Analysis of Planetary Surfaces: Crater Detection

Luís José Martinho Sequeira Barata Marques
luisbmarques@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

July 2019

Abstract

Impact craters can be found on all telluric bodies of the Solar System being the most studied geomorphic planetary pattern because they provide a cue to understand many geological processes like the age of a planet surface. This work is focused on impact craters from Mars, through images obtained from the HiRISE camera aboard the Mars Reconnaissance Orbiter mission from NASA. Since 1988 that attempts are being made to create a crater catalog from Mars as complete as possible. The first attempts, done by manual detection, revealed too laborious and could only detect craters down to 5km diameter [2]. In the following years, several approaches have been proposed to automate the task and there are studies that achieved results down to 1km diameter [19]. However, there are no catalog for craters with a diameter less than 1km. As craters of this size are in a much greater amount, its manual detection is impractical, requiring an automatic detection approach. The purpose of this work is to demonstrate that an automatic detection algorithm based on a convolutional neural network is a viable solution for this problem.

In this work, an automatic crater recognition method is proposed based on the creation and training of a deep learning architecture, a convolutional neural network. A consistent dataset is needed to properly train the network. From creating the dataset, definition of the network’s architecture to training the network, the challenges are appealing.

Moreover, the trained network is then used to detect craters at multiple scales, with pre-defined scales that range from 200m to 800m diameter, as these craters are not yet registered in the published catalogs. A sliding window technique will be used to scan an image at the different scales defined and for each one, the output images are sent to the CNN created and classified as crater, or not. After processing all scales, it was used non maximum suppression followed by a single link clustering to cluster the answer from the network for each scale, individually. In sequence, as there are several craters detected in more than one scale, another clustering method was used, k-means, to identify the size and position of each crater detected, and achieve a final response for the system.

Keywords: Impact Craters, Convolutional Neural Network, Automatic Crater Recognition, Sliding Window, Clustering

1. Introduction

Impact craters are natural excavations on a planet’s surface made by the instantaneous collision of meteoroids. They can be found on all telluric bodies in the solar system, but are more common on those that have slow surface erosion rates like the Moon, Mercury and Mars. They are the most studied geological features because they yield information about past geological processes and are a powerful tool to estimate the ages of planetary surfaces [3].

1.1. Motivation

The first crater catalogs were manually constructed, via visual inspection of images. The first global crater database of Mars was created in 1988 [2], comprising 42,284 craters with diameter greater or equal to 5km. Since this catalog was published, the technological advances in space probes have originated better sensors that capture higher resolution images from planetary surfaces that can perceive much smaller craters. Due to the much larger number of craters captured with these high resolution images (up to 25cm/pixel), it is fundamental to develop better automatic crater detection algorithms (CDA) that allow to catalog craters in a more efficient way than manual crater detection.

However, there are already some catalogs based on CDA, completed until 2km crater diameter [22]. The main motivation for this work is to develop a new method that can be able to process high volume of data, as the actual images are of much better resolution, allowing to perceive more details and thus, smaller craters. It is important to notice that as the smaller the craters are, the higher the number
of craters present in the planet surface is. There is also another catalog [19] that is complete until 1km diameter, but manually built during several years, so this work will focus on craters smaller than 1km.

The study of craters allows to obtain information about the age of terrain units, near surface properties and the planet’s degradation history. Craters are also very useful as landmarks for autonomous spacecraft navigation and control, high precision spacecraft landing missions.

Having Mars a large number of craters with a wide variety of morphological characteristics, makes the Martian surface a reliable foundation to apply automatic crater detection methods.

1.2. Problem Formulation

Given a collection of optical images of a planet’s surface, like Mars, it is intended to detect all visible craters in those images in a previously determined range of diameters. In figure 1 is presented a map projected image taken by HiRISE, the most powerful camera ever sent to another planet, onboard the Mars Reconnaissance Orbiter (MRO). This image is map projected with 25 cm/pixel of spatial resolution, with north up and 62 solar incidence angle.

There are some challenges that must be faced to achieve the result of identifying impact craters of multiple dimensions. The different shape and size of the craters, the level of erosion, shadows due to light incidence angle are all facts that require an effort to be suppressed.

Figure 1: Mars surface in January 2009. Image extracted from [10].

