Multi-agent 3D Trajectory Deconfliction With Obstacle Avoidance

Michael Ayomide Ogunrinde

Department of Computer Science, University of Cambridge

Reasoning

Trajectory deconfliction in 2D environments has been well-studied, but real-world scenarios often require navigation in 3D space with obstacles. This project extends an existing 2D solution to a 3D environment, incorporating obstacle avoidance. By addressing these complexities, we aim to develop more robust and practical solutions for applications such as drone traffic management and autonomous vehicle navigation.

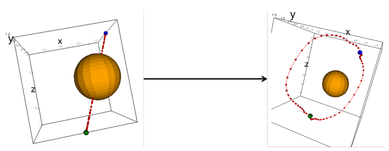


Figure 1. Before solved and After solved.

Objectives

- **Extend** the existing 2D trajectory deconfliction algorithm to operate in a 3D space
- **Maintain/Improve** the confliction resolution performance of the original 2D model
- **Implement** obstacle avoidance capabilities within the 3D environment
- **Evaluate** the scalability and efficiency of the 3D solution compared to its 2D counterpart

Methodology

Data Generation

Scenario Type	Probability	Obstacle Type	Placement Strategy	Quantity
Potentially Conflicting	70%	Trajectory-Based	Along agent trajectories Position = initial_pos + α(final_pos - initial_pos)α ∈ [0.1, 0.9]	N_Trajectory = floor(0.8*N_total)
		Random(Biased)	Normally distributed around (0,0,0) with σ = 5	N_random = N_total - N_trajectory
Conflict Free	30%	Random	Uniformly distributed within environment bounds	N_total

Figure 2. Dataset Sample

CNN Architecture

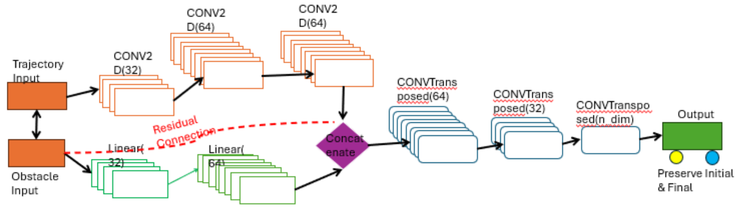


Figure 3. CNN Model

$$Loss = MSE(pred, target) + w^c * L_c + w^o * L_o$$

Where L_c = colision loss, L_0 = obstacle avoidance, w^c = collision weight, w^o = obstacle weight

(a) Custom Loss Function

$$x_{norm} = (x - \mu)/\sigma$$

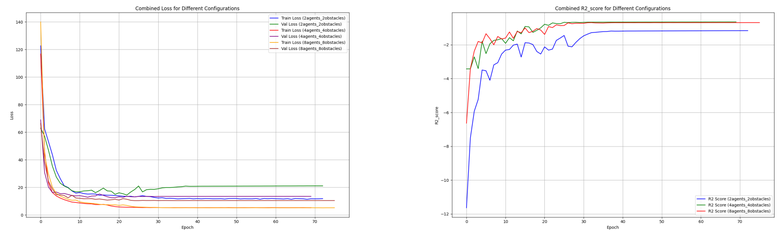
RAW → x_{norm} → *Ouput*

(b) Normalization Process

Figure 4. Loss Function and Normalization

Results

For each set of results, we used 1,000 samples: 100 for validation, 100 for testing, and 800 for training, consistent with the 2D implementation. However, we increased the number of epochs from 100-150 to 200-250. This adjustment was made because the prior tests were conducted with 20,000 samples, whereas our dataset consists of only 1,000 samples.



(a) Val/Train Loss (b) R2 Score

Figure 5. Comparison of Loss and R2 Score

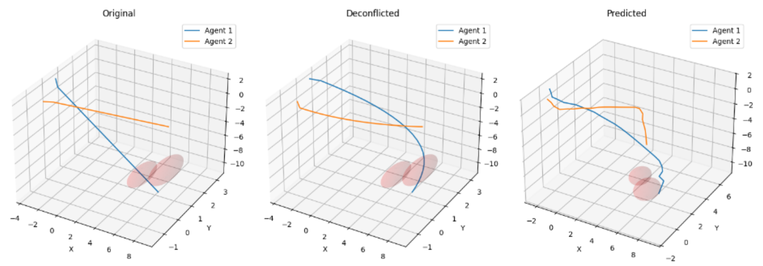


Figure 6. Comparison between the trajectories

	Samples	Before Model	After Model	Percentage
Agent - Agent	2a2obs	15	0	100%
	4a4obs	80	4	95%
	8a8obs	344	22	93.6%
Agent - Obstacles	2a2obs	411	58	86%
	4a4obs	871	229	73.7%
	8a8obs	1777	742	58.2%
Total	-	3498	1055	70%

