### Project 2: Sunset Detector

Graham Fuller

Jacob Oakes

Rose-Hulman Institute of Technology  
Email: [fullerga@rose-hulman.edu](mailto:fullerga@rose-hulman.edu)

[oakesja@rose-hulman.edu](mailto:oakesja@rose-hulman.edu)

#### Abstract

In this study we created a sunset detection application to classify images as sunsets or non-sunsets. Our process of accomplishing this included extracting 294 features from each image in a set of test images. For each practice image we used the classification to make a model for testing. With a set of new images, we ran them through our model and from it got a true positive rate above a 90% and a false negative rate under 10%. Then we took to neural nets to also model the test data and got similar results.

**1. Introduction**

Image recognition can have a big influence on the way we process images. With just some simple knowledge from an image we can determine if it is a sunset or not and from that process the image accordingly. For example, if we know that an image contains a sunset, we can detect the sky area and brighten the other parts of the image to bring out important details someone may not normally see in the original.

Detecting a sunset is a challenge because images of the sun setting are not clear and distinguished. The components to a sunset are a lot more complicated than, say an orange. There are clouds, birds, water and many other elements you need to take into account.

The proposed solution is interesting because it learns from only test sunset images how to classify almost all sunset images. No other base information about the images was needed to create a model that classifies so well.

**2. Baseline**

To overcome this problem, we first used an SVM to model the system for classifying images. The baseline system included a threshold of 0 and did not have normalized data.

2.1 Initial changes

The first change we made was to increase the threshold. After tampering with it we found a value that best classified the images. In addition, we also changed the sigma value passed to the kernel function.

2.2 Expected results

With a more accurate threshold, our accuracy should increase. In addition, we should find a good sigma value that will better separate the classifications.

**3. Feature Extraction**

To extract features from each image, we first broke up the image into a 7x7 grid so that there were 49 sections. PUT PIC OF GRID OVER IMAGE HERE. In order to make each grid cell equal in size, extra pixels in each dimension were cut off on the edge if needed. At most the image would lose 6 rows of pixels on the ends. This part of the image can be considered negligible because it is so small. Next, we calculated the LST values from RGB values using the following formulas:

L = R + G + B

S = R – B

T = R – 2G + B

To normalize the data we #################

We used the normalized data to create the features for each cell, creating 294 features for each image.

**1. Conclusion**

A couple challenges we ran into were selecting neuro net values for hidden nodes and trials to average the final weight of the nodes. With a higher value for these the more accurate results we got, but the more time it took. To narrow down our search for the correct values, we had to rerun the same program numerous times, sometimes taking up to hours to run.