Monte Carlo Approach To Echolocation

Allison Solano, Luis Mariano Ramírez
Algorithm Analysis - IC3002
Instituto Tecnológico de Costa Rica
Project Repository: github.com/Lumanter/Echo-location

***Abstract*— In this work a Monte Carlo probabilistic algorithm is used in the implementation of a sonar for echolocation of a map of lines, applying basics of path tracing and acoustics. The technique was proven to be a satisfactory tool to achieve an acceptable image of the map, while keeping the algorithm iterations deterministic.**

***Keywords*— Monte Carlo, Path Tracing, echolocation, probability, sonar**

I. Introduction

The aim of this project is to implement a sonar for echolocation. In the simulation there are the sonar and a map that consists of a set of line segments on a black background. The sonar shoots sound waves to the map that bounces in different directions, and, with the information of those who hit back the sonar, the perceived map is revealed with black and white monochromatic pixels on screen called echo pixels.

There are three roles for sound rays: primary ray, secondary ray and spotlight rays. The primary rays are the ones shot initially by the sonar and the ones that bounce the lines at the angle calculated by the reflection law. The secondary rays are the product of primary rays bouncing, a set is created within a reflection angle ration each time a primary ray bounces. And both primary and secondary rays have a fixed amount of companion weaker rays called spotlight rays that differ a little bit on the angle, producing a cone-like shape. All three types are treated the same when bouncing.

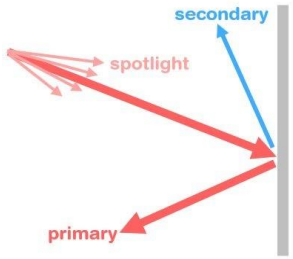

Fig. 1. Types of Rays.

Important to note that these rays are not displayed on screen, but their trajectories are perceived by the echo pixels. The transparency of the pixels depend on the energy the sound rays have when they return to the sonar, and they can be drawn further than the real lines according to the sonar functionality.

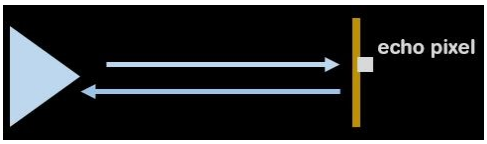

Fig. 2. Echo Pixel.

The rays that bounce more than once and then return to the sonar could create echo pixels further from the initially bounce line.

II. Related Work

A. Sonar

A sonar is an electronic device, commonly used in ships, that determines the distance to objects via bouncing acoustic waves and processing the sounds that return to it [1]. The distance between the sonar and the object is calculated as the half of the distance traveled by the sound wave [2].

B. Monte Carlo

A randomized algorithm is a technique that uses a source of randomness as part of its logic. The algorithm works by generating a random number, n, within a specified range of numbers, and making decisions based on n's values. This is normally achieved when the time it takes to take the best decision is very high, so it is replaced with a random decision [3] Randomized algorithms are usually designed in one of two common forms: as a Las Vegas or as a Monte Carlo algorithm. For this project we implemented Monte Carlo.

Monte Carlo algorithms find a solution, but the solution is only an approximation to the correct solution, so the context of the problem must be analyzed to see if Monte Carlo is the best option. The solution that is found with Monte Carlo and the resources that are possessed is acceptable and among more resources are used (in our case, the more rays we use), the solution improves [4]. The term for this algorithm was coined by mathematicians Nicholas Metropolis, Stanislaw Ulam, and John von Neumann, working on the Manhattan Project, around the 1940s.

C. Reflection

Just like light, sound waves modeled like rays follow the reflection law. This law states that a reflection event the incident angle is equal to the reflected angle, in respect to the normal of the reflection surface [5].

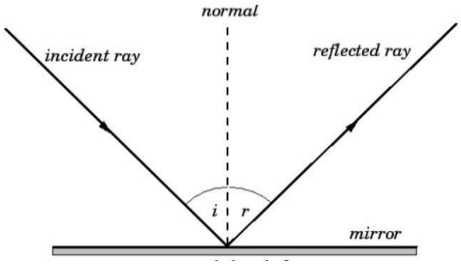

Fig. 3. Reflection Law.

When working with echolocation the formula to rotate a point around another is relevant for the echoes [8]:

$$y = (x_o - x_c)\sin\theta + (y_o - y_c)\cos\theta + y_c$$

$$x = (x_o - x_c)\cos\theta - (y_o - y_c)\sin\theta + x_c$$

Fig. 4. Point Rotation Formula.

D. Energy

The angle at the ray is reflected affects the energy, or intensity, it gets out with. The further the reflection angle is from the angle the incident ray came out of, the greater the energy loss is [6]. Different materials have different absorption coefficients, this value ranges from zero to one, one indicating that it fully absorbs all the sound that hits it [7].

