

Supplementary Materials: Figures and Tables Review

Pre-Occupancy Crowd Dynamics Analysis for Mondial College

Bram Leisink Myrthe Sas
Mondial College, Nijmegen, The Netherlands

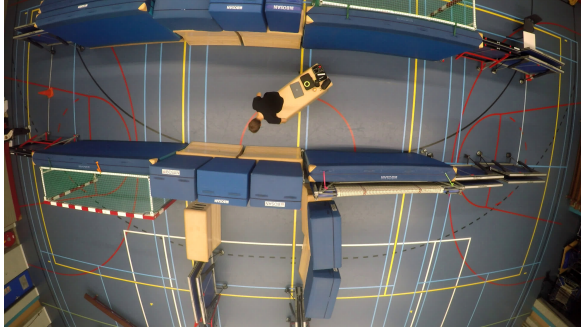
EUCYS 2026

Contents

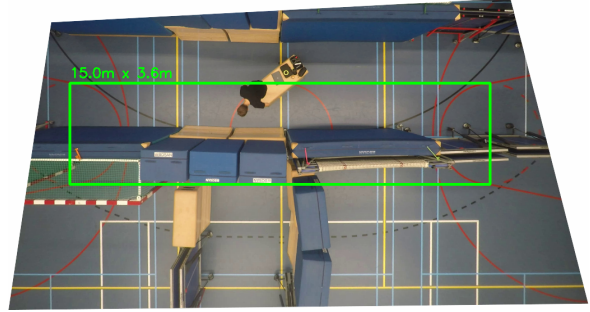
1	Phase 1: Controlled Behavioural Experiment	2
2	Phase 2: Topological Graph Analysis & Timetable Routing	4
3	Phase 3: Microscopic Pedestrian Simulation Pipeline	8

1 Phase 1: Controlled Behavioural Experiment

This section compiles the theoretical mathematical framework, sensor calibration pipelines, and empirical data distributions extracted from the live-scale tracking experiments.



(a) Original uncorrected tracking frame



(b) Corrected frame following camera calibration

Figure A.1: Visual documentation of the OpenCV lens undistortion pipeline. Planar barrel distortion is mathematically flattened to secure linear spatial coordinates.



Figure A.2: Isolated high-angle video frame extracted from the experimental trials, annotated with observed pedestrian tracking vectors and walking directions.

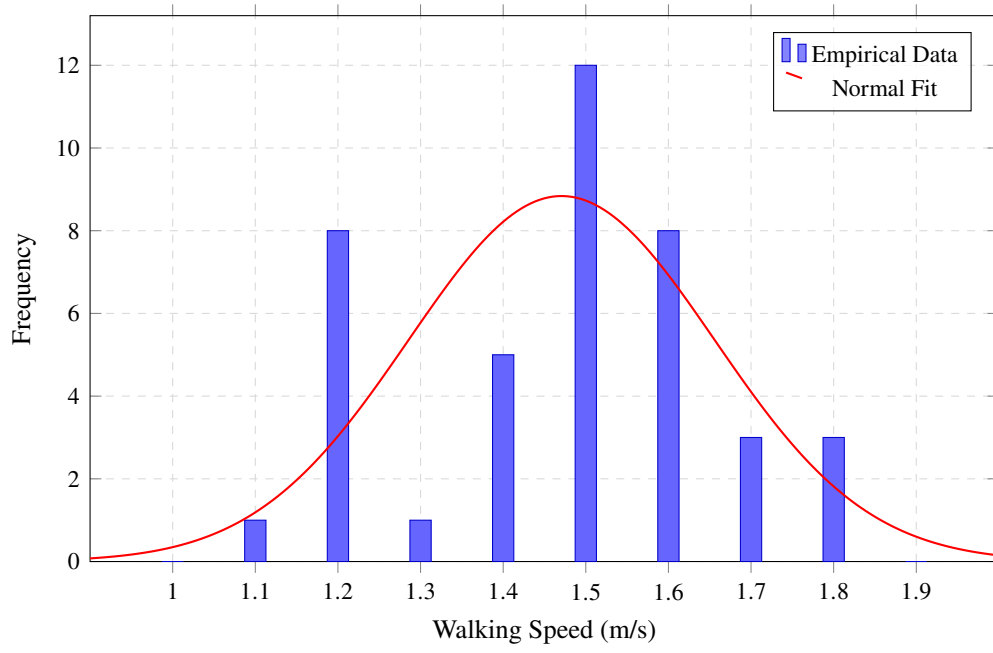


Figure A.3: Empirical free-flow walking velocity distribution calculated from manual tracking sequences ($N = 41$). The normal distribution profile ($\mu = 1.471$ m/s, $\sigma = 0.1851$ m/s) serves as a calibrated baseline input.

