Final Project - Group G
Pedestrian Trajectory Prediction Methods: Deep Learning and Knowledge-Based Approaches

Tasks addressed:	6
Authors:	Zhaozhong Wang (03778350)
	Anastasiya Damaratskaya (03724932)
	Bassel Sharaf (03794576)
	Thanh Huan Hoang (03783022)
	Celil Burak Bakkal (03712329)
Last compiled:	2025–02–06

The work on tasks was divided in the following way:

Zhaozhong Wang (03778350)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Anastasiya Damaratskaya (03724932)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Bassel Sharaf (03794576)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Thanh Huan Hoang (03783022)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Celil Burak Bakkal (03712329)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%

Report on task 0: Abstract

Pedestrian trajectory prediction is a fundamental task with applications in multiple areas, such as crowd dynamics analysis, autonomous driving, and safety-critical systems. Predicting how humans move in dynamic and interactive environments requires understanding individual behavior, social interactions, and environmental factors. To address this complex problem, researchers have developed a range of approaches, broadly categorized into knowledge-based (KB) methods and supervised deep learning (DL) approaches. This report compares both approaches by analyzing two knowledge-based models (ORCA and Social Force) and three deep learning models (Social LSTM, Social GAN, and Behavoir-CNN). Finally, we research existing hybrid models that combine both methods and summarize the current state of the research in the pedestrian trajectory prediction field.

Report on task 1: Knowledge-Based Models

In knowledge-based models, the rules that determine the movement of the pedestrians are manually defined by the authors. Generally, the rules contain how the pedestrians move toward the goal and avoid collisions with other pedestrians and obstacles at the same time. These rules can be rather logical, like in cellular automata, where the map is discretized into grid cells, and the movement of a pedestrian depends on the occupancy of the nearby grid cells. In many recent knowledge-based models, the rules are rather mathematical, where the velocity or acceleration of the pedestrian is obtained via mathematical equations or optimization problems. Such models are called velocity-based models, if the movement is defined by the velocity, or acceleration-based models, if the movement is defined by a virtual force or acceleration. In this task we discuss two representative models, which are ORCA and Social Force, to see how such models work. In the end, we briefly compare these two models.

1 ORCA

All the figures in this section are from [1] if not stated otherwise.

1.1 Introduction

One way to simulate the trajectory is to simulate the speed of each pedestrian in every time window τ. The method ORCA, proposed by Jur van den Berg et al. [1], is the most representative of this kind. In this model, the main goal of the moving agents is to avoid collision either with the other agents or the obstacles. ORCA stands for *Optimal Reciprocal Collision Avoidance*, and this term also represents a subset of the collision-free region in velocity space within the current time window τ. For an agent A, every other agent as well as obstacle within a certain range will determine a specific ORCA region, and the ideal velocity value of agent A should lie within the intersection of those ORCA regions while having the smallest distance to the desired velocity v^{pref}. The authors show that under certain assumptions, this process can be a linear programming (LP) problem and thus be solved efficiently in linear time $O(N)$ where N is the number of the agents and obstacles.

1.2 Assumptions

For the reason of simplicity and without loss of generosity, we assume that each agent is a circular entity in 2D space, and each agent has the ability to decide its own velocity as well as estimate the velocity of other agents. The obstacles are simplified as line segments. It is also assumed that every agent has its own maximum speed v_{max}.

1.3 Velocity Obstacle (VO)

For an agent A at position p_A with radius r_A and another agent B at position p_B with radius r_B, within a certain time window τ, the velocity obstacle for A induced by B for time window τ, denoted by the symbol $VO^{\tau}_{A|B}$, has a shape shown in Figure 1. Note that in this figure $VO^{\tau}_{A|B}$ is the whole gray region in velocity space, and this gray region is the forbidden zone for the relative speed of A with respect to B, i.e., we constrain that $v_A - v_B \notin VO^{\tau}_{A|B}$. (Side note: here it's estimated that B continues with his current speed v_B)

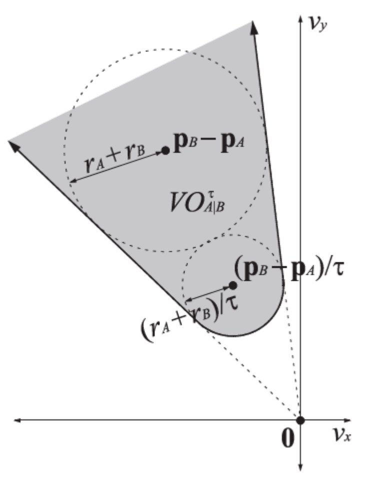


Figure 1: the velocity obstacle for A induced by B for time window τ, denoted by $VO^{\tau}_{A|B}$.

1.4 ORCA

To avoid a drastic change in the speed of agent A, we do not treat every part of the region outside the VO equally. Instead, we prefer the region that is rather close to $v_A - v_B$ so that the agent doesn't dash around. In the paper [1], the speed upon which we optimize is called v^{opt}, so here we let $v^{opt} = v_A - v_B$. If v_{opt} is outside the VO, then we do not need to add any extra constraint; if v_{opt} is inside the VO, we find the shortest way to escape the VO, which is a vector that starts from v^{opt} and points toward the nearest boundary of VO. This shortest way is shown as vector $\mathbf{u}$ in Figure 2.

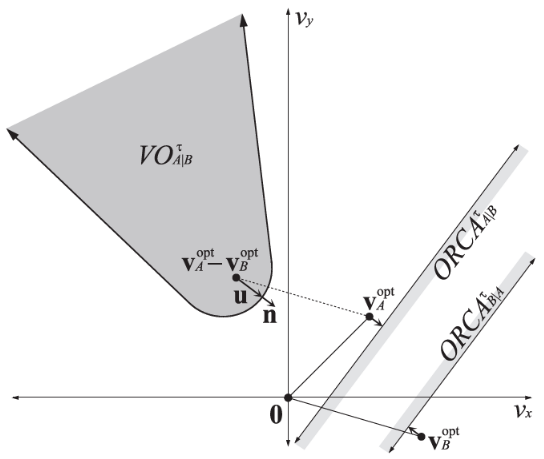


Figure 2: The shortest way to escape VO and the resulting ORCA.

To make agent A and agent B share the equal responsibility of collision avoiding, we rule that agent A only needs to make a speed change that accounts for at least $\frac{1}{2}\mathbf{u}$, instead of a speed change of at least $\mathbf{u}$. Technically, let Δv_A denote the necessary speed change of agent A, then a valid Δv_A should satisfy $proj_{\mathbf{u}}\Delta v_A \geq \frac{1}{2}||\mathbf{u}||$. Let ΔV_A denote the set of all the valid Δv_A, then $ORCA^{\tau}_{A|B}$ is defined as $v^{opt}_A + \Delta V_A$. This is also shown in Figure 2, and the same applies for agent B.

It is easy to see that between two agents A and B, $ORCA^{\tau}_{A|B}$ is a half-plane with a linear boundary. Let $ORCA^{\tau}_A$ denote the intersection of all the $ORCAs$ with respect to A, i.e., $ORCA^{\tau}_A = \bigcap_{B \neq A} ORCA^{\tau}_{A|B}$, then we can see that finding the best speed v^{new}_A within $ORCA^{\tau}_A$ is like minimizing $\parallel v - v^{pref} \parallel$ under a set of linear constraints, where v^{pref} is the desired velocity (e.g., the velocity with which the Dijkstra distance between agent A and the goal decreases the fastest). This is not yet a standard linear programming problem like stated in

Figure 3 because the thing we are minimizing is a quadratic function, i.e., $(v-v^{pref})^2$, and it can not be reduced to a form like $c^T x$. However, with the algorithm proposed in [2], we can still solve this in linear time $O(N)$. Another thing to note is that we still haven't paid attention to the maximum speed v^{max}, and the constraint $\parallel v \parallel < \parallel v^{max} \parallel$ is also quadratic. However, circumventing this constraint is not difficult. We can use a polygon to approximate the circle $v_x^2 + v_y^2 = v^{max\,2}$, and it will transform the quadratic constraint into a couple of linear constraints.

$$\begin{array}{ll} \text{Find a vector} & \mathbf{x} \\ \text{that maximizes} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0}. \end{array}$$

Figure 3: The standard form of linear programming.

Intuitively, $ORCA_A^\tau$ (together with the linearized constrains regarding v^{max}) will be a convex polygon, and if v^{pref} is within this convex polygon, we simply set v^{new} to v^{pref}. If v^{pref} is outside this convex polygon, v^{new} should be the point on the boundary of this polygon that has the shortest distance to v^{pref}. This v^{new} can be found by finding the closest point to v^{pref} on each edge of the convex polygon, and among those points, the one with the shortest distance to v^{pref} will then be chosen as v^{new}.

1.5 Special Cases

One special case is that $ORCA_A^\tau = \phi$, i.e., $ORCA$ is empty. This can happen when the agents are placed very densely with each other. An example is shown in Figure 4. Note that $ORCA$ is empty does not necessarily mean that a collision is inevitable because $ORCA$ does not cover the whole collision-free region in velocity space. Also, other agents may not continue with their current velocities as $ORCA$ implies. In this case, we relax the linear constraints by simultaneously shifting all the $ORCA$ boundaries outward with the same speed until their intersection is no longer empty. The first point that appears in the intersection will be chosen as v_A^{new}.

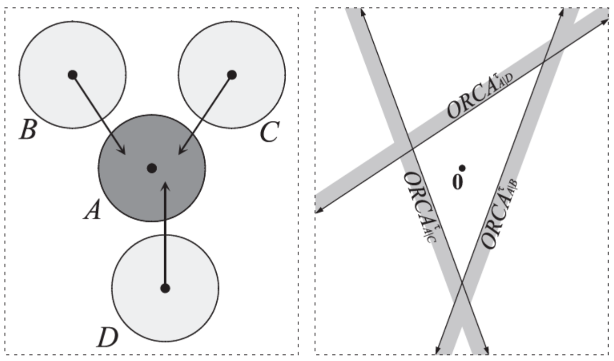


Figure 4: One case where three agents B,C, and D are all moving toward agent A and thus $ORCA_A^\tau = \phi$.

Another special case is that the agent is moving toward an obstacle. In this case, the authors set $v^{opt} = 0$ because then it is guaranteed that a collision-free solution always exists (in the worst case, the agent just stands still). Note that in case of mutual collision avoidance among the agents, we let $v^{opt} = v_A - v_B$ because we want to avoid a sudden change of the velocity of agent A, and even if this may result in smaller $ORCA$ regions, we can expect that the collision may still be avoided by the movement of agent B. Also, in real-life crowded situations, e.g., in a crowded subway station, a collision between two pedestrians sometimes happens, and it is acceptable if the probability of such collision is below a certain threshold. However, obstacles in the case of the subway example can be the edge of the platform, and even in simulation, we do not want to let any pedestrian fall into the rails, so we set $v^{opt} = 0$ for the safety guarantee.

Since we can always use a polygon to approximate the boundary of an obstacle with arbitrary shape, the authors in [1] only discuss the case where an obstacle is represented as a line segment, as shown in Figure 5, and the corresponding VO will have the shape of a truncated triangle as seen in (b) of this figure, and the corresponding $ORCA$'s boundary will be tangent to the truncation and covers the opposite side of the VO, as shown in (c) in this figure.

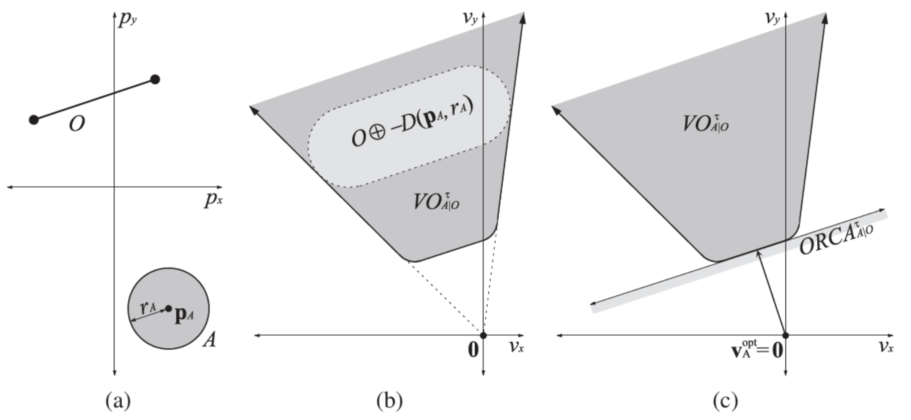


Figure 5: $ORCA$ of agent A against a line segment obstacle O.

