Motion Planning in Crowds using Statistical Model Checking to Enhance the Social Force Model

Alessio Colombo[1], Daniele Fontanelli[1], Axel Legay[2], Luigi Palopoli[1] and Sean Sedwards[2]

Abstract— **Crowded environments pose a challenge to the comfort and safety of those with impaired ability. To address this challenge we have developed an efficient algorithm that may be embedded in a portable device. The algorithm anticipates undesirable circumstances in real time, by verifying simulation traces of local crowd dynamics against temporal logical formulae. The model incorporates the objectives of the user, pre-existing knowledge of the environment and real time sensor data. The algorithm is thus able to suggest a course of action to achieve the user's changing goals, while minimising the probability of problems for the user and others in the environment.**

To demonstrate our algorithm we have implemented it in an autonomous computing device that we show is able to negotiate complex virtual environments. The performance of our implementation demonstrates that our technology can be successfully applied in a portable device or robot.

I. Introduction

With unimpaired ability, pedestrians are able to negotiate crowded areas with few problems. With reduced ability or under panic conditions [1], finding a good strategy to proceed can be challenging. As a result, people afflicted by a decline in physical or cognitive abilities can be discouraged from attending crowded places, with a consequent negative impact on their physical condition (reduced exercise), on the quality of their nutrition (reduced fresh food) and on their psychological wellbeing (reduced social contact). Motivated by these considerations, the DALi project [2] aims to devise an intelligent 'walker' (an assistive wheeled device) that detects the presence of other pedestrians in the environment, anticipates their intent and plans an appropriate path that is suggested to the user via a combination of audio, visual and haptic interfaces. In this work we present an efficient algorithm that employs advanced modelling and verification techniques to address the path planning problem in a crowded and unfamiliar environment.

Succinctly, the problem is one of devising an online motion planning algorithm for an autonomous agent (the user) in a dynamic environment. The position of most fixed objects (e.g., buildings and rooms) are known a priori, but the algorithm must account for the possibility of changes, such as temporary obstructions. The environment contains moving objects (i.e., other pedestrians), whose positions and velocities cannot be known before they are encountered. The overall goal is to allow the user to visit pre-defined locations in the environment, while avoiding collisions, crowding and delays. The output of the algorithm is a *suggested* trajectory, so the algorithm must be reactive to the potentially uncooperative response of the user. Practically, the algorithm will be implemented in a low power embedded computing device and must be sufficiently efficient to make course corrections in a time of the order of seconds. This time scale is dictated by the typical velocities of pedestrians and by the fact that frequent readings help to reduce the random errors produced by sensors.

Our solution is a two-tiered algorithm, comprising a low level predictive mathematical model of pedestrian dynamics, managed by a statistical model checking (SMC) engine that checks temporal logical properties expressing the high level goals and constraints of the user. The algorithm uses dynamic input from sensors to reconstruct the user's position from fixed objects and to account for non-fixed objects, such as other pedestrians and temporary obstructions.

A. Related work

Our work is related to sampling methods (e.g., [3]–[5]) and to recent methods using temporal logic (.e.g, [6]–[8]). It is also related to methods that predict behaviour based on models parametrised with data from sensors (e.g., [9]).

In common with existing sampling methods, our algorithm uses randomisation to cover an intractably large configuration space. In contrast to many existing uses of sampling, however, we do not assume a fixed environment. In our application the environment contains both fixed and dynamic elements, such that a single optimal path cannot be defined a priori. Hence, the problem we solve by sampling is not one of creating an optimal global plan, but one of finding an optimal local plan given a changing environment.

Model checking is an automatic process to verify that a system satisfies a property specified in temporal logic. In the present context, temporal logic can express complex dynamical properties such as "the user will visit all the desired locations in a specified sequence, within the specified time" and "the user will never get too close to any other pedestrian". If the notion of an optimal path can be so defined, the principles of model checking can be used to directly synthesise a 'correct' motion planner or to prove that an existing motion planner is correct [6]–[8]. The use of probabilistic model checking in combination with the

This work is partially funded by the Devices for Assisted Living (DALi) European project, reference 288917. http://www.ict-dali.eu/dali

[1]A. Colombo, L. Palopoli and D. Fontanelli are with the Dep. of Information Engineering and Computer Science, University of Trento, Via Sommarive 5, Trento, Italy {colombo,palopoli,fontanelli} at disi.unitn.it

[2]A. Legay and S. Sedwards are with INRIA Rennes Bretagne Atlantique, Rennes, France {axel.legay,sean.sedwards} at inria.fr

