ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Image Processing for Earth Observation

Remote detection of dunes on Mars

Student project

Authors:
Nemanja Stojoski
Michaël Pellet

Teacher:
Frank de Morsier

January 12, 2018

Contents

Abbreviations

EDR	Experimental Data Record
HiRISE	High Resolution Imaging Science Experiment
LBP	Local Binary Pattern
MGD^3	Mars Global Digital Dune Database
MOC	Mars Orbiter Camera
MRO	Mars Reconnaissance Orbiter
NASA	National Aeronautics and Space Administration
PA	Producer's Accuracy
PDS	Planetary Data System
RDR	Reduced Data Record
THEMIS	Thermal Emission Imaging System
UA	User's Accuracy

1 Introduction

In this report the authors will try to explain basic premises of remote detection of dunes on Mars and implement a basic algorithm on the same topic. Firstly, in this section, a short history of remote sensing of Mars and context of the topic are covered, after which some challenges specific to this field are listed. Next, a short literature review related to the area is given and lastly the details of the platform from which the data was used is given.

1.1 History

Remote sensing of Mars has had a long and rich history. The first successful mission to Mars occurred in 1965 when the NASA's *Mariner 4* mission flew by it and captured the first close-up images of Mars. The images were taken by a television camera in grayscale and they were later colored by hand. [2] As the technology progressed, Mars exploration became more advanced, with the Soviet Union's *Mars 2* being the first spacecraft that entered the orbit around Mars in 1971, therefore enabling long term observations of the surface. Numerous milestones were achieved later, with the first fully operational lander touching down on the surface of Mars in 1975, and the first rover landing in 1996. These, and dozens of other missions helped greatly increase our knowledge of Mars by utilizing different passive and active remote sensing techniques and observing energy emissions across the electromagnetic spectrum. As of 2018, there are 8 active spacecraft in an orbit around Mars or on its surface. [1]

1.2 Context

The topic of this report is the detection of dunes on Mars. Sand dunes are among the most widespread aeolian features present on its surface. On a planetary body, dunes accumulate when a supply of sand-sized grains exist or may be abraded by winds of sufficient strength. Those grains are then carried by the winds and are deposited when the winds weaken below the threshold needed to lift the sand grains. This means that the study of dune processes provides unique indicators of the interaction between the atmosphere and the surface, thereby contributing to both atmospheric and sedimentary sciences. Dunes are particularly suited for comprehensive planetary studies in part because they are abundant on the Martian surface over a wide range of elevations and terrain types, and in part because they are large enough to be studied using the wide suite of spacecraft sensors now available.

Previous aeolian studies of the Martian surface relied on *Mariner 9* and *Viking Orbiter* images to examine and map aeolian morphologies. More recent studies, using high-resolution images from the *Mars Global Surveyor* and *2001 Mars Odyssey*, have enabled scientists to re-examine surficial areas from earlier investigations and see new dune deposits undetected by previous instruments. [9]

1.3 Challenges

The challenges of the remote detection of dunes on Mars are numerous. Firstly, as there are only two active rovers (and zero humans) present on Mars, there exists no way to directly ground-truth the data obtained from the sensors. This does not present a serious challenge when the subject of the research is simply the detection of whether dunes exist on a given image of a surface. However, if for example the classification of the detected dunes or identification of

the exact chemical structure of the sand grains is required, this poses a large problem because obtaining a ground sample from a specific dune on Mars is currently not feasible. The second problem is the relative scarcity of data, compared to the abundance of remote sensing data of Earth. As there is at present no economic interest for sending imaging instruments to Mars, scientists must work with a couple of them that they have available, such as *THEMIS*, *MOC* and *HiRISE*. At present, no high-resolution atlas of Mars exists, with the most detailed one the authors of this report have been able to find being the *MOC Atlas of Mars* [7], with the resolution of approximately 230 m/pixel.

1.4 Literature review

One of the seminal papers dealing with this field of study is the *Mars Global Digital Dune Database and initial science results* paper published in 2007 in the Journal of Geophysical Research. The authors of that paper used images from the *THEMIS* instrument to provide a comprehensive view of the geographic distribution of moderate and large sized dune fields that would help researches understand global climatic and sedimentary processes that have shaped the surface of Mars. [11] In another paper, *Detection of Barchan Dunes in High Resolution Satellite Images*, the authors applied histogram equalization and noise reduction to multiple satellite images of Mars, before applying machine learning techniques to obtain candidate dunes of the specified type. [10] The last paper the authors of this report analyzed is *Edge-detection applied to moving sand dunes on Mars*, which dealt with the measurement of motion of the sand dunes by comparing images from 1999 and 2007. [12]

1.5 Platform

The main platform which was used to gather data for this report is the *HiRISE* (High Resolution Imaging Science Experiment) instrument on board the *Mars Reconnaissance Orbiter*. The orbiter was inserted into orbit around Mars in March of 2006 and is still fully operational, serving as a platform for *HiRISE* and other scientific instruments on board, as well as a relay between Earth and rovers on the surface of Mars. [5] The *MRO* is in a polar orbit, with orbital height ranging from 250 to 316 kilometers. [6] The *HiRISE* instrument itself is a camera that consists of a 0.5 m aperture reflecting telescope, which allows it to take pictures of Mars with resolutions of 0.3 m/pixel. It observes the electromagnetic radiation coming from the surface in three bands: Blue-Green (400 - 600 nm), Red (550 - 850 nm) and Near infra-red (800 - 1000 nm). It has a swath width of 6 km for the Red band and 1.2 km for Blue-Green and Near infra-red bands. [3] Further information about the technical specifications of the *HiRISE* instrument can be found on the following website: `https://www.uahirise.org/teknikos.php`.

2 Processing routine

The purpose of this section is to provide a general overview of the processing routine for the remote sensing task at hand. A flowchart of this processing routine is shown on Figure 1, and later parts of the section describe each processing step in more detail. Also, a description of used data is given in 1. More detailed description of each entry of the table is given throughout the report, where adequate. Specifics of the implementation of each of the processing steps are covered in section 3.

