Feature Tracking and Optical Navigation

Aparajithan Venkateswaran Thibaud Teil

Autonomous Vehicle Laboratory – Dr. Hanspeter Schaub
University of Colorado, Boulder

Abstract

This report aims at identifying and tracking craters in images for optical navigation in space. We first survey at existing image processing techniques[3], [4] that are likely to help us in identifying craters. We then proceed to bootstrapping a deep neural network[1], [2], [8] classifier with the help of TensorFlow Object Detection API[11] and images from NASA's Detecting Crater Impact Challenge[9]. We then implement a preliminary tracking algorithm that stores images and computes mean squared error to detect if the crater has already been seen before.

1. Introduction

Ever since humans landed on the moon, it became clear than deep space travel is a possibility in the future. One of the biggest issues faced by satellites and probes that we have sent into space is that they do are unable to react to the presence of other astronomical objects real time. This means that they must rely on scientists back at Earth for navigation. In addition, they do not have the ability to recognize an incoming asteroid on their own. Satellite images get sent back to Earth for scientists to study the situation. Often, this delay can cause a catastrophe. At the least, the time delay slows down the mission progress and causes overhead.

In this report, we attempt to provide a method to track features on astronomical objects. We will first identify potential craters on the astronomical object. Then we will start tracking these potential crates and calculate how far these craters have been displaced since the last image was taken. Using this information, it is possible to calculate relative positions of the satellite. This feature tracking technique will enable optical navigation allowing for the satellite to identify and respond to incoming threats real time. Existing technology allows us to perform these actions manually. The novelty of this paper lies in the fact that the entire process will be automated with a robust and fast feature tracking method.

The rest of the paper is divided as follows: **(2)** will focus on related works done in the past; **(3)** will discuss the methods we used; **(4)** will go into the detail of the experiments performed; **(5)** will discuss the results we got from our experiments and; **(6)** will summarize the results and discuss scope for further research.

2. Related Works

There has been a lot of work, especially in the past decade, in the field of image processing and object recognition with the resurgence of neural networks and deep learning. These range from relatively shallow convolutional neural networks, introduced in *Very Deep Convolutional Neural Networks for Large-Scale Image Recognition* by Karen Simonyan and Andrew Zisserman in 2014[1], to the complex inception model introduced in *GoogLeNet* by Szegedy, Liu et. al in 2015[2]. Another interesting work was performing region based convolutions as explained by Girshick, Donahue, et. al. in *Rich feature hierarchies for accurate object detection and semantic segmentation* in 2014[3].

On the side of detecting craters, extensive research has been conducted trying to detect craters for the purpose of navigation. One particularly interesting experiment was done by Urbach and Stepinski (*Automatic detection of sub-km craters in high resolution planetary images* in 2009[4] where they invert images to extract shadows and highlighted areas of an image. After some preprocessing, they were able to extract crater candidates that was fed into a classifier. This technique relies on the fact that all craters have a highlighted and shadowed region arising from their physical structure.

In tracking, *Tracking-Learning-Detection* by Kalal, Mikolajczyk, and Matas in 2010[5], describes a method for tracking objects in videos with a self-evaluating mechanism to help the tracker increase its accuracy the longer it tracks the object. This paper also described how the tracker can remember the object and begins tracking it again even after it goes out of the frame.

Another deeply researched area is simultaneous localization and mapping (SLAM). It is aimed at solving tasks carried out by mobile robots, such as search and rescue. This is useful in tasks where no accurate map exists, and has no reliance on external GPS. Since this is used in mapping, and navigating areas where no one has gone before, this technique could be another way to approach the problem at hand. It relies on making a 3D model of the object being navigated, and has shown promising results when used with drones. This work is being done by Raul Mur-Artal et. al[8]. It has also been investigated for spacecraft rendezvous as seen in *Application of ORB-SLAM to Spacecraft Non-Cooperative Rendezvous*[7].

3. Methods

This research area is very broad and involves three big topics – identification of features, tracking of features, and navigation. This report will mainly focus on the feature identification and will briefly touch upon tracking.

Our workflow, for tackling the identification of features, is to design two different models that achieve the same results simultaneously. The first will use pure image processing techniques to identify craters. The second will be a mixture of image processing and machine learning techniques. Once both these models have been designed, we will evaluate both of them on the basis of computation speed (fastness), efficiency (hardware and power requirements), and robustness (how good are the detected craters). This study should exhibit the pros and cons of both methods, and also verify which are feasible for our intended use. Then we will either choose one of either models, or a combination of both.

