LLM-driven game-theoretic planning

Zihan Yu, Yinghao Max Liu, Ruijia Hannah Guan

***Abstract*—Social navigation in human crowds introduces challenges in motion planning, sensor capabilities, and the design of algorithms for understanding complex interactions between humans. To model the interaction between autonomous agents and humans, many researchers adopt game-theoretic methods due to its superior ability in explicitly capturing the relationship using mathematical formulation. However, existing game-theoretic approaches rely on pre-determined Gaussian processes to represent the potential pedestrian trajectory distribution without considering social norms and human intentions. On the other hand, large language models (LLMs) have been widely used for autonomous agent decision making. Their strong inference, reasoning ability, and pre-trained prior knowledge in real-world common sense is suitable for tackling the interactions with humans. Therefore, in our project, we leverage LLMs to empower one mixed-strategy Nash equilibrium social navigation algorithm in human crowds, We evaluate our method in the SocNav benchmark focusing on collision rate, time to goal and path length. From the preliminary results, we believe our framework works and worth in depth exploration in the future.**

I. Introduction

Social navigation in human crowds presents significant challenges for autonomous systems, particularly in motion planning, sensor interpretation, and algorithmic understanding of complex human interactions. To model interactions between autonomous agents and humans, researchers predominantly adopt game-theoretic methods due to their superior ability to explicitly capture relationships through mathematical formulation. However, existing game-theoretic approaches predominantly rely on pre-determined Gaussian processes [16] to represent potential pedestrian trajectory distributions, critically overlooking nuanced social norms and scenario-specific variations.

Concurrently, large language models (LLMs) have emerged as a powerful tool for autonomous agent decision-making. Their robust inference capabilities, advanced reasoning abilities, and extensive pre-trained knowledge of real-world contexts make them particularly well-suited for navigating complex human interactions.

In this study, we propose an innovative approach that integrates LLM-powered insights into a mixed-strategy Nash equilibrium social navigation algorithm. Our method aims to enhance game-theoretic models by incorporating rich prior knowledge and contextual understanding. We evaluated our method using the SocNavBench [3] framework, focusing on key performance metrics including collision rate, time to goal, and robot path length. The simulation results demonstrate significant improvements in navigation safety, with a 64% reduction in collision incidents compared to the default game-theoretic approach. However, this safety improvement correlates with a slight increase in total travel time, highlighting the trade-off between collision avoidance and navigation efficiency.

Our research contributes to the growing body of work on autonomous social navigation by demonstrating the potential of LLM-augmented game-theoretic models in understanding and navigating complex human interaction spaces. Future work will focus on optimizing the balance between safety and navigation efficiency.

II. Related Work

A. Interaction Modeling in Motion Planning

Robot navigation in human environments has been a critical research topic, with methodological approaches evolving through multiple trajectory prediction techniques. Early works like Social GAN [4] and Social LSTM [1] established foundational methods for predicting human movement in crowded spaces. More recently, research has shifted towards sophisticated game-theoretical strategies, leveraging Nash Equilibrium models and game-theoretic Model Predictive Control (MPC) to predict human paths and optimize robot decisions in dynamic environments [16, 6]. These advanced frameworks typically employ Gaussian processes to initialize agent strategies, using constant velocity models for pedestrian movement.

Despite significant advancements, current approaches continue to face challenges, including high computational costs and simplified assumptions about human behavioral rationality. The ongoing research underscores the need for more nuanced methods that can efficiently capture the complex dynamics of human-robot interactions.

B. LLM in Social Navigation

Large Language Models (LLMs) offer significant advantages for social robot navigation by providing contextual understanding and adaptability in dynamic human environments. Typically, the LLM-based methods rely on prompt engineering to help LLMs or VLMs interpret interactions and provide suitable directives, such as adjusting speed or changing direction. In previous work, methods like Co-NavGPT used LLMs to coordinate multi-robot exploration by assigning frontiers based on scene comprehension or human commands [18]. Similarly, VLM-Social-Nav utilized Vision-Language Models to generate socially compliant navigation behaviors by scoring robot actions based on social norms [15].

To deploy LLM into the social navigation algorithm, there exist two potential challenges to overcome: the capabilities of managing numeric data and using them to reason about the robot's real-world behaviors. Previous work has demonstrated that LLM with well-designed frameworks could provide high-level planning based on merely the task specification and

numeric data [18, 8]. Moreover, the reliable planning could result from image data or merely textual input as well [15, 14, 12, 17]. However, despite their robust reasoning capabilities, LLM-based approaches face limitations like relatively longer inference time, making real-time application challenging. Furthermore, they struggle with precise low-level controls, highlighting the need for optimization to improve efficiency in practical settings.

III. Methodology

A. Problem Formulation

For social robot navigation, humans are no longer perceived only as dynamic obstacles but also as social entities [9]. Their behaviors may change according to different robot's decisions. In this research, we simulate a scenario involving one mobile robot and a variable number of pedestrians. We treat this system as a discrete-time model with a consistent time discretization interval. The state of the robot at time t is represented as R_t. We also incorporate pedestrians' information $P_{t-H:t}$, which includes the humans' positions across the preceding H time steps.