2 Phase 2: Topological Graph Analysis & Timetable Routing

This section presents the software configuration pipelines, raw structural infrastructure bottlenecks, network traffic load summaries, and multi-objective room allocation heuristics.

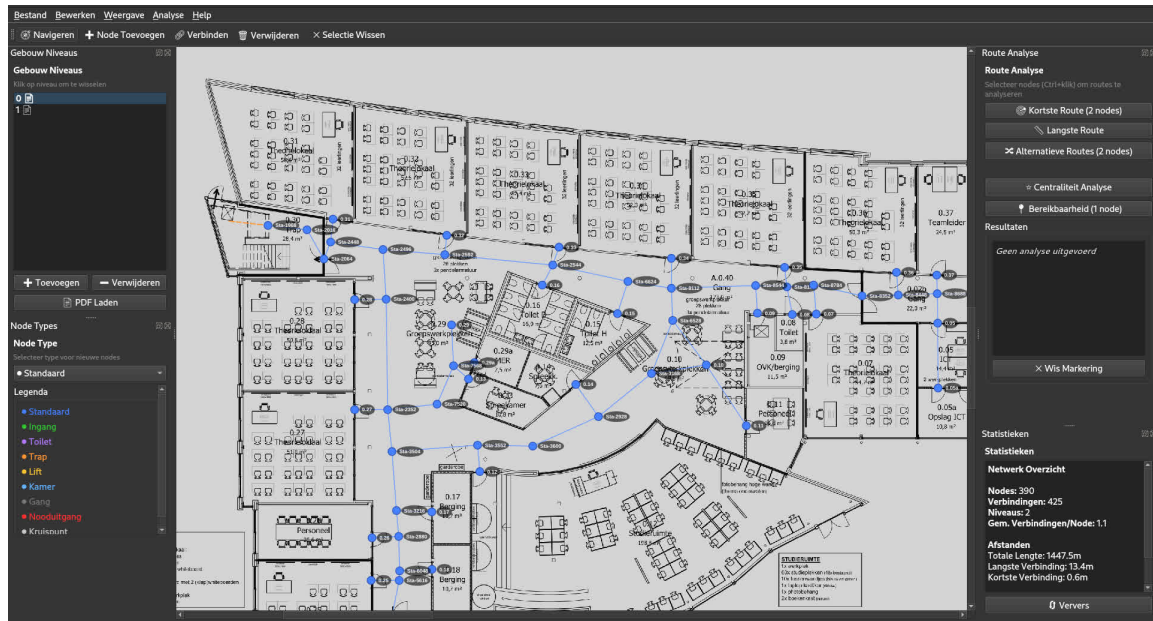


Figure A.4: User interface of the custom-developed Python graph-modeling application. The graphic demonstrates layout parsing, node registration, and relational edge construction across a ground floor sector.



(a) Standard peripheral stairwell profile.



(b) Central funnel-shaped staircase configuration.

Figure A.5: Physical structural elements of the relocation facility. While standard stairwells (a) maintain linear dimensions, the central architecture (b) funnels multi-directional occupant load onto an isolated, narrow landing upper-bound.

