Using Image Processing and Monte Carlo Simulation to Analyze Earth's Antipodes Final Project

Inst. Pavel Snopok, Spring 2015

Michael Lee

July 23, 2015

Abstract

Antipodes are two points on a map that are diametrically opposed and are connected by an imaginary line drawn through the sphere's center. They are an interesting geometric construction that have importance in cartography and geography. A useful detail about antipodes is that one place at noon would make the other exactly midnight. Also, with the exceptions of the areas between the Tropic of Cancer and Tropic of Capricorn, the longest day of the year would correspond to the shortest day on the other side. We report here the probability of antipodes on Earth being land-land, water-water, and land-water which gives us an estimate for the surface area of each type of antipode. Monte Carlo and image processing methods were used to obtain a solution for the variety of cases. We find that the probability of water-water antipodes consistently yield p values of 0.38, land-water antipodes 0.56, and land-land antipodes 0.058.

1 Introduction

The origins of antipodes goes all the way back to the Greeks. They knew that the Earth was round and often thought of what existed underneath them on the other side of the planet. With satellites and global mapping solutions, humans today no longer have to guess but from this, lost some ability to ask questions and run thought experiments about geography. The purpose here is to use today's data to provide an answer to some of the questions the Greeks may have had then.

In this simulation, we explore a deceptively simple geometric problem in geography: Suppose you have the planet Earth. If you were to draw a line straight through its center and cross both surfaces, what is the probability that the line both intersect with water?

This problem cannot be solved analytically because the very nature of it is non-linear. This means that we cannot produce a general formula that will output the probability for any given spherical object with definite coastlines. This problem is also hard analytically in that antipodal points are pairwise and the probabilities will vary greatly depending on where the land and water surfaces are. We will then have to employ numerical methods to receive an approximate value. We solve this problem in two ways: first through the Monte Carlo method and second by using image processing techniques.

2 Technique and Strategy

In our simulation, we have a projection of the planet Earth that uses the World Geodesic System (WGS84) for our geospatial information. We use this projection because it preserves latitude and longitude information on a x-y plane that corresponds to the pixel position. The image used is a 1280x800 bitmap of the planet Earth converted from a 16Kx81K PNG, filtered and monochromed, making black (1) referencing land and white (0) referencing water. Doing this makes our image processing highly efficient as we can reference each pixel as either 1 or 0. We do we lose some resolution but it is still fine enough to obtain a reasonable estimate of the antipodal probabilities.

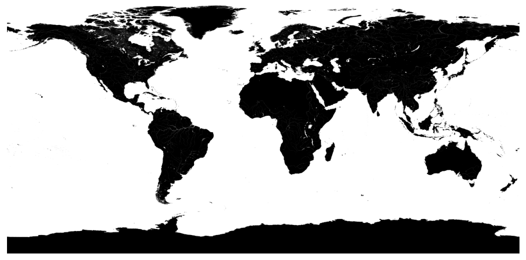


Figure 1: Monochrome World Geodesic System where land is black (bitwise 1) and water is white (bitwise 0).

The total pixel count for our image is 1,024,000. Each pixel is 0.28125° in longitude, or east-west, and 0.225° in latitude, or north-south. This translate to one pixel being approximately 496 km^2 or having a side length of about 22.3 km. There is a slight degree of error taking into account warping but the benefit of using the WGS84 projection is that it tries to scale longitude and latitude (θ, ϕ) linearly with pixel position (x, y). The antipodal points described by this mapping can then be expressed as:

$$(\theta,\ \phi) = (0.28125 * x - 180,\ 90 - 0.225 * y)$$

where positive refers to north and east, and negative, south and west.