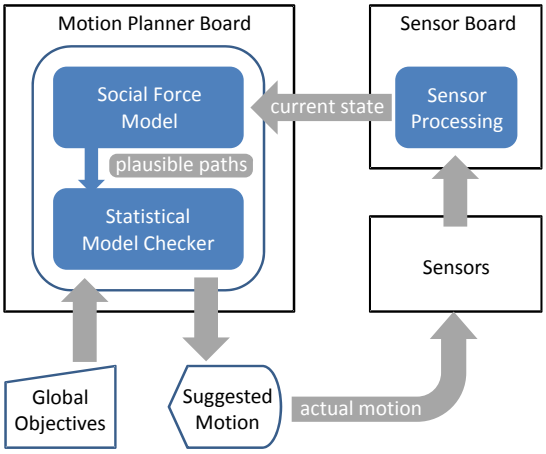

Fig. 1. Diagrammatic overview of the motion planning framework. The sensor board detects the current state of objects in the environment. This state is used by the social force model to generate plausible future paths of the user and other pedestrians. The distribution of paths is verified against the global objectives of the user in order to suggest an optimal course.

theory of stochastic hybrid automata [10] is particularly appealing for control and robotic applications where a non-zero probability of failing the mission can be tolerated. For example, in [11], [12] for air traffic control or in [13], [14] for industrial robotics. Combining model checking with sampling, algorithms can be constructed which provably converge to optimal schedulers [8]. Standard model checking algorithms are computationally intensive, hence existing applications have used model checking offline. By using *statistical* model checking, we are able to perform online verification. We do not prove correctness, but find a local plan that maximises the probability of success.

We believe our work is novel in using formal verification in real time to solve an adaptive motion planning problem in a dynamic environment.

B. Organisation of the paper

The rest of the paper is organised as follows. Section II-A gives an overview of our approach. Section II-B introduces our mathematical model in detail and Section II-C introduces the basic notions of statistical model checking. Section III gives a detailed description of our algorithm and implementation, while Section IV describes the results of a number of experiments that demonstrate the utility of our approach. Section V discusses our choices and highlights areas of ongoing development.

II. Preliminaries

A. Overview of the approach

Fig. 1 gives a high level overview of the algorithm. At each iterative step the algorithm acquires the state of the system, comprising the position of static objects and the position and velocity of the user and of other people in the environment.

Given the current state, the algorithm hypothesises alternative courses of action using the *social force model*. Each hypothesised trajectory is formally verified (model-checked) against properties that express goals and constraints required for the user's trajectory (i.e., where the user wants to go, obeying the appropriate social rules). This leads to a statistical distribution of potentially successful trajectories. The algorithm uses this distribution to choose an immediate action that maximises the probability of achieving the user's objectives and minimises the probability of problems. In this probabilistic context, the measurement noise is considered as an additional source of stochasticity.

The social force model may be programmed with the user's objectives and is an efficient way to describe the continuous interactions that allow pedestrians to avoid collisions. The model also includes stochasticity to model the typical unpredictability of human behaviour. We use the stochasticity to generate a random sample of possible futures and choose the course of action that maximises the probability of success. Such trajectories respect the basic social and physical laws of pedestrian interactions and include the possibility of unpredicted behaviour. Their *distribution* allows the algorithm to choose a course of action that maximises the probability of success.

The stochasticity, while realistic, places an upper bound on the predictive accuracy of the model. Moreover, the model alone cannot account for the overall "mission" of the user. The predictive model needs to be managed *reactively*. Fortunately, the field of statistical model checking (SMC) encapsulates the technologies that we require to do this. SMC provides efficient algorithms to verify hypothesised trajectories against the user's constraints and objectives expressed in temporal logic. SMC can estimate the probability of success and bound the error of the estimation.

The key elements of our approach are (*i*) the social force model to hypothesise trajectories that respect low level social and physical "forces"; (*ii*) temporal logic to express the high level goals of the user and (*iii*) a statistical model checker to verify the traces with respect to the goals.

B. The social force model

The social force model [1], [15]–[17] combines real and psychological forces to predict the behaviour of pedestrians in crowds, under normal and panic situations. The model recognises that pedestrians are constrained by the physical laws of motion and also by social 'laws' that can be modelled by external forces. The model considers an environment comprising fixed objects (walls) and moving agents (pedestrians) that respond to attractive and repulsive forces, originated by social and physical interactions.

The model is constructed in two dimensions [1], [15]–[17], with agents represented as circular discs. In what follows we adopt the convention of denoting vectors in bold type. Thus, agent i has mass m_i centred at position $\mathbf{x}_i \in \mathbb{R}^2$ in the environment, radius r_i and velocity $\mathbf{v}_i \in \mathbb{R}^2$. The linear model for the i-th agent is given by

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v}_i \\ \dot{\mathbf{v}}_i = \dfrac{\mathbf{v}_i^0 - \mathbf{v}_i}{\tau_i} + \dfrac{\mathbf{f}_i + \boldsymbol{\xi}_i}{m_i} \end{cases} \tag{1}$$

$\mathbf{v}_i^0$ is the *driving (desired) velocity* of agent i, represented by a product of speed amplitude v_i^0 and normalised direction $\mathbf{e}_i^0$, which is given by the direction of the line joining the initial and desired configurations. τ_i is the time taken to react to the difference between desired and actual velocity, while $\boldsymbol{\xi}_i$ is a *noise term* (a random variable) that models random fluctuations not accounted for by the deterministic part of the model. The inclusion of the noise term makes the model stochastic, such that a different trajectory is generated each time (1) is solved. This allows the application of SMC and serves to avoid deadlocks that might arise if, by chance, some of the deterministic forces are equal and opposite.