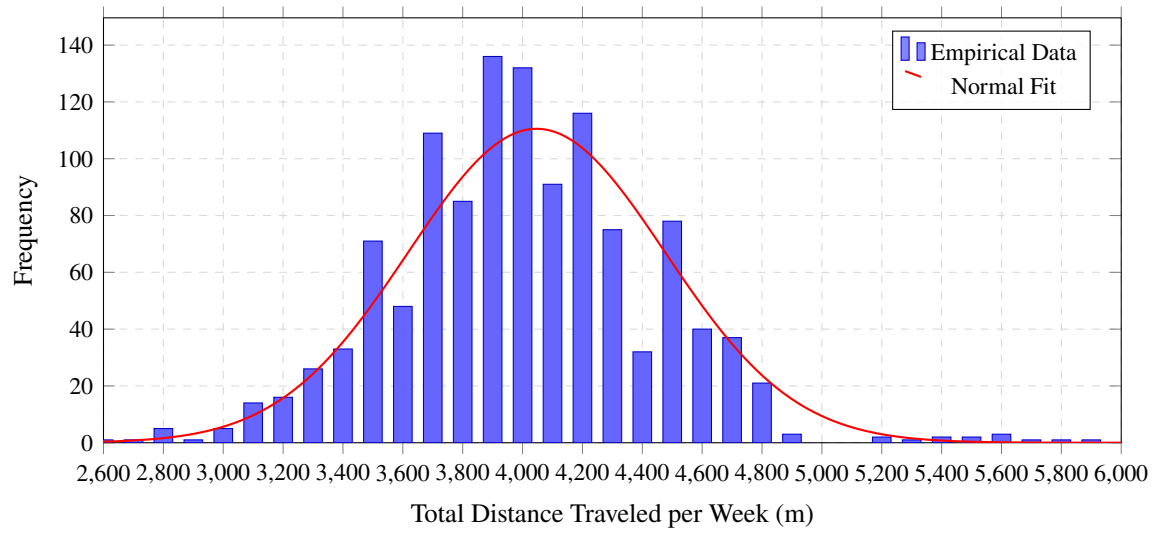


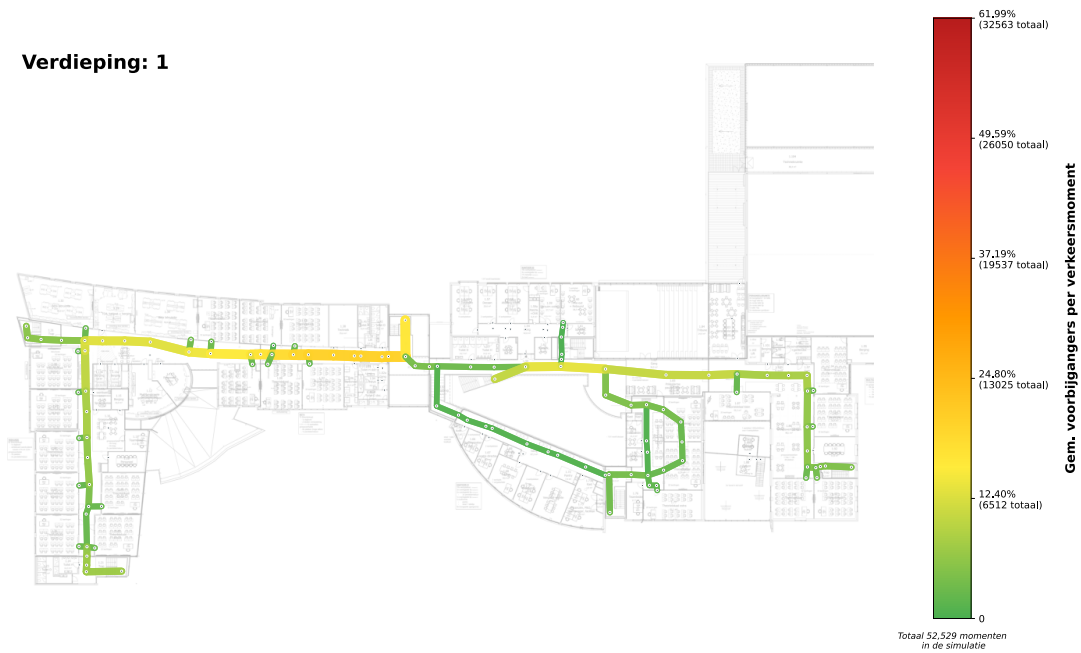
Figure A.6: Aggregated weekly student commute distances derived from matrix mapping of the structural timetable ($N = 1189$). Mean individual travel burden sits at $\mu = 4047.13$ m ($\sigma = 429.25$ m, bin width = 100 m).