The antipodes of (θ, ϕ) would be the points 180° in longitude and 180° in latitude across the sphere. So then the transformation becomes:

$$(\theta,\ \phi)_{antipode} = (0.28125 * x,\ 0.225 * y - 90)$$

For instance, the pixel (0,0) would refer to (-180° (W), 90° (N)), the pixel (640, 0) refers to (0°, 90° (N)), and the pixel (640, 400) maps to (0°, 0°). If we write the antipodal transformations in purely x and y, we receive the relationship:

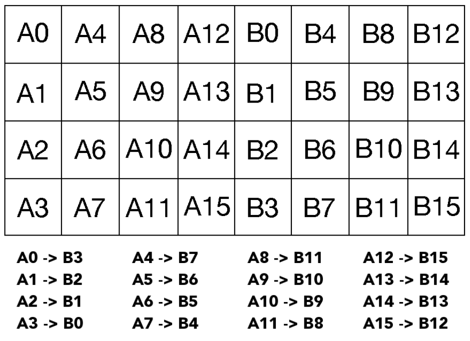
$$(x, y) \rightarrow (x, y)_{antipode} = (640{+}x, 800{-}y)\ for\ x\ <\ 640$$

$$(x, y) \rightarrow (x, y)_{antipode} = (x{-}640, 800{-}y)\ for\ x\ >\ 640$$

where we have a flip about the x axis for (x) because an antipode on the Eastern hemisphere corresponds to one in the Western hemisphere.

To solve the problem of land-land, land-water, and water-water antipodes, we use Monte Carlo simulations and image processing techniques. The implementation of the Monte Carlo method is simple. Since we have the coordinates of the antipodes, all we will do is run a pixel check of the antipodes over N samples and if they are match and are both water, then the program will count that sample. After the end of the pixel checking loop, we will divide the water count over N and receive the probability of water-water antipodes. The same can be implemented for land-water and land-land antipodes by changing how the program counts the pixels.

For image processing, we divide the map into 16 areas.



A0	A4	A8	A12	B0	B4	B8	B12
A1	A5	A9	A13	B1	B5	B9	B13
A2	A6	A10	A14	B2	B6	B10	B14
A3	A7	A11	A15	B3	B7	B11	B15

A0 -> B3	A4 -> B7	A8 -> B11	A12 -> B15
A1 -> B2	A5 -> B6	A9 -> B10	A13 -> B14
A2 -> B1	A6 -> B5	A10 -> B9	A14 -> B13
A3 -> B0	A7 -> B4	A11 -> B8	A15 -> B12

Figure 2: We split the image into 16 areas to show the antipodal area pairs.

The areas will then have their corresponding antipodal areas. After we cut the image into 16 separate parts, we transform the A parts by flipping it vertically within itself (figure 3). This is so that we can map the pixels (i, j) to each antipodal areas and compare them against one another. We then run a pixel comparison and collect the counts from each image pair. This method is efficient in that it can be parallelized and divided into more areas.

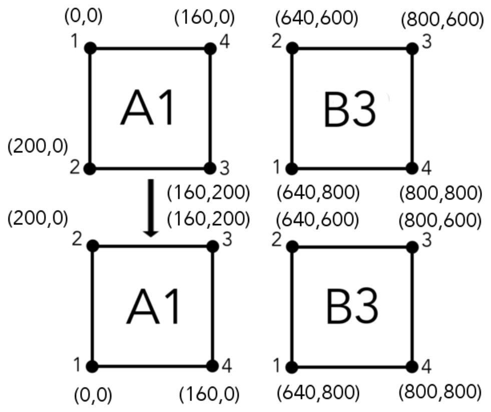


Figure 3: We make a transformation, flipping A1 in x so that we can compare pixels (i, j).

Another method we can use is mapping each pixel from one area to another and creating an array containing the sum. We can then count the number of times 0, 1, and 2 appears in the array and divide it by the length of the array to receive the probability of each possible outcome.

3 Analysis

There is a slight error with the image since through an area check, we found that it has a 33.7% to 66.1% land-water ratio. The actual figure is more like 29% to 71% but this ratio excludes small bodies of water like lakes and rivers, which with our pixel resolution of 496 km^2, will not show up.

