Socially-Aware Navigation Using Non-Linear Multi-Objective Optimization

Scott Forer1, Santosh Balajee Banisetty2, Logan Yliniemi3, Monica Nicolescu4, and David Feil-Seifer5

Abstract—For socially assistive robots (SAR) to be accepted into complex and stochastic human environments, it is important to account for subtle social norms. In this paper, we propose a novel approach to socially-aware navigation (SAN) which garnered an immense interest in the Human-Robot Interaction (HRI) community. We use a multi-objective optimization tool called the Pareto Conavity Elimination Transformation (PaCET) to capture the non-linear human navigation behavior, a novel contribution to the community. We use autonomously sensed distance-based features that captures the social norms and associated social costs for a given trajectory point towards the goal. Rather than use a finely-tuned linear combination of these costs, we use PaCET to select an optimized future trajectory point, associated with a non-linear combination of the costs. Existing research in this domain concentrates on geometric reasoning, model-based, and learning approaches, which have their own pros and cons. This approach is distinct from prior work in this area. We showed in a simulation that the PaCET based trajectory planner not only is able to avoid collisions and reach the intended destination in static and dynamic environments but also considers a human’s personal space in the trajectory selection process.

I. INTRODUCTION

Recent technological advancements in sensing and computation have stimulated a greater interest in the application of autonomous agents to real-world interaction. In particular, researchers and commercial interests have experimented using such mobile robots to provide assistive services such as guiding and carrying luggage in complex, pedestrian dense environments (shopping malls, airports, and other public places) [1]. One such example is the work conducted by Shi et. al [2], where an autonomous robot was used to distribute flyers at a shopping mall. Robot domains, especially socially assistive robotics (SAR), benefit from navigation; such movement extends the reachable service area of the robot [3]. However, navigation, if not performed properly, can cause negative social reactions [4].

Socially Aware Navigation (SAN) utilizes space, distance, and movement as a spatial communication medium. For human-human interaction, humans understand spatial communication and navigate in such a way that social norms are obeyed. HRI to match this human-human interaction property, the spatial communication between a human and a robot should not be neglected; it should be utilized to achieve human-friendly navigation [5]. An effective robot’s actions, including actions involving spatial communication, must be suitable for a given social circumstance. Hence, utilizing spatial communication between a robot and a human is very important for assistive robots.

Proxemics [6], social rules for interpersonal distance, is an important aspect of navigation; researchers interested in SAN are investigating methods to integrate the rules of proxemics into robot navigation behavior. Kruse et. al [7] authored an extensive review of methods like hard-coded rules, geometric reasoning, Model-based Inverse Reinforcement Learning (IRL) that tried incorporating the rules of proxemics. A current review of existing approaches show the following limitations:

1) Some of the approaches depend on exocentric sensing hence, limiting the robot’s services to a particular environment.

2) The environment/scenario is a singleton, i.e., Only a hallway, a room, etc is considered or only an approach behavior, or a passing behavior is considered.

3) Planners are optimized for single or few objectives with a linear combination or weighted sum.

Our prior work [5] presented an approach utilizing a spatial model over distance-based features to generate trajectories that are socially appropriate, which was validated with human partners. Limitations 1 and 2 were addressed in our recent work [8], [9], where a Gaussian Mixture Model (GMM) based approach was used to select an appropriate trajectory for an autonomously sensed social scenario. An egocentric laser-based sensing method was used to calculate distance based features that were used in sensing the social scenario. In this work, we propose a novel multi-objective optimization approach for a socially-aware navigation planner.

Linear methods for socially-aware navigation may be inadequate for a number of reasons:

- Navigation rarely involves optimizing for a single objective. For example, humans optimize for path length, execution time and most importantly, social norms while walking from one place to the other.
- The human environment is too complicated for linear approaches to effectively approximate. For example,
when a human optimizes on multiple objectives some of the objectives have a relationship that causes non-linear trade-offs.

