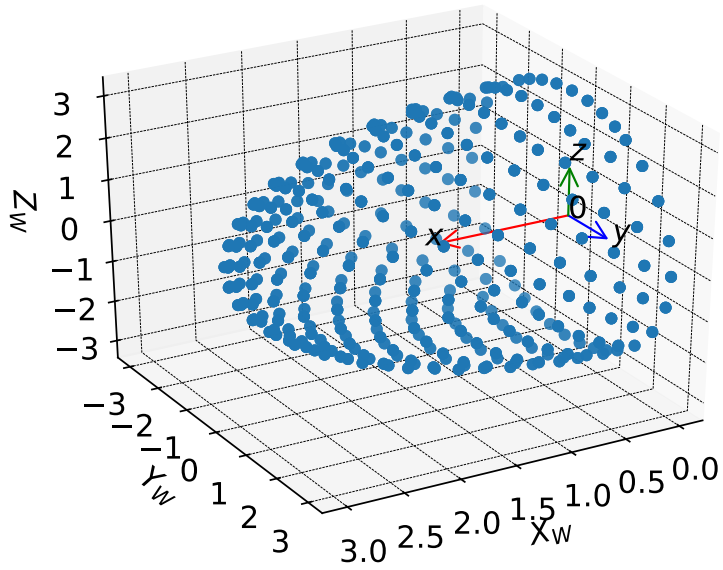


Figure 4: The post-processed icosphere, where all datapoints with $x^W < 0$ are removed, with the selected refinement order of 3 and an arbitrary distance of 3 km

distances) of the target while attaining realistic illumination conditions.

The camera viewpoints are created following a method proposed by Hinterstoisser et al.[41] This method uses an icosahedron, which is the largest convex regular polyhedron and has 12 vertices. The sampling is refined by recursively replacing each triangle with four almost equilateral triangles. These vertices are then projected onto a sphere to form an icosphere. The radius and refinement order of the icosphere are variables. Figure 4 shows the vertices representing the camera's locations for the selected refinement order and an arbitrary distance of 3 km with the datapoints for $x^W < 0$ removed. This refinement order creates a fine sampling with an angle of $\approx 8°$ in between vertices. Given the position of the Sun, only a certain part of the target is illuminated, whereas the other parts are in complete darkness. The only relevant camera positions for a vision-based navigation system are the ones that look at the partially illuminated part of the asteroid, which results in approximately half of the icosphere as shown in Fig. 4. The details regarding the distance and attitude are discussed separately in the upcoming subsections.

1. Distance of camera to the target

The target is placed at the origin of the icosphere and, as such, the vertex locations represent the camera position with respect to the target's origin. This is synonymous with the relative distance between the camera and the target. This places the camera within Blender and represents the position of the Blender camera with respect to the world frame, $\mathbf{r}^W_{B/W}$. The translational vector $\mathbf{t}^C = \mathbf{r}^C_{A/C}$ required to set up the pose matrix (Eq. (4)), can be constructed using the following:

- $\mathbf{r}^W_{B/W}$: The position vector from the origin of the world frame to the origin of the Blender camera frame presented in the world frame. However, only the orientation of the asteroid changes, i.e., the side that is illuminated, but the origin remains aligned with the world origin. Therefore, $\mathbf{r}_{B/W} = \mathbf{r}_{B/A}$
- $\mathbf{C}_{C,W} = \mathbf{C}_{C,B}\mathbf{C}_{B,W}$: The transformation matrix from the world frame to the camera frame.
- The origin of the Blender camera frame and the camera frame are the same, and therefore $\mathbf{r}_{B/A} = \mathbf{r}_{C/A}$.

The final translational vector $\mathbf{r}^C_{A/C}$ can henceforth be calculated as follows:

$$\mathbf{t}^C = \mathbf{r}^C_{A/C} = -\mathbf{r}^C_{C/A} = -\mathbf{C}_{C,W}\mathbf{r}^W_{C/A} \tag{5}$$

The distance can be selected arbitrarily within a certain range without influencing the CNN pipeline too much, as the pipeline will not directly solve the pose from the image, but makes use of an object detection and keypoint detection network. The distances are varied from 4.5 km to 9 km with 1.5 km increments, as listed in Table 2. This allows the asteroid to be fully within the camera field of view, while covering a large relative range without considerably increasing the size of the dataset. For closer distances, either the asteroid is not fully in the field of view or significant portions of the asteroid are cut off in off-nominal pointing cases, rendering the images useless. For larger distances (> 9 km), ground-based navigation or other techniques can safely navigate the spacecraft, and feature-based navigation can become complicated when the asteroid only makes up around 15% of the pixels within the image for the camera settings used.

2. Attitude of the camera with respect to the target

The distance is specified using the radius of the icosphere, whereas a given vertex can have any arbitrary camera orientation. This attitude of the camera with respect to the target is specified in Blender using a tracking constraint. This tracking constraint makes sure that the camera is pointing up (Y_B-axis is pointing up) and that the optical axis of the camera, $-Z$-axis, is pointing towards the origin (center of mass) of the target asteroid reference frame A. This results in a so-called nominal pointing case, where it is assumed that the origin of $\mathcal{F}_A$, lies on the same perspective line from the origin of $\mathcal{F}_B$, towards the center of the asteroid in $\mathcal{F}_P$ (Fig. 2), i.e., the 2D centroid ($\mathcal{F}_P$) of the asteroid lies on the same perspective line as the actual 3D centroid ($\mathcal{F}_A$) of the asteroid.

Therefore, in the nominal pointing case the target (asteroid) is located in the center of the image plane, such that the coordinates of the center of mass in the image reference frame are $\mathbf{p}^P = (\frac{Nu}{2}, \frac{Nv}{2})$. However, the asteroid will not necessarily lie in the center of the image throughout the actual mission and the deviation from the middle of the image can be set using the azimuth and elevation angles (α, β), i.e., bearing angles. The camera's position is fixed and only the orientation of the camera with respect to the target is changed through the use of these bearing angles. This is illustrated in Fig. 5, where the Blender camera frame is rotated over the Y and X-axes, respectively. A positive rotation of α and β results in the camera's $-Z$-axis (optical axis) pointing towards the top left corner of the 'old' image plane. This results in point p becoming the new principal point O and, as such, the asteroid has 'moved' to the lower right corner within the new image plane.

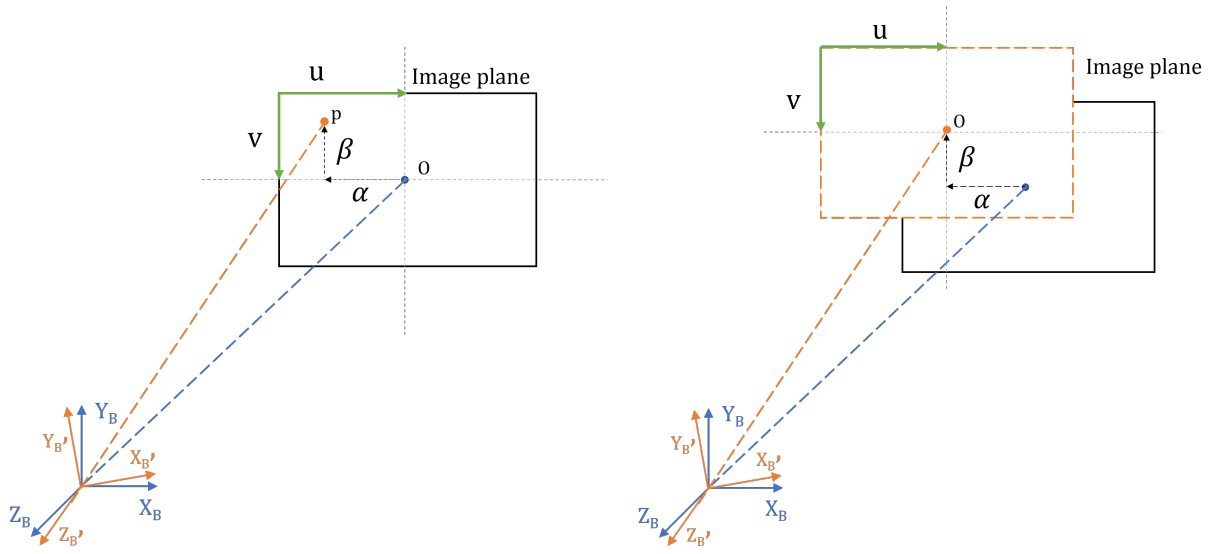
The values of α and β are randomly selected from the interval: $-5^\circ < \alpha, \beta < 5^\circ$. This interval allows the asteroid to be randomly spread over the image, but would still guarantee that most of the asteroid is in the field of view. Furthermore, a minimum off-nominal pointing angle for α and β is specified as a variable. This was implemented to ensure that the off-nominal pointing case did not closely resemble the nominal pointing case. The practical result is that a square region, governed by the minimum degree, surrounding the center is off-limits. The number of off-nominal cases that are to be generated per vertex is another variable that can be specified. The minimum degree was set to 1° and the number of off-nominal cases per vertex was set to 3, which can be found in Table 2.

The transformation matrix that rotates the nominal-pointing Blender camera frame B to the off-nominal pointing Blender camera frame B' is given by:

$$\mathbf{C}_{B',B} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ \sin\alpha\sin\beta & \cos\beta & \cos\alpha\sin\beta \\ \sin\alpha\cos\beta & -\sin\beta & \cos\alpha\cos\beta \end{bmatrix} \tag{6}$$

The position vector presented in B' reference frame can then be determined using $\mathbf{r}^{B'}_{\mathrm{B/A}} = \mathbf{C}_{B',B}\mathbf{r}^{B}_{B/A}$, where $\mathbf{r}^{B}_{B/A} = \begin{pmatrix} 0 & 0 & \left\|\mathbf{r}_{\mathrm{B/A}}\right\|_2 \end{pmatrix}^T$, and $\left\|\mathbf{r}_{\mathrm{B/A}}\right\|_2$ is the 2-norm of $\mathbf{r}_{\mathrm{B/A}}$. This vector only has a z-component, as in the nominal pointing case the $-Z$-axis (optical axis) points towards the center of mass of the target.

The B' reference frame is the final orientation of the Blender camera frame when bearing angles are applied, and $'$ is only used to distinguish between the nominal pointing case and the off-nominal pointing case. Both the nominal and the off-nominal camera poses describe the orientation of the Blender camera with respect to the world frame, $\mathbf{C}_{B,W}$, and are irrespective to the asteroid's orientation with respect to the world axes, i.e., the camera's orientation does not depend on how the target is placed within the world frame. However, the transformation matrix describing the orientation of the camera with respect to the asteroid, $\mathbf{C}_{C,A}$, required to set up the pose matrix (Eq. (4)), needs the orientation of the asteroid with respect to the



(a) This shows the image plane of the nominal pointing case where the asteroid is in the center of the image plane (O)

(b) This shows the situation in which the optical axis, $-Z$-axis, points towards the point p, making it the new optical point (O), moving the asteroid to the lower right corner

Figure 5: Graphically illustrating the off-nominal pointing implementation

world axes, ψ. The 3D model is imported to Blender using the +Y-forward and +Z-up convention, aligning the asteroid reference frame with the world axes for the initial orientation. The orientation determines which side of the asteroid is illuminated and the full circle of rotation is discretized into 60° intervals, as listed in Table 2. This captures the asteroid from different sides and recreates a variety of relative geometries and illumination conditions between the asteroid and the spacecraft, without considerably increasing the size of the dataset. The required transformation matrix $\mathbf{C}_{C,A}$ can be established from the following: i) $\mathbf{C}_{A,W}$: The transformation matrix from the world reference frame to the asteroid reference frame, which is calculated using the orientation angle ψ, ii) $\mathbf{C}_{B,W}$: The transformation matrix from the world reference frame to the Blender camera frame, iii $\mathbf{C}_{C,B}$: The transformation matrix from the Blender camera frame to the camera frame.