$\mathbf{f}_i$ is the force acting on agent i resulting from other objects in the environment and, hence, given by

$$\mathbf{f}_i = \sum_{j\neq i}[\mathbf{f}_{ij}^{\mathrm{soc}} + \mathbf{f}_{ij}^{\mathrm{att}} + \mathbf{f}_{ij}^{\mathrm{ph}}] + \sum_{b}[\mathbf{f}_{ib}^{\mathrm{soc}} + \mathbf{f}_{ib}^{\mathrm{ph}}] + \sum_{c}\mathbf{f}_{ic}^{\mathrm{att}}. \quad (2)$$

The first term on the right-hand side of (2) includes all the forces on agent i resulting from interactions with other agents: $\mathbf{f}_{ij}^{\mathrm{soc}}$ is the repulsive social force that inhibits agents getting too close, $\mathbf{f}_{ij}^{\mathrm{att}}$ is the attractive social force that brings friends together, $\mathbf{f}_{ij}^{\mathrm{ph}}$ is the physical force that exists when two agents touch. The second summation includes the forces acting on agent i as a result of the boundaries (walls): $\mathbf{f}_{ib}^{\mathrm{soc}}$ is the social force that inhibits agent i from getting too close to boundaries, $\mathbf{f}_{ib}^{\mathrm{ph}}$ is the physical force that exists when agent i touches boundary b. Finally, $\mathbf{f}_{ic}^{\mathrm{att}}$ is the attractive social force that draws agent i towards fixed objects of incidental interest (shops, cafés, toilets, etc.).

In general, the force acting on any agent is calculated with respect to the distance between its centre of mass and all other visible objects. Since the model mixes both notional (social) and real forces, the mass m_i is notionally the real mass of agent i. Other parameters can be used to model the unique characteristics of individual agents. For example, the latency factor τ_i can be used to model the possibly reduced mobility of agent i. Full details of these and other parameters can be found in [1]. In [18] we show how the model may be parametrised from captured motion.

C. Statistical and probabilistic model checking

Model checking is an automatic technique to verify that a system satisfies a property [19]. Typically, the system has discrete states and the property is specified in temporal logic. The output of standard model checking algorithms is either *true* or *false*, with the possibility to give corresponding examples or counter-examples. The logics used are desired to be expressive, but must be decidable and tractable. Typical logics for standard model checking include LTL and CTL [20]. To give a result with certainty, the algorithms effectively perform an exhaustive exploration of the state space of the system. The number of states scales exponentially with the number of interacting components in the system, leading to a 'state explosion problem' [19] that can make standard model checking slow or intractable.

Probabilistic model checking extends the standard notion to include probabilistic or stochastic transitions. These can express the uncertainties of modelling and reality. The output of probabilistic model checking algorithms is the probability that an arbitrary execution of the system will satisfy a given property. Such properties are specified in probabilistic or stochastic logics, such as PCTL and CSL [20]. The probabilistic model checking problem is solved with *numerical* model checking algorithms. These calculate the notionally exact probability by considering all the states. As such, they suffer the same state explosion problem as standard model checking algorithms.

Statistical model checking (SMC) is a type of probabilistic model checking that avoids the state explosion problem by estimating the probability of a property ϕ from executions (simulations) of the system. Given N independent simulation traces $\omega_i, i \in \{1 \dots N\}$, and a model checking function $\mathbb{1}(\omega_i \models \phi) \in \{0, 1\}$ that indicates whether $\omega_i \models \phi$ (read "ω_i *satisfies* ϕ"), the probability γ that an arbitrary execution satisfies ϕ is estimated using $\gamma \approx 1/N \sum_{i=1}^{N} \mathbb{1}(\omega_i \models \phi)$. This reduces the probabilistic model checking problem to estimating the parameter of a Bernoulli random variable, hence the confidence of the estimate can be guaranteed by standard statistical bounds (e.g., the Chernoff bound [21]). In general, the confidence of the estimate increases with increasing N. In comparison to standard and numerical model checking, SMC does not require decidable logics nor a finite state space, making it particularly suitable for the present application that considers continuous time and space.