1.6 Extension of ORCA Model

Some other concepts concerning pedestrian behaviors can also be integrated into $ORCA$. One example is "Less-Effort Collision Avoidance in Virtual Pedestrian Simulation"[3] published by Zhipeng Yin et al. in 2019. In this paper, the authors use the same procedure as we already described to find the intersection of $ORCA$ regions, but v^{new} is not obtained by finding the point with the shortest distance to v^{pref}. Instead, the authors try to find the velocity within the $ORCA$ intersection that minimizes a power consumption term P. The full name of P is "instantaneous power consumption per unit mass," and $P = e_s + e_d \mid v \mid^2 + e_r \mid \omega \mid^2$. Here e_s, e_d, and e_r are all constants, and the v and ω denote the velocity and angular velocity of the agent. The authors predefine the movement pattern of an agent as follows: The agent will first pick a valid velocity inside $ORCA$ intersection, and then after the time interval of τ, the pedestrian arrives at position $\mathbf{q}_A$, as seen in Figure 6. After reaching $\mathbf{q}_A$, the pedestrian changes the velocity into the expected one v^{pref} via a constant-speed circular motion centered at $\mathbf{o}$ with a turning radius of $\rho = v_{max}/\omega_{min}$ and reach the position $\mathbf{q}'_A$. In the figure, the authors assume that the target is far away from the pedestrian, so v^{pref} at $\mathbf{p}_A$ and $\mathbf{q}'_A$ are identical. v^{new} is then the velocity that can execute such a pattern of movement with the smallest energy consumption P.

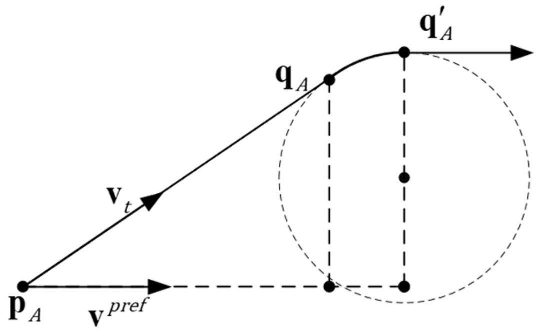


Figure 6: Predefined local dynamic process of the Energy Consumption Model [3].

Although this method is more realistic, it comes with the price that it's no longer a linear optimization problem, and the time complexity of the algorithm is $O(n \cdot m)$, where n is the number of pedestrians and obstacles in the neighborhood and m is the number of discrete sampling points that calculate the optimal energy consumption.

1.7 Advantages and Disadvantages

ORCA is a fast model, and with the development of GPU computation these years, linear programming (LP) problems can be solved very efficiently. [4] tried to implement ORCA model on GPU and it achieves 60 FPS with over 100,000 pedestrians in simulation, which is very remarkable. Also, as a general advantage of KB methods, collision-avoiding performance is guaranteed. Due to the design of v^{opt}, ORCA also produces smooth trajectories, which makes it not only just a simulation tool but also a method that can be implemented in real life in, e.g., robotics.

However, ORCA does not perform very well in densely packed crowds because every agent is always trying its best to avoid collision and may not want to proceed to the goal. We can see in subsection **Special Cases** that if $ORCA_A^\tau = \phi$, the agent will just find the best possible collision-avoiding velocity without considering v^{pref} anymore. So in certain applications like evacuation simulation, using ORCA will not be ideal since the agents in simulation tend to get stuck in front of the exit without trying to rush out. Besides, ORCA doesn't do well in situations where it's necessary to pass narrow passages.

Overall, ORCA is a relatively conservative model, it should not be used to simulate situations like evacuation or overpacked crowds. However, for general purposes, like simulating people wandering in the mall, ORCA is one of the fastest and simplest methods.

2 Social Force Model

All the figures in this section are from [5] if not stated otherwise.

2.1 Introduction

The Social Force Model (SFM), introduced by Helbing and Molnár [5], addresses the challenge of simulating pedestrian behavior and motion by providing a model with mathematical terms and functions that describe "social forces", which guide the motion of each individual pedestrian. There are mainly three mathematical terms of "social forces" described by SFM, which are acceleration term, term of repulsive forces and a term of attractive forces. The acceleration term describes the phenomenon that each pedestrian has a desired velocity of motion and tries to return to it when the actual velocity deviates from it. Furthermore, the repulsive forces represent the tendency of pedestrians to keep their distance from each other or from borders and obstacles. Lastly, the attractive forces describe the behavior of pedestrians to be attracted to (i.e. move close to) certain pedestrians or objects such as friends, family members or a artifact in a museum. The simulations realized by utilizing SFM demonstrates that this method of using "social forces" can replicate many behaviors of pedestrians that occurs in daily life.

2.2 Social Forces and Mathematical Terms

2.2.1 Acceleration Term

Each pedestrian α wants to reach a certain destination r_α^0 and for this it follows a path that leads to this destination, which is composed of partial edges $\vec{r}_\alpha^k$ with k $\in \{1, \ldots, n\}$ and can be defined as follows:

$$\vec{r}_\alpha^1, \ldots, \vec{r}_\alpha^n := \vec{r}_\alpha^0$$

With the previously defined notations and the location of the pedestrian α at time t, i.e. $\vec{r}_\alpha(t)$, the desired direction of pedestrian $\vec{e}_\alpha(t)$ is defined as follows:

$$\vec{e}_\alpha(t) := \frac{\vec{r}_\alpha^k - \vec{r}_\alpha(t)}{\|\vec{r}_\alpha^k - \vec{r}_\alpha(t)\|}$$

As it can be inferred from the normalization with the denominator of this equation, the desired direction $\vec{e}_\alpha(t)$ is a unit vector as it only contains direction not the speed. Furthermore, each pedestrian α also has a desired speed v_α^0, with which they move in their desired direction $\vec{e}_\alpha(t)$. Moreover, multiplication of these two terms result in the desired velocity $\vec{v}_\alpha^0(t) := v_\alpha^0 \vec{e}_\alpha(t)$. Due to avoidance of other pedestrians and borders of objects like walls or pillars, the actual velocity $\vec{v}_\alpha(t)$ of a pedestrian α can deviate from its desired velocity $\vec{v}_\alpha^0(t)$. The reason for this is that the avoidance behavior can slow the pedestrian or force it to move in a different direction than the desired direction for some time. Therefore, pedestrians aim to return to their desired velocity within a certain relaxation time τ_α. Finally, this behavior can be expressed by the following acceleration term:

$$\vec{F}_\alpha^0(\vec{v}_\alpha, v_\alpha^0 \vec{e}_\alpha) := \frac{1}{\tau_\alpha}\left(v_\alpha^0 \vec{e}_\alpha - \vec{v}_\alpha\right).$$

This term divides the difference between actual velocity $\vec{v}_\alpha(t)$ and desired velocity $\vec{v}_\alpha^0(t)$ with the relaxation time τ_α. Therefore, there are two cases that can make this term large and cause a strong acceleration effect. First one is pedestrian deviating from the desired velocity $\vec{v}_\alpha^0(t)$ too much. Second one is pedestrian aiming to return to desired velocity $\vec{v}_\alpha^0(t)$ quickly within a small time, i.e. having a small value for relaxation time τ_α.

2.2.2 Repulsive Forces

The SFM describes repulsive forces with two separate terms, one for repulsive forces from other pedestrians and one for repulsive forces from boundaries.

Repulsive forces from other pedestrians represent the common behaviour in daily life, which is that pedestrians usually refrain from getting too close if they do not know each other. This behaviour is reflected in SFM with the following vector definition:

$$\vec{f}_{\alpha\beta}(\vec{r}_{\alpha\beta}) := -\nabla_{\vec{r}_{\alpha\beta}} V_{\alpha\beta}[b(\vec{r}_{\alpha\beta})]$$

In this definition, the function $V_{\alpha\beta}(b)$ represents the repulsive potential and its output is the strength of the repulsion applied from other pedestrians β on pedestrian α. It is a monotonically decreasing function, which fits the behavior that the closer a pedestrian is to another pedestrian, the more discomfort the pedestrian feels, and tends to keep their distance more forcefully.

Furthermore, the repulsive potential function is guided by its input parameter b. In order to understand the term b, one can think of the personal space of a pedestrian as an ellipse, where a pedestrian is at the center C of it. The wide edge of the ellipse represents the front and back of the pedestrian, and the narrow edge represents the lateral sides of the pedestrian. In this ellipse, the term b represents the half of the lateral sides, which is referred to as the semi-minor axis formally, as illustrated in Figure 7. Note that the term "personal space" is used as a metaphor for repulsive potential around pedestrian here, as it makes the pedestrian maintain a certain space around itself.

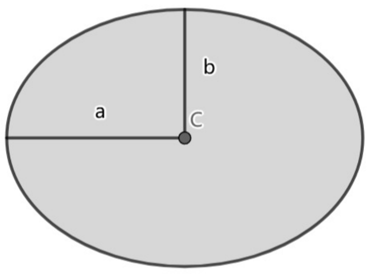


Figure 7: Visualization of the semi-major (a) and semi-minor (b) axes of an ellipse. (made using Geogebra)

The previously explained term **b** is computed by dividing the following equation by two:

$$2b := \sqrt{(\|\vec{r}_{\alpha\beta}\| + \|\vec{r}_{\alpha\beta} - v_\beta \Delta t \vec{e}_\beta\|)^2 - (v_\beta \Delta t)^2}$$

, where Δt denotes the time interval for a single step of pedestrian. The purpose of parameter **b** is to make sure that pedestrian has enough space for its step by adjusting repulsion potential strength according to it.

The gradient of the repulsive potential function $V_{\alpha\beta}(b)$ is calculated with regard to the difference between positions of pedestrians α and β, i.e., $\vec{r}_{\alpha\beta} := \vec{r}_\alpha - \vec{r}_\beta$. Finally, the effect of repulsive forces $\vec{f}_{\alpha\beta}$ is computed as the negative gradient of the repulsive potential function $V_{\alpha\beta}$.

Another case of repulsive forces is the one that caused by borders of objects like buildings, pillars, or barriers. This behavior is described as follows:

$$\vec{F}_{\alpha B}(\vec{r}_{\alpha B}) := -\nabla_{\vec{r}_{\alpha B}} U_{\alpha B}(\|\vec{r}_{\alpha B}\|)$$

Here, similar to the previous formula, a monotonically decreasing potential function $U_{\alpha B}(\|\vec{r}_{\alpha B}\|)$ and its negative gradient is utilized. The gradient is computed with regard to the term $\vec{r}_{\alpha B} := \vec{r}_\alpha - \vec{r}_B^\alpha$, which represents the difference between the positions of pedestrian α and border B, which are denoted as $\vec{r}_\alpha$ and $\vec{r}_B^\alpha$ respectively. It is also to note that this time the parameter b is not used. Only the magnitude of the vector $\vec{r}_{\alpha B}$ is given as input parameter for determining the repulsive potential.

2.2.3 Attractive Forces

As mentioned in the previous subsections, there are also pedestrians and objects like friends, family members, or interesting artifacts at a museum that also have an attractive force on a pedestrian α, i.e., make the pedestrian α move towards them. The effect of these attractive forces are defined with the following vector function:

$$\vec{f}_{\alpha i}(\|\vec{r}_{\alpha i}\|, t) := -\nabla_{\vec{r}_{\alpha i}} W_{\alpha i}(\|\vec{r}_{\alpha i}\|, t)$$

, where the term $\vec{r}_{\alpha i} := \vec{r}_\alpha - \vec{r}_i$ stands for the difference between the positions of pedestrian α and place i. Here, again a formula with a negative gradient of the potential function just like the formulation of repulsive forces is utilized. This time, the potential function $W_{\alpha i}(\|\vec{r}_{\alpha i}\|, t)$, which outputs the strength of attractive forces applied on pedestrian α at place i, is monotonically increasing instead of monotonically decreasing like the repulsive ones. It is also important to note that a second input parameter t, which represents time, is also given to the attractive potential function $W_{\alpha i}$ and the magnitude of attractiveness vector $\|\vec{f}_{\alpha i}\|$ is decreasing with it. This represents the behavior of pedestrians to have less interest in attractive entities as the time passes.

2.2.4 Effect of Perception

Although the formulas introduced in the previous sections for attractive and repulsive forces are suitable for objects and pedestrians in the desired direction of motion $\vec{e}_\alpha(t)$ of the pedestrian α (i.e. in front of it), it neglects the fact that usually in real life a pedestrian gives more attention to the pedestrians or objects in its field of view than the ones out of it. Therefore, SFM introduces a term of direction dependent weights, which makes sure that entities out of the field of vision have less influence than the ones in it. This term is defined as follows:

$$w(\vec{e}, \vec{f}) := \begin{cases} 1 & \text{if } \vec{e} \cdot \vec{f} \geq \|\vec{f}\| \cos\varphi, \\ c & \text{otherwise.} \end{cases}$$

, where $0 < c < 1$ is the influence factor, $\vec{e}$ is the desired direction of motion, $\vec{f}$ is the function of attractive or repulsive forces defined in the previous subsections and $\cos\varphi$ refers to the angle of sight 2φ.

After multiplying the direction dependent weights with the functions of attractive and repulsive forces defined in the previous subsections, we get the following two weighted functions for the attractive and repulsive forces:

$$\vec{F}_{\alpha\beta}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_\beta) := w(\vec{e}_\alpha, -\vec{F}_{\alpha\beta})\vec{f}_{\alpha\beta}(\vec{r}_\alpha - \vec{r}_\beta),$$

$$\vec{F}_{\alpha i}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_i, t) := w(\vec{e}_\alpha, \vec{f}_{\alpha i})\vec{f}_{\alpha i}(\vec{r}_\alpha - \vec{r}_i, t).$$

, where the first one is for the repulsive forces and the second one is for the attractive forces.