Verdieping: 0



(a) Ground floor macroscopic probability layout.

Verdieping: 1



(b) First floor macroscopic probability layout.

Figure A.7: Global network edge traversal probability distributions across separate building levels. These spatial indices isolate critical traffic patterns (magnified details provided in corresponding text chapters).

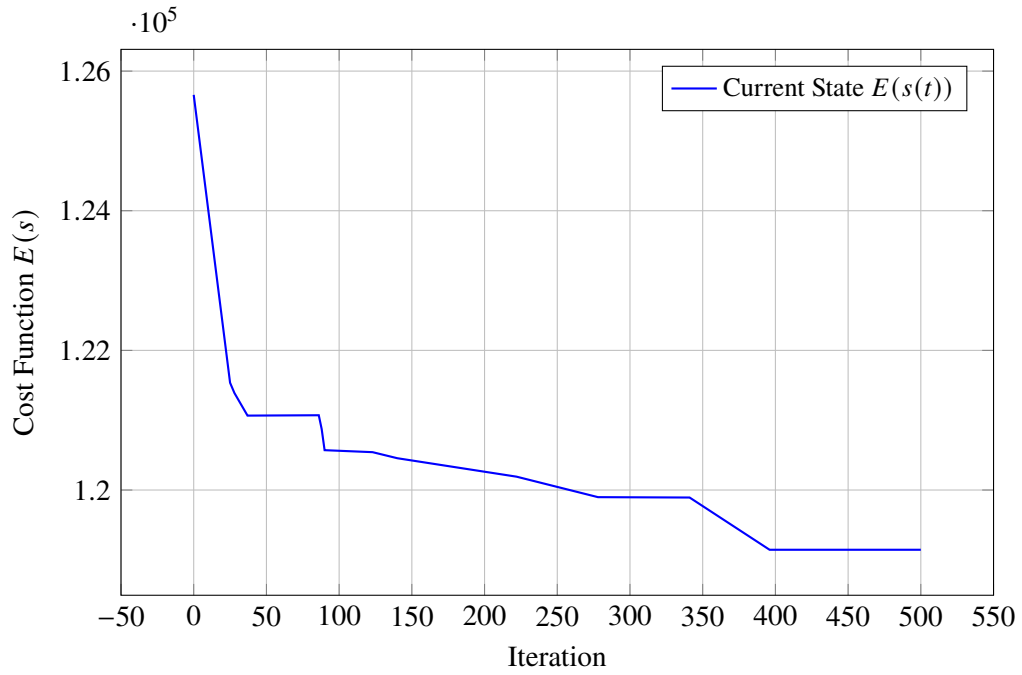


Figure A.8: Convergence trajectory of the Simulated Annealing heuristic loop. Stabilization occurs near iteration 400, securing a 10.2% net mathematical optimization penalty drop compared to unweighted baseline assignments.

Table A.1: Comparative performance metrics between baseline configurations and Simulated Annealing structural optimization output.

Evaluated Metric	Baseline Layout	SA-Optimized Layout	Relative Change
Mean Corridor Edge Load	4880.96	4868.07	−0.3%
Standard Deviation (σ)	6145.52	5864.99	−4.6%
Maximum Single Edge Load	29825.00	27926.00	−6.4%
Minimum Single Edge Load	4.00	7.00	+75.0%
Range Peak Delta (Max – Min)	29821.00	27919.00	−6.4%
Active Bottleneck Segments	18	14	−4 edges
Gini Inequality Coefficient	0.5848	0.5675	−3.0%

3 Phase 3: Microscopic Pedestrian Simulation Pipeline

This section presents the microscopic agent-based simulation pipeline developed inside the *JuPedSim* framework. The core setup logic below illustrates how multi-room boundaries, structural transitions, and corridor layouts are programmatically compiled from the underlying data structures to construct the scenarios analyzed in the figures below.

