Journal of Intelligent & Robotic Systems (2021) 101:36
https://doi.org/10.1007/s10846-020-01308-8

Crowded Environment Navigation with NEAT: Impact of Perception Resolution on Controller Optimization

Stefano Seriani[1] · Luca Marcini[1] · Matteo Caruso[1] · Paolo Gallina[1] · Eric Medvet[1]

Received: 30 April 2020 / Accepted: 30 December 2020

Abstract

Crowd navigation with autonomous systems is a topic which has seen a rapid increase in interest recently. While it appears natural to humans, being able to reach a target can prove difficult or impossible to a mobile robot because of the safety issues related to collisions with people. In this work we propose an approach to control a robot in a crowded environment; the method employs an Artificial Neural Network (ANN) that is trained with the NeuroEvolution of Augmented Topologies (NEAT) method. Models for the kinematics, perception, and cognition of the robot are presented. In particular, perception is based on a raycasting model which is tailored on the ANN. An in-depth analysis of a number of parameters of the environment and the robot is performed and a comparative analysis is presented; finally, results of the performance of the controller trained with NEAT are compared to those of a human driver who takes over the controller itself. Results show that the intelligent controller is able to perform on par with the human, within the simulated environment.

Keywords Artificial neural networks · Evolutionary robotics · NEAT · Crowd navigation · Robot controller

1 Introduction and Related Work

In recent years, the concept of autonomous delivery has begun to take hold, both in indoor environments [1–3] as outdoors, i.e., in urban areas [4–7], following the great investments in the field of autonomous driving [8]. One of the most common problems which is found in both fields is that of collision avoidance against pedestrians [5]. In particular, in an urban environment, automated delivery needs to solve the problem of navigating places which might be very crowded. On top of this, while traffic is regulated by precise laws and rules that contribute to create a certain structure in the behavior that an autonomous agent must have [9], in crowds, navigation is less structured and more related to courtesy and social norms. Humans navigate crowds using a set of skills that are acquired over time and are studied in the field of proxemics [10, 11]; furthermore, occasional collisions are not cause for concern. Conversely, autonomous robots do not have this prerogative and must in no case hurt or collide with a person [12]. Therefore, where dense crowds are concerned, it could happen that the robot becomes completely unable to move in order to avoid potential collisions; this is called *freezing robot problem* [13].

Trautman et al. [13] proposed an early statistical model based on dependent output Gaussian processes that estimates agents trajectories in a crowd and cooperatively generates feasible trajectories. Using data from surveillance cameras in public spaces, Luber et al. [14] approach the issue as an unsupervised learning problem, as opposed to proxemics-based methods. Building on previous studies on human navigation in crowds, Guzzi et al. [11] develop a fully-distributed algorithm for local navigation in densely populated spaces. Trautman et al. [15–17] explore the concept of navigation through cooperation between the robot and the human agent and conclude that cooperation is critical for dense human crowd navigation. This concept is expanded by May et al. [18] and then by Shresta et al. [19, 20] with the notion of intent signaling, where the robot actively employs visual cues to communicate with human agents. A review on the main social-aware frameworks for navigation is presented [21]. Narayanan et al. presented an adaptive strategy for the approach phase in human-robot interaction [22]. A human intent-aware strategy for safe and efficient robotic crowd navigation were reported by Park et al. [23]. A Low-cost approach is shown by Chatterjee

✉ Stefano Seriani
sseriani@units.it

[1] Department of Engineering and Architecture, University of Trieste, Trieste, Italy

Published online: 01 February 2021

that employs a vision system based on Microsoft's Kinect platform [24]. A considerably more complex approach is presented by Bohorquez, that considers a bipedal robot [25]. Beomjoon et al. [26] present an inverse reinforcement learning method used as an advance technique for crowd navigation. A similar method is shown by Anirudh et al. in [27]; a real-time based on Bayesian Learning and Proxemics called *SocioSense* is presented in the same year by Bera et al. [28].

More recently, Patle et al. [29] show an approach based on fuzzy logic with a specially tuned algorithm for robot navigation. For the same task, Alaraji et al. present a brief comparative study between some optimization-based algorithms [30]. Later, Das et al. in [31], present an application of thee Gravitational Search Algorithm (GSA) to the task of navigating cluttered unknown environments. With a more experimental approach Chen et al. [32] show a methodology for crowd navigation based on a graph convolutional network which makes use of the gaze information of the individuals in the crowd.

