Journal of Engineering and Technology for Industrial Applications

ITEGAM-JETIA

Manaus, v.11 n.52, p. 237-246. March./April., 2025.
DOI: https://doi.org/10.5935/jetia. v11i52.1677

ISSN ONLINE: 2447-0228

RESEARCH ARTICLE OPEN ACCESS

ENHANCING REAL-TIME ANIMATION: ENSURING DISTINCTIVENESS IN CROWD DYNAMICS THROUGH PHYSICS-BASED COLLISION AVOIDANCE

Imane DRIDI [1], Cherif FOUDIL [2]

[12] LESIA Laboratory, Institute of Science Department of Computer Science, Mohamed Khider University, Biskra, Algeria.

[1] https://orcid.org/0009-0009-7958-5462, [2]http://orcid.org/0000-0003-1539-835X

Email: imane.dridi@univ-biskra.dz, cherif.foudil@univ-biskra.dz

ARTICLE INFO

Article History
Received: March 03, 2025
Revised: March 20, 2025
Accepted: March 15, 2025
Published: April 30, 2025

Keywords:
Animation variety,
Collision prevention,
Newton's laws and the laws of momentum,
Optimization pool object algorith
Occlusion culling algorithm.

ABSTRACT

Crowds are essential in daily activities, performing as dynamic systems of human interaction. They involve numerous individuals collecting in specified locations for diverse activities, whether in urban environments, public events, or social interactions. In virtual environments, animated crowds frequently display repetitive behaviors and a deficiency in movement diversity, leading to unrealistic simulations. It is imperative to provide distinctive and diverse actions for each character, avoiding visual duplication to augment realism. This paper introduces a method to enhance crowd variety by ensuring distinct and realistic character movements. We propose providing various motion types to prevent the duplication of cloned characters. Our approach creates a set of animations and utilizes techniques to control character velocity, ensuring distinctive and convincing movements. Furthermore, we present collision prevention methods based on Newton's Laws, the conservation of momentum, and the laws of kinetic energy. Ray-casting determines collision velocities by considering each character's mass and velocity without external forces. We implement a hybrid pool object approach and occlusion culling techniques to optimize real-time performance, increasing FPS (framed per second) and reducing computational load. These experiments evaluate the efficacy of our technique under different conditions. The results demonstrate the method's effectiveness and flexibility in dynamic environments.

I. INTRODUCTION

A virtual crowd involves individual virtual entities that reflect behaviours identical to those of their adjacent neighbours, each possessing distinct actions and properties. Animators categorize the crowd into numerous entities and regulate the crowd's collective behaviours, the groups' movements, and the conduct of each virtual entity. Monitoring identical characters performing the same behaviour makes it feasible to regulate crowd movement, navigate around obstacles, and facilitate interactions with 3D characters or virtual humans.

Real-time crowd simulation is a technique for animating the movement of large groups of characters within a virtual environment. As the crowd moves, each character dynamically interacts with others based on predefined behaviours, such as avoiding collisions or navigating obstacles. These interactions are defined by algorithms that adjust character paths to maintain personal space, avoid obstructions, and respond to environmental changes. It results in an evolving, natural collective behaviour that unfolds in real time, allowing characters to react continuously to their surroundings and each other. The challenge lies in executing these processes instantly, ensuring fluid and realistic movement in virtual settings like games and simulations.

Our crowd animation technique produces distinct locomotion cycles for each character, creating diverse movements that enhance realism and complexity within the scene. Beyond movement variation, we introduce a technique to prevent character collisions, ensuring smooth and natural interactions among individuals in the crowd. Our approach is based on Newton's laws of motion, incorporating principles of momentum and kinetic energy conservation under the assumption of no external forces. By applying these physical principles, our method accurately computes the final velocities of characters after interactions, maintaining realistic and natural movement.

This integration of dynamic animation and collision avoidance leads to a more immersive and cohesive crowd simulation, making character behaviour more believable in complex virtual environments. Motion has a significant role in capturing crowds' diversity since virtual characters' physical appearance is often limited. To address this, many researchers have developed animation methods. For example, one study [1] aims to minimize memory usage while generating and animating various crowd characters.

This approach involves two main steps: first, simplifying and segmenting virtual character data into essential body parts; second, integrating installation and peeling information into a shared texture space for the new characters. Rigging and skinning data are incorporated into global textures through UV parameterization, allowing all created characters to share a single set of rigging and skinning textures for each gender or age variant (male, female, child). It reduces memory for efficient large-scale crowd simulations but may introduce artefacts, impacting animation quality and crowd smoothness [1].

The study in [2] includes different approaches to improve crowd simulation in important areas. It utilizes a multi-domain planning method and uses steps instead of depending exclusively on the speeds and placements of the character roots. The animation focuses on frame-based solutions to avoid foot-sliding artefacts and synthesizes character movements to ensure they follow the provided paths. In addition, the research introduces a distinctive rendering method based on joint impostors, which is validated by the results of validation experiments. The authors in [3] introduces a data-driven optimization strategy for dynamically adjusting individual agent velocities in response to real-world crowd movement patterns. It improves the realism of the simulated pedestrian behaviours.

The study in [4] focuses on minimizing memory usage and optimizing the rendering step using two complementing architectures. The first is a skeleton linked to octrees for each limb, which determines the amount of detail for geometry and animation. The second structure is scene tiling, which determines the level of detail required for geometry, animation, and character behaviour. A quad-tree is constructed on top of this tiling to improve rendering optimization, allowing geometry from many characters to be combined in locations away from the camera. Furthermore, this tiling functions as an evaluating mechanism, increasing the effectiveness of collision avoidance calculations.

The method presented in [5] uses Gaussian Process Dynamical Models to generate probabilistic low-dimensional poses for motion. Sampling trajectories through Monte Carlo Markov Chains creates infinite motion variants, including blended versions, without costly non-linear interpolation in the mesh domain. This approach allows for the efficient generation of diverse and realistic motion variations, offering animation flexibility while maintaining computational efficiency. The ability to generate such diverse movements without the overhead of complex interpolation techniques makes it a valuable tool for realistic and dynamic motion synthesis in various applications. The output variations generated by the method in [5] remain relatively similar to the input movements while allowing for changes in poses and timings.

The goal is to create infinite variations for each blended motion using a motion parameterization approach. This approach provides control for customizing motion and simulations, but parameterizing motion capture and controlling variations can be complex and less effective for complex movements.

