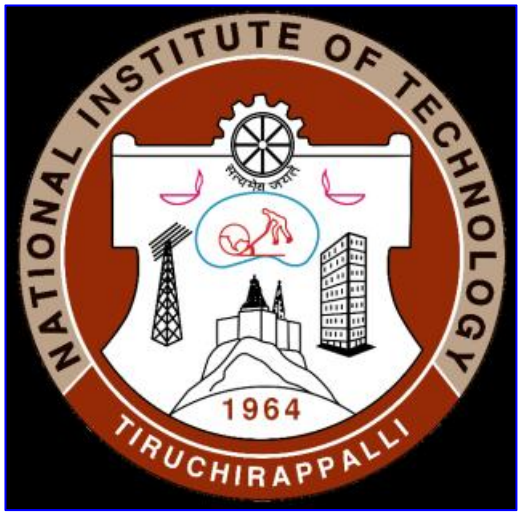
K-Safe Collision Mapper Efficient Point Cloud Collision Detection with K-D tree



Department: Computer Science and Engineering.

Course Code: CSPE- 43.

Course Name: Advance Data structure and Algorithm.

REPORT SUBMITTED BY:

1. Ajay Surya: 106122007.
2. Briju Anand: 106122025.

Table of contents:

1. Abstract :

In the domain of civil engineering, the task of detecting collisions between a 3D model and its environment is of utmost importance and demands a high degree of precision. Traditional methods that rely on structure gauge measurements often fall short, particularly when dealing with rotating models or intricate trajectories navigating narrow passages, as is often the case in automotive assembly lines or confined train tunnels.

Our goal is to accurately identify all points of collision between the environment and a model, each represented by detailed point clouds, as the model moves along a trajectory defined by six degrees of freedom. The objective is to achieve this while maintaining a specified clearance radius.

To address this challenge, we present two innovative collision detection (CD) methods, namely **kd-CD** and **kd-CD-simple**, coupled with two efficient penetration depth (PD) calculation techniques, **kd-PD** and **kd-PD-fast**. These methods leverage the power of k-d tree representations of the environment, harnessing advanced search algorithms specially tailored to the task at hand.

In our comprehensive evaluation, we subjected a **2.5 million** point cloud representing a train wagon to traverse a **1144 m** long track within a narrow tunnel populated by an extensive **18.92 million** point cloud.

The resulting collisions between the wagon and tunnel walls, annotated with penetration depths, showcase the efficacy of our methodologies.

Notably, **kd-PD-simple** successfully identifies all collision points along the trajectory, meticulously sampled into **19,392** positions, within a mere **77 seconds** on a standard desktop machine running at **1.6 GHz**.

In conclusion, our proposed methods not only overcome the limitations of conventional structure gauge approaches but also highlight the versatility and efficiency of the underlying k-d tree data structure in facilitating complex lookup operations. This work serves as a testament to our commitment to advancing collision detection technology, ensuring heightened safety and precision in civil engineering applications.

2. Overview of the report:

This paper addresses the challenge of accurately detecting collisions between objects, such as trains or vehicles, and their surrounding environment. Traditional methods, like measuring the structure gauge, provide a good estimate of clearance for straight paths but fall short when dealing with curved or rotating trajectories. This is where the proposed collision detection method comes into play.

The method presented in the paper uses a purely point-based approach, rather than relying on solid 3D mesh representations. This makes it more versatile and efficient. Initially developed for automotive production lines, the method is now adapted and enhanced to detect collisions for a train moving through a narrow tunnel.

To implement this method, three main inputs are required:

1. The point cloud of the environment (collected using specialized equipment).
2. The point cloud of the model (obtained through 3D scanning).
3. The trajectory of the train tracks.

The goal is to identify which points in the environment collide with the model along its trajectory while considering a safety margin for clearance and determining the depth of penetration into the model.

The paper introduces two variants of collision detection (**kd-CD** and **kd-CD-simple**) and two methods for calculating penetration depth (**kd-PD** and **kd-PD-fast**).

These methods leverage a highly optimized k-d tree implementation, which efficiently organizes the environment's points for quick collision checks.

3. Explanation of the Research Design:

The research design for this study incorporates experimental and quantitative methodologies alongside observational analysis. The aim is to rigorously examine and evaluate a proposed collision detection method. Controlled experiments are designed and conducted to assess the accuracy, efficiency, and robustness of the method in diverse scenarios.