The desired transformation matrix $\mathbf{C}_{C,A}$ can be calculated using the following:

$$\mathbf{C}_{C,A} = \mathbf{C}_{C,B}\mathbf{C}_{B,W}\mathbf{C}_{A,W}^{T} \tag{7}$$

where $\mathbf{C}_{C,B}$ is constant: $\mathbf{C}_{C,B} = \mathbf{C}_x(-\pi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\pi) & \sin(-\pi) \\ 0 & -\sin(-\pi) & \cos(-\pi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$.

Furthermore, $\mathbf{C}_{A,W}$ and $\mathbf{C}_{B,W}$ are defined to be:

$$\mathbf{C}_{A,W} = \mathbf{C}_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{C}_{B,W} = \begin{bmatrix} \left(q_0^2 + q_1^2 - q_2^2 - q_3^2\right) & 2\left(q_1q_2 + q_0q_3\right) & 2\left(q_1q_3 - q_0q_2\right) \\ 2\left(q_1q_2 - q_0q_3\right) & \left(q_0^2 - q_1^2 + q_2^2 - q_3^2\right) & 2\left(q_2q_3 + q_0q_1\right) \\ 2\left(q_1q_3 + q_0q_2\right) & 2\left(q_2q_3 - q_0q_1\right) & \left(q_0^2 - q_1^2 - q_2^2 + q_3^2\right) \end{bmatrix}$$

These depend on the asteroid orientation and the pose of the camera corresponding to that image, respectively. Figure 6 illustrates the effect of different camera poses and asteroid orientations on the resulting

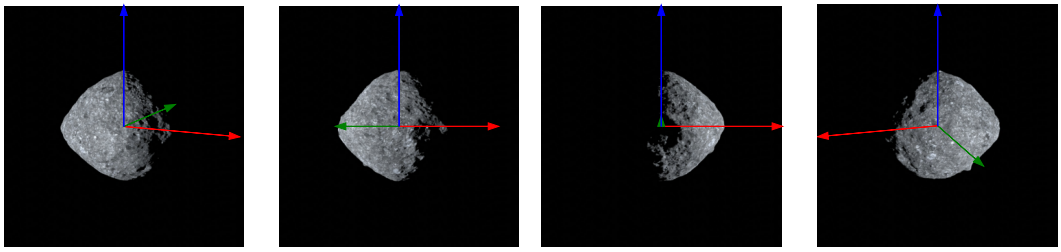


Figure 6: Sample images with varying $\mathbf{C}_{C,A}$ through different camera pose and asteroid orientation combinations, where red, green, and blue refer to the X,Y, and Z-axis of the asteroid reference frame, respectively

Table 2: The settings used for the generation of the dataset

Parameter	Description	Value(s)
R	The radius of icosphere, i.e., relative distance between camera and target	$[4.5; 6; 7.5; 9]$ km
n	Refinement order of the icosphere, i.e., the number and spacing of the vertices	3
min deg	The minimal value for the bearing angles, α, β for the generation of off-nominal pointing cases	1.0°
N	The number of off-nominal pointing cases per vertex	3
ψ	The asteroid reference frame orientation w.r.t world axes	$0 + k \cdot 60$ for $k = 1, 2, ..., 5$

relative pose.

B. Textured 3D model

As mentioned, the target asteroid is Bennu. There are several different 3D models available of Bennu, as it is common for asteroid missions that the shape model becomes more refined while the mission progresses, i.e., during different mission phases, different 3D models are available. The most detailed publicly available shape model, which has a 75-centimeter resolution (v20)[f], i.e., a resolution of 75 cm per vertex, is used in this work, although any other model would result in the same process. However, this 3D model does not have any texture applied, which is required for image generation.

As of 26 June 2021, a normal albedo map of Bennu is publicly available[g]. A normal albedo map represents the innate reflectance of the surface and has zero phase angle, i.e., it is an image texture without any shadows or highlights. However, this albedo map only covers the region between ± 55° latitude. Moreover, the images that were used to generate this albedo map may contain noise and other artifacts. Because the albedo map is incomplete it would be required to use the global basemap mosaic[h] for the higher latitudes. However, this global basemap was designed to highlight the surface morphology and has a phase angle of 30°. It therefore has shadows and highlights baked into the image. When such an image is used as the image texture, the object would then have two shadows, one caused by the synthetic light source and one that was already baked into the image. Therefore, when synthetically rendering images, an albedo texture map is vital to avoid unrealistic synthetic images.

[f] `https://www.asteroidmission.org/updated-bennu-shape-model-3d-files/`, Date accessed: 16-12-2020

[g] `https://astrogeology.usgs.gov/search/map/Bennu/OSIRIS-REx/OCAMS/Bennu_global_ShapeV20_GndControl_ROLOphase_ALBEDO_8bit_v6`, Date accessed: 16-8-2021

[h] `https://www.asteroidmission.org/bennu_global_mosaic/`, Date accessed: 16-8-2021

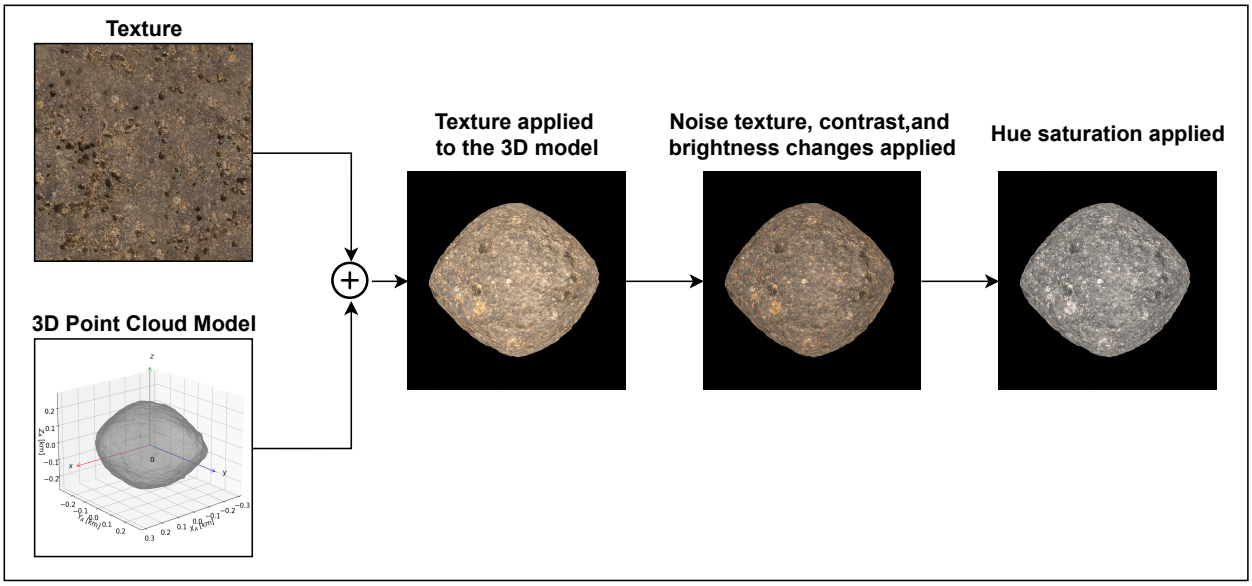


Figure 7: Illustrating the process of creating the textured 3D model in Blender

Consequently, to avoid unrealistic scenes, a texture was created and applied to the 3D model in Blender. The 3D model is imported to Blender using the +Y-forward and +Z-up convention, which determines the initial orientation of the asteroid, i.e., which side is illuminated. A rocky texture was downloaded from PolyHaven[i] and adjustments to this standard texture, shown in Fig. 7, were made in Blender to mimic an actual surface of an asteroid. A noise texture is applied to randomize the rocky texture and mimic white and dark spots randomly spread over the surface. Furthermore, contrast and brightness changes were applied alongside Hue saturation, which further changes the saturation and brightness.

This final texture deviates from the exact texture of Bennu, but this does not influence the machine learning algorithm too much, as it can be made invariant to texture and illumination conditions, this will be elaborated upon in subsection D. Moreover, the advantage of using a created texture is that it is not limited by the resolution of the available maps.

C. Annotations required for training the CNNs

This subsection discusses the two important aspects of the dataset, bounding boxes and keypoints, that have to be annotated to allow for the training of the object and keypoint detection networks, respectively.

1. Bounding box

The object detection network detects the object within the image and regresses the coordinates of the bounding box surrounding the target. The ground-truth bounding box coordinates are required to train the object detection network and have been determined during the synthetic dataset generation using Blender. These ground-truth bounding-box coordinates have been relaxed by 5% of the original width and height, as Chen et al.[37] demonstrated that by relaxing the ground-truth bounding box coordinates by a small margin the object detection training was improved. The following bounding box representation (x_{min}, y_{min}, w, h) is used, following the COCO format, where x_{min} and y_{min} refer to the coordinates of the top-left corner of the bounding box and w and h refer to the width and height of the bounding box, respectively, as can be observed in Fig. 8. However, when necessary, these encodings can easily be interchanged.

2. Keypoints

The designated keypoints need to be annotated to allow for the training of the keypoint detection network. These k designated keypoints are transformed from the 3D model space to the 2D image space using the PnP

[i] `https://polyhaven.com/textures/rock/natural`, Date accessed: 16-8-2021

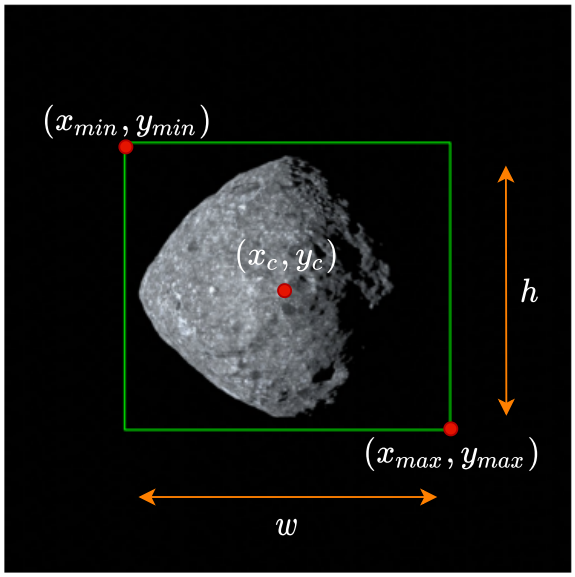


Figure 8: Visualization of the different bounding box encoding parameters

equation, Eq. (3), to derive the ground-truth locations of these k keypoints. During the synthetic dataset generation, the ground-truth camera pose data $\mathbf{P} \in \mathbb{R}^{3\times4}$, $\left[\begin{array}{c|c} \mathbf{C}_{C,A} & \mathbf{r}^{C}_{A/C} \end{array}\right]$, and the camera intrinsic parameters K are known and used to convert the keypoints from 3D to 2D.

