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

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

***Abstract*— The unpredictable social and physical interactions in 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 to assist such people. 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 same 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 their way through crowded areas with few problems. Under panic conditions [1] or when a pedestrian has reduced ability, finding a good strategy to proceed can be challenging. As a result, people afflicted by a decline in their physical and cognitive abilities (primarily older adults) can be discouraged from attending crowded public 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.

While the behaviour of individual pedestrians may be arbitrary, people nevertheless tend to respect certain social rules that can be formalised. Hence, our solution to the problem outlined above 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 that express 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 global path cannot be defined a priori. Hence, the problem we solve by sampling is not one of creating an optimal global plan (this is given), but one of finding an optimal local plan given a changing environment.

Temporal logic is a formal way to represent complex properties of paths and *model checking* is an automatic process that verifies whether a system satisfies such properties. If the notion of an optimal path is given, model checking can be used to prove that a motion planner will be 'correct' [6], [7].

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

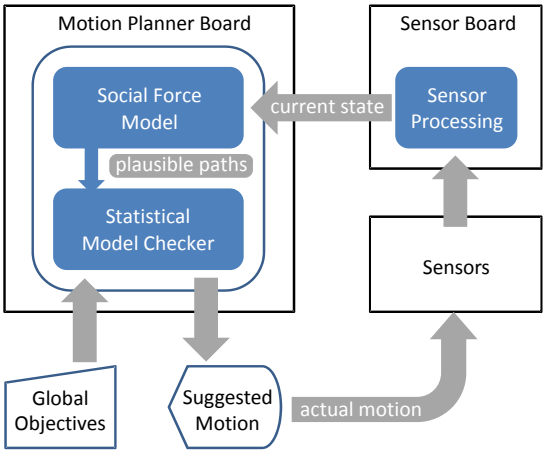

Fig. 1. Diagrammatic overview of the motion planning framework. The sensor board detects the current state of objects in the environment. This state is used by the social force model to generate plausible future paths of the user and other pedestrians. The distribution of paths is verified against the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

The use of probabilistic model checking in combination with the 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, such as 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, however, 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. As a starting point we refer the interested reader to the references already mentioned and to two recent reviews of sampling methods [15], [16].

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 V describes the results of a number of experiments that demonstrate the utility of our approach. Section VI Discusses our choices and highlights areas of ongoing development.

II. Background Material and Solution Overview

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. Despite these features, however, in our application the social force model is not sufficient on its own to adequately avoid "close encounters", nor to account for the overall "mission" of the user. Note that a single simulation trace that includes realistic stochasticity is nevertheless just one of infinitely many possible futures and could be completely incorrect. A trace without stochasticity could be equally incorrect, because the possibility of random deviations is not modelled.

Our algorithm overcomes these limitations by verifying hypothesised stochastic trajectories generated by the social force model with respect to goals and constraints expressed in temporal logic. Such trajectories respect the basic social and physical laws of pedestrian interactions, include the possibility of unpredicted behaviour, while their *distribution* allows the algorithm to choose a course of action that maximises the probability of success. The field of statistical model checking (SMC) encapsulates the technologies that we require to manage our mathematical model in this way, hence we use an SMC library [17] that provides efficient algorithms to estimate the probability of the user's success with guaranteed confidence.

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

As our mathematical model we have chosen the 'social force model' [1], [18]–[20] of pedestrian motion, which combines real and psychological forces to predict the behaviour of crowds in 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.

With minimal practical loss of fidelity [1], [18]–[20], the model is constructed in two dimensions, with agents represented as circular discs. In what follows we adopt the convention of denoting vectors in bold type. Thus, i–th agent

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 liner model for the i-th agent is given by

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

$\mathbf{v}_i^0$ 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 that models fluctuations not accounted for by the deterministic part of the model. As we describe later, in our application the noise term can also serve to avoid deadlocks and hypothesise alternative trajectories. $\mathbf{f}_i$ is the force acting on the i–th agent 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.).

Each element of (2) is available by knowing: the distance d_{ij} between the centres of mass of agents i and j; the "touching distance" $r_{ij} = r_i + r_j$; the direction of the repulsive force $\mathbf{n}_{ij}$; the motion tangential direction $\mathbf{t}_{ij}$; the social force parameters a_i and b_i; the weighting parameter $\lambda \in [0, 1]$ of the social forces. More details can be found in [1].

