UTILIZING DRONES AND MACHINE LEARNING FOR METEORITE SEARCHING AND RECOVERY.

S. L. Anderson[1], P. A. Bland[1], M. C. Towner[1], J. P. Paxman[1], [1]Curtin University-Space Science and Technology Center (GPO Box U1987, Perth, Western Australia, 6102. seamus.anderson@postgrad.curtin.edu.au, p.bland@curtin.edu.au, martin.towner@curtin.edu.au, j.paxman@curtin.edu.au).

Abstract: Camera networks such as the Desert Fireball Network (DFN) are designed to observe fireballs for meteorite recovery with associated orbits. Most parts of the data pipeline can be partially or fully automated, allowing a network to be maintained by a relatively small staff. However, the outstanding issue that requires large staff effort is the searching and meteorite fall recovery. We train convolutional neural networks to detect meteorites in drone-obtained images of the Australian outback.

Introduction: Fireball camera networks, such as the Desert Fireball Network in Australia, provide a unique insight into the meteoroid population and orbits of objects that both burn up and those that fall to Earth [1]. Recovery of the resultant meteorites help to further characterize the geochemistry and composition of their source regions, providing a valuable spatial context that is a useful complement to asteroid sample return. The limiting factor in meteorite-fall discovery for a digital facility like the DFN is not observing meteorite entry, but instead finding them once they have landed. Due to the limitations of the observational hardware, the predicted searching area for a meteorite is typically 3-4 km^2 but can be as high as 10 km^2 or more. Expeditions to search an area in the Australian outback are costly, in both money and human time, and in many cases, multiple expeditions are needed to cover the full search areas. Humans are also fallible, becoming fatigued during the day and through an expedition.

We propose replacing humans with drones and machine learning to improve the framework of recovering fallen meteorites. Convolutional Neural Networks (CNNs), a subset of machine learning, have proven useful for a staggeringly wide range of image recognition tasks [2] and can be easily applied to meteorite searching. Aerial drones are the perfect vehicle to obtain images of the search area, as they can cover an area in a fraction of the time as compared to traditional human-powered searches and maintain consistent image quality and therefore algorithm detection.

Previous attempts to detect meteorites in this way have some promising results [3]. We build on the work of Citron et al., improving the drone and imaging hardware we used in our trial run, while also making alterations to their CNN architecture, and taking a new approach to the training data.

Methods: To construct, train and use our neural network, we used the Keras module with TensorFlow backend [4], in Python 3.

Instead of using our model to predict on an entire 42 Mega-pixel image, approximately 100 m^2 in area, we chose to train and predict on 200 x 200 pixel-tiles, to allow us to highlight areas of high probability in each image. When we predicted on an image, we also implemented a 125 pixel stride, which allowed us to avoid splitting a meteorite onto two tiles, instead only training the model to recognize full meteorites in a tile. Otherwise, possible shadows or dark regions appearing at the edge of a tile could trigger false positives, or worse, result in a false negative when a meteorite is cut off or split up.

To train our network, we needed thousands of tiles viewing the ground, both with and without meteorites. Unfortunately, this unique data set did not exist in the numbers we required, ideally many thousands of images. So, we created a synthetic data set, by taking top-down images (with 1.75 mm/pixel resolution) of the of the ground via drone, near Kenwick Wetlands, WA, and split these images into 200 x 200 pixel-tiles. We also assembled an image library of 100 meteorites with intact fusion crusts to be pasted onto the tiles. We randomly chose half of the tiles for alteration whereby, we selected a meteorite, rotated it, resized it to between 15 and 75 pixels (2.5 to 13.1 cm), then pasted it over a tile such that the meteorite was fully in the frame, all with randomly generated parameters.