D. Augmented dataset

CNNs trained solely on synthetic imagery are prone to the *domain gap*, which refers to the gap in the performance of the CNN on synthetic images compared to actual space imagery. Bridging this domain gap and achieving robustness of the CNN, trained on synthetic images, is crucial in creating deep-learning systems that can be deployed in safety-critical applications. Steps were taken within this work to address the domain gap and mimic the actual space environment as much as possible, however, this is a distinct and topical research field both within general machine learning,[42–44] as well as machine learning for space applications.[20,45]

A dataset including image augmentations was generated to increase the robustness of the trained CNNs against common image corruptions originating from real sensors or the space environment, relating to, e.g., radiation effects, pluming events, and over/underexposure. Furthermore, it was used to make the CNNs more robust against different textures and illumination conditions.[45] The textures in the real images can be different from the training examples, as the real sensors might capture different textures or the actual surface properties are not known or cannot be modeled properly. The major advantage of using such a training process, i.e., augmented dataset, is that it allows the network to become robust to a variety of augmentations, textures, and illumination conditions with minimum effort, i.e., the exact properties of the target or sensor do not have to be modeled accurately. An additional pipeline created in Python[j] was used to augment the original dataset generated using Blender. The following corruptions were used, grouped for clarity: i) Gaussian, motion, defocus, and zoom blur, ii) Gaussian, impulse, shot, and speckle noise, iii) Spatter, color jitter, and random erase.

The augmentation pipeline randomly applies the different corruptions to an image with a certain probability. An image can have up to four augmentations applied, but never more than one type of blur or noise per image, to avoid creating images that are not representative of realistic corrupted imagery. A total of 27499 images were generated and split into training (82%) and validation (18%). 38% of the images was augmentation free, 25% had one augmentation, 26% had two augmentations, 10% had three, and 1% had four augmentations. Figure 9 illustrates the effect of the different augmentations in isolation and Table 3

[j] `https://github.com/kuldeepbrd1/image-corruptions` Date accessed: 4-11-2021

Table 3: Listing the number of images per corruption type

Description	Training	Validation
No augmentations	8762	1819
Random erase	1902	339
Color jitter	7133	1699
Gaussian blur	1683	368
Motion blur	1728	398
Zoom blur	1732	414
Defocus blur	1776	425
Spatter	1801	434
Gaussian noise	1806	359
Impulse noise	1761	346
Speckle noise	1727	350
Shot noise	1840	305
Total	22646	4853

lists the number of images per corruption type. The network's robustness against image corruptions, representative of real image artifacts and the space environment, is evaluated through the use of the corrupted synthetic validation dataset, which does not have the exact same distribution as the training set, as can be observed in Table 3. Moreover, the augmentation applied to each image is unique to mimic real-world corruptions, which also show variation of the corruption values even at fixed levels of intensity. This also ensures that not only the distribution of augmentations differs between the training and validation set, but also the corruption value.

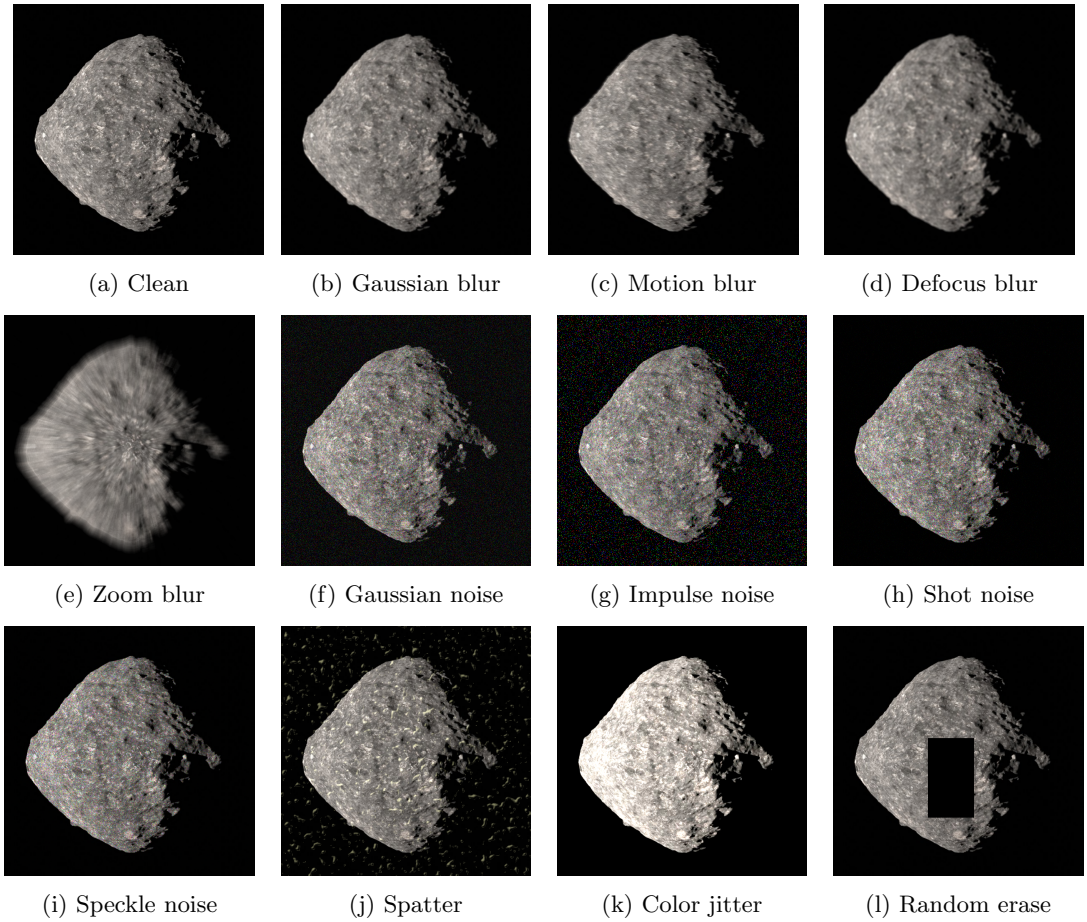
V. Algorithm

This section discusses the pose estimation pipeline developed in this work. This serves as an overview of how all the different parts fit together and which software is used and developed for each respective part. A top-level description of the developed pipeline is discussed in Subsec. A, whereas Subsec. B details the CNNs used and Subsec. C elaborates on the training, validation, and testing procedures for the networks.

A. Pose estimation pipeline

The main goal of the proposed monocular autonomous vision-based learning algorithm is to take a single 2D image and output an estimate of the distance of the camera to the asteroid's center of mass. A model-dependent feature-based learning approach is used, which consists of a CNN-based feature detector and a PnP solver in sequence, as shown in Fig. 10. This means that a CNN replaces a traditional IP algorithm and is used to detect n pre-defined features (keypoints) within a 2D image, which were designated offline on the 3D model, after which the 2D-3D correspondence can be used to estimate the distance to the asteroid by solving the PnP problem. This *feature-based* approach followed by a separate pose estimation step outperformed *end-to-end* CNN architectures in which a single CNN replaced the entire pipeline and thereby learning the complex non-linear relation between the 2D image and the pose directly.[18] Furthermore, it allows for easier incorporation in existing navigation architectures.[19] This pipeline relies only on the availability of a 3D model of the target. Furthermore, it does not require an initial state estimate and is completely software-based. The weights and biases of the CNNs can be updated and sent to the spacecraft throughout the mission, if required.

The usage of a CNN for feature extraction has several advantages compared to hand-engineered IP algorithms: 1) they are more robust against illumination conditions and image noise,[17] 2) the keypoints can be selected/designated offline so their 2D-3D correspondence is known; this avoids the cumbersome and computationally intensive matching of detected 2D features, e.g., Harris corners, to their location on the 3D model using methods such as Random sample consensus (RANSAC),[36] 3) the keypoint-based CNNs output



(a) Clean (b) Gaussian blur (c) Motion blur (d) Defocus blur

(e) Zoom blur (f) Gaussian noise (g) Impulse noise (h) Shot noise

(i) Speckle noise (j) Spatter (k) Color jitter (l) Random erase

Figure 9: The different image corruptions that have been incorporated into the augmented dataset. The severity for some augmentations have been increased for visibility

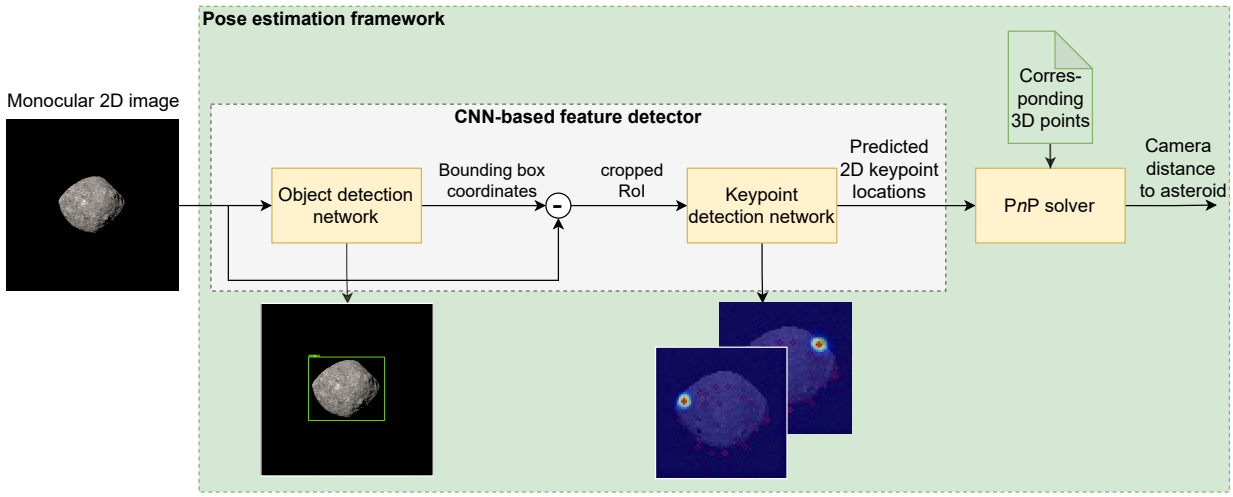


Figure 10: The proposed CNN-based pose estimation architecture

predicted detections for all the designated keypoints, even when they are occluded or not directly visible (e.g., at the back of the target).[46] This is because it learns the inherent spatial relationship between all the different keypoints.

Keypoints can either have a physical meaning, such as craters or boulders on the surface, or not, and be a mathematically determined point that stands out with respect to its surroundings. However, the use of physical features relies on the presence of these on the asteroid, which might not be the case. In this work, the keypoints were designated on the 3D model using the 3D Scale-Invariant Feature Transform (SIFT) algorithm [k]. This is an adaptation of the 2D SIFT algorithm for 3D point clouds, which estimates SIFT points based on the z-gradient of the 3D points. The advantage of designating surface keypoints is that they are closely related with the models features. A total of 68 keypoints distributed over the surface were designated on the 3D model and can be observed in Fig. 11 alongside the projected 2D keypoints for a given pose. The designation of 68 keypoints has the benefit that the network does not necessarily have to predict all of them perfectly well within every image, as for the subsequent pose estimation only a minimum of four points are required. The increase in training effort for 68 keypoints compared to a lower number is minimal.