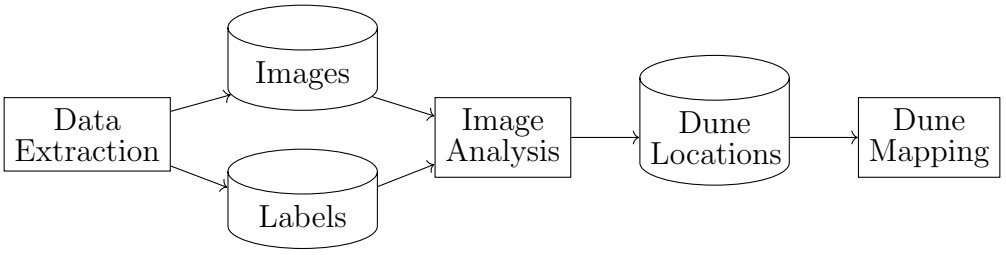

Figure 1: Dune detection processing flowchart

Platform and sensor names	*HiRISE* instrument on board the *MRO*
Source of data	`https://hirise-pds.lpl.arizona.edu/PDS/`
Acquisition date and method	The analyzed images were taken by *HiRISE* from 2006 until the present day
Spatial reference system	Planetocentric
Spatial and spectral resolution	0.3 m/pixel; Three bands: Blue-Green (400 - 600 nm), Red (550 - 850 nm) and Near infra-red (800 - 1000 nm).
Type of product	Processed (RDR)

Table 1: Description of used data

2.1 Data Extraction

As previously mentioned, the images from the *HiRISE* instrument needed to be gathered. All the images that have been taken so far by this instrument are available on: `https://hirise-pds.lpl.arizona.edu/PDS/`. This website provides two different types of products: EDR (Experimental Data Record) and RDR (Reduced Data Record). EDR products are the permanent record of the raw, unprocessed and unrectified images obtained by *HiRISE*, while the RDR products are radiometrically-corrected images resampled to a standard map projection which are formated and organized according to the standards of the PDS (Planetary Data System). All RDR images are accompanied by a detached label which provides supporting information about the observation. [8]

For the purposes of this project, only the RDR products were used. The size of each RDR image in JPEG2000 format is on the order of hundreds of megabytes. This poses a problem when the analysis of many images captured by the instrument is needed due to the constraints of time and available processing resources. Therefore, the authors of this report chose to work with the available images in JPEG format which are characterized by their greatly reduced size (typically less then 10 MB), while maintaining enough quality that is required for this analysis.

The *HiRISE* PDS offers two types of RDR products: *RED* and *COLOR*. *RED* images are a grayscale mosaic of all RED-filter channels, while the *COLOR* images are a combination of the three bands (BG, RED, NIR) previously mentioned in subsection 1.5. As the *RED* images have five times larger swath width than the *COLOR* ones [4], only they were processed, in order to perform dune detection over the maximum available surface of Mars.

2.2 Image Analysis

As the amount of data that needs to be analyzed is rather huge (several thousands of images with sizes between (2040 x 3000) pixels and (2040 x 23000) pixels), the dune detection algorithm needed to be as fast as possible. Initially, tests were made with GLCM but to analyze an image of only 256 x 256 pixels, it already took several minutes. So, other algorithms were investigated and after a few tries, it was decided to use the LBP (Local Binary Pattern) method combined with supervised classification and k-nearest neighbors algorithm.

2.3 Dune Mapping

After performing dune detection on individual images, a map of dune distributions across Mars was created. The main goal of this step was to provide a general overview of the locations of major areas on Mars where dunes are present. Another aim of this step was to compare the obtained dune distributions with the findings from the previously mentioned *Mars Global Digital Dune Database and initial science results* paper.

3 Implementation

3.1 Data Extraction

The code for the data extraction part of the project was written in Python. It consists of a single script which downloads all images from the `base_url` of `https://hirise-pds.lpl.arizona.edu/PDS/EXTRAS/RDR/ESP/`, that end with the extension: `_RED.NOMAP.browse.jpg`. All images from the `base_url` are divided into folders that represent a specific orbit of the *MRO*, with each orbit further subdivided into folders that correspond to specific images. Besides the images, relevant information from the label file for a particular image is extracted and saved in the same directory and with the same name as the image, but with a different extension (.LBL).

3.2 Image Analysis

3.2.1 Reference data

In order to perform supervised classification, 8 different classes were selected. They are: Dunes, Flood trace on tilted surfaces, Flat surface, Ridges and small mountains, High density of craters, Small irregularity on flat surfaces, Flat surface with small craters, Strange dark features. For each class, reference images were gathered by cropping small zones of several images taken by the *HiRISE* instrument. Those reference images have a size of 100 x 100 pixels and each of them show only one class. The reference images are named as follow: `Img_REF_cl_XX_YY.jpg`, where XX is the number of the class (label) and YY is the number of the reference image for a given class. An example of reference image for each of the classes is shown on Figure 2.

In order to quantify the performance of the dune detection algorithm a set of test images was labeled. As with the reference images, each test images contain only one class. The size of the test images is not fixed but was limited by the size of uniform zone for a given class found on the images of the *HiRISE* instrument. The size of the test images ranges from 130 x 80 pixels

to 900 x 500 pixels. The naming convention for the test images (`Img_TEST_cl_XX_YY.jpg`) is the same as the one for the references images (`Img_REF_cl_XX_YY.jpg`).

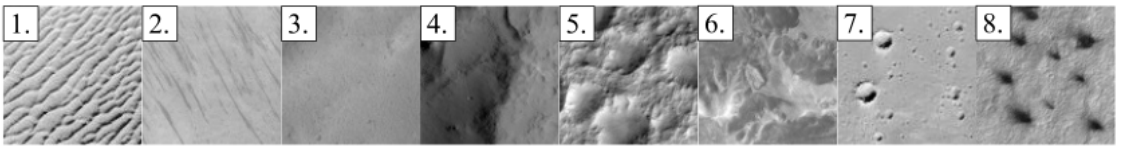

Figure 2: Example of reference data for each class. 1: Dunes. 2: Flood trace on tilted surfaces. 3: Flat surface. 4: Ridges and small mountains. 5: Hight density of craters. 6: Small irregularity on flat surfaces. 7: Flat surface with small craters. 8: Strange dark feature.

3.2.2 Dunes detection algorithm

The dune detection algorithm was implemented in MATLAB. A flowchart of the entire program is shown on Figure 3. It consist of a first part where the reference images are loaded after which the LBP algorithm is applied to each reference image. The data extracted from the reference images by to the LBP is then used, together with the labels of the reference images, to train a k-nearest neighbors model. Once the k-nearest neighbors model is trained, the test data is loaded and then the LBP algorithm is applied to each test image. After that, a classification of test images is performed, which then allows the computation of the confusion matrix and Kappa statistics.

Once the performance of the algorithm is computed, the actual analysis of images can start. First, the name and the georeferenced data of all the images that need to be analyzed are loaded and put in a cell by reading each .LBL file in the folder where the images are stored. With this data, the total number of images that need to be analyzed is computed. This value is used to set a FOR loop in which one image is analyzed at each cycle of the loop.

The FOR loop starts by loading an image. Then, histogram matching is performed with a reference image (always the same one). After that, the image is cropped to a multiple of the LBP window (see subsection 3.2.3) and then, the LBP algorithm is applied to the cropped image. Once the LBP has been computed, the classification of the image is performed. This allows the creation of a BW image with black pixels in zones without dunes and white pixels in zones where dunes were detected.

This image is then used to compute the total number of dunes present on the image (see subsection 3.2.4) and the area covered with dune zones. The percentage of the image covered with white dune zones together with the position of the center of the image (latitude and longitude) is then saved in a .TXT file (for each image, a new line is added in the .TXT file). Finally, an image with a red overlay of the dune zones is created and then saved. The overlaid image has the same name as the original image, but the prefix `Dunes_overlay_` is added. Then another cycle of the FOR loop starts. This is repeated until all the images have been analyzed.