E. Collisions

Collisions are essential in path tracing. The intersection between a vector, represented as an origin (ori) and direction (dir) point, and a line segment, represented as two points p1 and p2, can be calculated as follows [9].

```
def raySegmentIntersect(ori, dir, p1, p2):

    #calculate vectors
    v1 = ori - p1
    v2 = p2 - p1
    v3 = Point(-dir.y, dir.x)


    dot = v2.dot(v3)
    if (abs(dot) < 0.000001):
        return -1.0

    t1 = v2.cross(v1) / dot
    t2 = v1.dot(v3) / dot

    if (t1 >= 0.0 and (t2 >= 0.0 and t2 <= 1.0)):
        return t1

    return -1.0
```

Fig. 5. Ray-Line Segment Intersection.

In the vector-circle intersection, a vector L is created from the origin to the center of the circle, that vector is projected onto the original vector. In this way a point p is obtained in the original vector. This point is the point closest to the center of the circle. If the distance d from the point to the center of the circle is less than the radius, the vector intersects the circle [10].

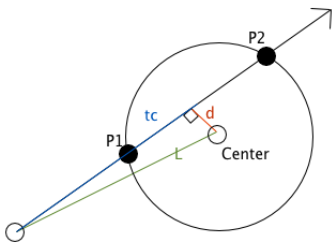

Fig. 6. Vector Circle Intersection.

III. Methods

A. Sonar

Sonar working unit is the sound ray. The sound ray contains the following data: initial angle from sonar, energy, traveled distance, number of bounces and a vector that indicates point of origin and direction. When a ray hits back the sonar, the distance traveled and the ray energy are used to draw an echo pixel in the direction of its initial angle from the sonar. To find the coordinates of the echo pixel the formula to rotate a point another is used. This formula is also used to rotate the sonar display points and its visual field of view.

Since the rays are not displayed, they don't actually travel in time, hence, the time is not a variable used. So once the ray is created three paths can happen: hits nothing and is discarded, hits a line segment and is reflected or hits the sonar and produces an echo pixel.

B. Monte Carlo

The Monte Carlo technique is used in three parts of the program: the number of primary rays that the sonar emits, the number of spotlight rays per ray and the number of secondary rays generated after bouncing off a surface. For all these cases a random angle between a range of angles is generated, that is the random factor. In real life these cases would produce an infinite amount of rays with different intensities in all directions possible, but with this technique the runtime is kept deterministic and an acceptable result can be achieved.

C. Reflection

The reflection law is used as expected, in the reflection of rays. To keep the angles between 0 and 359 degrees, simple adjustments have to be made to the angle taking into consideration if the line segment is vertical or horizontal, and if the ray comes from the left or right. Just like most graphic user interface tools, in Pygame the positive y axis goes downwards and the positive x axis goes to the right, this is important when looking at angle and coordinate data.

D. Energy

When generating the echo pixels, the ray energy translates to a number between 0 and 255 in transparency, so the energy itself goes from 0 to 100. The initial rays emitted from the sonar all have full 100 energy. With each reflection this energy is decreased by a percentage per degree and the spotlight rays are generated with a little percentage of energy of their parent ray. When the ray hits back the sonar a fixed amount of energy loss per pixel traveled is subtracted before creating the pixel. And the line segment can have different absorption coefficients, so the loss with each reflection can vary.

E. Collisions

The algorithm for the ray-line segment collision is used to determine if the ray bounces on any of the map's line segments, and if so, the exact intersection coordinate is calculated. The ray-circle algorithm is used for detecting a ray hitting back the sonar. Before checking for collisions any ray that has zero or less energy is discarded, since the pixel would not be visible.

IV. Analysis of Results

The absorption coefficients are expressed in the line segments by how bright they are, the line placement and

absorption assignation play an important role in the chosen optimized amount of rays, since they were optimized for the maps created, their design and the performance go hand in hand. Threads had to be added to the project for performance optimization and responsiveness, without them the amount of each ray where limited to two with an acceptable response time.

This is the optimized case where we use 9 initial rays, 8 secondary rays and for each of those rays, 12 of spotlight rays, it can be seen that it presented a good distribution and a solid image of what was in front of it and getting responsive execution times.

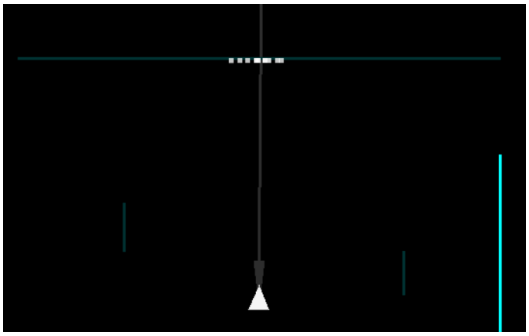

Fig. 7. Optimized shoot.

When shooting 11 initial rays, 10 secondary and 18 focus, it presents a result very similar to the optimized case, however the execution time is not the best, leading sometimes to window crashes, so this data is not the best option because balance between quality and resources is not achieved.

