Proactive Kinodynamic Planning using the Extended Social Force Model and Human Motion Prediction in Urban Environments

Gonzalo Ferrer and Alberto Sanfeliu

***Abstract*— This paper presents a novel approach for robot navigation in crowded urban environments where people and objects are moving simultaneously while a robot is navigating. Avoiding moving obstacles at their corresponding precise moment motivates the use of a robotic planner satisfying both dynamic and nonholonomic constraints, also referred as kynodynamic constraints. We present a proactive navigation approach with respect its environment, in the sense that the robot calculates the reaction produced by its actions and provides the minimum impact on nearby pedestrians. As a consequence, the proposed planner integrates seamlessly planning and prediction and calculates a complete motion prediction of the scene for each robot propagation. Making use of the Extended Social Force Model (ESFM) allows an enormous simplification for both the prediction model and the planning system under differential constraints. Simulations and real experiments have been carried out to demonstrate the success of the proactive kinodynamic planner.**

I. Introduction

The impact produced by the deployment of service robots is of vital importance for the acceptance of robots in crowded urban environments, specifically among humans in its natural habitat.

In the present work, we propose a planner that predicts human motion and minimizes its impact on all those nearby pedestrians. Time restrictions are significant in social environments: people walk and change their positions during time. A cost-based navigation path is calculated while satisfying both dynamic and nonholonomic constraints, also referred as kinodynamic constraints.

Prediction methods are of great importance. Most approaches separate planning and prediction and although a joint approach may seem to boost the problem complexity, we will present a simple method to jointly account predictions and planning by considering a union state of people and robots.

Human motion prediction can be achieved through learning techniques, like in the works of [1], [2] and [3], where they make use of maximum entropy learning methods using a linear combination of different kinds of features.

In this work, we apply geometrical based predictors such as the works of [4] and [5] that infer human motion intentions and afterwards predict human motion in a continuous space, according to the Social Force Model (SFM) [6], and the Extended SFM [7]. Works such as [8] and [9] already proposed to model people as a summation of a Potential Field (PF), so it is not a novel idea.

The authors are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC. Llorens Artigas 4-6, 08028 Barcelona, Spain. {gferrer,sanfeliu}@iri.upc.edu. Work supported by the Spanish Ministry of Science and Innovation under project RobTaskCoop (DPI2010-17112).

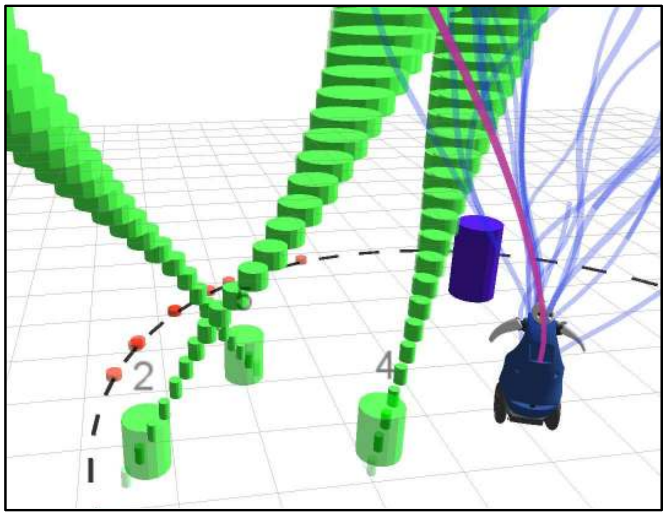

Fig. 1. Simulation environment in a $space \times time$, where people are plotted as green cylinders and their predictions are drawn in the z axis, which corresponds to time. The tree of paths calculated by the robot appears in blue and the best path is a red line.

In addition, PF navigation algorithms [10] have been extensively studied in the literature. Despite their advantages, there exist many well known limitations, such as local minima or oscillations. Several approaches try to overcome these limitations, such as [11], by using a randomized walking path when a local minimum is reached.

The dynamic window approach [12] and other velocity constrained approaches [13] permitted to consider obstacles and collisions. Unfortunately, they suffer from local minima as well. Approaches combining a DWA with a global planner like [14] solve the problem by introducing a global function. Our approach also relies on a global planner.