Listing 1: Python implementation routine of multi-room geometry mapping, surface adapter execution, and agent assignment.

```
1 import hallways.building.geometry as geometry
2 from hallways.adapters import jupedsim_adapter
3 import jupedsim as jps
4 import shapely
5
6 # 1. Map entire multi-vertex ground floor plan structural profile
7 ground = geometry.Floor(
8     level=0,
9     vertices={
10         "v1": (0.0, 0.0), "v2": (4.0, 0.0), "v3": (0.0, 5.0), "v4": (4.0, 5.0),
11         "v5": (0.0, 10.0), "v6": (4.0, 10.0), "v7": (6.0, 0.0), "v8": (6.0, 2.0),
12         "v9": (6.0, 10.0), "v10": (15.0, 0.0), "v11": (15.0, 2.0)
13     }
14 )
15
16 # 2. Establish rooms, transition thresholds, and outer envelope
17 door_1 = geometry.Door(door_id="d1", parent_segment=("v4", "v2"), offset=1.8, width=1.0)
18 door_2 = geometry.Door(door_id="d2", parent_segment=("v6", "v4"), offset=1.8, width=1.0)
19
20 room_1 = geometry.Room(room_id="r1", vertex_ids=["v1", "v2", "v4", "v3"], doors=[door_1])
21 room_2 = geometry.Room(room_id="r2", vertex_ids=["v3", "v4", "v6", "v5"], doors=[door_2])
22 wall_1 = geometry.Wall(wall_id="w1", vertex_ids=["v2", "v7", "v10", "v11", "v8", "v9", "v6"])
23
24 ground.rooms.extend([room_1, room_2])
25 ground.walls.append(wall_1)
26
27 # 3. Transform layout map into a valid walkable tracking poly-surface
28 walkable_poly = jupedsim_adapter.build_walkable_area(ground)
29 simulation = jps.Simulation(model=jps.CollisionFreeSpeedModel(), geometry=walkable_poly)
30
31 # 4. Generate egress points, global trajectories, and route structures
32 exit_poly = shapely.Polygon([(14.0, 0.0), (15.0, 0.0), (15.0, 2.0), (14.0, 2.0)]).intersection(
33     walkable_poly)
34 exit_id = simulation.add_exit_stage(exit_poly)
35 journey_id = simulation.add_journey(jps.JourneyDescription([exit_id]))
36
37 # 5. Distribute agents dynamically inside both room polygons
38 positions = []
39 for rm, s_id in [(room_1, 1), (room_2, 2)]:
40     poly_coord = shapely.Polygon(rm.polygon_coordinates(ground.vertices)).intersection(
41         walkable_poly)
42     positions.extend(jps.distributions.distribute_by_number(
43         polygon=poly_coord, number_of_agents=30, distance_to_agents=0.4, distance_to_polygon
44         =0.2, seed=s_id
45     ))
46
47 for pos in positions:
48     simulation.add_agent(jps.CollisionFreeSpeedModelAgentParameters(
49         journey_id=journey_id, stage_id=exit_id, position=pos, radius=0.15, desired_speed=1.2
50     ))
```