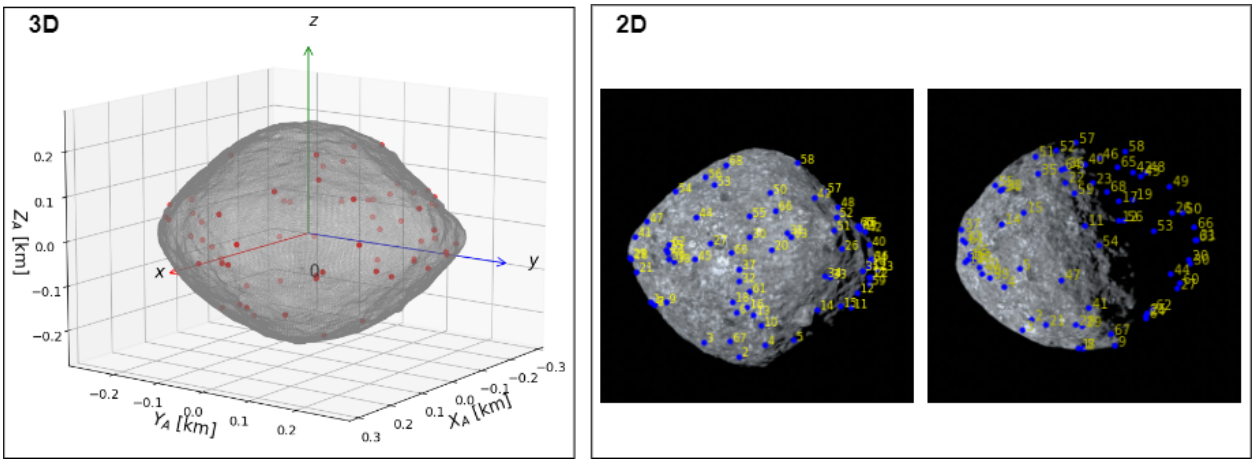


Figure 11: Illustrating the designated keypoints in 3D and examples of their 2D projections for a given pose

As shown in Fig. 10, a top-down approach is used, which consists of an object detection (OD) network in front of the keypoint detection (KD) network, making the CNN pipeline more robust to the scale of the object within the image, which often troubles traditional IP algorithms, allowing for accurate keypoint detections.[17] This OD network detects the object within the image and regresses the bounding box coordinates encompassing the object. This bounding box is used to crop the region of interest (RoI) of the original image. This RoI image is resized to match the input size of the keypoint detection network, which convolves the input image and outputs heatmap predictions around the pre-selected keypoints. The 2D pixel coordinates of the keypoint location correspond to the heatmap's peak intensity, where the shape and the intensity characterize the confidence of detecting the corresponding keypoint at that location.[47] Figure 12 shows heatmaps with different confidence levels. This heatmap approach resulted in better performance compared to the direct regression approach[48,49] and it allows for the extraction of statistical information (covariance matrix) regarding the detection uncertainty.[34] Furthermore, by selecting a keypoint detection network that outputs heatmap predictions, the pipeline is more robust to the domain gap, as proven by.[20]

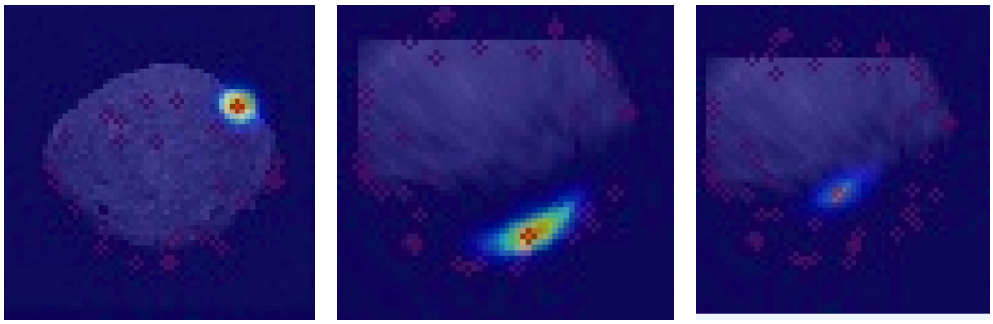
The extracted locations of the detected 2D keypoints are then fed to the EPnP solver alongside their corresponding 3D locations on the target. It returns an estimate of the pose of the camera with respect to the asteroid. However, in this work only the distance estimate is considered.

This pipeline can easily be incorporated into any navigation architecture. As the CNN guarantees predictions for all n designated keypoints it overcomes challenges faced by traditional feature-tracking approaches for which the number of detections can deviate, resulting in highly-variable measurements sent to the filter. Moreover, a covariance matrix quantifying the uncertainty of the detections (measurements) derived from the heatmaps can be used to improve the filters robustness and allowing for ease of implementation.

B. Convolutional Neural Networks

The best practice within computer vision is to use an open-source implementation of a network architecture that has been optimized for a given task, e.g., object or keypoint detection. The selection of the CNN

[k] `https://github.com/sjtuytc/betapose`, Date accessed: 25-11-2021



(a) Confident detections (b) Less confident detections (c) Bad detections

Figure 12: Visualization of different types of keypoint heatmap detections

architecture determines the accuracy and robustness of the respective object or keypoint detections. However, for the actual application of machine learning to space missions not only the performance is important, but also the computational effort (floating points operations (FLOPs)) and the required memory (number of parameters) to run these networks in inference on the spacecraft processing unit.

State-of-the-art *object detection networks* can generally be divided into *two-stage* and *single-stage* detectors. *Two-stage* detectors such as Faster R-CNN[50] are generally more accurate, but slower than *one-stage* detectors such as SSD.[51] These networks are trained and evaluated on challenging datasets, such as COCO, containing a plethora of different object classes with images containing multiple objects and often heavily occluded scenes. The datasets used within this work consist of a single class object on a black background (underlit starfields) with little to no occlusion. Soviany and Ionescu[52] showed that for such relatively easy images a single-shot detector performs equally well compared to the more complex and slower two-stage Faster R-CNN. Therefore, given the relative simplicity of the dataset the most important consideration in selecting the object detection network was the efficiency. Based on this, the highly efficient one-shot detector, SSD-FPN-MobileNetV2-Lite, which is available through the TensorFlow 2 Detection Model Zoo, is adapted within this work. Recent research within spacecraft pose estimation using CNN architectures also used one-shot detectors.[36,53] Moreover, by using the multi-scale SSD-MobileNetV2-FPN-Lite object detection network the pipeline is more robust against image corruptions as was proven by Hendrycks and Dietterich.[43] The network is adapted using the TensorFlow Object Detection API. An input size of 320×320 px is used, which is slightly larger than the input size of the subsequent keypoint detection network to minimize loss of information.

Keypoint detection is a challenging computer vision problem and the research revolving around keypoint detection is mostly linked to human pose estimation, detecting the joints of person instances within images. However, the problem of detecting *pre-defined* keypoints is essentially the same throughout different applications and levels of abstraction. Therefore, these network architectures are considered suitable for this work and were adapted to the purpose. The selection of the CNN architecture determines the accuracy of the keypoint detections.

Newell et al.[54] proposed an hourglass architecture that downsamples the input and subsequently upsamples the input to detect features at different resolutions (scales). Chen et al.[55] proposed the Cascaded Pyramid Network (CPN), which uses a similar approach as the hourglass structure, but concatenates the feature maps from different scales to arrive at the final heatmap output. Furthermore, Xiao et al.[56] proposed a simple architecture based on the hourglass that replaces the upsampling operations by deconvolutional layers, which combines convolution and upsampling, thereby changing the way the high resolution feature maps are obtained. However, currently the state-of-the-art keypoint detection network on the COCO dataset is the High Resolution Network (HRNet) proposed by Sun et al.[57] This network uses certain sub-networks in parallel that each have a different resolution, compared to the aforementioned networks that have different resolutions in series. The network then shares information between the different resolution feature maps (multi-scale) fusion to improve the heatmaps precision. The HRNet has been used for keypoint detection on uncooperative spacecraft.[37,45,53]

However, these networks have a relatively large number of FLOPs and parameters, requiring a substantial amount of memory, making them unsuitable for resource-limited devices. Currently, research is being

performed on the development of lightweight networks for keypoint detection, making them suitable for embedded devices. However, this research is relatively new and not as established as for the object detection networks. Zhang et al.[58] proposed the Lightweight Pose Network (LPN), which is based on the architecture proposed in.[56] The performance of the LPN is comparable to other state-of-the-art networks on the challenging COCO dataset, but it is able to achieve this with only a fraction of the parameters and FLOPs, i.e., 81% fewer parameters and 80% fewer FLOPs compared to the HRNet. Therefore, from an accuracy-efficiency point of view, which is in line with the desired lightweight properties required for implementation on space hardware, the LPN (ResNet-101) was selected. The LPN is written in PyTorch[l] and is based on the HRNet[m] repository, which has a structured format of files and functions, allowing for a structured approach in adapting the network to keypoint detection on asteroids. Furthermore, initial results indicated that the LPN outperformed the HRNet on the asteroid dataset.

The input to the keypoint detection network is the cropped RoI, detected by the object detection network. The aspect ratio of the input size of the network is used during the cropping to avoid deformations in the cropped images. This means that for a network input size of 256×256 px, i.e., aspect ratio of 1, a bounding box with a width and height of 500 px and 480 px, respectively, will be cropped to 500×500 px, around the center of the bounding box. This cropped RoI is then rescaled to the desired input size. During the rescaling, the scaling parameters, namely the center and scale, are preserved. The input size directly affects the computational demand of the network and an input size of 256×256 px was found to work best for the dataset.

C. Training, validation, and test

During training, the validation dataset is used to test the generalization performance of the network and avoid overfitting. The networks have been optimized in isolation. The object detection network is initialized using the weights and biases of the model that has been trained on the COCO 2017 dataset. The network is trained for 50,000 steps with a mini-batch size of 32 images using the momentum optimizer with a cosine learning rate decay with warmup. The initial learning rate α is 0.0266 and this rate increases for a 1000 "warmup" steps until is reaches the base learning rate of 0.08, which then gradually decreases to zero using a cosine decay until it reaches 50,000 steps. The momentum hyperparameter β is set to 0.9. The random scaling/cropping augmentation, referring to randomly cropping a part of the image with a certain aspect ratio and area and then rescaling it back to the original image size, is used. This makes the network more invariant against the scale of the target object within the image. Furthermore, the random horizontal flip augmentation is employed.