In addition, to create a diversified heterogeneous crowd, [6] presents a multi-agent reinforcement learning-based method that incorporates a variety of input settings. This approach demonstrates generalized crowd navigation, allowing for the simulation of diverse individual behaviours within the crowd. The system can account for complex agent interactions by utilizing reinforcement learning, generating realistic and dynamic crowd movements. This method enhances the ability to simulate large crowds with varying characteristics, ensuring a more natural representation of human behaviour in crowded environments. These techniques benefit applications in gaming, virtual reality, and crowd management simulations, where realistic and diverse animations are necessary.

The study in [7] presents a local collision avoidance algorithm for uncrewed surface vehicles (USVs) based on the International Regulations for Preventing Collisions at Sea (COLREGs). The algorithm is divided into three main sections: collision risk assessment, which evaluates the potential for collisions; steering occasion determination, which identifies the need for course adjustments; and navigation waypoint updates, which modify the vehicle's path to avoid obstacles. This approach ensures safe navigation for USVs in complex environments by considering real-time collision risks and applying the rules of maritime navigation to maintain safe distances from other vessels and obstacles.

The algorithm uses the closest point of approach (CPA) method to evaluate collision risk within an analytically defined angle range, assessing potential threats from other vessels or obstacles. The steering occasion determination calculates a supplemental steering angular velocity, allowing the USV to adjust its course and maintain a safe distance. The navigation waypoint update generates temporary waypoints, ensuring compliance with COLREGs for safe passing and crossing situations. Finite state machines (FSMs) control these three components, coordinating the decision-making process. This method guarantees adherence to COLREGs, reduces collision risks, manages dynamic situations, and enhances USV efficiency.

Nevertheless, incorporating COLREGs may elevate computational requirements, thus affecting performance. Moreover, environmental factors limit the algorithm's practical application. In contrast, [8] used a rule-based Bayesian Network approach to model collision issues for ships and marine structures. This method offers clear reasoning and high visibility, making it easier to understand the decision-making process while providing stability in the system's behaviour.

However, it may lead to uncertain or inconsistent ship behaviour due to the inherent conditions required for state transitions. For example, conflicts can arise between the triggering situations for specific behaviours, which may result in unexpected or incorrect decisions. These conflicts can cause system failures or degrade the algorithm's reliability, especially when dealing with complex or dynamic maritime scenarios. Additionally, the algorithm's ability to handle intricate situations

is limited, which affects its overall performance in real-world applications. The challenge of analyzing and resolving these complicated scenarios means that while intuitive and interpretable, the rule-based approach may not always be the most effective solution in dynamic environments where rapid and flexible decision-making is required [9],[10]. As a result, alternative approaches or hybrid systems might be necessary to improve robustness and adaptability in collision avoidance algorithms for maritime navigation.

The authors in [11] proposed an innovative approach for modelling agent movements in crowd simulations with deep reinforcement learning (DRL). Based on environmental awareness, this technique emphasizes the development of agents' navigation strategies, including locomotion and obstacle avoidance. This research substantially advances the field by establishing a parametric framework for developing crowd simulation environments, thus allowing performance evaluation across numerous scenarios.

The authors of [12] present the CrowdMoGen framework employs a dual-phase approach for generating crowd motion based on text input. The Crowd Scene Planner employs a Large Language Model (LLM) to analyze textual descriptions and generate motion plans, specifying both semantic (activities) and spatial (trajectories) characteristics.

The Collective Motion Generator employs a diffusion-based model, incorporating InputMixing and Control Attention approaches, to synthesize character movements that realistically conform to intended dynamics. This method facilitates zero-shot motion generation, obviating the necessity for paired training data. Consequently, CrowdMoGen improves adaptability and authenticity in crowd animation.

In this paper [13], authors proposed an approach incorporates Anisotropic Fields (AFs) to address the limitations of replicating homogenous behaviors. By capturing the uncertainty and variability in crowd motions, it enhances realism and behavioral diversity in simulations. The produced AFs are integrated into the crowd simulation engine, influencing agent decision-making processes. The method examines video data in real crowds to identify movement patterns and transform them into AFs. This approach involves capturing individual motions and calculating local directional preferences to create AFs that represent observed behaviors.

This paper addresses a significant challenge in real-time crowd simulation: preventing unintended character movements in virtual environments and optimizing real-time performance by combining the pooling and culling algorithms to improve the efficiency and quality of interactive models. We aim to ensure consistent frame rates, reduce the computational load of hardware (CPU, GPU and memory), and ensure precise navigation for realistic interactions while maintaining movement within a reasonable range of animations.

This approach improves visual realism and ensures a more convincing crowd simulation experience. We apply Newton's laws of motion, momentum, and kinetic energy to improve collision detection and avoidance. By estimating and modifying character movement in real-time, we can prevent collisions while maintaining physical realism, resulting in smoother, more realistic character interactions and a more realistic virtual environment. This approach ensures that characters move and interact in a way consistent with the laws of physics, contributing to an immersive simulation.

II. MODELING VIRTUAL HUMAN ANIMATION

Simulating different crowds is important for real-time applications such as games, animated films, and navigation systems. These applications benefit from added features and details that improve the visual quality of the virtual environment and its elements, thereby enhancing the overall realism and performance of the system. It is essential to explore how animation variation and collision avoidance can generate a range of motions by creating unique movements for each character model, improving FPS (frames per second) stability by reducing the rendering load and making the FPS calculations consistent. This approach allows for the production of diverse animations, ensuring each character moves in its distinct way. Applying animation diversity is crucial in creating a crowd that highlights each character's individuality, contributing to a more dynamic and realistic crowd simulation (see Figure 1).