1.3. State of The Art

Since the 1960 decade there have been several attempts [25] to reach Mars, being by flyby, orbiting or landing in the surface in order to study the climate, atmosphere and geology of the planet. Follows a brief description of the main successful missions to Mars and the characteristics of the images.

1.3.1 Main Probes and Catalogs

The first successful flybys of Mars were carried out by the spacecraft Mariner 4 in 1965, collecting the first close-up photographs of another planet [15]. The pictures were played back from a small tape recorder over a long period and had a resolution of 3000m/pixel. Figure 2 shows a picture of Mars’ surface, taken from Mariner 4.

Figure 2: Photograph taken by Mariner 4. Image extracted from [13].

Another mission to return images of the surface of Mars was NASA’s Viking Project [18]. Consisting of two identical spacecrafts (Viking 1 and Viking 2), they reached Mars orbit in 1976 and ended communication in 1982. The cameras installed on the orbiter could take photographs with a resolution of 300m/pixel.

In 1988, Nadine Barlow built the first global catalog for Mars craters [2], based on the images retrieved by the Viking project. This catalog, constructed manually, includes 42,284 craters with diameter greater or equal to 5km.

A new milestone was achieved with the program Mars Global Surveyor. This mission started in 1997 and ended in 2006, becoming the first successful mission to Mars in two decades [16]. Its sensor known as Mars Orbital Camera (wide angle mode), collected images enough to build up a global map of the planet with images at a resolution of 200m/pixel. The same camera in its narrow angle mode could capture smaller regions at a much higher resolution (up to 1.4m/pixel).

In 2011, Salamuniccar[22] made a catalog based on the information provided by Mars Global Surveyor, containing 85,783 with diameter greater or equal to 2km, and used a cratered detection algorithm (CDA) consisting of a sequence of image processing procedures like digital elevation-map (DEM) and optical-based CDA, re-projection of coordinates, correction
of brightness and contrast of optical images and fin-
ally an integration of results.

NASA initiated another mission, called Mars Odyssey, in 2001 and still ongoing being the NASA’s longest-lasting spacecraft at Mars [17]. For the first time, a mission globally mapped the amount and distribution of many chemical elements and minerals that make up the martian surface. Its images reach a resolution of 18m/pixel.

In 2012, Robbins published a complete catalog of Mars craters based on the data retrieved from the program Mars Odyssey [19]. It contains 384,343 craters with diameter greater or equal to 1km. Several geologic and morphometric descriptors for each crater are also included.

In 2005, NASA’s Mars Reconnaissance Orbiter (MRO) was launched from Cape Canaveral, arriving at the red planet fourteen months later, in 2006 with the main mission of searching for clues about where watery martian habitats may lie [14]. It is equipped with HiRISE (High Resolution Imaging Science Experiment), a high resolution camera that allows to reveal small-scale objects in the surface. HiRISE has photographed hundreds of targeted swaths in Mars’ surface. The camera operates in visible wavelengths but with a telescopic lens that produces images with unprecedented detail in planetary exploration missions, and has 5 times better resolution than the camera equipped in the Mars Odyssey. These high-resolution images, up to 25cm/pixel like figure 3 shows, enable scientists to distinguish 1m size objects on Mars and to study the surface structure in a much more comprehensive manner than ever before [10].

Figure 3: Photograph taken by HiRISE. Image extracted from [11].

1.4. Objectives
The objective of this work is to find the best solution for two distinct problems. In the first problem, crater recognition (CR), the goal is to identify if a given image has a crater or not. For that, there is a previously provided dataset of images from which some have a centered crater that covers a large part of the image and some have not a crater at all or does not have a full crater. In this problem, the position and scale of the image is known. Figure 4 shows some examples used for the CR problem.

Figure 4: Samples used to the CR problem.

The second problem, multiscale crater detection (MSCD), is a different challenge. Given an image with a large planetary surface, it is intended to detect all visible craters available being its position and scale unknown. The performance of the solution for this problem depends in large extent on the results achieved in the first problem. In figure 5 is presented a small region of Mars surface with 1km², where several craters of different sizes can be seen.

Figure 5: Surface with multiple craters of different sizes.