In [15] they obtain a kinodynamic compliant trajectory by decoupling the problem into a search in space and a posterior optimization of the path satisfying the restrictions. Our approach integrates the search of a path avoiding obstacles as well as provides the inputs required to execute that trajectory considering kinodynamic constraints.

During the last decade sampling based techniques have become quite popular. Furthermore, sampling based methods may take into account kinematic and dynamic constraints such as [16] and [17].

[18] proposes a social aware *reactive* planning in human environments. A joint calculation of people's path and a robot path is done in [19] using Gaussian processes, and [9] uses a PF and minimizes its cost through people's Potential Fields.

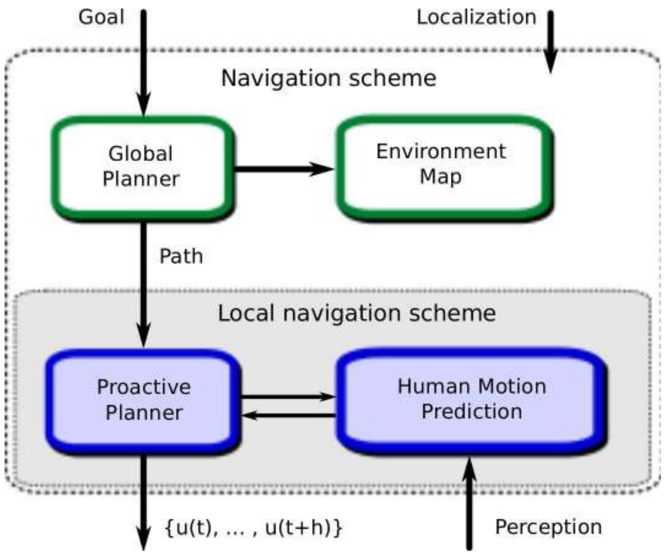

Fig. 2. Overview of the proposed planning scheme.

II. Planning overview

On the one hand, people typically move on the scene changing their positions during some time, and therefore, we must consider time-variant scenarios. On the other hand, a robot trajectory must satisfy those strong time restrictions while considering kinodynamic constraints. A state-space formulation is explained in Sec. III.

The Extended Social Force Model is explained in Sec. IV and how we calculate the forces affecting people and robots. The use of the ESFM will simplify enormously the planning under differential constraints since there is no need to solve the boundary value problem (BVP) to link poses.

In Sec. V we propose to integrate the prediction algorithm with the planning algorithm and solve the problem in a holistic way. The prediction is carried out seamlessly and simultaneously while the planning is being calculated, as explained in Sec. V-E. Every new robot propagation entails an estimation of the motion prediction of nearby people. As a result, our planner presents *proactive* characteristics since we tend to initiate a change rather than a reaction to events.

An important assumption is done: a global planner provides a valid path to the goal unobstructed by static obstacles. A general planning scheme can be seen in Fig. 2.

Our algorithm calculates for each iteration a path to a goal avoiding moving obstacles like pedestrians on the scene. The first output action in $\{u(t_{ini}), \dots, u(t_{horizon})\}$ is executed, and in the next iteration, a new plan is calculated, and a new action is executed. This approach permits a fast adaptation to changing environments, especially if the prediction estimation changes drastically. As we will see in Sec. VI and VII, our algorithm is implemented in real time to provide an adaptable local planning.

The computed plan takes into account the reaction produced by its actions and produces the minimum disturbances to other nearby pedestrians.

III. State-space formulation

For planning purposes in the present work, we consider that both robots and people move in a two-dimensional space which represents the urban environment. Let $\mathcal{X}$ denote the workspace and $\boldsymbol{x} \in \mathcal{X}$ describes the position $\boldsymbol{x} = [x, y]^\top$ in a two dimensional space, as for moving objects (including people) and as robots. The configuration space $\mathcal{C}_z$ is defined as a configuration $q_z \in \mathcal{C}_z$, where z denotes different configuration spaces for people and robots.

Since we take into account a kinodynamic treatment of the planning scheme, we define the phase space $\mathcal{S}_z$ that only considers the first order derivative, where $s'_z \in \mathcal{S}_z$ is defined by $s'_z = [q_z, \dot{q}_z]^\top$. In addition, we deal with strong time constraints, where object movements alter the outcome of the planning calculations, and we should consider the augmented phase space $\mathcal{ST}_z = \mathcal{S}_z \times time$, where $time \in \mathbb{R}^+$, and $s_z \in \mathcal{ST}_z$ is a state $s_z = [q_z, \dot{q}_z, t]^\top$ at time t. In general, we will address the planning problem using the state $s_z \in \mathcal{ST}_z$.