It is clear from the presented works that a collaborative approach between the robot and humans is critical for safe and efficient navigation of crowds; furthermore, intent conveying and proxemics-aware robots that can deal with social cues, both as receivers and transmitters, are of prime importance [12, 14, 16, 18, 21–23].

In this work we will investigate the interaction of a robot with a dense moving crowd without intent-signalling, in order to define a baseline for future more advanced work. The case-study environment is a corridor where the general direction the people move is against the planned motion of the robot. Our delivery robot will be a small hypothetical autonomous vehicle equipped with a 2D laser scanner. The cognition model of the robot is based on an ANN, which can be used to control a robot to navigate a moving crowd efficiently and safely [33–35]. It has been shown by Dudarenko et al. that data from a laser scanner can be used for navigation using a stochastic approach [36].

Furthermore, it has been shown that the process of differential evolution applied to neural networks, delivers good results in a variety of reinforcement learning problems [37–40].

In our case, we have implemented the automatic design of the ANN through the NeuroEvolution of Augmented Topologies (NEAT) methodology presented by Stanley et al. [41]. This approach proved useful in solving reinforcement learning problems. In a standard Neuroevolution algorithm only the weights of the neural net can be optimized while the topology of the net itself is fixed during the process [38]. NEAT was devised to overcome this limitation and is capable of optimizing both the weight and the topology at the same time. The evolution of the ANN is implemented through a specially tuned genetic algorithm, similarly to other works, e.g., from Buk et al. [42] and Caceres et al. [43] especially, where the implementation allowed the navigation of a structured environment with static obstacles.

Perception in the simulated environment is made to closely mimick a 2D laser scanner (LiDAR) and does not involve explicit mapping. It is based on a raycasting model to perceive range and is able to perceive both at current time, and in the past; we call this Past Range Perception (PRP).

In order to characterize the implemented approach, we elected to explore part of the parameters-space of the methodology, namely the number of rays in the raycasting model, the PRP delay parameter and the speed and number of individuals in the crowd.

Some of these parameters, especially the number of rays, have a large impact on the complexity of the search for an optimal solution; while they provide increased expressive capability of the perception model, this comes at the cost of an increased search space. However, the NEAT algorithm is very appropriate to address this problem due to the fact that it implements the *complexification* paradigm [42]: it starts off with small and simple ANNs that are grown when necessary. That is, the space of ANNs in which NEAT performs the search is initially small and grows during the optimization process.

To summarize, we implement a methodology based on the NEAT algorithm to the problem of crowd navigation via mobile robot, with perception based around a raycasting model. We carry out an in-depth characterization on the main parameters of the models, highlighting their influence on the performance of the system. We compare numerical results of the ANN-based system with a human driver acting in the stead of the ANN. Finally, we implement a physical perception methodology using LiDAR acquisitions of a structured environment and compare these with perception in the simulated environment.

The main contributions of this work with respect to the state-of-the-art presented up to this point are:

- the development of a methodology to correctly state the problem of crowd-navigation in order for it to be approachable by ANN-based navigation such as NEAT,
- the characterization of the influence of perception resolution in a population-based optimization algorithm for crowd navigation,
- to analyze of the performance of NEAT against the variability of the environment.

In Section 2 the problem statement is given along with the description of the cognition and perception models; in Section 3 the in-depth exploration of the parameters space is performed; in Section 4 the experimental results are shown for a set of different conditions including an experimental validation of the crowd perception via LiDAR;

finally, in Section 5 we conclude giving some insight in the methodology limits and future work.

2 Methodology

The description of a crowd-navigating robot use-case must include at least 3 objects: a model of the environment, a model of the crowd, and a functional model (i.e., including the control system) of the robot. In this section, the use-case will be broken down in these three main topics in order to convey a complete description of the problem.

2.1 The Environment

The environment which is described in this section was chosen carefully with the following perspective in mind:

- The environment should represent a crowd in a repeatable and parametric way, i.e., the characteristics of the crowd should be connected to few very influential parameters,
- The environment boundaries should be as simple as possible while at the same time providing some effect to the robot; for example, a no-wall environment would not provide any parameterizable effect on the robot,
- The environment should provide a baseline characterization of the problem, useful as a starting point for further developments and refinements.

These guidelines are important to keep the number of degrees of freedom of the simulated environment as low as possible, while at the same time keeping a certain degree of adherence to reality.

