Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research 71 (2023) 3748–3760

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Deep learning for asteroids autonomous terrain relative navigation

Pierpaolo Mancini[1], Marco Cannici[1], Matteo Matteucci[1]

Politecnico di Milano, Via Giuseppe Ponzio 34/5, Milan 20133, Italy

Received 30 December 2021; received in revised form 6 March 2022; accepted 7 April 2022
Available online 19 April 2022

Abstract

The future of humanity in space will require, more and more frequently, proximity operations with unexplored celestial bodies. In particular, asteroids and comets will have a crucial role in the future space economy and exploration. These bodies are often characterized by unknown terrain maps and lack of navigation infrastructure; this makes autonomous navigation challenging to accomplish. In this context, visual matching algorithms are not able to perform navigation if the map and the images captured online by the probe differ significantly for illumination conditions, scaling or rotation. To overcome these issues, in this work, we propose a siamese convolutional neural network capable of image matching and a position retrieval system for reliable autonomous navigation. The system is robust to image noise, reusable on multiple terrains and landing sites, and it does not require deploying any additional hardware component. In this work, the OSIRIS-REx Nasa's mission was taken as reference for defining the navigation requirements and a 3D model of Bennu has been built to render training data. The image matching capabilities of the system have been tested on one validation dataset made of rendered images and one made of real images provided by NASA's OSIRIS-REx mission. Besides, realistic descent scenarios have been simulated to test the system navigation accuracy in simulated but realistic conditions, and to evaluate the error recovery capabilities of the developed system. The system achieved mission compliant navigation accuracy on both real and simulated terrain maps, showing remarkable generalization capability highlighting the generality of the proposed solution.

Keywords: Deep learning; Asteroid landing; Terrain relative navigation; Convolutional neural network; Siamese convolutional neural network; Optical navigation; Artificial intelligence

1. Introduction

The celestial bodies traveling in our solar system represent undoubtedly an important resource, both in purely economic terms and for the enormous scientific relevance of the discoveries that could come from their exploration and study. As a result, the future of humanity in space will require, more and more frequently, spacecrafts capable of performing autonomous operations with little to no human intervention.

Among these, one of the fundamental operations that need to be reliably executed for a successful mission is terrain relative navigation. As such, navigation technologies built by space organizations for prior asteroid landing missions, including NASA's OSIRIS-REx and the Hayabusa I and II missions, all required a significant amount of human labor and specialized hardware to achieve a high degree of navigation precision. For instance, when preparing the probe for descent, either natural terrain features had to be manually selected and 3D modeled or critical hardware deployed on the asteroid surface, potentially affecting the mission timeline. While the first solution is mission-specific and must be redesigned for each landing mission,

[1] Department of Systems and Industrial Engineering, Polytechnique of Milan, Milan 20133, Italy.
E-mail addresses: pierpaolo.mancini@mail.polimi.it (P. Mancini), marco.cannici@polimi.it (M. Cannici), matteo.matteucci@polimi.it (M. Matteucci)

the second relies on external critical hardware that has limited on-board availability and may fail.

In this regard, artificial intelligence offers a great opportunity for the space industry to achieve unprecedented levels of machine autonomy. In this work, we propose to alleviate the human effort as well as the hardware required to prepare and execute a landing mission by leveraging deep learning techniques. We design a system capable of retrieving the position of a spacecraft relative to a celestial body's surface that only utilizes images of the underlying terrain captured by on-board cameras. The system is general, meaning that it can be employed unchanged for any landing mission, and it does not require additional human intervention for its design and deployment. We accomplish this by integrating a neural module within the traditional relative position retrieval system based on RGB cameras. Instead of searching the area flown over by the spacecraft directly within the landing map, as typically done by other navigation systems, we perform the search into an embedding space which we train to be invariant from environmental conditions. This results in a system robust to illumination changes and that does not rely, to operate, on accurate information regarding the current attitude and position of the spacecraft.

We conducted experiments in a simulated environment built using the data provided by the NASA OSIRIS-REx mission. The proposed solution is completely software-based, therefore it could be deployed and updated later in the mission, and it does not rely on additional hardware components as the artificial landmarks in the Hayabusa I and II missions. In all the performed tests, the developed system shows mission compliant navigation accuracy and the conducted experiments underline the robustness of the system to different sources of image noise together with good error recovery capabilities.

This document is structured in 4 sections. We start by outlining previous space navigation systems used in past asteroid landing missions together with the first use of a CNN for terrain relative navigation. Then, in the following section, we present a formalization of the terrain relative navigation problem and the implementation details of the proposed system, together with its main element of innovation. We then move to our testing and validation approach, which was structured to test each component of the proposed system either individually and combined. In this section each test is described together with the used accuracy metrics, the requirements we developed, and the obtained results in the individual tests and on the overall navigation simulations. We conclude this document with a review of the main contribution and conclusions of our research, with an overview of its potential future development.

2. Previous space missions & related works

In this section, past navigation systems used in the same context of our research are briefly described, together with an overview of the machine learning techniques used in this work.

2.1. Navigation systems in previous space missions

According to a 2008 survey from NASA's Jet Propulsion Laboratory (Johnson and Montgomery, 2008), the main sensing techniques for optical terrain relative navigation can be divided into *active range sensing* (i.e., LIDAR sensors, altimeters) and *passive imaging* (i.e., optical cameras). This work focuses on passive imaging, as it only exploits cameras as primary source of data.

Cameras are mature sensors that can be easily integrated inside a probe, even of small size, and require low power to operate. Their altitude operating range is wider compared to LIDAR sensors and they can be used for both orbital navigation and proximity operations (Opromolla et al., 2017). On the other hand, cameras can not operate in the darkness, therefore a minimal amount of illumination is required for optical camera navigation to be performed.

In the history, there have been only five asteroid landing missions. All of them were characterized by a phase of relative velocity cancellation, orbit determination, and descent till probe's touch down on the asteroid surface. Among past asteroid landing missions, the OSIRIS-REx of NASA (Lorenz et al., 2017), and the Hayabusa I and II missions (Hashimoto et al., 2010), distinguished themselves for the use of a real-time terrain relative navigation system performing onboard navigation during the descent. These missions achieved a greater level of landing accuracy compared to Rosetta's Philae lander and NEAR-Shoemaker mission (Ulamec et al., 2015; Antreasian et al., 2001).

Hayabusa I and II missions. The system developed by the JAXA - Japan Aerospace Exploration Agency for the Hayabusa missions (Kawaguchi et al., 2008; Watanabe et al., 2017) is based on *Artificial Landmark tracking*, or ALT (Ogawa et al., 2020). During the landing phase, an artificial landmark is dropped onto the asteroid surface when the probe is under 100 meters. This object is taken as a reference point during the landing procedure to adjust the trajectory of the lander, and cancel the velocity relative to the asteroid's surface. The accuracy achieved by the Hayabusa missions is incredibly high. In the Hayabusa 2, the first touchdown occurred just a few meters away from the target point (Ogawa et al., 2020). However, this approach has some drawbacks. It can only be used for surface proximity navigation and not orbital navigation, it relies on additional hardware components that can potentially fail, and it is limited by the number of artificial landmarks the probe can carry.

OSIRIS-REx mission. An alternative to ALT has been used by NASA for the OSIRIS-REx mission (Lauretta et al., 2017). The system was called *Natural Features Traking* or NFT and instead of using artificial landmarks, it used natural features of the asteroid's surface as reference points for pose estimation (Lorenz et al., 2017). In this context, landmarks are natural formations that can be reliably extracted from an image due to their distinctive characteristics. NFT works using a catalog of natural 3D landmarks, each of which is associated with a known coordinate point. Each feature in the catalog is pre-selected by the ground team on Earth and uploaded into the spacecraft after a long iterative process of candidate feature selection, data acquisition to build its digital model, and testing. During the descent, a real-time rendering engine integrated onboard renders the landmarks to match the light direction and orientation based on the current probe's position. The renderings are searched inside the real images taken by the onboard cameras during the descent. When a match is found, since each feature has known coordinates, the position of the probe is updated accordingly. For the matching, a *normalized cross correlation* algorithm is used (Lorenz et al., 2017). The features' catalog is terrain-specific and it can be built only when the probe reaches the asteroid orbit. It must be rebuilt for each mission, and depending on its characteristics, it might be rebuilt if the landing site changes, affecting the mission timeline.