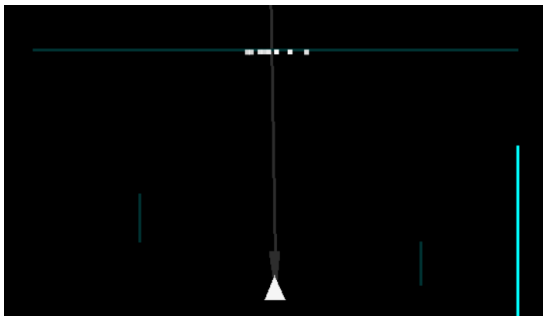

Fig. 8. Upper limit shoot.

This is the result of 7 initial rays, 4 secondary rays and 6 spotlight rays. Although time is not affected, since the result is almost instantaneous, the image result is not satisfactory, this forces us to generate a greater number of shots to achieve a good recognition of the map.

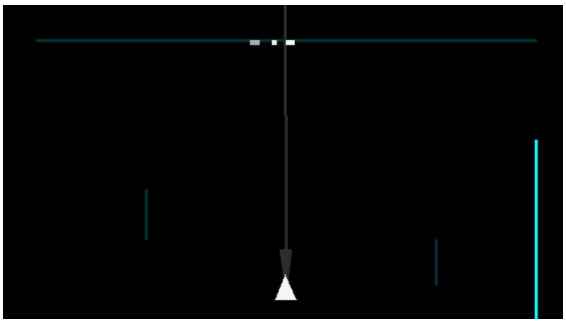

Fig. 9. Lower limit shoot.

A. Opening Angle

It was modified throughout development as it was found to have a better application if it was a small range, so the angles coming out of the Sonar are more in focus and better cover the area that the Sonar is pointing to.

B. Energy Decrease

The energy is modified by two factors, the distance and the angle of bending with respect to the angle of the initial ray. We determined the energy that would have the spotlight rays and secondary rays, using an energy loss factor per degree. The distance factor is applied at the moment that the ray collides with the sonar, the current energy minus the distance traveled in pixels is multiplied by the energy factor per pixel traveled.

C. Pixel Size

The pixels are 5x5 because if they are used in a smaller size, their display will be very limited, so it was decided to increase the size to better observe the execution of the algorithm

V. Conclusions

- The Monte Carlo technique is an effective tool to make approximate echolocation simulations, with deterministic runtimes.
- The addition of absorption coefficients contributed to a more realistic simulation.
- The model idea of primary, secondary and spotlight rays led to a satisfactory image quality.
- The reflection law is an excellent tool to keep calculations simple when working with only horizontal and vertical lines.
- The ray-circle collision for the sonar proved to be effective and efficient, with acceptable and almost undetectable margins of error for a triangle shape.
- Ray tracing is so cpu intensive and that without the use of threads, quality options are very limited.
- The energy decrements applied to angles and distances traveled, when producing echo pixels, helped to visual realism on echolocation.

VI. References

[1] M. Rouse. Sonar (sound navigation and ranging), 2011. [Online]. Available: https://whatis.techtarget.com/definition/sonar-sound-navigation-and-ranging [Accessed: 7 - Nov - 2020]

[2] University of Colorado Boulder, Measuring Distance with Sound Wave, 2020. [Online]. Available: https://www.teachengineering.org/activities/view/nyu_soundwaves_activity1 [Accessed: 8 - Nov - 2020]

[3] Rajeev Motwani and Prabhakar Raghavan. 1996. Randomized algorithms. ACM Comput. Surv. 28, 1 (March 1996), 33–37. DOI:https://doi.org/10.1145/234313.234327

[4] Illana, Jose Ignacio. *Métodos Monte Carlo*. Jan. 2013, ugr.es/~jillana/Docencia/FM/mc.pdf.

[5] Richard Fitzpatrick, Law of Reflection, 2007. [Online]. Available: http://farside.ph.utexas.edu/teaching/302l/lectures/node127.html [Accessed: 7 - Nov - 2020]

[6] California Institute of Technology, Reflection from Surfaces, 2013. [Online]. Available:https://www.feynmanlectures.caltech.edu/II_33.html#:~:text=The%20intensity%20of%20the%20reflected,%CE%B8i%2B%CE%B8t [Accessed: 9 - Nov - 2020]

[7] D. Foley, What Is A Sound Absorption Coefficient, 2014. [Online]. Available: https://www.acousticfields.com/sound-absorption-coefficient/ [Accessed: 10 - Nov - 2020]

[8] D. Rose, Rotating Points in Two-Dimensions, 2014. [Online]. Available: http://danceswithcode.net/engineeringnotes/rotations_in_2d/rotations_in_2d.html [Accessed: 9 - Nov - 2020]

[9] L. Yuen, collision between a ray and a line segment, 2020. [Source code]. Available: https://github.com/yuenlw/2DRaytracer [Accessed: 11 - Nov - 2020]

[10] L.Yuen, Algoritmos Geométricos, 2020.[Accessed: 22 - Set - 2020]