The performance of the object detection network is assessed through the Intersection-over-Union (IoU) metric:

$$\text{IoU} = \frac{\text{size of intersection}}{\text{size of union}} = \frac{A \cap B}{A \cup B} \tag{8}$$

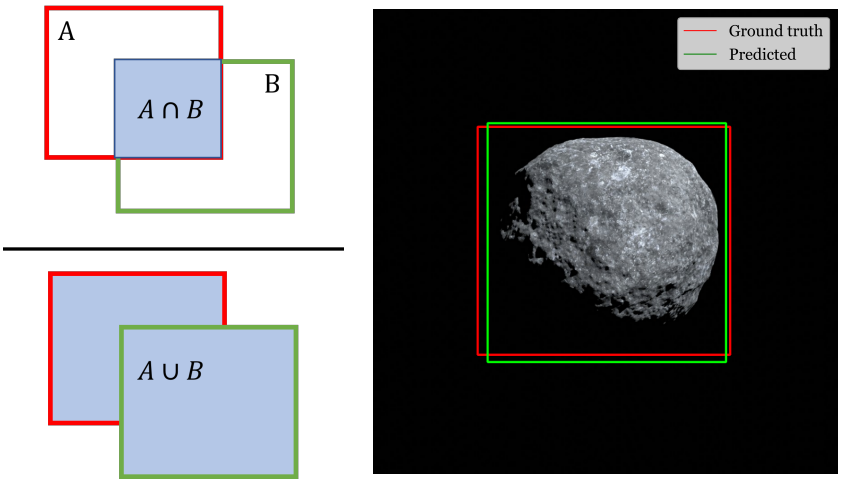
Figure 13 visualizes this metric and illustrates its physical meaning with respect to this dataset.

The keypoint detection network is initialized using the weights and biases of the model that has been trained on the COCO train2017 dataset. The network is trained for 150 epochs using the Adam optimizer[59] with a mini-batch size of 64 images from the training dataset, whereas the model is validated after completion of one *epoch* using the images in the validation dataset. The learning rate α decays using a multi-step schedule and starts with a base learning rate of 10^{-3} and is changed to 10^{-4} at epoch 90 and to 10^{-5} at epoch 120, which is then kept constant until the end epoch of 150. Furthermore, following the original authors of the LPN model,[58] an iterative training strategy was used to test whether the performance could be improved even further. This means that in the first stage the network is trained for 150 epochs, after which the best model determined in stage 0 is used to initialize the model for stage 1, which then restarts the training from epoch 60 using the same *hyperparameter* settings, such as the scheduled learning rate.

Random scaling, random horizontal flip, and random rotation are the affine augmentations used. Random scaling refers to multiplying the scale parameter as previously discussed with a scaling factor. The scaling factor is set to 0.3, which implies that the scale parameter is multiplied by a randomly selected value within the range $[0.7; 1.3]$. The random horizontal flip augmentation is used with a probability of 0.5. Random rotation applies a random rotation to the original image orientation with the rotation factor is set to 40°. This means that the image is randomly rotated by an angle within the range $[-80°, 80°]$. The probability of

[l] `https://github.com/zhang943/lpn-pytorch`, Date accessed: 25-11-2021

[m] `https://github.com/leoxiaobin/deep-high-resolution-net.pytorch`, Date accessed: 25-11-2021



(a) Visualization of the IoU performance metric

(b) A dataset example for a detection with an IoU of 0.907

Figure 13: Illustrating the IoU performance metric

this augmentation is set to 0.6. During training, validation, and testing of the keypoint detection network the ground-truth bounding boxes are used to optimize the network in isolation. However, when evaluating the entire pipeline during inference, the cropped RoI is used as input. Further detailed information regarding the set-up of the object and keypoint detection networks can be found in the files made available by the author.

The predicted keypoint's 2D location is given by $(\hat{u}, \hat{v})$, which is then compared to the ground-truth location (u, v). The prediction can have an error in both u and v and therefore the total error for a single detection i is calculated using the Euclidean distance:

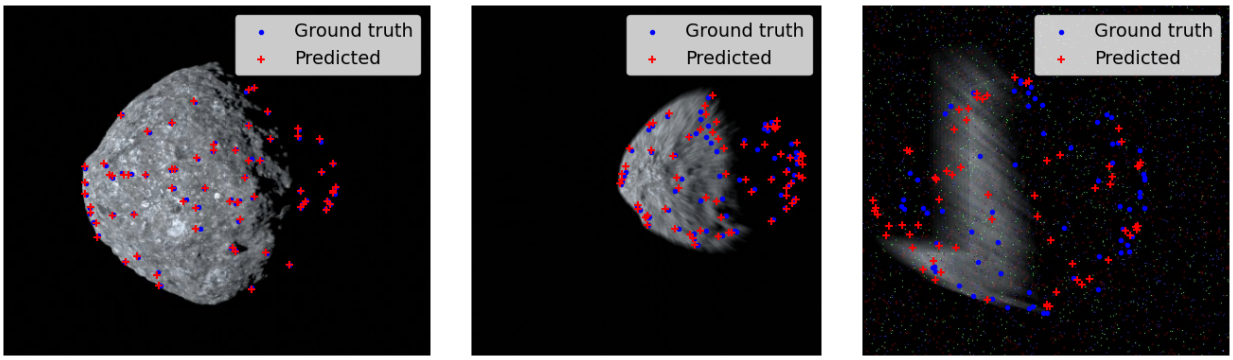
$$E_{px,i} = \sqrt{|\hat{u}_i - u_i|^2 + |\hat{v}_i - v_i|^2} \tag{9}$$

However, the network outputs n detections for every image k, so the performance of the keypoint detection network for any given image k is assessed using the following:

$$\text{ME,k} = \frac{\sum_{i=1}^{n_{\text{tot}}} \sqrt{|\hat{u}_i - u_i|^2 + |\hat{v}_i - v_i|^2}}{n_{\text{tot}}}. \tag{10}$$

The performance of the keypoint detection network over the entire dataset is evaluated using the mean and median mean error (ME) over all the images. The order of the error of the keypoint detection network depends on the input size of the original image. The image size used in this work is 1024×1024 px, however, to allow for comparison of the performance of the network against different input sizes, the performance is normalized using the scaling parameters as previously discussed. This *input-normalized* error provides a consistent reference of the keypoint detection network's performance across datasets. However, the accuracies stated within this work refer to the non input-normalized errors, unless stated otherwise.

The CNNs were trained and evaluated on both the clean and augmented datasets, resulting in four different combinations, e.g., trained on clean, evaluated on augmented, etc. However, the network trained solely on clean images did not generalize well to the augmented dataset and had a large mean and median ME of 59 px and 12 px, respectively. The model trained on augmented images achieved a mean and median ME of 3.7 px and 3.0 px on the augmented test set and 3.3 px and 2.84 px on the clean test set. This indicates that this network is able to accurately detect the keypoints within the images, even when they contain challenging illumination conditions, brightness/contrast changes, and other image corruptions. This illustrates that it relies more on the global shape of the object and the inherent spatial relationship between the keypoints. Generally, the more accurate the keypoint detection, the more accurate the resulting pose estimation. Figure 14 shows some examples of keypoint detections for different levels of accuracy. The object detection models trained on the augmented dataset also achieved 100% of feasible detections (IoU ¿ 0.75),



(a) $E_{px} = 1.54$ px (b) $E_{px} = 12.5$ px (c) $E_{px} = 158$ px

Figure 14: Examples of high (a), low (b) and failed (c) detection accuracies under the influence of image corruptions and other effects from the augmented dataset

which refers to detections that can effectively be used by the subsequent keypoint detection network. The aforementioned indicates the importance of training procedures in creating robust models and therefore only the models trained on the augmented dataset are used for the pipeline.

VI. Results

This section focuses on the evaluation of the performance of the entire pose estimation pipeline as shown in Fig. 10, which consists of the CNN-based feature detection part and the EPnP solver. The pipeline has been trained on the augmented images dataset. The distance estimation results are presented for the clean and augmented test sets, both consisting of 4583 images.

A. Metric

As previously discussed, the object and keypoint detection networks have been trained and evaluated independently. This pose estimation pipeline uses the bounding box coordinates outputted by the object detection network to crop the RoI, which is fed to the keypoint detection network. The KD network then predicts the locations of the keypoints and feeds those points to the EPnP solver alongside the corresponding 3D points, based on which the distance of the asteroid with respect to camera is calculated.

The keypoint detection network outputs predictions for all 68 designated keypoints, however, some of the keypoints locations are not within the image (Fig. 3). The predictions for the visible keypoints within the image are sent to the EPnP solver. The EPnP solver only needs a minimum of 4 keypoints to be able to produce a unique solution. The performance of the pose solver, however, increases when more, accurate detections are available ($n > 6$).

The accuracy of the pose estimate was found to deteriorate slightly when utilizing the detections of all the visible keypoints within the image. This is explained by the fact that the detection confidence can deviate between keypoints within an image, as some are more challenging to recognize from a given camera pose and illumination condition (Fig. 12). Using inaccurate detections by the pose solver leads to poorer pose estimations. Therefore, a heuristic approach was implemented that selects the user-defined n most confident keypoint detections, where, as before, the confidence level is equal to the peak value of the heatmap. This approach does not set a 'general' confidence threshold on the detection accuracy for all images, but bases the selection of keypoints on the 'difficulty' of the image.

The pipeline is evaluated by analyzing the translational error, which is calculated using the following, where $\mathbf{t}^C$ and $\hat{\mathbf{t}}^{\mathbf{C}}$ represent the ground-truth and estimated translational position vector, respectively:

$$E_t = \left|\hat{\mathbf{t}}^{\mathbf{C}} - \mathbf{t}^{\mathbf{C}}\right| \tag{11}$$

An accuracy requirement for the performance of the designed algorithm was set: *The pose estimation pipeline shall have a relative line-of-sight distance to the center of mass of the target with a knowledge error*

Table 4: The results of the pose estimation pipelines using the *39* most confident keypoints

Pipeline	Training/evaluation dataset	Mean $\mathbf{E}_t$ [m]	Mean $\|\mathbf{E}_t\|$ [m]	Median $\mathbf{E}_t$ [m]	Median $\|\mathbf{E}_t\|$ [m]	Failure cases [#/%]
SSD-LPN-EPnP	Bennu+/Bennu	$\begin{pmatrix} 2.608 & 2.065 & 42.53 \end{pmatrix}$	42.82	$\begin{pmatrix} 1.709 & 1.358 & 29.62 \end{pmatrix}$	29.86	0/0
SSD-LPN-EPnP	Bennu+/Bennu+	$\begin{pmatrix} 3.077 & 2.501 & 47.96 \end{pmatrix}$	48.32	$\begin{pmatrix} 1.889 & 1.469 & 31.09 \end{pmatrix}$	31.36	1/0.021