2.2. Previous CNN-based navigation approaches

In recent years machine learning has proven its effectiveness in image analysis, especially after the introduction of deep learning algorithms and convolutional neural networks. Many traditional image recognition algorithms and also machine learning techniques for image analysis were based on handcrafted feature extractors designed based on the researcher's domain knowledge (Fujiyoshi et al., 2019).

The novelty brought by deep learning was to introduce an unsupervised feature extractor capable of learning the most relevant features to be used by the neural network. Convolutional neural networks (CNNs) are a type of deep learning neural network that is made of a series of convolutional layers which convolve the input data with a set of K kernels, to produce a $K-dimensional$ array of feature maps. The kernels are learned during the training process and constitute the feature extractor part of the neural network. The output of the feature extractor can then be fed into a series of fully connected layers depending on the specific architecture and purpose of the neural network (Litjens et al., 2017).

Classification. The use of deep learning techniques for terrain relative navigation was first proposed by Campbell et al. (2017). In this work a *Convolutional Neural Network* (CNN) is used to classify the spacecraft location within a 1024 × 1024 nadir image of the Apollo 11 landing site. This work proposes a simplified navigation system that only considers the motion of the probe along the horizontal axis. Each pixel in the middle row of the image is considered a class, and a 1024-way classification is performed to localize the probe. The results of this study were promising in terms of prediction accuracy, however, the biggest limitation of using a classifier is that it usually learns class-specific features and not general ones. Therefore, the model obtained with this approach is typically terrain-specific and not reusable on different terrains. It must be retrained for each asteroid and landing site with the data produced by the probe.

3. A novel CNN-based natural features traking system

In agreement with NASA OSIRIS-REx mission's design decision, we propose a software based system for asteroids autonomous terrain relative navigation. Our goal is to remove the need for a features catalog and a real time renderer, thus reducing the human effort and time to prepare the spacecraft for lading, as well as dedicated hardware as in the Hayabusa missions.

Inspired by scientific literature related to the progress of deep learning in computer vision applications (Nandy et al., 2020; O'Mahony et al., 2019; Fujiyoshi et al., 2019; Litjens et al., 2017) we decided to use a CNN to build a patch searching algorithm that does not require a pre-computed catalog of known natural features. This task is particularly challenging for traditional image matching algorithms, which are often not robust to illumination or scale variations. Indeed, Deep Learning techniques can achieve incredible results concerning image processing, especially for their robustness to various sources of noise.

3.1. Problem formulation

The idea behind the implemented approach is to address the relative navigation problem by framing it as a best match search between the central patch P_c of the image coming from the probe during the descent and the patches in the landing site map M. We decide not to perform this search in the image space, but rather, to build an embedding space invariant to environmental changes. In this paper we consider as a map a georeferenced nadir image of the asteroid surface, as shown in Fig. 1.

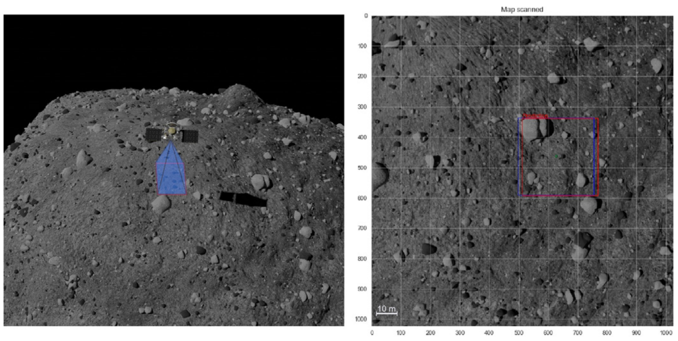

Fig. 1. With these two images, we propose a visualization of the navigation problem formulation. The left image shows an overview of the landing site (the size of the probe is raised for visualization purposes). The right image shows the corresponding nadir map, taken from an altitude of 200 meters with a pixel density of 0,13 m/pixel. In the two images, the blu square represents the correct location of the field of view of the probe, while the red square represents the prediction of our navigation system.

Given a map M of a landing area on the asteroid, it is possible to identify sub-images $s_i \in M$, we call them patches, that identify each location inside that landing area. To detect the correct coordinates for each s_i, we take inspiration from Campbell et al. (2017) but propose to substitute the classifier with an embedding learning approach based on *siamese convolutional neural networks* (Nandy et al., 2020). Given a sample s belonging to some data domain, computing its embedding means to describe relevant features of s with a vector of real numbers $e_s \in R^N$, as illustrated in Fig. 2. The vector e_s is called embedding of the sample s and, in our case, is obtained as $e_s = cnn(s)$.

Each embedding identifies a point inside an N-dimensional features space and the distance between points can be measured with the traditional L_2 norm. In this work, the CNN performing the embedding transformation is trained to cluster together similar images. Given two sample images p_1, p_2 and their embedding, $e_1 = cnn(p_1), e_2 = cnn(p_2)$ the Euclidean distance, $d(e_1, e_2)$, between the embeddings can be seen as a measure of the similarity of the corresponding input images.

The navigation system is based on the processing pipeline described in Fig. 3, that starts with the data acquisition and pre-processing phases. Afterward, the map of the selected landing site is scanned searching for high-similarity matchings with the current image acquired by the probe. In this phase, a similarity map is built over the landing site map, and it is weighted using a probability map built with odometry information to improve the prediction accuracy. From the resulting heat map it is possible to retrieve the areas that, according to the neural network, are more likely to frame the same area of the images visible from the probe.

3.2. System overview

Initial state setup. Before starting the descent procedure, a landing site is chosen and a nadir image of the selected area is acquired as the map of the landing area. Each pixel of the map is labeled with meaningful coordinates, either

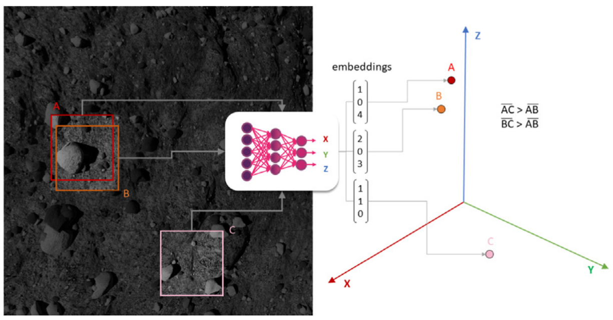
A
B
C
embeddings
1 0 4
2 0 3
1 1 0
X
Y
Z
A
B
C
$\overline{AC} > \overline{AB}$
$\overline{BC} > \overline{AB}$

Fig. 2. Images A, B, C, and D are given as input to the model, which computes an embedding vector. Each vector identifies a point in a hypothetical 3D features space. The considered CNN is trained to produce similar embedding of images framing the same area.