2. Background
In very complex recognition problems like object recognition in images, there are no pre-defined rules that lead to a solution. In the search for a solution, some methods have been developed that are able to learn from a large number of examples and their respective classification. The construction of algorithms that can learn from and make predictions on data falls into an area called machine learning. Deep learning is a branch of machine learning that uses algorithms to simulate the thinking process, using artificial neural networks (NN). Many of these methods learn statistical properties from data as-
associated to each class and use that information to predict the class of new examples.

The way the human brain works and executes highly complex operations like speech recognition and object recognition has been an inspiration for the development of machine learning algorithms.

2.1. Neural Model

The basic computational unit of an artificial neural network (ANN) is also called neuron. Just like a biological neuron has dendrites to receive signals, a cell body to process them, and an axon to send signals out to other neurons, the artificial neuron has a number of input vectors, a processing stage, and one output that can connect to other neurons.

The first mathematical model of a neural network (see figure 6), a simple neuron, was proposed in 1943 [12] by mathematician Walter Pitts and neurophysiologist Warren McCulloch with the aim of describing the human brain and thus, the thought process.

Figure 6: McCulloch and Pitts model

Where \( x_1, \ldots, x_p \) are \( p \) inputs, \( w_1, \ldots, w_p \) are a set of weights and \( y \) is a binary output.

The output is computed in two steps, \( s = \sum_{i=1}^{p} w_i x_i \) and \( y = f(s) \), where \( f(.) \) is the unit step function (Heaviside function). This function is known as the activation function.

2.2. Multilayer Perceptron

However, the model created by McCulloch and Pitts composed by one unit only was a very restrictive logic model as it could recognize two different categories of inputs by testing whether \( f(.) \) is positive or negative. For the model to correspond to the desired categories, the weights had to be manually set. In 1957, Frank Rosenblatt developed the first perceptron[20], that computes a single output from multiple real value inputs, instead of binary inputs. For the first time, a model could learn the weights for the categories, given examples of inputs from each category.

2.3. Activation Functions

Each unit performs a linear combination of its inputs followed by a non-linear transformation \( f(.) \), denoted as activation function. Several activation functions can be found in practice like logistic function, hyperbolic tangent and rectified linear unit, being the last the most commonly used.

2.3.1 Rectified Linear Unit

The rectified linear unit (ReLU) computes the function:

\[
 f(x) = \max\{0, x\}. \tag{1}
\]

It simply eliminates the negative values, thresholding them to zero. Since it is linear, for positive values it does not saturate and it was found to accelerate up to a factor of 6 the convergence of stochastic gradient descent method [1]. However, during training phase of a network, if weights updates to zero or are initialized with zero values, the local gradient on the unit will forever be zero and the respective neuron is said to be dead.

Figure 7: Rectified Linear Unit

The ReLU, represented in figure 7, has simple, inexpensive operations to implement, in comparison to the previously presented activation functions and has become very popular being one of the most widely used activation function in CNN.

2.3.2 Training of the multilayer perceptron

The multilayer perceptron (MLP) is formed by multiple neurons, organized in layers and interconnected in a feed-forward manner (interconnections that do not form any loops or cycles). These networks usually have a supervised training, so they need a training set which contains input patterns and the corresponding desired output patterns. The training is based on a minimization of an error measure between the output obtained and the desired output. The training also involves a backward propagation of the error in a process called backpropagation.

The multilayer perceptron learning rule was originally developed by Frank Rosenblatt [20]. Training patterns are presented to the network’s inputs and the output is computed. Then, the weights are updated according to equation 2:
by the non linear activation function, is a 3 dimensional input. The output of this operation, transformed with the input, computing the dot product between the weights of the filter and the input, at any position. Each kernel slides across the width and height of the input pattern. The learning rule follows 3 simple steps:

Step 1: Initialize the weights with small random numbers;
Step 2: Give an input $x$ to the network and calculate the output;
Step 3: update the weights according to equation 2.

Then steps 2 and 3 are repeated until the error is less than a specified value or a pre-defined number of iterations is reached. After one of these conditions are met, the multilayer perceptron is said to be trained.

### 2.4. Convolutional Neural Networks

In the last 25 years, artificial neural networks had a substantial improvement and the first convolutional neural networks appeared: LENET5 [9] for handwritten recognition was the first CNN and even with slow CPUs, image features are distributed across the entire image and convolutions with learnable parameters allow to extract similar features at multiple location with few parameters; ALEXNET [1] being a much deeper and wider version of LENET, won the IMAGENET challenge by a large difference and started a revolution. Convolutional neural networks were now the underpinning of deep learning; VGG [23] was the first to use smaller 3x3 filters in each convolutional layers and combine them as a sequence of convolutions.