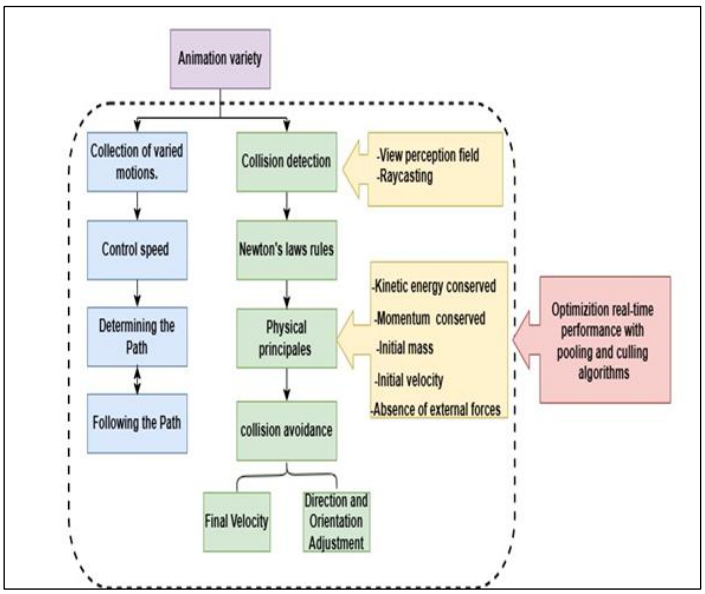

Figure 1: Animation and collision avoidance approach in crowd simulation.
Source: Authors, (2025).

II.1 ANIMATION APPROACH

We utilise an animator controller and scripting to create diverse and unique movements for our models. The animator controller efficiently manages and blends multiple animations. At the same time, scripting allows us to activate animations based on specific conditions or user interactions, giving each model its personality and individuality. This approach ensures that our models move realistically and interactively, enabling walking, running, jumping, turning, and idling (see Figure 2).

We developed a collection of animated clips and integrated them with scripting in the animator component to achieve this. This setup triggers animations based on defined conditions or user inputs, ensuring fluid and lifelike character behaviours. We efficiently assign and control various movement patterns by constructing multiple animations with distinct attributes and motions. Additionally, we enhance diversity by seamlessly blending animations, resulting in more dynamic and natural motion transitions. Our animation system is enhanced by C# scripting, ensuring accurate animation behaviours while preserving clarity and flexibility. Through custom scripting algorithms, we introduce realistic variations in motion. A key aspect of this process involves dynamically generating velocity

values to regulate animation speed, ensuring smooth transitions between different movements. These velocity-based transitions create a seamless animation flow, enhancing realism and immersion. We initially position models randomly in different directions to ensure a varied animation cycle for different virtual characters. Each model is assigned a unique animation type with specific speed parameters and the ability to jump to a certain height. These techniques generate diverse locomotion cycles, resulting in more visually realistic crowd movement.

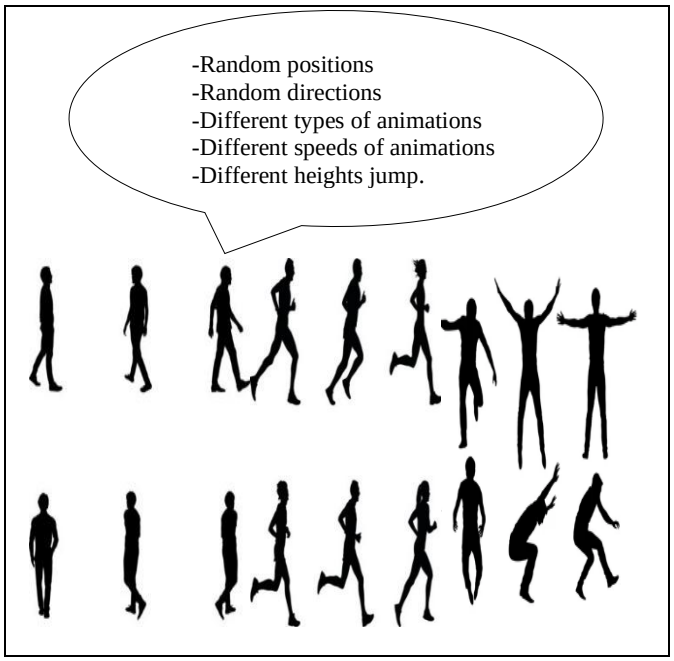

Figure 2: Animation and collision avoidance approach in crowd simulation. Employing a variety of animation styles for different models, each with its own speed.
Source: Authors, (2025).

II.2 MOVEMENT PATH VISUALIZATION

Creating a unique animation style for characters in a virtual world or interactive environment is crucial for defining their personalities and improving the fluidity of their movements. Blending various animations with controlled movement along predefined paths allows characters to be represented distinctly and dynamically. This method ensures that each character moves uniquely, engaging and consistent with their role in the virtual world, which adds depth and realism to the overall experience. We explore a strategy demonstrating how our crowd can distinguish its animation style.

Algorithm: Character animation with paths

Input: character, movement, velocity, position list of points.

Output: a path to follow.

- Determinate the initial position of the path
- Determinate the destination position of the path
- Determinate the movement.
- Determinate the velocity
- Determinate the path
- Following the path

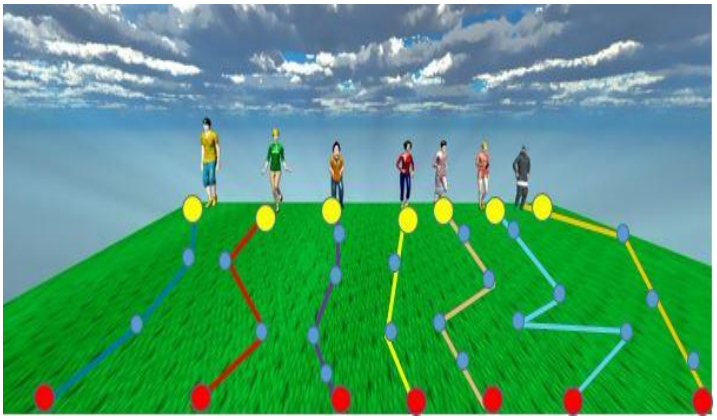

Figure 3:Various animations with different speeds to follow the path sequence.
Source: Authors, (2025).

It includes animating characters with diverse movements and controlling their speed along a specified path. In our environment, we set a starting point and a final destination for each character, guiding them through a path to reach their goal effectively. A key aspect of this strategy is using varied motion styles to reflect the characters' personalities and actions. These styles encompass a range of movements, including idle states, walking, strolling with a briefcase, performing a feminine gait, a unique older man's walk, walking with joy, rhythmic hip-hop dancing, texting while walking, and running. Each motion style adds depth and individuality to the characters, ensuring they are dynamic and realistic in different situations, ultimately enhancing engagement and realism within the virtual environment (see Figure 3).