Finally, an action $u_z \in \mathcal{U}_z$ modifies the states s_z, as it will be discussed below.

A. People in the state-space

People are treated as free moving particles, and therefore, no orientation is required. Accordingly, the configuration space is equal to the workspace $\mathcal{C}_P = \mathcal{X}$ and the person's phase space $\mathcal{ST}_P$ is described as $s_p \in \mathcal{ST}_P$, where $s_p = [x, y, v_x, v_y, t]^\top$.

The input space $\mathcal{U}_P$ for the person's action is $u_p \in \mathcal{U}_P$ which are linear accelerations $u_p = [a_x, a_y]^\top$. The kinodynamic model describing the person's motion is constrained by the following differential equations:

$$\dot{s}_p = dc(s_p, u_p) = \begin{bmatrix} v_x \\ v_y \\ a_x \\ a_y \\ 1 \end{bmatrix}. \tag{1}$$

It will be discussed later how the input variables u_p are calculated.

B. Robot in the state-space

We consider the robot model to be characterized by a unicycle robot model. Thereby, there appear nonholonomic constraints in the robotic dynamic model due to the rolling contacts between the rigid bodies. The phase state of the robot $\mathcal{ST}_R$ is described as $s_r \in \mathcal{ST}_R$, where $s_r = [x, y, \theta, v, \omega, t]^\top$. The robot action space is defined as $u_r \in \mathcal{U}_R$ where $u_r = [a_v, a_\omega]^\top$ are the translation acceleration and the rotation acceleration. Then, the resultant differential constraints are:

$$\dot{s}_r = dc(s_r, u_r) = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \\ a_v \\ a_\omega \\ 1 \end{bmatrix}. \tag{2}$$

C. Joint state-space

The joint state space $\mathcal{ST}$ consists of $\mathcal{ST} = \mathcal{ST}_R \times \bigcup \mathcal{ST}_{P_i}$, which considers the robot phase space $\mathcal{ST}_R$ and the union of every person's phase space $\mathcal{ST}_P$. Correspondingly, the joint state $s \in \mathcal{ST}$ is defined as $s = [s_r, s_{p_1}, \dots, s_{p_N}]^\top$.

Note that the variable time $t = t(s)$ is equal to all the states that s consists of. We will refer to the robot state $s_r(s) \in \mathcal{ST}_R$ and the person ith state $s_{p_i}(s) \in \mathcal{ST}_P$.

IV. Extended Social Force Model

We employ the Extended Social Force Model (ESFM) [7], based on [6], for navigation purposes since it provides a realistic model describing interactions among humans in typical social environments [5]. The ESFM considers humans and robots as free particles in a 2D space abiding the laws of Newtonian mechanics. The ESFM uses attractors and repulsors in the continuous space.

The attraction forces assume that the pedestrian n tries to adapt his or her velocity within a *relaxation time* k^{-1},

$$\boldsymbol{f}_n^{goal}(q_n^{goal}) = k\big(\boldsymbol{v}_n^0(q_n^{goal}) - \boldsymbol{v}_n\big), \quad (3)$$

where $\boldsymbol{v}_n^0(q_n^{goal})$ is the desired velocity vector to reach q_n^{goal}, and $\boldsymbol{v}_n$ is the current velocity.

The repulsive interaction forces are defined as follows:

$$\boldsymbol{f}_{n,z}^{int} = a_z e^{(d_z - d_{n,z})/b_z} \hat{\boldsymbol{d}}_{n,z}, \quad (4)$$

where $z \in Z$, being $Z = P \cup O \cup R$ is either a person, or a static object of the environment, or a robot. For each kind of interaction force corresponds a set of force parameters $\{k, a_z, b_z, \lambda_z, d_z\}$. The distance $d_{n,z}$ from the person n to the target z and $\hat{\boldsymbol{d}}_{n,z}$ is the unity vector $z \to n$. For further details, see [7], [5].