With these considerations in mind, as mentioned in Section 1, the environment chosen for this use-case is that of a rectangular corridor long L and wide W. The corridor, illustrated in Fig. 1, is defined by 4 *fixed walls*. During the simulation a virtual *mobile wall* is implemented to force the mobile robot to move forward in the direction of the *target* location.

The robots or agents are spawned at point $\mathbf{G}$, which is located at the far left end of the domain, as visible in Fig. 1.

The target location $\mathbf{T}$ is randomly assigned at the beginning of a simulation: it is always in the right end of the domain but not too close to the wall so that the agent can detect incoming obstacles and react accordingly. Figure 1 shows the position of the target spawn area within the environment. The target is spawned according to the following formal rule,

$$\begin{aligned}\mathbf{T} = \{\ (x, y) :\quad & x \in \left[\mathbf{T}_{c,x} - T_L,\ \mathbf{T}_{c,x} + T_L\right], \\ & y \in \left[\mathbf{T}_{c,y} - T_W,\ \mathbf{T}_{c,y} + T_W\right]\ \}\end{aligned} \tag{1}$$

where T_L and T_W are the length (along x) and width (along y) of the spawn area. Hence, $\mathbf{T}_{c,x} = L - d_w - T_L/2$ and $\mathbf{T}_{c,y} = W - d_w - T_W/2$ are the coordinates of its barycentre, with d_w the gap between walls and the spawn area.

This approach was chosen because it promises to reduce considerably the possibility of *overfitting*. Indeed, using adaptive solutions based on evolution or optimization generally increases the risk that the resulting model becomes tailored to a specific case, rather than to a general case [44, 45]. Changing frequently the location of the target prevents the agent to find workarounds to reach the target.

2.2 The Crowd Model

Within the corridor environment boundaries, we can define a set of points $\mathbf{Q}_j$ with $j = 1, ..., m$ that represent m individuals in the crowd, i.e., persons. Based on the concept of proxemics [10], each individual is modeled as a disk of radius R that represents its personal space, rather than its physical body geometry.

The velocity at which every person walks depends from a variety of factors such as genre, age and height. To express this variability, each j-th element in the simulation is assigned with a constant average speed $\bar{v}$ to which a random fluctuation u is added or subtracted, as follows,

$$v = \bar{v} + u \quad \text{with} \quad u \in \left[-\frac{\bar{v}}{4},\ \frac{\bar{v}}{4}\right]. \tag{2}$$

Each individuals moves in a straight line, parallel to the top wall, towards the left wall, at the assigned speed v and starting from a random position $(\mathbf{x}, \mathbf{y})$. During the simulation, once the individual crosses the left wall, it appears again at the right end of the domain at a new random y coordinate. The combination of this two random factors contribute in avoiding the risk of repetitions. Since personal space is considered as opposed to the physical body, collisions between individuals are ignored. Each overlap can be interpreted as two or more people moving together.

2.3 Model of the Robot

The model is based on a differential drive robot. The robot geometry is defined as a disk of radius R_R centered in point $\mathbf{P}$, with the local frame of reference $\langle \mathbf{x}', \mathbf{y}' \rangle$ aligned with the $\mathbf{x}'$ axis in the forward direction. The kinematics of this kind of robot is simple: referring to Fig. 2, the non-holonomic constraint is $\dot{\mathbf{y}}' = 0$, which allows the robot to drive along the $\mathbf{x}'$-axis, rotate in place and a combination of these two motions; in other words, the robot is unable to move sideways.

We define ϑ as the orientation of the robot, i.e., the angle between $\hat{\mathbf{x}}'$ and $\hat{\mathbf{x}}$. Let $\xi = [\mathbf{P}_x, \mathbf{P}_y, \vartheta]^T$ be the pose of the