In addition to animation variation, controlling the movement speed and path traversal is another critical factor. Each model is defined to perform particular types of movement and can change how it moves along a predetermined path. The animation is controlled by a speed parameter ranging from 1 to 50 (m/s). These speed decisions are precisely selected to ensure the movement appears realistic and visually appealing, enhancing the overall sense of motion within the defined parameters. As a result, each character's movement feels distinct, representing their characteristics and activities. A character's movement speed may also reflect its emotional condition and give additional depth to the animation. We ensure that each character moves effortlessly and naturally by blending different animations with varying movement speeds. The variations in speed and animation styles emphasize their individuality, making their behaviours more noticeable.

Figure 4: Enhancing visual animation diversity across various templates in an urban context.
Source: Authors, (2025).

This method not only improves the visual dynamic of the environment but also enhances transmitting emotions and intentions through character movements and animations (see Figure 4).

II.3 CROWD COLLISION DETECTION APPROACH

We present a comprehensive description of our approach, which aims to improve and achieve realistic results in collision avoidance within crowds. Collision avoidance is a critical challenge in crowd simulation, as it ensures individuals within the environment move without unrealistic interactions. Addressing this issue is essential for creating an efficient crowd system. Many researchers have proposed various methods to implement collision avoidance behaviour in crowd simulations.

The algorithm used is based on a technique called Raycasting, Newton's Laws, along with the principles of momentum and kinetic energy conservation for collision detection and possibly for determining visibility or field of view:

• Raycasting Basics: It estimates the intersection between our models during their movements from the projection of the current character's position to the other character. It involves casting a ray (a straight line) from a specific point (often a character's position) in a particular direction (e.g., direction of movement or field of view). The ray continues until it intersects with an object in the scene or reaches a maximum distance.

• Collision Detection: Raycasting is a technique used in character movement to detect collisions with nearby objects. When a character moves forward, a ray is cast in the direction of movement. This ray checks for obstacles within its path, such as objects or other characters. If the ray intersects with an object, it indicates a potential collision, allowing the game to respond appropriately by stopping the character or adjusting its movement. This method is efficient for real-time environments, enabling accurate collision detection without complex geometry calculations.

• Field of View Representation: Raycasting can also assess what is visible from a character's perspective. By casting rays in several directions within a defined field of view, we can determine which objects or entities fall within that area and are visible to the character. Based on this information, we calculate the mass and initial speed (e.g., WalkSpeed, RunSpeed) to apply to the character's movement.

• Collided characters: The raycasting method determines if characters have collided after a set number of time steps. It calculates the distance between them, considers their masses and velocities, and selects the faster character. By casting rays along the movement path, the method checks for potential collisions. If a collision is detected, the model can adjust the characters' positions or velocities based on their masses and speeds, ensuring realistic interaction and response to obstacles or other characters.

• We applied Newton's Laws, along with the principles of momentum and kinetic energy conservation, to model the dynamics of motion and calculate the final velocities and direction changes for each character during collisions. These physical laws govern the interactions between objects and are essential for accurately simulating realistic motion. The conservation of momentum ensures that the total momentum before and after a collision remains constant, while kinetic energy conservation applies to elastic collisions where no energy is lost.

These principles apply without external forces, allowing us to predict the results of character collisions with high accuracy and realism.

Where, $p_1, K_1, p_2, K_2, p'_1, K'_1, p'_2$ and K'_2 are the momentum and kinetic energy of character 1 and character 2 before and after the collision, respectively [14].

Table 1: Determination of the final velocity of the characters after the sensing collision by the raycasting technique, Newton's laws, the concepts of momentum and conservation of kinetic energy.

Scenarios	Various mass and velocity cases.	The velocities at the time of collision.	Final velocities after collision detection.
Scenario1	M >> m And $V_M > V_m$	$v'_M = \frac{v_M(M-m)}{M+m}$	$v'_M = v_M$ $v'_m = 2\,v_M$
Scenario2	M = m And $V_M \neq V_m$	$v'_m = \frac{2M * v_M}{M+m}$	$v'_M = 0$ $v'_m = v_M$
Scenario3	M>m And $V_M<V_m$	$v'_M = v'_M$ $v'_m = v'_m$	$v'_M = v'_M$ $v'_m = v'_m$
Scenario4	M=m And $V_M=V_m$		$v'_M = v'_M$ $v'_m = v'_m$

Source: Authors, (2025).

$$p_1 + p_2 = p'_1 + p'_2 = cte \quad (1)$$

$$K_1 + K_2 = K'_1 + K'_2 = cte \quad (2)$$

When our two characters collided at two different velocities, v_M and v_m and two different masses, M for the heavier object and m for the lesser one, along with the sum of the two momenta, the sum of the two kinetic energies is conserved in elastic collisions.

The two conservation equations are written as follows:

$$\tfrac{1}{2}Mv_M^2 + \tfrac{1}{2}mv_m^2 = \tfrac{1}{2}Mv'^2_M + \tfrac{1}{2}mv'^2_m \quad (3)$$

$$Mv_M + mv_m = Mv'_M + mv'_m \quad (4)$$

Where the final velocities of both objects v'_M, and v'_m are given. The sums of the velocities before and after the collision for each object are equal when the two equations are rearranged with the velocities of one object on one side of Eq. (3) and Eq. (4) and those of the other on the opposite side, then dividing the rearranged Eq. (3) with the rearranged Eq. (4) [14].

$$v_M + v'_M = v_m + v'_m \quad (5)$$

The ultimate velocities following the collision are determined by solving the set of linear Eq. (4) and Eq. (5):

$$v'_M = \frac{(M-m)v_M + 2mv_m}{M+m} \quad (6)$$

$$v'_m = \frac{(m-M)v_M + 2Mv_M}{M+m} \quad (7)$$

Depending on the values of m and M for our colliding characters in Eq. (4) and Eq. (5), we disregard the smallest value and get the final velocities $v'_M$and v'_m (Table 1). We derive a valid method for collisions from Newton's Laws, the laws of

momentum, and kinetic energy conservation. It provides a way to determine the final velocities after detection collision using the Raycasting technique without external net forces, according to each character's initial masse and velocity (see Figure 5). In the second scenario, when the masses are equal, the final velocity of this character will be reduced to a small value for some seconds. At this time, the second character will change its direction and take the velocity of the first character to move. After steps of seconds, both of them will take their first velocity. To ensure that we examine all possible conditional cases that could occur between characters, we have added two additional cases based on the same previous data. In these two situations, the end velocity used to predict the occurrence of a collision is the same as the initial velocity of the two characters before the collision.

