Solar System Planet Orbitals Dataset From Stellarium

Arjun Chandran *
arjun_chandran@berkeley.edu

Adam Chang
adamchang2000@berkeley.edu

Zijin Cui
zijincui281@berkeley.edu

Jacob Yeung
jacobyeung@berkeley.edu

Abstract

In order to train a strictly computer-vision based model to learn the orbits of astronomical bodies in our Solar System, a clean dataset is required with raw images, Solar System object labels, and timestamps, preferably at regularly spaced intervals. In this paper, we generate this dataset using simulated images extracted from Stellarium, an open-source astronomical simulator. We also compiled a few scripts to readily generate various fourier and convolutional kernel flavors of the data for different types of models. By using a simulator we were able to solve several logistical issues allowing us to generate a clean dataset with ease. We provide an example model which is capable of using standard convolutional neural network techniques to learn patterns in the movements of Jupiter as proof that our dataset is easily use-able.

1 Introduction

1.1 Motivation

The goal of our dataset is to provide people a base to perform image based sequence prediction on images of our solar system. To provide our data users with a plethora of options, we have developed two datasets comprising of featurized image data and masked image data. Additionally, we have provided and tested extra functionalities for generating various flavors of our data including scripts for generating Fourier featurized data and convolutional kernel expansions of our data.

2 Method

Our dataset generation process can be broken down into 3 distinct sections. They are broken down as follows.

2.1 Stellarium Data Collection

First, the scenarios must be simulated in Stellarium so that the data can be extracted in the form of screenshots. This was performed using Stellarium's built-in scripting API, with code written in ECMAScript. The responsible file in the provided repository is $collect\_data.ssc$. The scripts can be broken down into the following parts:

1. Initialize proper settings for Stellarium.

*All authors contributed equally.

2. Record metadata for the current images, such as location on Earth and FOV.
3. Begin iterations:
    - Increment time by 1 minute.
    - Perform an image capture.

Specific choices of settings were motivated by the purpose of the dataset, to predict the movement of the planets, the Sun, and the Moon.

- Atmosphere disabled: This allows the image to contain images with both the Sun and stars at the same time. With the atmosphere enabled, the sky would become bright blue, as the apparent magnitude of the Sun is much greater than any other solar system object. Without the atmosphere, we were able to collect a continuous set of images over a 25000 second interval, still collecting data about the positions of the planets with the Sun in view.
- Stars disabled: This is to emulate the night sky in a light-polluted environment, such as within a suburban or urban area. Only the brightest objects in the night sky are visible, such as the Moon and most of the planets closer to the Earth

The resulting output of this section of the code is a folder containing a metadata text file and a sequence of images at Stellarium's original resolution. The metadata text file contains:

- The position we are on Earth in longitude and latitude.
- The FOV of the screenshot.
- The positions of all Solar System objects in pixel coordinates per image.

2.2 Generating Dataset using Python

We then post process the images from the raw screenshots outputted by the script using a Python script, $process.py$. The process script iterates through every image and reads the metadata corresponding to each image. It then performs the following steps:

1. Resize the image to a desired resolution. This is necessary to output images of a size capable of being used by a neural network.
2. Perform flood fills to extract bit masks from the images, which form the labels for our dataset.

The resulting output of this section of the code is a folder containing a metadata text file and 2 separate sequences of images. The metadata text file contains:

- The position we are on Earth in longitude and latitude.
- The FOV of the screenshot.
- The original image resolution.

There are 2 separate sequences of images outputted from $process.py$.

- $object\_bit\_mask$: An overall bit mask, with 0 corresponding to a pixel with no object and 1 corresponding to a pixel with an object.
- $object\_indexed\_mask$: Contains folders with the naming <Image Number>_images. Inside these folders are individual bit masks corresponding to the location of specific Solar System objects. For example, Sun.png would contain a bit mask of the pixels corresponding to the Sun in the image. Respectively, Jupiter.png would contain a bit mask of the pixels corresponding to Jupiter in the image.

2.3 Generating Augmented/Featurized Data

After creating the resized images and bit masks in the previous segment of our dataset construction, we include additional scripts to augment and featurize the data. There are two Python scripts with different functions for this purpose.