For these reasons, a multi-objective optimization approach capable of handling multiple objectives independently is warranted.

Further details on one such approach is presented in section IV. The remainder of this work is organized as follows: Section II presents previous work that we build upon. Section III provides the necessary background. Section IV describes our experimental approach. Section V provides our experimental validations. Finally, Section VI concludes this work and provides directions of future research.

II. RELATED WORK

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

Recently, research and applications related to SAN are of great interest in the HRI field as it offers a way to incorporate right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

Recently, research and applications related to SAN are of great interest in the HRI field as it offers a way to incorporate right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.

When we talk about SAN, there are many scenarios that researchers address, such as maintaining pace with a person, following a person, approaching a person, and other social hallway behaviors. There are many ways of achieving one or more behaviors. For example, for a socially appropriate hallway behavior, one can increase the cost for traveling on the left side of the hallway so that the robot always navigates on the right side. Similar behavior can be achieved by utilizing supervised/unsupervised learning and reinforcement/inverse reinforcement learning techniques. We will discuss some of these methods in this section.
Multi-objective optimization has already started to play a role in real world applications [19]. Some examples of real world multi-objective scenarios are high speed civilian aircraft transportation [20], urban planning [21], and for designing trusses [22]. The trajectory planner used in this paper builds off of a pre-existing one that utilizes a multi-objective approach along with linear combination [23].

B. PaCcET

In some cases optimizing a single objective does not yield the performance that is desired and therefore multiple objectives need to be considered when evaluating a policy’s fitness. A common method is to simply multiply a preset scalar value to each objective’s fitness score and then add them all together. In some domains this can lead to an optimal set of policies however, in some complicated domains this method will yield sub optimal policies. A solution to this is to use a multi-objective tool, such as PaCcET, to properly evaluate policies on multiple objectives [24], [25]. PaCcET works by first obtaining an understanding of the solution space and finding the Pareto optimal solutions. Next PaCcET transforms the solution space and then compares each solution giving single fitness score representative of how well each solution performed in the transformed space.

At a high level, PaCcET works by transforming the Pareto Front in the objective space in a way that it is forced to be convex. This allows the linear combination of transformed objectives to find a new Pareto Optimal point. PaCcET iteratively updates this transformation to always force non-explored areas of the Pareto Front to be more highly valued than points dominated by the Pareto Front or points that are on the explored areas of the Pareto Front.

PaCcET has seen a variety of applications: it has been used to extend the life of a fuel cell in a hybrid turbine-fuel cell power generation system [26], the operation of the electrical grid on naval vessels [27], the coordination of multi-robot systems [28], and for the efficient operation of a distributed electrical microgrid [29], where a series of small power generation systems coordinate to meet the demands of consumers. In each of these applications, it has been shown that it functions at or above the solution quality of other techniques like NSGA-II or SPSA2 [24], with as low as one tenth of the run time.

For the purpose of this project PaCcET was used over other multi-objective tools because of its computational speed [24]. PaCcET was used to evaluate the possible trajectories developed in the local planner. At each time step the sensor data is analyzed and the desired features are evaluated for each of the potential future trajectories. PaCcET then uses the fitness values for each feature of each future trajectory to develop the solution space and obtain the optimal future trajectory. Since at each time step, a future trajectory is developed independently, PaCcET develops a brand new solution space at each time step. By using PaCcET like this the local planner can be optimized in real time.

IV. Method

In this section we detail our methodology of the navigation planner, the features or objectives that we used to optimize the trajectories, and how PaCcET was implemented in the local trajectory selection process. The overall function of the local trajectory planner at each time step is to generate an array of possible future trajectories points and evaluate each future trajectory point based on the predefined feature set as shown in Figure 2. In previous work the features were assumed to have either no relationship or a simple linear relationship with one another however, this is not always the case and therefore we need to consider the possibility that the features are not only dependent on each other but also have nonlinear relationships.

