Proximity Detection Warning System using Ray Casting

Prof. Sneha Vanjari[1], Gayatri Kulkarni[2], Ashutosh Bhorde[3], Bhakti banne[4], Samiksha Kudale[5]

[1]Professor, Dept. of IT Engineering, Zeal College of Engineering and Research, Pune, India.
[,3,4,5] UG Student, Dept. of IT Engineering, Zeal College of Engineering and Research, Pune, India.

Abstract - *In the current era of technology, we are seeing a lot of objects which work autonomously. These include drones, transportation vehicles, haulers, robots and many more. When these things are being operated autonomously, they need to learn their environment by detecting threats so human workers can work safely.*

The proximity detection analysis has importance for various working such as logistics, transportation, operation management, network and mining line. The shortest path analysis is an artificial intelligence concept which has significance of developing the capability about think cause and effect, learning and thinking like a human being. Heuristic technique is a method for solving a drag which gets a result or not. The solution based on this technique may return unexpected value which varies according to the given problem. In the 3D game industry, real-time strategists use shortest path analysis control for movement of character, vehicle, and animals. Heuristic algorithms increase efficiency to avoid barrier objects on maps and search for the shortest path for the operator in an environment. Thus, Heuristic Search Algorithms are core algorithms used for artificial intelligence applications like games, robotics, astronomy and network control. This research proposes two essences, detecting obstacles beforehand and finding an optimal shortest path for the user to navigate freely in an environment. Our project aims on developing a prototype system to give an early indication using ray casting and pathfinding algorithms.

Key Words: Proximity detection, Raycasting, Pathfinding Algorithm, Heuristic technique, Artificial intelligence

1. INTRODUCTION

Nowadays, we are seeing a lot of objects which work autonomously. These include drones, transportation vehicles, haulers, robots and many more. When these things are being operated autonomously, they need to learn their environment by detecting threats so they can work safely. In this project, we are using raycasting[5] to anticipate potential collisions before they happen, so these autonomous objects can traverse around without getting damaged.

While experiencing Collision may be a negligible event to most beings, some objects are so fragile that even a slight collision could be catastrophic.

Our method for detection is based upon vector algebra results on "Ray Casting "using mesh colliders which is provided by Unity game engine to each object present in the game scene.

We have developed and implemented a collision detection method whose pre-processing step is to construct a decomposition of free space into a 3D scene and obstacles map, marking the facets that correspond to obstacle boundaries.

In this paper, we give details of our algorithm, and we describe the experiments that we have conducted to test the efficiency of this method. A primary goal of our study was to determine if, indeed, the methods based on computational vector algebra have a practical impact on the collision detection[2]problem.

Our method for detection is based upon vector algebra results on "Ray Casting "using mesh colliders which is provided by Unity game engine to each object present in the game scene. A practical contribution of this project is our work on systematic experimentation for the Collision detection problem.

The system we are going to design will use RayCasting Concept to address proximity of objects in an environment.

The first phase of the project is going to be to design a prototype scene where all kinds of movable and immovable obstacles will be placed along with the user. Then we will use vector algebra to calculate the distance and proximity between the obstacle and the user to send an early warning to the operator that a potential threat is near our reach.

In the last phase we will calculate optimal and shortest path using A * algorithm which is one of the most accurate heuristic pathfinding algorithms.[6]

2. LITERATURE SURVEY

The purpose of the literature review was to cover collision awareness and proximity detection and research based recommendations for providing obstacle fewer environments. Articles reviewed focused on efficient proximity detection. Special attention was paid to learn more about ray casting algorithms. More than 10 articles, publications (including reports, manuals and guidance materials) and 2 online training modules for Unity game engine and fact sheets were reviewed. The system uses object recognition information to augment existing object recognition algorithms. Its performance should thus continue to improve as researchers develop better recognition software, and roboticists develop better PDS (Proximity Detection System) software. We design algorithms with a fixed radius for the client and server respectively, with the purpose of reducing unnecessary collision with the user. That is we will place the objects randomly to create an environment and the user will dodge and review the proximity of the object far from him. The problem of proximity detection is often encountered in massively multiplayer online games and in network applications. The implementation is carried out using Unity Game Engine and Ray tracing Algorithms. Object detection and classification are major challenges where applications are based on robotic advancements. Navigation, detection, calculating proximity are based on the ability to recognize objects. Object detection algorithms are expected to detect and classify all instances of an object. They should be detected even if there are variations of position, scale and environment variations such as intensity.

3. PROBLEM STATEMENT

These days, we are seeing a lot of objects which work without human interaction. When these things are being operated autonomously, they need to adapt the information about them on the go by detecting threats so they can work safely.

While experiencing collisions may be a negligible event to most beings, some objects are so fragile that even a slight collision could be dangerous for the situation.

Our method for detection is based upon vector algebra results on “Ray Casting “using mesh colliders which is provided by Unity game engine to each object present in the game scene.

Therefore, the need for us to build and implement a proximity detection[1] system that will preemptively detect threats to autonomous objects and will allow them to traverse in a safe manner by avoiding collisions has arrived.