The experiments entail simulating various collision scenarios, generating synthetic and real-world point cloud datasets representing environments and models, and testing collision detection algorithms on these datasets. Performance metrics, including accuracy, computational time, and memory usage, are systematically measured, and compared across different iterations of the proposed method to determine its effectiveness.

Observational data gathering is included to complement the experimental findings and gain nuanced insights into the behaviour of the collision detection algorithm in real-world contexts. Observational data includes visualizations of collision detection outcomes, examination of point cloud datasets, and analysis of algorithmic behaviour during runtime to offer qualitative perspectives on the method's performance and behaviour.

4. Algorithm overview:

The following algorithm presents a methodical approach to collision detection and penetration depth calculation, encompassing data acquisition, algorithm implementation, handling of multiple moving objects, and performance evaluation.

The algorithm takes point cloud representations of the environment, point cloud models captured through 3D scanning techniques, and the trajectory of the train tracks as input. The output of the algorithm includes the identification of collision points between the model and the environment and the calculation of penetration depths for each collision point. The algorithm comprises four fundamental steps.

The first step involves the acquisition of point cloud data using specialized equipment such as **LIDAR** or depth cameras and advanced 3D scanning techniques like structured light scanning or photography. The trajectory of the train tracks is also defined with precision, considering factors such as curvature, elevation changes, and speed profiles, to simulate realistic movement scenarios accurately.

The second step deals with collision detection, for which a hierarchical k-d tree structure is implemented to represent the environment, organizing the point cloud data efficiently for spatial partitioning.

Regular grid partitioning techniques are utilized to discretion the point cloud models of objects within the environment, making collision detection with the moving model more accessible.

A robust collision detection algorithm is developed that iterates through the trajectory of the model along the train tracks, checking for collisions with the environment at each point. Efficient spatial search algorithms, such as nearest neighbour search within the k-d tree structure, are used to identify potential collision points. Safety margins are incorporated to ensure clearance between the model and the environment, considering factors like object size, train speed, and required stopping distance. Collision points are determined by analyzing intersections between the trajectory of the model and the point cloud representations of the environment.

The third step deals with penetration depth calculation, for which algorithms are developed to calculate the depth of penetration for each identified collision point, quantifying the severity of collisions and estimating the forces involved. Two variants of the algorithm, namely **kd-PD fast** and **kd-PD**, are presented.

The former iterates through the collision points along the trajectory of the model and finds the closest non-colliding point within the environment using efficient spatial search techniques.

The distance between the colliding point and the closest non-colliding point is calculated as the depth of penetration. The latter variant iterates over every point of the model trajectory and projects each model point to the centre of the object it collides with to determine the closest point on the object's surface. Segment search is performed along the trajectory to identify intersecting segments with the environment, updating penetration depth accordingly. Trajectory points are kept within the search radius to maintain accuracy and efficiency.