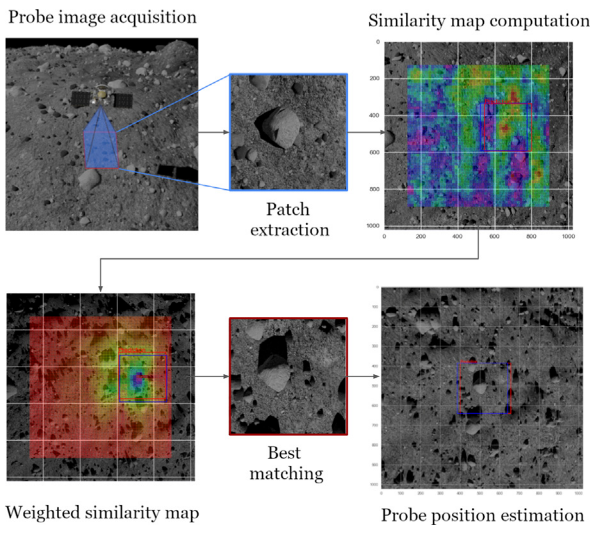

Fig. 3. The proposed navigation system pipeline is illustrated through its main steps. It is worth noticing that the captured image and the scanned map can be affected by different noise, such as the different illumination conditions of the extracted patch and the best matching patch showed in the picture.

relative to the target spot or the global asteroid coordinate system.

Data acquisition and pre-processing. The probe, during its descent, captures images of the underlying surface. From these images, only a central patch is extracted, allowing the size of the searched image to be set depending on the altitude of the spacecraft, and then it is digitally scaled to match the size of the input layer of the neural network (see Fig. 4). This dynamic scaling approach is used to increase the system robustness to the scale mismatch between the map and the images acquired by the spacecraft.

The extracted patch P_c is encoded using the neural network to obtain its embedding representation e_c. Indeed, the main component of the patch searching algorithm is a similarity function that receives as input a pair of two image

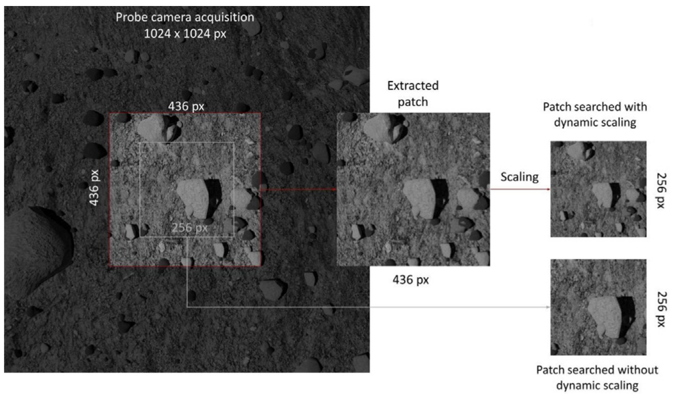

Fig. 4. Dynamic scaling patch extraction example. The size of the extracted patch depends on the distance from the surface. As can be seen, the extracted patch using the dynamic scaling contains more natural features than its counterpart extracted without dynamic scaling.

patches p_1, p_2 and produces a real value s quantifying how close the two input images are in the features space. The input images p_1 and p_2 are patches of size 256×256 pixels. The similarity value s between p_1 and p_2 is computed using the L_2 euclidean distance between the embedding vectors of the input images, $e_1 = cnn(p_1)$ and $e_2 = cnn(p_2)$ computed using a siamese convolutional neural network. The expected behavior of this similarity function is to obtain a similarity score smaller for pairs of images framing the same area than for images framing different locations, regardless of the resolution and scaling level, illumination conditions, and small rotations.

The patch searching algorithm scans the map extracting patches p_i of size 256×256 with a stride of 4 pixels. The extracted patches are encoded as embedding vectors $e_i = cnn(p_i)$ using the trained neural network. The distance between the embedding of the patch coming from the probe and the embedding of the patches extracted from the map is computed $s_i = d(P_c, e_i)$, normalized as a value in $(0, 1)$, and stored in a heat map M_s as shown in Fig. 5.

Patch searching & Similarity map.

Weighted similarity map & outliers detection. Ideally, the best matching patch could already be found inside M_s by taking the point with the maximum similarity score. However, as shown in Fig. 5, there is more than one area with high probability of being the target location of the spacecraft. This can happen because of the mismatch in resolution, the different zoom levels, illumination conditions, and small rotations. We call *outliers* the areas that receive a high similarity score but that do not correspond to the correct position of the searched patch P_c.

The key idea to detect and filter out the outliers is to use some prior function to refine the information coming from the similarity map. We propose to use the previous position of the spacecraft predicted by the system.

This information is modeled with a position map M_p built by assigning to each point in the map a value in {0, 1}, that is maximum in the current position of the probe and decreases with the distance from the current position of the probe. The value decreases radially according to a sigmoid function.

M_p is used to weight the similarity map computed in the previous step of the navigation algorithm, in order to produce a *weighted similarity map*, $M_{ws} = M_s \ * M_p$ as shown in Fig. 6. Due to the training strategy of the model, each point i of M_{ws} measures the likelihood that the patch p_i and the searched patch P_c are framing the same location, based on the similarity evaluation produced by the neural network and the distance from the last predicted position of the spacecraft. At the end of this step, given the map M_{ws} and its set of points S_p, the coordinate of the point with maximum likelihood is output as the predicted spacecraft position inside the reference map M

$$S_c = \max_{\{p_i \in S_p\}} (M_{ws}) \tag{1}$$

Additional odometry information could have been used; however, our idea was to keep at minimum the dependency on the current state and attitude of the probe to avoid compromising the navigation performance in case of a failure in their estimate.

3.3. Implementation details

In general, a siamese neural network is a particular type of convolutional neural network, which processes two, or

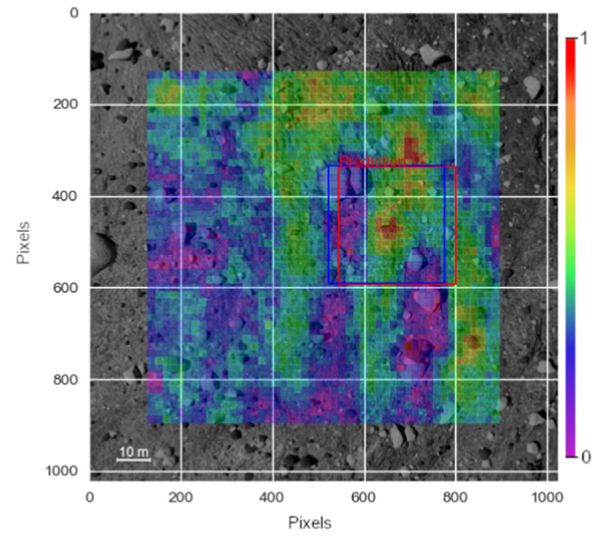

Fig. 5. Similarity heat map M_s taken from an altitude of 200 meters with a pixel density of 0,13 m/pixel. In each point i the similarity score s_i between the patch in that point and the searched patch. The red areas are the ones with the highest similarity score, which is computed as $s_i = 1 - d(p_{searched}, s_i)$, where d is the euclidean distance, normalized in $\{0, 1\}$, between the embedding of the searched patch and the embedding of the patch extracted in point i.

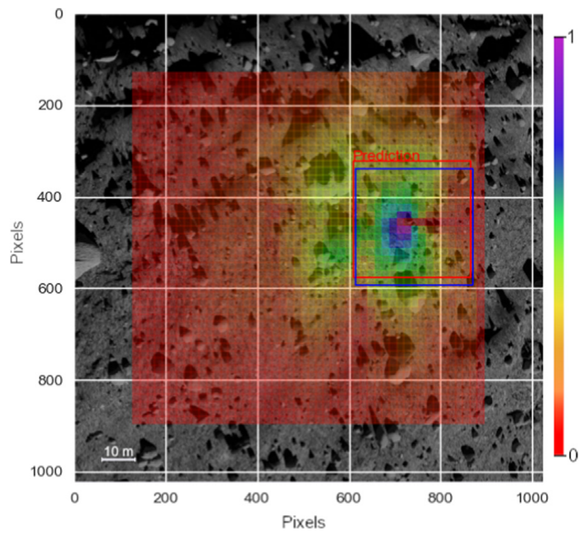

Fig. 6. Weighted similarity map M_{ws} produced by mixing the similarity map with the spacecraft position probability map built with odometry information. This image is taken from an altitude of 200 meters with a pixel density of 0,13 m/pixel. The red areas are those with zero probability of founding the probe, while the maximum probability is associated with purple areas.

more, distinct inputs in parallel twin branches in order to produce comparable output vectors. The training process typically exploits the a priori knowledge of the relationship that the outputs of the network should have, e.g., belonging or not to the same class, to learn an embedding function or a metric function.