A. Features/Objectives

In previous work the features that are extracted are each assigned their own cost. For example the path distance cost is the length that the robot has already traveled and the goal distance cost is the distance the robot is from the goal. In this case the path distance will have a linear relationship with the goal distance since the change in one has a direct linear impact on the other. Once each feature has a cost associated with it, each cost is multiplied by a pre-tuned scalar and then added together thus giving the linear combination, or weighted sum in this case, cost function shown in Equation 1. We can think of this cost function as an objective, where each possible future trajectory point has a cost or fitness associated to that objective. Since the purpose is to minimize the overall cost function, the planner will take the best path possible that minimizes the function, which in this case will minimize both features.

\[
\text{cost}(v_x, v_y, v_\theta) = \alpha(\Delta_{\text{path}}) + \beta(\Delta_{\text{goal}})
\]  

More recently this cost function has been adapted to include a heading difference feature and an occupancy (occ) cost feature, where the heading difference is the distance that the robot is from the global path and the occ cost is the occupancy cost used to keep the robot from hitting
something. The same approach as in the previous cost function is taken in Equation 2. By taking a closer look at just the heading difference and how that might affect the path distance or goal distance it becomes less clear if there is only the linear relationship between the four. For example, if there is an obstacle in the robot’s path, it will try and minimize goal distance by changing its heading, thus increasing the cost heading feature cost. This in turn also increases the path distance cost, though this may or may not be linear.

$$\text{cost}(v_x, v_y, v_\theta) = \alpha(\Delta_{\text{path}}) + \beta(\Delta_{\text{goal}}) + \gamma(\Delta_{\text{heading}}) + \delta(\Delta_{\text{occ}})$$ \hfill (2)

Building upon the prior work done in this area, we include a socially-aware navigation feature such as interpersonal distance (ID). As a way to dissuade the robot from getting too close to a human a cost function was developed to penalize the robot at an exponential rate as the interpersonal distance decreases as seen in Equation 3. Although we could penalize the robot based on this at all times, it really isn’t necessary if the interpersonal distance is so large that it wouldn’t be considered a socially inappropriate distance. Therefore the robot is only penalized if the interpersonal distance is less than or equal to 1.5 meters.

$$ID_f = e^{1/ID}$$ \hfill (3)

Instead of adding this feature’s cost into the previous cost function, we make the assumption that its relationship with other features might be nonlinear and therefore gets treated as its own objective. Since we know that the above cost function works sufficiently enough from previous work, we can treat that as its own objective as well. Now instead of optimizing on just one objective we need to optimize on multiple objectives, hence our multi-objective approach. As one can imagine using a multi-objective tool like PaCcET requires computational time and since this is intended to work in real time any chance to improve the computation time should be utilized. In this case treating the first four features used in the previous cost equation as a single objective not only speeds up this process, but in turn allows for the possibility to add even more features to our local trajectory planner. Using PaCcET to do the multi-objective transformations we essentially get a new cost function with a PaCcET fitness denoted by $P_f$, which was modeled under the assumption of nonlinear relationships between the objectives. Equation 4 shows how $P_f$ is a transformation function dependent on multiple variables.

$$P_f = T_f(Obj_1, Obj_2, ..., Obj_n)$$ \hfill (4)

In this work we are only interested in two objectives. The first objective is the original cost function, which is the linear combination of the path distance, goal distance, heading difference, and occ cost. The second objective is the interpersonal distance, a social feature. Equation 5 shows the PaCcET fitness function with our proposed objectives.

$$P_f = T_f(\text{cost}(v_x, v_y, v_\theta), ID_f)$$ \hfill (5)

### B. Trajectory Planning

The robot’s trajectory can be broken into three parts, the global planner, the local planner, and a low-level collision detection and avoidance. The global trajectory planner works by using knowledge of the map to produce an optimal route given the robots staring position and the goal position. The global path is created as a high level panning task however, the global path can be recreated if the robot has to deviate to far from the current global path. The role of the traditional local planner is to stay in line with the global path unless an obstacle makes it so the robot has to deviate from the global path. The low-level collision detector simply works by stopping the robot if it gets too close to an object. In this work we use the traditional global trajectory planner and low-level collision detector [23] and make adaptations the local trajectory planner to incorporate interpersonal distance and PaCcET.