3.2.3 LBP algorithm

First, the image is divided into cells of 30x30 pixels (LBP window). Then, the LBP histogram is computed for each cell. Once computed, the LBP histograms are put in an array (one histogram per line) and then each histogram is normalized using L1 norm.

3.2.4 Dunes counter

To perform the counting of dunes, edge detection is performed on the original image (eg. the image that needs to be analyzed). Then several operations are performed with the MATLAB function bwmorph to reduce each edge into one single pixel. This image is then multiplied by the BW image of the dune zones. This allow the authors to suppress all the single pixels that are not in a dune zone. Finally, the number of white pixels is counted. This value is the number of dunes present on the image.

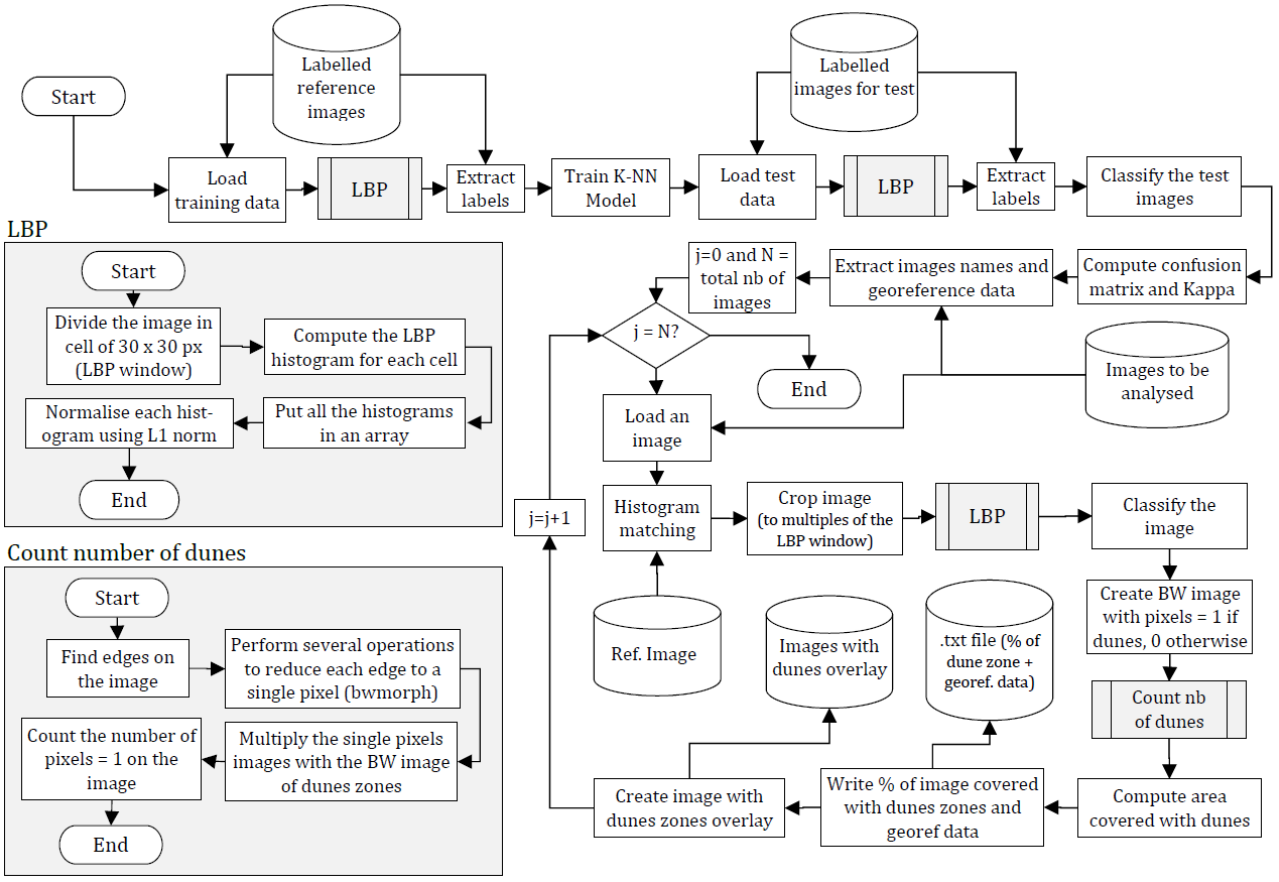

Figure 3: Flowchart of the dune detection program implemented in MATLAB.

3.3 Dune Mapping

The base map of Mars, on which the dune coverage was overlaid, was created from the images obtained from the *MOC* instrument. These images were also used to create the *MOC Atlas of Mars*, previously mentioned in section 1.3. The 29 images comprising this atlas were downloaded from `http://www.msss.com/mgcwg/mgm/` and mosaicked together to create a base map of Mars. While this map covers the entire longitude of the planet (from -180 to 180 degrees), it does not cover the entire latitude of the planet, covering the range from 90 to -65 degrees. The south pole image was omitted because it does not fit adequately with the other ones. However this does not pose a large problem as there are relatively few available images from the *HiRISE* instrument that cover the poles.

As stated in subsection 3.2.2, while performing image analysis on individual images, the percentage of dune coverage for a specific image is saved to a .TXT file along with the position of that image (latitude and longitude). The information contained in the file is used in this step of the processing routine. Each row of the file contains a triplet: [latitude, longitude, dune_coverage]. For each row of the file, a circle on the map is placed at the corresponding coordinates (latitude, longitude) and its transparency was set to reflect the dune_coverage intensity.

4 Results

4.1 Image analysis

The Figure 4 (left) shows an image overlaid with dune zones after the analysis. Globally it appears that the dune detection algorithm works well.

As the spatial resolution of the LBP algorithm is rather low (30 x 30 pixel (ca. 15 x 15 m), small dunes are not detected (red ellipses on Figure 4 (a) and (c)). This could be improved by reducing the size of the LBP window.