Our goal is to generate discrete states containing positions and yaws for the robot for the upcoming h time steps, based on its current states, final goal G and surrounding pedestrians' information. This procedure is illustrated by Formula 1.

$$R_{t+1:t+h} = BUM(R_t, G, SIE(R_{t-H:t}, P_{t-H:t})), \quad (1)$$

where function $SIE()$ is implemented by a pretrained-LLM designed to extract the social intentions of humans to help the decision making of the robot, while the function $BUM()$ is a Bayesian update model to generate the determined strategy for the robot based on the social intentions of surrounding humans, thereby complete the social navigation task.

An overview of our method is depicted in Fig 1. Specifically, for pedestrians within the robot's safety threshold, we employ the Social Intention Extractor (SIE) to extract their intentions over the next few frames. The input of this LLM-based model includes the pedestrian's current state, the state of surrounding pedestrians or robots, and the pedestrian's historical trajectory from recent frames. The output of SIE is humans' high-level intention (left, right or straight). With the high-level social intention of humans, the Bayesian Update Model (BUM) initializes the mixed strategies of humans and iterates the robot's mixed strategy until convergence, the final norminal trajectory of the converged mixed strategy is the planned trajectory.

B. LLM-based Social Intention Extraction Module

Unlike previous works directly using LLMs or VLMs to generate high-level commands to assist the control of the robot. The most powerful advantage of such foundation models is their prior knowledge about human society such as social norms. They inherently understand some human behaviors. Therefore, to utilize such powerful skills, we prompt the LLM to act as the pedestrians and generate the high-level intention in the short term.

To adapt the LLM for the role playing task of pedestrians, we have crafted a detailed background description prompt to aid the LLM's comprehension of the task's specific requirements. This background description outlines the role the LLM will play (as a walking pedestrian), the walking environment (such as in a lounge), the objective (to generate walking intention in the next 1 second), the types of input information required (such as surrounding information, historical trajectory and current state), as well as the type of output expected. This universal context will be incorporated into every scenario. An example is depicted in Fig. 2. With these comprehensive task background details, the LLM is expected to grasp the situation and the corresponding requirements more effectively.

For the parameterized representations of surrounding information, history trajectory and current states, we transform them into language descriptions. Specifically, we use detected surrounding pedestrians or robot, parameterized by their ids and locations, as the main content of the input. By utilizing these scenario information prompts, the LLM is informed about the environment and prompted to address the defined problem. Through simple mathematical calculation, LLM can get the relative positions of surrounding agents, which is within the ability domain of current mainstream LLMs.

The expected output the SIE is one of the three high-level intentions, we also record the log probabilities of such three options. We utilize such output to help determine the policy of the robot.

C. Bayesian Update Model

In this study, we choose to model mixed strategies for the robot as mixed strategies enhance adaptability and handle uncertainty in dynamic environments. They enable robots to probabilistically balance social compliance and efficiency, and avoid high-risk actions. Specifically, we follow [16] to use a Bayesian Update framework. It enables iterative belief refinement under uncertainty, ensuring adaptive and robust decision-making in dynamic environments. Combined with mixed strategies, it balances probabilistic action selection and evidence integration. This synergy allows robots to navigate efficiently while maintaining social compliance, leveraging uncertainty modeling to optimize interactions in multi-agent scenarios.

The process of the Bayesian Update Model is illustrated in Algorithm 1. We model the initial nominal mixed strategy for each agent using Gaussian processes. We use a constant velocity model as the mean function for each pedestrian and a global planner generates the mean function for the robot as a trajectory from the robot's current location toward the navigation goal. After the initialization, we get the robot's mixed strategy p_r^k, ith pedestrian's mixed strategy p_i^k, and the corresponding nominal strategy $p_r^{'k}$, $p_i^{'k}$.

With the generated high-level intention from SIE, we use function $Resample$ to resample the mixed strategy for the pedestrian. The process of $Resample$ function is demonstrated in Algorithm 2. We use cross-product to judge left, right and straight trajectories from original mixed strategies and then