There is no unique architecture describing all possible siamese neural networks (Zagoruyko and Komodakis, 2015; Leal-Taixé et al., 2016). Indeed, multiple architectures can be designed depending on the specific function we want the network to approximate and how the twin branches interact i.e., In-Network merging (Dosovitskiy et al., 2015), Binary classifier (Huang and Chen, 2021), Cost function (Chopra et al., 2005), etc.

Using the taxonomy presented in the work by Leal-Taixé et al. (2016), the architecture we have developed can be described as an In-network merging CNN. The two twin branches converge in a hidden linear layer that processes simultaneously the result of the two branches. Regarding the type of output, the binary classifier approach (Koch et al., 2015) was discarded in favor of an output layer directly producing an embedding vector (Simo-Serra et al., 2015). This enables the possibility of ranking images based on their similarity with respect to an input patch, instead of having a binary matched or not-matched output.

Inspired by the conclusion of Bui et al. (2018), Bui et al. (2017), the developed siamese neural network uses twin branches with shared weights, since all the inputs of the model belong to the same domain. We prefer this configuration over other architectures considering independent branches. The CNN structure is inspired by the work by Campbell et al. (2017), using 3 convolutional layers followed by 3 fully connected layers. The input convolutional layer has 3 input channels (RBG channels), 8 output channels, and a kernel size of 10×10. The second and third convolutional layers have respectively 16 and 32 output filters with a kernel size of 5×5 pixels. Each of the convolutional layers has a stride of 1 pixel, 0 padding, and it is followed by a Rectified Linear Unit activation function and a max-pooling layer.

The output of the convolutional layers is fed into 3 fully connected layers. The two hidden layers have respectively 4096 and 1024 output channels, while the output layer of the network is of size 256 and it represents the size of the embedding.

During the training, a dropout layer with a rate of 0.25 is used, for its regularization effect, and the Adam optimizer is used, with a learning rate $\lambda = 10^-5$ that is kept constant. In this work, we have studied two approaches to train the proposed siamese network. The two share the same architecture, but use a different loss during training, namely a *contrastive* or *triplet loss* (Deepak and Ameer, 2020; Bui et al., 2017), and their margins are set respectively to $m_c = 5$ and $m_t = 10$.

Contrastive loss function. Let use the name *anchor sample* to identify a reference image a_i in the set of sub-images S of all the possible locations in a map M. Given an anchor image a_i, the set of *positive samples* is the set of all the images $p_i \in S$ that frame the same area of a_i and that differ from it only for illumination conditions, zoom level, and small rotations. The set of *negative samples* is the set of all the images $n_i \in S$ that differ from a_i also for the location that they are framing.

From a geometrical point of view, the contrastive loss function tries to push as close as possible to zero the distance between the embeddings of the anchor and the positive samples, while maximizing the anchor-to-negative distances (Deepak and Ameer, 2020). Given the embedding vectors $\vec{v_1} = CNN(a_i), \vec{v_2} = CNN(p_i)$ computed with the siamese CNN, the contrastive loss function is defined as follows:

$$L(\vec{v_1}, \vec{v_2}, \vec{y}) = \frac{1}{2}(1-y)d^2 + \frac{1}{2}(y)max(0, m - d^2) \quad (2)$$

where $\vec{y}$, is an N-dimensional vector of labels, while m and d are respectively the margin and the distance metric. Each label is an integer value in $\{0, 1\}$. For each pair of patches, the label is 1 if the two patches are different, 0 otherwise.

Triplet loss function. As for the contrastive loss, the triplet loss ensures that an anchor image is closer to the positive samples than it is to the negative samples. However, the contrastive loss pushes to zero the distance between the anchor and the positive samples while the triplet loss tries to establish a relative distance measure that is more suited in a context where a ranking of the samples is needed (Hermans et al., 2017; Dong and Shen, 2018). From a mathematical point of view, the equation used for the triplet loss function is

$$L(\vec{a}_i, \vec{p}_i^+, \vec{n}_i) = max(0, d(a_i, p_i^+) - d(a_i, n_i) + m) \quad (3)$$

where $\vec{a}_i$, $\vec{p}_i^+$ and $\vec{n}_i$ are respectively the anchor, positive and negative embeddings vector, while m and d are respectively the margin and the distance metric. From the mathematical definition of the triplet loss, it is clear that this function does not penalize the model when the difference between the feature space distance of the negative sample and the positive distance is greater than the margin. In formula $d(a_i, n_i) - d(a_i, p_i^+) > m$.

3.4. Dataset & training

In our experiments, the input of the model is a set of patches P_i of size 256×256 extracted from maps M_k, where $k \in [1, 41]$ is an index identifying the location of the map of size 1024×1024 of the asteroid surface. This set of images is produced using the NASA 75-cm resolution model obtained in March 2019 using imagery data taken at 1.5-km by OSIRIS-REx's PolyCam camera through stereophotoclinometry (Mazarico et al., 2017). The 3D model is enriched with textures and random natural features, to reproduce realistic lighting and shadows. We used Blender as 3D software and Cycles as render engine. The set of rendered maps used for training is made of 41 renders of different asteroid locations M_k, rendered in 10 different illumination conditions and at 3 different zoom levels, for a total of 1230 maps. For the test set, 5 new illumination conditions are used for a total of 615 maps. An example of dataset samples can is shown in Fig. 7.

In each map, there are 589’824 possible valid positions out of which extracting a patch of size 256×256. Therefore, the training set is made of 725 million patches. Due to the huge amount of patches in the dataset, in every epoch only a subset of 800 patches is selected randomly for the training. Image noise is also added introducing, with a probability of 50%, small random rotations up to 10 degrees and a random scaling factor $f_s \in [0.25, 2]$. Also brightness, contrast, and saturation noise is added, respectively with a factor of 0.6, 0.6, 0.2, and a probability of 40%.

The dataset used for training is the same for both losses. However, as the two need different supervision, we change the labelling scheme accordingly as it follows, in the Contrastive loss approach, the patches are coupled in order to provide the network with pairs of similar images and pairs of dissimilar ones. The training dataset is divided exactly in half, with 50% of the pair providing an example of similar patches, and the remaining 50% providing examples of dissimilar pairs of patches. Two images are considered to be the same if they render the same coordinates on the same map, meaning that they are framing the same location, regardless of any other differences in the image. Indeed, it is considered as noise any other parameters, such as different illumination conditions, shadows, scaling, and small rotations.

The dataset used with the Triplet loss is also balanced in order to guarantee a meaningful and effective distribution of easily satisfied triplets and tricky ones. The criterion used for this purpose is the following:

- Given an anchor sample, a *positive sample* is a patch taken in the same map and in the proximity of the anchor sample. Two patches are said to be in proximity if their centers are within a range of 25 pixels from each other. A *trivial positive* is defined as a positive patch with the same scaling level of the anchor. A *tricky positive* is defined as a positive patch with different illumination conditions and a different scaling level.