Table 1. Comparison of Results Before and After the Model

Comparison: 2D vs 3D

Deconfliction Benchmarks			
Model Name	Conflicts before Model	Conflicts after Model	Reduction
mlp_2a_16p	25,913	2,260	91.28%
gan_2a_16p	25,913	3	99.99%
mlp_4a_16p	40,342	13,206	67.26%
gan_4a_16p	40,342	15,219	62.28%
mlp_8a_16p	47,708	28,003	41.30%
gan_8a_16p	47,708	18,530	61.16%

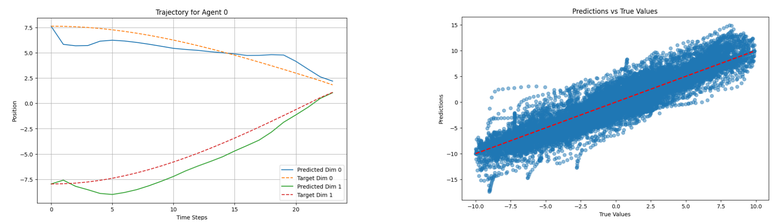
Figure 7. 2D results

When we closely examine the "mlp_8a_16p" model in Figure 7, we see that it was able to resolve only 41.3% of conflicts compared to our model, which achieved a conflict resolution rate of 58.2%. Although this difference might not seem significant at first glance, it's important to note that our model was trained on a dataset of just 1000 samples, demonstrating its efficiency despite limited data.

Challenges and solution

1. **Loss Function Complexity**: The current loss function balances multiple objectives (MSE, collision avoidance, obstacle avoidance, initial and final position constraints). Finding the right balance between these components is challenging. A potential solution is to implement adaptive weighting of loss components during training, allowing the model to focus on different aspects at different stages of learning.
2. **Long-term Trajectory Planning**: The current approach might struggle with long-term dependencies in trajectories. Incorporating recurrent elements or transformer-based architectures could help the model better capture and utilize long-range temporal information in trajectory planning.
3. **Computational Efficiency**: As the number of agents and obstacles increases, the computational complexity grows significantly. Implementing more efficient algorithms for collision detection or using approximate methods for less critical calculations could improve scalability.

Additional Results



(a) Solvers Trajectory vs Modal Trajectory (b) Solver Points vs Model Points

Figure 8. Agents Trajectory and Points

The results, including those shown in Figure 6, indicate that while the model is effective, it is not yet perfect and still has room for improvement. Specifically, the model performs more consistently for values near the center of the range but exhibits increased variance at the extremes. This suggests that although the model has grasped the general relationship, it faces challenges with precision, particularly for outlier cases. ***8agents8obstacles?***

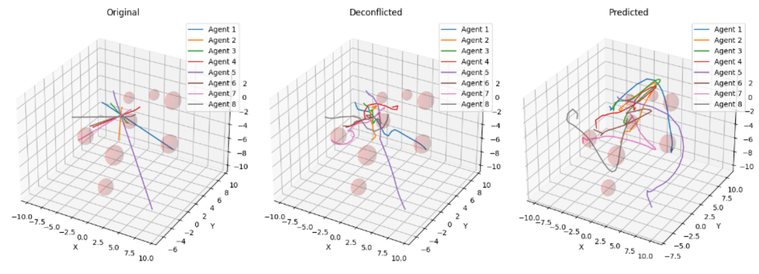


Figure 9. 8 Agents and 8 Obstacles

Conclusion

I can confirm that this project was a success. The results exceeded our expectations compared to the 2D model, despite the added complexity of an additional dimension. The performance of our CNN in obstacle avoidance is quite promising and represents a solid foundation. Looking ahead, with the integration of a GNN, we anticipate at least a 30% improvement in performance.

Future Work

- Increasing dataset size and diversity
- Exploring more complex obstacle scenario's
- Potential real world applications (e.g., drone traffic management).