From our simulation, the probability of hitting water on both sides is approximately 38%. The probability of hitting land on both sides is 5.8%, making it about 6.5 times less likely than water-water. To our surprise, the probability of hitting water from either side on land or water is about 94%, which is significant. This means that although 33.7% of the map is land, only 17% of that land area has a land-land antipode.

Interestingly, the probability of hitting water starting from land from the Eastern hemisphere is approximately equal to the chance of hitting water starting from land from the Western hemisphere at 28%. Through several Monte Carlo simulations, the average value of the Western hemisphere is slightly larger only by 0.1%.

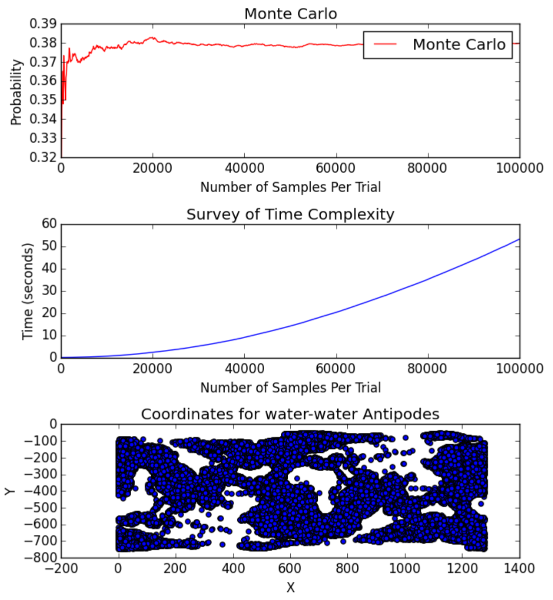


Figure 4: The results of Monte Carlo Simulation for water-water antipodes. The probability converges to 0.38. Time complexity is linear with just finding the probability without finding the coordinates and the scatter plot. Note that the scatter plot is the pixel coordinates, which has (0,0) at the top left corner, meaning that this is flipped vertically in comparison to the WGS84 map so Y has to be negative.

There are errors in our simulation in that pixels are flat while the Earth's surface is curved. The Earth is also not a perfect sphere and the antipodal points we use may not be exact. However, there is no obvious place to obtain this data about the probabilities and it may have some relevancy to communications as to the best places to install satellites for optimal coverage. High energy physics and astrophysics may also benefit since they can find the greatest possible diameter, aside from going to space, to run their experiments.

From our scatter plot, we see that the locations of the land-land antipodes are largely concentrated in Northern Canada, Central Russia, Central China, Eastern South America, and the Eastern and Western parts of Antarctica.

It took the computer 0.256 μs to reference each pixel. If we scale that to the 16K x 8.1K image, the total time to reference all the pixels through one thread would be about 33.2 seconds. The Monte Carlo method, as expected, performed better than

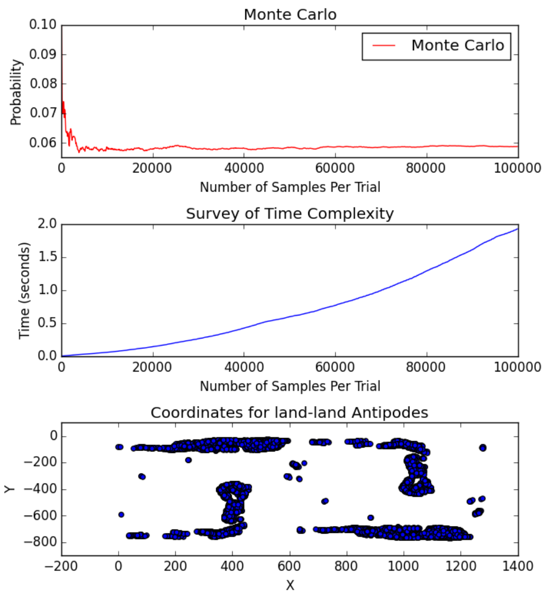


Figure 5: The results of Monte Carlo Simulation for land-land antipodes. The probability converges to .06.

the image processing technique primarily through heuristically performing less comparisons than the latter process. The advantages for image processing, though, is that you can preserve more information about the image and perform operations in parallel. If you received a very high resolution image, it is advisable to split it up and process it by pieces than by using Monte Carlo to compare variables, since at high N, Monte Carlo will get beaten if the program is parallelized enough.

