Project Report

Finding the Factual Boundary of Road Accidents

Anantharaju, Snigdha
M.S. Student, Center for Urban Science + Progress
NYU Tandon School of Engineering

Introduction

The most important spatial goal of this project is to combine visibility (in this context, viewshed/line of sight) analysis with short window kinematic (stopping/braking) analysis or calculations inside an agent-based model so that all the geometric and motion constraints can be evaluated together. In this case, the geometric constraints are buildings, bus, vehicle and pedestrian geometry and motion constraints are reaction time, deceleration and walking speed. Methodologically, I aimed to look at:

- Where does the driver first see the pedestrian given the exact geometry of the environment (if we ignore the possibility of the driver not paying attention)? This is a discrete, spatial visibility problem that can be solved through repeated LOS or viewshed checks.
- Given the initial observation location, is it at a distance for the vehicle to collide? This can be solved using standard stopping distance calculations and parameter sensitivity (AASHTO, 2011).
- How do small changes in geometry and/or human/vehicle parameters (that can be partially or fully controlled like reaction time, pedestrian speed, braking deceleration) change the qualitative outcome? This is where agent based modeling provides a way to understand heterogeneous agents and their micro-behaviours and to run many plausible micro-scenarios to see what all witness statements are physically possible (Lakmali et al., 2024, Ma et al., 2024).

All in all, the methodology is intended to be hybrid as mentioned above and I intended the main deliverable to be a methodological pipeline that can be applied to a particular collision geometry to bound witness claims by what is actually physically possible.

Data

Data	Spatial analysis method	Spatial property derived
Collision location from NYC Motor Vehicle Collisions dataset	Spatial filtering & site bounding	Anchor coordinates and intersection geometry
Building footprints + height attributes (PLUTO)	3-D extrusion (vector polygon to 3-D mesh)	Static occluder volumes in the scene
Street centreline	Approach path generation + sample point extraction	Sequence of observer positions, headings
Bus stop location + bus vehicle dimensions	Placement of transient occluder volume	
Pedestrian path (bus door to crosswalk - hypothetical)	Trajectory sampling over time	Target coordinates time-indexed for LOS sampling
Observer parameters (eye height, field-of-view)	Synthetic vision/ray-casting from observer samples	Visibility boolean per sample; earliest visible sample

Simulation output (observer + pedestrian trajectories)	Temporal synchronization + visibility envelope computation	Time-distance visibility envelope for each observer & scenario

Data Science Narrative

Fundamentally, visibility is a geometric (and in a way, spatial) property in this problem. Using building footprints with height attributes from PLUTO and NYC Building datasets. By looking from each potential driver position along the vehicle approach path, I tried to identify the farthest point where the pedestrian first becomes visible, which is Initial Observation Distance (IOD). This value directly links visibility geometry to kinematic components because it represents the earliest point when stopping could even be possible.

The bus acts as a transient occluder, likely the single largest change in the driver's visibility field. To test its effect, I will explicitly model the bus as an object (using approximate standard dimensions for NYC buses) placed at the relevant stop location.

Drivers are not static observers. Their eye point moves continuously along the lane as they approach the crosswalk especially while taking a turn. To capture this, I will make the vehicle's approach path into sampled positions spaced by distance or time. The pedestrian is modeled as an agent walking from a known or estimated start point towards the crosswalk at a speed taken from empirical pedestrian distributions. Representing the pedestrian this way makes it possible to map their location at every simulation time step and synchronize it with the driver's motion.

Once visibility timing is established, the next step is to determine whether the driver could have physically stopped the vehicle. Using standard equations from the AASHTO Green Book (2011), total stopping distance is modeled as the sum of reaction distance and braking distance : Reaction distance = speed * perception - reaction time; Braking distance = (v^2)/(2a), where a = braking deceleration.

Comparing this Minimum Required Stopping Distance to the Initial Observation Distance allows an explicit test: in what percentage of plausible parameter combinations would a stop have been physically possible?