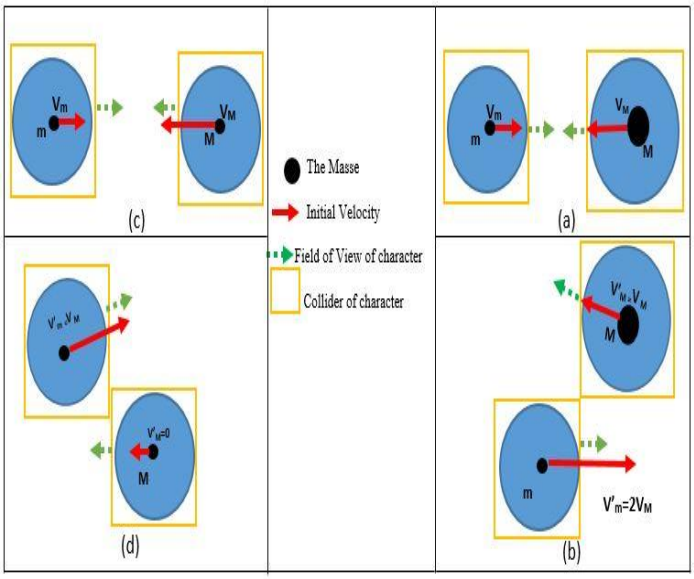

Figure 5: Examples of two cases of collision
Source: Authors, (2025).

Perception with no collision of the first Scenario. (b) Final velocities are determined according to the initial different velocities and masses of characters by Newton's Laws, as well as the concepts of momentum and kinetic energy conservation after detection collision by the Raycasting technique. (c) Perception with no collision of the second Scenario. (d) Final velocities are determined according to the different initial velocities and equal masses of characters by Newton's Laws, as well as the concepts of momentum and kinetic energy conservation after detection collision by the Raycasting technique.

Algorithm: Collision detection

Input: initial masse, direction, initial velocity, ray range, characters, character controller, positions, and orientations.

Output: final velocity, final direction and rotation. distance

If (collision occurs) **then**

- Calculate distance.

-If ($V_M > V_m$ and $M > m$) **then**

($v'_M = v_M$ and $v'_m = 2\,v_M$)

-Else if($V_M > V_m$ and $M = m$) **then**

($v'_M = 0$ and $v'_m = v_M$)

-Else if($V_m > V_M$ and $m > M$) **then**

($v'_m = v_m$ and $v'_M = 2\,v_m$)

-Else if($V_m > V_M$ and $m = M$) **then**

($v'_m = 0$ and $v'_M = v_m$)

-Else if($V_M < V_m$ and $M > m$) **then**

($v'_m = v_m$ and $v'_M = 2\,v_m$)

-Else if($V_m < V_M$ and $m > M$) **then**

($v'_m = v_m$ and $v'_M = v_M$)

-Else if($V_m = V_M$ and $m = M$) **then**

($v'_m = v_m$ and $v'_M = v_M$)

-End if

-Change direction.

-Change rotation.

End if

III. CROWD SYSTEM RESULTS AND DISCUSSION

Virtual humans simulated on a Windows 10 computer with an Intel i7 processor running at 2.80 GHz, an NVIDIA GeForce GTX 780 graphics card, and 8 GB of RAM. The pedestrian system utilizes shader files for rendering the models and C# scripts to simulate the behaviour of virtual individuals within Unity 3D, version 2018.3.2f1 (64-bit).

The crowd system enhances the visual experience by including diverse character movements, rendering the simulation more dynamic and realistic. These movements are specifically selected to ensure that the virtual humans behave naturally in various scenarios. Combining high-quality shaders, detailed character animations, and realistic behaviours allows a seamless and immersive simulation of crowded environments. This setup offers a strong platform for real-time simulation and visualization of extensive pedestrian systems, providing valuable insights for various applications such as crowd control and urban planning.

III.1 CROWD DENSITY AND ANIMATION VARIATION IN OPEN ENVIRONMENT

To ensure a variety of movements among the crowd, we set the models in an open environment where each model has specific motions and behaviours while avoiding collisions. The actions include idle poses, jumping, walking, strolling with a briefcase, feminine gait, an older adult's unique motion, walking with joy, performing rhythmic hip-hop dancing, walking while texting, and running. Each model can perform specific movements that vary along a predetermined path. The animations are set to a specific speed, ranging from 1 to 50 units per second.

Different Scenarios	Number of characters	Number of animations type applied	Range of movements speed applied	Frame per seconds(ms)
Scenario1	7	11	[1 m/s,50 m/s]	60
Scenario2	14	22	[1 m/s,50 m/s]	59.1
Scenario3	21	33	[1 m/s,50 m/s]	58.7
Scenario4	63	99	[1 m/s,50 m/s]	55

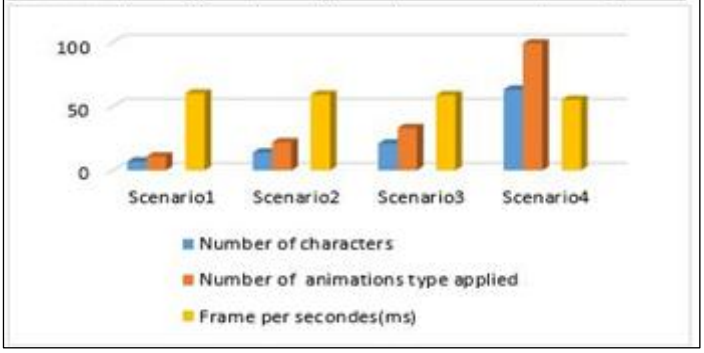

Figure 6: The performance evaluation of our approach across four different scenarios involved examining how adding various types of animations with different speed values affects the changes in FPS values.
Source: Authors, (2025).

These speed values are determined to achieve realism and visual attractiveness, enhancing overall movement dynamics within the specified parameters. As a result, each model's movement represents its unique characteristics and activities (see Figure 6).

In another approach, we manage character animations in scenarios where collisions occur between them (Figure 7). Each character is animated with various movements and speeds, allowing for realistic, interactive collisions. We use collision detection algorithms based on Newton's Laws to effectively manage these collisions, focusing on momentum and kinetic energy conservation.