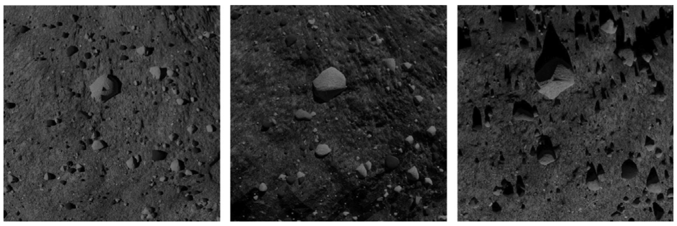
Fig. 7. An example of the map used in the dataset. From left to right the same map is shown in different illumination conditions and increasing zoom level, which should not be confused with the digital scale of the map.

- Given an anchor sample, a *negative sample* is a patch taken in a different area from the anchor or in the same area but far from the anchor’s coordinates. Two patches are defined far away if their centers are more than 50 pixels apart. A *trivial negative* is defined as a negative patch in different illumination conditions with respect to the anchor and/or rendered in a different scaling level. A *tricky negative* is defined as a negative patch in the same illumination condition of the anchor and/or a patch sampled in a range from 50 to 150 pixels away from the anchor’s coordinates.

The balanced dataset is then built as follow: for the positive samples, the center is initialized to be the same as the anchor with a 60% probability, while with a probability of 40% the center is shifted away of a random distance within the 25 pixels. For the negative sample, there is a 50% probability to sample it in a random area and in a random position, different from the anchor sample. The other 50% of the samples are positioned in the range from 50 to 150 pixels from the anchor coordinates. It must be underlined that the 50% negative samples, sampled in a random position, must not be considered all as trivial triplets. Indeed, a consistent amount of patches are similar even being far away from each other, leading to hardly distinguishable samples. In all the mentioned cases, the images are sampled in different illumination conditions and zoom levels with a uniform probability. The same image noise used in the Contrastive loss approach is added also in this case (i.e., random scaling factor, small rotations, brightness, contrast and saturation).

4. Experimental evaluation

We tested the system on three different tasks: patch matching, patch searching, as described in Section 3.2, and the actual navigation task. Each of these task builds upon the previous ones. Thus, we decide to evaluate the first two separately and then combined into the third task to assess the network performance in each of them.

4.1. Accuracy metrics

We measure the system performance using a modified accuracy score that converts similarity scores into location predictions. The accuracy metric is defined introducing the concept of *acceptable error threshold* t_a. The coordinates of the predicted position c_p of the patch seen by the spacecraft with respect to the map are compared with its true position C_{true}. If the distance between the predicted position and the true one is under the acceptable error threshold, $d(c_p, C_{true}) < t_a$, the prediction is considered as correct, $N_{correct}$. The accuracy a is defined as the ratio between the number of correct evaluation over the number of tests performed, $a = \frac{N_{correct}}{N_{tot}}$.

The pixel density function presented in the work Lorenz et al. (2017) was used to choose a pixel error threshold, in agreement with the NASA's landing accuracy requirement for the OSIRIS-REx mission. The error threshold is thus fixed to $t_a = 50$ pixels. Maintaining a navigation pixel error under t_a during the descent implies having a navigation error under 25 meters for the part of trajectory under 1.5 km from the asteroid surface. It is worth mentioning that when the probe is under 500 meters, an error of 50 pixels corresponds to a navigation error of 7 meters.

The performance of the system is measured in terms of average prediction error E_{avg} and prediction accuracy. The average prediction error is computed by averaging the sum of the prediction error E on each prediction. The prediction error is defined as the Euclidean distance, measured in pixels, between the true coordinates (C_{true}) of the searched patch P_c and the system's predicted coordinates (c_p); therefore $E = d(C_{true}, c_p)$. Notice that, the value of the Euclidean distance between the coordinates of the centers of the two patches, does not have a direct correlation with the Euclidean distance between their respective embedding vectors.

A small distance in the map coordinate system means that the two patches are close in the map, whereas a small distance between the embeddings means that the two patches are close in the features space. Having similar features is not always correlated with being close inside the map. Indeed, two generics patches p_1 and p_2 may be far from each other in the 2D map space coordinate system but they might be very similar with each other and therefore they could have a small embedding distance in the features space. Vice versa, being close inside the map does not always imply to have similar features.

4.2. Patch matching task

Our siamese neural network is trained to solve a patch matching problem, that is, given two input patches p_1, and p_2, and their embedding $e_1 = cnn(p_1), e_2 = cnn(p_2)$, to evaluate their similarity by computing their embedding distance, and produce small distances if the two patches are similar, and larger ones if they are not.

For the model trained with contrastive loss, p_1 and p_2 are evaluated as matching patches if the Euclidean distance between their embeddings is under a given threshold, $d(e_1, e_2) < t_v$. The model is trained to push toward zero the distance $d(e_1, e_2)$ if p_1 and p_2 match, while pushing the distance above the threshold t_v if the two patches do not match. In our setting, experimental results show that the threshold that maximizes the accuracy is $t_v = 1.5$. With this setup, on the test set, the model reached an accuracy of 94%, as shown in Fig. 8.

The model trained with triplet loss does not necessarily push toward zero the embedding distance of two matching patches, therefore in this scenario a prediction is considered as correct if the anchor-positive distance is smaller than the anchor-negative distance inside the triplet, $d(a_i, p_i) < d(a_i, n_i)$. With this setup, on the test set, the model reaches an accuracy of 90%. The test set, for both the models, is built with the same strategy of the training set described in Section 3.4, but on different maps, never used for the training.

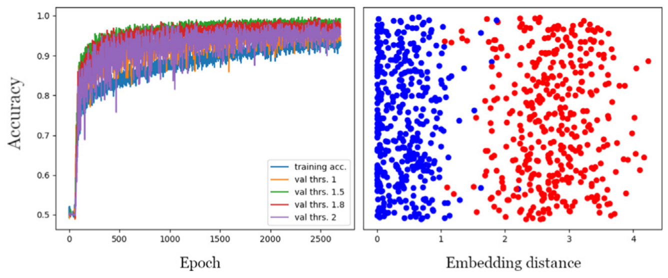

Fig. 8. On the left the values of both training and testing accuracy of the model trained with contrastive loss are shown. The training accuracy is measured with a threshold of 1, while for the test accuracy multiple values are shown. On the right image, the red dots represent pairs of patches that are labeled as dissimilar, while the blue ones are labeled as matching patches (patches that frame the same area but in different noise condition). The dots are displaced by a random offset along the y axis for visualization purposes, while on the x axis the dots are plotted according to the embedding distance between the elements of the pair. As can be seen, the red dots are clustered in an area of a higher embedding distance value, while the matching patches are closer to the area of zero distance. In both the chart, can be seen that the model learned a threshold value $t_v = 1.5$ as to discriminate matching patches from the other ones.

4.3. Patch searching task

On top of the patch matching task, a patch searching algorithm is built using the approach described in Section 3.2. In each test, a random map M_t is selected from the maps test set and a random patch p_t is extracted from a map M_p that frames the same area of M_t, but has different illumination condition and zoom levels. The system receive as input M_t and p_t and outputs the predicted location c_t for p_t. Scientific literature already studied the advantages and disadvantages of machine learning compared with traditional matching algorithm (O'Mahony et al., 2019; Melekhov et al., 2016). However, during this work, the use of two traditional matching algorithm is also tested in order to analyze their performance in this specific domain and obtain fair and comparable results. In particular, we use *template matching* (Briechle and Hanebeck, 2001) as well as *ORB* (Luo et al., 2019) as baselines in this test and use the OpenCv2 implementation for both.

The template matching algorithm returns the coordinate of the point in the map with the maximum matching score with the searched patch. We used as criterion the OpenCv2 template matching normalized sum of squared differences. On the other hand, ORB returns a list of the most relevant key point identified by the algorithm itself. Since the list of key points is not always contained in a specific perimeter, the key point with the maximum matching score is taken as the predicted position of the searched patch, as shown in Fig. 9. These algorithms are used to solve a patch searching task and their performance are reported, for comparison, in Table 1.