Accordingly, the resultant force is calculated as the summation:

$$\boldsymbol{f}_n = \boldsymbol{f}_n^{goal}(q_n^{goal}) + \sum_{j \in P \setminus n} \boldsymbol{f}_{n,j}^{int} + \sum_{o \in O} \boldsymbol{f}_{n,o}^{int} + \sum_{r \in R} \boldsymbol{f}_{n,r}^{int}, \quad (5)$$

where each target on the scene, either a person, or an obstacle, or a robot, contributes to $\boldsymbol{f}_n$.

A. SFM applied to the robot

The objective is to treat a robot as a free moving particle in the space, similarly to people as explained above. Unfortunately, nonholonomic constraints reduce the robotic platform mobility, although it has full reachability in $\mathcal{C}_R$. We need to bridge the gap and provide an adjustment that permits the robot being compatible with the ESFM. The resultant robot force $\boldsymbol{f}_r = \boldsymbol{f}_{r||\theta} + \boldsymbol{f}_{r\perp\theta}$ consists of a component in the translation direction $\boldsymbol{f}_{r||\theta}$, which directly transforms into a translational acceleration and an orthogonal force $\boldsymbol{f}_{r\perp\theta}$ that does not contribute to the robot translation. The robot rotation acceleration is computed in the following way:

$$\tau_r = \boldsymbol{r} \times \boldsymbol{f}_{r\perp\theta} + k_\tau \omega, \quad (6)$$

where $\boldsymbol{r}$ is the vector radii of our platform, oriented to θ and k_τ is a damping factor in order to avoid oscillations.

Algorithm 1 Proactive planning($q_r^{goal}, s_{ini}, t_{horizon}, K$)

```
1: Initialize $\mathcal{T}(\mathcal{V},\mathcal{E}) \leftarrow \{\emptyset\}$
2: $\mathcal{V} \leftarrow s_{ini}$
3: $s_{parent} = s_{ini}$
4: $\{q_{p_i}^{goal}\}$ = intentionality_people_prediction()
5: for $j = 1$ to $K$ do
6:     if $t(s_{parent}) > t_{horizon}$ then
7:         $q_r^g$ = sample($\mathcal{C}_R, q_r^{goal}$)
8:         $s_{parent}$ = nearest_vertex($q_r^g, \mathcal{T}$)
9:     end if
10:    $u_r$ = calculate_edge($s_{parent}, q_r^g$)
11:    $s_{new}$ = propagate_vertex($u_r, s_{parent}$)
12:    $J_{new}$ = calculate_cost($s_{new}, u_r, q_r^{goal}$)
13:    $\mathcal{V} \leftarrow \mathcal{V} \cup \{[s_{new}, J_{new}]\}$
14:    $\mathcal{E} \leftarrow \mathcal{E} \cup \{u_r\}$
15: end for
16: return minimum_cost_branch($\mathcal{T}$)
```

V. Proactive kinodynamic planning

All the main features of our proposed planner have been discussed in Sec. II, and can be summarized as:

- A kinodynamic solution is calculated.
- Proactive planning in which planning uses prediction information, and prediction is dependent on the plath calculated.
- Prior requirement: a global planner provides a valid global path.
- At each iteration, the planner provides a locally valid path.
- The path computed minimizes the perturbations on the scene, according to a cost function.

Additionally, the proactive planner is fast enough for a real time implementation, as demonstrated in Sec. VII.

Algorithm 1 has four inputs: the goal q_r^{goal}, the initial state s_{ini}, the horizon time $t_{horizon}$, and the number of vertices K. The q_r^{goal} provides the position and orientation of the final robot configuration. The initial state $s_{ini} \in \mathcal{ST}$ contains the information of the robot state plus all people's states considered on the scene. The horizon time $t_{horizon}$ specifies the temporal window used to forecast the plan and the predictions.

The algorithm builds a tree $\mathcal{T}(\mathcal{V},\mathcal{E})$ and returns the minimum cost branch. The edges $\mathcal{E}$ are the robot control inputs $u_r \in \mathcal{U}_R$, and the vertices $\mathcal{V}$ consist of the joint state $s \in \mathcal{ST}$ and the accumulated cost $J \in \mathbb{R}$ to reach that vertex.

With all those requirements in mind, we propose Alg. 1 which is inspired in a randomized kinodynamic planner [16], except for some particularities that will be discussed below.