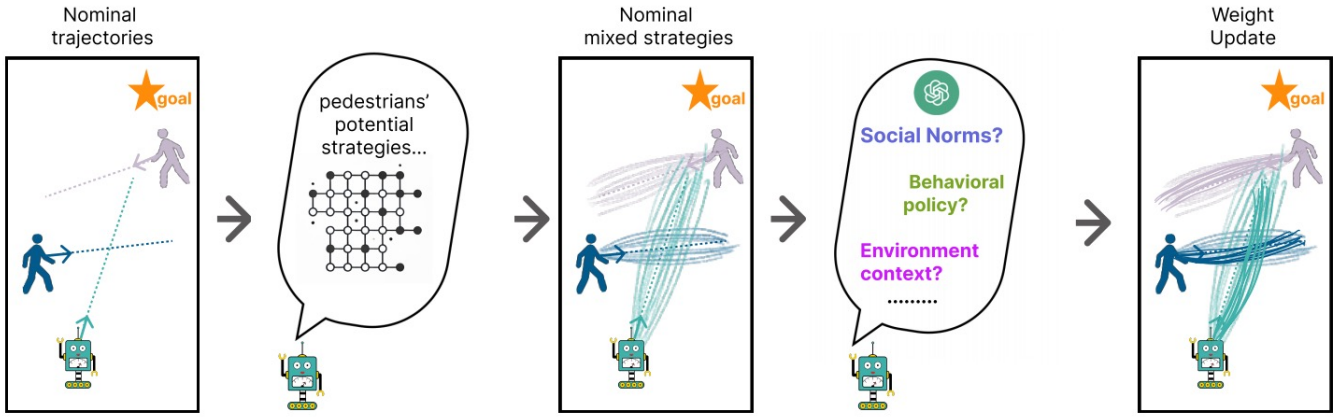

Fig. 1. An overview of our method.

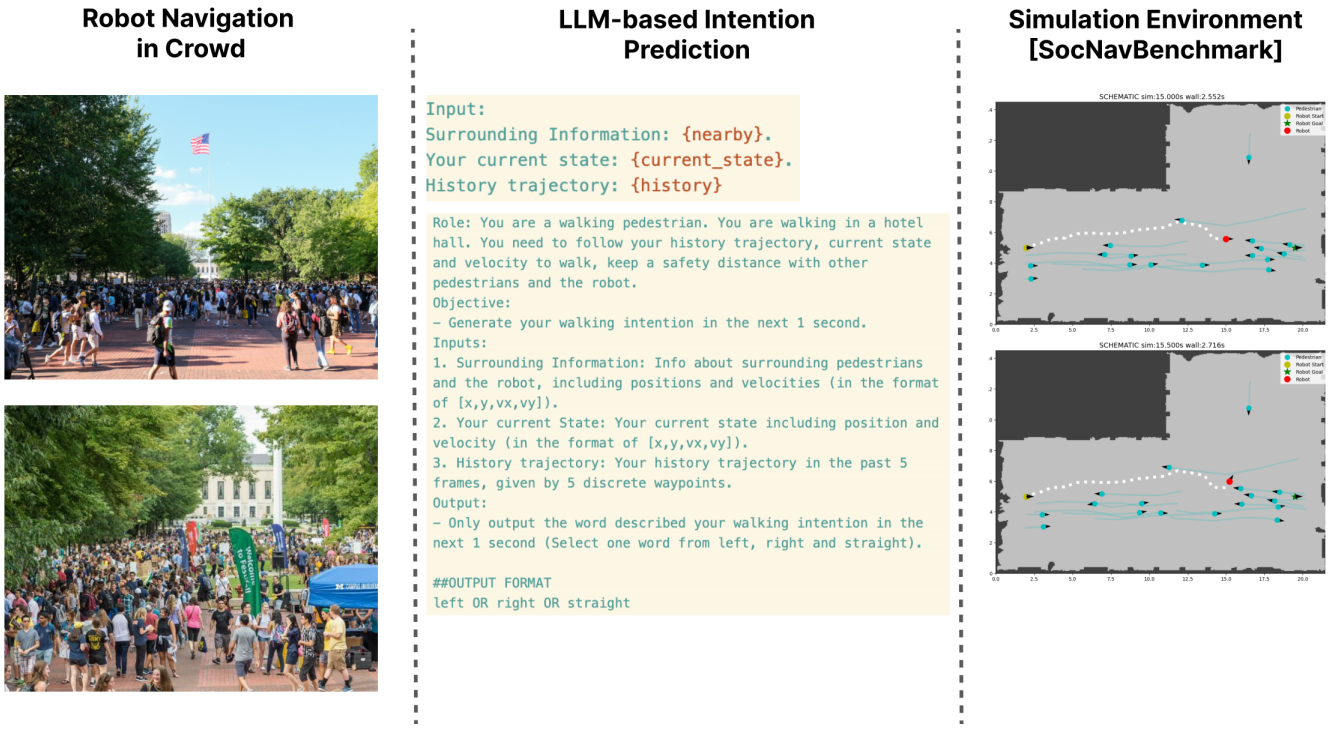

Fig. 2. An example of our background description prompt. Crowd images on the left were sourced from the University of Michigan website [10, 11].

utilize the recorded log probabilities from SIE to resample the trajectories and get the pedestrian's mixed strategy with intention.