This method accurately calculates the final velocities of characters after a collision by utilizing raycasting techniques. The algorithm considers each character's initial mass and velocity to compute the result, providing no external forces are applied. It ensures that character interactions behave realistically, with adjustments to their velocities reflecting the laws of physics. By incorporating these principles, we achieve a more immersive and believable simulation of collisions between characters. This approach is beneficial for creating dynamic environments where characters interact with one another in real time, such as in crowd simulations.

III.2 THE PERFORMANCE EVALUATION OF OUR ANIMATION VARIATION APPROACH BY EXPLORING THE MAXIMUM COMBINATIONS OF ELEMENTS

To ensure the precision of our animation results, we identify combinations of elements that increase diversity. We begin by selecting key factors such as each character's movement type and speed variations. The process calculates the possible ways to combine these factors at each time step. We can generate a wide range of character behaviours by identifying the maximum combinations for these elements across multiple scenarios.
This approach ensures diverse and realistic animations, allowing characters to interact in varied ways. It also enhances the simulation by accounting for different kinds of motion and speeds in dynamic environments, where:

- **H** : It represents the maximum number of characters our model utilises for simulation and animation.
- **T** : It represents the maximum number of animation types each character uses in the simulation or model.
- **V** : It represents the maximum number of speed values assigned to each character in the simulation.
- **Max_Combinations_Anim:** It represents the value of the maximum feasible combination of terms for each model, which contributes to the overall variety and diversity of animations within the simulation.

$$H \times T \times V = Max\_Combinations\_Anim \quad (3)$$

The maximum feasible combination of model terms that contribute to the variety of animations in the simulation:
$(18 \times 10 \times 50) \times 5 = 9000$.

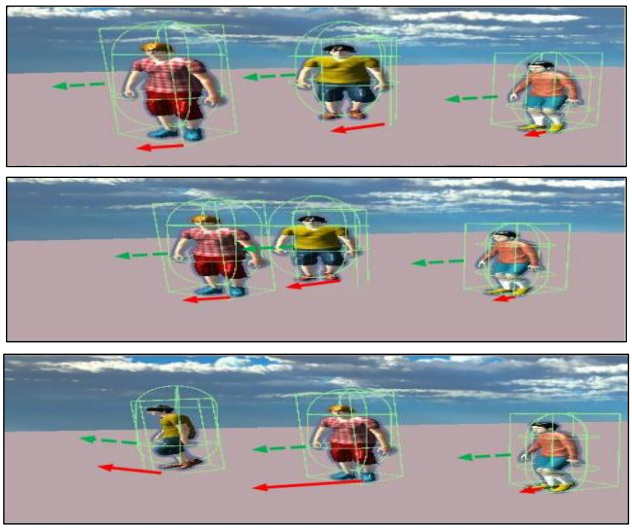

Figure 7: The scenario of crowd animation with collision detection: (a) Perception with collision: The green arrow represents the character's field of view, while the red arrow indicates the velocity before the collision. (b) Collision detection between two characters. (c) After the collision is detected using the Raycasting technique, the red arrow shows the final velocities, which are determined based on the characters' initial velocities and masses by Newton's Laws and the principles of momentum and kinetic energy conservation.
Source: Authors, (2025).

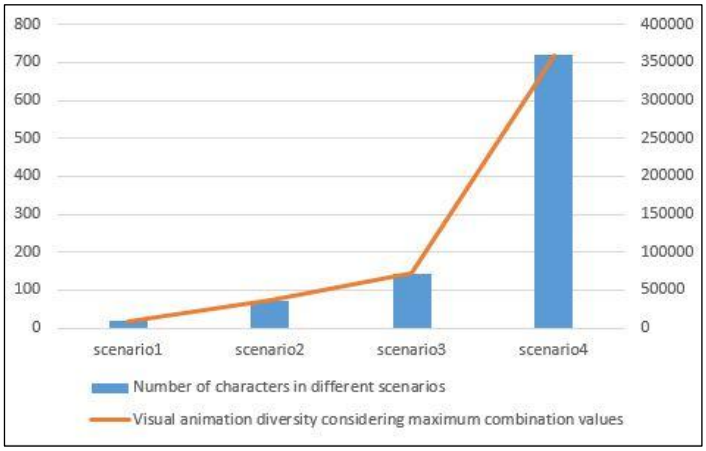

Figure 8: Exploring the maximum feasible combinations of elements in our diverse animations based on the number of characters involved.
Source: Authors, (2025).

We realized that our algorithm could generate various motion types by combining key factors such as the number of characters and the range of movement types at different speeds. As the number of characters in various scenarios increases, so does the potential for diverse motion combinations. For instance, using 1080 characters, our algorithm can produce approximately 540000 unique combinations. This feature significantly increases the range of feasible character movements, making the simulation more dynamic and varied. We can introduce additional character types and movement styles to enhance this variety further, resulting in more complex and visually realistic animations.

This expansion improves the realism of individual character interactions and significantly improves the overall quality of crowd simulations. By providing these combinations, we can create a more immersive and immersive experience, offering a wider range of diverse animations for large-scale crowd scenes and dynamic environments.

Table 2: The maximum combination values improve the visual animation diversity of various templates in different scenarios.

Number of characters in different scenarios	Visual animation diversity considering maximum combination values
Scenario1 ,H=18	9000
Scenario2 ,H=72	36000
Scenario3,H=144	72000
Scenario4,H=720	360000

Source: Authors, (2025).

III.3 EVALUATION OF THE PERFORMANCE OF POSSIBLE COMBINATIONS OF OUR APPROACH FOR CHANGING ANIMATIONS USING A PARTICULAR NUMBER OF ELEMENT SETS FOR DIFFERENT CHARACTERS.

The following process presents how to increase animation variation by selecting elements simultaneously. We determine the number of combinations for different terms across various contexts, considering factors like character types, movement styles, and speeds. This approach maximizes diversity in animations, allowing for more dynamic and varied character behaviours in different simulation scenarios.

$$C_H^h . C_T^t . C_V^v . = Possible\_\text{Animation\_Variety}$$

To begin, we assign specific labels to each item associated with an account, ensuring accurate identification and categorization for efficient processing and analysis.

• **H** represents the maximum number of characters while **h** denotes the number of characters under consideration in a given scenario.