Algorithm 1 shows the main functions of the local trajectory planner how the future trajectory points were stored to be used with PaCcET. The trajectory planner is called every time step, which in this case is every 0.1 seconds. Once the trajectory planner is called the Transform_Human_State function is called to compute the human state from the human reference frame to the robots state from the robots odometry reference frame, which allows the interpersonal distance corresponding to each possible trajectory to be calculated in the Generate_Trajectory function. Now there are two methods of calculating the possible trajectories. The first is assuming that the robot can only move forwards, backwards, and turn. To produce the possible trajectories for this physical set up we loop through every combination of a sample of linear velocities ($v_x$) and angular velocities ($v_\theta$) to generate trajectories. Once a trajectory is created, we determine if it is valid based on the constraints for the first objective. For example, trajectories that would make the robot hit a wall, obstacle or human are not considered valid trajectories and therefore will not be stored in the Store_Trajectory function. By not storing these invalid trajectories the speed at which PaCcET runs can be improved.

The second method is assuming that the robot is capable of holonomic movements i.e., the robot can move forwards, backwards, left, right, and turn. Given these movements, we again loop through all the possible movements given the predefined number of $v_x$ samples, $v_y$ samples, and $v_\theta$ samples. Again, if the trajectories are valid they are stored. Once all the valid trajectories are stored for all possible movements, the Run_PaCcET function runs giving back the best possible trajectory, $(T_b)$, based on its multi-objective transformation process.

In order to run a multi-objective tool like PaCcET each objective’s fitness needs to be calculated. Algorithm 2
Algorithm 1: Local Trajectory Planner Algorithm. The trajectory planner generates multiple trajectories \((T)\) given a number of \(V_x\) samples and \(V_y\) samples and calculates the independent cost for each feature. The cost for each feature is based on the robot's state \((H_s)\) and the robot's state \((R_s)\). At the end of a time step the best trajectory \((T_B)\) is returned.

Input: \(V_x\) samples, \(V_y\) samples, \(H_s\), \(R_s\)
Output: Best_Trajectory

1. for Each time step do
   2. Transform_Human_State \((H_s, R_s)\)
   3. for Each \(V_x\) do
      4. \(T \rightarrow \text{Generate_Trajectory}(T, H_s)\)
      5. if valid trajectory then
         6. Store_Trajectory \((T)\)
      7. for Each \(V_y\) do
         8. \(T \rightarrow \text{Generate_Trajectory}(T, H_s)\)
         9. if Valid Trajectory then
            10. Store_Trajectory \((T)\)
   11. if Holonomic Robot then
       12. \(T \rightarrow \text{Generate_Trajectory}(T, H_s)\)
       13. if Valid Trajectory then
          14. Store_Trajectory \((T)\)
   15. Run_PaCcET \((T)\)
   16. Return \(\iff(T_B)\)

Algorithm 2: Generate Trajectory Algorithm. The generate trajectory function takes in a trajectory \((T)\) and the human's state \((H_s)\) to compute the cost function for each feature. The trajectory \((T)\) is then returned to the local trajectory planner.