A. Horizon time and depth exploration

The horizon time parameter sets the amount of time that the planner forecasts in order to obtain a path similar to a model predictive control (MPC). Although the set of inputs $\{u_r(t_{ini}), \dots, u_r(t_{horizon})\}$ is calculated, only the first input command is executed and a new set of inputs is calculated in the next iteration. The horizon time bounds the region of exploration $\mathcal{C}_R$ to a circle radii equal to $t_{horizon} \cdot v_{max}$ as depicted in Fig. 3.

Usually $q_r^{goal} \notin \mathcal{C}_R$ due to the horizon time limit, and accordingly, our approach obeys a depth search strategy to develop branches of the tree $\mathcal{T}$ until the horizon time is reached (Line 6 in Alg. 1).

B. Space exploration

The space exploration is done in $\mathcal{C}_R$ where a set of random goals q_r^g are randomly calculated in order to extend the tree $\mathcal{T}$. These random goals q_r^g are attractors that generate valid robotic paths in $\mathcal{ST}$ until $t_{horizon}$ is reached, as explained below. Since there is a strong time restriction, q_r^g are sampled in the boundary of $\mathcal{C}_R$, to avoid bias and to ensure that the paths generated indeed expand $\mathcal{T}$.

The random goals q_r^g can be seen in Fig. 3. The sampling is done using a Gaussian distribution centered at the nearest $q \in \mathcal{C}_R$ to q_r^{goal} and the variance values depend on the density of nearby people: when the density grows, the variance also augments.

C. Find nearest vertex

Once a new random goal q_r^g is generated, the planner finds the nearest vertex $\mathcal{V}$ in $\mathcal{T}$ to be the parent vertex for the new branch to be calculated. If only pure distances are calculated in $\mathcal{C}_R$, there would exist a strong bias to select "old" vertices that are near $t_{horizon}$. The calculation of the new weighted distance is done as follows:

$$d(s_r, q_r^g) = ||\mathbf{x}_r - \mathbf{x}_r^g|| + c_\theta ||\theta_r - \theta_r^g|| + c_{time} ||t_r - t_{ini}||. \quad (7)$$

D. Edge calculation

Edges $\mathcal{E}$ are robot control inputs $u_r \in \mathcal{U}_R$. Despite that the joint state takes into account all people, we can only select the input actions for the robot platform. We calculate the resultant robot force $\boldsymbol{f}_r$ by making use of the ESFM (5). This force is transformed into an acceleration using (6), and thus, a robot action $u_r = f_r/m_r$ that takes into account the mass of the robot m_r.

The goal of the robot is the random goal q_r^g and at the same time reacts to the environment obstacles, that is, it takes into account the states of the nearby people $s_{p_1}, \dots, s_{p_N}$. The different paths are computed by introducing a set of random goals q_r^g that steer the robot to rapidly explore $\mathcal{C}_R \times time$.

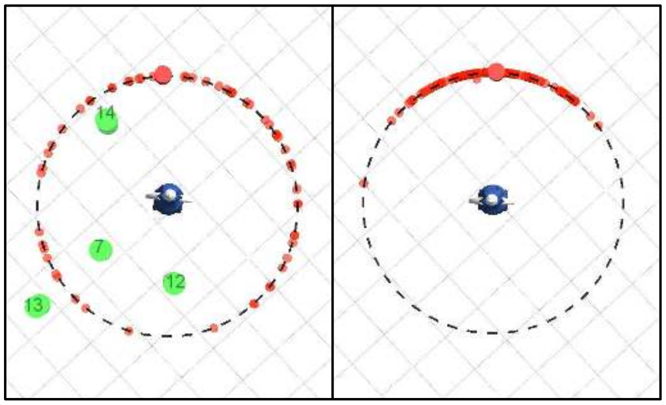

Fig. 3. Random goals q_r^g distribution, on the *right* there are no people in $\mathcal{C}_R$ and search is concentrated on the goal direction. On the *left* the density of nearby people on the scene is higher, and thus, the q_r^g sampling distribution is widespread.