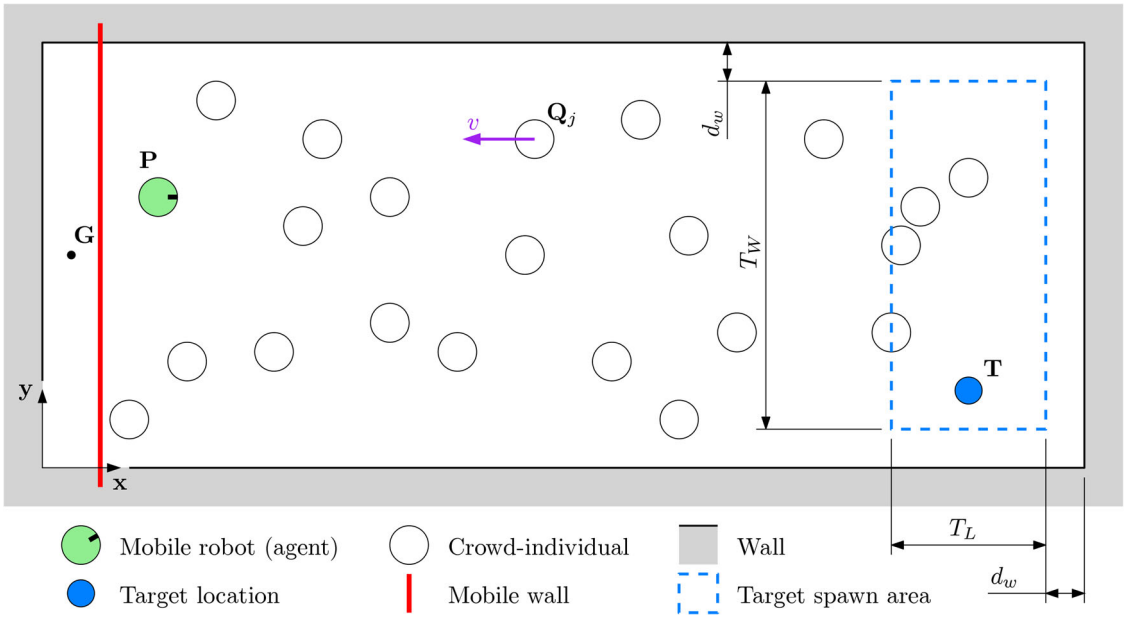

Fig. 1 Model of the environment, showing the robot on the left, in green, and the target point on the right, in blue. The white circles indicate the individuals that constitute the crowd. Walls are shown enclosing the environment, and the mobile wall is indicated in red

robot with respect to the inertial frame of reference $\langle \mathbf{x}, \mathbf{y} \rangle$; if we indicate with $\dot{\varphi}_1$ and $\dot{\varphi}_2$ respectively the right and left wheel rotational speed, we can write the model as follows,

$$\dot{\xi} = \begin{Bmatrix} \dot{\mathbf{P}}_x \\ \dot{\mathbf{P}}_y \\ \dot{\vartheta} \end{Bmatrix} = \mathbf{M}^{-1}(\vartheta)\rho \begin{bmatrix} \frac{\dot{\varphi}_1}{2} + \frac{\dot{\varphi}_2}{2} \\ 0 \\ \frac{\dot{\varphi}_1}{d} - \frac{\dot{\varphi}_2}{d} \end{bmatrix}, \tag{3}$$

where $\mathbf{M}$ is a 2D homogeneous rotation matrix,

$$\mathbf{M} = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

and ρ is the radius of the wheels; finally d is the offset of each wheel along the $\mathbf{y}'$ direction with respect to the $\mathbf{x}'$ axis.

In order to control the robot, we can express the velocities $\dot{\varphi}_1$ and $\dot{\varphi}_2$ in terms of the controlled variables $\dot{\mathbf{P}}'_x$ and ϑ, as follows,

$$\dot{\varphi}_1 = \frac{1}{\rho}\left(\dot{\mathbf{P}}'_x + \frac{d\vartheta}{2}\right), \quad \dot{\varphi}_2 = \frac{1}{\rho}\left(\dot{\mathbf{P}}'_x - \frac{d\vartheta}{2}\right). \tag{5}$$

2.4 Perception

Perception in the robot is implemented through a ray casting algorithm that closely mimics the discrete nature of the physical LiDAR present in many industrial robots. In Fig. 2 the model is described; a fixed number n of $i = 1, ..., n$ rays are cast uniformly outwards from the geometric center $\mathbf{P}$ of the robot. Each ray is defined by a unit vector $\hat{\mathbf{r}}_i$ and by the parameter R_{range}, which is used to define the ray-casting sensing range. A fixed number η of interconnected walls limit the environment; each wall is indexed with $h = 1, ..., \eta$.

A LiDAR-based system perceives the environment through range-finding in a straight line around the agent; a ray casting algorithm is a very similar approach achievable in a 2D virtual environment. In fact the agent does not work with the position of the obstacle, but with its (discretized) frontier created by the ray casting. In other words, the agent is not aware of the position $\mathbf{Q}_j$ of the individuals in the crowd, rather it is only aware of the points determined by the collision of each ray with the individuals' bodies. This methodology contributes in decreasing the *reality gap* when it comes to the experimental phase.