CNNs are constructed by layers, each one containing one or more operations, producing an output also called feature map. These features are the input of the next layer, where another set of operations is executed, and so on until the output layer where a classification is made.

The main operations used in any CNN are convolution, non-linearities (activation functions and classification) and sub-sampling. Follows the detailed explanation of those operations.

#### 2.4.1 Convolutional Layers

The convolutional layer consists of a set of learnable filters (also called kernels) that are convolved with the input transformed by activation functions. Each kernel slides across the width and height of the input, computing the dot product between the weights of the filter and the input, at any position. It is also said that the kernel convolves with the input. The output of this operation, transformed by the non linear activation function, is a 3 dimensional activation map (or feature map) and can also be interpreted as the output of a neuron that looks only at a region of the input.

Formally, for two dimensional functions $I[x, y]$ and $\phi[x, y]$ of discrete variables $x$ and $y$, like images, convolution is defined as:

$$I[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} I[n_1, n_2] \cdot \phi[x-n_1, y-n_2]$$

#### 2.4.2 Pooling Layer

The pooling layer is applied after an activation function. This operation reduces the spatial dimensions (width and height) but not depth of the input. The amount of parameters (input values) is reduced by grouping several elements and replacing them by one value only, improving the computational costs. It also helps to control overfitting.

Intuitively, this unit is used when a high activation value is present (a specific feature is in the input) so its exact location is not as important as its location relative to other features. When processing an image, it is possible to detect meaningful features. The pooling operation allows features detection being independent to small changes of its spatial position. Pooling can be performed by using several functions such as average, L2-norm or max pooling[6]. The other functions have been gradually fallen into disuse due to the best performance obtained by max pooling in practice.

#### 2.4.3 Fully Connected Layers

The last layers of the network are fully connected, allowing to perform the classification on the features extracted on the previous convolutional layers and downsampled by the pooling layers. In this layer, every node is connected to every node in the preceding layer. It is followed by a softmax activation function to classify the input image into on of the classes based on the training dataset. Although this is an activation function, it deserves a special attention as its output is the final classification result of the network. The softmax activation is usually used in multiclass problems. It takes a vector of real value scores as input and convert it to a vector of probability values between 0 and 1, assuring the sum of all output probabilities is 1. Formally, the softmax function is given by

$$softmax(z) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

A model that converts the unnormalized values to normalized probabilities for classification is called a softmax classifier.
2.5. Training

The training of a CNN requires the creation of a training set before the network can be used. As it is trained in a supervised manner, the training set consists of pairs input-output \((x', y')\) and each pattern must have a label associated that will allow the network to learn. The labels correspond to the classes in which the CNN will classify data.

Before a CNN can be used, there must be a learning phase for the network to learn what is intended to. The estimation of the network weights is called training of the network. The training phase requires a training set i.e., a collection of pictures and their respective labels to be given to the network that will learn through a process of minimization of a predefined cost criteria.

A very important step of the training is the evaluation of the output. A measure of the error between the output given by the network and the desired output for each particular pattern must be calculated and minimized. There are several optimization algorithms used, being gradient descent the most common. The learning phase is also a very important step since the weights (kernel values and weights of fully connected layers) have random values when the network is started. This phase can be separated into three main steps: the forward pass, the loss function and back-propagation.

2.5.1 Back Propagation

The training of the network is based on the minimization of an error measure between the network’s output and the desired output. It involves a back propagation through the network and for this reason training is also called back-propagation\[21\]. Back propagation refers to the method for computing the gradient of the cost function with respect to the parameters, with the aim of minimizing the cost function. To achieve this minimum, the gradient is calculated based on the back propagation of the error on the networks output through a backward network. This backward network is obtained by replacing all non linear elements of the original network by linear branches with a gain. After this, the network is transposed: the flow direction of all branches is reversed, summing nodes are replaced by divergence nodes (and vice-versa), the outputs are converted into inputs and inputs converted in outputs. This new network is called the back-propagation network.