4. SYSTEM ARCHITECTURE

diagram

As the objective of the project is to make sure the primary object reaches the destination without colliding with anything. Most of the code in the project basically keeps running until the outcome is achieved. Following are the major modules in the project:

Object detection - As soon as the program is initiated, we start using ray casting to detect all the foreign objects in the vicinity of the primary object. The primary object will usually have a range within which we will be able to detect all the objects and the objects outside of the range will be ignored.

Data Preprocessing - Once we receive basic information about all the objects around the primary object, we will check if any of them are threats or not based on the distance to the primary object. We will also try to detect if a foreign object is on the move. All the information will be stored in the memory of the primary object. The primary object will also use steering behaviors[3] of the threats to detect what type of threat the object is dealing with.

Keeping track of the objects - Every render frame of the application, Object detection, and Data processing steps will run until the primary object reaches its destination. This will allow us to get the updated information about the environment of the primary object. Regardless of whether a foreign object is a static object or not, if it is a threat, we can add it to the unsafe list. All the objects in the unsafe list will be treated as if they will cause some damage to the primary object.[4]

Route calculation - In this step, based on the data we collected so far, we will calculate a safe route for our primary object. The system will be using A* algorithm, as it will provide us with heuristics so we can calculate the fastest route to the destination. To calculate the safest route, the system uses the collected data and will make sure that the primary object will be out of range of the potential collisions with the threats.

5. ALGORITHM

A) **Ray Casting**

1. Gather basic information about all surrounding objects

2. Cast a ray into the environment

3. If the collides with one of the objects

Then check if any of them are threats

4. calculate distance from current position to the said obstacle

5. alert the operator about current situation

B) **Pathfinding Algorithm (A* Algorithm)**

The A* ("A star") algorithm[7] has three important properties:

It will always return the least expensive path if a path exists to the destination, other algorithms may find a path faster but it is not necessarily the "best" path we can take.

A* uses a heuristic (a "guess") to search nodes considered more likely to lead to the destination first, allowing us to often find the best path without having to search the entire map and making the algorithm much faster.

A* is based on the idea that each node has some cost associated with it. If the costs for all nodes are the same then the best path returned by A* will also be the shortest path but A* can easily allow us to add different costs to moving through each node.

Analysis of A-Star (A*) Algorithm: -

A* is a graph search algorithm that finds the least-cost path from a given initial node to one goal node (out of one or more possible goals). It uses a distance-plus-cost heuristic function (usually denoted f(x)) to determine the order in which the search visits nodes in the tree. The distance plus-cost heuristic is a sum of two functions: the path-cost function (usually denoted g(x), which may or may not be a heuristic) and an admissible heuristic estimate of the distance to the goal. The path-cost function g(x) is the cost from the starting node to the current node.

A-Star Algorithm Pseudo Code

1. Create Start Node with Current Position
2. Add Start Node to Queue
3. While Queue Not Empty
4. Sort Node Queue byf(N) Value in Ascending
5. Get First Node From Queue call Node “N”
6. If N is Goal Then Found and Exit Loop
7. Else
8. Mark N Node as Visited
9. Expand each reachable Node from N call Node “Next N”
10. f(Next N) = g(Next N) + h(Next N)
11. Loop

6. PROPOSED OUTCOME: -

This proposed system should have a primary object which will autonomously move around in a 3D environment. The primary object will be able to detect proximity with objects within its vicinity based on the line of sight it has available. The primary object will also have some destination information which will be used to calculate the fastest path to the destination. Once the path is calculated the primary object will try to learn its surroundings using raycasting and will try to avoid all possible collisions to reduce any damage.

7. EXPECTED RESULTS

This Proximity Detection system will be able to detect obstacles beforehand.

The real-time information gathered from the detection system will be sent to the report generator which will provide a detailed output of the happening surroundings, shortest path and proximity of the obstacles on the screen.

The system should also assess the situation and generate an early level indication to indicate the operator about incoming threat. This is how an operator will be able to decide whether to change the current path or to continue with the same path.

8. CONCLUSIONS

This proximity detection application will let autonomous objects traverse safely in an unsafe environment. By avoiding collisions, a lot of resources can be saved while performing autonomous tasks. This will solve safety-related problems for many applications like automated mining carts.

In the future, the system can be upgraded with many new features. If a collision is imminent, then we can implement a system to try to detect the damage profile and reduce it despite colliding with foreign objects. In addition to that, after the collision happens, the primary object will try to stabilize its orientation and calculate a new path to its safety. Also, if possible, the primary object will try to clear the threat by itself if possible.

For further development of the study, we always need to develop more heuristic techniques, as Heuristic algorithms are never complete as they are based on Logic which is based on Hope that “This solution must be the answer of my question”. As they are developed on the basis of

resources and previous results available, they will try to provide the best solution available.

REFERENCES

1. https://ieeexplore.ieee.org/document/8904257/ (proximity detection)

2. https://ieeexplore.ieee.org/abstract/document/8835462 (Threat prediction)

3. https://www.red3d.com/cwr/steer/gdc99/ (Steering Behaviors)

4. https://www.researchgate.net/publication/265236262_Towards_Obstacle_Avoidance_and_Autonomous_UAV_Operation (obstacle avoidance)

5. https://ieeexplore.ieee.org/document/6851204 (Ray casting)

6. https://ieeexplore.ieee.org/document/6324627 (Path finding algorithm)

7. https://ieeexplore.ieee.org/document/7550934 (A* algorithm)