Bounded Linear Temporal Logic: Our model checking engine is based on the PLASMA-lab library [22]. PLASMA-lab implements the function $\mathbb{1}(\omega_i \models \phi)$ using bounded linear temporal logic (BLTL [23]) to express the property ϕ:

$$\phi = \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{F}_{\leq t}\phi \mid \mathrm{G}_{\leq t}\phi \mid \phi\,\mathrm{U}_{\leq t}\phi \mid \mathrm{X}\phi \mid \alpha$$

$\vee, \wedge$ and $\neg$ are the standard logical connectives and α is a Boolean constant or an atomic proposition constructed from numerical constants, state variables and relational operators. X is the *next* temporal operator: $\mathrm{X}\phi$ means that ϕ will be true on the next step. F, G and U are temporal operators bounded by time interval $[0, t]$, relative to the time interval of any enclosing formula. We refer to this as a *relative interval*. F is the *finally* or *eventually* operator: $\mathrm{F}_{\leq t}\phi$ means that ϕ will be true at least once in the relative interval $[0, t]$. G is the *globally* or *always* operator: $\mathrm{G}_{\leq t}\phi$ means that ϕ will be true at all times in the relative interval $[0, t]$. U is the *until* operator: $\psi\mathrm{U}_{\leq t}\phi$ means that in the relative interval $[0, t]$, either ϕ is initially true or ψ will be true until ϕ is true. Combining these temporal operators creates complex properties with interleaved notions of *eventually* (F), *always* (G) and *one thing after another* (U). A detailed description of the semantics of BLTL is given in [23].

III. SMC–BASED MOTION PLANNER

Our motion planner is based on the scheme depicted in Fig. 1 and Algorithm 1. The planner assumes the existence of a pre-calculated *global plan* (GlobalPlan) that visits the user's objectives in an a priori optimal way, that is, considering all things known in advance. Typically, the global plan

Algorithm 1 The planning algorithm

```
1: function FINDLOCALPATH(state_user, state_ped1, state_ped2, ..., Map,
   GlobalPlan, Formula, N)
2:     Real P_curr, d_curr, P_best, d_best;
3:     [P_best, d_best]=[0,∞];
4:     for α_curr ∈ {0, ±25, ±50, ±75, ±90} do
5:         [P_curr, d_curr] = SMC(N, Formula);
6:         if is_better([P_curr, d_curr], [P_best, d_best]) then
7:             α_best = α_curr;
8:             [P_best, d_best] = [P_curr, d_curr];
9:         end if
10:    end for
11:    if P_best == 0 then
12:        return STOP;
13:    else
14:        return α_best;
15:    end if
16: end function
```

is computed with respect to a map of the static objects in the environment, the user's objectives and predicted anomalies (e.g., known crowded areas). Any contradiction of the a priori assumptions (e.g., an unforeseen blockage) triggers a recalculation of the global plan. We do not describe this recalculation in the present work.

The sensor board provides the current state of the local environment, located with respect to the global plan: the position and velocity of the user (state_{user}); the positions and velocities of other pedestrians ($\text{state}_{ped_1}, \text{state}_{ped_2}, \ldots$); the position of static objects (Map). The algorithm calculates a local way point $\mathbf{w}$, which is the user's point of greatest straight line progress along the global plan within the sensor range. $\mathbf{w}$ is used to calculate the user's driving velocity $\mathbf{v}^0$, assuming a constant desired speed. The driving velocities of the other pedestrians are estimated from their current velocities.

The algorithm uses the above information to construct social force models (1) of the local environment. Specific characteristics (e.g., τ_i) of other pedestrians are unknown to the algorithm, so it assumes the default values given in [1]. In the current implementation we construct the noise term $\boldsymbol{\xi}_i$ from two normal distributions; one for the magnitude and one for the direction. The results presented here are based on heuristically estimated parametrisations of these distributions, which appear to be adequate. In [18] we present a way of obtaining better parameters from captured motion.

The motion planner assumes the user will follow the global plan, but need to temporarily deviate to avoid collisions. The output of the algorithm is a suggested deviation, α_{best}, in the range ± 90 degrees relative to the user's direct path to $\mathbf{w}$. To find α_{best}, the algorithm constructs models for each hypothesised deviation in the set $\{0, \pm 25, \pm 50, \pm 75, \pm 90\}$. These values are chosen to span ± 75 degrees using a tractable number of different values, with ± 90 included in case the user needs to sidestep an obstacle (see [18]). Each model is then investigates using statistical model checking.

The algorithm sets $\alpha_{curr} \in \{0, \pm 25, \pm 50, \pm 75, \pm 90\}$ and calls function SMC with arguments N and Formula. SMC estimates the probability of success P_{curr} for a particlar deviation α_{curr} by the proportion of N simulation traces that satisfy the BLTL property Formula. The value of α_{curr} is used as the *intial* deviation: the user's driving velocity is initially rotated by α_{curr}, but at each successive step of the simulation the deviation from a direct path to $\mathbf{w}$ is reduced to zero. This ensures that the user will eventually be close to the global plan.