Input: \(T, H_s\)
Output: \(T\)

1. \(S \rightarrow \text{Calculate_State}(T)\)
2. \(\text{path_dist} \rightarrow \text{Compute_Path_Dist}(S)\)
3. \(\text{goal_dist} \rightarrow \text{Compute_Goal_Dist}(S)\)
4. \(\text{occ_cost} \rightarrow \text{Compute_Occ_Cost}(S)\)
5. \(\text{heading_diff} \rightarrow \text{Compute_Heading_Diff}(S)\)
6. \(\text{cost} \rightarrow \text{Compute_Cost}()\)
7. \(\text{TD} \rightarrow \text{Calculate_Interpersonal_Distance}(H_s, S)\)
8. Return \(\iff\text{Trajectory}(T)\)

running PaCcET’s main functions an instance of PaCcET must be created. Next the solution space and Pareto front are created by giving each trajectory to the Pareto_Choice function. Now that the Pareto front and its geometry has been calculated, PaCcET can transform the solution space and give a single fitness value for each trajectory in the Compute_PaCcET_Fitness function. Once each trajectory has its PaCcET fitness they are sorted from best to worst in the Sort_Trajectories function, which allows to not only to easily ascertain the best trajectory but is also useful for debugging purposes. Algorithm 3 concludes by returning the best trajectory to the local trajectory planner algorithm.

Algorithm 3: PaCcET Algorithm. PaCcET \((P)\), takes in the vector of valid possible trajectories \(T\) to compute the multi-objective space and the PaCcET fitness \((P_f)\) for each trajectory.

Input: \(T\)
Output: \(T_B\)

1. for Each trajectory do
   2. Store_Objectives \((T)\)
   3. \(P \rightarrow \text{Initialize_PaCcET}()\)
   4. for Each trajectory do
      5. \(\text{Pareto_Choice}(T)\)
   6. for Each trajectory do
      7. \(P_f \rightarrow \text{Compute_PaCcET_Fitness}(T)\)
   8. Sort_Trajectories \((T)\)
   9. Return \(\iff(T_B)\)

V. VALIDATION

When using a multi-objective tool like PaCcET, two key validations play a role in our experimental design: first, does using PaCcET yield local trajectories that get to the goal in an efficient manner; second, is there a clear distinction that the
other objective is being properly incorporated when selecting a local trajectory? To test if this approach was working, four experiments were conducted in simulation to test the quality and robustness of our method in static and dynamic environments. In the four distinct scenarios we show that not only is PaCcET assisting in selecting efficient trajectories based on the original cost function, but it can also include interpersonal distance into its evaluation process without any need for tuning parameters. In each experiment the path of the robot using the traditional local planner and the PaCcET based local planner are shown along with the simulated human’s position or trajectory.

The simulated environment for each experiment was second floor hallway of the Scrupham Engineering and Mines building at the University of Nevada, Reno. The map of the building used in simulation was built using gmapping on the actual PR2. The simulated PR2 is identical to the real-world PR2 in terms of sensing capabilities and is using AMCL for localization on the map. For example, the simulated PR2 uses a 30 meter range laser scanner which is identical to the real PR2 robot’s laser scanner.

**A. Experiment 1**

In the first experiment, the robot was tasked with getting to a goal while passing closely to a static simulated human. Figure 3 shows that when using the traditional planner the
robot made sure to avoid a collision with the simulated human however did not consider any social distance. This will be the case for the other experiments as well since the traditional planner does not consider interpersonal distance into its cost function. The PaCcET-based planner did consider interpersonal distance and therefore the robot deviated from a more straight lined path as a way to satisfy the second objective. Once the threshold for the interpersonal distance was no longer an issue the robot only needed to minimize the first objective therefore returning to a straight-line path. It’s worth noting that in all the experiments conducted the robot also considered a wall as an obstacle and was required to disregard trajectories that would lead to a collision with the wall, which is why the robot did not deviate from global trajectory even more.

B. Experiment 2

The second experiment was developed to mimic a passing scenario where the robot has a set goal but needs to pass by a simulated human who is traveling much slower in the same direction. Figure 4 shows that with the traditional trajectory planner it merely made sure that a collision would not take place as it tried to minimize its cost function. The PaCcET-based planner clearly deviated from its global trajectory in order to consider the interpersonal distance objective, then return to the global trajectory once the once threshold for the interpersonal distance was no longer an issue.