To answer the second question, tracking the identified features, we will develop on the models we designed earlier and implement a tracker that keeps track of the identified features in multiple frames. One way to go about this would be to use a modified implementation of the algorithm described in *Tracking-Learning-Detection*[5]. Eventually, we will compare both implementations and once again choose the better model on the basis on speed, efficiency, and robustness.

Our results would be given to a Kalman filter in order to get the spacecraft state estimates. The reach goal of this project is to make it possible to use our algorithm with existing techniques for navigation.

4. Experiments

4.1 Machine Learning

We first implemented a simple linear regression that will output a bounding box for craters. Then we took it further and developed a deep neural network. Finally, we implemented a convolutional neural network. For these experiments, we used Python along with Numpy and TensorFlow libraries. The data was obtained from NASA[9]. We, then used the TensorFlow Object Detection API[11] that allows researchers to use pre-trained models to train on their own datasets. We used the data obtained from NASA and trained our model on a pre-trained model on COCO dataset[10] using the Fast RCNN algorithm[8]. *Figure 1* shows some sample images collected from the NASA *Detecting Crater Impact Challenge* dataset[9].

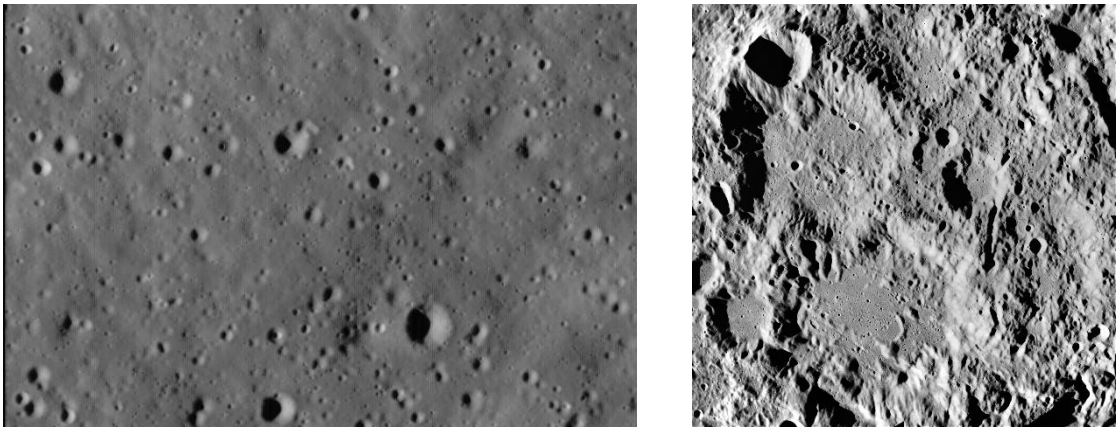

Figure 1 – Sample images from the NASA dataset

4.2 Image Processing

We first used off-the-shelf feature detection algorithms to test their robustness. Then, we used pre-processing techniques described in *Automatic detection of sub-km craters in high resolution planetary images*. This involves inverting image, background removal, applying a power filter, shape filter and combining the inverted and non-inverted images to combine highlights and shadows present in craters. This can be used to extract crater candidates. Another technique used

will be applying a Fourier transformation to study any patterns that are consistent with presence of craters. Applying convolutions to these might yield interesting results. For these simulations, we will be using Python and OpenCV[12]. OpenCV offers many off-the-shelf feature detectors like Hough Circle Finding method and Harris Corner Detection. It also offers many other image processing functionalities.

5. Results

Each subsection describes the results derived from the experiments performed in the corresponding subsection number in the **(4)**.

5.1 Machine Learning

The linear regression model performed very badly. The bounding box values that the trained model output was often negative, or beyond the size of the image. The neural network performed better in that aspect in that the bounding box was present inside the image. And it contained multiple craters inside it. However, it was not robust enough to identify significantly large craters, and picked smaller ones. This could be because one image was trained against multiple bounding boxes, which could have confused the model as to which one should be predicted. Convolutional neural networks could not be tested because they demanded large amounts of memory which could not be provided even by a powerful laptop.

The model we got from using the TensorFlow Object Detection API performed well and could predict multiple craters in an image with ~95% confidence. *Figure 2* shows an example result we got.