After getting the resampled pedestrian's mixed strategies, we use the idea of Bayesian Update to iteratively calculate the interaction score and update the weights for the trajectories in the mixed strategy. The $CalculateScore$ function measures the performance of the trajectory. We choose to use a logistic function to effectively mapping the cost into the desired range and reflecting the desired behavior in the cost function. The detailed procedure is shown as Formula 2

$$C_{r,i,m,l} = \beta \cdot \max_{t} \left(2 - \frac{2}{1 + \exp\left(-\alpha \cdot d_{r,i,m}(t)\right)} \right), \quad (2)$$

$$d_{r,i,m,l}(t) = \left\| p^{k}_{r,m}(t) - p'^{k}_{i,l}(t) \right\|^2, \quad (3)$$

where $p^{k}_{r,m}(t)$ and $p'^{k}_{i,l}(t)$ are position vectors in R^2 representing the robot and pedestrian i at time t. α and β are scaling parameters.

Algorithm 1: Bayesian Update Model

Notation:
- Robot's mixed strategy at kth step: p_r^k
- Robot's nominal strategy at kth step: $p_r^{'k}$
- Pedestrian i's mixed strategy at kth step: p_i^k
- Pedestrian i's nominal strategy at kth step: $p_i^{'k}$
- Total number of pedestrians: N
- Intention of pedestrian i: I_i

Input: $p_r^{'k},\ p_i^{'k}$
Output: $p_r^{'k+1}$

for $i \in [1, N]$ **do**
  I_i = $SIE(p_i^{'k})$;
  $p_i^{'k}$ = $Resample(I_i,\ p_i^{'k})$;
while *not converge* **do**
  for $i \in [1, N]$ **do**
    `RobotScore` = $CalculateScore(p_r^{'k},\ p_i^{'k})$;
    `FinalRobotScore` += `RobotScore`;
    for $j \in [i, N],\ j \neq i$ **do**
      `HumanScore` = $CalculateScore(p_j^{'k},\ p_i^{'k})$;
      `FinalHumanScore` += `HumanScore`;
    w_i^k = $GetNormalizeAverage$(`FinalHumanScore`);
    `HumanWeight.append(`w_i^k`)`;
  w_r^k = $GetNormalizeAverage$(`FinalRobotScore`);
p_r^{k+1} = p_r^k;
$p_r^{'k+1}$ = $ComputeWeightedMean(p_r^{k+1},\ w_r^k)$;

Algorithm 2: Function Resample

Notation:
- Left trajectory probability: P_{left}
- Straight trajectory probability: P_{straight}
- Right trajectory probability: P_{right}
- Total number of trajectories to sample: M
- Reference direction vector: $\mathbf{v}_{\text{ref}}$
- Direction vector of trajectory τ_m: $\mathbf{v}_{\tau_m}$
- c_z: The z-component of $\mathbf{c}$
- Counts of trajectories: $N_{\text{left}},\ N_{\text{straight}},\ N_{\text{right}}$

Input: $p_i^k,\ P_{\text{left}},\ P_{\text{straight}},\ P_{\text{right}},\ \mathbf{v}_{\text{ref}},\ M$
Output: p_i^{k+1}

Set $N_{\text{left}} \leftarrow 0$, $N_{\text{straight}} \leftarrow 0$, $N_{\text{right}} \leftarrow 0$.
for $m = 1$ ***to*** M **do**
  Sample trajectory τ_m from p_i^k based on probabilities $P_{\text{left}},\ P_{\text{straight}},\ P_{\text{right}}$.
  Obtain direction vector $\mathbf{v}_{\tau_m}$ of τ_m.
  Compute cross product $\mathbf{c} = \mathbf{v}_{\text{ref}} \times \mathbf{v}_{\tau_m}$.
  if $c_z > 0$ **then**
    Trajectory τ_m is classified as **left**.
    Update $N_{\text{left}} \leftarrow N_{\text{left}} + 1$.
  else if $c_z = 0$ **then**
    Trajectory τ_m is classified as **straight**.
    Update $N_{\text{straight}} \leftarrow N_{\text{straight}} + 1$.
  else
    Trajectory τ_m is classified as **right**.
    Update $N_{\text{right}} \leftarrow N_{\text{right}} + 1$.
$p_{\text{left}}^{k+1} = \frac{N_{\text{left}}}{M}$, $p_{\text{straight}}^{k+1} = \frac{N_{\text{straight}}}{M}$, $p_{\text{right}}^{k+1} = \frac{N_{\text{right}}}{M}$
$p_i^{k+1} = \{p_{\text{left}}^{k+1},\ p_{\text{straight}}^{k+1},\ p_{\text{right}}^{k+1}\}$.

After getting the interaction scores, we use $GetNomalizeAverage$ function to calculate the normalize average score and update the weight of trajectories in the mixed strategy. The process is illustrated as follows:

$$\bar{C}_{r,i,m} = \frac{1}{N\lambda} \sum_{i=1}^{N} \sum_{l=1}^{\lambda} C_{r,i,m,l} \cdot w_{r,i}, \tag{4}$$

$$w_{r,i} = \frac{\exp\left(-\gamma \cdot \bar{C}_{r,i,m}\right)}{\frac{1}{\lambda} \sum_{i=1}^{\lambda} \exp\left(-\gamma \cdot \bar{C}_{r,i,m}\right)}, \tag{5}$$

$$p_r^{'k+1} = p_r^{'k} + \sum_{i=1}^{N} w_{r,i} \cdot \delta p_{r_i}^{'k}. \tag{6}$$

After calculation, the final weighted mean of the mixed strategy is outputted as the robot planned trajectory.