These algorithms have been tested in different illumination, rotation and scaling scenarios. As can be seen in Table 1, the traditional matching algorithms perform well when the searched patch is acquired in the same condition of the map, therefore when there is no noise. This result is expected, since if a perfect match exists, a traditional algorithm is likely to find it. However, when the noise is present the performance drops significantly for both traditional matching algorithms. The key point detection algorithm is particularly prone to illumination variations, while the template matching is prone to a mismatch in the scaling of the searched patch.

Conversely, the developed neural networks have better preserved the performance in noisy scenarios. The performance achieved by the contrastive loss and the triplet loss models are respectively of 95% and 96% in the scenario of sun conditions mismatch between the searched patch p_t and the map M_t, while they score 83% and 79%, respectively, when introducing also small rotations and scaling mismatch. We remark that this performance is achieved just from the image analysis process without relying on the current spacecraft location, attitude, or dynamics described in Section 3.2.

4.4. Navigation simulation

The patch searching algorithm was used also in different realistic landing simulation scenarios to test the possibility of using the proposed model as a terrain relative navigation system. The approach is the same as for the patch searching task. However, in this case, there is a connection between the previous prediction and the following, allowing the use of odometry information to refine the prediction, that is, the weighted similarity map and outliers detection algorithm described in Section 3.2.

Table 1
Patch searching accuracy in different noise scenarios.

Tested algorithm	No noise	Sun variation	Sun, scale, rot. variation
Triplet loss model (Ours)	100%	95%	83%
Contrastive loss model (Ours)	100%	96%	79%
Template matching (Briechle and Hanebeck, 2001)	100%	80%	61%
ORB (Luo et al., 2019)	100%	29%	27%

The simulated scenario involves a model of the OSIRIS-REx spacecraft, performing a descent on a parabolic trajectory, as shown in Fig. 10, and an image M_a taken as the georeferenced map of the landing area. The initial altitude of the probe, in all the simulations, is 300 meters from the asteroid's surface, while the map M_a is taken at a point l_m at an altitude of 250 meters. During the simulations, the probe performs a parabolic descent to a close-up point of 10 meters from the surface.

During the descent, images M_i are taken in each time instant of the simulation at different altitudes depending on the current position of the spacecraft. From the images M_i, the central patch p_i is extracted with the dynamic scaling algorithm mentioned in Section 3.2 and the patch searching algorithm is executed to find p_i inside the map M_a of the landing area. The details of the process have been described in Section 3.2.

The simulations are performed in different illumination conditions represented by the degree of inclination of the sunlight vector with respect to the z axis identified in Fig. 10, which is the axis perpendicular to the surface overflown by the probe during the descent. As an additional

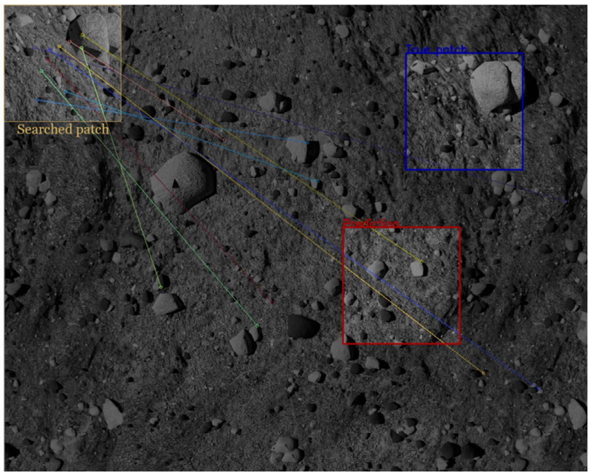

Fig. 9. Example of diffuse list of matching key points. The blue square represent the correct location of the searched patch, while the red one is the predicted value computed using the key point with the maximum matching score according to ORB algorithm.

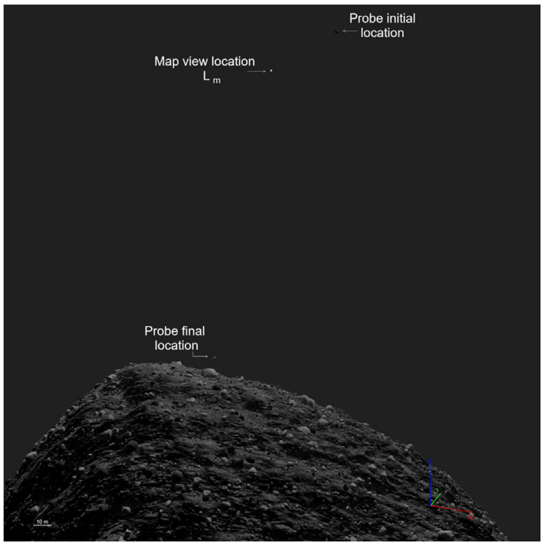

Fig. 10. Simulated descent on asteroid Bennu. The location l_m is where the map of the area is taken and remains the same during the descent. The probe performs a parabolic trajectory from the initial to the final position. The probe performs a descent of 290 meters relative to the asteroid surface that, due to the shape of the asteroid, corresponds to a descent of 250 meters relative to its initial position.

source of noise, we simulated small rotations of the probe during the landing from -5° to $+5^\circ$ degrees. The map is taken with a sun vector perpendicular to the surface (0° of tilt) and it is kept at the same altitude during the descent causing a scale mismatch between the map and the probe images. An example of noise magnitude is shown in Fig. 11.

In all the simulated scenarios the performance of the navigation system remained under the 10 meters of error during the descent till a close up point of 50 meters. Below this threshold the navigation accuracy decreases in all the simulations. We suppose that, rather than the altitude mismatch, the navigation accuracy is penalized by the lack of details and natural features in the digital model when the probe is close to the surface, but we expect it to increase if a higher quality model is available.

The results of the simulations are summarized in Table 2 and in Fig. 12. To better analyze the performance of the system we provide the average error before and after the altitude threshold of 50 meters. We recall that the error in meters is computed using a pixel density of 0.15 pixel/meter, which is the pixel density of the map taken at an altitude of 250 meters. For comparison, the requirement of NASA's OSIRIS-REx mission was a landing accuracy within 25 meters from the designed spot, which was narrowed down to 8 meters due to unexpected Bennu's surface roughness (Berry et al., 2020; Lauretta et al., 2019).

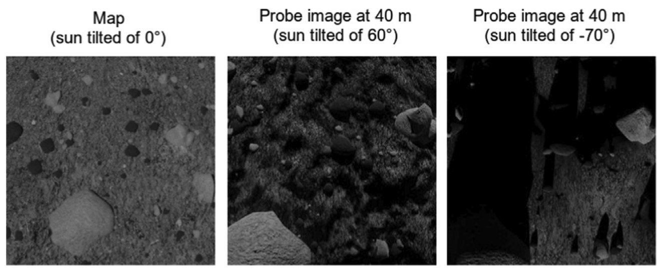

Fig. 11. Example of noise magnitude during the simulations. These three images show the same location, the leftmost image is the corresponding patch inside the map M_a at 250 meters from the surface, while the two on the right are captured by the probe at 40 meters of altitude in two different illumination condition.

Table 2
Simulations navigation error in different lighting scenarios.