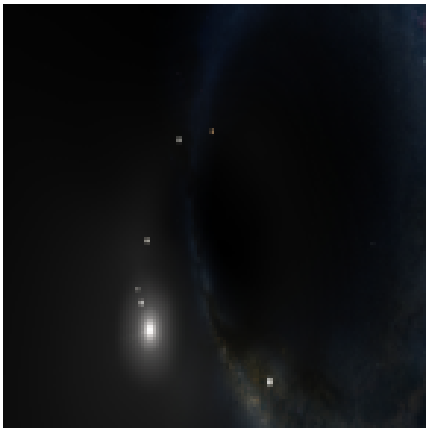

Figure 1: Downsized Square Image

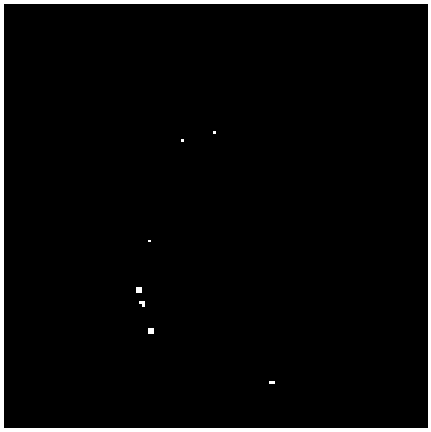

Figure 2: Bit-masked Flood-filled Image

1. $helper.py$:
   Function: create_data_matrix_expanded
   Reads in all bit masks for a specific Planetary body.
   Expands each 1 bit to its 5x5 surrounding region. (Sets all bits in the mask to 1 if they are within a 5x5 region around a 1 bit in the original bit mask).
   Purpose is to increase the granularity of the dataset to better support predictive neural networks.
   Implemented using scipy's convolve function, using a 5x5 matrix of 1's as the convolution kernel, and then casting to boolean. All non-zero values in the output are casted to true and all zero values are casted to false. In this fashion, any center location for a kernel is set to true in the resulting output when a bit is set to true in the input within a 5x5 area.
2. $main.py$:
   Function: Provides a frequency domain set of data points for a model to possibly train on. Since the orbit of planets is periodic by nature; its path across the sky can be represented via a frequency and since Fourier features are good at approximating frequencies this may aid a model in modeling these types of phenomenons. The implementation works by preforming a two dimensional fast Fourier transform across the image data. The intuition behind this is that as time goes on and planets move across the night sky they lead to repetitive rises and falls in pixel intensity values. By converting these intensity values into a frequency domain representation a model could potentially learn the frequencies of these rises and falls and therefore accurately predict how the planets move across the night sky over time. Due to numpy's encoding of complex numbers the files generated for this kind of dataset are pretty big (approx. 8GB per file). For this reason we have not provided this dataset for download however this dataset can be readily generated very quickly by running main.py. See the ReadMe for more details on this process.

3 Results

Figures 1 and 2 (above) show the down-scaled and bit-masked image, and Figures 3 to 5 (below) show the separate bit mask image for Jupiter, Mars, and the Moon.

Using the techniques described in the Methods section of this paper, we were able to create a high quality, noise-free image dataset using Stellarium. We were able to simulate the motion of the planets, Sun, and Moon through the night sky, and generate proper labels via Stellarium's scripting API.

While constructing this dataset, we had to make several considerations that hint towards the difficulty of the problem of predicting Solar System object traversals via just images.

Firstly, we hypothesize that a neural network would be better suited for learning the motion of these objects if the images were taken at an extremely large FOV (230 deg) and in a continuous manner

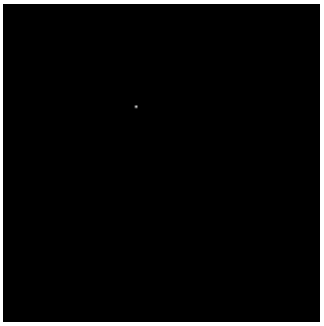

Figure 3: Jupiter

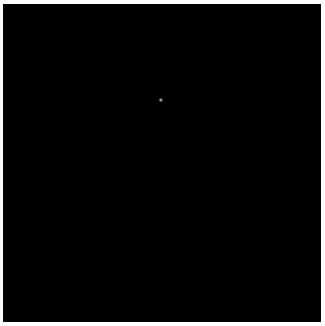

Figure 4: Mars

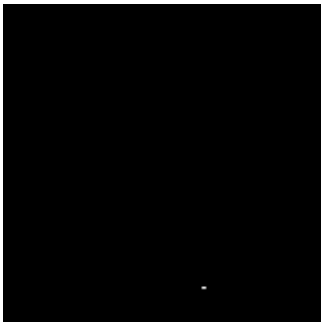

Figure 5: Moon

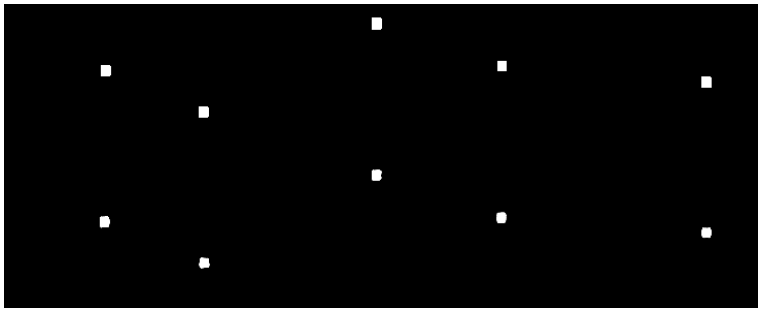

Figure 6: Sun Reconstruction

(each image regularly spaced and without movement on the Earth's surface). The reasoning behind this is the disjoint nature of the planets' motion. If allowed to leave the frame they then re-enter the frame from a different location at a very different time. However, it may be a desired feature for the network to learn to jump across the screen for predicting images at next time steps at the edge of the image.