IV. Evaluation

A. Environment Overview

For our evaluation methods, we adopted a simulator called SocNavBench [2], a prerecorded pedestrian simulation framework. The simulation provided 33 representative navigation scenarios with real-world pedestrian trajectories. Pedestrian trajectory data comes from the UCY [7] and ETH [13] dataset. The pedestrian data includes varied crowd densities, and demonstrates interesting pedestrian behaviors such as grouping, following, passing, pacing, and waiting. The pedestrian paths are replayed at the simulator tick rate with a 1:1 ratio to the recorded time. In addition, the simulator also provides 3D rendering and a depth map for algorithms to use to provide a more realistic simulation.

There are four main maps in SocNavBench: ETH, Univ, Zara, DoubleHotel. In each scenario, there are a number of episodes, each consisting of a diversity of crowd sizes, speeds, and densities, and the directions of motion with respect to the robot's traversal task. All the episodes have about 44 pedestrians on average. The output of each episode run evaluate the robot trajectories based on four general evaluation categories: path quality, motion, quality, pedestrian disruption, and meta-statistics. We collected these metrics as a part of our experiment benchmark.

Path quality quantifies the quality and efficiency of the path generated by social navigation algorithm. Some metrics are collected to measure path quality. Path length is the total distance traversed by the robot in the episode in meters. Path length ratio is the ratio of straight line distance between the start and goal to the robot's path length for any episode. Path irregularity is the radians averaged over the absolute angle between robot heading and the vector pointing to goal. Goal traversal ratio is calculated for incomplete episodes. Path traversal time is total simulator time taken to traverse the robot's path.

Motion quality includes average speed, average energy

expenditure, average acceleration, and average jerk. Average jerk is the time derivative of acceleration of the robot over its entire trajectory. It also includes the closest distance for pedestrians and the time to the collision. Closest distance for pedestrians is calculated by finding the distance to the closest pedestrian at each robot trajectory segment. Time-to-collision is found by considering the minimum time-to-collision to any pedestrian in the environment at each robot segment.

Meta-statistics include the overall success rate, total pedestrian collisions, failure cases, and average planning waiting time. Average waiting time is the average time the simulator waits for the commands from the planner.

In our experiments, we focused on three key metrics: collision number, travel time, and path length. We consider collision number as a meta-statistics metric. Collision number is the total number of collisions associated with the episode. Travel time is our measure of motion quality. Travel time is the total number of seconds in simulation time that the robot takes to complete a single episode. Furthermore, we consider path length as a criteria for path quality. Path quality is the total distance traversed by the robot in the episode in meters. In order to help us garner insights for qualitative results, we take advantage of the result of each episode which also includes a short video of the robot navigating in the map space. It also includes images of each frame of the video. We manually go through each video and its associated frames to observe the patterns to which the robot react when navigating through crowd.

B. Benchmarking Methods

In our evaluation, we implemented four methods in total to assess the efficiency and quality of our proposed methodology.

The first method with which we experimented is the sampling method. It samples trajectories from a connectivity graph and evaluates them via heuristic cost functions and produces the minimum cost trajectory.

The second method we implemented is the BRNE algorithm which leverages Bayesian updates of non-symmetric and multi-modal mixed strategies for multi-agents and generates the trajectory from the trajectory samples from the agents' mixed strategies.

We also implemented a variant of the BRNE approach, effectively generating the final trajectory from trajectory samples using the Social GAN method [5]. A generative model would captures the data distribution and a discriminative model estimates the probability whether the sample comes from the training data rather than the generator.

The fourth method is our proposed solution, which integrates LLM to extract pedestrian intentions to generate robot trajectories from the samples.

V. Results

A. Quantitative Results

We evaluated the performance of the proposed algorithm against three social navigation algorithm variants, focusing on three key performance metrics: total number of collisions, average travel time, and average robot path length. The study utilized the comprehensive SocNavBenchmark dataset, which includes 33 distinct map configurations. To enhance experimental variability, we introduced four reactive agents with predefined start and end positions.

TABLE I
Comparison of algorithms

Algorithm name	Collision Number	Travel time	Path length
Sampling	57	17.79	16.51
BRNE	14	18.675	16.12
BRNE+Prediction	12	22.15	16.65
BRNE+LLM	5	21.95	16.55

The results, detailed in Table I, reveal significant performance differences across the algorithms. All three (BRNE)-based algorithms demonstrated substantially lower collision rates compared to the baseline Sampling algorithm. Within the BRNE group, two algorithms enhanced with advanced human intention prediction exhibited superior performance.

Notably, the LLM-driven BRNE+LLM algorithm achieved the most remarkable collision reduction, generating only 36% of the collisions produced by the default algorithm. While the average robot path length remained consistent across all four algorithms, the human intention prediction techniques (Mean = 22.05) marginally increased average travel time relative to the Sampling and default BRNE approaches (Mean = 18.23).