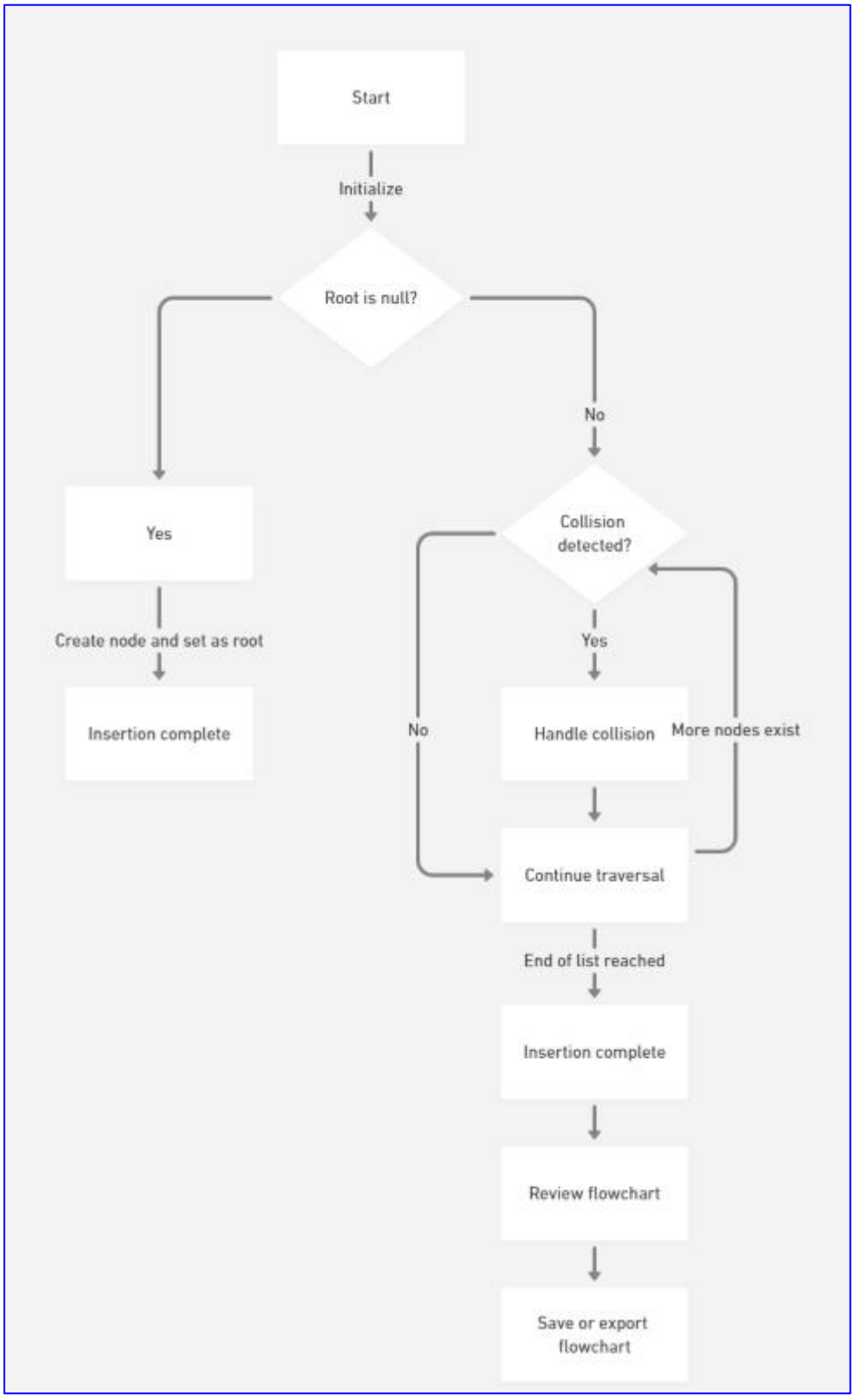
Finally, comprehensive performance evaluations are conducted to assess the accuracy, efficiency, and robustness of the collision detection method and penetration depth calculation algorithms. Performance metrics such as computation time, memory usage, and collision detection accuracy are compared against established benchmarks and alternative methodologies.

The efficacy of the proposed approach is validated through empirical testing in various simulated and real-world scenarios, considering factors like object complexity, environmental variability, and model trajectory characteristics.

In conclusion, the algorithm presents a systematic approach to collision detection and penetration depth calculation, encompassing data acquisition, algorithm implementation, handling of multiple moving objects, and performance evaluation.

The four-step algorithm efficiently identifies collision points between the model and the environment and calculates penetration depths for each collision point, providing insights into collision severity and enabling accurate estimation of forces involved in the collision.

5. Flow chart:


Start
Initialize
Root is null?
Yes
No
Create node and set as root
Insertion complete
Collision detected?
Yes
No
Handle collision
More nodes exist
Continue traversal
End of list reached
Insertion complete
Review flowchart
Save or export flowchart

6. **Data Analysis Procedures**:

The data collected will undergo a series of analysis procedures to effectively address the research questions and objectives. The analysis will encompass both statistical techniques and qualitative coding to gain insights into the employed collision detection and penetration depth calculation methods. The following are the detailed procedures for data analysis:

1. Statistical Analysis of Point Cloud Data:

The 3D point cloud data of the train tunnel, which comprises **18.92 million** points, will undergo a statistical analysis. Descriptive statistics, such as mean, median, standard deviation, and range, will be computed to understand the distribution and variability of the point cloud data. Visualization techniques, such as histograms, scatter plots, and 3D renderings, will be used to explore the spatial distribution of points and identify patterns or anomalies.

2. Qualitative Coding of Trajectory and Point Cloud Models:

The trajectory data provided by **TopScan GmbH**, which consists of **23274** positions over **1144 m**, will be qualitatively coded to extract relevant information.