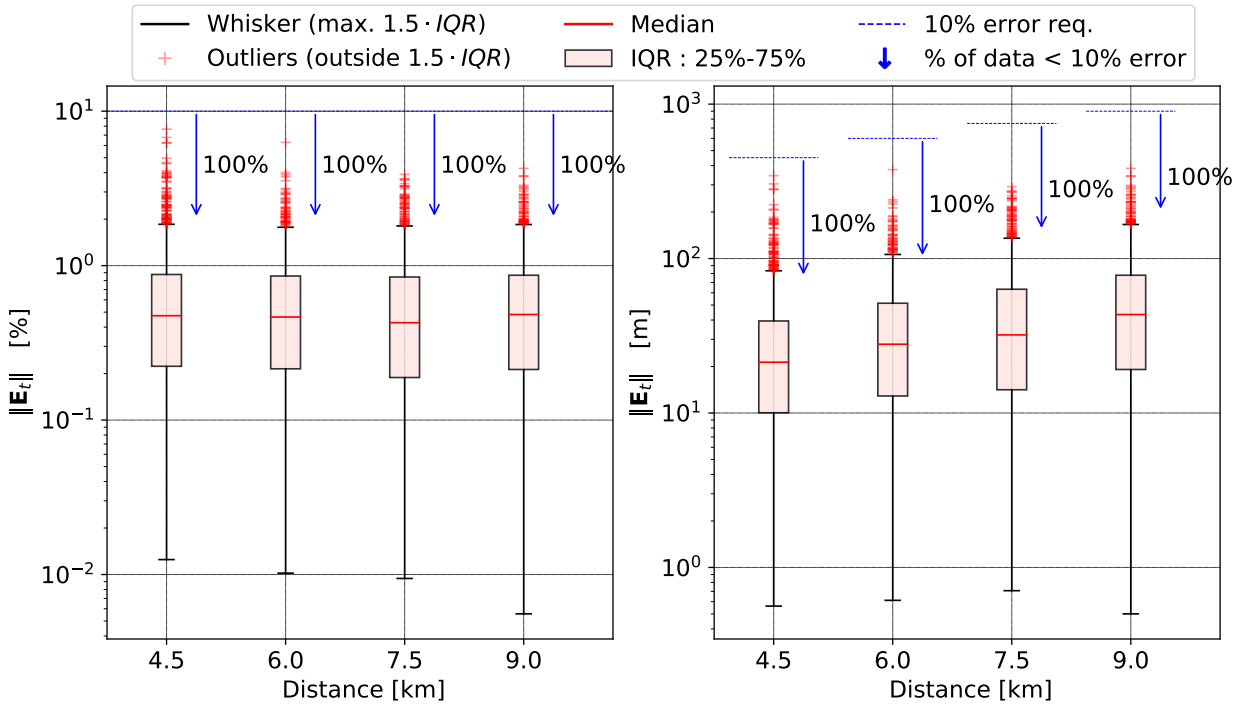


Figure 15: Performance of the pipeline that has been trained and evaluated on the Bennu+ and Bennu dataset, respectively, the outliers make up 5.23% (254 images) of the total of 4853 images

lower than 10% of the real distance with 99.73% probability at 90% confidence level.

B. Accuracy assessment

The performance of the pipeline is shown in Table 4 and Figs. 15 and 16, which show the distance estimation error as a function of distance. The 39 most confident keypoints were used for each image, as this resulted in the lowest mean error across the dataset. The following can be observed:

- The pipeline trained on the augmented dataset Bennu+ is able to achieve highly accurate performance on both the clean and augmented dataset with a mean *line-of-sight* distance estimation of around 42 m and 48 m, respectively, and a median distance estimation of around 30 m and 31 m, respectively, at a confidence level of 90%. This performance was achieved on the large relative range to the target of 4.5-9 km. The closer to the asteroid the more accurate the performance with a median error of around 22 m from a distance of 4.5 km. Furthermore, the developed system satisfied the accuracy requirement of $< 10\%$ knowledge error with respect to the ground-truth distance in 99.979% and 100% of the cases for the Bennu+ and Bennu dataset, respectively.
- Comparing the performance of the pipeline on the clean images of the Bennu dataset, with the performance obtained on the augmented dataset, it can be seen that a lower mean and median translational error $\|\mathbf{E}_t\|$ of 42.82 m and 29.86 m can be obtained. The images present in the clean dataset do not contain augmentations and are therefore less challenging, resulting in improved performance of the overall pipeline.

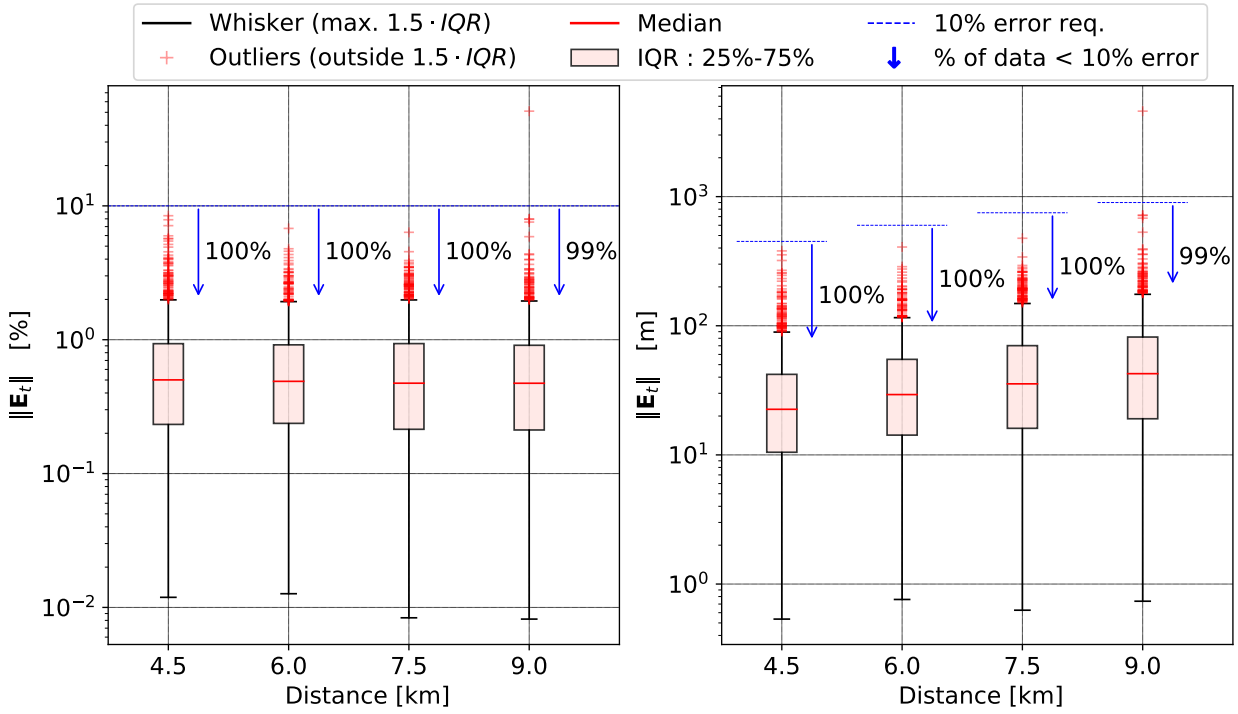


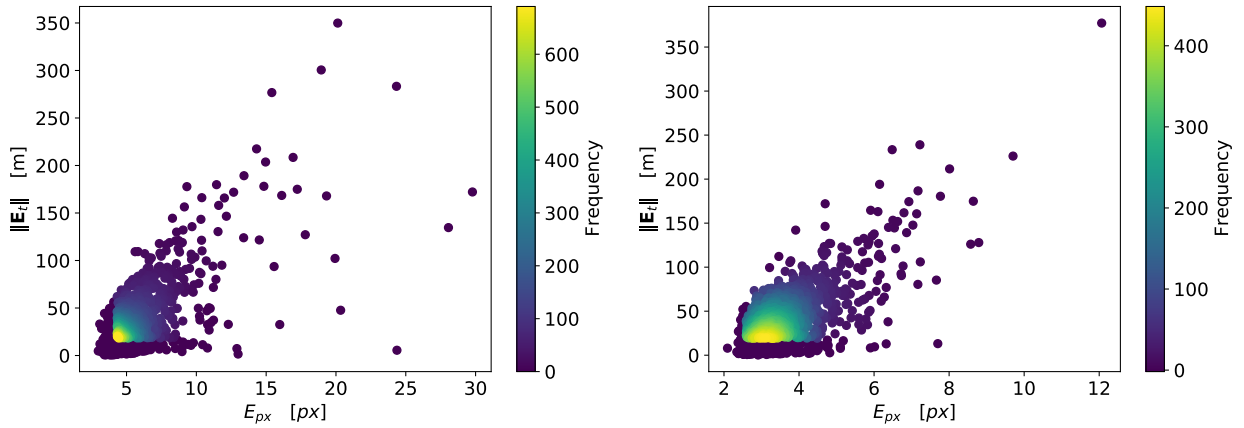
Figure 16: Performance of the pipeline that has been trained and evaluated on the augmented dataset Bennu+, the outliers make up 5.96% (289 images) of the total of 4853 images

- Both Figs.15 and 16 show that the position error increases with increasing distance. However, this is in accordance with the behavior of pose solvers such as the EPnP.[38] The performance of these pose solvers deteriorate when the distance along the optical axis is increased and this trend is typically observed in monocular pose estimation systems.[18,36,45,60] Estimating the range from 2D imagery is challenging and this is caused by the nonlinear relation existing between z^C and the pixel location of the detected keypoints. This means that the pose solver is sensitive to pixel errors persisting in the keypoint detection when the relative distance increases, resulting in inaccurate pose estimates. This can also be observed in Fig. 17, where the average keypoint detection error E_{px} for larger distances are generally more accurate. However, a relatively large pixel error has a much larger effect on the resulting pose error for these large relative distances. Furthermore, for all distances it can be observed that in general, a lower keypoint prediction error results in a lower pose estimation error.

- The mean and median translational error achieved in this work is similar to comparable pose estimation pipelines for uncooperative spacecraft on the Envisat dataset[34] and the SPEED dataset.[18] Where the mean and median distance error is adjusted for the differences in relative range existing between the different datasets and the ones in this work. Furthermore, the proposed pipeline has a position error estimate of around 0.4%, whereas another CNN-based navigation system proposed in[25] are able to achieve a position error below 5% of the range relative to the asteroid. This indicates the superiority of the proposed method within this work.

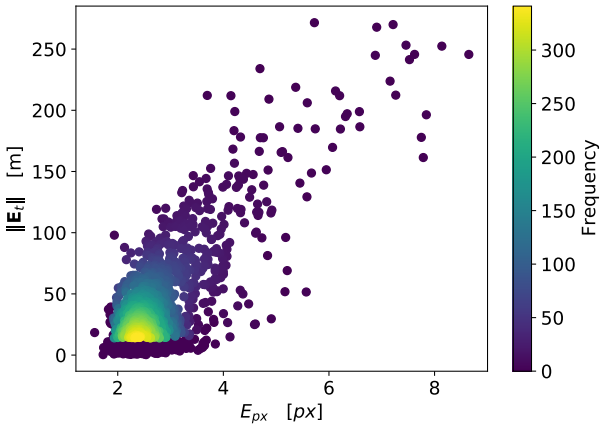
The main take-aways of the results of the pipeline are listed below:

- This pipeline is able to accurately estimate the instantaneous position of the camera and does not depend on an initial pose estimate, making it suitable for *lost-in-space* scenarios. This CNN-based architecture can be used to accurately navigate at distances between 4.5 km and 9 km from the asteroid. Furthermore, as it provides an instantaneous position estimate, it can also be used to (re)-initialize an *unknown feature tracking* (relative navigation) approach that allows navigation from distances closer to the asteroid.