2.3 Final Definition of Social Force Model

After summing up all the functions defined in the previous subsections for acceleration, repulsion and attraction, the following function for total motivation $\vec{F}_\alpha(t)$ of a pedestrian α can be defined:

$$\vec{F}_\alpha(t) := \vec{F}^0_\alpha(\vec{v}_\alpha, v^0_\alpha \vec{e}_\alpha) + \sum_\beta \vec{F}_{\alpha\beta}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_\beta) + \sum_B \vec{F}_{\alpha B}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}^\alpha_B) + \sum_i \vec{F}_{\alpha i}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_i, t).$$

In order to also consider the randomness of the pedestrian behaviour, a fluctuation term is added. This gives us the following SFM definition:

$$\frac{d\vec{w}_\alpha}{dt} := \vec{F}_\alpha(t) + \text{fluctuations}$$

, where $\vec{w}_\alpha(t)$ is the preferred velocity of pedestrian α.

Finally, using this definition of preferred velocity $\vec{w}_\alpha(t)$, the actual velocity $\vec{v}_\alpha(t)$ of pedestrian α is defined as follows:

$$\frac{d\vec{r}_\alpha}{dt} = \vec{v}_\alpha(t) := \vec{w}_\alpha(t)\, g\left(\frac{v^{\max}_\alpha}{\|\vec{w}_\alpha\|}\right)$$

In this definition of actual velocity, preferred velocity $\vec{w}_\alpha(t)$ is multiplied with another function **g**, which makes sure the actual velocity $\vec{v}_\alpha(t)$ does not exceed a certain maximal acceptable speed $v^{\max}_\alpha$. This function is defined as follows:

$$g\left(\frac{v^{\max}_\alpha}{\|\vec{w}_\alpha\|}\right) := \begin{cases} 1 & \text{if } \|\vec{w}_\alpha\| \le v^{\max}_\alpha, \\ \frac{v^{\max}_\alpha}{\|\vec{w}_\alpha\|} & \text{otherwise.} \end{cases}$$

In the definition of function **g** above, it can be seen that it does not change the preferred velocity $\vec{w}_\alpha(t)$ if its magnitude is less than the maximal acceptable $v^{\max}_\alpha$. Otherwise, it normalizes the maximal acceptable speed $v^{\max}_\alpha$ with the magnitude of preferred velocity $\vec{w}_\alpha(t)$, i.e. $\frac{v^{\max}_\alpha}{\|\vec{w}_\alpha\|}$, and assigns this value to the actual velocity $\vec{v}_\alpha(t)$.

2.4 Simulations

Although the mathematical formulations of SFM guide the motion of each pedestrian individually, it is still able to realize collective behaviors. Two of these collective behaviors are shown in Figure 8 and Figure 9. Both cases simulate two groups of pedestrians (i.e. empty and full circles) moving in opposite directions but in different environments. In Figure 8, simulation takes place in a straight and quite narrow corridor. Here, the ability of pedestrians to form lanes with SFM is demonstrated. It can be observed that in this case around 4 lanes are formed.

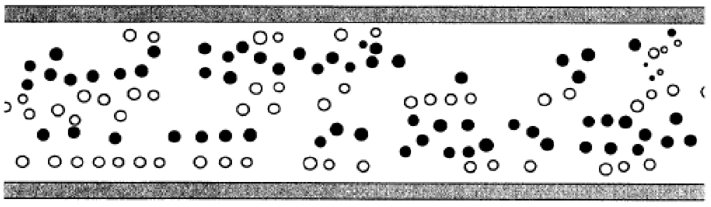
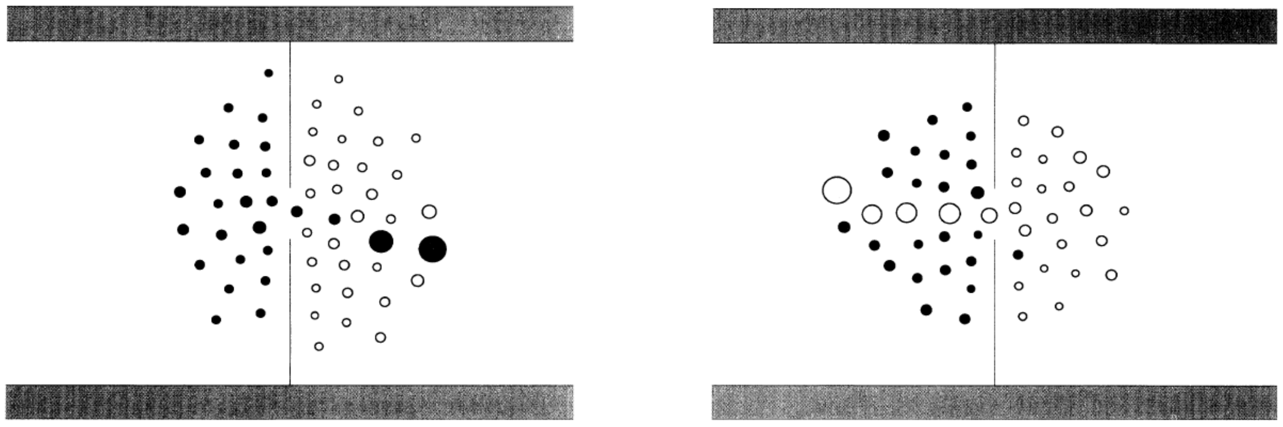


Figure 8: Two pedestrian groups moving in a corridor and forming lanes.

In Figure 9, another simulation with different environment, which is a wider corridor with a vertical wall and a narrow door in the middle can be seen. This time, the ability of pedestrians to pass through a narrow door in alternation is shown. This alternation behavior is caused by the repulsive forces, which different groups apply to each other and makes one group stop after some time and the other group starts moving.


(a) Full circle group passing through the door from left to right.

(b) Empty circle group passing through the door from right to left.

Figure 9: Two pedestrian groups passing through a narrow door in alternation

It is also important to note that in these simulations grouping behavior is realized by attractive forces.

2.5 Advantages and Disadvantages

SFM is a model that is able to capture various individual behaviors of pedestrians like acceleration, repulsion, and attraction. Furthermore, it is also able to replicate collective behaviors like lane formation or passing through doors in alternation. One of the main advantages of SFM is its interpretability because it does all of this by using few mathematical formulations that describe pedestrian behavior. Therefore, the resulting behavior is often understandable and explainable to observers. Moreover, SFM also provides a certain degree of flexibility by allowing to manually tune the constant parameters in the functions of social forces.

On the other hand, SFM also comes with many disadvantages and limitations. The motion and behavior of pedestrians simulated by it highly depend on manually chosen constrained functions (e.g. potential functions of repulsive and attractive forces) and the parameters that are determined by the user. If these functions and parameters are not chosen correctly, the model may fail at simulating the pedestrians properly. Furthermore, the real world is highly variable and contains factors like changing weather conditions (e.g., rain, snow, fog), natural disasters (e.g., earthquake, fire), changing group dynamics based on different group types (e.g., repulsion and attraction potentials would be different between family, friends, business colleagues or a tourist group). The only randomness in SFM is the random fluctuation term it adds, but it might not be enough to handle the high randomness of the real world.

2.6 Improvements

Possible improvements of classical SFM [5] can be made by modifying its functions and parameters. One of the methods that does this is introduced in the paper “Social force model with explicit collision prediction” [6] published by F. Zanlungo et al. This method aims to improve the classical SFM by predicting the time when the next collision will happen and replacing the parameter Δt of classical SFM with this predicted time value. In the classical SFM, Δt represents the length of a single step of a pedestrian. This replacement is made by removing the input parameter b of repulsive potential function in classical SFM, which is calculated based on the parameter Δt and was explained in previous subsections. After that, a term based on the predicted collision time is used in the repulsive potential function. The computation of the predicted collision time can be explained using the Figure 10.

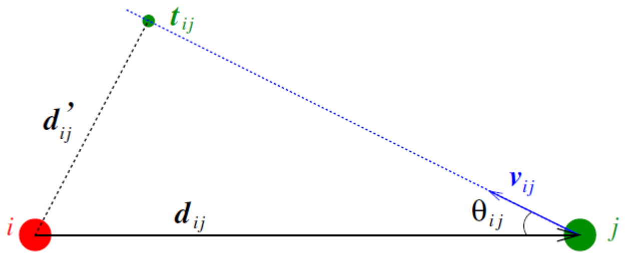


Figure 10: Visualization of the terms of distances, angles and velocities on an example scene with two pedestrians. [6]

In Figure 10, the red and green dots represent the pedestrians i and j respectively. The distance and predicted future distance between them are denoted as d_{ij} and d'_{ij}, respectively. For each pedestrian j in a certain proximity to the main pedestrian i, the future moment t_{ij}, at which the two pedestrians have the smallest distance between each other, is computed. This computation occurs only if the angle θ_{ij} between $v_{i,j}$ and $d_{i,j}$ is less than $\frac{\pi}{4}$. It is also important to note that both pedestrians are assumed to follow a straight trajectory when performing these calculations. After computing the future point of time t_{ij} corresponding to the smallest distance for each pedestrian, the minimum of these time points $t_i = \min_j\{t_{i,j}\}$ is taken, and the locations of all pedestrians, including the main pedestrian i itself at this future time point, are computed. Finally, with the computed minimum time t_i and predicted locations at it, the following repulsive forces of other pedestrians j on the main pedestrian i is constructed:

$$f_{i,j}\left(\{d_{i,j}\}, \{v_{i,j}\}, v_i\right) = A\frac{v_i}{t_i}e^{-d_{i,j}/B}\frac{d'_{i,j}(t_i)}{d'_{i,j}(t_i)}.$$

In this equation, it can be seen that the multiplication with the inverse of predicted minimum collision time t_i makes the repulsive potential high when it has a small value. This is logical as the pedestrian should be more cautious when there is only a short time until the collision. It can also be seen that repulsive potential is directly proportional to the velocity of the main pedestrian i, i.e., v_i, as the pedestrian should be more careful about collision if it is faster. It is also to note that the parameters A and B in this equation are predefined constants.

Although this method of constructing potential functions with predicted time of collisions enables pedestrians to avoid collisions better than classical SFM and also reduces the number of parameters by removing the parameters $\mathbf{b}$ and Δt, it still creates a computational burden on the model because of the computations of predicted collision times. Therefore this method can make the model slower, especially when there are many pedestrians j approaching the main pedestrian i with degree θ_{ij} less than $\frac{\pi}{4}$ in close distance, because it will make the model perform many calculations.

3 Comparison Between the Knowledge-Based Models

After introducing two KB models, we compare them, highlighting their differences and similarities.

In ORCA, there is no tunable parameter, and collision avoidance has the highest priority. On the contrary, Social Force is tunable, e.g., we can prioritize the movement toward certain pedestrians or objects by increasing the coefficients concerning the attractive forces. For large crowd simulation, the Social Force Model is more expensive because it requires calculating the force from many other pedestrians, while ORCA simply needs to check collision avoidance in the local neighborhood. Regarding simulation, the Social Force Model offers more flexibility because each individual can have its own parameters, and group behaviours can also be realized by the model. On the other hand, the complexity of the Social Force Model also makes it sometimes unrealistic because the force from other pedestrians can be highly changeable, especially in a crowded scenario, and therefore, we can often observe oscillations in the pedestrian's trajectory in such cases. ORCA, due to its choice of v^{opt}, can produce smooth trajectories even in a large scene. As a result, ORCA can also be used as a real-life path-planning strategy in robotics, while the Social Force Model is mainly used for simulation.

These comparisons can be summarized in Table 1.

Feature	**ORCA**	**Social Force Model**
Tunable Parameters	No	Yes (pre-defined parameters in functions of social forces)
Collision Avoidance	Highest Priority	Implicit through force calculations
Computational Cost for Large Crowds	Lower	Higher
Flexibility	Lower	Higher (individual parameters, group behaviors)
Trajectory Smoothness	Higher	Lower (prone to oscillations in crowds)
Real-World Applicability	Suitable for robotics path planning	Primarily used for simulation
Model Complexity	Lower	Higher

Table 1: Comparison of ORCA and Social Force Model

Report on task 2: Supervised Deep Learning Models

In contrast to knowledge-based approaches, deep learning models can capture the complex, non-linear dependencies inherent in human motion. Such models leverage large-scale data to learn the patterns and dynamics of human trajectories and can easily identify a group or a couple moving. In this chapter, we present models like Social LSTM and Social GAN that have demonstrated remarkable success in capturing temporal dependencies and social interactions, as well as Behavior-CNN, which uses neural networks to extract spatial features and model human-environment interactions effectively. At the end of task 2, a brief comparison between these models is provided.

1 Social LSTM

1.1 Introduction

In order to predict human motions in crowd dynamics, knowing common sense rules, modeling them, and complying with social conventions are essential and challenging tasks. It requires comprehending possible interactions between pedestrians, foreseeing interactions that could occur in the distant future, and considering different motion patterns of pedestrians (Figure 11). A data-driven model called Social LSTM addresses these challenges. Alahi et al.[7] first mentioned the idea of using Long-Short Term Memory networks (LSTM) for predicting human motion and developed Social LSTM, bringing deep learning approaches for human trajectory prediction into more consideration than ever before.