• **T** represents the total number of animation types used by our models, while **t** specifies the specific type of animation applied to a particular model.

• **V** refers to the maximum height value of speed animation for our characters, representing the highest possible speed in the animation. Meanwhile**, v** denotes the specific speed value assigned to each character during their animation, allowing for variation in movement and enhancing the realism of character actions.

• **Possible_Animation_Variety** indicates the number of possible combinations of these different terms, such as character types, animations, and speeds that contribute to the overall variety and diversity of animations in our simulation or model.

Next, we determine the number of combinations in our scene based on the various terms defining the animation variety for our ten distinct models. To do this, we define the following variables:

H = 10 for the total number of characters, and **h** = 1 for the characters considered.

T = 10 for the total number of animation types utilized by our models, with **t** = 1, **t** = 4 specifying the types of animation used by the model during its animation.

V = 50 represents the height value of the speed animation for our characters, and **v** = 1 signifies the specific speed value assigned to each model.

Using these variables, we can calculate the total number of unique animation combinations within our scene. The combination of these elements (character types, animation styles, and speed values) results in a wide variety of possible motion sequences for each character. This allows us to generate a more dynamic and diverse simulation, ensuring that each character behaves in a unique and realistic way. The formula derived from these variables will provide the total number of animation possibilities for the crowd, enhancing realism and interaction within the simulation.
Possible combinations are:

$$(C_{10}^1 . (C_{10}^1 + C_{10}^4) . (C_{50}^1)) \times 18 = (10* \quad (10 \quad +210) \quad *(50)) \quad \times 18 = 1980000.$$

IV. OPTIMAZING REAL TIME PERFORMANCE

The reason for measuring the efficacy of crowd simulation depends on an accurate characterization of the aims of the crowd simulation tasks. Computer simulations of crowd behaviour may emphasize different performance metrics depending on the objectives of the application [15]. Some studies [16-18] propose simulating virtual crowd systems that are identical to real crowds. This enables them to clarify the operational dynamics of real crowds and identify potential risks within the system. Thus, their performance indicators reflect the accuracy of the simulation, its

similarity to actual crowd behaviour, and the behaviour of individual agents.

Enhancing our real-time model's efficiency requires reducing computational and rendering complexity in character animation. We achieve this by combining techniques such as object pooling, frustum culling, and occlusion culling, ensuring smooth performance even with several animated characters displayed simultaneously. It produces greater frame rates and less latency.

IV. 1 OBJECT POOLING FOR PERFORMANCE OPTIMIZATION

This method involves creating a collection of inactive characters, which are subsequently utilized as required. This method reduces memory allocations and enhances performance as animated characters increase. Rather than repeatedly creating and eliminating characters, which incurs significant computational costs, we utilize pre-existing objects, thus minimizing memory overhead and garbage collection, ultimately improving real-time performance.

Algorithm: pooling for performance optimization
Input: 3D character models, collection size, distance between characters, velocity range, mass range
Output: different animations, avoid collision, reduce memory allocation, optimize real-time

```
For each character in the allocation set do
        Instantiate character number
        Sets the character inactive
        Adjust character size and rotation.
        Incorporate characters into the collection list.
end for
For each character in the allocation set do
        Iterates through the collection
        Activate available character
        Give a unique random position
        Apply different animations at varying speeds.
        Avoid collision
end for
For each character stored in the collection list do
        Iterates through collection list
        Return the first inactive object
end for
```

IV. 2 FRUSTUM CULLING FOR PERFORMANCE OPTIMIZATION

This technique prevents the display of characters outside the camera's vision by specifying the camera's perspective, defined as the frustum. Only characters within this viewpoint are rendered, enhancing performance by conserving computational resources and reducing the GPU load. By verifying if the animated character is within the camera's field of vision, we can selectively deactivate or activate them, optimizing resource utilization and improving rendering efficiency.

IV. 3 OCCLUSION CULLING FOR PERFORMANCE OPTIMIZATION

This method determines if a character, although within the camera's field of vision, has been hidden or occluded by other objects. It prevents the engine from displaying characters hidden by different elements. Eliminating the rendering of hidden characters conserves computational resources. This approach increases FPS and improves performance by eliminating unnecessary rendering.

IV. 4 PERFORMANCE OPTIMIZATION OF HYBRID METHODS

Our method simulates multiple characters, each having distinct motions and velocities. We utilize Newton's Laws, laws of momentum, and the conservation of kinetic energy to prevent collisions between the models. It enables the computation of final velocities after collision detection using the Raycasting method, without external net forces, depending on each character's starting mass and velocity.

Then, we implement a hybrid approach that combines the pool object algorithm with frustum culling and occlusion culling approaches to enhance the frames per second during character movement. This technique improves real-time FPS performance, eliminates insignificant computations and rendering for characters unavailable to the camera, and decreases frequent memory allocation and deallocation.

By integrating these methods, we achieve significant results: our models perform at consistent frame rates, eliminate using resources on invisible characters and objects, minimize unnecessary frame computations, and prevent the rendering of occluded objects by analyzing the scene's geometry and identifying disabled characters (see figure 9).

Algorithm: object pooling and culling algorithms

for performance optimization

Input: 3D character models, collection size, distance between characters, velocity range,
mass range, primary camera reference, Fractional camera culling
Output: different animations, avoid collision, reduce memory allocation, optimize real time,
reducing the processing load

```
Algorithm pooling for performance optimization
For each active character in the collection set do
        Obtains renderer
        Checks character within frustum
        Detects hiding character
        If the object is visible and not occluded
        remains active
        else
        remains inactive
        end if
end for
Measure object-main camera distance.
Perform Raycast
Check if another object blocks the character.
Deactivate objects that are not visible
```

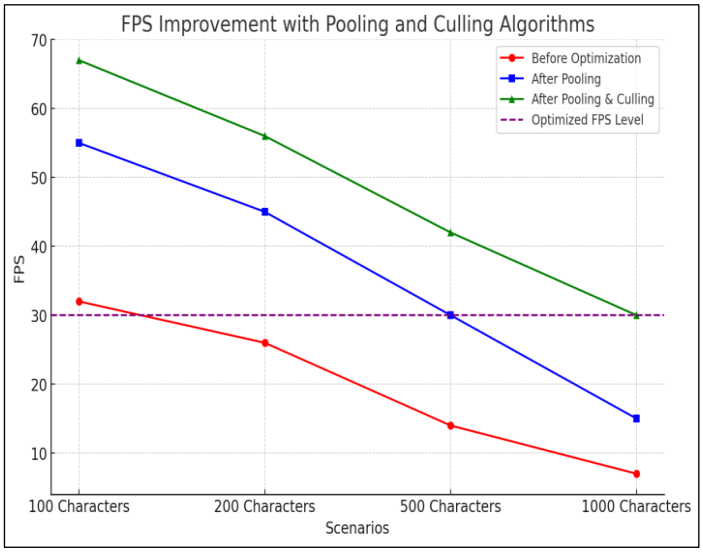

Figure 9: Improvement of FPS performance with pooling and culling optimisation algorithms.
Source: Authors, (2025).