To express the position of an obstacle in a 2D environment, two coordinates are needed (r_i, α_i in a polar system); we encode the LiDAR information as a fixed number of n uniformly distributed rays with angle defined by α_i, and for which the range value r_i is determined by the ray casting collision algorithm.

Additionally, the r_i, α_i information is retained for a certain number of steps in order to implement memory in the controller; this process is called Past Range Perception (PRP) and the distance of the lookback step is represented by the parameter s_{lb}.

The intersections with the walls are performed following the approach outlined by Antonio [46] in 1992. In this regard, and taking into consideration Fig. 2e, we consider a general ray r_i and a general h-th wall $\overline{W_h W_{h+1}}$. The ray starts in point P and ends in point $P + r_i$, while the wall starts in point W_h and ends in W_{h+1}. The intersections are labeled as $\mathbf{S}_i$.

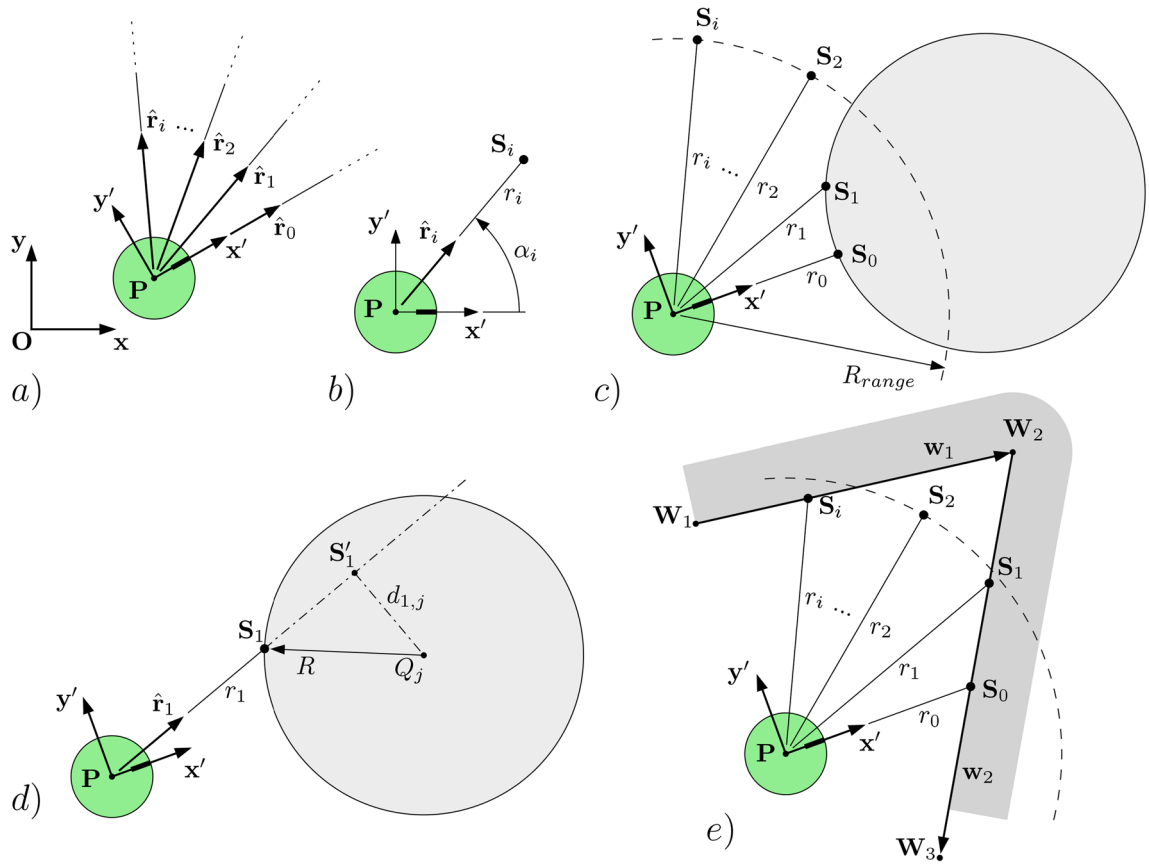

Fig. 2 Perception model through *ray casting*. In **a**) the robot is shown in point **P** in the $\langle \mathbf{x}, \mathbf{y} \rangle$ frame of reference, with the rays $\hat{\mathbf{r}}_0$, ..., $\hat{\mathbf{r}}_i$; in **b**) the main variables of one ray are shown; in **c**) an individual is shown as a grey disk, and the impact points $\mathbf{S}_0$ and $\mathbf{S}_1$ of the rays are shown on the boundary of the obstacle; the dashed line shows the range limit of the rays; in **d**), the geometry of the ray-casting detection is shown for person $\mathbf{S}_i$