C. Statistical and probabilistic model checking

Model checking is an automatic technique used to verify that a system satisfies a property. Typically, the system comprises discrete states and transitions, while the property is specified in temporal logics, such as LTL and CTL [21]. Logics are required to be expressive (able to express complex dynamical phenomena), but also decidable and tractable. To give a result with certainty, standard model checking 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, causing an exponential increase of the system state dimension that can often make the process slow or completely intractable.

The output of standard model checking is either *true* or *false*, with corresponding examples or counter-examples of the property. *Probabilistic* model checking extends the standard notion to include probabilistic transitions and rewards, which can express uncertainty and quantitative performance of system [21]. Exhaustive probabilistic model checking (commonly called numerical model checking) suffers similar problems of state dimension explosion problem as standard model checking.

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 a number of independent executions (simulations) of the system. Given N independent simulation traces ω_i and a 'local' model checking function $\mathbb{1}(\omega_i) \in \{0, 1\}$ that indicates whether $\omega_i \models \phi$ (read "ω_i *satisfies* ϕ"), the probability γ that $\omega \models \phi$ holds is estimated using $\tilde{\gamma} = 1/N \sum_{i=1}^{N} \mathbb{1}(\omega_i)$. The confidence of the estimate can be guaranteed by standard statistical bounds (such as the Chernoff bound [22]), allowing SMC to trade certainty for reduced confidence plus tractability. In comparison to exhaustive model checking techniques, SMC does not require a finite state space, making it particularly suitable for the present application that considers continuous time and space.

Probabilistic Bounded Linear Temporal logic: We use temporal logic to express and combine abstract 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". Our model checking engine is based on the logic of PLASMA [23] and, in particular, the PLASMA-lab library [17]. PLASMA-lab accepts nested linear temporal formulae ϕ using the following abstract syntax:

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

When applied to trace ω_i, ϕ evaluates to logical *true* or *false*, such that $\mathbb{1}(\omega_i \models \phi) \in \{0, 1\}$. $\vee, \wedge$ and $\neg$ are the standard logical connectives and α is a Boolean constant or an atomic proposition constructed from numeric expressions of constants and state variables. 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 t, that is relative to the temporal constraints of any enclosing formula. F is the *finally* or *eventually* operator: $\mathrm{F}_{\leq t}\phi$ means that ϕ will be true at least once in the relative time 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 temporal operators allows the definition of complex properties with interleaved notions of *eventually* (F), *always* (G) and *one thing after another* (U). A precise classification of the expressivity of our logic is beyond the scope of the present work, however it can likened to a bounded time PCTL [21]. The semantics is essentially the same as that of PBLTL in [24].

Algorithm 1 The planning algorithm

1: **function** FINDLOCALPATH(state$_{user}$, state$_{ped_1}, \ldots,$ state$_{ped_N}$, Map, GlobalPlan, Formula)
2: **Real** P_{curr}, d_{curr}, P_{best}, d_{best};
3: [P_{best}, d_{best}]=[1,∞];
4: **for** $\alpha_{curr} \in \{0, \pm 20, \pm 50, \pm 75, \pm 90\}$ **do**
5: [P_{curr}, d_{curr}] = SMC(state$_{user}$, state$_{ped_1}, \ldots,$ state$_{ped_N}$, Map, GlobalPlan,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** $P_{best} == 1$ **then**
12: **return STOP**;
13: **else**
14: **return** α_{best};
15: **end if**
16: **end function**

III. SMC–BASED MOTION PLANNER

Our algorithm assumes the existence of a pre-computed long-term trajectory (the *global plan*) that visits the user's objectives in an optimal way, starting from the user's initial position. In practice, this trajectory is computed based on the information about the environment, anomalies (e.g., crowded areas), static objects and the user's objectives. A change in the user's objectives triggers the recalculation of this global plan.

The sensor subsystem integrated in the device, periodically provides the state of the local environment: 1) position and velocity of the user (localization), 2) position and velocity of the pedestrian (user sensing), and 3) position of non static obstacles and anomalies (environment sensing).