2.5.2 Loss Function

A loss function is used to compare the network output with the ideal output. It defines how training penalizes the deviation between the prediction and the desired value. Usually what is found is always a local minimum for the loss function. Let \(c(x)\) be the cost function. The optimization problem is:

\[
\hat{c}(x) = \arg \min_w c(x).
\]

There are several functions used for minimization, in this work the one used is cross-entropy.

Formally, given two distributions over a discrete variable \(x\), where \(q(x)\) is the estimate for the true distribution \(p(x)\), the cross entropy \(H(p, q)\) is given by:

\[
H(p, q) = - \sum_x p(x) \cdot \log(q(x)).
\]

The minimization of the cross-entropy is equivalent to maximum likelihood estimation of the weights. A key feature of the multic和平 loss is that it rewards/penalizes probabilities of correct classes only, being the value independent of how the remaining probability is split between incorrect classes. Joining this cost function with a one-hot encoded label (a single true class per label), will allow extra optimization.

There are several different methods used to look for the minimum of the error function in weight space, being one of the most simple the gradient descent method (GD), presented in (7). This method consist of iteratively updating the weights, proportionally to the negative gradient of the function to be minimized. Let \(w\) be the weight to be updated, \(C\) the error function and \(t\) the iteration.

\[
w^{t+1} = w^t + \Delta w,
\]

where

\[
\Delta w = -\eta \frac{\partial C}{\partial w}.
\]

The parameter \(\eta\) is designated step size (or learning rate) and with a correct value choice, after some iterations will lead to a local minimum of \(C\).

2.5.3 Optimization Algorithm

Back propagation refers only to the method of computing the gradient, while optimization algorithms are used to perform learning using this gradient. Follow a description of the optimization algorithm used.

Adaptive Moment Estimation (ADAM) \[7\] is an algorithm for first-order gradient based optimization of stochastic cost functions, based on adaptive estimates of low order moments. It has the advantages of being computationally efficient and adequate to problems with large datasets and parameters. This method computes adaptive learning rates for each parameter from estimates of first moment (the mean, see (9)) and second moment (the uncentered variance, see (10)) of the gradients:
The network was implemented using as basis the networks from Tensorflow\textsuperscript{TM} Deep MNIST for Experts [24] and Hinton [8] and has four hidden layers and a softmax classifier. After the network is trained and tested, it is ready to be used on the following step.

3.2. Proposed Solution II: Multi-Scale Crater Detection (MSCD)

On the second problem of this work, one intends to detect all craters in an image (up to a given size) and distinguish them by their size, being possible to build a catalog based on number and diameter of existing craters in a certain image.

It is intended to scan a large image at multiple scales to allow the detection of craters of different diameters in a given range. The previously trained classifier will now be used at each scanned position to detect craters in an image of the surface of Mars.

As it is intended to obtain information about all craters, for each scale, a sliding window technique is used to scan the main image and store all the images obtained from the sliding window to use as input to the CNN in order to detect the existing craters.

Based on the classification output given by the CNN, the detected craters are catalogued by scale. However, it may arise the situation that one same crater be detected in consecutive scales. To address this problem, an algorithm based on the coordinates and diameter of the craters is implemented with the aim to merge the craters that are detected in several scales and estimate its diameter.

4. Results

The experimental results can be divided into two main parts. The first one where a CNN is created and trained to identify craters centered in an image and a second part where the same CNN is used to detect craters within multiple scales, using a clustering method to assign a diameter to each crater as accurate as possible.

4.1. Dataset

The dataset used consists of 21000 images. The division in training and testing set was made with a ratio of 73\% ÷ 27\% so that there are 15348 images in the training set and 5652 images in the test set. One can also observe the training set consists of 6174 positives and 9174 negatives and the test set consists of 2826 positives plus 2826 negatives. The training set is not balanced, consisting of a higher value (3000 more) of negative samples than positive.