While the wall-detecting algorithm is fast, the process of detecting individuals can be considerably slower. In order to keep the computational complexity under control a bounding-box approach is applied to the process. First, a square bounding box is defined around the robot **P** of width $b = 2(R_{\text{range}} + R)$. A subset Ξ_{bb} of individuals is then determined based on the following equation:

$$\Xi_{\text{bb}} = \left\{ \mathbf{Q}_j : \mathbf{Q}_{j,x} \in \left[\mathbf{P}_x - \frac{b}{2}, \mathbf{P}_x + \frac{b}{2} \right] \wedge \mathbf{Q}_{j,y} \in \left[\mathbf{P}_y - \frac{b}{2}, \mathbf{P}_y + \frac{b}{2} \right] \right\} \tag{6}$$

The set Ξ_{bb} is made by all points $\mathbf{Q}_j$ which are contained in the bounding box. If we indicate with k the index of an element $\mathbf{Q}_j$ within the bounding box, i.e., $\mathbf{Q}_k \in \Xi_{\text{bb}}$, then for each of these individuals the following distance can be calculated,

$$d_{i,k} = (\mathbf{Q}_k - \mathbf{P}) \bullet \left(\mathbf{M} \left(\frac{\pi}{2} \right) \hat{\mathbf{r}}_1 \right) \tag{7}$$

It follows that, when $d_{i,k} \leq R$ (*collision condition*), the i-th ray touches the boundary of the k-th individual and point $\mathbf{S}_i$ can be defined on the same boundary. In case multiple instances of k-th individuals for which the collision condition is true, the one with the smallest distance $\|\mathbf{Q}_k - \mathbf{P}\|$ is selected. In order to differentiate points which are within range R_{range}, the explicit calculation of the euclidean distance $\|\mathbf{S}_i - \mathbf{P}\|$ is necessary at this point,

$$\|\mathbf{S}_i - \mathbf{P}\| = -\hat{\mathbf{r}}_i \sqrt{R^2 - d_{i,j}^2} \tag{8}$$

It is worth noting that all the points $\mathbf{S}_i$ defined up to this point are both related to collisions with individuals, with walls, and with the perception range boundary. Some rays $\hat{\mathbf{r}}_i$ may not touch any k-individual nor any walls, especially considering the finite nature of range R_{range}. This being considered, we can define the perceived distance r_i as follows,

$$r_i = \begin{cases} \|\mathbf{S}_i - \mathbf{P}\| & \text{if} \quad \|\mathbf{S}_i - \mathbf{P}\| \leq R_{\text{range}} \\ R_{\text{range}} & \text{if} \quad \|\mathbf{S}_i - \mathbf{P}\| > R_{\text{range}} \end{cases} \tag{9}$$

Collisions between individuals in the environment or the walls and the robot are determined by evaluating the distance between these elements. Specifically, in the case of

a k-th individual, a collision occurs when the following is true,

$$\|\mathbf{Q}_k - \mathbf{P}\| \leq R_R + R \tag{10}$$

In case of a general h-th wall, a collision occurs when the following condition is true,

$$(\mathbf{P} - \mathbf{W}_h) \bullet \left(\mathbf{M}\left(\frac{\pi}{2}\right)\hat{\mathbf{w}}_h\right) \leq R_R \tag{11}$$

with $\mathbf{T}$ the usual rotation matrix.

2.5 Cognition Model: Artificial Neural Network

The cognition model used to implement the control of the agent is based on an Artificial Neural Network. This choice is part of the methodology which involves the use of NEAT. In order to apply the cognition model to the problem at hand, we implement it in a discrete time simulation with a fixed time-step t.