These findings highlight the potential of advanced intention prediction techniques in social navigation while also underscoring the need for further research to optimize real-time planning algorithms. Future work should focus on refining travel time efficiency without compromising the significant safety improvements demonstrated by the intention prediction methods.

B. Qualitative Analysis

For qualitative results, we manually go through the video frames and assess the robot trajectory when interacting with pedestrian agents. We included four representative scenes here to assess our solution's behavior under various circumstances, as outlined in Fig 3.

The first scenario is on the first row. We used this scenario to assess our algorithm on the robot's crossing behavior across a dense crowd. This takes place when the robot is approaching directly to the side of the pedestrian. The robot understands the pedestrian's intention to move to the right. Once the LLM evaluates the pedestrian's intention and concludes that it is unable to pass the pedestrian from the front, it would produce left as the pedestrian's intention. Our algorithm would then choose the trajectory samples to the left. The generated robot trajectory directs the robot to the left of the pedestrian and passes the pedestrian at its rear while the pedestrian keeps heading to the right direction.

The second scenario is on the the second row. We used this scenario to assess our algorithm on the robot's behavior when moving against a dense crowd. This happens when the robot is approaching directly to the front of the pedestrian. In

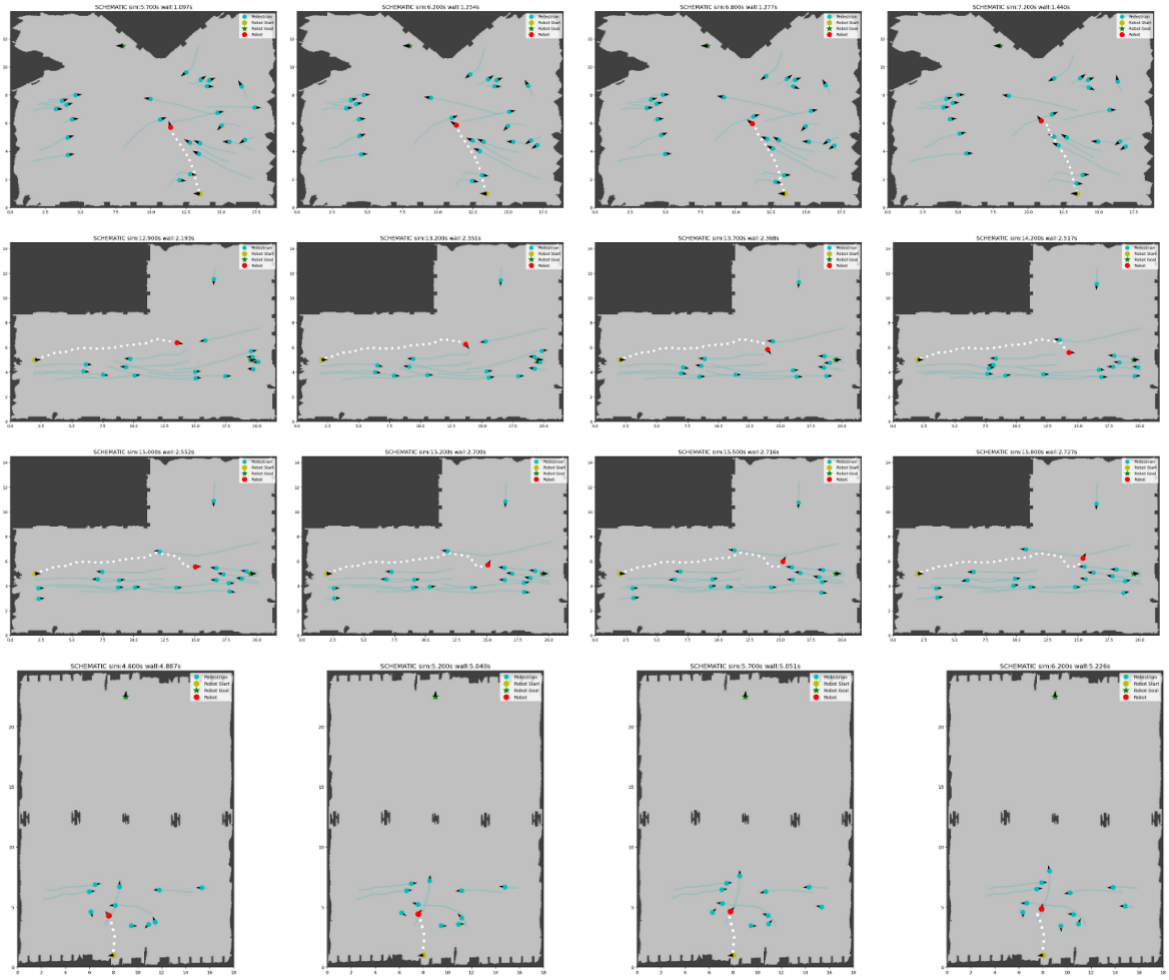

Fig. 3. Qualitative assessment of our method.

this case, the pedestrian moves from top right to the bottom position. Once the LLM understands the pedestrian's intention, it would conclude right, thus prompting the planner to sample from the right-side trajectories of the generated trajectories. The robot would follow the final trajectory and pass through the pedestrian by turning to it's right. This allows the robot to pass the pedestrian from his the front while the pedestrian keeps heading to the left direction.