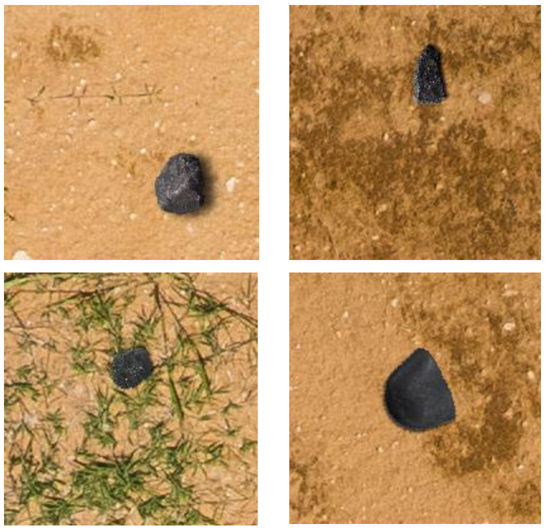
Figure 1.
Examples of our synthetic training set.

We created 5000 tiles with meteorites inserted into the frame, and 5000 unaltered tiles, totaling 10,000 for training and validation. After we had obtained images to create this training data, we also placed black-spray painted rocks (meteor-wrongs) on the ground and photographed them at the same height and resolution as the training images. After we trained the model, we predicted on these meteor-wrong images as a last check of our model fit, this was as close to detecting real meteorites as we could achieve.

Results: Our model attained a 97% training accuracy and a 95% validation accuracy. In our test to detect meteor-wrongs, we set our prediction threshold to 0.99, meaning any tile with a 99% chance or more of being a meteorite was labelled as such. With this threshold set, the model correctly found 57 out of 75 meteor-wrongs and incorrectly labelled 56 false positives. In this context, false positives are acceptable, although not ideal, as this merely results in more follow up searching. Of more concern is false negatives, whereby a genuine meteorite is not detected. Analysis of a 5168 x 3448 image took an average of 8.5 seconds on a 2.5 GHz CPU. At this rate, a 20 minute flight resulting in 500 images would take 70 minutes to process on one laptop.

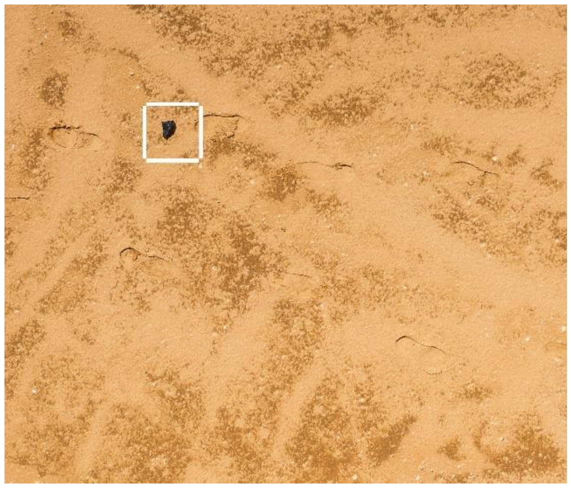
Figure 2.
Prediction over a meteor-wrong image.

We have investigated similar, better fitting models to the parameter set used here, with smaller kernels and strides, at the cost of longer prediction times, averaging 35 seconds per image. Two ways to counter the longer processing time would be to split the computing task between multiple machines or, upgrade our computation hardware to use one or more GPUs instead of a single CPU. We will explore this balance of model performance in future works.

Conclusion: Our methodology of training a model to detect meteorites using a fully synthetic training set will allow us in the future to quickly adapt our model to a new searching area, without having to physically place meteorites (or wrongs) in the field.

Before we attempt to use this method to search for fallen meteorites observed by our network, we still need to validate our detection with real meteorites, and test in more challenging situations, such as when the target is under foliage or in more diverse backgrounds.

References:

[1] Bland P. A. et al. (2012) *Australian Journal of Earth Sci., 59*, 177-187.
[2] Krizhevsky A. et al. (2012) *Advances in Neural Information Processing Systems*, Abstract #4824.
[3] Citron R. I. et al. (2017) *Lunar & Planetary Sci. Conf. 48*, Abstract #2528.
[4] Abadi et al. (2015) www.tensorflow.org