Secondly, we found that training a model on images with a single pixel output required careful consideration of the loss function used to penalize the model. Simply using binary cross-entropy loss, which measures the correctness of the reconstruction, incentivized the model to simply return a black image, as the loss incurred would not be very high.

3.1 Example Usage of Image Data

In order to ensure the dataset's quality and practicality, we implemented a β-variational autoencoder (VAE) to reconstruct images. However, we faced the issue of the network returning a black image. Consequently, we increased the size of the objects with the function in Section 2.3.1. In Figure 6 (above) we see a reconstruction of various samples of the sun in different locations.

The β-VAE consists of an encoder and decoder network combined with a latent distribution. To train this network, we constrained the Kullback–Leibler (KL) divergence to a set of values. We linearly increased the constraints from 0 nats to 25 nats. We tuned a hyperparameter of β - the weight of the KL divergence compared to the reconstruction loss which forms the overall loss.

We then wondered if we could use our β-VAE to impute positions of the solar objects as they travel across the sky. To obtain such images, we decided to test if the network's latent space learned the movement of the objects, as a latent traversal to impute positions is spiritually reminiscent of predicting planet positions. Therefore, we traversed the latent distribution by inputting a static range of values for the mean while keeping the variance the same. We see in Figure 7 the traversal of the latent space. We observe that the images show some variation. However, our simple network was not able to smoothly show a traversal of the latent space in that we could not accurately nor precisely impute trajectories. We speculate this may be due to several issues - β hyperparameter tuning and model complexity.

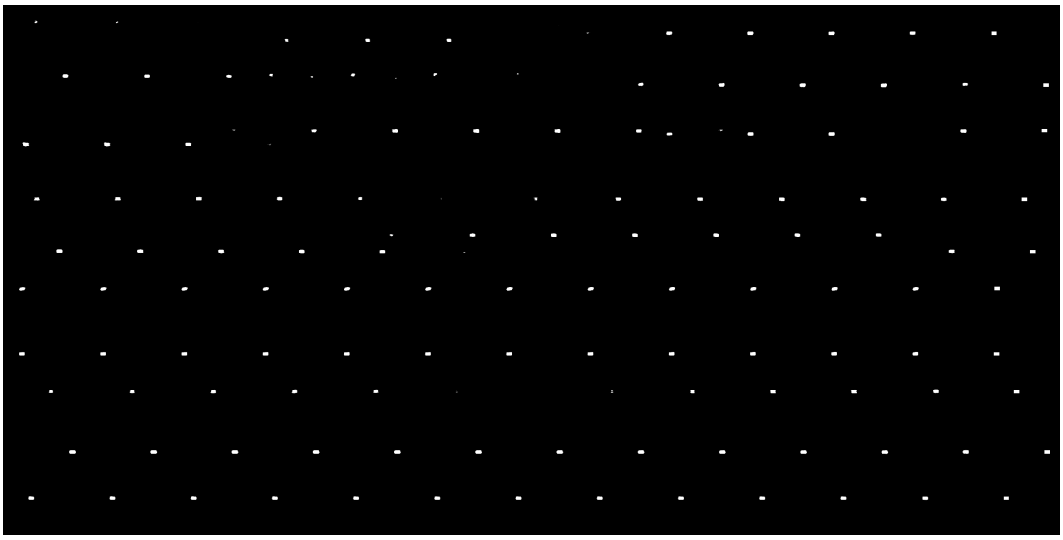

Figure 7: Sun Latent Space Traversal | The images on the leftmost column are the input images. The column one to its right are the reconstructions. The rest of the images are a traversal of the latent distribution.

4 Conclusion

Through this process we were able to learn about data generation and the creation of multiple datasets that can be utilized for a multitude of purposes, including time series image to image predictions of planetary positions. Additionally, we were able to validate the utility of the data through our reconstructions with the VAE. This allows us to confidently provide a useful dataset for anyone attempting to generate machine learning models, at the most basic level, to reconstruct simple objects.

4.1 Lessons Learned

We centralized the core data in our Github repository. We attempted using Github and Google Drive to store images generated from the core data, but we quickly realized uploading dozens of gigabytes of image data was infeasible. In addition, downloading such large datasets is bulky. However, we realized we were storing redudant information - data two degrees of separation from Stellarium. Therefore, we decided to store the dense data and provide scripts for users to generate additional data. Fundamentally, our data storing technique enables portable, quick, and informative usage.

4.2 Notes for those who use this dataset

This dataset provides a clean set of noiseless images for use of predicting the moment of the objects in our Solar System. However, certain design choices were made that need to be considered when moving the dataset to real data. For one, we found greater success in predicting the motion of one planetary object as opposed to the motion of all objects at the same time. For real images, some work might need to be done to isolate the single object that one might want to predict the motion of. This hand labeling is made more necessary by the lack of stars in this simulated data. This dataset makes no strides towards being able to distinguish planets from stars in real world data.

References

[1] Project Repository: https://github.com/jacobyeung/CS-189-Project

[2] Tancik, M. & et al. (2020) *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains.*

[3] Stellarium: https://stellarium.org/