BLTL is expressive enough to define complex sequences of high and low level requirements. For the results presented here, Formula merely expresses the basic constraints of the user:

$$(\mathrm{G}_{[0,T_{horizon}]} \bigwedge_{i \neq u} \|\mathbf{x}_u - \mathbf{x}_i\| > 0.5) \wedge (\mathrm{F}_{[0,T_{horizon}]} \|\mathbf{x}_u - \mathbf{w}\| < 0.2) \quad (3)$$

$\mathbf{x}_u$ denotes the position of the user and $\|\cdot\|$ denotes Euclidean distance. Intuitively, (3) means that "in the next $T_{horizon}$ time units the user will get no closer than 0.5m to any other pedestrian and will eventually be less than 0.2m from the global plan".

$T_{horizon}$ is chosen to be the expected time for the user to walk a distance equivalent to the range of the sensors. Using a higher value might produce impossible trajectories that pass through unseen fixed objects; using a lower value might exclude possible collisions. In our implementation we use $T_{horizon} = 4$s.

For each hypothesised deviation α_{curr}, the SMC function returns the probabillity of success P_{curr} and the expected distance from the global plan, d_{curr}. These are used by function is_better to decide α_{best}. is_better chooses the smallest $\mid \alpha_{curr} \mid$ which maximises P_{curr}. Ties are resolved by choosing the $alpha_{curr}$ with smallest d_{curr} or randomly if d_{curr} also ties. If $P_{best} == 0$ the user is required to stop (the global plan will be recalculated).

$T_{decision}$ is the actaul time the algorithm takes to make its predictions and must be less than the time period it is predicting, i.e., $T_{horizon}$. In practice $T_{decision}$ is bounded below by the performance of the hardware, the complexity of the environment (fixed and moving objects) and the confidence required (controlled by the number of simulations, N). In our implementation, $T_{decision} \approx 1s$.

At each decision point α_{best} is suggested to the user. The user may ignore this suggestion and move in a different direction, but the operation of the algorithm in the next decision period remains the same: α_{best} is calculated according to the global plan and the actual positions and velocities of the user and other pedestrians. Since the user specifies the global plan, when generating hypothesised traces the algorithm assumes that the user is compliant, however $\boldsymbol{\xi}_{user}$ may be used to model a lack of compliance. The present work does not consider how the user's non-compliance might affect predictions.

Given an accurate stochastic model of the behaviour of pedestrians, the Chernoff bound [21] predicts that with $N = 10$ simulation runs the estimate of the probability of success has a maximum error of ± 0.3 with probability 0.7. With $N = 50$ the probability of success has a maximum error of ± 0.2 with probability 0.90. In general, the statistical confidence

of the estimate increases with increasing N, but this only increases the probability of choosing the correct α_{best}. The predictive power of the model is bounded by its stochasticty. Thus, given finite computational power, we choose a value of N that balances the reactive and predictive aspects of the algorithm. That is, we choose a value of N that allows us to make $T_{decision}$ sufficiently small.

The algorithm solves (1) using a standard ODE solver [24], which produces traces comprising a sequence of states at discrete time points. Since the model given in Section II-B is based on continuous time and space, to guarantee properties that rely on the distance between objects it is necessary to choose time points that are sufficiently close. This is achieved by the ODE solver using adaptive time steps.

Simulating the traces accounts for most of the computational cost of the algorithm. We have found our chosen ODE solver to be efficient and presume its performance scales in a standard way with respect to the number of visible moving agents M and the complexity of their interactions. Since the forces in the model are dependent on the distances between agents, there is an additional $\mathcal{O}(M^2)$ cost, however M is bounded by the range of the sensors.

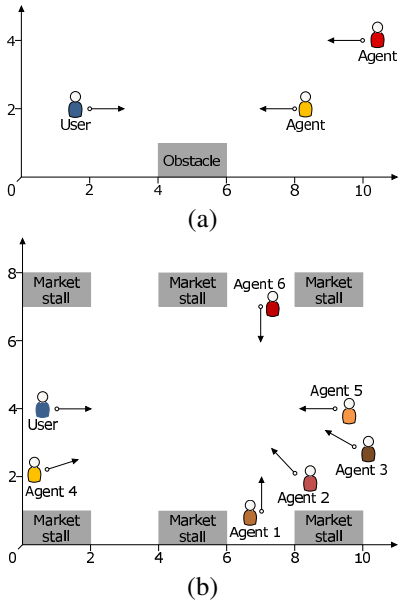

Fig. 2. Scenarios used to test the algorithm, *scenario 1* (a) and, *scenario 2* (b).