C. Experiment 3

Similar to the previous experiment, the third experiment involves both the simulated human and robot moving however, in this case the simulated human is now moving at a normal walking speed in the opposite direction of the robot. The robot and simulated human pass close to one another but not close enough to cause a collision. Figure 5 shows that the traditional trajectory planner altered its path ever so slightly to ensure that a collision would not happen, where the PaCcET based trajectory planner not only ensured that a collision would not take place but also considered interpersonal distance and provided the simulated human with additional space while passing.

D. Experiment 4

The previous experiments show that when using a PaCcET-based trajectory planner interpersonal distance can be considered when selecting a local trajectory in static and dynamic conditions when a collision is not imminent; however, the case of a collision that will occur unless either the simulated human or the robot moves out of the way also needs to be considered. This experiment considers a simulated human not paying attention or unwilling to change their course and walking directly towards the robot. Figure 6 shows that the traditional trajectory planner was successful at avoiding the collision as expected however, did so along with minimizing its cost function as much as possible which caused the robot to get very close to the simulated human. When using the PaCcET based trajectory planner the robot not only avoided the collision but also gave the simulated human additional space as to satisfy the interpersonal distance objective. It is worth noting that once the interpersonal distance threshold was no longer an issue the robot for a short time used its holonomic movement as a way to quickly minimize the heading difference portion of the original cost function objective.

VI. DISCUSSION AND FUTURE WORK

From the results in the previous section, it is clear that our proposed planner behaved more consistently with an agent that considers the social factor specified, when compared to a traditional planner. The PaCcET navigation planning attributes, such as path length and time to reach the goal are slightly greater than that of the traditional planner. This means that the proposed approach generated trajectories that were sub-optimal in the sense of efficient path planning however, more optimal from a social aspect. The PaCcET navigation planner generated trajectories that are not only safe (for both the agents and environment) but also considered the personal space for the simulated human partner.

By utilizing egocentric sensing, our proposed planner achieved socially acceptable trajectories that are optimized for various objectives, which also includes interpersonal distance. All this was tested in simulation using a 2D simulator called Stage on a machine with an Intel 6th-generation i7 processor @3.4 GHz, 32 GB of RAM. The PR2 robot in the simulation is identical to the real PR2 in terms of navigation capabilities and sensing. Implementing it on the PR2 in a real-world setting such as hallways, open spaces, etc, and measuring both qualitative and quantitative metrics is our next step. In future work, we plan on collecting data with regards to the performance of the planner, i.e. time, path length, distance maintained from a human, etc, and compare this to the traditional planner [23] or an openly available IRL SAN planner [17]. We will also compare the social aspects of the PaCcET planner to the traditional planner or an openly available IRL SAN planner.

VII. CONCLUSION

We presented a novel approach to socially-aware navigation and showed in simulation that the trajectories generated by this approach are better in comparison with a traditional planner. This new planner optimizes not only for shortest distance and other traditional planning performance metrics, but also includes a social factor, interpersonal distance. As more socially assistive robots are deployed in human environments, navigation planners should account for the uncertain human environments that demand social norms to be followed. Unlike other approaches, our proposed approach doesn’t need any training data or an expert that can teach the robot how to navigate in a socially appropriate way. It is also important to note that our approach utilizes a non-linear optimization tool, PaCcET, over multiple cardinal objectives to come up with a trajectory that is best suited for the considered objectives.
ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support of this work by the National Science Foundation (NSF, #IIS-1719027), Nevada NASA EPSCoR (#NNX15AI02H), and the Office of Naval Research (ONR, #N00014-16-1-2312, #N00014-14-1-0776). We would like to acknowledge the help of Gaetano Evangelista, Andrew Palmer, and Roya Salek Shahrezaie.

REFERENCES