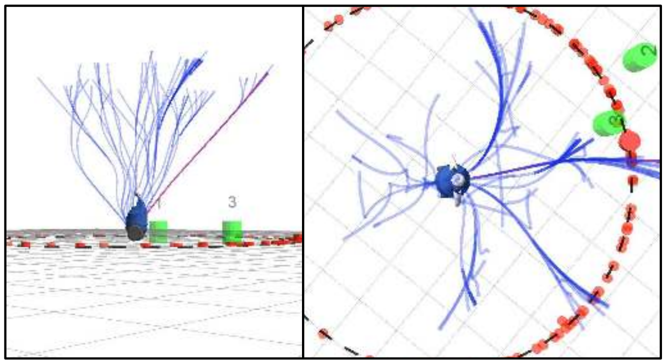

Fig. 4. Tree $\mathcal{T}$ of paths in the space $\mathcal{X} \times time$. On the *left*, the z axis represents time. On the *right* projection of $\mathcal{T}$ in $\mathcal{X}$.

Algorithm 2 Vertex propagation(u_r, s_{parent})

```
1: $s_r^{new} = s_r(s_{parent}) + dc(s_r(s_{parent}), u_r) \cdot \Delta t$
2: for $i = 1, \dots, N$ do
3:     if $s_{p_i}(s_{parent}) \nsubseteq q_{p_i}^{goal}$ then
4:         $u_{p_i} = \boldsymbol{f}(q_{p_i}^{goal}, s_{parent}, s_r^{new})/m_i$
5:         $s_{p_i}^{new} = s_{p_i}(s_{parent}) + dc(s_{p_i}(s_{parent}), u_{p_i}) \cdot \Delta t$
6:     end if
7: end for
8: return $s_{new} = [s_r^{new}, s_{p1}^{new}, \dots, s_{pN}^{new}]^\top$
```

E. Vertex propagation

Since the action calculated u_r only propagates the robot state s_r, we must update the joint state s for every person considered. We propose a proactive approach in which the computed planning action u_r is integrated with the prediction algorithm.

First of all, we need to infer human intentions. As proposed in [4], we calculate the most expectable $q_{p_i}^{goal}$ for every person on the scene. This calculation is carried out only once, at the initialization of the algorithm (line 4 in Alg. 1).

We adapt the human prediction algorithm [5] to obtain a single propagation s_{new} from a given initial state $s_{parent} \in \mathcal{ST}$.

Algorithm 2 first propagates the robot state s_r accordingly to u_r and integrates the differential equation (2) by using Euler integration. Then, for every person on the scene, and if the person has not reached its inferred goal $q_{p_i}^{goal}$ (line 3 in Alg. 2), an action u_{p_i} is calculated depending on the people on the scene and the new robot state s_r^{new} (line 4 in Alg. 2).

F. Cost function and path selection

We propose a metric that measures social disturbances while navigating: the *social work* [18]. The amount of *social work* carried out by the robot from t_{ini} to $t_{horizon}$:

$$W_R = \sum_{t=t_{ini}}^{t_{hzn}} \boldsymbol{f}_r(t) \cdot \Delta \boldsymbol{x}_r(t), \quad (8)$$

where $\boldsymbol{f}_r$ takes into account both the steering force (3) and the summation of social-forces due to nearby people or other obstacles (5) and it is multiplied by the variation of position $\Delta \boldsymbol{x}_r$ at each t. Similarly, we can define the summation of *social work* carried out by the people on the scene induced by the robot movement:

$$W_P = \sum_{t=t_{ini}}^{t_{hzn}} \sum_{i \in P} \boldsymbol{f}_{r,i}(t) \cdot \Delta \boldsymbol{x}_i(t). \tag{9}$$

We have measured the *social work* of the people W_P due to the robot plan, and the *social work* carried out by the robot W_R.

For every new vertex the total cost is:

$$J = ||\boldsymbol{x}_r - \boldsymbol{x}_{goal}|| + k_\theta ||\theta_r - \theta_{goal}|| + \\ k_{robot} W_R + k_{people} W_P. \tag{10}$$

Most of the time, reaching q_r^{goal} may not be possible. Nevertheless, the algorithm calculates several branches in the tree $\mathcal{T}$ and returns the branch with minimum cost function J at time $t_{horizon}$.

VI. Simulations

Real experimentation is a delicate matter when there are people involved. For this reason, we validate our planner in a simulated environment before any real interaction with people takes place. The simulated environment [7] consists of static obstacles and multiple people modeled as dynamic obstacles following the ESFM, quite similar to a real urban environment.