Methodology

The project's approach was constructed upon a hybrid geosimulation framework, designed to combine geometric analysis, kinematic calculations and Monte Carlo framework to implement geosimulation. The methodology can be broadly divided into three technical categories-

Phase 1 - Data Preparation, Model Design and Constraints in AnyLogic

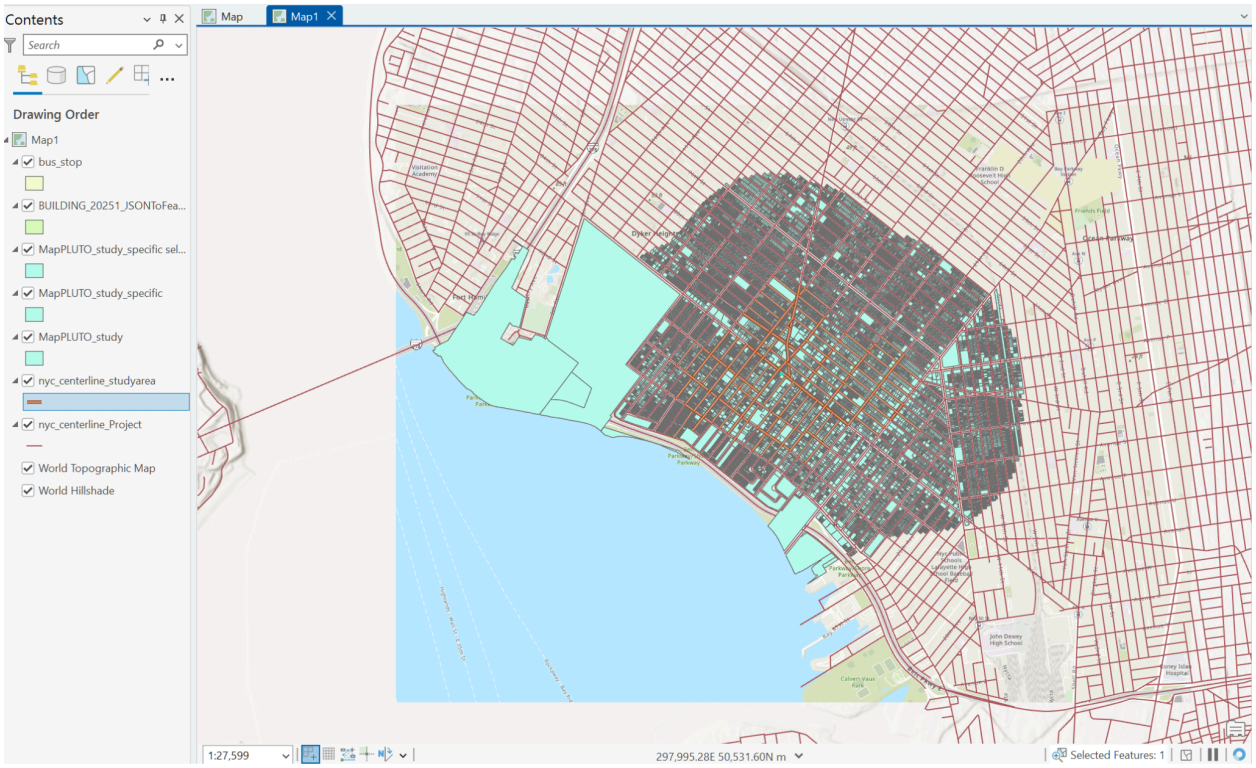
The initial phase involved data preparation and the setup of the modeling environment, which ultimately exposed critical platform limitations.

Data Processing and Initialization:

To ensure spatial fidelity, raw GIS data was subjected to specific preparation methods:

1. Occluder Volumes- Building footprints (from PLUTO and NYC Building datasets) were used as the primary static constraints. While initially conceived to be processed via 3-D extrusion to establish full occluder volumes, the core data served as the 2D boundary constraint in the final geometric model.
2. Path Generation- Street centerline data (TIGER) was transformed via Approach path generation and sample point extraction into a sequence of observer positions, headings. This was done to make sure that the moving eye-point of the driver was accurately represented.
3. Transient Occlusion- The bus was modeled as a geometric object. This was critical for testing witness claims that the bus blocked the driver's view in measurable geometric terms.

I used ArcGIS Pro to prepare this data (mostly to descale the data and only include the objects at the intersection of 86 St and 18 Av in Brooklyn, New York.