1.2 Architecture

At its core, the Social LSTM utilizes standard LSTMs, which are well-suited for sequence modeling tasks due to their ability to capture long-term dependencies. Each pedestrian has its own LSTM, which learns the current state and predicts the pedestrian's forthcoming positions. Since LSTMs alone cannot exchange information with each other, the naive use of one LSTM model per person does not capture the human interaction with its neighborhood. For that purpose, Alahi et al.[7] introduced a new pooling strategy called social pooling or S-pooling, which connects LSTMs and allows them to transfer information. In order to do so, the mechanism aggregates information about the pedestrian's hidden states within a spatial neighborhood and later uses it as additional input in the next step to predict an individual's future trajectory.

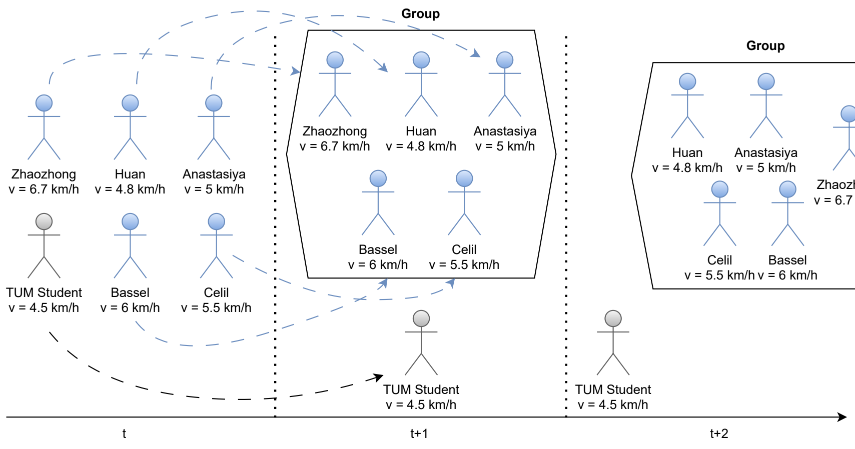


Figure 11: Human trajectories over time considering interactions such as a group movement (blue elements).

From a more detailed perspective, the hidden state h_j^t of each LSTM at time t encodes the motion dynamics of the j-th pedestrian in the scene. In order to share this representation with the neighbors, the model uses a social tensor H_i^t for each pedestrian i, which considers the relative positions and interactions within a predefined neighborhood radius. This social tensor H_i^t is calculated as follows:

$$H_i^t(m,n,:) = \sum_{j \in \mathcal{N}_i} \mathbb{1}_{mn}[x_j^t - x_i^t, y_j^t - y_i^t] h_j^{t-1}$$

where each grid cell (m,n) accumulates contributions from neighbors $j \in \mathcal{N}_i$, weighted by an indicator function $\mathbb{1}_{mn}$ to determine the current occupancy of the grid cell. Then, the pooled social hidden-state tensor is combined with the individual trajectory embeddings e_i^t to produce a context-aware input a_i^t:

$$a_i^t = \phi(H_i^t; W_a), \quad e_i^t = \phi(x_i^t, y_i^t; W_e), \quad h_i^t = \mathrm{LSTM}(h_i^{t-1}, e_i^t, a_i^t; W_l)$$

where $\phi(.)$ is an embedding function with ReLU non-linearity, W_a and W_e are embedding weights and W_l is LSTM weights.

To predict the distribution of the trajectory position at the text time step $t+1$, we again use hidden state at time t, assuming a bivariate Gaussian distribution for the pedestrian's next position $(\hat{x}_i^{t+1}, \hat{y}_i^{t+1})$. This distribution is parameterized by its mean μ, standard deviations σ, and correlation coefficient ρ and results in the following predicted coordinates:

$$(\hat{x}_i^{t+1}, \hat{y}_i^{t+1}) \sim \mathcal{N}(\mu_i^t, \sigma_i^t, \rho_i^t), \quad [\mu_i^t, \sigma_i^t, \rho_i^t] = W_p h_i^t.$$

The Social LSTM model is trained to minimize the negative log-likelihood loss for all the trajectories in a training dataset:

$$L^i(W_e, W_p) = -\sum_{t=T_{\mathrm{obs}}+1}^{T_{\mathrm{pred}}} \log P\big((x_i^t, y_i^t) \mid \mu_i^t, \sigma_i^t, \rho_i^t\big).$$

The social pooling layer does not introduce new parameters to the model but, compared to the traditional LSTM, couples multiple LSTMs' hidden states and back-propagates at every step. The whole Social LSTM model is visualized in Figure 12.

1.3 Experiments and Results

To evaluate the effectiveness of the Social LSTM model, Alahi et al.[7] compared it against several baseline models on publicly available human trajectory datasets (ETH and UCY), assessing prediction accuracy and the ability to model complex social interactions. The chosen datasets include rich annotations and reflect

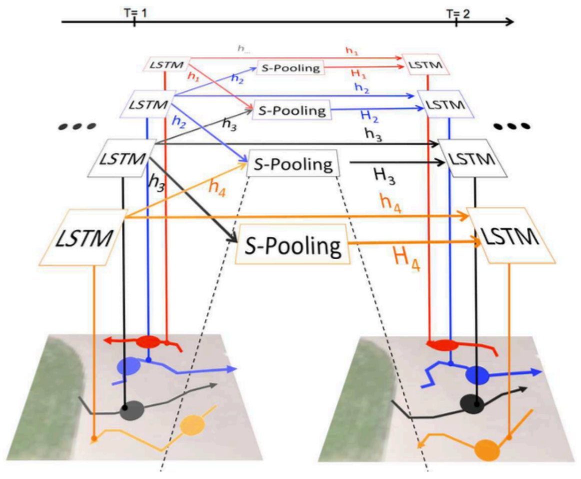


Figure 12: Social LSTM model by Alahi et al. [7]. One LSTM is responsible for one pedestrian each, while the hidden states are used for pooling and predicting the next position in the scene.

diverse real-world pedestrian dynamics with thousands of non-linear trajectories: ETH includes trajectories of pedestrians in outdoor scenarios, while UCY captures various crowd densities and behaviors in different environments.

As evaluation metrics, the researchers used Average Displacement Error (ADE), which measures the average Euclidean distance between predicted trajectories and ground truth over all time steps, Final Displacement Error (FDE), examining the Euclidean distance between the predicted and actual positions at the final prediction step, and Average Non-linear Displacement Error focusing on the non-linear regions of a trajectory.

The Social LSTM model was compared against the linear model, collision avoidance, social force, iterative Gaussian process, naive LSTMs without pooling, and LSTM with occupancy maps (O-LSTM), which pools the neighbors' coordinates at every time step. The evaluation results demonstrate that Social LSTM significantly outperforms all baseline models in almost all datasets, especially for the UCY datasets due to the different crowd densities.

Additionally to quantitative results, the investigators presented visualizations of anticipated trajectories to validate the model's ability to predict socially compliant movements. The Social LSTM accurately modeled collision avoidance and group behaviors, which simpler models did not capture.

1.4 Advantages and Disadvantages

Social LSTM introduces a unique social pooling mechanism, which allows the model to consider the interactions between pedestrians in crowded environments and adapt quickly to various scenarios. By capturing how individuals change their trajectories based on the movement of their neighbors or correspondence to the specific group, the model achieves more natural predictions than approaches that ignore social dynamics and common sense rules. Additionally, the LSTMs in the core can process long sequential data, modeling the temporal dependencies in human trajectories and enabling it to predict future positions more effectively. Lastly, it uses an end-to-end learning approach, eliminating the need for hand-crafted features, and the model can also be extended to other domains involving multi-agent systems.

However, using a social pooling layer increases the computational burden of the model, especially in environments with high pedestrian density, since processing interactions among many pedestrians can result in significant resource consumption. Social LSTM is a deep learning approach often regarded as a "black box," meaning that understanding why the model predicts specific trajectories can be challenging, if even possible. Although it utilizes the existing human trajectory datasets and does not require additional annotations, the model's accuracy fully depends on the training data's quality and diversity and can result in poor generalization. Finally, the model cannot capture the dependencies between multiple correlated sequences.

1.5 Improvements

Yu et al.[8] used an improved version of Social LSTM in order to recreate realistic crowd dynamics and model complex pedestrian behaviors at multiple levels. Addressing the problem of agents being treated as point particles in Social LSTM, which led to limitations in modeling naturalistic local interactions in crowds, they started handling agents as disks with specific radii, creating more accurate and realistic local interaction behavior. Since the Social LSTM may experience performance issues in unseen scenarios, the researchers integrated an anticipatory collision avoidance model, which predicts potential collisions and adjusts agent trajectories to mitigate abrupt velocity changes or trajectory jitter.

Han et al.[9] proposed notable improvements to the Social LSTM model to address its limitations in real-world applications, particularly in predicting pedestrian trajectories for vehicle collision avoidance systems since the original method lacks explicit mechanisms to account for static and dynamic environmental obstacles. The authors integrated the YOLOv5 detection model to reduce target loss and enhance performance, as well as the DeepSORT tracking algorithm to lower the number of target transformations. To address the limitations of pixel-based trajectory predictions in the Social LSTM and allow accurate identification of collision points, the researchers introduced Perspective Transformation and Direct Linear Transformation, which correct distortions caused by camera angles and convert predicted trajectories from pixel coordinates to real-world coordinates.

2 Social GAN

Unless stated otherwise, all the figures in this section are taken from [10].

2.1 Introduction

Despite significant progress in trajectory prediction, existing methods have two main limitations. First, they mainly focus on the local neighborhood of each agent, failing to capture the broader context of the entire scene when making decisions. Second, most models tend to learn the "average behavior" due to using L2 loss functions, resulting in a single predicted outcome rather than considering multiple possibilities.

To address these challenges, the Social GAN (SGAN) [10] is introduced, a model that leverages generative adversarial networks (GANs) with a social pooling mechanism.

2.2 Architecture

In a typical GAN setup, the generator creates new data from random noise to resemble real data, while the discriminator assesses the authenticity of these samples by comparing them to actual data. The generator and discriminator are trained together in a competitive manner, with the generator aiming to produce more realistic outputs to mimic the real data. In contrast, the discriminator tries to correctly distinguish between actual and generated data [11].

In the context of trajectory prediction, the generator in SGAN consists of an RNN-based encoder-decoder framework augmented with a pooling module. The encoder uses LSTMs to process the observed trajectories for each agent, and the pooling module captures social interactions. The decoder combines the pooled information with noise vectors sampled from $N(0,1)$ to generate diverse and socially acceptable future trajectories. The discriminator, an RNN-based encoder, processes the trajectories using LSTMs to assess their realism by comparing them against true trajectories. The model architecture is visualized in Figure 13.

Social GAN introduces two key novelties to improve trajectory prediction:

- **Variety loss**: Unlike the typical L2 loss, which minimizes the difference between predicted and ground-truth values, the SGAN uses adversarial loss to encourage the generator to learn the distribution of socially acceptable behaviors. This allows the model to generate multiple plausible trajectories for each agent rather than a single deterministic path. To find the best outcome, variety loss is introduced. For a scene, k (a hyperparameter) candidate trajectories are generated, and then the best one is selected based on the L2 distance to the ground truth. The variety loss function is defined as:

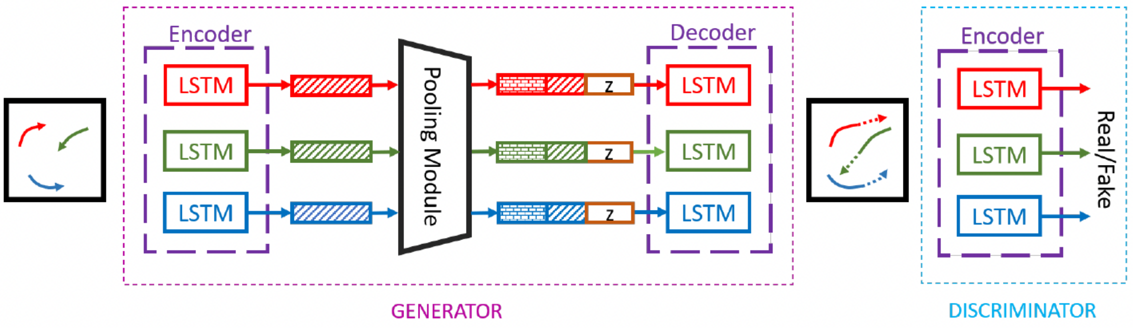


Figure 13: Model architecture of the Social GAN.

$$L_{\text{variety}} = \min_k \|Y_i - \hat{Y}_i^{(k)}\|^2$$

where Y_i is the ground truth trajectory for the i-th agent, and $\hat{Y}_i^{(k)}$ represents the k-th candidate trajectory generated by the model for the same agent. This encourages the model to explore a range of potential outcomes while selecting the most fitting trajectory for each agent.

- **Global pooling mechanism**: A new global pooling method is introduced to model interactions across the entire scene. Consider the red person in Figure 14. Unlike traditional methods like social pooling, which focus only on local neighborhoods (the red grid around the red person), this approach computes the relative positions (distances) between the target agent and all other agents in the scene. These relative positions are then concatenated with each agent's hidden state, processed through a multi-layer perceptron (MLP), and then aggregated using max pooling to create a compact representation. The pooled vector, which is translation-invariant due to the use of relative positions, summarizes all the information needed for the agent to make a decision, effectively capturing both local and global interactions.