The results obtained with a balanced training dataset were not satisfying enough since there were many "small" craters being detected as positives, creating a high value of false alarms. To improve this situation, 3000 negative samples were added to

\[ m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \frac{\partial C}{\partial w} \]  
\[ \nu_t = \beta_2 \cdot \nu_{t-1} + (1 - \beta_2) \cdot g^2_t, \]

where \( g^2_t \) represents the elementwise square of \( \frac{\partial C}{\partial w} \), \( \beta_1 \) and \( \beta_2 \) are the exponential decay rates for the moment estimates and \( t \) the timestep. As \( m_t \) and \( \nu_t \) are initialized as zero vectors, it was noticed\textsuperscript{[7]} that these moments are biased towards zero during the initial steps and when the decay rates are small (\( \beta_1 \) and \( \beta_2 \) values near 1). These biases are counteracted by computing a bias-corrected first and second moment estimates:

\[ \hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \text{and} \quad \hat{\nu}_t = \frac{\nu_t}{1 - \beta_2^t}. \]

After this normalization, the weight update can be done:

\[ w_{t+1} = w_t - \frac{\eta}{1 - \nu_t^t + \epsilon} \cdot \hat{m}_t. \]

The authors propose default values for \( \beta_1, \beta_2 \) and \( \epsilon \) respectively of 0.9, 0.999 and \( 10^{-8} \).

3. Implementation

This work is divided into two main problems: Crater Recognition (CR) and Multiscale Crater Detection (MSCD). The first problem is to distinguish images with a crater from images without a crater, being this a binary classification problem. For simplification it is considered that all craters are centered in the image, with the same scale and having approximately the same dimensions. The negative samples consist of images without craters, only a part of a crater or images with small craters.

To address the problems, the image in figure 1 is divided in two halves, one is used to create the dataset for the CR problem as the base to create a dataset to train the classifier and the other half of the image is used to allow the test on CR problem.

In the second problem it is considered a large image containing several craters in unknown locations and with different sizes. This is a multiscale crater detection problem.

The second half of the image is now used for detection of different diameter craters in the MSCD problem.

3.1. Proposed Solution I: Crater Recognition (CR)

In the first problem, crater recognition, there’s the need to create a classifier that can be able to correctly classify images as crater or not crater. Being a binary classification problem and considering the data available are images, a Convolutional Neural Network is proposed.
the training dataset, which did not affect the detection performance and lowered the false alarm value significantly.

To guarantee the independence between the training set and test set, the HiRISE image was divided in two parts (see Figure 8). The upper part was used for training and the lower part for testing.

4.2. Training and testing of the CNN

The network was trained with 600 epochs, using mini batches of 10 images each (corresponding to a total of \(9.2 \times 10^6\) iterations). The network update was done using Adaptive Moment Estimation (ADAM) and cross-entropy as cost function. It took about 36 hours to train the CNN in a MacBook Pro with a Intel Core i5 2.7Ghz and 8Gb of RAM.

The CNN was trained and then applied to classify the test images. Comparing the output of the network with the true class (crater vs non-crater), the statistics presented in tables 1 and 2 are obtained.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Missed Detection</th>
<th>False Alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>95.53%</td>
<td>4.18%</td>
<td>0.28%</td>
</tr>
</tbody>
</table>

Table 1: Crater recognition results in the test set.

Another interesting result to observe is some performance measurements, the confusion matrix:

\[
\text{Accuracy} = \frac{TP}{TP + FN} = 0.958.
\]

One more interesting performance measure to observe is the harmonic mean, in equation 15 that combines the two previous measures:

\[
F_{\text{measure}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 0.9767.
\]

Since the \(F_{\text{measure}}\) combines the precision and recall results, being the harmonic average of these, the result obtained demonstrates the CNN has a good performance on detecting craters.

4.3. Multiscale Detection Results

As a second part of this work, it’s intended to catalog craters with unknown location and size, using the diameter of the craters as parameter. For the aim of this work, the detection will be made for diameter in the range \([200, 800]\) meters. Using a scale duplication with 5 steps, one obtains 11 distinct scales.

When scanning each single scale, a clustering criteria was used for each scale, individually because many craters were detected more than once in the same scale and thus, having more than one center defined. To solve this situation, non maximum suppression was used followed by the average of the remaining centers (when the scores from the CNN had the same value). Thereafter, the use of a hierarchical method, a single link with a cut-off of \(1.1 \cdot r\) and Euclidean distance, led to the results presented in the following table:

<table>
<thead>
<tr>
<th>Diam. [m]</th>
<th>800</th>
<th>700</th>
<th>600</th>
<th>520</th>
<th>460</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Craters</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Diam. [m]</td>
<td>350</td>
<td>300</td>
<td>260</td>
<td>230</td>
<td>200</td>
<td>-</td>
</tr>
<tr>
<td>N. Craters</td>
<td>18</td>
<td>18</td>
<td>27</td>
<td>27</td>
<td>35</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: Number of craters detected per scale

Those results are now used to achieve a new cluster group, based on different scales.