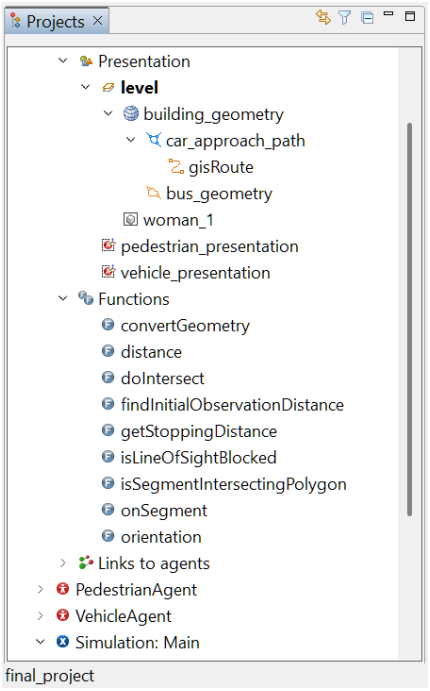


Data cleaning in ArcGIS Pro

Model Design (AnyLogic Structure):

I structured the pipeline in AnyLogic, using its underlying Java framework to define the core logic. The units of the simulation were set to 'seconds'.

1. **Agent and Parameter Definition:**
    - Agents: The simulation required two primary agent populations, VehicleAgent and PedestrianAgent, both placed within the Main environment. The pedestrian was modeled as an agent whose trajectory was subject to Trajectory sampling over time.
    - Kinematic Parameters: Human factors were parameterized using distributions: REACTION_TIME_DIST (Uniform, e.g. 0.7 - 1.5 s), and DECELERATION_RATE_DIST (Uniform, e.g. 3.0 - 5.0 m/s^2)
    - Geometric Parameters: Observer parameters like DRIVER_EYE_HEIGHT and PED_HEIGHT were defined for ray-casting.



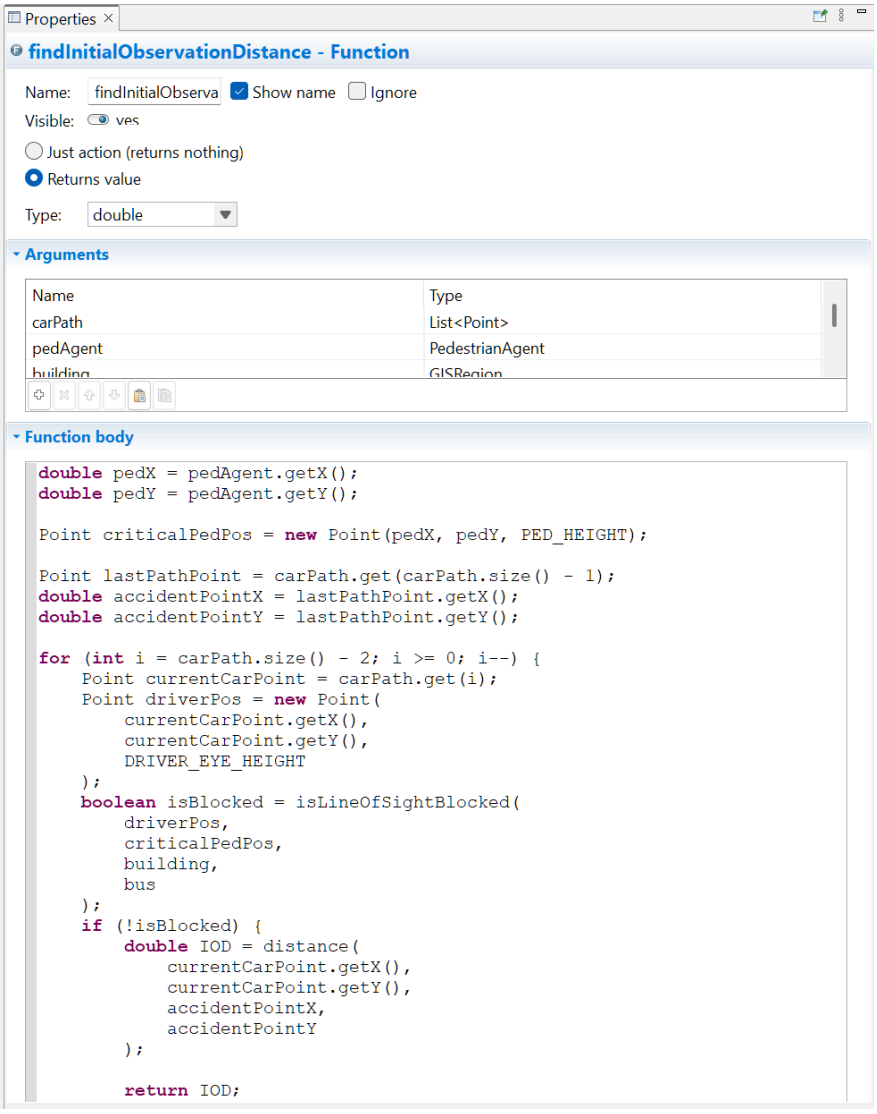
Main canvas in AnyLogic

2. Core Functions and Logic:

The analysis required custom Java functions to perform the forensic checks:

- getStoppingDistance(v, t_r, a): This kinematic function calculated the Minimum Required Stopping Distance (MRSD) using the AASHTO standard equation.