4 Conclusion

Through Monte Carlo simulations and image processing, we found that the probability of water-water antipodes was about 38%, a low number considering that 66% of the map was water. The land-land antipodes probability turned out to be 5.8%, an even lower figure. The most significant figure that we found was that about 94% of the surface of Earth had a water paired antipodes. This means that you probably won't end up in China if you dig straight through the center of the Earth. The probability of land-water antipodes from Eastern and Western hemispheres were about equal at 28%. This may tell us either that the distribution of land is about even across the Eastern or Western hemispheres or that there is some conservation of land mass and that the continents began drifting away from a single land mass. This arguments works well when the land-land antipodal percentage is low.

The Greeks wondered what world was beneath their feet. We know now that the answer was probably water.

References

[1] R. Landau, Manual J. Paez, Cristian C. Bordeianu. *A Survey of Computational Physics*, 2010: Princeton University Press.

[2] D. Beazley and Brian K. Jones. *The Python Cookbook*, 2013: O'Reilly.

[3] Newman, M. E. J. *Computational Physics*, 2013: Createspace.

[4] Matt Williams. *What Percent of Earth Is Water?* Universe Today. N.p., 01 Dec. 2014.

5 Appendix

5.1 Land and Water Surface Area Percentages

```python
from PIL import Image
import random as r
import time

im = Image.open("water_1280.bmp")
pix = im.load()

water = 0.0
land = 0.0
count = 0.0
pixwater = 0.0
pixland = 0.0
pixtotal = 800.*1280.

brute_force_time = time.time()
for i in range(0,1280):
    for j in range(0,800):
        if pix[i,j]:
            pixland += 1
        else:
            pixwater += 1
print "Brute force of land/water surface ratio:"
print "Land percentage: %f " % (pixland/pixtotal)
print "Water percentage: %f " % (pixwater/pixtotal)
print "%s seconds\n" % (time.time() - brute_force_time)

monte_carlo_time = time.time()
while count < 250000:
    x = r.randint(1,1279)
    y = r.randint(1, 799)

    if x <= 639:
        X, Y = x + 640, 800 - y
    elif x >= 640:
        X, Y = x - 640, 800 - y

    if (pix[x,y]):
        land += 1
        count += 1
    else:
        water += 1
        count += 1

print "Monte Carlo Sim. of land/water surface ratio:"
print "Land percentage: %f" % (land/count)
print "Water percentage: %f" % (water/count)
print "%s seconds\n" % (time.time() - monte_carlo_time)
```