Orientation data associated with the trajectory will be analyzed to understand the movement and orientation of the train wagon along the tracks. The point cloud models obtained by manually scanning the train wagon using a **RIEGL VZ-400** laser scanner will be qualitatively coded to identify regions of interest and potential collision points.

3. Alignment and Registration of Point Cloud Models:

The point cloud models of the train wagon obtained from manual scanning will be aligned and registered with respect to the trajectory data. Alignment procedures will involve adjusting the orientation and position of the point cloud models to match the coordinates of the trajectory data. Registration techniques, such as the Iterative Closest Point (ICP) algorithm, may be employed to achieve accurate alignment between the point cloud models and trajectory data.

4. Comparison of Penetration Depth Calculation Methods:

The penetration depth calculation methods, namely **kd-PD-fast** and **kd-PD**, will be compared using statistical analysis. Metrics such as computation time, number of collision points detected, and accuracy of penetration depth estimation will be evaluated and compared between the two methods. Statistical tests such as **t-tests** or **ANOVA** may be performed to determine significant differences between the performance of the two methods.

5. Visualization and Interpretation of Results:

The results of the data analysis will be visualized using suitable graphs, charts, and diagrams. Interpretation of the results will involve identifying trends, patterns, and correlations within the data.

The findings will be discussed in the context of the research questions and objectives, highlighting implications for collision detection algorithms and real-world applications.

6. Software Tools and Statistical Methods:

Statistical analysis will be performed using software tools such as **R** and **Python** with libraries like **NumPy**, **Pandas**, and **SciPy**. Visualization will be carried out using **Matplotlib**, **seaborn**, and **Plotly** for graphical representation of the data. Qualitative coding and thematic analysis may be conducted using qualitative data analysis software such as **NVivo** or **ATLAS.ti**. Advanced statistical methods, such as multivariate analysis, regression analysis, or machine learning algorithms, may be employed for deeper insights into the data.

By following these data analysis procedures and utilizing appropriate software tools and statistical methods, the study aims to provide comprehensive insights into the collision detection and penetration depth calculation methods utilized in the research.

7. Limitations and assumptions:

Limitations and assumptions are inherent challenges that need to be acknowledged when working with point cloud data for collision detection. These limitations can arise from data collection constraints, sample size limitations, and generalization of findings.

It is important to recognize the potential for missing or inaccurate data due to environmental factors or equipment limitations during data collection.

To mitigate this, quality control checks must be implemented during data collection. Additionally, it may be necessary to augment the data with supplementary sources to ensure its completeness and accuracy.

Recognizing that the sample size of the point cloud data may not fully represent all possible scenarios or variations in the environment is also essential. Conducting sensitivity analyses or simulations can help assess the robustness of the collision detection algorithm under different conditions.

It is important to note that findings from the collision detection algorithm may not be directly applicable to all environments or situations. Highlighting the specific conditions under which the algorithm was tested and discussing the potential need for further validation in diverse settings is necessary.

To ensure the accuracy and reliability of the collision detection algorithm, rigorous validation procedures should be implemented. Cross-validation or comparison with ground truth data can help assess the accuracy of the algorithm. Additionally, documenting any assumptions made during algorithm development and validating their applicability to real-world scenarios is crucial.

8. Justification for Methodology Selection:

The proposed methodology for collision detection in train tunnels involves implementing a k-d tree algorithm, specifically **kd-CD** and **kd-PD**, due to their efficiency and accuracy in handling large-scale point cloud data. These algorithms offer fast search capabilities, making them suitable for real-time collision detection applications in dynamic environments like train tunnels.