The algorithm then uses this knowledge to construct a system of ordinary differential equations according to (Eq. 1). For every pedestrian we only have information about position and velocity, we thus decided to use the parameters of the model given in [1]. A future challenge will be to estimate these parameters on-line. The noise term $\boldsymbol{\xi}_i$ is used to model the natural variance seen in real pedestrian behavior and to resolve deadlocks (agents unable to pass each other due to near equal and opposite velocities). In the current implementation we estimate this variance with two normal distributions; one for intensity and one for direction. However, one issue to be addressed is an accurate estimate of these distributions when pedestrains moving in a mall (or similar) environment are considered.

In order to compute the local plan, the algorithm estimates the evolution of the system state (comprising the position of the user and of other pedestrians) for $T_{horizon}$ time units in the future starting from an initial state that is estimated by the sensing system.

The pseudo-code of the algorithm is reported in Agorithm 1. The input parameters of the algorithm are positions and velocity of the user (state$_{user}$) and of the other pedestrian in the scene (state$_{ped_1}, \ldots,$ state$_{ped_N}$), the map of the environment annotated with static obstacle (Map), the global plan (GlobalPlan) and the logic formula expressing the safety constraints we want to impose (Formula). The direction to take is denoted as α_{best}, where α_{best} take value in the finite set of possible directions ($\{0, \pm 20, \pm 50, \pm 75, \pm 90\}$). The algorithm cycles over all possible directions (`for` cycle between line 4 and 10 in Agorithm 1). Each possible assignment of directions is model checked against the logical property (`SMC` function call). In the experimental data collection reported below, we have used the following formula expressed in temporal logic: $\left(\mathrm{G}_{[0,T_{horizon}]} \bigwedge_{i \neq u} \|\mathbf{x}_u - \mathbf{x}_i\| > 0.5\right) \wedge \left(\mathrm{F}_{[0,4]} \|\mathbf{x}_u - \mathbf{w}\| < 0.2\right)$, where $\mathbf{x}_u$ denotes the position of the user, $\mathbf{w}$ is the current local waypoint and $\|\cdot\|$ denotes the Euclidean distance. This property can be translated in plain english as *in the next $T_{horizon}$ time units the user will get no closer than 0.5 m to any other pedestrian and will eventually be less than 0.2 m from the global plan*. The local waypoint $\mathbf{w}$ is a point on the global plan that serves as attractor for the user. This point is identified as the point that an "ideal" user, moving with the reference velocity would reach at the end of the time horizon if he/she followed exactly the global plan. To estimate the probability of violation (P_{curr}) for a given decision α_{curr}, the SMC algorithm generates N independent simulation runs of the model (where N is a parameter) from the current time (T_0) to the end of the time horizon ($T_0 + T_{horizon}$). The decision is implemented by rotating the desired velocity vector of the user v_i^0 of the quantity α_{curr}. The system evolves "freely" using the social force model up to the end of the time horizon. Each simulation produces a different trajectory due to the influence of the random parameters. The probability of violation of the property is estimated by counting the number of executed trajectories that violate it. In addition to P_{curr} the SMC also returns the expected value of the distance from the global plan achieved through the set of simulation runs, d_{curr}.

The decision α_{curr} that attains the best trade-off between safety and distance from the plan is noted during the execution of the `for` loop and returned at the end of the simulation (α_{best}). This trade-off is coded in a function named is_better, which considers a trajectory only if the maximum probability of violation P_{best} is below a given threshold. If none of the choices attains this threshold, the user is required to stop.

After the algorithm is terminated, the decision α_{best} is suggested to the user. In this report we have restricted our focus to "compliant" user: the decision is accepted and executed. Different possibilities in the behavior of the user will be considered in our future work. The planning algorithm can be re-executed after the subsequent decision point is reached implementing a receding horizon paradigm. In this work we assume decision points are spaced $T_{decision}$ time units apart from each other, where $T_{decision} < T_{horizon}$.

Assuming that $\boldsymbol{\xi}_i$ is an accurate model of the random behaviour of pedestrians, the Chernoff bound [22] predicts that with $N = 10$ simulation runs the estimate of the probability of success has maximum error of ± 0.3 with 70% confidence. The confidence increases and the maximum error decreases as N increases, for example, with $N = 50$ the

probability of success has a maximum error of ± 0.2 with 90% confidence. The algorithm improves on these figures by effectively re-estimating the probability as the user moves along the path – the algorithm updates in a time that is approximately an order of magnitude less than the time of the simulated path.