We were also able to track craters in a video. However, some of the craters were not continously tracked. But, this will not greatly affect our navigation system as we do not need continous measurements. At the same time, this raises the question of tagging craters so that the computer recognizes a crater as either, previously seen, or a new crater. The object detection API does not provide a mechanism for tagging identified objects and "remembering" them. Right now, we are storing the identified crater with a unique ID and comparing it subsequently identified craters to test if the newly identified has already been identified. We calculate the mean squared error between the image being tagged and previously tagged image and assign the image a new tag if the error is greater than a manually selected threshold value. This naïve method's results are not always as expected and sometimes the the same crater gets assigned with different tags and sometimes different craters get assigned with the same tag. *Figure 3* shows one such falsely tagged crater where two different craters get assigned with the same tag. This method could possibly cause more problems in the future if we start modelling lighting conditions and angles i.e., the algorithm should be able to recognize the crater in different lighting conditions and angles.

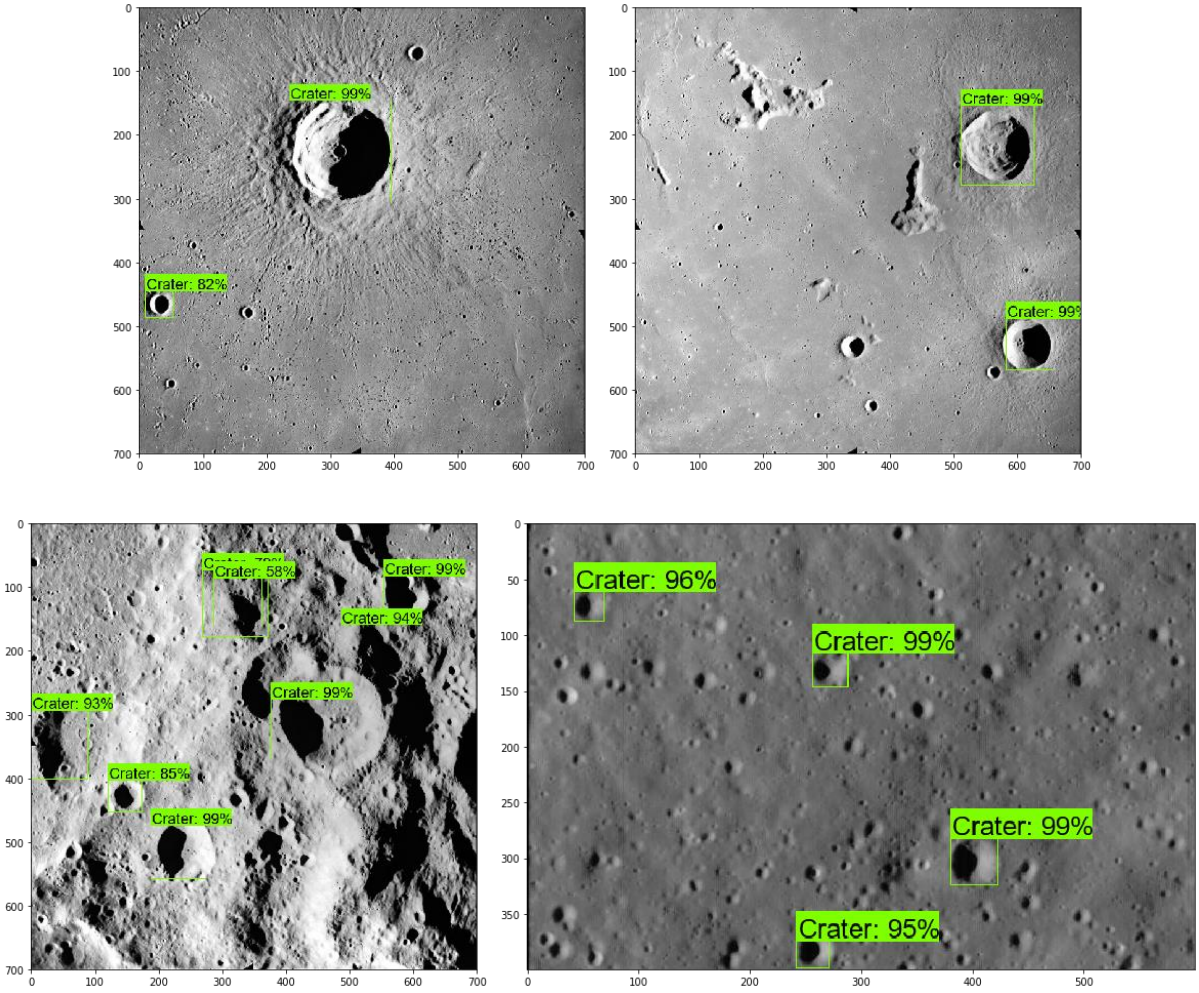

Figure 2 - Sample results

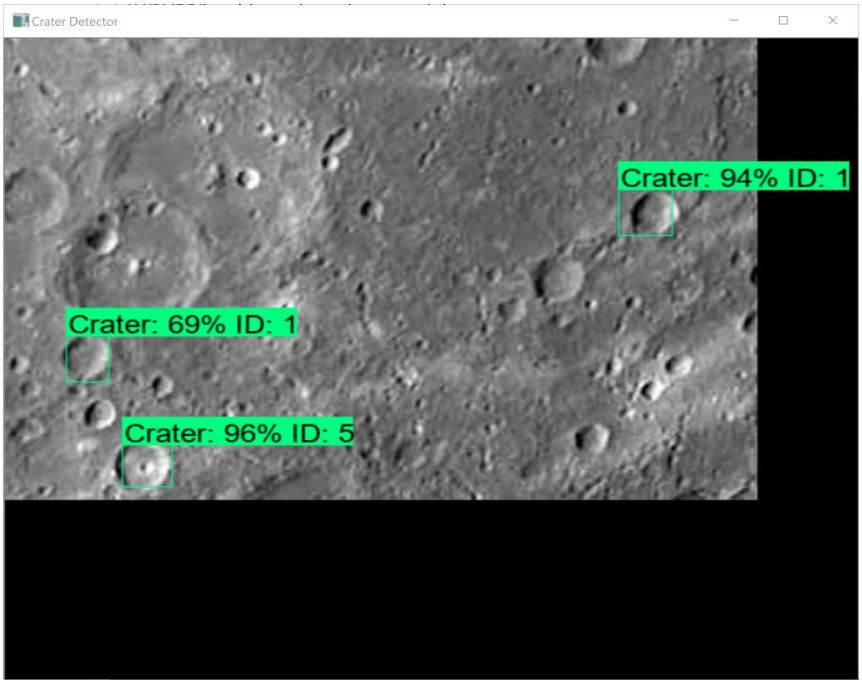

Figure 3 – Snapshot of our tagging algorithm in progress. Notice that 2 different craters get assigned the same ID.

5.2 Image Processing

Off-the-shelf feature detectors provided by OpenCV allow for decent crater findings, but need significant tuning. This leads to questions regarding robustness, and this is where machine learning might lead to better results.

We are also tried taking Fourier transforms of the image to try to better identify craters. The spectral content of the images may help to filter out unwanted information, but also help us hone in on craters and other features by identifying spectral patterns.

6. Conclusion

We conclude that traditional image processing techniques fail at identifying craters in an image. We need lots of manual fine tuning and preprocessing to make it work. At the end of the day, we want to automate this process and this is where traditional algorithms fail.