The number of vertices $K = 500$ in the planner is fixed throughout all simulations and experiments. The processing time highly depends on the number of people evaluated since we are considering the propagation of the state for every person on each vertex calculated. The average processing time, if an average of 8 people is considered to appear on the scene, is around $0.19s$, processed in a Intel Core2 Quad CPU Q9650 @ 3.00GHz and memory 3.8 GiB. The simulated scenario is as follows: the robot receives a query to a goal while a group of pedestrians walk in the area in different situations.

A. Learning parameters in simulations

Before evaluating the method performance, we should provide an initial estimation of the planner parameters. Parameters in (7) were manually chosen ($c_\theta = 1.0$, $c_{time} = 0.25$) to build a tree without bias as explained in Sec. V-C. For the parameters in (10) and $t_{horizon}$, we have performed over 7000 experiments using a Monte Carlo approach to sample the cost parameters, and then, we averaged the experiment performances to find the optimal values. We provide statical robustness to the method by doing a large number of experiments and then calculate the expectation of the performance depending on the parameters. In Fig. 5 is depicted the learning results for the $t_{horizon}$ parameter.

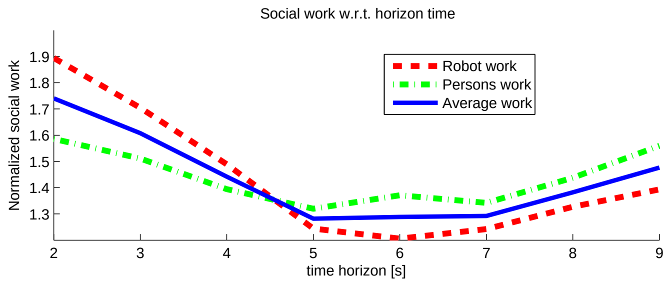

Fig. 5. Learning results in the simulated environment for the $t_{horizon}$ parameter, showing a minimum in $t_{horizon} = 5s$ for the average function. These functions are normalized for comparing purposes.

A short $t_{horizon}$ results on a straight trajectory, where the robot accelerates and stops if an obstacle appears, which is not an efficient behavior. Intuitively, the higher $t_{horizon}$ the better. However, as can be seen in Fig. 5 there is a degradation of the performance of both *social works*. As we observed for high horizons, there were a short number of branches in the tree to explore good solutions, and thus, a higher number of vertices would be required.

The results, $k_\theta = 1$, $k_{robot} = 2$, $k_{people} = 8$ and $t_{horizon} = 5s$. These parameters are highly dependent on the learning scenario and the number of vertices K.

B. Simulations testing

We have tested the algorithm in the same simulated environment using the parameters obtained above. We have compared our approach with a *reactive* planner proposed in [18], which takes into account people on the scene.

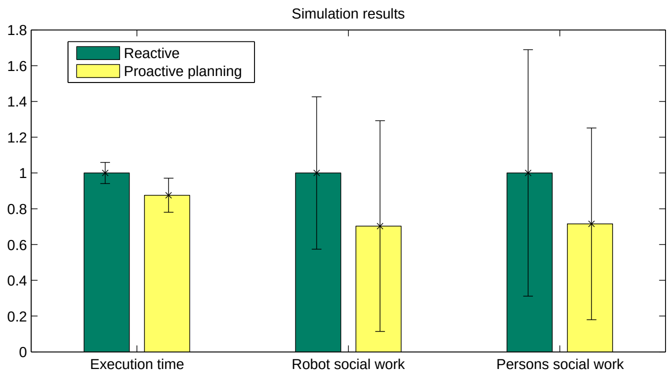

Fig. 6. Simulation results, normalized to the *reactive* approach results.

We have compared three different values as can be seen in Fig. 6. The execution time refers to the average time for the robot to get to its goal under the presence of people. Our approach shows a better performance for the *social work* produced by the people and the robot. These magnitudes are clearly correlated since we have observed that in general the proactive planner tries to describe trajectories that avoid interactions if possible, while the *reactive* approach is not "intelligent" enough to avoid unnecessary interactions on time.

VII. Experiments

Finally, we have implemented the algorithm to run in a real robot, the Tibi&Dabo robots [7] in a controlled real environment.