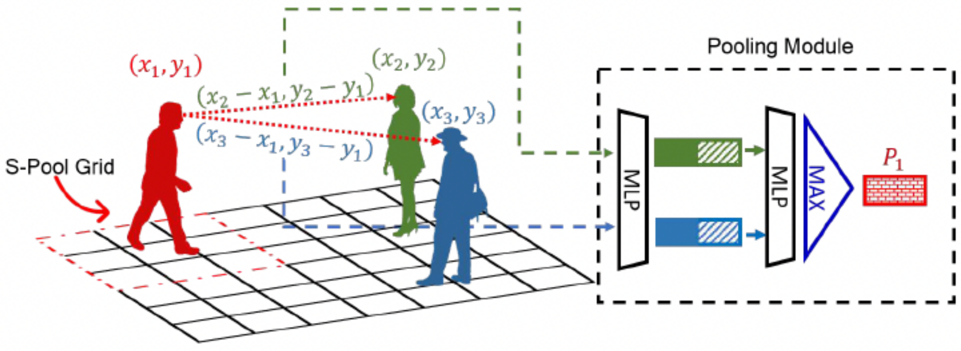
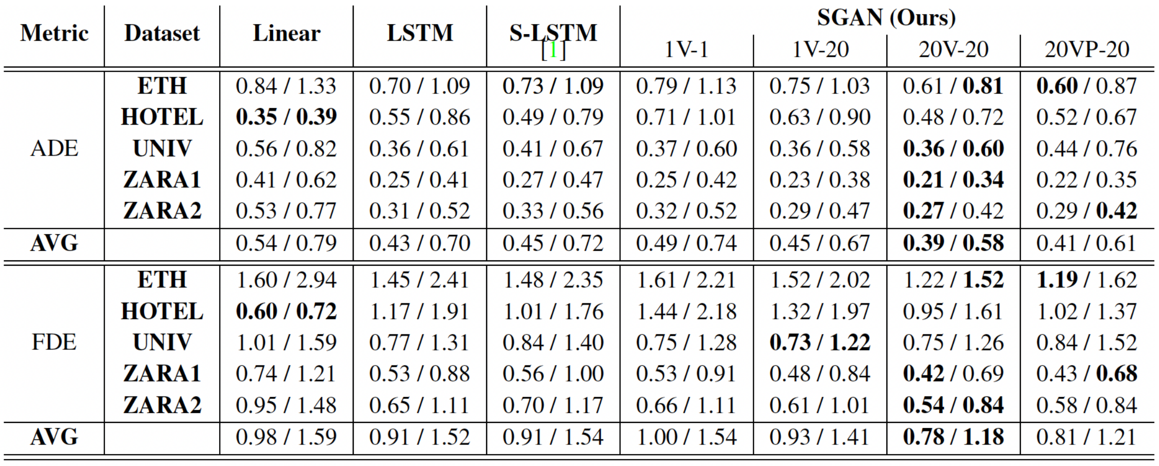


Figure 14: Comparison between social pooling (red grid) and global pooling mechanism (pooling module).

2.3 Experiments and Results

Experiments were conducted on the ETH [12] and UCY [13] datasets, comprising five subsets of real-world pedestrian trajectories from four crowded scenes. Performance was evaluated using ADE and FDE.

The SGAN model achieved state-of-the-art performance, outperforming baseline methods, including linear regression, standard LSTM without pooling layers, and the Social LSTM (S-LSTM) [7], which is essentially the same as our model but uses a social pooling layer instead. SGAN achieved the best overall performance in four out of five subsets, with the linear regression model outperforming it in one case. On average, SGAN performed best with $k = 20$ and no pooling layer, demonstrating the lowest ADE and FDE scores. While the complete model with pooling showed slightly lower performance, the ablation study confirms its role in predicting more socially plausible trajectories. Full results are in Figure 15.



Metric	Dataset	Linear	LSTM	S-LSTM [1]	SGAN (Ours)			
					1V-1	1V-20	20V-20	20VP-20
ADE	**ETH**	0.84 / 1.33	0.70 / 1.09	0.73 / 1.09	0.79 / 1.13	0.75 / 1.03	0.61 / **0.81**	**0.60** / 0.87
	HOTEL	**0.35 / 0.39**	0.55 / 0.86	0.49 / 0.79	0.71 / 1.01	0.63 / 0.90	0.48 / 0.72	0.52 / 0.67
	UNIV	0.56 / 0.82	0.36 / 0.61	0.41 / 0.67	0.37 / 0.60	0.36 / 0.58	**0.36 / 0.60**	0.44 / 0.76
	ZARA1	0.41 / 0.62	0.25 / 0.41	0.27 / 0.47	0.25 / 0.42	0.23 / 0.38	**0.21 / 0.34**	0.22 / 0.35
	ZARA2	0.53 / 0.77	0.31 / 0.52	0.33 / 0.56	0.32 / 0.52	0.29 / 0.47	**0.27** / 0.42	0.29 / **0.42**
AVG		0.54 / 0.79	0.43 / 0.70	0.45 / 0.72	0.49 / 0.74	0.45 / 0.67	**0.39 / 0.58**	0.41 / 0.61
FDE	**ETH**	1.60 / 2.94	1.45 / 2.41	1.48 / 2.35	1.61 / 2.21	1.52 / 2.02	1.22 / **1.52**	**1.19** / 1.62
	HOTEL	**0.60 / 0.72**	1.17 / 1.91	1.01 / 1.76	1.44 / 2.18	1.32 / 1.97	0.95 / 1.61	1.02 / 1.37
	UNIV	1.01 / 1.59	0.77 / 1.31	0.84 / 1.40	0.75 / 1.28	**0.73 / 1.22**	0.75 / 1.26	0.84 / 1.52
	ZARA1	0.74 / 1.21	0.53 / 0.88	0.56 / 1.00	0.53 / 0.91	0.48 / 0.84	**0.42** / 0.69	0.43 / **0.68**
	ZARA2	0.95 / 1.48	0.65 / 1.11	0.70 / 1.17	0.66 / 1.11	0.61 / 1.01	**0.54 / 0.84**	0.58 / 0.84
AVG		0.98 / 1.59	0.91 / 1.52	0.91 / 1.54	1.00 / 1.54	0.93 / 1.41	**0.78 / 1.18**	0.81 / 1.21

Figure 15: Quantitative results in meters across subsets: ADE and FDE for prediction timesteps of 8 and 12. The full method is referred to as SGAN-kVP-N, where kV indicates whether variety loss was used ($k = 1$ means no variety loss), P signifies the use of the pooling module, and N represents the number of samples taken during testing. The best prediction is selected based on the L2 distance at test time.

In terms of speed, SGAN outperformed the S-LSTM baseline. While a standard LSTM was the fastest, it lacked accuracy in multi-modal predictions and collision avoidance. SGAN was 16 times faster than S-LSTM, allowing 20 samples to be generated in the time S-LSTM produced one. This significant difference is due to SGAN not requiring pooling at each time step and its simpler pooling mechanism than S-LSTM's more complex occupancy grid.

The effect of diversity was also explored by comparing SGAN-1V-N (without variety loss) and SGAN-NV-N (with variety loss). The results showed that adding more samples to a model trained without variety loss did not immensely improve accuracy. However, using variety loss (with k=100) led to a performance increase of 33%, highlighting the importance of diversity in the generated predictions.

An ablation study showed how crucial the pooling module is in modeling human interactions like collision avoidance, group behavior, and person following. Comparisons showed that pooling enforces social norms, leading to more plausible trajectories, such as yielding the right of way and avoiding collisions. It also improves predictions in scenarios like merging, group avoidance, and following behavior by adjusting speed and direction appropriately.

2.4 Advantages and Disadvantages

Overall, the SGAN generates socially acceptable and diverse trajectories while achieving state-of-the-art performance in trajectory prediction. Its pooling module helps capture interactions between pedestrians, and the variety loss allows the model to generate multiple plausible trajectories for each person. This approach provides better predictions than traditional methods, making it highly effective for trajectory prediction tasks.

The model has several advantages. It produces socially plausible predictions by capturing human interactions such as group behavior and group following. The pooling mechanism enforces social norms, leading to more realistic and acceptable paths, and the model demonstrates robustness in various scenarios, including obstacle avoidance and navigating crowded areas. Furthermore, it performs well on real-world datasets, aligning closely with realistic human movement patterns.

Despite its strengths, the model has some drawbacks. While improving social behavior, the pooling mechanism slightly reduces performance in quantitative metrics such as ADE and FDE. Its added complexity might make tuning and optimizing harder compared to simpler models. Lastly, even though SGAN helps with collision avoidance, collisions can still happen frequently and remain a significant problem due to weaknesses in the discriminator design, which fails to fully model human-human interactions and differentiate real trajectories from fake ones. This limitation will be addressed by the model in the following subsection.

2.5 Improvements

An upgraded version of SGAN, called SGANv2 [14], was introduced a few years later with several significant improvements to enhance the model's prediction accuracy, address challenges like safety in crowded environments, and ensure more realistic trajectory prediction. The key improvements in the model include:

- **Spatio-temporal interaction modeling:** Spatio-temporal interaction refers to the dynamics of how pedestrians move through space over time. Unlike SGAN, which primarily focuses on spatial interactions, spatio-temporal modeling incorporates both spatial and temporal dynamics to better capture how pedestrian movement evolves. This modeling is integrated into both the generator and discriminator, enhancing the discriminator's ability to distinguish between real and generated trajectories.
- **Collaborative sampling strategy:** To prevent generating a narrow range of outcomes, SGANv2 introduces collaborative sampling [15] at test time. Usually, we sample from the generator and discard the discriminator during testing. However, SGANv2 leverages the trained discriminator by using its gradients—without updating the generator's parameters—to refine poor predictions (e.g., in cases of collisions) (see Figure 16). By incorporating the discriminator during sampling, SGANv2 increases prediction variety, producing more socially acceptable and realistic outputs while reducing the chance of mode collapse, where the generator produces limited and similar outputs, ignoring the diversity of the target distribution.
- **Transformer-based discriminator:** SGANv2 uses a transformer-based discriminator with an attention mechanism to improve the generator's learning process since transformers have been proven to be able to outperform RNNs in most of the sequence modeling tasks, which also include trajectory prediction [16]. The transformer handles temporal sequences and better models long-range dependencies in time-series data. This results in a discriminator that more accurately distinguishes between realistic and unrealistic trajectories, significantly improving the quality of generated outputs, even in scenarios with complex, long-term interactions.

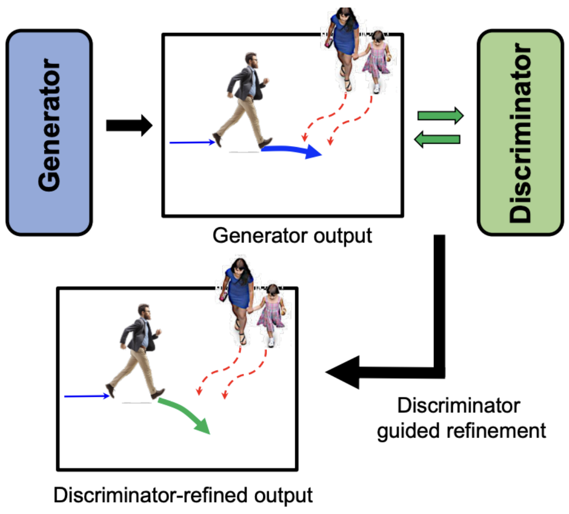


Figure 16: Collaborative sampling strategy used in SGAN v2 [14]

SGANv2 demonstrated significant improvements over the original SGAN in prediction accuracy and safety. Using the same data subsets as SGAN did, SGANv2 outperformed SGAN and other models (e.g., LSTM, S-LSTM, transformer) in terms of ADE and FDE in four out of five cases. The authors also introduced a new metric called prediction collision, which indicates the probability that the predicted trajectories of a primary pedestrian will collide with the trajectories of surrounding pedestrians in the future scene. SGANv2 consistently achieved the lowest probability across all subsets, with a collision rate of around 1-2%, while SGAN exhibited higher rates, ranging from 2.2% to almost 12%.

2.6 Future Work

Future work can incorporate adaptive attention mechanisms in both SGAN and SGANv2 to better capture dynamic pedestrian interactions, especially in highly crowded or complex environments. Additionally, introducing reinforcement learning to encourage long-term consistency in trajectory generation could address challenges like unrealistic or abrupt changes. Finally, leveraging multimodal data, such as environmental context (e.g., maps, obstacles), could make the predictions more robust and context-aware, further improving real-world applicability.

3 Behavior-CNN

3.1 Introduction

Before Behavior-CNN, pedestrian behavior prediction relied primarily on agent-based models (like Social Force Model and ORCA) and optical flow analysis. While interpretable, these approaches struggled with complex crowd dynamics, particularly in handling occlusions and crossing paths. Behavior-CNN addresses these limitations through a novel deep learning approach that uses a "displacement volume" encoding scheme to effectively capture spatial relationships between pedestrians while maintaining robustness in dense crowd scenarios.