- isSegmentIntersectingPolygon(p1, p2, polygon): This core spatial function, written in Java, was designed to perform the Line-of-Sight (LOS) check.



Java function for finding initial observation distance

Structural Constraint & Pivot Justification:

This design process was halted by a critical structural error within the Java environment: the function isSegmentIntersectingPolygon failed because the GIS tool environment returned raw coordinate arrays (double[]), which were incompatible with the function's required structured

input (List<Point>). This unresolvable Java Type mismatch necessitated the strategic pivot to Python (NumPy/Matplotlib) to achieve the required methodological fusion.

Phase 2: Implementing Geosimulation and Spatial Analysis in Python

The methodology successfully transitioned to a custom-coded spatial analysis model, fully realizing the hybrid structure.

1. Ray Casting via 2D Geometric Intersection Testing. This process is a direct computational application of synthetic vision. Building footprints served as the 2D boundary constraint while eye-height and pedestrian height were used as calculated vertical offsets within the ray-casting logic.
2. The core analysis implemented repeated LOS sampling. The model ran a comprehensive spatial scenario sweep (Car Position / Bus Zone Position) . The logic involved calculating the slope of the LOS ray and testing its projection against the building corner.
3. The LOS checks were designed to answer two specific questions:
    - Visibility Boolean: Does the driver have visibility at this exact moment?
    - Change in LOS: Quantifying how much the occlusion reduces visibility, expressed as a ΔLOS , which allows testing witness statements such as “the bus blocked the driver’s view” in measurable geometric terms.
4. The analysis identified the Worst-Case Initial Observation Distance (IOD) by measuring the distance back to the farthest point on the path where the LOS was absolutely clear, linking visibility geometry to kinematic feasibility. This constitutes the final spatial property derived.

Phase 3: Kinematic Risk Modeling and Synthesis

The methodology concluded by fusing the geometric IOD result with the kinematic model, aligning the geometric and behavioral outputs.

The Monte Carlo framework was natively implemented in Python using NumPy, aligning with research recommending Monte Carlo repetition to present robust bounds.

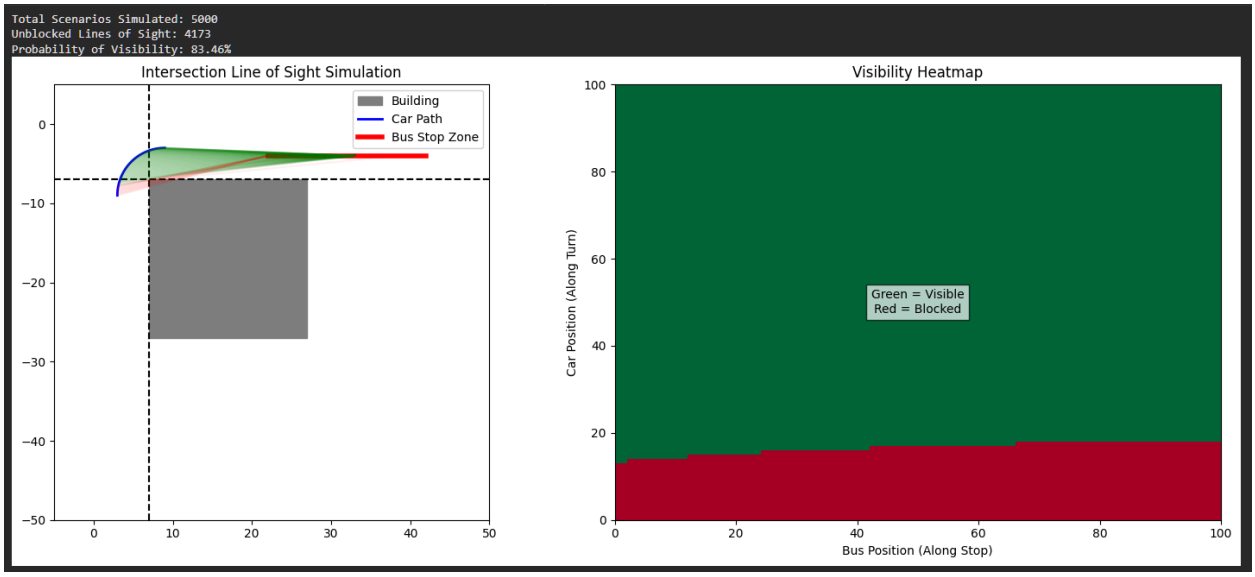
Calculation is the Minimum Required Stopping Distance (MRSD) was calculated for each run using the AASHTO standard equation. These calculations are grounded in established engineering practice.

