Tiny Planet : Morphing Images & Videos into Planet Shapes

Liz Zhang
Carnegie Mellon University, John Heinz III College, School of Information Systems
5000 Forbes Ave, Pittsburgh, PA 15213
yizhizha@andrew.cmu.edu

Figure 1. Source image.

Figure 2. Target image ($size = 0.4, bulge = 0.15, blending = 0.1$).

Abstract

The purpose of this project is to morph ordinary images and videos into planet-like shapes. The basic technique includes the transformation between cartesian coordinate system and polar coordinate system. The project then applied a series of special effects on the result like functions that controls size, bulging and blending. For videos, this project added endless loop effect.

1. Introduction

The idea of this project comes from the app Living Planet by Samer Azzam[1], which converts images into stereographic projections to produce planet like shapes. This project achieves similar effects by mapping the polar coordinates to its cartesian counterparts. It also implement the size, bulging and blending effects in the app. What the app doesn't have though, is the endless loop effect. By taking a video with a moving object in the foreground going across the screen, techniques in this project can be used to produce an authentic looping video that allows the foreground object to move around the "globe" endlessly.

2. "Globalize" the Image

2.1. Basic morphing

The basic technique involves converting the cartesian coordinates system into polar coordinates, then do the inverse morphing to get the pixels in the original image. Given the cartesian coordinates (x, y), the polar coordinates (θ, rho) can be calculated by

$$\theta = atan(y, x)\ , rho = \sqrt{(x.^2 + y.^2)}$$

If an image is taken in landscape (with sky on top and ground on the bottom). The ground is closer to the center of the globe and the sky on the outer peripherals. Therefore, the rho can be calculated by getting the target pixel's relative position to the bottom. Similarly θ can be inferred by its relative position to the left/right end of the image.

Without any extra special effects, a simple polar morph result is shown as figure 3.

Figure 3. Simple polar morph without special effects

2.2. Size, bulge and blending

The default setting using above technique will give us linear matching between the pixels in the polar system and the original photo. To be specific,

$$rho/r = (height - y)/height$$

To make a more authentic globe, this function can be changed to reflect the unilinear matching between polar system and cartesian.

The size function that this project settled with is

$$rho/r = (height - y)/height * (rho/r)^{size\_factor}$$

With the size factor we can control how much space we want the ground to take compared to the sky, or, in the result image, how big the planet is, but there is a trade off between ground and sky, and with only the size function we can not control both. Thus, I added one more function called the bulging function so that we can change both the ground proportion and the sky proportion.

The bulging function is essentially two size functions with a threshold to decide which function to go with. Above I have already given the size function for the ground, below is the size function for the sky:

$$rho/r = -(height - y)/height * (-rho/r)^{size\_factor}$$

This is the inverse function of the ground size function. Once we have these two, we can set a threshold(called bulge_factor) between 0 and 1, so that the complete bulging function would look like:

$$C = 1/(bulge\_factor^{size\_factor} + (1 - bulge\_factor)^{size\_factor})$$

Figure 4. Polar morph with $size = 0.5, blending = 0.09, bulge = 0.2$

$$\text{if } \frac{rho}{r} > bulge\_factor:$$
$$bulge = C * (\frac{rho}{r} - bulge\_factor)^{size\_factor} + C * bulge\_factor^{size\_factor}$$
$$\text{if } \frac{rho}{r} <= bulge\_factor:$$
$$bulge = -C * (-(\frac{rho}{r} - bulge\_factor))^{size\_factor} + C * bulge\_factor^{size\_factor}$$

At last, to make the stitching part between the left and right end of the image less obtruding, in this project I used cross dissolving to blend the two parts. A polarized image with all three effects added is shown in figure 4.

3. Creating Endless Loop

The endless loop part is the most challenging part of this project. It is roughly consisted of these steps: Detecting trajectory of foreground and rectify the image based on the rotation angle of the trajectory; Detect the entering point and exit point frame of the foreground moving object in the video and match them; Blend the entering and exit frames; Find a pair of same frames in the stitched video. Details are explained below.

3.1. Corner detection for getting position of foreground object

For endless loop, first the video needs to be rotated so that the trajectory of the foreground moving object is parallel to the planet center plane. The way to do this is to detect corners of the foreground object, finding matching corners and calculate the trajectory from these points(Figure 6&7).

Figure 5. The image difference used to detect corners instead of original image.

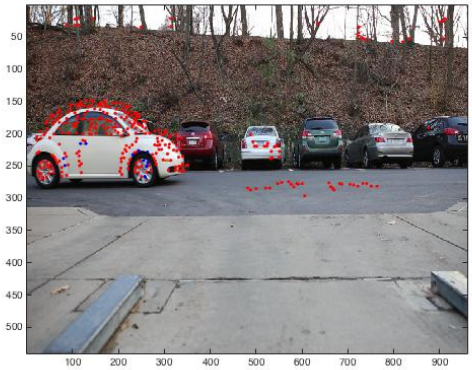

Figure 6. Corners(matching corners in blue) detected in first image

Figure 7. Corners(matching corners in blue) detected in second image

For corner detection, I referred to methods described in the paper by Multi-Image Matching using Multi-Scale Oriented Patches[2] , but instead of looking for corners that are scattered spatially distributed, only corners of the moving object in foreground is detected and used. First use harris corners to detect corners in the image. In order to get the corners of the moving car (the foreground) instead of getting corners all over the place, first get the difference between the two frames(Figure 5), then use the absolute difference as weight to the corner strength.