3.2 The Behavior-CNN Innovation

What sets Behavior-CNN apart is its innovative approach to encoding pedestrian behavior. While previous deep learning approaches like Social LSTM [7] focused on sequence modeling, Behavior-CNN introduced a novel "displacement volume" concept.

The displacement volume encoding scheme uses M uniformly sampled time points from previous frames $(t_1, ..., t_M)$, where t_M represents the current time point. For each pedestrian p_i at time t_m, the model encodes their normalized spatial location as:

$$l_i^m = [x_i^m/X, y_i^m/Y] \tag{1}$$

where x_i^m and y_i^m represent spatial coordinates, and $[X, Y]$ defines the spatial size of input frames. This encoding creates a 2M-dimensional displacement vector that captures the complete walking path information while maintaining spatial relationships [17]. In the implementation, $M = 5$ time points are uniformly sampled with an interval of 20 frames (0.8 seconds), providing a balance between historical information and computational efficiency.

3.3 Network Architecture Details and Implementation

The Behavior-CNN architecture introduces several inventive design choices that differentiate it from conventional CNN approaches. Figure 17 illustrates the complete pipeline, where input walking paths (a) are first encoded into displacement volumes (b), processed through the Behavior-CNN network (c) to generate predicted displacement volumes (d), which are then decoded into future walking paths (e). At its core, the network processes pedestrian movement through this carefully structured pipeline of encoding, analysis, and prediction [18].

The Behavior-CNN architecture is structured as a carefully designed hierarchy of convolutional layers, as illustrated in Figure 18. The network follows a CNN encoder-decoder structure, with a learnable location bias map added at the bottleneck layer. The location bias map enables the network to adapt predictions based on spatial position within the scene. A deconvolution layer in the decoder ensures the output matches input dimensions.

Building upon this foundation, as shown in Figure 18(c), the network incorporates a location bias map that enables it to automatically adapt predictions based on spatial location (e.g., entrances and open spaces). The decoder structure, illustrated in Figure 18(d-e), processes these features to generate the final prediction volume.

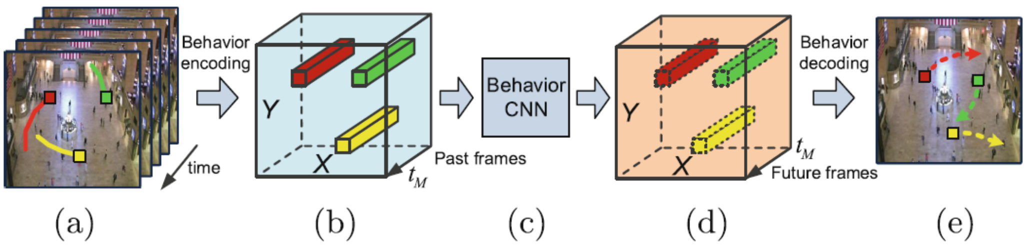


Figure 17: System flowchart showing the complete Behavior-CNN pipeline: (a) Input pedestrian walking paths from previous frames, (b) Encoded displacement volume representing pedestrian movements, (c) Behavior-CNN processing stage, (d) Predicted displacement volume for future frames, (e) Decoded future pedestrian walking paths.

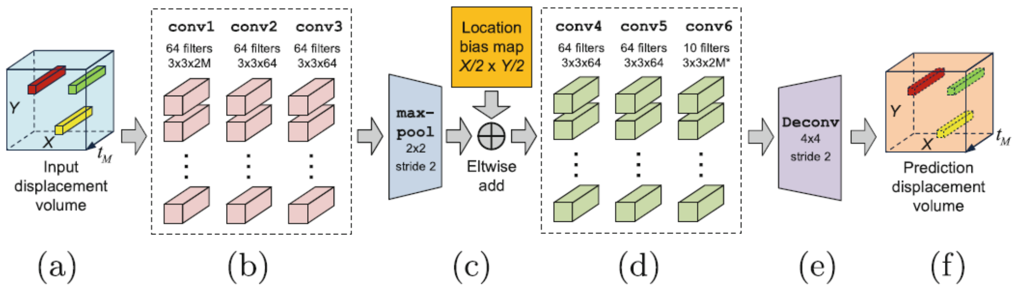


Figure 18: Behavior-CNN architecture: (a) Input displacement volume, (b) Three bottom convolution layers, (c) Max-pooling and location bias addition, (d) Three top convolution layers, (e) Deconvolution layer, (f) Output prediction displacement volume.

3.4 Learning Process and Training Strategy

At the heart of the training process lies a specialized loss function that focuses exclusively on valid entries, ignoring the numerous zero-value regions that naturally occur in sparse movement data. The optimization process employs stochastic gradient descent, with parameters carefully tuned to balance learning speed with stability. This combination proves particularly effective in handling the unique characteristics of pedestrian movement data while maintaining robust learning performance.

The loss function is mathematically expressed as:

$$Loss = \frac{1}{\sum B} \|(D^* - \hat{D}^*) \circ B\|_2^2 \tag{2}$$

where D^* represents ground truth movements, $\hat{D}^*$ denotes predicted movements, and B is a binary mask ($B \in \{0, 1\}$) identifying valid entries. The operator $\circ$ represents the Hadamard (element-wise) product between matrices. The denominator $\sum B$ counts the total number of valid (non-zero) entries for normalization. This formulation ensures the network focuses its learning on actual pedestrian movements by masking out background regions and normalizing only over valid pedestrian locations, thus appropriately handling the sparse nature of the data.

3.5 Location Awareness Properties

A distinguishing feature of Behavior-CNN is its sophisticated location awareness capability, which manifests through multiple complementary mechanisms. As demonstrated in Figure 19, the network develops learned bias maps that capture subtle variations in movement patterns across different regions of the scene. The scene is divided into 8×8 grids, with each region showing distinct movement patterns (Figure 19(b)). For example, the ”crossing” grid exhibits three primary movement directions (up, down, and left), while the ”corridor” grid shows predominantly bidirectional (up-down) traffic patterns.

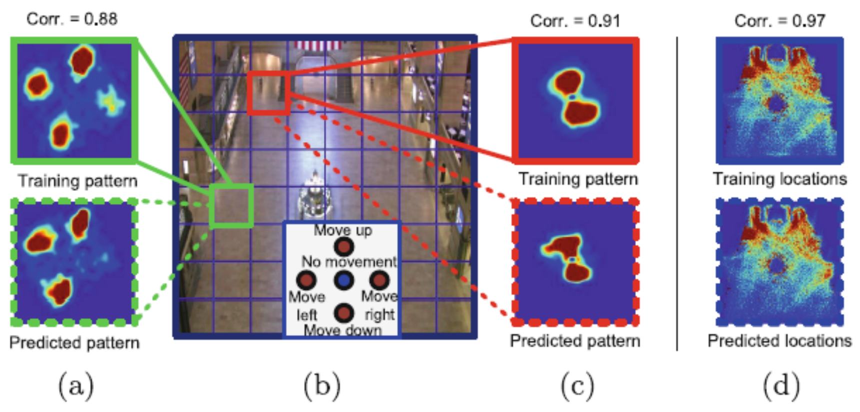


Figure 19: Investigation on location awareness of Behavior-CNN: (a-c) Behavior-CNN captures different motion patterns for different regions of the scene. The scene is segmented into 8×8 grids in (b). The motion patterns of training samples and prediction results of the ”crossing” grid (green) is shown in (a) and those of the ”corridor” grid (red) are shown in (c). Warmer color indicates higher frequency of corresponding motion. (d) Correlation between the spatial distributions of training samples and predictions, reflecting scene structure understanding.

The preservation of spatial correlations throughout the network ensures that relationships between different areas remain intact during processing. The network achieves strong correlations between predicted and actual movement patterns 0.88 for crossing regions and 0.91 for corridor regions (Figure 19(a,c)). Furthermore, as shown in Figure 19(d), the model successfully learns to identify and adapt to scene-specific features such as obstacles and common gathering areas.

This location awareness proves particularly valuable in real-world applications, where pedestrian behavior often varies significantly based on environmental features such as entrances, exits, obstacles, and common gathering areas. The network's ability to learn and adapt to these spatial characteristics contributes significantly to its superior prediction performance compared to traditional approaches.

3.6 Experiments and Results

Behavior-CNN was evaluated on two primary datasets. Dataset I contains 12,684 annotated pedestrians over 4,000 seconds of video, while Dataset II comprises 797 pedestrians with complete trajectory annotations. The evaluation used both human-annotated ground truth and KLT tracker output to demonstrate robustness to different input sources.

The model achieved a mean squared error (MSE) of 2.421% on Dataset I with human annotations and 2.517% with KLT tracking input. These results significantly outperformed baseline approaches:

- Constant velocity model: 6.091%
- Social Force Model: 4.280%
- Linear Trajectory Avoidance: 4.723%
- Temporal Information Model: 4.075%

Behavior-CNN demonstrated exceptional versatility across multiple applications. In destination prediction, it achieved 53% top-1 accuracy and 84% top-3 accuracy, surpassing existing energy map modeling approaches.

The model's ability to handle long-term prediction was demonstrated through recurrent forward propagation, a technique where the network's predictions are iteratively fed back as input to predict further into the future. Specifically, after predicting the next 4 seconds of movement, these predictions are encoded into a new displacement volume and used as input for the next prediction cycle. This recursive process enabled accurate prediction of pedestrian paths up to 15 time points into the future, proving particularly valuable for complex scenarios involving multiple pedestrian interactions and obstacle avoidance.

3.7 Improvements

When integrated with the KLT tracker, Behavior-CNN significantly improves tracking performance by providing behavioral priors for trajectory association. As shown in Figure 20, the proposed method (red dots) successfully maintains correct and complete trajectories where traditional methods like RFT (blue dots) fail due to wrong associations or lost targets.

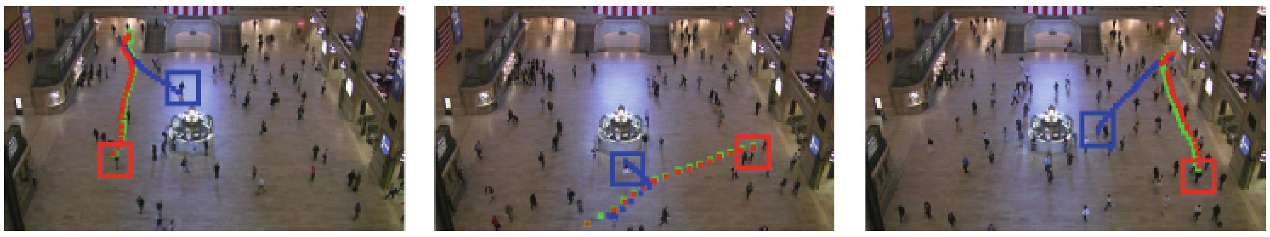


Figure 20: Improved pedestrian tracking results comparing Behavior-CNN (red dots) with RFT (blue dots). Ground truth trajectories are shown as green dots. Successfully tracked pedestrians using our method and mis-tracked pedestrians by RFT are highlighted by red and blue rectangles respectively.

Quantitatively, Behavior-CNN reduces the average L2 distance error from 411.71 (basic KLT) and 228.33 (KLT+RFT) to 83.79, demonstrating substantial improvement in tracking accuracy.

3.8 Advantages and Disadvantages

While Behavior-CNN shows remarkable performance, several limitations warrant consideration. The model's scene-specific training nature means that transfer between significantly different environments requires retraining. Additionally, the fixed input/output time windows may limit flexibility in some applications. The computational requirements, while manageable, are higher than simpler approaches, particularly during the training phase.

3.9 Future Work

While Behavior-CNN presents significant advances in pedestrian behavior prediction, several opportunities for enhancement remain. Scene understanding could be deepened by incorporating semantic information about the environment, like distinguishing between walkways, obstacles, and gathering areas, to inform predictions. The current scene-specific nature of the model limits its applicability across different environments; developing transfer learning approaches could enable the model to adapt more readily to new scenes while preserving its learned behavioral patterns.

The temporal aspects of prediction also warrant further exploration. Currently limited to fixed time windows, extending the prediction horizon while maintaining accuracy would significantly benefit long-term planning applications. Additionally, the model could be enriched by considering social dynamics more explicitly like recognizing group behaviors, social interactions, and how environmental context influences movement decisions. These extensions would move us closer to truly comprehensive pedestrian behavior understanding.

4 Comparison Among Deep Learning Models

Among the three deep learning models discussed here, Social GAN can be seen as an extension of Social LSTM. Behavior-CNN, due to its relatively simple design, often enables higher inference efficiency. However, CNN models' accuracy is usually outperformed by RNN models in terms of sequential data, so CNN models have gained less popularity in recent years. Nevertheless, they remain a fast, lightweight solution to pedestrian trajectory prediction in small- and middle-scale scenarios. Since all the information is integrated into one tensor via *Behavior Encoding* in Behavior-CNN, there is no need to design an extra information integration block, which facilitates the deployment of this model.