The inputs to the ANN are summarized as follows:

- Current (timestep t) robot position $(\mathbf{P}_x(t), \mathbf{P}_y(t))$,
- Position of the robot at the lookback step $(\mathbf{P}_x(t - s_{\text{lb}}), \mathbf{P}_y(t - s_{\text{lb}}))$,
- Speed of the robot at the lookback step $\dot{\mathbf{P}}(t - s_{\text{lb}})$,
- Length of rays at the current step $r_i(t)$,
- Length of rays at the lookback step $r_i(t - s_{\text{lb}})$,
- Position of the target point T.

The total number of input nodes in the ANN will thus be $\text{ANN}_{\text{in}} = 2 + 2 + 2 + n + n + 2 = 2n + 8$, where n is the number of perception rays.

High-resolution perception of the environment necessarily requires a large number of rays. Due to the fact that the number of input neurons in the ANN is linked to the number of rays in the perception model, the higher this number gets, the higher the complexity of the ANN becomes; in turn, a functional ANN should have a high expressivity to make meaningful use of the inputs. The main problem is that the higher the expressivity of the network, the higher the search space becomes for the optimization of the net. Given these considerations, it is clear that the number of rays impacts directly and heavily on the complexity, expressivity and ultimately the ease-of-optimization of the ANN; in the following sections, this aspect will be evaluated in depth.

Since an input neuron can only process one information, two of them would have to collaborate to establish the position of an object, i.e., in a polar description, one for the angle, one for the distance. By making the angle α_i a function of the line of sight number, we actually make it a function of the number (or index) of the neuron itself, which is therefore able to communicate two information simultaneously in a natural way. Thanks to this implicit variable the neurons number can be halved, speeding up both optimization and calculation.

The output structure is much simpler. Since the robot kinematics model has only 2-d.o.f., the number of output nodes of the ANN is 2, as follows:

- Steering output, $o_1 \in [0, 1]$,
- Acceleration speed output, $o_2 \in [0, 1]$.

The next section will describe how the output nodes values are translated in control signals to the motors.

2.6 Control and Life Cycle

The control system in our approach is represented by the ANN which is the result of the optimization carried out by the NEAT algorithm.

As discussed up to this point, the ANN produces 2 outputs which can have values between 0 and 1. One is assigned to steering, and the other to forward acceleration. The control approach works on a time-step dt basis, as follows

$$\begin{cases} v(t) = \min\left(\{v(t-1) + (k_a a^*)\, dt\ ,\ \ v_{\max}\}\right) \\ \omega(t) = k_\omega \omega^* \end{cases} \tag{12}$$

where $a^* = 6m/s^2$ is the commanded acceleration parameter, while $\omega^* = 2\pi\, 1/s$ is the commanded rotational speed; k_a and k_ω are the control coefficients which are actually controlled by the ANN and are mapped as follows, for o_1,

$$\begin{cases} o_1 \leq 0.333 & \text{then } k_\omega = 1 \\ o_1 \in\ ]0.333,\ 0.667[& \text{then } k_\omega = 0 \\ o_1 \geq 0.667 & \text{then } k_\omega = -1 \end{cases} \tag{13}$$

and for o_2,

$$\begin{cases} o_2 \leq 0.333 & \text{then } k_a = 1 \\ o_2 \in\ ]0.333,\ 0.663[& \text{then } k_a = 0 \\ o_2 \geq 0.667 & \text{then } k_a = -1 \end{cases} \tag{14}$$

It is clear from Eqs. 13 and 14, that a discretization is applied to the control signals so that the values can either be zero, or saturated in either sense (positive or negative). This approach was selected to reduce noise and to provide a comparable basis for the human driver experiment.

The life cycle of the general agent defined up to this point starts with its creation in the spawn point, and continues until either it collides with an obstacle or wall, or it reaches the target T. When one of these conditions becomes true, the agent is eliminated and *dies*.

2.7 Evolution

A classical NeuroEvolution problem consists in the optimization of a given weights set, the topology of the net is fixed and decided *a-priori*: usually an input layer, one

or more hidden layers, and one output layer. The process searches the space of the weights configuration of the ANN to find the best performing individuals. The NEAT method does the same but it also looks for the best possible ANN topology [41] by adding or removing connections, as illustrated in Fig. 3. In principle, this greatly enlarges the space over which NEAT searches for the optimal ANN and relieves the user from the burden of deciding in advance the topology of the network (i.e., the number of hidden layers). The latter consequence is particularly relevant in the case considered in this work, since the number of perception rays directly impact on the size of the ANN input layer, which in turn likely impacts on the optimal topology of the ANNs.