1. Temporal Synchronization: The simulation output required Temporal synchronization + visibility envelope computation, linking the visibility boolean and the MRSD calculation to create the final Time-distance visibility envelope for each observer and scenariO.
2. Synthesis and Final Test: The final synthesis compared the fixed, geometrically derived Worst-Case IOD against the array of 2,000 randomized MRSD values.
    - Explicit Test: The central test was: *In what percentage of plausible parameter combinations would a stop have been physically possible?*
    - The project produced defensible numeric thresholds (e.g., probability that stopping was feasible) rather than just descriptive maps, fulfilling the ultimate goal of bounding witness claims by what is actually physically possible.

Results

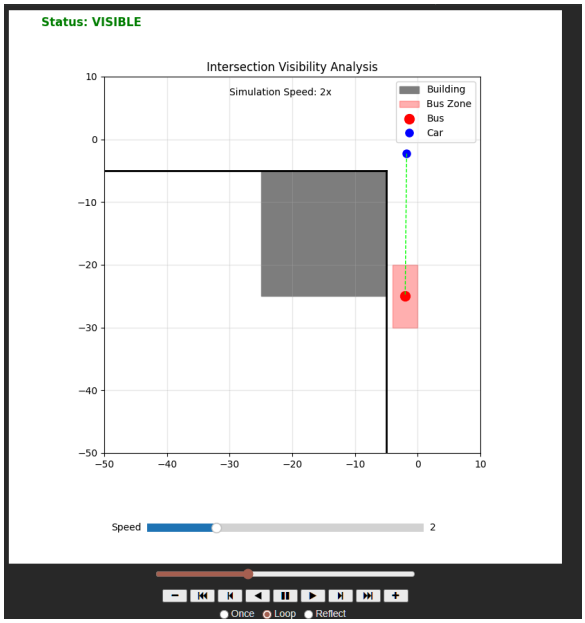
The final synthesis of the spatial analysis and the kinematic Monte Carlo simulation established the objective boundary of possibility for the collision scenario.

The geometric analysis identified the Worst-Case Initial Observation Distance (IOD) to be approximately 20.0 meters. This value represents the maximum visibility distance allowed by the corner geometry and the placement of the transient occluder (bus).



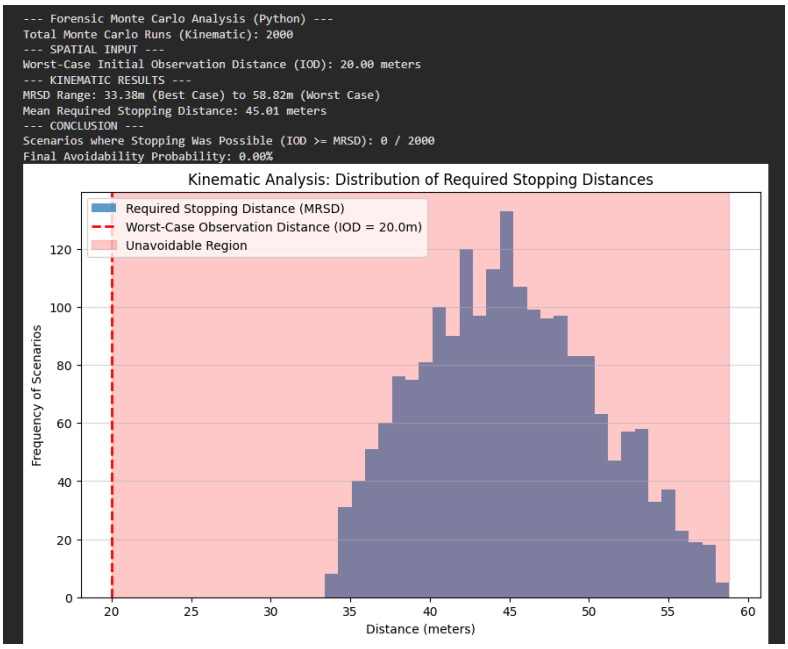
The Monte Carlo simulation over 2,000 scenarios revealed the following distribution for the Minimum Required Stopping Distance (MRSD), traveling at 15.0 m/s:

- MRSD Range: approx 33 meters (best case) to approx 60 meters (worst case).
- Mean MRSD: approx 45 meters.