The third scenario is on the third row. We used this scenario to assess our algorithm on the robot's behavior when moving against a dense crowd. One difference is that the robot is current moving from bottom right to top left. The passing would occur when the robot approaches directly against a dense crowd. The LLM extracts the pedestrians intention and yields the output as left. This would allow the robot to sample from the left-side trajectories of the generated trajectories from mixed strategies. The robot would make a left turn to pass the pedestrian from the front while the pedestrian keeps heading to the right direction.

The fourth scenario is on the last row. We used this scenario to assess our algorithm on the robot's behavior when crossing a dense crowd. This is a variation of the first scenario. In this case, the robot is approaching directly to the side of the pedestrian. The robot understands the pedestrian's intention to move to the left. However, the LLM adapts to the complex environment of the pedestrian trajectories. Initially, the LLM evaluates the pedestrian's intention and concludes that it is possible pass the pedestrian from the front. Hence, the LLM would yield left as a response, driving the robot to take a left turn and heading to the front of the pedestrian. However, in the next few iterations, the LLM planner realizes that the robot is unable to pass the pedestrian through its front. The LLM changes its output to right, directing the robot to take the right turn instead. Therefore, the robot would make a right turn to pass the pedestrian at its rear while the pedestrian keeps heading to the left.

VI. Discussion

Based on the result we collected from the experiments, our proposed method to integrate LLM reasoning ability in the BRNE algorithm shows a significant reduction of the number of collisions. Qualitative results also confirm this as

we discover the strong adaptability of LLM reasoning ability to extract pedestrian intentions when the number of pedestrian agents are dense in close proximity with the robot.

We also extracted travel time and path length from the meta statistics collected from each episode run. Despite a slight increase in travel time due to extended prediction in the game-theoretic approach, we did not observe a significant variance in our proposed method's performance compared to the other benchmarking methods mentioned above. We have several hypothesis for the result. One possible reason could be due to the limited selections of environments for testing. Currently, SocNavBench only includes four maps. For each map, the robot has an unoccluded view to the goal state. If we introduce the map types to more complicated variants where more features would require the robots to learn and explore, then the path qualities of the generated trajectories would change drastically.

Another reason might be related to the setup of reactive agents we manually added to the simulation. In our experiments, we added four reactive agents on top of the pre-recorded agents provided by the simulation. The reactive agents are steered in the direct of the robot, aiming to eventually intersect with the robot's paths. This would increase the potential collisions to help us evaluate the planners' behaviors under stress. One difficulty we face when adding the additional reactive agents is it is hard to control the timing when collisions between our reactive agents and the robot take place. Hence, some reactive agents failed to probe the planner in the way we conceived. Furthermore, we also notice that our number of reactive agents might be too few to induce constant change of directions the robot trajectory, hence this leads to the similar path quality and motion quality.

Regarding the slight increase in travel time observed in BRNE+Prediction and BRNE+LLM, potential contributing factors may include taking longer detours to avoid collisions and the additional computational overhead of the models. Future research should focus on identifying the key drivers behind this phenomenon and proposing targeted solutions.

Our work benefits significantly from the highly accessible pedestrian simulation environment provided by SocNavBench, facilitating our investigation of the existing BRNE approach and implementation of our novel approach. Nonetheless, SocNavBench comes with its own limitations. We were only able to use the sampling baseline in SocNavBench. We were also interested in adding the other baselines such as Social Force as our benchmarks against our proposed solution. However, the other methods were built on an outdated version of gym, resulting in a series of dependency incompatibility issues. Due to our tight time schedule, we eventually decided to adopt the three aforementioned planners.

Furthermore, it is also important to point out Sim-to-Real gap between SocNav environment and implementing the algorithm on a physical robot. It is worth noting that our experiments running on SocNav represent the pedestrian locations as 2D coordinates. However, in order to test our novel solution on a physical robot, it is important to revise our planner to accommodate the increased degrees of freedom. It is also important to note that the simulation runs in a global view with the assumption that the robot knows about the trajectory of the pedestrians a priori. This assumption could be very dangerous in the real world because the perception modules of the robot is always subject to deviations. In this case, it is important for the planner to correct the trajectory candidates from pedestrian mixed strategies when it is deployed in real life. Although we have shown promising results for our proposed method to significantly reduce the number of collisions, we should leverage the less computationally expensive perception abilities of the physical robots to eliminate some configuration spaces before allowing the BRNE algorithm to generate initial trajectories.