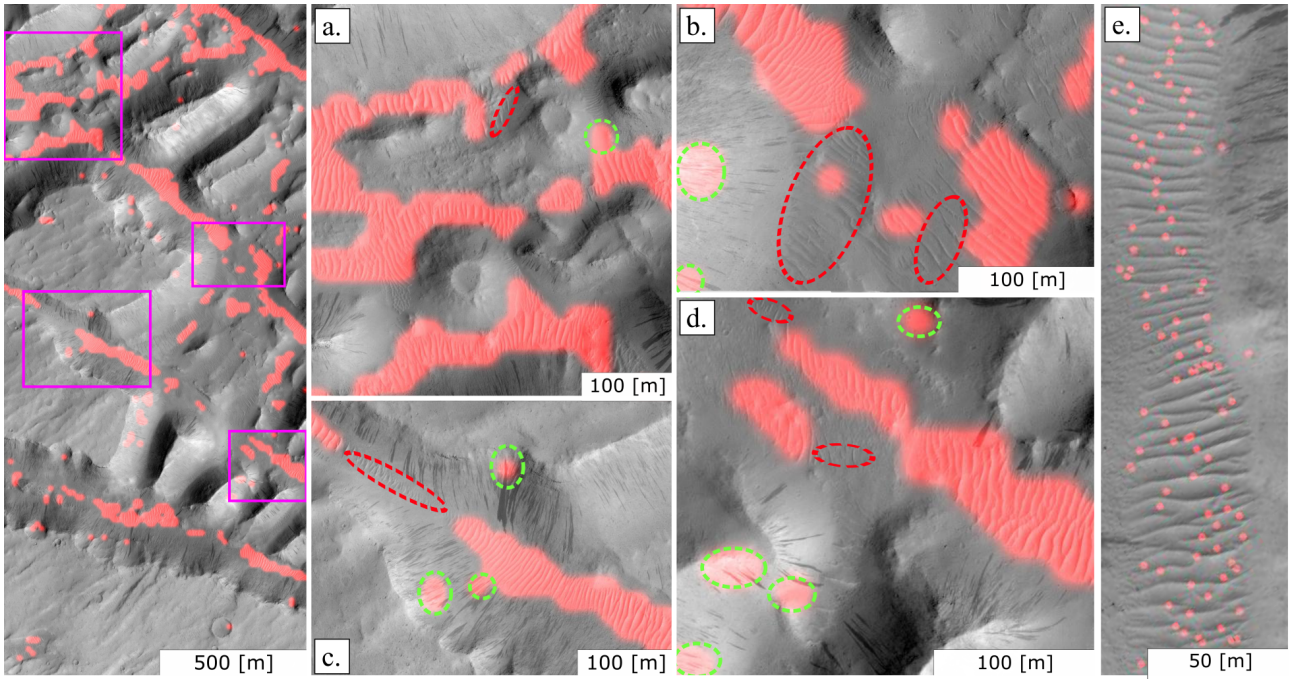

Figure 4: Right: Example of image overlaid with dunes zones (north is up). (a): Zoom in the upper-left pink rectangle. (b): Zoom in the upper-right pink rectangle. (c): Zoom in the lower-left pink rectangle. (d): Zoom in the lower-right pink rectangle. Red dashed ellipse highlight zones that should have been overlaid (dunes). Green dashed ellipses highlight overlaid zones that are not dunes. (e): Evaluation of the performances of the dune counter.

There are also zones without dunes that are overlaid in red (see green ellipses on figure 4). It seems that this occurs mainly with the classes "Flood traces on tilted surfaces" and "High density of craters". This is confirmed with the omission error (respectively 1.45% and 2.19%) on the confusion matrix, as shown in Table 2.

With the confusion matrix, it appear that the performance for the dunes detection is very good (UA = 79.1 % and PA = 82.6 %), especially if the fact that the implemented dune detection algorithm is fast is taken into account. Indeed, it takes less than 20 seconds to analyze one image of 2040 x 4500 pixels (MATLAB running on a laptop with Intel Core i7-4800MQ 2.70GHz and 16 GB of RAM).

The performance for the detection of the other classes is much lower. This is due to the fact that the algorithm was optimized for the detection of dunes only. Indeed, for the supervised classification, 28 reference images (of 100 x 100 pixels) were used for the class "Dune" whereas for the other classes only between 3 and 5 reference images were used (exception for the class "Flood traces on tilted surfaces" where 12 reference images were used as this class could be easily confused with dunes). The only exception is the class "Strange dark feature" were the UA and PA are excellent (with only 3 references images). This is due to the fact the texture of these features is very particular and occurs only at specific places on Mars where there is nothing else.

The value of Cohen's kappa coefficient is 0.44 which is rather low. This is due to the reasons explained in the previous paragraph.

A test done with the implemented dune counter algorithm was performed in order to characterize it, where the counted pixels (see subsection 3.2.4) were highlighted in red (see figure 4 (e)). On the figure, the number of red dots is 110 whereas the number of dunes is 72. So it appears that the dune counter overestimates the number of dunes by a factor of 1.5. This overestimation occurs because when there are many small dunes there is also a lot of small edges and thus more dots than dunes appears in these zones. The performance could be increased by playing with the threshold of the edge detection, which can be done as part of the further improvements.

[%]		True labels								UA
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
Predicted labels	**1**	**21.23**	0.68	1.90	0.99	1.53	0.00	0.47	0.02	**79.14**
	2	1.45	**11.48**	4.01	0.41	0.04	0.17	1.47	0.04	**60.24**
	3	0.00	0.00	**2.42**	0.50	0.23	0.00	0.43	0.04	**66.86**
	4	0.50	1.20	4.38	**2.17**	2.58	0.60	0.54	0.10	**17.98**
	5	2.19	4.52	6.94	3.04	**2.75**	0.39	1.55	0.64	**12.48**
	6	0.12	0.04	0.31	0.17	0.10	**0.14**	0.10	0.17	**10.71**
	7	0.02	0.08	1.57	0.23	0.12	0.00	**0.31**	0.00	**13.27**
	8	0.21	0.00	0.12	0.02	0.00	0.04	0.35	**12.20**	**94.26**
PA		**82.57**	**63.76**	**11.16**	**28.85**	**37.36**	**10.77**	**5.93**	**92.34**	**52.69**

Table 2: Confusion Matrix. 1: Dunes. 2: Flood trace on tilted surfaces. 3: Flat surface. 4: Ridges and small mountains. 5: High density of craters. 6: Small irregularity on flat surfaces. 7: Flat surface with small craters. 8: Strange dark feature. All values in [%].

4.2 Dune Mapping

The results of the dune mapping phase are shown on Figure 5. In total, approximately 2000 images have been processed and their dune coverage plotted. It can be noted that most of the detected dunes are located near or south of the equator. Most prominent areas with high percentage of dune coverage are *Valles Marineris* (highlighted in green), north-western slopes of *Hellas Planitia* (highlighted in blue), the region around *Olympus Mons* (highlighted in white) as well as the south-eastern slopes of *Chryse Planitia* (highlighted in purple). This map can be compared with the Mars Global Digital Dune Database (MGD[3]) available at this website: `https://astrogeology.usgs.gov/geology/mars-dunes/the-mars-global-digital-dune-database`. It can be seen that all of the highlighted areas, except the *Olympus Mons* one, contain large or moderate-sized dune fields, which coincides with the findings of this report. The discrepancy between the results might come from the fact that the MGD[3] includes only moderate and large size dune fields, while here individual images that cover a much smaller area were analyzed.

However it is worth noting that, due to constraints in time and available processing power, only 4% of the entire available catalog of *HiRISE* images (2000 of approximately 52,000) was analyzed. Even with this small subset of data, results comparable to the ones from the MGD[3] were obtained.