To generate simulation runs, the algorithm solves (1) using a standard ODE solver [25]. The solver uses an adaptive step size to increase efficiency, minimise errors and to ensure that important points (e.g., hidden peaks) are not lost due to sampling.

IV. Simulation model

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*[2] as the statistical model checking library. To test the algorithm we have created a virtual environment containing 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. These informations are unknown to the planner in order to increase the sense of reality. We thus used a different set of parameters for the Social Force Model and the noise term ξ in the planner. 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.

V. Simulations

In order to show the benefits of using SMC and the Social Force Model together ($SMC + SFM$), we compared our proposed solution with two different strategies:

- SMC with a linear motion model ($SMC+LIN$): when detected, an agent is suposed to keep moving with same speed and same direction. We added a normally distributed noise (equal to ξ_i in $SMC+SFM$) in order to randomize simulations.
- 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 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*) the user moves on a straight line close to a fixed obstacle, while three agents are moving towards him. In the second one (namely, *scenario 2*) the user attempts to visit a market stall at the end of the market while some pedestrians 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. Figures 2 and 3 depict the trajectories on the plane and the distances over time with respect to the user, respectively, for one particular run of *scenario 2*. We defined 4 parameters to measure performance: 1) the 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) ϵ_x and 4) ϵ_θ. The last two are defined below.

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 w.r.t. a fixed frame, $\tilde{x}(t)$ the long term plan and $\tilde{\theta}(t)$ the orientation decided according to the long term plan. One possible way for quantifying the performance of the planner is by the integral error of the difference between the corrected plan and the long term plan: $\epsilon_x = E\left\{\sqrt{\frac{1}{T}\int_0^T |x(t) - \tilde{x}(t)|^2 \, dt}\right\}$ A similar performance metric can be defined for the orientation: $\epsilon_\theta = E\left\{\sqrt{\frac{1}{T}\int_0^T \left|\theta(t) - \tilde{\theta}(t)\right|^2 dt}\right\}$. This second error term along with 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 results we obtained for *scenario 1* and *scenario 2*, respectively.

Scenario 1 is the most problematic for SFM due to the limitations of this model discussed previously. This is reflected in the results, as $SMC+SFM$ and $SMC+LIN$ exhibit a higher P_{safe} and a lower ϵ_θ, especially when $T_{horizon}$ is large. $SMC + SFM$, in turn, outperforms $SMC + LIN$ on all parameters. In *scenario 2*, from the safety and comfort point of view of the user, $SMC+SFM$ approach obtain a higher P_{safe} and a lower ϵ_θ with respect to SFM and $SMC + LIN$, when $T_{horizon} \leq 6$. We observed a decrease of performance of the SMC-based approaches when $T_{horizon} > 6$. This is motivate 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 (higher ϵ_θ). In general, SFM tends to keep the user closer to the global plan, even though it is not able to provide the same comfort level as $SMC + SFM$.

b) Temporal Performance on the Beagleboard: We measured the CPU time needed by the Beagleboard xM to

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

[2] https://project.inria.fr/plasma-lab/

TABLE I

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

Parameter	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 MODELS. 500 SIMULATIONS EACH WERE CONDUCTED.

Parameter	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

TABLE III

PERFORMANCE (CPU TIME/SIMULATION STEP) ON THE BEAGLEBOARD, FOR $SMC + SFM$ ($T_{decision} = 1$, $T_{horizon} = 4$) AND SFM. 500 SIMULATIONS EACH WERE CONDUCTED.

Scenario	$SFM + SMC$ [ms]		SFM [ms]	
	μ	σ	μ	σ
Scenario 1	228.9	392.1	1.1	1.2
Scenario 2	2026.1	2432.1	10.1	11.2

execute the scenarios presented in the previous section. We executed 500 simulations of the two scenarios for $SMC + SFM$ and SFM. We timed the execution of every single simulation step using the standard POSIX libraries available on UNIX systems and finally we computed the average time μ and the standard deviation σ reported in Table III. We observed a large standard deviation, related to the chosen type of ODE solver that uses an adaptive step size. However, if $T_{horizon} = 1$ seconds, the current implementation is able to successfully (i.e. within the deadline of 1000 ms) compute a step of the proposed algorithm in 93.4% of the cases for the simple scenario and in 40.9% of the cases for the market scenario.