IV. Simulations

To demonstrate our algorithm we have implemented a prototype on a off-the-shelf, low power embedded system, the Beagleboard xM[1]. It is a portable device that may run from battery power and provides performance comparable to a small computer. We use PLASMA-lab [22] as the statistical model checking library. To test the algorithm we have created a virtual environment that evolves according to the Social Force Model and contains fixed objects and other pedestrians that react to the user's presence.

The pedestrians are assigned individual global plans to simulate their objectives and individual parameters that reflect the variation seen in reality. The values of the parameters are based on the ones estimated in [1] and two different but correlated sets, one for the planner and one for the virtual environment, have been defined in order to increase the sense of reality. The noise term ξ has been differentiated as well, the standard deviation of the two normal distributions in the planner has been set as the double of the one in the virtual environment.

In this way we simulate pedestrians that are reactive to the user and each other, with behaviour that is realistically unpredictable. Moreover, the simulated device has limited omnidirectional sensing range, we suppose it is able to detect agents moving within a radius of 4 meters with respect to the current position of the user. In the final application, a sensor board connected to the single board computer will provide the real (estimated) positions and velocities of the user and nearby pedestrians.

We compared three different strategies:

- SMC with the Social Force Model ($SMC+SFM$): our novel approach, where Algorithm 1 computes the local plan. When detected, an agent is supposed to evolve according to the Social Force Model.
- SMC with a linear motion model ($SMC + LIN$): similar to $SMC+SFM$ but agents evolve according to a different and simpler model. When detected, an agent is suposed to keep moving with same speed and same direction.
- Social Force Model only (SFM): we analyze the evolution of the environment without any decision points ($T_{decision} = \infty$).

For $SMC+SFM$ and $SMC+LIN$ we use the temporal logic formula defined by Eq. (3). We also used the following parameters: $T_{horizon} = \{1, 2, 4, 6, 8\}$, $T_{decision} = 1$ and $N = 50$. We performed 500 independent runs for SFM and 500 for every combination of $T_{horizon}$ for $SMC+SFM$ and $SMC + LIN$. Our objective was to demonstrate that 1) the higher complexity of our approach leads to valuable payoff in terms of performance and 2) it can be implemented online on an embedded device with limited computing power.

a) Algorithm performance analysis: We have devised two scenarios that challenge our algorithm and highlight significant features of its performance. In the first one (namely, *scenario 1*, depicted in Fig. 2(a)) the user moves on a straight line close to a fixed obstacle, while two agents are moving towards him following a straight line as well. In the second one (namely, *scenario 2*, showed in Fig. 2(b)) the user attempts to visit a market stall at the end of the market while some pedestrians (Agent 1-6) block the user's progress by entering the scenario and moving from one market stall to another. The user's global plan is a straight line from the left to the right of the market. Figure 3 depicts the distances over time with respect to the user, respectively, for one particular run of *scenario 2*.

We defined 4 indicators to measure performance: 1) the

[1]http://www.beagleboard.org

time needed for the user to reach the right side of the scenario (T_{exit}), 2) the measured probability of respecting the minimum safety distance to agents (P_{safe}), 3) the average deviation in position from the global plan (ϵ_x) and 4) the average deviation of the orientation of the user with respect to the ideal orientation of a user perfectly following the global plan (ϵ_θ). These indicators are formally defined as follows.

Let $x(t)$ represent the cartesian coordinates of the position of the user after the planning for each time t, $\theta(t)$ represent its orientation with respect to a fixed frame, $\tilde{x}(t)$ the long term plan and $\tilde{\theta}(t)$ the orientation decided according to the long term plan. The integral error of the difference between the corrected plan and the global plan can be defined as $\epsilon_x = E\left\{\sqrt{\frac{1}{T}\int_0^T |x(t)-\tilde{x}(t)|^2\,dt}\right\}$. A similar performance indicator ϵ_θ is defined for the orientation $\theta(t)$.

Indicators ϵ_θ and P_{safe} can be used to quantify the "comfort" of the user. Indeed, frequent changes in the direction reduce the user experience, especially if elderly, and so does the probability of accidents. Table I and Table II reports the performance we obtained for *scenario 1* and *scenario 2*, respectively, using different values for $T_{horizon}$.

Scenario 1 is the most problematic for SFM due to the limitations of this model we discussed in [18]. The SMC-based strategies exhibit a higher P_{safe} and a lower ϵ_θ when $T_{horizon} \geq 6$. $SMC + SFM$, in turn, outperform $SMC + LIN$ on all indicators.

In *scenario 2*, from the safety and comfort point of view of the user, $SMC + SFM$ approach obtains a higher P_{safe} and a lower ϵ_θ with respect to SFM and $SMC + LIN$, when $T_{horizon} \leq 6$. Nonetheless, P_{safe} decreases and ϵ_θ increases when $T_{horizon} > 6$. This is motivated by the fact that the tested temporal logic formula is less likely to be satisfied over a large horizon in a crowded environment. As a consequence, the planning algorithm suggests the user to stop and/or to change direction in order to avoid the unfeasible path, thus raising ϵ_θ.