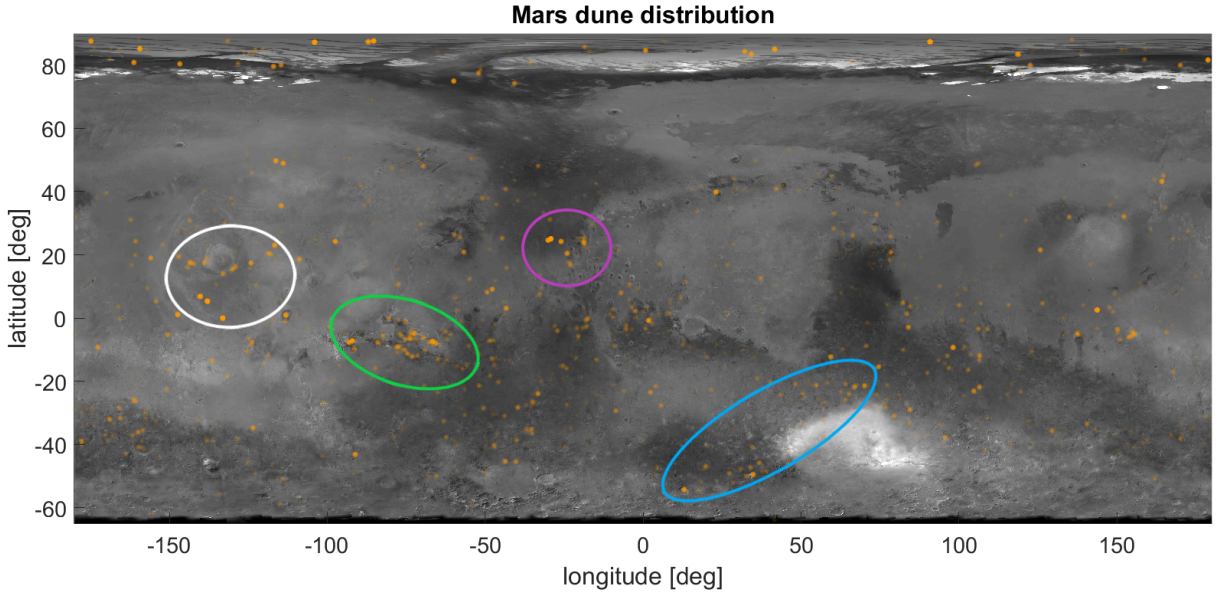

Figure 5: Map of dune distributions on Mars with some of the areas with high dune coverage highlighted.

5 Conclusion

5.1 Summary

The aim of this project was to detect dunes on high-resolution images of Mars and to create a general overview of the dune distribution on its surface. Firstly, some historical elements and geographical concepts were explained, in order to introduce the reader to the topic. Then, each part of the processing routine was explained. The details of the implementation in Python and Matlab are given for the data extraction, image analysis and dune mapping. In the end, the results that were obtained are further discussed and compared to some previous research findings. This project demonstrated that significant and meaningful image processing results can be attained, even with limitations in time and processing power.

5.2 Further improvements

The authors of this report see a lot of improvements that could be done in the future. Firstly, applying the dune detection algorithm to the entire image catalog of the *HiRISE* instrument would make the map of dune distributions more detailed and closer to ground truth.

Another potential improvement would be to apply a sliding window to the LBP algorithm in order to improve accuracy of dune detection. However, this would increase the processing time so a compromise would be needed. An approach which would improve the results without affecting the computational time significantly could be to perform the existing algorithm two times: once with a small LBP window, which would allow the algorithm to detect small isolated dunes, and the second time with a big LBP window that would detect more homogeneous dunes with a large surface area.

One interesting feature that would allow the potential users of this system to approach this topic in a more engaging manner could be the creation of an interactive map of dune distributions. The users could click on a specific region of Mars where dunes were detected, and the map would show the original analyzed image of that location with the displayed overlay of dune coverage.

References

[1] Active Missions on Mars. http://www.planetary.org/explore/space-topics/space-missions/missions-to-mars.html. accessed: 28.12.2017.

[2] First TV Image of Mars. http://www.directedplay.com/first-tv-image-of-mars. accessed: 26.12.2017.

[3] HiRISE Camera Technical Specifications. https://www.uahirise.org/teknikos.php. accessed: 27.12.2017.

[4] Information for Scientific Users of HiRISE Color Products. https://www.uahirise.org/pdf/color-products.pdf. accessed: 30.12.2017.

[5] Mars Reconnaissance Orbiter as a Communications Relay. https://mars.nasa.gov/mro/mission/timeline/mtcommunicationsrelay/. accessed: 26.12.2017.

[6] Mars Reconnaissance Orbiter Flight Path. https://mars.jpl.nasa.gov/mro/newsroom/pressreleases/20060912a.html. accessed: 26.12.2017.

[7] MOC Atlas of Mars. http://www.msss.com/mgcwg/mgm/. accessed: 27.12.2017.

[8] Readme file for HiRISE EDR and RDR Archive Volumes. https://hirise-pds.lpl.arizona.edu/PDS/AAREADME.TXT. accessed: 30.12.2017.

[9] USGS Astrogeology Science Center - Description of dunes on Mars. https://astrogeology.usgs.gov/geology/mars-dunes. accessed: 26.12.2017.

[10] A. Azzaoui, A. Manare, H. Elbelrhiti, I. E. Chaouki, and C. Masmoudi. Detection of barchan dunes in high resolution satellite images. XLI-B7:153–160, 06 2016.

[11] R. K. Hayward, K. F. Mullins, L. K. Fenton, T. M. Hare, T. N. Titus, M. C. Bourke, A. Colaprete, and P. R. Christensen. Mars global digital dune database and initial science results. *Journal of Geophysical Research: Planets*, 112(E11), 2007.

[12] A. C. Sparavigna. Edge-detection applied to moving sand dunes on mars. *CoRR*, abs/1308.5315, 2013.