VI. CONCLUSION AND ONGOING WORK

The social force model is essentially a deterministic model of the *average* behaviour of individual pedestrians in crowded situations. The trajectory of a real pedestrian will often include apparently random deviations, arising from unexpected attractions and distractions along the way. In our current implementation these deviations are modelled by the noise term $\boldsymbol{\xi}_i$ in (1). In this way, our estimate of the probability of success can include the potentially highly non-linear consequences of pedestrians deviating from their nominal trajectories. If $\boldsymbol{\xi}_i$ is an accurate model of the non-deterministic behaviour of other pedestrians, our estimate of the probability of success of the user's path will also be accurate and *on average* we can be reasonably confident that our prediction is correct. It is important to note, however, that in any concrete situation the trajectories of real pedestrians may vary significantly from our prediction, as a result of the variance of $\boldsymbol{\xi}_i$. A goal of our ongoing research is therefore to predict deviations from the average trajectory more deterministically.

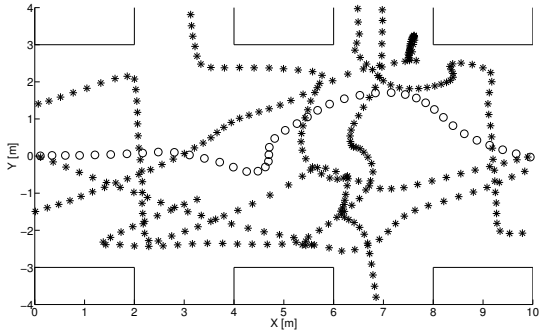

Fig. 2. Scenario 2. The user (circle-shaped) enters the scenario from the left-hand side and his goal is to exit on the right-hand side, following a straight line. In the meanwhile, some agents move around the area following their respective goals.

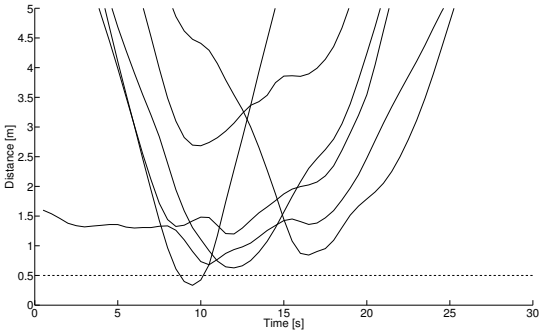

Fig. 3. Scenario 2. Distances of the agents with respect to the user during the run showed in Fig. 2. The safety distance is set to 0.5 m (dashed line) and has been violated once in this particular run.

One means by which we propose to achieve greater determinism in our model is to enrich the information about the user and other pedestrians. 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 would apply to pedestrians in general and would be known in advance.

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 [20].

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 computing. Since statistical model checking lends itself to parallelisation, requiring multiple independent simulation runs, we propose to exploit the GPU to gain a significant increase in performance (potentially several orders of magnitude). A further gain in performance might be obtained by code optimisation, but to simplify ongoing development we propose to leave this until the overall design has stabilised.

In the current implementation $T_{decision} < T_{horizon}$, hence, the algorithm generates a number of trajectories that effectively overlap by a relevant amount of time units the trajectories it generated on the previous iteration. Since each simulation trace is an independent realisation of a random variable, the new and old trajectories are unlikely to coincide exactly. Moreover, the predictions of older simulations are likely to be less accurate with respect to the current reality. Despite this, data from the three 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 very likely to be unsuccessful and to provide haptic feedback if the user chooses to diverge from the proposed path.

Our immediate goal is to validate our approach with more complex simulations and under real conditions.

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, Aug.

[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] S. Lindemann and S. LaValle, "Current issues in sampling-based motion planning," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Springer Berlin Heidelberg, 2005, vol. 15, pp. 36–54.

[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

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

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

[19] 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.

[20] 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.

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

[22] 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.

[23] C. Jegourel, A. Legay, and S. Sedwards, "A Platform for High Performance Statistical Model Checking – PLASMA," in *Proceedings of the 18th international conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 498–503.

[24] 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.

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