Using the Calinski-Harabasz [4] criterion, it is possible to predict the optimal number of data clusters. With the help of MATLAB, the implementation of this algorithm is straightforward, and an optimal number of clusters is obtained for k-means.
In the end, the multiscale detection must be compared with the marked ground-truth so it is possible to measure the quality of the results. In figure 9 the ground-truth (GT) is in black and in blue are the craters from the ground-truth within the diameter range [200-800]m.

![Figure 9: Ground-truth and clustering comparison](image)

In table 4 the information from the final results is resumed.

<table>
<thead>
<tr>
<th>GT</th>
<th>Detected</th>
<th>Correct Detect.</th>
<th>M.D</th>
<th>F.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>59</td>
<td>17</td>
<td>3</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 4: Final results

The system achieved the results presented in equations 16 and 17:

\[
\text{Precision} = \frac{TP}{TP + FP} = \frac{17}{17 + 39} = 0.303. \quad (16)
\]

\[
\text{Recall} = \frac{TP}{TP + FN} = \frac{17}{17 + 3} = 0.850. \quad (17)
\]

One more interesting performance measure to observe is the \textit{Fmeasure}, in equation 18 that combines the two previous measures:

\[
\text{Fmeasure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 0.4472. \quad (18)
\]

The Precision of the system is below 50%, and this is due to the high number of false alarm (FA). However, from all detections considered as FA, if one goes further down the scales, 32 of those detections are indeed craters of smaller diameter than the considered for this work, meaning that only 7 false alarms are indeed errors, that is, detections where no crater exist. Consequently, the \textit{Fmeasure} also has a low value. Nevertheless, the Recall is consistent. This fact happens because 83% of the positive patterns used to train the CNN were created to allow data augmentation, using the method of creating 5 random centers from the 1500 manually selected craters. This makes the CNN able to detect craters the size of the window as well as smaller craters.

5. Conclusions

Automating the process of identifying and cataloging craters is a very challenging work. Crater detection algorithms are constantly evolving, and machine learning appears as one of the most eminent techniques to apply in these kind of algorithms. As presented before, the more recent works with deep neural networks show that it is one of the best methods nowadays to detect craters and may become a standard tool for crater detection [5].

5.1. Achievements

This work has two distinct parts, CR and MSCD, being the second one highly dependant on the performance of the first. CR is very challenging as it requires a consistent dataset to be carefully created and, after several attempts with different architectures for the CNN, it was possible to achieve an accuracy of 95.53% in the test of the network.

For the MSCD, the trained CNN was used in each chosen scale. Here, the performance is lower, reaching an \textit{Fmeasure} of 44.72%. However, it can be considered the fact that most of the false alarms correspond to the detection of smaller craters outside the diameter range considered in this study. From the 39 false alarm results, 32 of them are indeed smaller craters that would be classified in smaller scales if more scales would be considered. If we consider this fact, the system would achieve a precision of 0.708 and a recall value of 0.850. Consequently, the \textit{Fmeasure} would raise up to 0.7727.

5.2. Future Work

There are several ways where this work can be extended. The network can be trained with images from other planets, images with different illumination and conditions, improving the generalization capabilities of this CNN.

For the MSCD problem, there can be other methods of clustering, although the results achieved with the method used in this work lead to a consistent result. Considering the clustering method proposed, the estimated diameter of the craters have a deviation for the real value. The more scales a crater is detected in, the greater is this deviation. This fact happens because the method used considers all scales a crater is detected in to assign a diameter. Ideally, for the purpose of this work, a best
result could be achieved if it would be considered the smallest scale the crater is detected in as the diameter for that cluster.

The fact that more scales, smaller, can be tested is obvious for the performance the network show in the smallest scale used in the scope of this work. For the results obtained, it could also be used smaller scales (175m, 150m, 130m, 115m and 100m) to remove from the candidates for the 200m scale those which would be classified in the 175m scale. Using this artifact, the results would be considerably improved, as demonstrated before, however it was chosen not to do it and present the results using through all the work exclusively the scales initially proposed.

References