For RNN models, typically, each pedestrian has its own RNN that processes its trajectory. The key problem is integrating the neighborhood information or global information into these individual RNNs. Social LSTM chooses to let each individual LSTM receive neighborhood information with a high frequency, while Social GAN focuses more on global information and makes the exchange of information less frequent. Since there are usually multiple acceptable solutions to the pedestrian trajectory problem, Social GAN tries to use the GAN structure to produce a set of potential trajectories and pick the best one. This makes Social GAN more accurate but simultaneously increases the computational burden to some extent, despite the fact that it's still faster than vanilla Social LSTM.

It is also worth noting that for computational efficiency and memory usage reasons, individual RNNs usually share the same parameters. If the scenario contains highly diverse pedestrian characteristics, RNN models can fail to capture this diversity.

These comparisons are summarized in Table 2.

Feature	**Behavior-CNN**	**Social LSTM**	**Social GAN**
Architecture	CNN	RNN (LSTM)	GAN-based RNN (LSTM)
Information Integration	Behavior Encoding (single tensor)	Frequent, localized information sharing to individual LSTMs	Less frequent, global information sharing; GAN-based trajectory selection
Accuracy	Lower than RNN models for sequential data	Higher than CNN, but lower than Social GAN	Highest accuracy among the three
Computational Efficiency	Highest (fast and lightweight)	Low	Moderate
Memory Usage	Low	Moderate	High
Strengths	Simple design, efficient inference, easy deployment	Captures sequential data better than CNNs	Generates diverse potential trajectories, selects the best one, captures global context
Weaknesses	Lower accuracy for sequential data, may not capture complex social interactions	Struggles with diverse pedestrian characteristics, can be computationally expensive with frequent updates	memory intensive, training takes more efforts
Suitable Scenarios	Small- and middle-scale scenarios, resource-constrained applications	General pedestrian trajectory prediction, moderate complexity	Scenarios requiring high accuracy, complex social interactions, less constrained by computational resources

Table 2: Comparison of Deep Learning Models for Pedestrian Trajectory Prediction

---

Report on task 3: Comparison Between KB and DL Models

Generally speaking, KB models are mainly used for large-scale crowd simulation, where we focus on the accuracy of group behavior instead of individual behavior. This is because the rules in KB methods reflect most of the time the *average* behaviors, while an individual pedestrian can have a lot of random behaviors such as a sudden halt or change of direction. DL methods, on the contrary, can capture these more subtle features of individual pedestrians. However, this comes at a price that the computational burden of DL methods is higher, and as a result, they are mostly used in local trajectory predictions. Since KB models are based on empirical assumptions, we need to verify the result by comparing it with the collected real-life data, and calibrate the parameters if needed. DL methods are fully data-driven, so once the parameters are determined after the training process, they will not be changed anymore. As a consequence, although DL models can be more precise in certain scenes, their generalization ability is highly dependent on the diversity of the dataset, and if in the dataset there are similar scenes with significantly different pedestrian behaviors (e.g., the same subway station in rush hour and holiday), the models can be confused and yield an unexpected result.

Another perspective is interpretability. KB models are interpretable because the rules and the parameters are all defined by users themselves. For example, in the Social Force Model, we can increase the coefficient controlling the attraction to the goal so that the pedestrians are more goal-oriented or increase the coefficient of the repulsive force among the pedestrians so that the pedestrians tend to shy away from approaching others. DL models do not have this interpretability because the parameters in such models are only used to approximate the behavior in the provided dataset. However, if we only want a model that covers a small number of specific scenes, DL models can significantly outperform KB methods. For example, in autonomous driving, a car needs to predict the upcoming movement of pedestrians to avoid collision. When a group of pedestrians are crossing the road, some may walk faster, some may walk more slowly, and their speed may also rely on the time left before the traffic light turns red. In such situations, the parameters of KB models are hardly flexible enough to capture all these elements. In contrast, DL models, based on the individual behavior of the pedestrians, are much more reliable.

These comparisons are summarized in Table 3.

Feature	**Knowledge-Based (KB) Models**	**Deep Learning (DL) Models**
Focus	Group behavior accuracy	Individual behavior accuracy
Computational Cost	Lower	Higher
Typical Use	Large-scale crowd simulation	Local trajectory prediction, specific scene simulations (e.g., autonomous driving)
Parameter Tuning	Requires manual calibration and comparison with real-life data	Parameters fixed after training, no further manual adjustment
Generalization	Less dependent on dataset, but may be less precise	Highly dependent on dataset diversity, prone to unexpected results in dissimilar scenes
Interpretability	High (rules and parameters defined by users)	Low (parameters approximate behavior, not directly interpretable)
Performance in Specific Scenes	Can be less effective for complex individual behaviors	Can significantly outperform KB models when trained on relevant data
Example Use Cases	General crowd flow in a stadium	Pedestrian behavior prediction for autonomous driving, capturing nuanced individual actions

Table 3: Comparison of Knowledge-Based (KB) and Deep Learning (DL) Crowd Simulation Models

---

Report on task 4: Hybrid Models

Hybrid models combine the strengths of KB and DL approaches to improve trajectory prediction. While KB models ensure physical plausibility and interpretability, DL models capture complex patterns and interactions. By integrating these two approaches, both accuracy and generalization can be improved, leading to more reliable predictions in diverse scenarios. This section explores different hybrid strategies and their effectiveness in trajectory forecasting.

1 Hybrid Model with Realistic Residual Block

Unless stated otherwise, all the figures in this section are taken from [19].

1.1 Introduction

To address the limitations of pure KB and DL methods, Bahari et al. introduced the realistic residual block (RRB) [19], a hybrid framework that is designed specifically for vehicle trajectory prediction and combines KB models as a foundation with a data-driven component to learn deviations. This approach ensures more realistic, feasible, and accurate trajectory predictions by leveraging the strengths of both paradigms.

1.2 Architecture

The model introduces a structured way to inject knowledge into DL models. The core of the method is the realistic residual block (RRB), which refines the predictions by learning residuals (the difference between predicted and actual outcomes). By focusing on learning only the residual corrections, the model improves accuracy while reducing the risk of overfitting to unnecessary complexities. The term *realistic* refers to the incorporation of real-world constraints, such as traffic flow, speed limits, lane changes, and vehicle interactions. These contextual constraints make the model's predictions more robust in reflecting driving behavior dynamics. The KB trajectory is formed using domain knowledge like road lanes, treated probabilistically with fixed variance based on training data. The RRB then learns complex interactions and adjusts this KB trajectory by adding data-driven residuals.

The inverse variance weighted (IVW) addition step ensures that the final trajectory is a weighted sum of the KB trajectory and residuals, minimizing uncertainty and prioritizing more certain predictions. Mathematically, the merged trajectory y_{ref} is expressed as:

$$y_{ref} = w y_{kd} + \tilde{w} y_{ad}$$

where y_{kd} represents the knowledge-driven trajectory, and y_{ad} is the adjusted trajectory including residuals. The weights w and $\tilde{w}$ are determined by minimizing the total variance of the final prediction. The weight matrices are calculated as follows:

$$w = \begin{bmatrix} \frac{\sigma_{ad,11}}{\sigma_{kd,11}+\sigma_{ad,11}} & \frac{\sigma_{ad,11}}{\sigma_{kd,11}+\sigma_{ad,11}} \\ \frac{\sigma_{ad,22}}{\sigma_{kd,22}+\sigma_{ad,22}} & \frac{\sigma_{ad,22}}{\sigma_{kd,22}+\sigma_{ad,22}} \end{bmatrix}, \quad \tilde{w} = \begin{bmatrix} \frac{\sigma_{kd,11}}{\sigma_{kd,11}+\sigma_{ad,11}} & \frac{\sigma_{kd,11}}{\sigma_{kd,11}+\sigma_{ad,11}} \\ \frac{\sigma_{kd,22}}{\sigma_{kd,22}+\sigma_{ad,22}} & \frac{\sigma_{kd,22}}{\sigma_{kd,22}+\sigma_{ad,22}} \end{bmatrix}$$

where σ_{kd} and σ_{ad} are the covariance matrices of the KB trajectory and adjusted residual trajectory, respectively, and $\sigma_{kd,ad}$ is the cross-covariance matrix between these two trajectories. This process ensures that the final trajectory y_{ref} incorporates the most certain predictions, as the residuals are weighted based on their uncertainties.

Additionally, the model is extended to multimodal prediction using the winner-takes-all (WTA) loss function, which helps select the trajectory closest to the ground truth during training. The WTA loss function is defined as:

$$l(\theta) = \sum_{n=1}^{N} \sum_{m=1}^{M} 1(m = m^*) \left[\log p(x_{n,m} | S_n, \mu_{ref}, \sigma_{ref})\right]$$

where N and M represent the number of samples and modes respectively, and m^* is the mode closest to the ground truth in terms of ℓ_2 distance. In this context, "mode" refers to a distinct possible future trajectory among multiple plausible trajectories predicted by the model.

Finally, model predictive control (MPC) is employed to ensure the kinematic feasibility of the predictions. Instead of computing control commands post-prediction, the model directly estimates future positions under physical constraints. The optimization problem for MPC is formulated as:

$$y_t^p = \arg \min_{s_{t:t+T_p}, u_{t:t+T_p}} \sum_{j=1}^{T_p} \left\| s_{t+j}[0:1] - y_{ref,t+j} \right\|^2 + \lambda \left\| u_{t+j} - u_{t+j-1} \right\|^2$$

subject to the system dynamics:

$$s_{t+1} = F_{bic}(s_t, u_t), \quad s_0 = s_{\text{init}}, \quad u_{\min} \leq u_t \leq u_{\max}$$

where $s_t = [x_t, y_t, \phi_t, v_t]$ denotes the state parameters for the vehicle at time t (position, orientation, and velocity), and $u_t = [a_t, \gamma_t]$ represents the control parameters (acceleration and steering angle). The MPC solves for the optimal control inputs u_t that minimize the trajectory error while satisfying the vehicle's dynamic constraints. The architecture of the model, including the interactions between these components, can be seen in Figure 21.

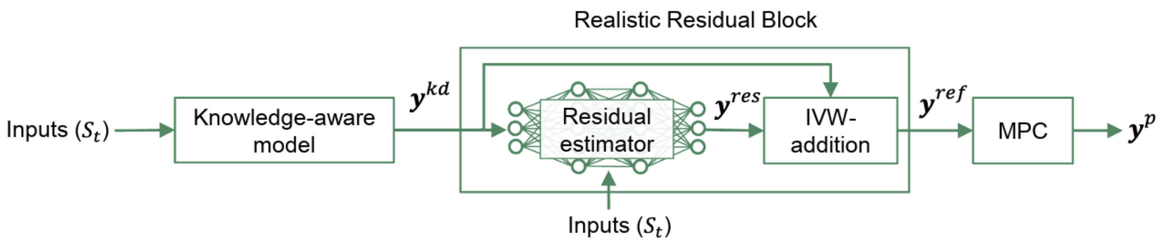


Figure 21: Proposed architecture for the model with RRB

1.3 Experiments and Results

The Interaction dataset [20] is evaluated in two scenarios: scene-overfitting, with 20% reserved for testing to assess interaction generalization, and scene-generalization, where three unseen scenes test model adaptation to new environments.

Baselines include several approaches: the Naive method (e.g., Kalman filter), the KD method (e.g., Constant Velocity [21]), the DL method (e.g., Social LSTM [7]), and mixed approaches (e.g., Road Loss [22]). As for the metrics, ADE and FDE, along with road violation (RV), which measures the percentage of predicted points that fall off the road, and cross track (CT), which measures the distance between the actual destination and the final predicted point, were used.

The RRB model, particularly its variant RRB_M + MPC, outperforms all other models in both scene-generalization and scene-overfitting scenarios. RRB_M refers to the multimodal version of RRB, which allows the model to predict multiple possible outcomes (two modes in this case), while RRB_M + MPC further incorporates MPC to constrain the model's predictions, ensuring more realistic trajectories, particularly in dynamic environments. In scene-generalization, RRB_M + MPC achieves the best ADE/FDE (2.13/5.02) and RV (0), with a slightly higher CT score (1.81) compared to a KD model (1.74). In scene-overfitting, RRB_M leads in ADE (1.49) and ranks second in FDE (3.68). Ablation studies confirm the importance of key model elements, showing that realistic constraints (e.g., c and IVW-addition) are crucial for maintaining reliable, realistic predictions.

KD models generalize well but have lower accuracy, while DL models are more accurate but produce unrealistic predictions in unseen scenes. The proposed RRB bridges this gap by learning residuals to refine knowledge-driven predictions, incorporating a physically constrained IVW approach and MPC for realistic outputs. RRB outperforms other models in both accuracy and generalizability.

1.4 Future Work

Future work can explore RRB with more complex knowledge-driven methods and apply it to diverse agents like pedestrians who follow social rather than road constraints. Integrating knowledge-driven priors like the

Social Force model could enhance pedestrian dynamics modeling. RRB's framework could also be extended to multi-agent environments with vehicles, pedestrians, and cyclists, ensuring safe and accurate predictions.

2 Hybrid Model: Deep Social Force

Unless stated otherwise, all the figures in this section are taken from [23].