Using deep neural networks, with the TensorFlow Object Detection API, yields much better results and it is also efficient and inexpensive. But, for tracking we need to individually track the motion of craters. This is where this method lacks. There is no way to track craters. We implemented a naïve, but preliminary, tagging algorithm that stores images and calculates mean squared error to check if the crater has already been tagged. However, this will become computationally expensive if run for a long time. It also lacking in robustness in that it cannot identify the same crater in different lighting conditions and angles.

Sophisticated methods that do exist for this purpose are computationally expensive and make the program unsuitable for our desired application. Further research needs to be done in this area before we can successfully use our program in actual simulations.

Potential future research questions could explore more closely how feature tracking can help with identification and tagging and how to handle different lighting conditions and angles for more precise navigation. Finally, another topic that is more applicable would be implementing navigations filters that use these results as inputs.

References

[1] Simonyan, Zisserman (2014) "*Very Deep Convolutional Neural Networks for Large-Scale Image Recognition"*

[2] Szegedy, Liu et. al (2015) "*GoogLeNet"*

[3] Girshick, Donahue, et. al. (2014) "*Rich feature hierarchies for accurate object detection and semantic segmentation*"

[4] Urbach, Stepinski (2009) "*Automatic detection of sub-km craters in high resolution planetary images"*

[5] Kalal, Mikolajczyk, Matas (2010) "*Tracking-Learning-Detection"*

[6] Dor, Tsiotras "*Application of ORB-SLAM to Spacecraft Non-Cooperative Rendezvous*"

[7] Mur-Artal, Tardos (2016) "*ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras*"

[8] Ross Girshick (2015) "*Fast R-CNN*"

Datasets

[9] NASA Detecting Crater Impact Challenge
https://www.nasa.gov/feature/detecting-crater-impact-challenge

[10] COCO Dataset
http://cocodataset.org

Tools

[11] TensorFlow Object Detection API
https://github.com/tensorflow/models/tree/master/research/object_detection

[12] OpenCV
https://opencv.org