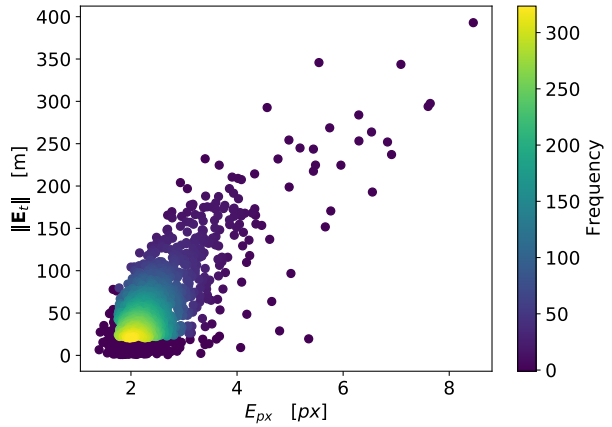


(a) Distance = 4.5 km

(b) Distance = 6 km



(c) Distance = 7.5 km



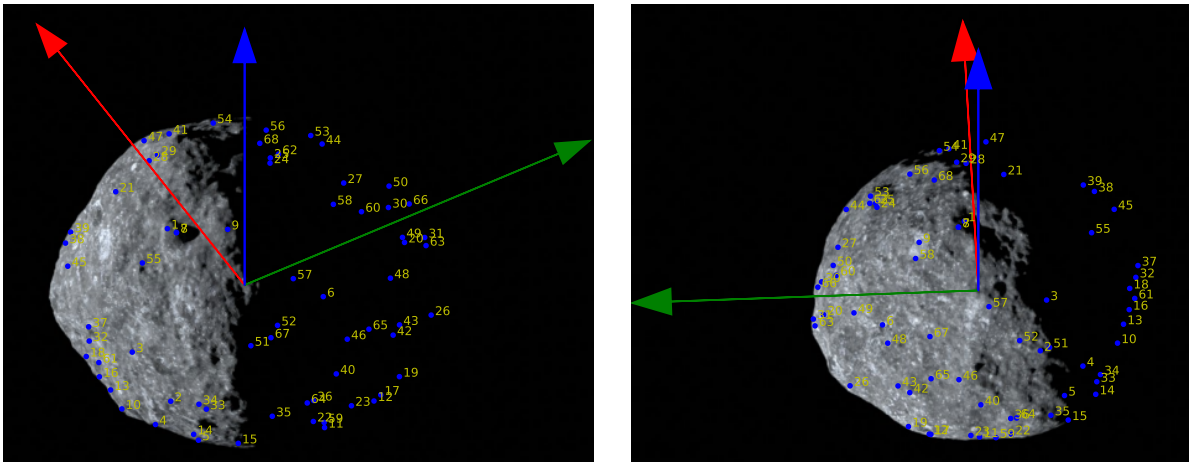
(d) Distance = 9 km

Figure 17: The relationship between the average keypoint detection error per image E_{px} and the resulting pose estimation error $\|\mathbf{E}_t\|$ evaluated using the test set of the Bennu dataset

- The synthetically-trained CNN-based feature extractor, consisting of the object and keypoint detection network, has proven to provide accurate distance estimates for a wide range of relative geometries between the camera and the asteroid. Moreover, it appeared to be robust against illumination conditions, occlusions, textures, and image corruptions. This is especially relevant for the application of the CNN to space applications, as the space environment is characterized by extreme contrast and low Signal-to-Noise ratio.

- The employed training procedure and datasets are crucial in achieving this robustness, and the results stress the importance of the data augmentations used in the augmented Bennu+ dataset. The object and keypoint detection networks trained solely on clean images failed to generalize well to these corruptions, i.e., lacking the robustness required for safety-critical applications. Through the use of an augmented dataset, this robustness to illumination conditions, occlusions, textures, and image corruptions can be achieved with a minimal effort, as it is not required to model the exact surface textures, encountered illumination conditions, and sensors. Furthermore, this training procedure achieves the aforementioned highly accurate results without depending on the availability of accurate information regarding the target body's properties, which is often required for hand-engineered IP algorithms.

- The CNN-based pipeline developed in this work achieved great performance, while using only a fraction of the parameters and FLOPs of other state-of-the-art deep learning networks and pipelines, making it suitable for implementation on space hardware.



(a) (b)

Figure 18: Visualization of the distribution of the keypoints for high inclination poses with $x^W \approx 0$

C. Error analysis

The CNN-based pipeline developed in this work achieved great performance, however, it is also important to find and uncover possible trends underlying cases in which the position error is fairly large, i.e., outliers. The outliers make up 5-6% (250-290 images) of the respective datasets, clean and augmented. These outliers are the result of poor keypoint detection, which were found to occur in the following cases:

- About 80% of the outliers occurred on off-nominal pointing cases, meaning that the asteroid is not in the center of the image. This off-nominal pointing results in part of the asteroid being cut off, as shown in Fig. 3, and less information, such as global shape or texture, for the network to base its prediction on resulting in inaccurate keypoint detections.

- The distribution of outliers per distance to the asteroid is similar, which demonstrates that the overall performance for each distance is similar and that the CNN detector simply struggles with some of the most challenging images for each distance.

- The majority of outliers has camera poses from high inclinations, both above and below the asteroid. This results in challenging illumination conditions and asteroid viewing orientations. Furthermore, around 30% of the outlier images have an camera position that lies on the concentric circle of $x^W = 0$ (Fig. 4) and as such, these images may show similarities when viewing the asteroid from the top or bottom, i.e., the symmetry problem. This problem is common in keypoint detection, as the keypoints that the network must predict, may appear the same as other keypoints, resulting in a reduced detection accuracy.[46] Furthermore, the designated keypoints on the asteroid appear predominantly on the rim of the asteroid when viewed from high inclinations (including directly above and below) as can be observed in Figs. 18a and 18b, whereas from different views the keypoints are more spread over the asteroid as shown in Fig. 11. The combination of high inclination, off-nominal pointing, and challenging illumination conditions could result in the slightly worse performance of the network on these images.

- Specifically for the augmented dataset, the CNN detector may fail on heavily corrupted images with a combination of different augmentations as shown in Fig. 14 (c). However, in reality it is highly unlikely that such combinations of effects will occur simultaneously.

In conclusion, the CNN detector predominantly struggles with challenging camera poses that have high inclinations and challenging illumination conditions combined with off-nominal pointing. However, the pipeline can still produce satisfactory position estimates for these cases and would allow a spacecraft to safely navigate, even under the influence of these challenging conditions.

VII. Conclusions

This work developed a first-of-a-kind CNN-based pose estimation pipeline suitable for autonomous navigation around asteroids. The choice of a top-down feature-based approach, consisting of an object detection and keypoint detection network in sequence, with a pose solver proved to produce accurate results. This means that a CNN is used to detect pre-defined keypoints on the 3D model within a 2D image after which the 2D-3D correspondence can be used to estimate the distance to the asteroid by solving the PnP problem. This method only relies on the availability of a 3D model of the target and can provide accurate *line-of-sight* distance estimates for a range of 4.5 km to 9 km from the asteroid. The pose estimation pipeline has been proven to produce accurate results for a variety of different camera viewpoints and distances and has shown to be robust against illumination conditions, occlusions, textures, and image corruptions. This robustness was achieved through the selection of the architecture of the pipeline and through the training procedure and datasets. This pipeline is able to produce an estimate of the instantaneous position (pose initialization) and is thus useful for *lost-in-space* scenarios.

A major improvement of the developed CNN-based pipeline compared to other pipelines, which were designed for pose estimation of uncooperative spacecraft, is the usage of lightweight networks. The CNN-based pipeline has only a fraction of the parameters and FLOPs compared to other state-of-the-art deep-learning networks and pipelines. This is a crucial contribution of this work, showcasing that accurate and robust performance can be achieved using lightweight networks suitable for embedded devices.

The results in this work demonstrate the efficacy of using CNN-based architectures for navigation around small bodies and serve as a stepping stone for future research of this topic.

References

[1]Cheng, A. F., "Near Earth Asteroid Rendezvous: Mission Summary," Asteroid III, University of Arizona Press, 2002, pp. 351-366. https://doi.org/10.1109/ICCV.2017.322

[2]Mastrodemos, N. and Rush, B. and Vaughan, A. and Owen, W., "Optical Navigation For The Dawn Mission At Vesta", *Advances in the Astronautical Sciences*, Vol. 140, 2011, pp. 1739–1754

[3]Hashimoto, T. and Kubota, T. and Kawaguchi, J. and Uo, M. and Shirakawa, K. and Kominato, T. and Morita, Hi., "Vision-based guidance, navigation, and control of Hayabusa spacecraft - Lessons learned from real operation -," *IFAC Proceedings Volumes*, Vol. 43, No. 15, 2010, pp. 259–264. https://doi.org/10.3182/20100906-5-JP-2022.00045

[4]Ogawa, N., Terui, F., Mimasu, Y., Yoshikawa, K., Ono, G., Yasuda, S., Matsushima, K., Masuda, T., Hihara, H., Sano, J., Matsuhisa, T., Danno, S., Yamada, M., Yokota, Y., Takei, Y., Saiki, T., and Tsuda, Y., "Image-based autonomous navigation of Hayabusa2 using artificial landmarks: The design and brief in-flight results of the first landing on asteroid Ryugu," *Astrodynamics*, Vol. 4, No. 2, 2020, pp. 89-–103. https://doi.org/10.1007/s42064-020-0070-0

[5]Flandin, G., Polle, B., Lheritier, J., and Vidal, P., "Vision based navigation for autonomous space exploration," *2010 NASA/ESA Conference on Adaptive Hardware and Systems*, 2010, pp. 9—16. https://doi.org/10.1109/AHS.2010.5546273

[6]Gil-Fernandez, J., and Ortega-Hernando, G., "Autonomous vision-based navigation for proximity operations around binary asteroids," *CEAS Space Journal*, Vol. 10, No. 2, 2018, pp. 287—294. https://doi.org/10.1007/s12567-018-0197-5

[7]Opromolla, R., Fasano, G., Rufino, G., and Grassi, M., "A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations," *Progress in Aerospace Sciences*, Vol. 93, 2017, pp. 53—72. https://doi.org/10.1016/j.paerosci.2017.07.001

[8]Pasqualetto Cassinis, L., Fonod, R., and Gill, E., "Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100548. https://doi.org/10.1016/j.paerosci.2019.05.008

[9]Sharma, S., Ventura, J., and D'Amico, S., "Robust Model-Based Monocular Pose Initialization for Noncooperative Spacecraft Rendezvous," *Journal of Spacecraft and Rockets*, Vol. 55, No. 6, 2018, pp. 1414–1429. https://doi.org/10.2514/1.A34124

[10]Tortora, P. and Di Tana, V., "LICIACube, the Italian Witness of DART Impact on Didymos," *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, 2019, pp. 314–317. https://doi.org/10.1109/MetroAeroSpace.2019.8869672

[11]Goldberg, H., Karatekin, O., Ritter, B., Herique, A., Tortora, P., Prioroc, C., Gutierrez, B.G., Martino, P. and Carnelli, I., "The Juventas CubeSat in Support of ESA's Hera Mission to the Asteroid Didymos," *Small Satellite Conference*, Logan, UT, 2019

[12]Razgus, B., Mooij, E., and Choukroun, D., "Relative Navigation in Asteroid Missions Using Dual Quaternion Filtering," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 9, 2017, pp. 2151—2166.