The SFM strategy exhibits the lower ϵ_x because it tends to keep the user closer to the global plan. However, this reflects negatively on the "comfort" of the user, especially on P_{safe}, because the model doesn't have an explicit notion of *minimum safety distance to agents*. This behaviour is more evident in *scenario 1*.

b) Timings on the Beagleboard xM: We measured the time needed by the Beagleboard xM to execute the scenarios presented in the previous section. We ran 500 simulations each and we timed the execution of every single decision step for the $SMC + SFM$ strategy, that is, the time needed for a single run of Algorithm 1 using the Social Force Model as the model for the agents. We also set $N = 50$. We computed both the average $\mu_1 = 228.9$ ms and $\mu_2 = 2026.1$ ms and standard deviations $\sigma_1 = 392.1$ ms and $\sigma_2 = 2432.1$ ms of the timings for *scenario 1* and *scenario 2*, respectively. If we allow a maximum 1000 ms latency to compute the decision step, the current implementation is able to satisfy it in 93.4% of the cases for *scenario 1* and in 40.9% of the cases for *scenario 2*.

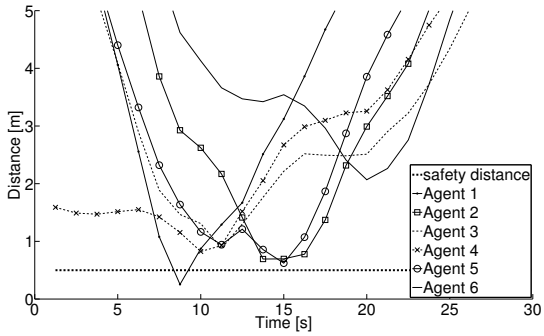

Fig. 3. Scenario 2. Distances of the agents with respect to the user during one particular run of *scenario 2*. In this case the safety distance has been set to 0.5 m (dashed line) and has been violated once.

V. Conclusion and Ongoing Work

The social force model is a generative model of pedestrian behaviour, which we have embedded in an efficient online motion planning algorithm. We parametrise the model with data from sensors in real time and thus hypothesise future trajectories. The model's stochastic element limits its predictive ability but allows us to consider a distribution of possible futures. The reactive part of the algorithm is provided by statistical model checking technology. Thus, the algorithm verifies the hypothesised trajectories against the user's goals and constraints expressed in temporal logic. In this way the algorithm finds the immediate course of action that maximises the user's probability of success.

The apparently random behaviour of pedestrians is often the result of deterministic choices on their part. We hope to to improve the performance of our algorithm by recognising these choices and thus replacing some of the stochasticity. To this end, in [18] we have identified behavioural templates that may be incorporated into the social force model. Also, equation (2) includes the possibility to explicitly model incidental attractive and repulsive forces that might, for example, arise from interesting shops and areas with high probability of crowding, respectively. Such forces apply to pedestrians in general and could enrich the existing model.

As part of our larger project [2], we also propose to include advanced sensor techniques to recognise known interesting or hostile people (e.g., using facial recognition) and to generally avoid people exhibiting hostile behaviour. Such forces apply asymmetrically and would obviously have to be included in an anisotropic version of the social force model [17].

A significant part of the challenge of our motion planning application is the performance of its implementation. Current hardware performance forces us to accept the necessity of multiple boards to handle the overall computational burden, but there is a clear advantage if a portable device can be made to work on a single board. The embedded computing boards we have chosen for our implementation include high performance graphical processor units (GPUs) that can be used for general purpose parallel computing. Since statistical model checking requires multiple independent simulation

TABLE I

SCENARIO 1: PERFORMANCE FOR $SMC + SFM$, $SMC + LIN$ AND SFM STRATEGIES. 500 SIMULATIONS EACH WERE CONDUCTED.

	Unit	$SMC + SFM$					$SMC + LIN$					SFM
$T_{horizon}$	[s]	1	2	4	6	8	1	2	4	6	8	-
T_{exit}	[s]	23.08	23.38	22.72	21.68	21.11	23.17	24.63	24.55	24.42	24.19	23.56
P_{safe}	-	0.7444	0.8923	0.9933	0.9981	0.9985	0.7511	0.8709	0.9565	0.9989	0.9925	0.7386
ϵ_x	[m]	0.3504	0.9914	1.4377	1.6131	1.7386	0.3322	0.9832	1.4761	1.9007	2.0384	0.3062
ϵ_θ	[DEG]	53.19	37.22	13.93	10.84	9.36	55.89	48.03	40.11	24.66	22.47	36.86

TABLE II

SCENARIO 2: PERFORMANCE FOR $SMC + SFM$, $SMC + LIN$ AND SFM STRATEGIES. 500 SIMULATIONS EACH WERE CONDUCTED.