Annex 1: Code

Data Extraction (Python)

```
"""Script for downloading images from the HiRISE PDS.
Authors: Michael Pellet and Nemanja Stojoski
26.12.2017
"""
import requests
import shutil
from bs4 import BeautifulSoup
import os

#list of keywords that are extracted from the .LBL files
keywords = ['MAP_PROJECTION_ROTATION', 'MAP_RESOLUTION', 'MAP_SCALE',
            'MAXIMUM_LATITUDE', 'MINIMUM_LATITUDE',
            'LINE_PROJECTION_OFFSET', 'SAMPLE_PROJECTION_OFFSET',
            'EASTERNMOST_LONGITUDE', 'WESTERNMOST_LONGITUDE']

def download_image_with_lbl(image_url, image_name):
    """A function for downloading a single image.
    This function downloads the image with the image_name
    from the specified image_url."""
    global download_count
    #increase the download count
    download_count += 1
    #get image
    r_img = requests.get(image_url + image_name, stream = True)
    url_list = image_url.split('/')
    label_list = url_list[:4] + url_list[5:]
    label_url = ('/').join(label_list)
    label_name = image_name.split('.')[0] + '.LBL'
    image_path = 'D:/IPEO/Project/images/'
    file_content = []
    #200 - HTTP status code for successful request
    if r_img.status_code == 200:
        #create folders to the image if they don't exist
        if not os.path.exists(image_path):
            os.makedirs(image_path)
        print('Downloading ' + str(download_count) + ': '
                + image_url + image_name)
        #write image into file
        with open(image_path + '/' + image_name, 'wb') as f:
            for chunk in r_img:
                f.write(chunk)
        #get label
        r_url = requests.get(label_url + label_name)
        if r_url.status_code == 200:
            print(label_url + label_name)
```

```
            for line in r_url.text.split('\n'):
                    pair = line.split('=')
                    #extract all keyword values
                    if pair[0].strip() in keywords:
                        #add them to a list
                        file_content.append(pair[1].strip().split(' ')[0])
            with open(image_path + '/' + label_name, "w") as label_file:
                #print list joined with ', ' to file
                print(f"{', '.join(file_content)}", file = label_file)

def download_all_images(base_url, ext):
    """A function for downloading multiple images.
    This function downloads all images with the extension ext
    from the specified base_url."""
    r = requests.get(base_url)
    soup = BeautifulSoup(r.text, 'html.parser')
    #iterate through all orbits
    for link in soup.findAll('a'):
            if link.string != '../':
                orbit_url = base_url + link.string
                r = requests.get(orbit_url)
                soup = BeautifulSoup(r.text, 'html.parser')
                #iterate through all images in an orbit
                for link in soup.findAll('a'):
                    if link.string != '../':
                        img_url = orbit_url + link.string
                        image_name = link.string[:-1] + ext
                        download_image_with_lbl(img_url, image_name)

#specify the base_url from which the images are downloaded
base_url = 'https://hirise-pds.lpl.arizona.edu/PDS/EXTRAS/RDR/ESP/'
#specify the image extension
ext = '_RED.NOMAP.browse.jpg'
download_count = 0
#start download
download_all_images(base_url, ext)
```

Image analysis (MATLAB)

Main code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% REMOTE DUNES DETECTION ON MARS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nemanja Stojoski and Michael Pellet
% 10.12.2017

clear all
```

```
close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Image_directory='F:\EPFL\2_Master\Semestre 3\Earth ...
    Imaging\Project\Images_to_analize'; %Directory where there are the images
%Image_directory='F:\EPFL\2_Master\Semestre 3\Earth ...
    Imaging\Project\LBP_Dunes_Mars\Images\ESP_052355_2070';
Save_image_overlayed = 1;   % 1 if you want to save a JPG image with an ...
    overlay of the zones with dunes 0 if you don't want to save the images ...
    (just compute percentage of surface covered with dune and write it in ...
    a .txt file)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PARAMETERS FOR COMPUTATION OF LBP AND K-NN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LBP
LBPWindowSize = 30; %Size of the cells when computing the LBP
Nb_of_Cells_LBP = 3; %(Number of cells in the REF_window = Nb_of_Cells_LBP^2)

numNeighbors = 8; %Number of neighbors
R=2; %Radius of circular pattern to select neighbors
rotLBP=true; %Rotation

%% K-NN
k_knn = 9; %Number of classes

%%
ThresholdBinarize = 1.5; %Threshold to have only the 1st class (dunes) ...
    when making the BW image of dunes zone (pixels = 1 if dunes zone, 0 ...
    otherwise)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LOADING AND LBP COMPUTATION OF REFERENCE DATA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Read the references images and put all of them into a cell

Ref_dir='Images\References\';
d = dir(strcat(Ref_dir,'Img_REF_cl_*_*.jpg'));
NameRefImageFile = {d.name};

for j=1:size(NameRefImageFile,2)
    REFj = imread(cell2mat(strcat(Ref_dir,NameRefImageFile(j))));
    REFj = rgb2gray(REFj);
    REFj = imcrop(REFj, [0 0 ...
        (LBPWindowSize*floor(size(REFj,2)/LBPWindowSize))...
        (LBPWindowSize*floor(size(REFj,1)/LBPWindowSize))]); %crop the ...
           image so it size is a multiple of the LBP window

     REF(1,j) = {REFj};
end

%% Compute the LBP reference data for training

% Change the number of bins depending if rotation or not
if rotLBP == false
    numBins = numNeighbors+2;
```

```
55 else
56     numBins = numNeighbors*(numNeighbors-1)+3;
57 end
58
59 for j=1:size(NameRefImageFile,2)
60
61 % Extract unnormalized LBP features
62     lbpFeaturesREFj = extractLBPFeatures(REF{j},'CellSize',[LBPWindowSize...
63         LBPWindowSize],'Normalization','None','Radius',R,'Upright',rotLBP);
64 %%
65 % Reshape the LBP features into a _number of neighbors_ -by- _number of ...
      cells_ array to access histograms for each individual cell.
66     lbpCellHistsREFj = reshape(lbpFeaturesREFj,numBins,[]);
67 %%
68 % Normalize each LBP cell histogram using L1 norm.
69     lbpCellHistsREFj = ...
         bsxfun(@rdivide,lbpCellHistsREFj,sum(lbpCellHistsREFj));
70 %%
71     if j==1
72         lbpCellHistsREF=lbpCellHistsREFj;
73     else
74         lbpCellHistsREF = [lbpCellHistsREF,lbpCellHistsREFj];
75     end
76
77 end
78
79 %% LBP data for training
80 LBP_data_train_sc = transpose(lbpCellHistsREF);
81
82 %% Creat a vector with the labels of the training datas
83 for j=1:size(NameRefImageFile,2)
84     a = cell2mat(NameRefImageFile(j));
85     LBP_label_trainj = a(12:13);
86
87     if j==1
88         LBP_label_train= str2num(LBP_label_trainj)...
89             *ones(1,Nb_of_Cells_LBP^2);
90     else
91         LBP_label_train = [LBP_label_train,str2num(LBP_label_trainj)...
92             *ones(1,Nb_of_Cells_LBP^2)];
93     end
94 end
95
96 LBP_label_train = transpose(LBP_label_train);
97
98 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99 %% LOADING AND LBP COMPUTATION OF TEST DATA
100 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101 %% Read the tests images and put all of them into a cell
102
103 Test_dir='Images\References\Test_Data\';
104 d = dir(strcat(Test_dir,'Img_TEST_cl_*_*.jpg'));
105 NameTESTImageFile = {d.name};
106
107 for j=1:size(NameTESTImageFile,2)
```