5.2 Monte Carlo Method

```python
"""
Written by Michael Lee.

```

```python
PHYS440: Computational Physics
Final Project, Using Image Processing and Monte Carlo Simulations to Analyze Earth's
Antipodes
"""

from PIL import Image
import random as r
import time
import matplotlib.pyplot as plt

begin_time = time.time()
im = Image.open("water_1280.bmp")
pix = im.load()

water = 0.0
count = 0.0
wol = 0
timearray = []
probabilityarray = []
coordinates = []

for n in range(100, 100000, 100):
    while count < n:
        x = r.randint(1,1279)  # Use 1279 because the pix max indices are (1279, 799)
        y = r.randint(1, 799)  # The index starts from 1 as well.

        if x <= 639:
            X, Y = x + 640, 800 - y
        elif x >= 640:
            X, Y = x - 640, 800 - y

        # Makes the time complexity O(n^2) if you're
        # searching for the coordinates as well

        if (pix[x,y] == 1 and pix[X,Y] == 1):
            if ((x,y) or (X,Y)) not in coordinates:
                coordinates.append((x,y))
                coordinates.append((X,Y))

        if (pix[x,y] == 0 and pix[X,Y] == 0):
            water += 1
            count += 1
        else:
            count += 1

    timearray.append((time.time()-begin_time))
    probabilityarray.append(water/count)

    print "The probability of hitting water on both sides is: %f" % (water/count)
    print "%s seconds" % (time.time() - begin_time)

x = range(100,100000,100)
y = probabilityarray
z = timearray
x_scatter = [item[0] for item in coordinates]
y_scatter = [item[1] for item in coordinates]

plt.subplot(311)
plt.plot(x, y, 'r', label="Monte Carlo")
plt.title('Monte Carlo')
plt.xlabel('Number of Samples Per Trial')
plt.ylabel('Probability')
plt.legend()

plt.subplots_adjust(hspace=.5)

plt.subplot(312)
plt.title('Survey of Time Complexity')
plt.plot(x, timearray, 'b', label="time")
plt.xlabel('Number of Samples Per Trial')
plt.ylabel('Time (seconds)')

plt.subplots_adjust(hspace=.5)

plt.subplot(313)
plt.title('Coordinates for Land-Land Antipodes')
plt.scatter(x_scatter, map(lambda x: x*-1, y_scatter))
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

5.3 Image Processing Method

```
from PIL import Image
import os, sys
import pprint
import time
import multiprocessing

def crop_reduce_and_invert_vertically(
    image, start, end, number_of_boxes_in_Y,
    width, height, flip=False
    ):
    pixel_corner_locations = []
    regions = []
    for i in range(start, end):
        for j in range(0, number_of_boxes_in_Y):
            """
            Generally, we want even mapping so we want to keep the box sizes equal.
            Use 160 partitions for maximum effect (8 x 5 boxes).
            """
            box = ((width*i), (height*j), (width*(i+1)), (height*(j+1)))
            if flip:
                image_crop_flipped = image.crop(box).transpose(Image.FLIP_TOP_BOTTOM)
                regions.append(image_crop_flipped)
            else:
                image_crop = image.crop(box)
                regions.append(image_crop)
            pixel_corner_locations.append(box)
    return regions, pixel_corner_locations

def unpack_into_bits(image, width, height):
    bit_sequence = [image[i].getpixel((j,k)) for i in range(len(image))
    for j in range(width) for k in range(height)]
    return bit_sequence

if __name__ == '__main__':

    im = Image.open("water_1280.bmp")
    pix = im.load()
    # The picture needs to be inverted on one half in order to make correct comparisons
    # with the antinode pair.
    inverted_picture = crop_reduce_and_invert_vertically(im, 0, 1, 1, 640, 800, True)[0][0]

    begin_time1 = time.time()
    box_width = 8
    box_height = 10

    # splits into 640 squares on the left hemisphere
    west_hemi, w_locations = crop_reduce_and_invert_vertically(inverted_picture,
        0, 80, 80, box_width, box_height)
    # splits into 640 squares on the right hemisphere
    east_hemi, e_locations = crop_reduce_and_invert_vertically(im, 80, 160, 80,
        box_width, box_height)

    west_unpacked = unpack_into_bits(west_hemi, box_width, box_height)
    east_unpacked = unpack_into_bits(east_hemi, box_width, box_height)
    compared_sequences = map(lambda x,y: x+y, west_unpacked, east_unpacked)

    # count(2) for land-land, count(1) for land-water, count(0) for water-water
    probability_of_water = compared_sequences.count(0)/float(len(compared_sequences))

    print "The probability of hitting water on both sides is %f" % probability_of_water
    print "%s seconds\n" % (time.time() - begin_time1)
```