	Unit	$SMC + SFM$					$SMC + LIN$					SFM
$T_{horizon}$	[s]	1	2	4	6	8	1	2	4	6	8	-
T_{exit}	[s]	26.82	23.89	24.08	21.12	20.27	27.33	31.48	36.03	29.98	23.71	23.16
P_{safe}	-	0.9908	0.9998	0.9993	0.9977	0.9316	0.9882	0.9965	0.9977	0.9925	0.9486	0.9665
ϵ_x	[m]	0.7677	0.7927	0.6497	0.5701	0.5430	0.7902	1.4279	1.2607	0.8282	0.6760	0.3825
ϵ_θ	[DEG]	9.97	5.50	9.20	10.59	19.97	10.62	14.25	20.33	21.56	21.52	13.67

runs, we propose to exploit the GPU to gain a significant increase in performance..

$T_{decision}$ is necessarily less than $T_{horizon}$, hence the algorithm predicts traces in time periods that overlap from one iteration to the next. While the predictions of older simulations are likely to be less accurate with respect to the current reality, data from the previous iterations may be employed, suitably weighted, to build a probabilistic map of the good and bad locations in the local environment. This map can be used to avoid simulations that explore directions that are unlikely to be successful and to provide haptic feedback if the user chooses to diverge from the proposed path.

REFERENCES

[1] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, September 2000.

[2] DALi project web site, http://www.ict-dali.eu/dali.

[3] J. Barraquand and J.-C. Latombe, "A monte-carlo algorithm for path planning with many degrees of freedom," in *Proc. IEEE International Conference on Robotics and Automation*, 1990, pp. 1712–1717 vol.3.

[4] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, August 1996.

[5] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State university, Tech. Rep. TR 98-11, October 1998.

[6] S. Loizou and K. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on ltl specifications," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, Dec., pp. 153–158 Vol.1.

[7] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3116–3121.

[8] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *IEEE Conference on Decision and Control (CDC)*, Shanghai, China, December 2009.

[9] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," *Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit*, 2008.

[10] J. Hu, J. Lygeros, and S. Sastry, "Towars a theory of stochastic hybrid systems," in *HSCC*, ser. Lecture Notes in Computer Science, N. A. Lynch and B. H. Krogh, Eds., vol. 1790. Springer, 2000, pp. 160–173.

[11] J. Hu, M. Prandini, and S. Sastry, "Aircraft conflict prediction in the presence of a spatially correlated wind field," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 326–340, 2005.

[12] M. Prandini, J. Lygeros, A. Nilim, and S. Sastry, "A probabilistic framework for aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–220, 2000.

[13] R. Asaula, D. Fontanelli, and L. Palopoli, "Safety provisions for human/robot interactions using stochastic discrete abstractions," in *Proc. of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010)*, October 2010.

[14] ——, "A probabilistic methodology for predicting injuries to human operators in automated production lines," in *Proc. of the 14th IEEE Conference on Emerging Technologies and Factory Automation (ETFA2009)*, Mallorca, Spain, September 2009.

[15] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, 1995.

[16] D. Helbing, I. J. Farkas, and T. Vicsek, "Freezing by heating in a driven mesoscopic system," *Phys. Rev. Lett.*, vol. 84, pp. 1240–1243, 2000.

[17] D. Helbing, I. Farkas, P. Molnár, and T. Vicsek, "Simulation of Pedestrian Crowds in Normal and Evacuation Situations," in *Pedestrian and Evacuation Dynamics*, M. Schreckenberg and S. D. Sharma, Eds. Springer, 2002.

[18] A. Colombo, D. Fontanelli, D. Gandhi, A. De Angeli, L. Palopoli, S. Sedwards, and A. Legay, "Behavioural Templates Improve Robot Motion Planning with Social Force Model in Human Environments," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2013.

[19] E. Clarke, E. A. Emerson, and J. Sifakis, "Model checking: algorithmic verification and debugging," *Commun. ACM*, vol. 52, no. 11, pp. 74–84, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592761.1592781

[20] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, May 2008.

[21] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations," *Ann. Math. Statist.*, vol. 23, no. 4, pp. 493–507, 1952.

[22] PLASMA-lab web site, https://project.inria.fr/plasma-lab/.

[23] S. K. Jha, E. M. Clarke, C. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A bayesian approach to model checking biological systems," in *Computational Methods in Systems Biology*, ser. Lecture Notes in Computer Science, P. Degano and R. Gorrieri, Eds. Springer Berlin Heidelberg, 2009, vol. 5688, pp. 218–234.

[24] K. Ahnert and M. Mulansky, "Odeint – Solving Ordinary Differential Equations in C++ ," *AIP Conference Proceedings*, vol. 1389, no. 1, pp. 1586–1589, 2011.