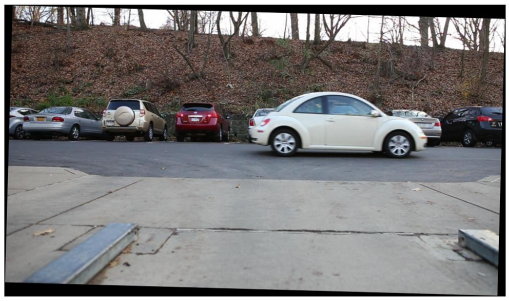

Figure 8. rotated image. (trajectory rotation degree: 0.011)

3.2. Rotate image based on trajectory angle

After getting the matching corners, the angle between the foreground object trajectory and the horizontal line is easily calculated from the the mean position of these matching points. (Figure 8)

3.3. Slit scan to choose key frame

For getting the frame in the video where the entering of the foreground object could match up with the leaving of it, I used a technique similar to slit scan. Given the blending factor, it can be calculated which column of the image will be the exiting point for the moving object. Therefore, I slit scan the first column and the exiting column and get the sum of that column of pixels values. One example of slit scanning the exiting column of above example is shown in Figure 9.

3.4. Choose start and end frame by image difference

After stitching the two identical videos by matching their entering frame, I used cross dissolving to blend the stitching frames. Then I choose a random frame from the first half of the video, and get its difference with the frames from the second half. Theoretically there should be a frame whose difference with the frame chosen is zero since it is the composite of two identical videos with only time difference. Once the identical frames are found, the endless loop gif has one full loop.

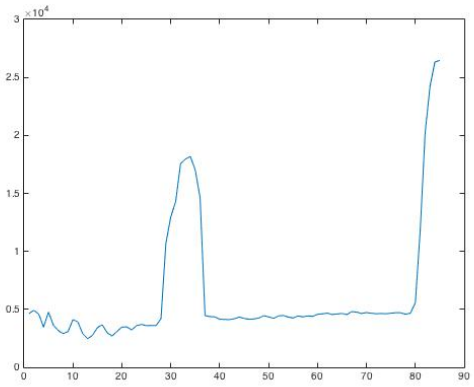

Figure 9. Column sum by slit scanning video at 0.9*image width

Figure 10. Endless loop, $size = 0.4, blending = 0.09, bulge = 0.1$

4. Example Results

4.1. Simple polar morphing of still image

4.2. Video

4.3. Endless Loop

References

[1] S. Azzam. Living planet - tiny planet videos and photos. https://itunes.apple.com/us/app/living-planet-tiny-planet/id858575030?mt=8.

[2] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, page 510517. IEEE, 2005.

Figure 11. Endless loop, $size = 0.4, blending = 0.1, bulge = 0.1$

Figure 12. Endless loop, $size = 0.4, blending = 0.09, bulge = 0.15$

Figure 13. Endless loop, $size = 0.5, blending = 0.09, bulge = 0.25$

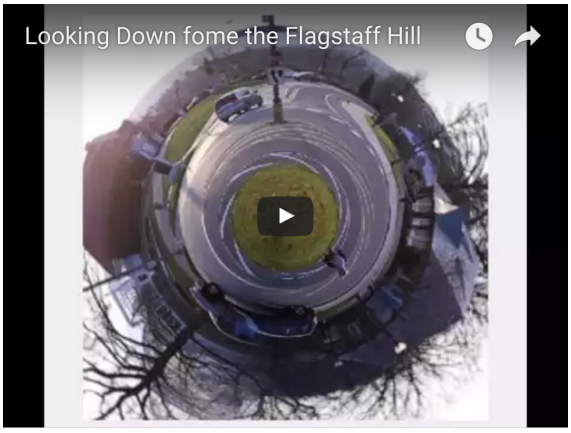

Figure 14. Endless loop, $size = 0.4, blending = 0.1, bulge = 0.15$
https://youtu.be/0-Z6iii1hi4

Figure 15. Endless loop, $size = 0.4, blending = 0.1, bulge = 0.15$
https://cloud.githubusercontent.com/assets/11666005/11861689/8f04a54a-a450-11e5-866f-a399434a4622.gif

Figure 16. Endless loop, $size = 0.5, blending = 0.09, bulge = 0.1$
https://cloud.githubusercontent.com/assets/11666005/11861688/8f04066c-a450-11e5-9b73-6026c1785deb.gif