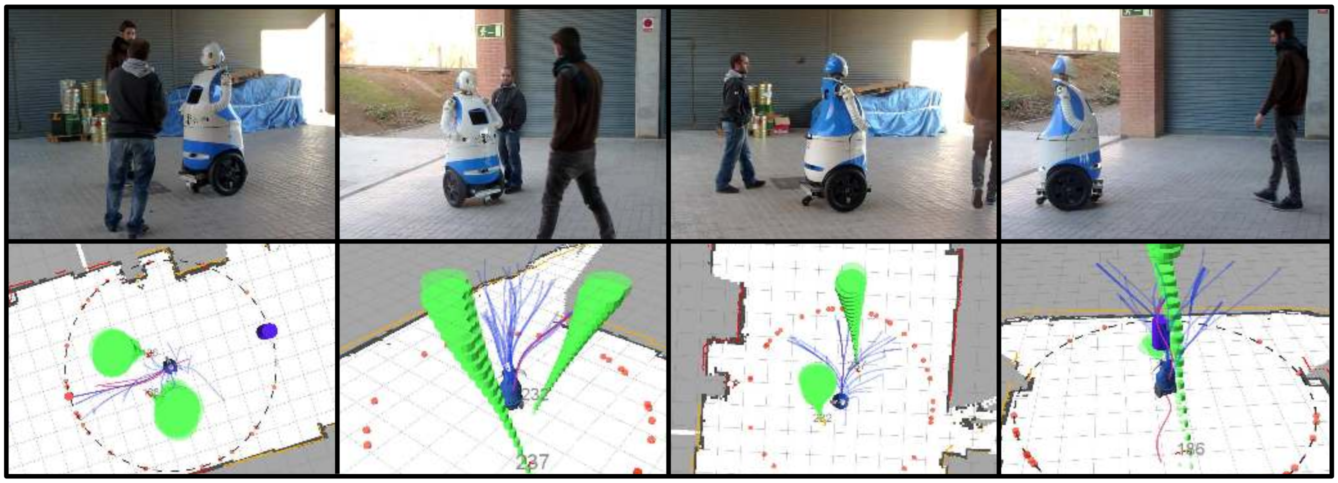

Fig. 7. Examples during experimentation in real environments. Dabo navigates while considering other people on the scene. In the bottom row of the figure appears an interface, where people are plotted as green cylinders and their predictions are drawn in the z axis, which corresponds to time. The tree of paths calculated by the robot appear in blue and the best path is a red line.

In Fig. 7 is depicted a series of experiments with people. We observed a better behavior of the proactive kinodynamic planner with respect to our previous *reactive* approach, although more experiments are required, in different scenarios, and more people on the scene. For more details, check the videos at the author's webpage `http://www.iri.upc.edu/people/gferrer`

VIII. Conclusion and future work

In this paper we have presented a proactive kinodynamic planner for urban environments that calculates in real time a local path that minimizes the disturbances to other pedestrians. The solution trajectories take into account kinodynamic and nonholonomic restrictions which are mandatory considerations for a realistic navigation in such highly time-variant scenarios like urban environments.

Additionally, our approach shows proactive traits since the robot tends to initiate change rather than reacting to events. The cost for being proactive is a more intensive processing since we must propagate the state of moving pedestrians accordingly to the robot propagation.

The trajectory calculated minimizes the amount of *social work* produced by the robot navigation at the same time that minimizes its navigation work and distance to goal.

For future works we need to extensively test the algorithm in different environments and with more people.

References

[1] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3931–3936.

[2] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[3] M. Luber, L. Spinello, J. Silva, and K. O. Arras, "Socially-aware robot navigation: A learning approach," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 902–907.

[4] G. Ferrer and A. Sanfeliu, "Bayesian human motion intentionality prediction in urban environments," *Pattern Recognition Letters*, vol. 44, pp. 134–140, 2014.

[5] ——, "Behavior estimation for a complete framework of human motion prediction in crowded environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 5940–5945.

[6] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[7] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1688–1694.

[8] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[9] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4293–4298.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[11] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[12] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[13] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, 1996, pp. 3375–3382.

[14] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 1999, pp. 341–346.

[15] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 508–513.

[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[17] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[18] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *European Conference on Mobile Robotics, ECMR.*, 2013, pp. 331–336.

[19] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 2145–2152.