Simulation	Sun vector	Prediction error (over 50 meters)		Prediction error (below 50 meters)	
		mean [m]	std [m]	mean [m]	std [m]
Sim. 1	60°	5,51	2,5	7,88	3,71
Sim. 2	45°	5,33	2,7	21,48	13,02
Sim. 3	30°	5,07	2,30	22,10	12,94
Sim. 4	15°	4,74	2,31	22,10	12,94
Sim. 5	0°	4,24	2,23	23,45	14,86
Sim. 6	-15°	4,72	2,20	21,96	12,93
Sim. 7	-30°	4,05	1,98	23,35	14,59
Sim. 8	-45°	5,04	2,53	23,65	14,22
Sim. 9	-60°	6,68	2,87	28,98	10,73
Sim. 10	-70°	6.89	3,12	24,32	6,89

4.5. Odometry error accumulation

The downside of using the current position information is that some odometry data is required by the system, and the prediction error could be accumulated leading to drifting issues. Despite this potential issue, the proposed algorithm showed good robustness and a significant gain in navigation accuracy. The performed simulations show that the prediction error is not constantly accumulated over time, as can be seen in Fig. 12. Nevertheless, to further test the robustness of the system, an additional experiment has been conducted following the setup used in the simulations. In this experiment, we perturb the previous known position of the spacecraft, simulating miss predictions, to assess the ability of the system to recover from potential errors. We repeat the simulation multiple times, each time linearly increasing the perturbation magnitude.

In all these new simulated scenarios the system was able to recover the correct position denying the concerns about the odometry error accumulation issue. More specifically, the most relevant tested scenarios involved the simulation of different magnitudes of prediction errors. Small prediction errors, under 100 pixels were corrected and absorbed immediately without affecting the overall accuracy in the long term. For errors between 100 pixels and 200 pixels, the system was able to recover the correct position after few steps. For an even bigger error, up to 300 pixels, the system recovered the correct position after 18 steps (around half of the simulation duration).

4.6. Performance on real data

In all previous tests, experiments were conducted in simulation, i.e., on rendered images. This enabled us to test the system in challenging scenarios which rarely occur in real conditions. Nonetheless, we decided to further test the system on real data taken in the OSIRIS-REx mission to verify the generalization capabilities of the proposed system. The tests are conducted in the same way as the patch searching problem described above. We select six real nadir

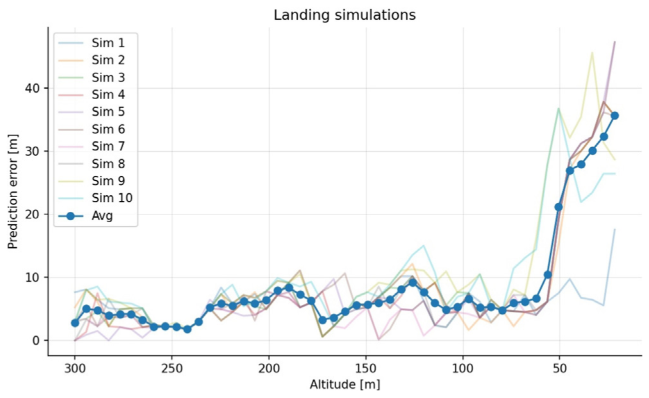

Fig. 12. Landing simulations results showing the trend of the navigation error during the descent.

images of Bennu's surface to be used as a canvas for the patch searching task. The images were taken from the NASA OSIRIS-REx repository (link in Fig. 13 caption). We select images taken at different altitudes and in different illumination conditions, to test the model in different scenarios. We used the Site Nightingale from Recon B, the Site Nightingale Crater Surrounding Region, and the Sweeping View of Bennu's Northern Hemisphere. From this last image we extracted four sub-images of 1024 × 1024 to test a scenario with a small degree of tilt with respect to the asteroid surface, see Fig. 13 for more details. From each image, 100 patches were extracted randomly for a total of 600 patch searching tests.

The Contrastive and Triplet models reached respectively 93% and 95% of accuracy in finding the correct location of the searched patch. It must be underlined that the task solved here is easier than the one solved in the previous simulations since the searched patch is in the same illumination condition of the map. Nevertheless, the fact that the model is able to match with such high accuracy patches of real images even if it is trained on rendered images, is remarkable and highlights the potentiality of the proposed technology. Moreover, the results obtained on the real images are the same obtained on the rendered images, providing an evidence regarding the generalization capability of the system from simulation to real navigation.

4.7. Computational analysis

We used the Facebook AI Research's fvcore library (facebookresearch, 2022) to estimate the floating-point operation per second (FLOPs) required by our model to compute an embedding of an input image. The size of the model is 1 Gb and it requires 330 M FLOPs for computing the embedding of an input image. This is the most expensive operation of our algorithm. We recall that the map can be stored as a set of embedding vectors before the descent, therefore only the central patch of the images coming from the probe should be processed during the landing. In

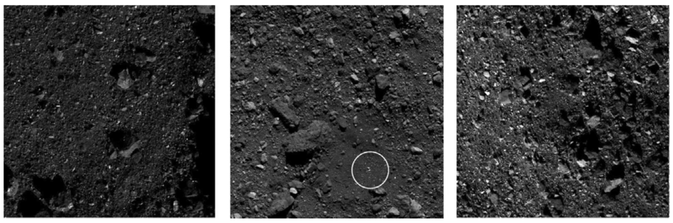
Fig. 13. Images of Bennu provided by the OSIRIS-REx mission that we used to perform the patch searching tests. From left to right, we used the Site Nightingale from Recon B image taken at 0.6 km, the Site Nightingale Crater Surrounding Region image taken at 4.8 km, and the Sweeping View of Bennu's Northern Hemisphere image taken at 115 meters. From this last image we extracted four sub-images of 1024 × 1024 to test a scenario with a small degree of tilt with respect to the asteroid surface. From these images, random patches of 256 × 256 have been extracted to perform a patch searching task. https://www.asteroidmission.org/galleries/spacecraft-imagery.

our experiments, the embedding is a vector of 256 elements, each of which can be represented with 8 bytes, for a total of 2 KB of memory for each embedding. Using the same settings of our experiments, i.e. patch size of 256 pixels and a sample rate of 16 pixels, an image of 1024 × 1024 pixels can be encoded using 2304 embeddings, for a total of 4.6 MB. We further recall that after the computation of the embedding of the image coming from the probe, the rest of the algorithm can be executed with a single scan of the 2034 map's embedding vectors.

For comparison with other navigation systems, the Hayabusa mission implements a lightweight tracking algorithm but relies on hardware components to perform the navigation. At the best of our knowledge, there are no official documents describing the computational need of the OSIRIS-REx NFT system, however, in addition to the matching algorithm, it requires a real-time rendering engine to navigate. While our system is not specifically optimized for a space-graded computing system, and we leave this further optimization to future works, recent studies explored the use of low-density FPGAs to implement efficient CNN for image processing with similar or even higher complexity to that of our system in contexts of strict energy and computational requirements similar to that of previous space missions (Lentaris et al., 2018; Véstias et al., 2020). We thus believe the proposed system could easily be optimized to met mission requirements with a proper hardware implementation.

5. Conclusions & future work

The main contribution of our research was to study a system to solve the terrain relative navigation problem by using deep learning to improve traditional navigation techniques and achieve greater level of machine autonomy in space missions. The system aims to estimate the current position of the spacecraft by computing a probability map that merges together odometry information and image matching scores produced by a neural network specifically trained to estimate the similarity between pairs of ground images.

The results described in Section 4.3 showed the advantage of using deep learning in the processing pipeline over traditional features descriptors such as ORB or the template matching algorithm, which experienced a greater performance loss due to image noise.

The proposed system has shown performance comparable to current terrain relative navigation systems, under low mismatch of illumination conditions between the map and the probe camera images. The system has the advantage of not relying on dedicated hardware components, as the artificial landmarks, nor terrain-specific hand-crafted features. The generality of the proposed solution is supported by tests conducted on simulated terrain as well as on real images without measuring any performance loss despite the domain shift. It is reasonable to conclude that data coming from different missions could be used to improve the neural network to reach even better performance.