2.1 Introduction

As explained in section Social Force Model, SFM [5] is a model using mathematical terms to represent the effects guiding the motion of pedestrian. One of these effects is the effect of repulsive forces, which is critical for determining how the pedestrian will interact with other pedestrians or objects. Although SFM can simulate many aspects of pedestrian motion successfully, there are still situations where it can fail due to the constraints it imposes on the functions describing repulsive forces. For example, it can fail when two pedestrians are moving in opposite directions in a corridor and cannot decide which way to go when they are in front of each other and get stuck. To address these issues, Sven Kreiss introduces the deep social force model (DSFM) [23], which provides a more flexible version of SFM by learning the functions describing repulsive forces with deep neural networks instead of using constrained functions.

2.2 Architecture

In DSFM, instead of using constrained and fixed functions like SFM, it is proposed to learn a function with a multi-layer perceptron (MLP) architecture like the following:

$$V_{\alpha\beta}(b) = \text{Softplus}\, L_{1\times 5}\, \text{Softplus}\, L_{5\times 1}\, b$$

In the MLP structure above, the part $L_{1\times 5}$ represent a linear layer with input dimension 5 and output dimension 1 and vice versa. Note that the layers of MLP are written in reverse order and the subscripts of linear layers denoted with "L" should also be read as "(number of output features) × (number of input features)". These linear layers contain learnable parameters (i.e. weights). The parts denoted as "Softplus" represent the application of Softplus activation function.

By using this MLP, the need to use manually chosen functions, which are constrained and require domain expertise, is removed. This makes the model more flexible. To improve the MLP performance even further, the DSFM model utilizes Fourier Features [24], which is a feature mapping method that allows to capture a high-frequency function even when only low-dimensional information is available. In the case of DSFM, Fourier Features are used with the 3-dimensional input vector:

$$\begin{bmatrix} b & d_\perp & d_{||} \end{bmatrix}^{\mathrm{T}}$$

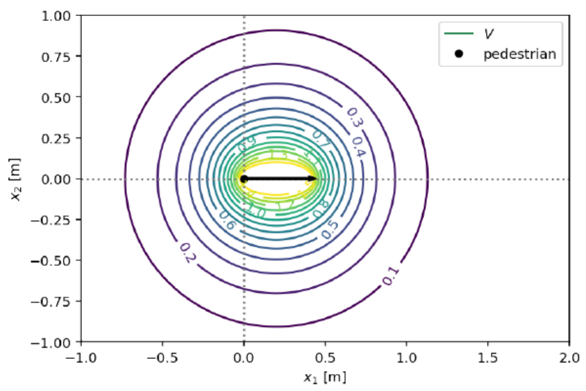
, where $d_\perp$ and $d_{||}$ denote perpendicular and parallel distances respectively. Parameter b was explained in previous section Social Force Model. The utilization of this 3-dimensional input vector is demonstrated in the improved MLP structure shown below.

$$\text{MLP}(b, d_\perp, d_{||}) = \text{Softplus}\, L_{1\times 64}\, [\text{Softplus}\, L_{64\times 64}]^3\, \text{Softplus}\, L_{64\times 192}\, \text{FF}_{192\times 3} \begin{bmatrix} b \\ d_\perp \\ d_{||} \end{bmatrix}$$

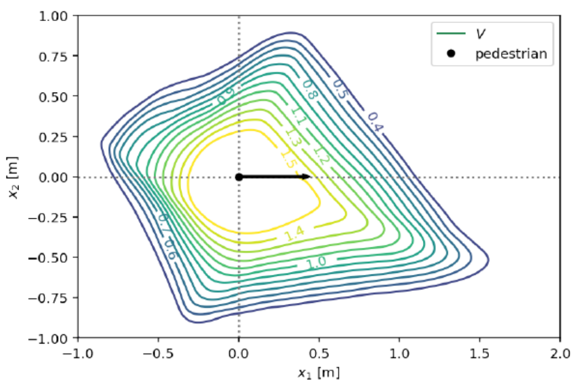
Note that this MLP structure is also written in reverse order. So, when reading from right to left, the 3-dimensional parameter vector introduced previously is given as input and converted to a 192-dimensional vector of Fourier Features first. After that, linear layers and Softplus activation functions are applied subsequently until a single scalar output is produced.

2.3 Experiments and Results

The experiments made with DSFM show that DSFM is able to learn complex repulsive potential shapes while SFM is not. This is demonstrated in Figure 22.



(a) Example of a typical elliptical repulsive potential produced by SFM



(b) Shifted and diamond-like repulsive potential learned by DSFM

Figure 22: Comparison of repulsive potentials used by SFM and DSFM

In SFM, repulsive potential produced by its repulsive potential function usually has an elliptical shape composed of equipotential lines, which can be seen in Figure 22a. The problem with this kind of repulsive potential is that it can cause problems like pedestrians getting stuck when they are in front of each other as they do not have any preferred direction over other with such a shape of a repulsive potential.

Contrary to that, DSFM is able to learn more complex repulsive potential shapes as shown in Figure 22b. It resembles a diamond shifted to the right side. With such a function we can avoid the problem of getting stuck as in SFM. The reason for this is that the potential is shifted right. Therefore, pedestrian will choose to move to the right side when it is in front of another pedestrian in close distance instead of getting stuck like in SFM. The reason why DSFM learns such a shape is that it is trained on data, in which pedestrians' preferred direction of motion is the right side.

The improvements can also be seen in the collective behaviours realized by DSFM. Figure 23 shows the corridor simulation made with DSFM using the same settings and environment as in SFM before, which was demonstrated in Figure 8.

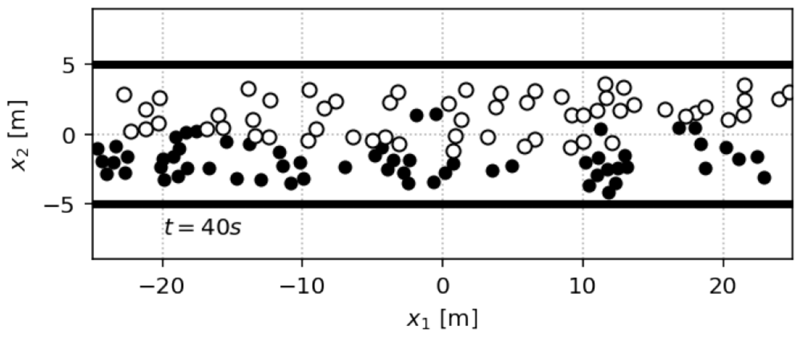


Figure 23: Lane formation behaviour realized with DSFM

The corridor simulation made with SFM in Figure 8 was forming many lanes, whereas in Figure 23 it can be seen that with DSFM each pedestrian group just forms one lane on the right side of their direction of motion, which makes two lanes in total. This is the result of the more complex potential learned by DSFM in Figure 22b, which was shifted to right side. This right shift makes pedestrians prefer to move to right side when they need to perform avoidance. This way, DSFM produces motion behavior more realistic than SFM because in real life, especially in crowded corridors, people usually do not form many lanes in the middle of the corridor. Instead of this, they choose to move toward one side and form a lane on this side.

These comparisons and simulations show that DSFM performs better than the KB model SFM by capturing more complex potential shapes and therefore producing more realistic behaviour of pedestrians. Furthermore, it does not only rely on data like a pure DL method, but uses Fourier Features based on the pre-defined input parameters. One of these parameters (i.e. parameter b) is also used in KB model SFM. This way, DSFM provides both the capability to learn complex behaviours from datasets like a DL method and the ability to guide the model with manually chosen parameters like a KB model by transforming them into Fourier Features in the beginning of MLP.

2.4 Future Work

Future work for DSFM could include applying DL methods not just for repulsive forces but also for attractive forces, which could make the DSFM model more capable of performing more complex grouping behaviors. Moreover, Kreiss [23] argues that DSFM usually performs poorly with data extracted from the real world, and predicting the trajectories beforehand could help the model with such complex data. Another aspect of DSFM that could be improved is the number of useful input parameters given to the MLP with Fourier Features. In the current version of DSFM just 3 input parameters are given to MLP and more of these guiding parameters could make the model even more powerful. Therefore, finding other useful input parameters and comparing their performance could be a research area for improving DSFM.

Report on task 5: Future Work

In this report, we identified that both knowledge-based and deep-learning approaches for pedestrian trajectory prediction and hybrid models show superior performance. This chapter discusses several remaining challenges and opportunities for future research.

One of the most critical areas for future work is improving how models handle environmental context since current approaches consider only fundamental obstacles and boundaries instead of the whole environment. Adding dynamic environmental factors in the models, such as weather conditions and temporary barriers, significantly influences pedestrian behavior in real-world scenarios.

In order to predict the trajectories more realistic, improving sensor noise and occlusions in real-world tracking and model generalization across cultural contexts and urban environments can be taken into account. Additionally, to better reflect real-world performance requirements, further evoluation metrics can be developed.

Another crucial direction involves improving social interaction modeling and understanding complex group movements, which includes forming and splitting groups. Since social relationships strongly influence pedestrians' movement patterns, comprehending complex group dynamics could be particularly valuable in scenarios with significant pedestrians and social groups, such as at the mall or in the city tours. Human feedback and the ability to adapt to changing behavior patterns over time can further improve the existing approaches, which can be realized through integrating interactive learning mechanisms.

The development and improvement of hybrid models is considerable for future research as well. While current hybrid approaches show better performance by combining the strengths of knowledge-based and deep-learning models, there is still room for improvement in how these approaches perform together. Exploring more sophisticated integration methods that preserve knowledge-based components' interpretability while fully leveraging deep learning's pattern recognition capabilities can be helpful for identifying more sophisticated hybrid methods in the future.

A further significant challenge is computation efficiency, particularly for real-time applications, which can include lots of data and have to consider multiple variables in the scenario. This can be addressed by developing hierarchical approaches that dynamically adjust their complexity based on the required precision and available computational resources. Additionally, hardware-specific optimizations could help bridge the gap between academic research and practical applications.

As this report highlights, existing models often struggle with unexpected situations or scenarios that differ significantly from the provided training data. Future models should focus on robust methods for handling uncertainty, including better techniques for quantifying prediction confidence and creating fallback mechanisms for handling edge cases.

Other open issues are ethical considerations and privacy concerns, which can be addressed by developing privacy-preserving methods for collecting and using trajectory data and establishing frameworks for the responsible deployment of the models.

In conclusion, while the field of pedestrian trajectory prediction has been developing rapidly, many improvements and open issues remain for future research. Successfully resolving the mentioned challenges could lead to more robust and realistic models combining both knowledge-based and deep-learning methods for pedestrian trajectory prediction that can be reliably deployed in real-world applications.

References

[1] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[2] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, Heidelberg, 3rd edition, 2008.

[3] Zhipeng Yin, Jinghao Liu, and Lei Wang. Less-effort collision avoidance in virtual pedestrian simulation. pages 488–493, 07 2019.

[4] John Charlton, Luis Rene Montana Gonzalez, Steve Maddock, and Paul Richmond. *Fast Simulation of Crowd Collision Avoidance*, page 266–277. Springer International Publishing, 2019.

[5] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[6] F. Zanlungo, T. Ikeda, and T. Kanda. Social force model with explicit collision prediction. *Europhysics Letters*, 93(6):68005, mar 2011.

[7] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[8] Yingfei Yu, Wei Xiang, and Xiaogang Jin. Multi-level crowd simulation using social lstm. *Computer Animation and Virtual Worlds*, 34(3-4):e2180, 2023.

[9] Yong Han, Xujie Lin, Di Pan, Yanting Li, Liang Su, Robert Thomson, and Koji Mizuno. Pedestrian trajectory prediction method based on the social-lstm model for vehicle collision. *Transportation Safety and Environment*, 6(3):tdad044, 2024.

[10] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[12] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part I 11*, pages 452–465. Springer, 2010.

[13] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3542–3549, 2014.

[14] Parth Kothari and Alexandre Alahi. Safety-compliant generative adversarial networks for human trajectory forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 24(4):4251–4261, 2023.

[15] Yuejiang Liu, Parth Kothari, and Alexandre Alahi. Collaborative sampling in generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4948–4956, 2020.

[16] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*, pages 10335–10342. IEEE, 2021.

[17] Teng Wang, Min Qiao, Zicheng Lin, Chengkang Li, Hichem Snoussi, Zhe Liu, and Chao Choi. Deep learning for video-based crowd behavior analysis. *Visual Informatics*, 2(4):224–230, 2018.

[18] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Pedestrian behavior understanding and prediction with deep neural networks. In *European Conference on Computer Vision*, pages 263–279. Springer, 2016.

[19] Mohammadhossein Bahari, Ismail Nejjar, and Alexandre Alahi. Injecting knowledge in data-driven vehicle trajectory predictors. *Transportation research part C: emerging technologies*, 128:103010, 2021.

[20] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[21] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.

[22] Matthew Niedoba, Henggang Cui, Kevin Luo, Darshan Hegde, Fang-Chieh Chou, and Nemanja Djuric. Improving movement prediction of traffic actors using off-road loss and bias mitigation. In *Workshop on'Machine Learning for Autonomous Driving'at Conference on Neural Information Processing Systems*, volume 1, page 2, 2019.

[23] Sven Kreiss. Deep social force, 2021.

[24] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020.