Another aspect worth considering is the assumptions of human agents and robots in the simulation. In the experiments we ran, human agents move in a very low speed as they are put under a speed threshold. Similarly, robots are also subject to low speed limit. This assumption could be very dangerous when implementing the proposed solutions onto physical robots. Human pedestrians are very unpredictable in real life and one very possible situation for the robot planner to consider is when pedestrians interchange between high and low speed. One possible future work is that more acceleration variations of the robot and human can be introduced so that the behavior of our proposed planner can be evaluated on this new scenario. In this way, the robot motion planner can be evaluated thoroughly while ensuring path quality and motion quality once deploying to physical robots. More importantly, this would help diminish the likelihood for our proposed solution to induce high-stake collision with pedestrians when they vary their acceleration in their paths.

When we were choosing LLM candidates for reasoning over the pedestrians' trajectories, we adopted ChatGPT 3 API to output pedestrian intention. Using the base model out of the box might not be a computationally efficient and performance oriented approach. In the future, it might be helpful to curate relevant datasets that include pedestrians history states, the robot's configurations, and the robot's reaction to this situation to ensure safe and efficient navigation scheme. It is likely to further increase the robot's path quality and motion quality while decreasing the likelihood for collision. In addition, the adoption of API might not be desirable on physical robots due to the latency of network communication and instability of network connections. One possible solution to the instability is leveraging on-device inference off large language models by fine-tuning smaller size base models with thoughtfully curated datasets that would achieve roughly the same level performance as the larger large language models.

Lastly, we currently use prompts to describe the past trajectories of pedestrian agents and expect the language models to output the pedestrians' intentions. We see a potential to integrate visual language models for the foundational models to capture more nuanced social navigation clues. An example would be asking the robot to stop at the gesture of a policeman in the case of an emergency. Under such circumstances,

the foundational models need visual input to reason over the pedestrians intention beyond their next possible position. Furthermore, new ways to encode the contextual information of the environment around the robot can be explored to reduce the computational cost to tokenize the language model input.

VII. Conclusion

In this paper, we introduced a LLM-driven game-theoretic planning approach and benchmarked it against three different algorithms, spanning non-game-theoretic methods, the default game-theoretic baseline, and a game-theoretic approach equipped with non-LLM trajectory prediction. Evaluation results demonstrates that LLMs can accurately capture social norms, leading to safer navigation behaviors across multiple real-world scenarios. These findings suggest that LLMs hold significant promise for enriching robot motion planning through more sophisticated reasoning about pedestrian intentions. Future research should explore the application of LLMs in more complex environmental settings and rigorously investigate both their strengths and limitations in reasoning.

Acknowledgments

References

[1] 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.

[2] Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation. *CoRR*, abs/2103.00047, 2021. URL https://arxiv.org/abs/2103.00047.

[3] Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(3):1–24, 2022.

[4] 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.

[5] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks, 2018. URL https://arxiv.org/abs/1803.10892.

[6] Viet-Anh Le, Vaishnav Tadiparthi, Behdad Chalaki, Hossein Nourkhiz Mahjoub, Jovin D'sa, Ehsan Moradi-Pari, and Andreas A Malikopoulos. Multi-robot cooperative navigation in crowds: A game-theoretic learning-based model predictive control approach. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4834–4840. IEEE, 2024.

[7] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3): 655–664, 2007. doi: https://doi.org/10.1111/j.1467-8659.2007.01089.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01089.x.

[8] Jiageng Mao, Yuxi Qian, Junjie Ye, Hang Zhao, and Yue Wang. Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*, 2023.

[9] Reuth Mirsky, Xuesu Xiao, Justin Hart, and Peter Stone. Conflict avoidance in social navigation—a survey. *ACM Transactions on Human-Robot Interaction*, 13(1):1–36, 2024.

[10] University of Michigan. Campus involvement. https://campusinvolvement.umich.edu/, . Accessed: 2024-12-11.

[11] University of Michigan. Student organization resource center. https://campusinvolvement.umich.edu/sorc/node/1, . Accessed: 2024-12-11.

[12] Amirreza Payandeh, Daeun Song, Mohammad Nazeri, Jing Liang, Praneel Mukherjee, Amir Hossain Raj, Yangzhe Kong, Dinesh Manocha, and Xuesu Xiao. Social-llava: Enhancing robot navigation through human-language reasoning in social spaces.

[13] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. pages 261–268, 09 2009. doi: 10.1109/ICCV.2009.5459260.

[14] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[15] Daeun Song, Jing Liang, Amirreza Payandeh, Xuesu Xiao, and Dinesh Manocha. Socially aware robot navigation through scoring using vision-language models. *arXiv preprint arXiv:2404.00210*, 2024.

[16] Muchen Sun, Francesca Baldini, Peter Trautman, and Todd Murphey. Mixed-strategy nash equilibrium for crowd navigation. *arXiv preprint arXiv:2403.01537*, 2024.

[17] Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. *arXiv preprint arXiv:2309.16292*, 2023.

[18] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. Co-navgpt: Multi-robot cooperative visual semantic navigation using large language models. *arXiv preprint arXiv:2310.07937*, 2023.