[13]Rowell, N., Dunstan, M. N., Parkes, S. M., Gil-Fernández, J., Huertas, I., and Salehi, S., "Autonomous visual recognition of known surface landmarks for optical navigation around asteroids," *The Aeronautical Journal*, Vol. 119, No. 1220, 2015, pp. 1193–1222. https://doi.org/10.1017/S0001924000011210

[14]Lorenz, D. A., Olds, R., May, A., Mario, C., Perry, M. E., Palmer, E. E., and Daly, M., "Lessons learned from OSIRIS-REx autonomous navigation using natural feature tracking," *2017 IEEE Aerospace Conference*, Institute of Electrical and Electronics Engineers (IEEE), Big Sky,MT, 2017, pp. 1—12. https://doi.org/10.1109/AERO.2017.7943684

[15]Pellacani, A., Graziano, M., Fittock, M., Gil, J., and Carnelli, I., "HERA vision based GNC and autonomy," *8th European Conference for Aeronautics and Space Sciences (EUCASS)*, Madrid, Spain, 2019. https://doi.org/10.13009/EUCASS2019-39

[16]D'Amico, S., Benn, M., and Jørgensen, J. L., "Pose Estimation of an Uncooperative Spacecraft from Actual Space Imagery," *International Journal of Space Science and Engineering*, Vol. 2, No. 2, 2014, pp. 171—189. https://doi.org/10.1504/IJSPACESE.2014.060600

[17]Pasqualetto Cassinis, L., Fonod, R., Gill, E., Ahrns, I., and Gil-Fernández, J., "Evaluation of tightly- and loosely coupled approaches in CNN-based pose estimation systems for uncooperative spacecraft," *Acta Astronautica*, Vol. 182, 2021, pp. 189–202. https://doi.org/10.1016/J.ACTAASTRO.2021.01.035

[18]Kisantal, M., Sharma, S., Park, T. H., Izzo, D., Martens, M., and D Amico, S., "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 5, 2020, pp. 4083–4098. https://doi.org/10.1109/TAES.2020.2989063

[19]Song, J., Rondao, D. and Aouf, N.,"Deep learning-based spacecraft relative navigation methods: A survey," *Acta Astronautica*, Vol. 191, 2022, pp. 22–40. https://doi.org/10.1016/J.ACTAASTRO.2021.10.025

[20]Park, T. H., Märtens, M., Lecuyer, G., Izzo, D., and D'Amico, S., "SPEED+: Next Generation Dataset for Spacecraft Pose Estimation across Domain Gap," arXiv preprint arXiv:2110.03101, 2021.

[21]Harl, N., Rajagopal, K. and Balakrishnan, S. N., "Neural Network Based Modified State Observer for Orbit Uncertainty Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 4, 2013, pp. 1194–1209. https://doi.org/10.2514/1.55711

[22]Pugliatti, M. and Topputo, F., "Small-Body Shape Recognition with Convolutional Neural Network and Comparison with Explicit Features Based Methods," *2020 AAS/AIAA Astrodynamics Specialist Conference*, AAS 20-515, 2020, pp. 1–20.

[23]Ravani, K., Mathavaraj, S. and Padhi, R., "Site Detection for Autonomous Soft-Landing on Asteroids Using Deep Learning," *Transactions of the Indian National Academy of Engineering*, Vol. 6, No. 2, 2021, pp. 365–375. https://doi.org/10.1007/s41403-021-00207-0

[24]Mancini, P., Cannici, M. and Matteucci, M., "Deep learning for asteroids autonomous terrain relative navigation," *Advances in Space Research*, 2022. https://doi.org/10.1016/j.asr.2022.04.020

[25]Pugliatti, M. and Topputo, F., "Navigation About Irregular Bodies Through Segmentation Maps," *31st Space Flight Mechanics Meeting*, 2021, pp. 21-383.

[26]Shi, J.-F., Ulrich, S., and Ruel, S., "CubeSat Simulation and Detection using Monocular Camera Images and Convolutional Neural Networks," *2018 AIAA Guidance, Navigation, and Control Conference*, AIAA, Grapevine, TX, 2018. https://doi.org/10.2514/6.2018-1604

[27]Sharma, S., and D'Amico, S., "Pose Estimation for Non-Cooperative Spacecraft Rendezvous Using Neural Networks," *29th AAS/AIAA Space Flight Mechanics Meeting*, Ka'anapali, Maui, HI, 2019.

[28]Alimo, R., Jeong, D., and Man, K., "Explainable Non-Cooperative Spacecraft Pose Estimation using Convolutional Neural Networks," *AIAA Scitech 2020 Forum*, AIAA, Reston, VA, 2020. https://doi.org/10.2514/6.2020-2096

[29]Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[30]Manning, J., Langerman, D., Ramesh, B., Gretok, E., Wilson, C., George, A., Mackinnon, J., and Crum, G., "Machine-Learning Space Applications on SmallSat Platforms with TensorFlow," *Small Satellite Conference*, 2018.

[31]Black, K., Shankar, S., Fonseka, D., Deutsch, J., Dhir, A., and Akella, M. R., "Real-Time, Flight-Ready, Non-Cooperative Spacecraft Pose Estimation Using Monocular Imagery," *arXiv preprint arXiv:2101.09553*, 2021.

[32]Volpe, R., Sabatini, M., Palmerini, G. B., and Mora, D., "Testing and Validation of an Image-Based, Pose and Shape Reconstruction Algorithm for Didymos Mission," *Aerotecnica Missili & Spazio*, Vol. 99, No. 1, 2020, pp. 17–32. https://doi.org/10.1007/s42496-020-00034-6

[33]Harvard, A., Capuano, V., Shao, E. Y., and Chung, S.-J., "Spacecraft Pose Estimation from Monocular Images Using Neural Network Based Keypoints and Visibility Maps," *AIAA Scitech 2020 Forum*, AIAA, Orlando, FL, 2020.

[34]Pasqualetto Cassinis, L., Fonod, R., Gill, E., Ahrns, I., and Gil Fernandez, J., "CNN-Based Pose Estimation System for Close-Proximity Operations Around Uncooperative Spacecraft," *AIAA Scitech 2020 Forum*, AIAA, Orlando, FL, 2020.

[35]Lepetit, V., Moreno-Noguer, F., and Fua, P., "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision*, Vol. 81, No. 2, 2009, pp. 155—166. https://doi.org/10.1007/s11263-008-0152-6

[36]Park, T. H., Sharma, S., and D'Amico, S., "Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft," *2019 AAS/AIAA Astrodynamics Specialist Conference*, Portland, ME, American Astronautical Society (AAS), 2019.

[37]Chen, B., Parra, , Cao, J., Parra, A., and Chin, T.-J., "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, South Korea, 2019, pp. 2816—2824. https://doi.org/10.1109/ICCVW.2019.00343

[38]Sharma, S., and D'Amico, S., "Comparative assessment of techniques for initial pose estimation using monocular vision," *Acta Astronautica*, Vol. 123, 2016, pp. 435—445. https://doi.org/10.1016/j.actaastro.2015.12.032

[39]Movshovitz-Attias, Y., Kanade, T., and Sheikh, Y., "How useful is photo-realistic rendering for visual learning?" *arXiv*, Vol. arXiv:1603.08152, 2016.

[40]Sierks, H., Keller, H. U., Jaumann, R.,Michalik, H., Behnke, T., Bubenhagen, F., Büttner, I., Carsenty, U., Christensen, U., Enge, R., Fiethe, B., Gutiérrez Marqués, P., Hartwig, H., Krüger, H., Kühne,W.,Maue, T.,Mottola, S., Nathues, A., Reiche, K. U., Richards, M. L., Roatsch, T., Schröder, S. E., Szemerey, I., and Tschentscher, M., "The Dawn framing camera," *Space Science Reviews*, Vol. 163, No. 1-4, 2011, pp. 263–327. https://doi.org/10.1007/s11214-011-9745-4

[41]Hinterstoisser, S., Benhimane, S., Lepetit, V., Fua, P., and Navab, N., "Simultaneous Recognition and Homography Extraction of Local Patches with a Simple Linear Classifier," *Proceedings of the British Machine Vision Conference 2008*,British Machine Vision Association, 2008, pp. 10.1-–10.10.

[42]Jackson, P. T., Atapour-Abarghouei, A., Bonner, S., Breckon, T. P., and Obara, B., "Style Augmentation: Data Augmentation via Style Randomization," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019, pp. 83–92.

[43]Hendrycks, D., and Dietterich, T. G., "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," arXiv preprint arXiv:1903.12261, 2019.

[44]Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W., "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," arXiv preprint arXiv:1811.12231, 2018.

[45]Pasqualetto Cassinis, L., Menicucci, A., Gill, E., Ahrns, I., and Gil-Fernández, J., "On-Ground Validation of a CNN-based Monocular Pose Estimation System for Uncooperative Spacecraft," *8th European Conference on Space Debris*, 2021.

[46]Zhao, Z., Peng, G., Wang, H., Fang, H.-S., Li, C., and Lu, C., "Estimating 6D Pose From Localizing Designated Surface Keypoints," arXiv preprint arXiv:1812.01387, 2018.

[47]Pavlakos, G., Zhou, X., Chan, A., Derpanis, K. G., and Daniilidis, K., "6-DoF Object Pose from Semantic Keypoints," arXiv preprint arXiv:1703.04670, 2017.

[48]Tompson, J., Jain, A., LeCun, Y., and Bregler, C., "Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation," *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Vol.1 , MIT Press, Cambridge, MA, 2014, pp. 1799—1807.

[49]Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z., "Rethinking the Inception Architecture for Computer Vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Institute of Electrical and Electronics Engineers (IEEE), Las Vegas, NV, 2016, pp. 2818—2826. https://doi.org/10.1109/CVPR.2016.308

[50]Ren, S., He, K., Girshick, R., and Sun, J., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv preprint arXiv:1506.01497, 2016.

[51]Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C., "SSD: Single Shot MultiBox Detector," arXiv preprint arXiv:1512.02325, 2016.

[52]Soviany, P., and Ionescu, R. T., "Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction," arXiv preprint arXiv:1803.08707, 2018.

[53]Barad, K. R., Robust Navigation Framework for Proximity Operations around Uncooperative Spacecraft (MSc Thesis), Delft University of Technology, 2020.

[54]Newell, A., Yang, K., and Deng, J., "Stacked Hourglass Networks for Human Pose Estimation," *Computer Vision - ECCV 2016. Lecture Notes in Computer Science*, Vol. 9912, edited by B. Leibe, J. Matas, N. Sebe, and M.Welling, Springer, 2016, pp. 483–499.

[55]Chen, Y.,Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J., "Cascaded Pyramid Network forMulti-person Pose Estimation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Institute of Electrical and Electronics Engineers (IEEE), 2018, pp. 1703–1712.

[56]Xiao, B.,Wu, H., andWei, Y., "Simple Baselines for Human Pose Estimation and Tracking," *Computer Vision - ECCV 2018*, 2018, pp. 472—487. https://doi.org/10.1007/978-3-030-01231-1_29

[57]Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D.,Mu, Y.,Wang, X., Liu,W., andWang, J., "High-Resolution Representations for Labeling Pixels and Regions," arXiv preprint arXiv:1904.04514, 2019.

[58]Zhang, Z., Tang, J., and Wu, G., "Simple and Lightweight Human Pose Estimation," arXiv preprint arXiv:1911.10346, 2019.

[59]Kingma, D. P., and Lei Ba, J., "Adam: A Method For Stochastic Optimization," Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.

[60]Sharma, S., and D'Amico, S., "Reduced-Dynamics Pose Estimation for Non-Cooperative Spacecraft Rendezvous using Monocular Vision," *40th Annual AAS Guidance and Control Conference*, Breckenridge, CO, 2017.