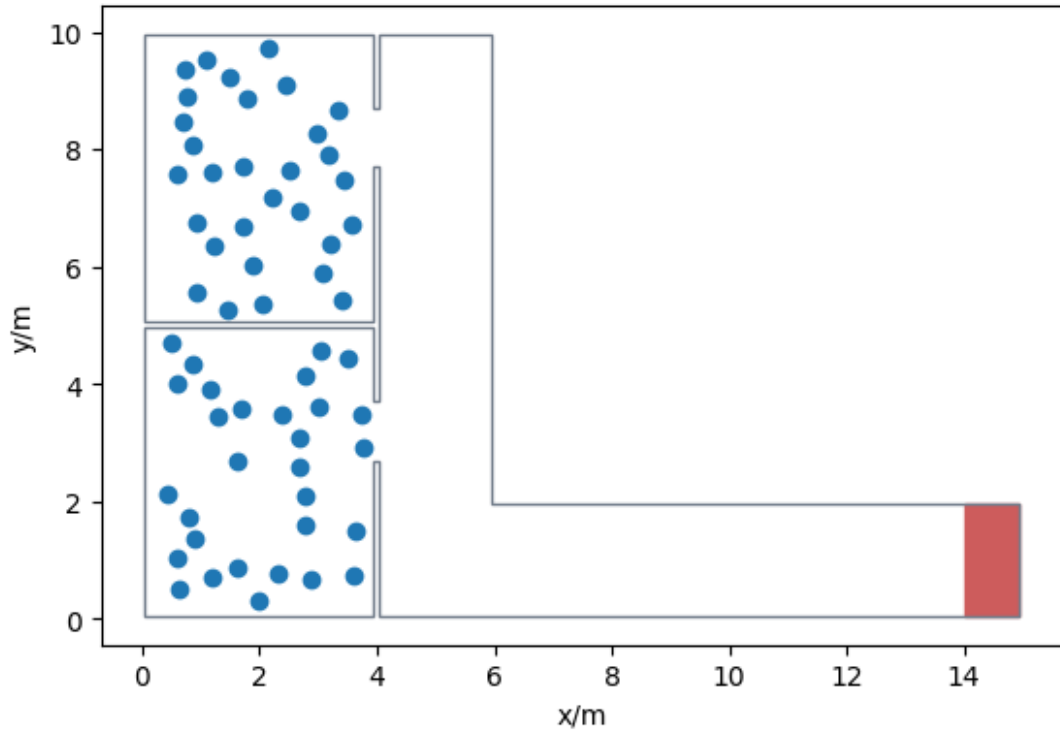


Figure A.9: Microscopic simulation environment environment initialization (matching Listing 1). Spatial CAD/BIM definitions are programmatically parsed into distinct walkable cells, wall borders, and door node thresholds.

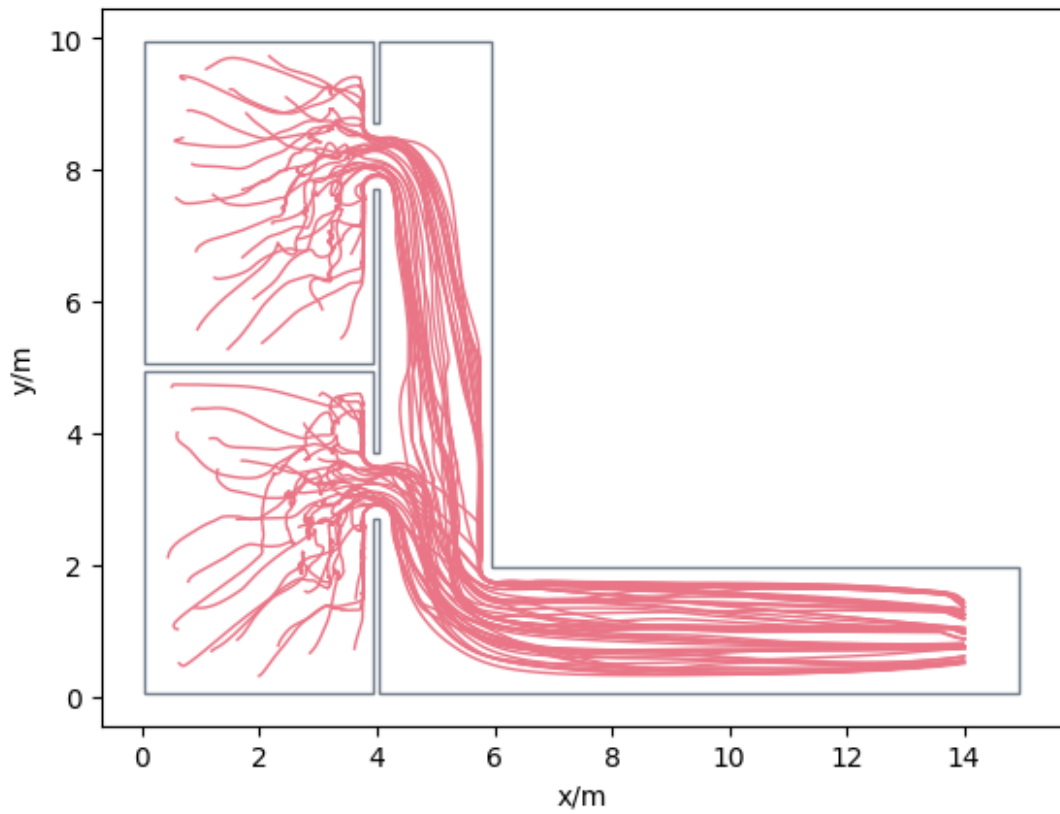


Figure A.10: Global agent routing networks and trajectory track paths across multi-level corridors, showcasing the operational speed model and the intersection of directional visibility graphs.