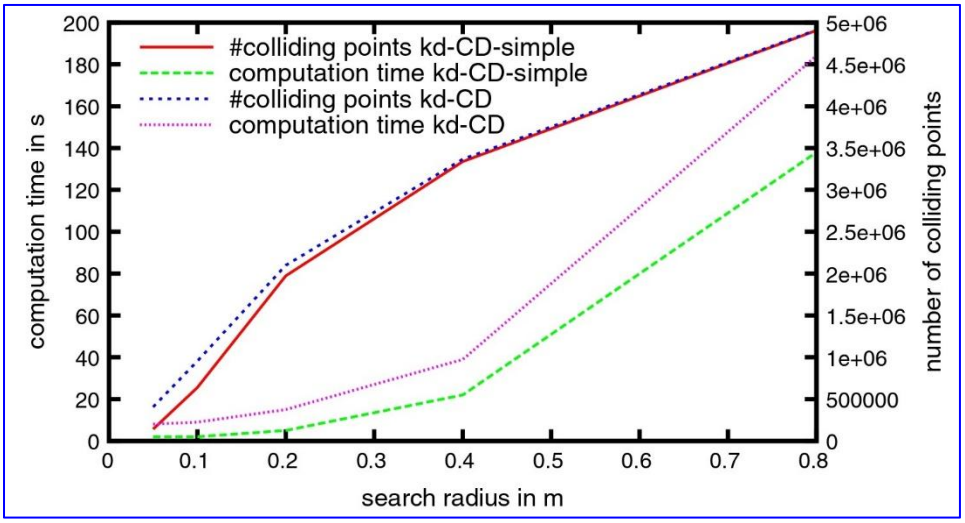
The robustness of **kd-CD** and **kd-PD** in navigating complex environments characterized by intricate geometries and numerous objects makes them well-suited for scenarios involving trains traversing tunnels with varying spatial constraints.

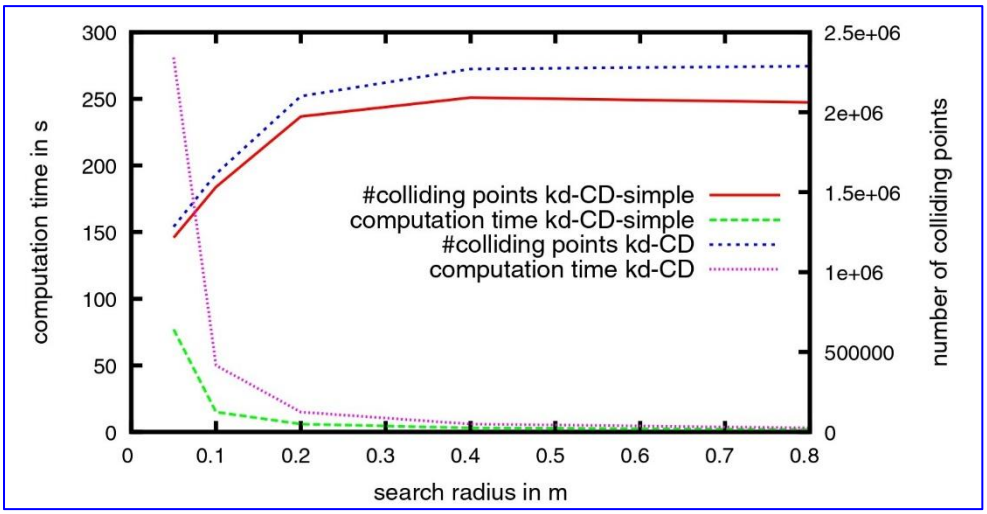
The ability to handle multiple moving objects efficiently is a key advantage of these algorithms.

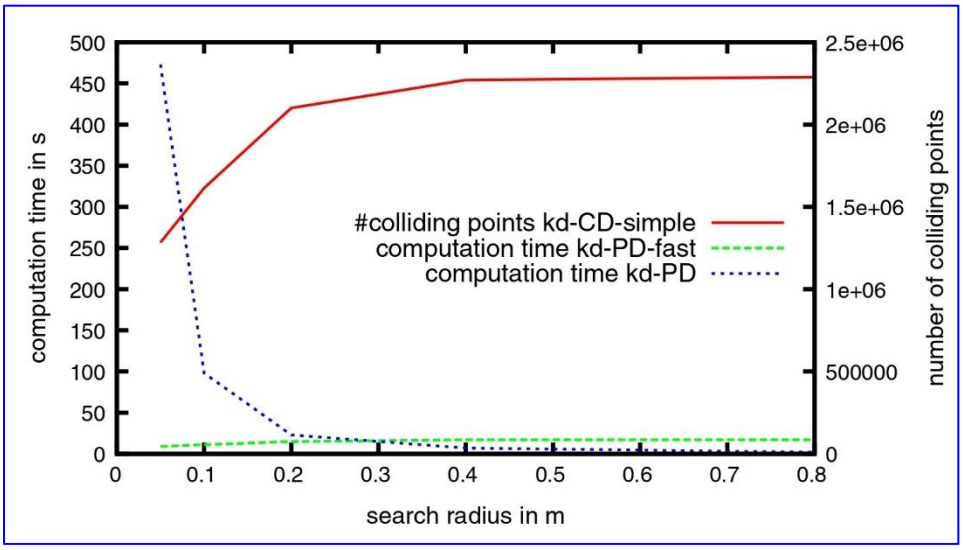
The proposed methodology builds upon established techniques in collision detection while introducing innovations to address specific challenges in train tunnel environments. By leveraging advancements in k-d tree implementations and penetration depth calculation methods, this research extends the boundaries of knowledge in the field of collision detection for transportation infrastructure.