V. CONCLUSIONS

We propose in this paper a method to enhance the diversity of crowd animation by integrating numerous animation styles and assigning each character distinct movements with varying velocities. This approach involves establishing unique animation cycles specific to each character model, facilitating a diverse array of movements during the simulation. Our method aim to obtain a more dynamic and realistic representation of crowd animation in virtual spaces while avoiding characters appearing too similar to each other. The study considers several human genders and age groups, ensuring that each character exhibits distinct movements.

Furthermore, we introduce an algorithm that utilizes Ray-casting for collision prevention between characters. Based on fundamental physics principles such as Newton's laws, momentum conservation, and kinetic energy conservation, the algorithm calculates the final velocities of characters during potential collisions, ensuring realistic and accurate results in crowd simulations. This results in more convincing and realistic crowd visualizations.

Additionally, we implement a hybrid approach that combines the object pooling algorithm with frustum culling and occlusion culling techniques. This method stabilizes frames per second in real-time and enhances performance, producing significant results by reducing the number of active objects processed. It ensures that only characters actively contributing to rendering and animation are considered, optimizing overall simulation efficiency.

As a prospective enhancement, we plan to improve our system by integrating crowd behavior principles to enhance simulation realism. Additionally, we aim to apply reinforcement learning for character animation and obstacle avoidance, enabling adaptive crowd navigation across diverse scenarios. This will facilitate effective autonomous navigation for multiple agents in more complex environments.

VII. REFERENCES

[1] Ruiz, S., Hernández, B., Alvarado, A., & Rudomín, I. (2013). Reducing memory requirements for diverse animated crowds. In Proceedings of Motion on Games (pp. 77-86).

[2] Beacco, A. (2014, October 16). Simulation, Animation and Rendering of Crowds in Real-Time. Simulation, Animation and Rendering of Crowds in Real-Time. https://upcommons.upc.edu/handle/2117/95561.

[3] Liu, P., Chao, Q., Huang, H., Wang, Q., Zhao, Z., Peng, Q., ... & Jin, X. (2022). Velocity-based dynamic crowd simulation by data-driven optimization. The Visual Computer, 38(9), 3499-3512.

[4] Toledo, L., De Gyves, O., & Rudomín, I. (2014). Hierarchical level of detail for varied animated crowds. The Visual Computer, 30, 949-961.

[5] Boukhayma, A., & Boyer, E. (2017, October). Controllable variation synthesis for surface motion capture. In 2017 International Conference on 3D Vision (3DV) (pp. 309-317). IEEE.

[6] Hu, K., Haworth, B., Berseth, G., Pavlovic, V., Faloutsos, P., & Kapadia, M. (2021). Heterogeneous crowd simulation using parametric reinforcement learning. IEEE Transactions on Visualization and Computer Graphics.

[7] Wang, D., Zhang, J., Jin, J., & Mao, X. (2021). Local collision avoidance algorithm for a unmanned surface vehicle based on steering maneuver considering colregs. Ieee Access, 9, 49233-49248.

[8] Yu, Q., Liu, K., Yang, Z., Wang, H., and Yang, Z. (2021). Geometrical risk evaluation of the collisions between ships and offshore installations using rulebased Bayesian reasoning.Reliability Eng. System Saf.210, 107474. doi:10.1016/ j.ress.2021.107474

[9] Tam, C., Bucknall, R., and Greig, A. (2009). Review of collision avoidance and path planning methods for ships in close range encounters.J. Navigation 62 (3), 455–476. doi:10.1017/S0373463308005134.

[10] Wang, C., Wang, N., Xie, G., and Su, S. F. (2022).“Survey on collision-avoidance navigation of maritime autonomous surface ships,”inOffshore robotics(Singapore: Springer), 1–33. doi:10.1007/978-981-16-2078-2_1

[11] Li, Y., Chen, Y., Liu, J., & Huang, T. (2025). Efficient crowd simulation in complex environment using deep reinforcement learning. Scientific Reports, 15(1), 5403.

[12] Guo, X., Zhang, M., Xie, H., Gu, C., & Liu, Z. (2024). Crowdmogen: Zero-shot text-driven collective motion generation. arXiv preprint arXiv:2407.06188.

[13] Li, Y., Liu, J., Guan, X., Hou, H., & Huang, T. (2025). Introducing anisotropic fields for enhanced diversity in crowd simulation. The Visual Computer, 1-16.

[14] Hernández, A. G. Y., & Alberú, M. del P. S. (2021). Modeling the Interaction of an Elastic Collision between two Objects. Journal of Physics: Conference Series, 1929(1), 012016. https://doi.org/10.1088/1742-6596/1929/1/012016

[15] Li, Y., Chen, Y., Liu, J., & Huang, T. (2025). Efficient crowd simulation in complex environment using deep reinforcement learning. Scientific Reports, 15(1), 5403.

[16] Rempe, D., Luo, Z., Bin Peng, X., Yuan, Y., Kitani, K., Kreis, K., ... & Litany, O. (2023). Trace and pace: Controllable pedestrian animation via guided trajectory diffusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13756-13766).

[17] Colas, A., van Toll, W., Zibrek, K., Hoyet, L., Olivier, A. H., & Pettré, J. (2022, May). Interaction fields: Intuitive sketch-based steering behaviors for crowd simulation. In Computer Graphics Forum (Vol. 41, No. 2, pp. 521-534).

[18] Panayiotou, A., Kyriakou, T., Lemonari, M., Chrysanthou, Y., & Charalambous, P. (2022, July). Ccp: Configurable crowd profiles. In ACM SIGGRAPH 2022 conference proceedings (pp. 1-10).