```
108     TESTj = imread(cell2mat(strcat(Test_dir,NameTESTImageFile(j))));
109     TESTj = rgb2gray(TESTj);
110     TESTj = imcrop(TESTj, [0 0 ...
111         (LBPWindowSize*floor(size(TESTj,2)/LBPWindowSize))...
112         (LBPWindowSize*floor(size(TESTj,1)/LBPWindowSize))]); %crop the ...
            image so it size is a multiple of the LBP window
113
114      TEST(1,j) = {TESTj};
115 end
116
117 %% Compute the LBP data for testing
118
119 for j=1:size(NameTESTImageFile,2)
120
121 % Extract unnormalized LBP features
122     lbpFeaturesTESTj = ...
          extractLBPFeatures(TEST{j},'CellSize',[LBPWindowSize...
123         LBPWindowSize],'Normalization','None','Radius',R,'Upright',rotLBP);
124 %%
125 % Reshape the LBP features into a _number of neighbors_ -by- _number of ...
      cells_ array to access histograms for each individual cell.
126     lbpCellHistsTESTj = reshape(lbpFeaturesTESTj,numBins,[]);
127 %%
128 % Normalize each LBP cell histogram using L1 norm.
129     lbpCellHistsTESTj = ...
          bsxfun(@rdivide,lbpCellHistsTESTj,sum(lbpCellHistsTESTj));
130 %%
131     if j==1
132         lbpCellHistsTEST=lbpCellHistsTESTj;
133     else
134         lbpCellHistsTEST = [lbpCellHistsTEST,lbpCellHistsTESTj];
135     end
136
137 end
138
139 %% LBP data for test
140 LBP_data_test_sc = transpose(lbpCellHistsTEST);
141
142 %% Creat a vector with the labels of the test datas
143 for j=1:size(NameTESTImageFile,2)
144     a = cell2mat(NameTESTImageFile(j));
145     LBP_label_testj = a(13:14);
146
147     if j==1
148         LBP_label_test= str2num(LBP_label_testj)...
149             *ones(1,((floor(size(TEST{j},2)/LBPWindowSize))...
150             *(floor(size(TEST{j},1)/LBPWindowSize))));
151     else
152         LBP_label_test = [LBP_label_test,str2num(LBP_label_testj)...
153             *ones(1,((floor(size(TEST{j},2)/LBPWindowSize))...
154             *(floor(size(TEST{j},1)/LBPWindowSize))))];
155     end
156
157 end
158
```

```
159 LBP_label_test = transpose(LBP_label_test);
160
161 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
162 %% TRAINING OF K-NN MODEL
163 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
164
165 typeNorm = 'minmax'; % use 'std' to rescale to a unit variance and zero mean
166 [LBP_data_train_sc, dataMax, dataMin] = ...
167     classificationScaling(double(LBP_data_train_sc), [], [], typeNorm);
168 LBP_data_test_sc = ...
169     classificationScaling(double(LBP_data_test_sc), dataMax, dataMin, ...
        typeNorm);
170
171 % Train a k-NN model
172 LBP_model_knn = ...
    fitcknn(LBP_data_train_sc,LBP_label_train,'NumNeighbors',k_knn);
173
174 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
175 %% COMUPTE ACRUACY MEASURES
176 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
177
178 % Run the trained classifier on the validation set
179 class_knn_test = predict(LBP_model_knn,LBP_data_test_sc);
180
181 % Get the Confusion tables
182 CT_knn = confusionmat(LBP_label_test, class_knn_test); % build confusion ...
    matrix
183
184 % Get OVerall Accuracies
185 OA_knn = trace(CT_knn)/sum(CT_knn(:));
186
187 % Get Kappa statistics
188 CT_knn_percent=CT_knn./sum(sum(CT_knn));
189 EA_knn = sum(sum(CT_knn_percent,1)*sum(CT_knn_percent,2));
190 Ka_knn= (OA_knn - EA_knn)/(1-EA_knn);
191
192 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
193 %% LOADING AND LBP COMPUTATION OF IMAGES (WITH DUNES(MAYBE:)))
194 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
195 %%Load the image name and directory as well as the txt data for each image
196 [Directory,NameImageFile,TxtData]=Read_text_file(Image_directory);
197
198     %Signification of each line in TxtData:
199     %   1:  MAP_PROJECTION_ROTATION [deg]
200     %   2:  MAP_RESOLUTION [px/deg]
201     %   3:  MAP_SCALE [m/px]
202     %   4:  MAXIMUM_LATITUDE [deg]
203     %   5:  MINIMUM_LATITUDE [deg]
204     %   6:  LINE_PROJECTION_OFFSET [px]
205     %   7:  SAMPLE_PROJECTION_OFFSET [px]
206     %   8:  EASTERNMOST_LONGITUDE [deg]
207     %   9:  WESTERNMOST_LONGITUDE [deg]
208
209 %% Start the for loop in which all the images will be analysed
210 for j=1:size(NameImageFile,2)
```

```
211
212     %Calculate the latitude at the center of the image
213     Latitude = (TxtData(4,j)+TxtData(5,j))/2;
214
215     %Calculate the longitude at the center of the image
216     Longitude = (TxtData(8,j)+TxtData(9,j))/2;
217
218 %% Load reference image for histogram matching
219     IMG_HIST_MATCH = imread('Images\References\Histogram_matching.jpg');
220
221 %% Load and put all the images in a cell of 1 x nb.images and do ...
    histogram matching with the reference image
222
223     IMG = imread(cell2mat(strcat(Directory(j),'\',NameImageFile(j))));
224     IMG = imhistmatch(IMG,IMG_HIST_MATCH); %
225     IMG = imcrop(IMG, [0 0 ...
        (LBPWindowSize*floor(size(IMG,2)/LBPWindowSize))...
226         (LBPWindowSize*floor(size(IMG,1)/LBPWindowSize))]); %crop the ...
            image so it size it a multiple of the LBP window
227
228 %% LBP
229
230     % Extract unnormalized LBP features so that you can apply a custom ...
        normalization.
231     lbpFeatures = extractLBPFeatures(IMG,'CellSize',[LBPWindowSize...
232         LBPWindowSize],'Normalization','None','Radius',R,'Upright',rotLBP);
233     %%
234     % Reshape the LBP features into a _number of neighbors_ -by- _number ...
        of cells_ array to access histograms for each individual cell.
235     lbpCellHists = reshape(lbpFeatures,numBins,[]);
236
237     % Normalize each LBP cell histogram using L1 norm.
238     lbpCellHists = bsxfun(@rdivide,lbpCellHists,sum(lbpCellHists));
239
240     LBP_data_sc = transpose(lbpCellHists);
241
242     %% Rescale accordingly all image pixels
243     LBP_data_sc = classificationScaling(double(LBP_data_sc),...
244         dataMax, dataMin, typeNorm);
245
246 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
247 %% CLASSIFICATION OF IMAGE
248 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
249
250     statLoop=j %Jute to see the progression of the for loop
251
252     % Classifying entire image for k-NN:
253     LBP_class_knn = predict(LBP_model_knn,LBP_data_sc);
254
255     %% Creat an image were pixels =1 if in dunes zone, 0 otherwise:
256     LBP_class_knn_MAT = transpose(vec2mat(LBP_class_knn,...
257         floor((size(IMG,1)/LBPWindowSize))));  %Creat a matrix of the size ...
            of the numbers of cells in the image
258     LBP_class_knn_MAT_BW = ...
        imbinarize(LBP_class_knn_MAT,ThresholdBinarize); % Binarize with a ...
```