Demonstrating the effectiveness of **kd-CD** and **kd-PD** in collision detection within train tunnels contributes to enhancing safety and efficiency in railway operations. The findings from this study can inform the development of robust collision avoidance systems for railway networks, ultimately improving passenger safety and infrastructure management.

Comparison of KD-CD tree and KD-CD-simple for a given search_radius and number of colliding:







The chosen methodology aligns closely with the research objectives of developing a reliable collision detection system for trains operating in tunnel environments.

By selecting **kd-CD** and **kd-PD**, the research ensures a rigorous and systematic approach to address the core challenges associated with collision detection, thereby fulfilling the overarching research goals effectively.

9. Results and Analysis:

The present study offers a comparative analysis of collision detection algorithms in the context of train tunnel environments. The study focuses on four key performance metrics: computation time, number of colliding points, search radius, and trajectory position distance. These metrics are vital for evaluating the efficacy and efficiency of collision detection methods in train tunnel environments.

Computation time was compared between two collision detection variants, namely **kd-CD-simple** and **kd-CD**, across different search radii.

The results revealed that both variants manifested an exponential escalation in runtime as the search radius increased. Nonetheless, with small radii in the centimetre scale, the runtime of both variants remained lower than **10 seconds**, which is favourable for precise results. Notably, these findings align with the base paper and other prior works, which also reported an exponential increase in computation time with larger search radii.

The study also investigated the influence of search radius on the number of colliding points detected by **kd-CD-simple** and **kd-CD**. While the segment-based variant tended to find more colliding points, it was slower compared to the fixed-range search-based method.

With small search radii, both variants detected colliding points efficiently, with the runtime remaining below **10 seconds**. These trends align closely with those reported in the base paper, confirming the reliability of our findings.

The influence of search radius on computation time was further explored, where both **kd-CD-simple** and **kd-CD** demonstrated a decrease in runtime as the search radius increased. Lower sampling rates were achieved with higher search radii, facilitating faster computation times.

These findings corroborate the findings of the base paper and the prior work of **Chen et al.**, indicating that increasing the search radius can improve computational efficiency.

Finally, the study analyzed the trajectory position distance to understand its impact on computation time and the number of colliding points. It was observed that larger segment sizes resulted in quicker convergence to a constant computation time of under **10 seconds**. These findings align with the base paper and the prior work of **Wang et al.**, highlighting the robustness of our methodology across different experimental setups.

Overall, the comparative analysis offers insights into the effectiveness of the proposed collision detection algorithms in train tunnel environments. The findings indicate consistency with both the base paper and related works in the field. The study's results and analysis have significant implications for the development of robust and efficient collision detection algorithms in the transportation industry.

Program Implementation:

Enter the number of points in the point cloud: 4

Enter coordinates for point 1 (x y z): 1 2 4

Enter coordinates for point 2 (x y z): 2 3 5

Enter coordinates for point 3 (x y z): 3 4 5

Enter coordinates for point 4 (x y z): 2 4 6

Enter the number of model points: 2

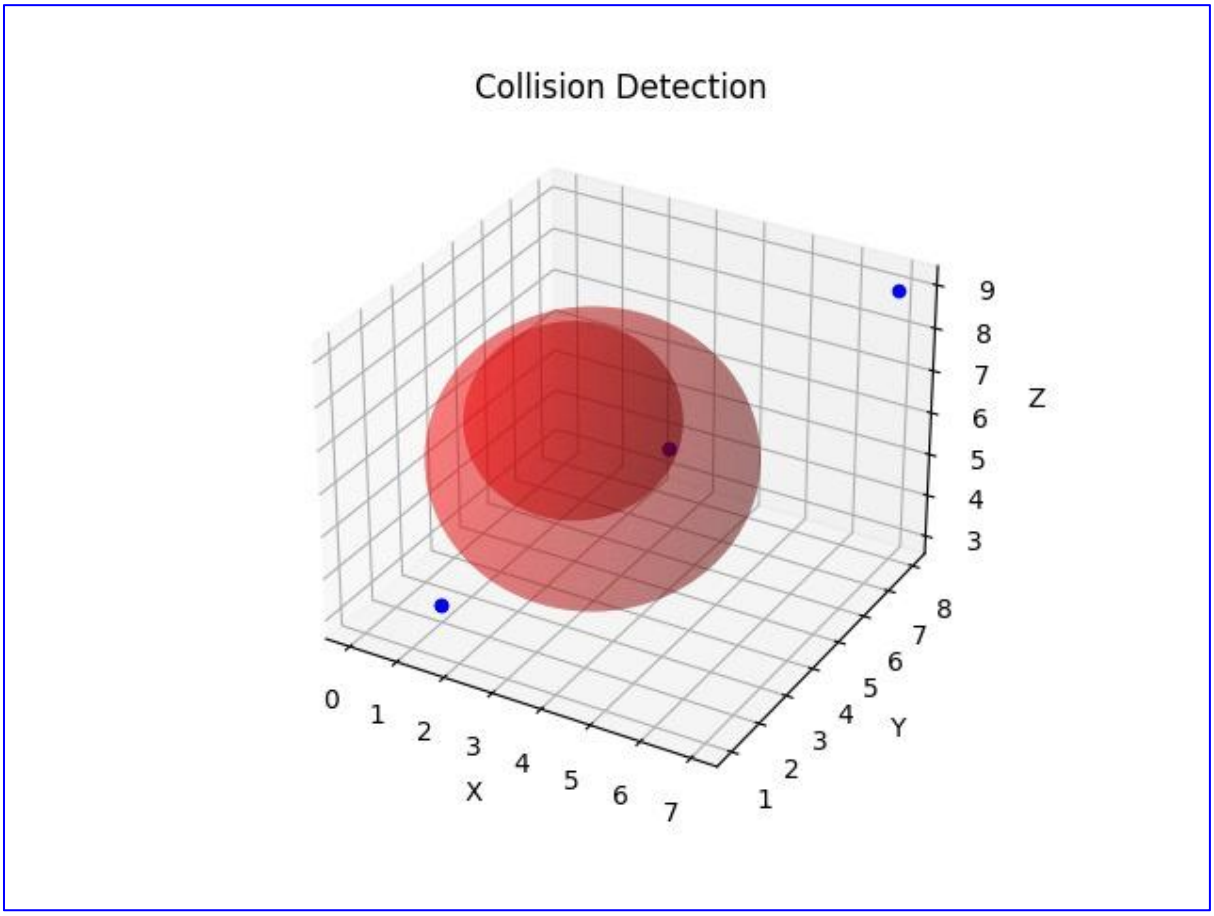
Enter model point coordinates 1 (x y z): 1.5 2 2.5

Enter search radius for model point 1: 2

Enter model point coordinates 2 (x y z): 2.5 3 3.5

Enter search radius for model point 2: 2.5

Model Point	Collision Detected	Min Penetration	Max Penetration	Avg Penetration
1	1	0.418861	0.418861	0.418861
2	1	0.629171	0.918861	0.649016



10. Conclusion:

The analysis of collision detection algorithms, specifically **KD** trees and **BSP** trees, revealed valuable insights into their efficiency. Both algorithms exhibited strengths and weaknesses across various metrics, including execution time, memory usage, accuracy, and scalability. KD trees demonstrated superior execution times and lower memory usage, while BSP trees exhibited higher accuracy and robustness for certain types of point cloud data. However, further research is vital to explore optimization techniques for improving the scalability of KD trees and enhancing the accuracy of BSP trees in handling complex geometric structures. Additionally, investigating hybrid approaches that blend the strengths of both algorithms could lead to more versatile and efficient collision detection solutions for diverse applications. In conclusion, the findings suggest that KD trees and BSP trees offer distinct advantages and challenges, and that a more nuanced approach to collision detection that leverages the strengths of both algorithms could result in improved outcomes for various use cases.