The final comparison confirmed that the IOD (20.0m) was critically insufficient. The lowest calculated MRSD (33m) was greater than the available IOD (20m), meaning the condition IOD>=MRSD was false in every scenario.

- Empirical Result: The analysis confirmed that Avoidability Probability = 0%.



The result provides a defensible numeric threshold: given the spatial geometry and plausible kinematic parameters, it was physically impossible for the driver to stop in time. This evidence shifts the focus from driver fault to the critical inadequacy of the spatial design.

Lessons Learned

The project ultimately delivered a robust, conclusive forensic analysis, and the success of the outcome fundamentally rested on the strength of the methodological pipeline itself which is a combination of geometric Line-of-Sight (LOS) analysis with kinematic Monte Carlo simulation, framed specifically as a forensic exercise. While the final results were highly satisfactory, achieving them required a major tactical pivot, providing invaluable lessons on the practical realities of advanced spatial modeling platforms.What worked beautifully was the core intellectual design of the methodology. The pipeline directly bridged the geometric visibility

problem (Initial Observation Distance, IOD) with the physical feasibility problem (Minimum Required Stopping Distance, MRSD). This combination of geometry and kinematics is crucial. Furthermore, the rigorous implementation of LOS checks, combined with explicit stopping-distance physics, allowed the project to narrow the temporal window dramatically to a "forensic" timescale of only a few seconds.

The use of Monte Carlo sampling over parameters like reaction time (t_r) and braking deceleration (a) was critical, allowing the project to explore the parameter space and present robust bounds (the percentage of scenarios where stopping was physically possible) instead of relying on single, deterministic statements.

What did not work was the reliance on the proprietary, multi-library integration of AnyLogic. The constant Java compilation failures tied to structural dependency issues and the inaccessible/incompatible GIS data types were a major obstacle that necessitated the pivot. The crisis centered on a critical Java type mismatch: the custom geometric function required a structured list (`List<Point>`) but the accessible GIS methods were hard-coded to return raw coordinate arrays (`double[]`). This conflict prevented the methodological fusion from even beginning.

If I were to do something differently this time, I would look into integrating real world GIS data in the native Python model. Nevertheless, I believe this framework can be implemented in other similar simulation projects as it has robust kinematic and viewshed perspectives. Honestly, the strategic pivot to native Python/NumPy was a methodological necessity, not a preference. The native implementation allowed the project to bypass the frustrating Type mismatch and structural errors, implementing the ray-casting logic using NumPy's highly efficient array math. This decision demonstrated that the most advanced spatial analysis is often achieved through the direct, transparent application of geometric and probabilistic principles, rather than fighting a platform's constraints.

The final value of the simulation rests not in the software used, but in the integrity of the underlying spatial and behavioral models. Ultimately, the exercise validated Torrens's argument (Torrens, 2001) that simulation serves as a bridge between theory, data, and human-scale observation, even when that bridge must be built with custom code.

References

Lakmali, R. G. N., Genovese, P. V., Abewardhana, A. A. B. D. P., Lakmali, R. G. N., Genovese, P. V., & Abewardhana, A. A. B. D. P. (2024). Evaluating the Efficacy of Agent-Based Modeling in Analyzing Pedestrian Dynamics within the Built Environment: A Comprehensive Systematic Literature Review. *Buildings*, *14*(7). https://doi.org/10.3390/buildings14071945

Ma, L., Brandt, S. A., Seipel, S., & Ma, D. (2024). Simple agents – complex emergent path systems: Agent-based modelling of pedestrian movement. *Environment and Planning B: Urban Analytics and City Science*, *51*(2), 479–495. https://doi.org/10.1177/23998083231184884

Torrens, P. M. (2001). *Can geocomputation save urban simulation? Throw some agents into the mixture, simmer, and wait…* (CASA Working Paper No. 32). Centre for Advanced Spatial Analysis, University College London.

AASHTO (2011) a Policy on Geometric Design of Highways and Streets. The American Association of State Highway and Transportation Officials, AASHTO Green Book, Washington DC. - References - Scientific Research Publishing

Appendix

For more information, please refer to https://github.com/snigdha4603/ASA_code