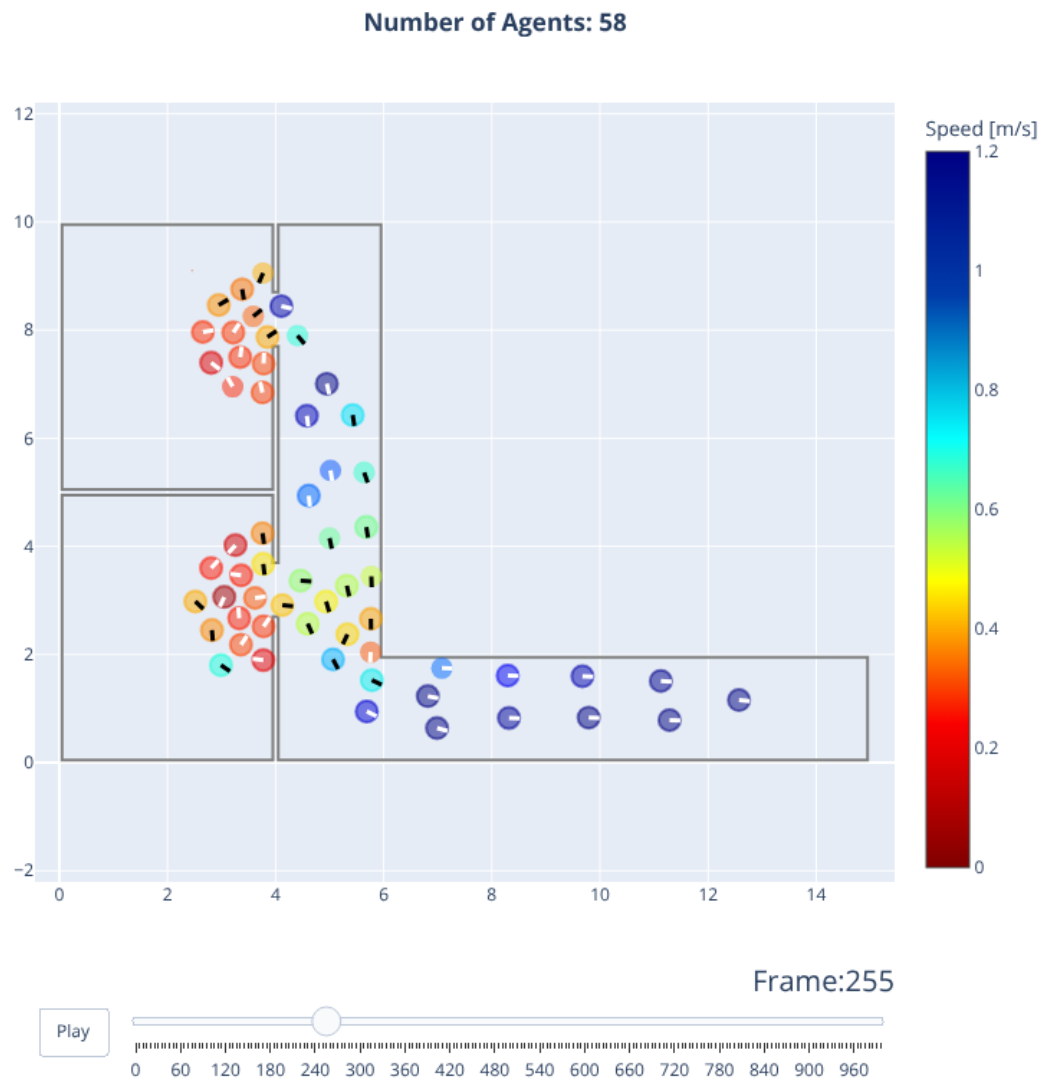


Figure A.11: Quantitative results capturing localized pedestrian flow density metrics, step distributions, and velocity changes across the critical lesson-transition window.