The deep learning image matching algorithm has shown good robustness under different illumination conditions, zoom levels, and small rotations. Moreover, the results has shown good recovery capabilities even in the presence of severe, artificially added, odometry miss-predictions. We further remark that all navigation systems employed in real missions rely on previous probe positions and attitude, while the proposed system only uses these information to refine its prediction, not totally relying on them.

The system showed a decrease in prediction accuracy with the increase of the altitude mismatch between the probe and the position where the map was taken. Future work can test the possibility of using different maps taken at different altitudes to keep constant the mismatch between the camera images and the map to improve the performance in the last part of the landing. In addition, other odometry information can be used to refine the navigation prediction. Another possible research direction is that of optimizing the algorithm to meet space technology requirements, especially concerning the size of the model, which we believe could be easily met drawing on TinyML Dutta and Bharali (2021), network quantization Liang et al. (2021) and embedded deep learning Wu et al. (2021) research.

A recurrent problem in deep learning is the lack of data. Especially in a scenario like space operations, obtaining data is an expensive and resource-demanding process. Future research may explore the use of transfer learning techniques or data fusion to exploit both the information coming from the optical cameras and the map built with LIDAR technology. Besides, other information regarding the probe dynamics could be included to further refine the spacecraft position probability map.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Antreasian, P., Chesley, S., J., et al., 2001. The design and navigation of the near-shoemaker landing on eros, pp. 1–28.

Berry, K., Getzandanner, K., Moreau, M.C., et al., 2020. Revisiting osiris-rex touch-and-go (tag) performance given the realities of asteroid bennu. In: Annual AAS Guidance, Navigation and Control Conference GSFC-E-DAA-TN77488, pp. 1–10)

Briechle, K., Hanebeck, U.D., 2001. Template matching using fast normalized cross correlation. In: Optical Pattern Recognition XII. vol. 4387. International Society for Optics and Photonics, pp. 95–102.

Bui, T., Ribeiro, L., Ponti, M., et al., 2017. Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network. Comput. Vision Image Understand. 164, 27–37.

Bui, T., Ribeiro, L., Ponti, M., et al., 2018. Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression. Comput. Graphics 71, 77–87.

Campbell, T., Furfaro, R., Linares, R., et al., 2017. A deep learning approach for optical autonomous planetary relative terrain navigation. Spaceflight Mech. 160, 3293–3302.

Chopra, S., Hadsell, R., LeCun, Y., 2005. Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, IEEE, pp. 539–546.

Deepak, S., Ameer, P., 2020. Retrieval of brain mri with tumor using contrastive loss based similarity on googlenet encodings. Comput. Biol. Med. 125, 103993.

Dong, X., Shen, J., 2018. Triplet loss in siamese network for object tracking. In: Proceedings of the European conference on computer vision (ECCV), pp. 459–474.

Dosovitskiy, A., Fischer, P., Ilg, E. et al., 2015. Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2758–2766.

Dutta, L., Bharali, S., 2021. Tinyml meets iot: A comprehensive survey. Internet Things 16, 100461.

facebookresearch, 2022. fvcore. URL: https://github.com/facebookresearch/fvcore/blob/main/docs/flop_count.md [Online; accessed 28. Feb. 2022].

Fujiyoshi, H., Hirakawa, T., Yamashita, T., 2019. Deep learning-based image recognition for autonomous driving. IATSS Res. 43 (4), 244–252.

Hashimoto, T., Kubota, T., Kawaguchi, J., et al., 2010. Vision-based guidance, navigation, and control of hayabusa spacecraft-lessons learned from real operation. IFAC Proc. Vol. 43 (15), 259–264.

Hermans, A., Beyer, L., Leibe, B., 2017. In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737, pp. 1–17.

Huang, L., Chen, Y., 2021. Dual-path siamese cnn for hyperspectral image classification with limited training samples. IEEE Geosci. Remote Sens. Lett. 18 (3), 518–522. https://doi.org/10.1109/LGRS.2020.2979604.

Johnson, A.E., Montgomery, J.F., 2008. Overview of terrain relative navigation approaches for precise lunar landing. In: 2008 IEEE Aerospace Conference, IEEE, pp. 1–10.

Kawaguchi, J., Fujiwara, A., Uesugi, T., 2008. Hayabusa–its technology and science accomplishment summary and hayabusa-2. Acta Astronaut. 62 (10–11), 639–647.

Koch, G., Zemel, R., Salakhutdinov, R. et al., 2015. Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol. 2, Lille, p. 0.

Lauretta, D., Balram-Knutson, S., Beshore, E., et al., 2017. Osiris-rex: sample return from asteroid (101955) bennu. Space Sci. Rev. 212 (1), 925–984.

Lauretta, D., DellaGiustina, D., Bennett, C., et al., 2019. The unexpected surface of asteroid (101955) bennu. Nature 568 (7750), 55–60.

Leal-Taixé, L., Canton-Ferrer, C., Schindler, K., 2016. Learning by tracking: Siamese cnn for robust target association. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 33–40.

Lentaris, G., Maragos, K., Stratakos, I., et al., 2018. High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. J. Aerospace Informat. Syst. 15 (4), 178–192.

Liang, T., Glossner, J., Wang, L., et al., 2021. Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing 461, 370–403.

Litjens, G., Kooi, T., Bejnordi, B.E., et al., 2017. A survey on deep learning in medical image analysis. Medical Image Anal. 42, 60–88.

Lorenz, D.A., Olds, R., May, A. et al., 2017. Lessons learned from osiris-rex autonomous navigation using natural feature tracking. In: 2017 IEEE Aerospace Conference. IEEE, pp. 1–12.

Luo, C., Yang, W., Huang, P. et al., 2019. Overview of image matching based on orb algorithm. In: Journal of Physics: Conference Series. vol. 1237, IOP Publishing, pp. 1–12.

Mazarico, E., Rowlands, D.D., Sabaka, T.J., et al., 2017. Recovery of bennu's orientation for the osiris-rex mission: implications for the spin state accuracy and geolocation errors. J. Geodesy 91 (10), 1141–1161.

Melekhov, I., Kannala, J., Rahtu, E., 2016. Image patch matching using convolutional descriptors with euclidean distance. In: Asian Conference on Computer Vision. Springer, pp. 638–653.

Nandy, A., Haldar, S., Banerjee, S., et al., 2020. A survey on applications of siamese neural networks in computer vision. In: 2020 International Conference for Emerging Technology (INCET). IEEE, pp. 1–5.

Ogawa, N., Terui, F., Mimasu, Y., et al., 2020. Image-based autonomous navigation of hayabusa2 using artificial landmarks: The design and brief in-flight results of the first landing on asteroid ryugu. Astrodynamics 4 (2), 89–103.

Opromolla, R., Fasano, G., Rufino, G., et al., 2017. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. Progress Aerospace Sci. 93, 53–72.

O'Mahony, N., Campbell, S., Carvalho, A., et al., 2019. Deep learning vs. traditional computer vision. In: Science and Information Conference. Springer, pp. 128–144.

Simo-Serra, E., Trulls, E., Ferraz, L. et al., 2015. Discriminative learning of deep convolutional feature point descriptors. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 118–126.

Ulamec, S., Biele, J., Blazquez, A., et al., 2015. Rosetta lander–philae: landing preparations. Acta Astronautica 107, 79–86.

Véstias, M.P., Duarte, R.P., De Sousa, J.T., et al., 2020. A configurable architecture for running hybrid convolutional neural networks in low-density fpgas. IEEE Access 8, 107229–107243.

Watanabe, S.-I., Tsuda, Y., Yoshikawa, M., et al., 2017. Hayabusa2 mission overview. Space Sci. Rev. 208 (1), 3–16.

Wu, R., Guo, X., Du, J., et al., 2021. Accelerating neural network inference on fpga-based platforms–a survey. Electronics 10 (9), 1025.

Zagoruyko, S., Komodakis, N., 2015. Learning to compare image patches via convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4353–4361.