```
        threshold of 1.5 so there is only the classe 1 (dunes) that is ...
        remaining
259     LBP_class_knn_MAT_BW_comp = imcomplement(LBP_class_knn_MAT_BW); % To ...
        actualy have dunes => pixels =1, other stuff => pixels =0
260     LBP_class_knn_MAT_BW_comp = ...
261         imresize(LBP_class_knn_MAT_BW_comp ,[size(IMG)],'bicubic'); ...
            %Resize the image of dunes zone so it's the same as the ...
            original image
262
263     %% Dunes counting on the image
264
265     Number_of_dunes = Dunes_counter(IMG,LBP_class_knn_MAT_BW_comp);
266
267     %% Evalation of the area covered with dunes on the image
268
269     [Area_of_dunes,Percent_of_dunes] = ...
        Dunes_area(TxtData(3,j),LBP_class_knn_MAT_BW_comp);
270
271     %%
272     %Open text file at the begining
273     if j==1
274         fileID = fopen('Dunes_position.txt','w');
275     else
276     end
277
278     %Write the Latitude, Longitude and Percentage of surface covered with ...
        dunes on the image in the text file:
279     A = [Latitude; Longitude; Percent_of_dunes];
280     fprintf(fileID,'%17.17f, %17.17f, %06.5f\r\n',A);
281
282     %Close text file at the end
283     if j==size(NameRefImageFile,2)
284         fclose(fileID);
285     else
286     end
287     %% Create and save an image with red dunes overlay
288     if Save_image_overlayed==1
289         Dunes_overlay = cat(3,255*uint8(LBP_class_knn_MAT_BW_comp),...
290             zeros(size(LBP_class_knn_MAT_BW_comp)),...
291             zeros(size(LBP_class_knn_MAT_BW_comp)));
292
293             Dunes_overlay_smooth = imgaussfilt(Dunes_overlay,10); %to ...
                smooth a bit the overlay (as the overlay is made of cells ...
                of 30x30 px (so big squars)
294         IMG_and_dunes_overlay = 1*cat(3, IMG, IMG, IMG)...
295             + 0.9*Dunes_overlay_smooth;
296
297         Name_images_overlayed = cell2mat(strcat(...
298             'Images\Output_Images\','Dunes_overlay_',NameImageFile(j)));
299         imwrite(IMG_and_dunes_overlay,Name_images_overlayed,'jpg');
300     else
301     end
302 end
```

Function for counting the number of dunes

```
function [Number_of_Dunes] = Dunes_counter(IMG,dunes_zone)
%% Function used to count the number of dunes
%%Code in comment used to make figure with red points in the repport
%(with code at the end)
%IMG_preserved=IMG;

%Edge detection:
[IMG,threshold] = edge(IMG,'Prewitt',[]);
IMG = edge(IMG,'Prewitt',threshold*1.2);
%Reduction of the edges to singles pixels:
IMG = bwmorph(IMG,'clean');
IMG = bwmorph(IMG,'thicken');
IMG = imfill(IMG, 'holes');
IMG = bwmorph(IMG,'shrink',inf);
%Multiplication of the singe pixels image with the image where dunes zone ...
    are white (so there are singles pixels only were there are dunes)
IMG=immultiply(IMG,dunes_zone);


%%Code in comment used to make figure with red points in the report
%IMG = bwmorph(IMG,'thicken',1);
% IMG = cat(3,255*uint8(IMG),...
%             zeros(size(IMG)),...
%             zeros(size(IMG)));
%    IMG_and_dunes_overlay = 1*cat(3, IMG_preserved, IMG_preserved,...
%   IMG_preserved) + 0.9*IMG;
%         Name_images_overlayed = strcat(...
%             'Images\Output_Images\','Count_Dunes_overlay_test');
%         imwrite(IMG_and_dunes_overlay,Name_images_overlayed,'jpg');

Number_of_Dunes = round(sum(IMG(:)));
```

Function for calculating the surface covered with dunes

```
function [Area_of_Dunes,Percent_of_Dunes] = Dunes_area(resolution,dunes_zone)
%% Function used to calculate the area covered with dunes
%count the number of pixels in zones where there is dunes (eg white pixels)
Number_of_pixel = round(sum(dunes_zone(:)));

%claculate the area where there are dunes (in m^2)
Area_of_Dunes = Number_of_pixel*(resolution^2)

%calculate the ratio of the image covered with dunes
Percent_of_Dunes = Number_of_pixel/(size(dunes_zone,1)*size(dunes_zone,2))
```

Function for reading the text file

```
function [Directory,NameImageFile,TxtData]=Read_text_file(Image_directory)
%% Function used to read the name of the images and the text data (with ...
    lat, long, resolution, etc...)

d = dir(strcat(Image_directory,'\ESP_*_*_RED.LBL'));
NameTxtFile = {d.name};

d = dir(strcat(Image_directory,'\ESP_*_*_RED.NOMAP.browse.jpg'));
NameImageFile = {d.name};

Directory={d.folder};

for j=1:size(NameImageFile,2)
    fileID = fopen(strcat(char(Directory(j)),'\',char(NameTxtFile(j))),'r');
    TxtDataj = cell2mat(textscan(fileID, '%f', 'Delimiter',','));
    fclose(fileID);

    if j==1
        TxtData=TxtDataj;
    else
        TxtData = [TxtData,TxtDataj];
    end

end
```

Dune mapping (MATLAB)

```
%% LOADING AND MOSAICKING BASE MAP IMAGES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
numImages = 29;
map_images = cell(1, numImages);

for k = 1:numImages
    %load all images
    map_images{k} = imread(strcat('Project/mc', sprintf('%02d.jpg',k)));
end

%merge the images north to south, east to west
mapTop = cat(1, map_images{1}, cat(2, map_images{2:7}));
mapUpper = cat(1, mapTop, cat(2, map_images{8:15}));
mapMiddle = cat(1, mapUpper, cat(2, map_images{16:23}));
mapLower= cat(1, mapMiddle, cat(2, map_images{24:29}));
%uncomment the next line to include the south pole
%map = cat(1, map_lower, map_images{30});
map = mapLower;


%% PLOTTING THE BASE MAP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

```

```
%latitude and longitude ranges
xValuesDeg = [-180 180];
yValuesDeg = [90 -65];

figure
np = newplot;
%set plot parameters
np.FontSize = 22;
np.Layer = 'top';
np.YDir = 'normal';
np.View = [0 90];
%display map
imagesc('XData', xValuesDeg, 'YData', yValuesDeg, 'CData', map)
colormap(gray);
axis equal tight
%label axes
xlabel('longitude [deg]');
ylabel('latitude [deg]');
title('Mars dune distribution');

%% PLOTTING DUNE DISTRIBUTIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%load the file with dune distributions
M = csvread('Dunes_position.txt');

%itereate through matrix rows
for row = 1 : size(M, 1)
    %process each row
    dune = M(row, :);
    %obtain dune location and coverage from row
    duneCell = num2cell(dune);
    [lat, long, coverage] = duneCell{:};
    %skip images below -65 degrees latitude
    if lat < -65
        continue
    end
    %shift longitude range from [0, 360] to [-180, 180]
    if long ≥ 180
        long = long - 360;
    end

    %draw circle at the specified position
    h = viscircles(np, [long lat], 0.4);
    %set transparency of the drawn circle
    h.Children(1).Color = [1 0.6 0 coverage];
    h.Children(2).Color = [1 0.6 0 coverage];
end
```