NEAT employs a number of mechanisms to efficiently perform the search in the space of ANNs with free topology: we summarize here the key ones here and refer the reader to [41] for a detailed description of the algorithm. Similar topologies are grouped together (*speciation*) and compete among themselves, giving time to optimize the weights of promising individuals, before being released in the general population.

Elitism is implemented by cloning the fittest individual of each species in the past generation and by insertion in the current generation. However, since the position of the target T changes at each generation, the score of the elite individuals may be lower than that of the same individuals at the preceding generation.

Evolution was performed on a population of $n_{\text{pop}} = 600$ individuals with NEAT. This number was found to provide a good balance between ease of computation and population diversity; indeed, since the evolutionary method we used implements speciation and elitism, diversity in the population is paramount to reaching meaningful results. A low number of individuals would mean that too many species would become extinct in the selection and crossover phases.

The number of generations n_{gen} is either 100 or 1000, based on the investigation reported in the following

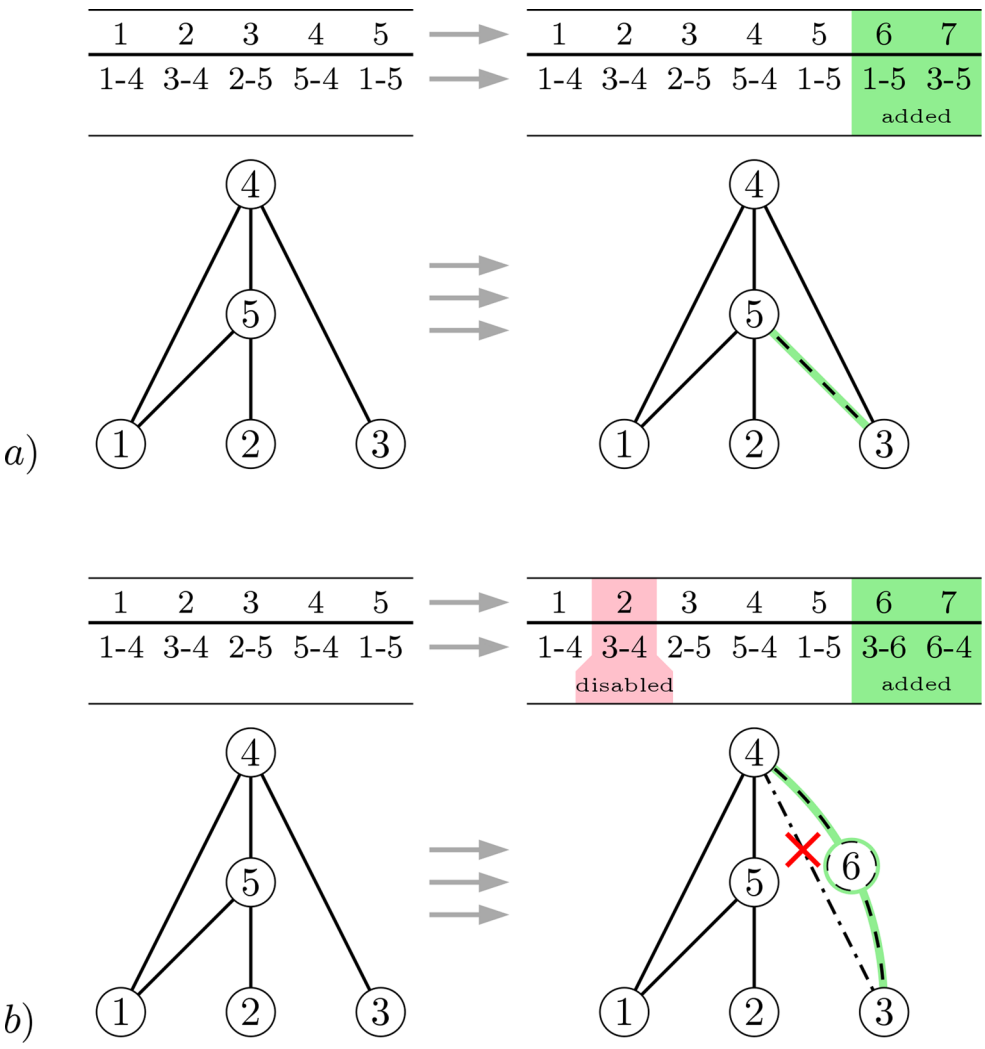

Fig. 3 Process of topological modification of the ANN during the optimization process of NEAT. In **a**) the process of addition of a connection between two nodes is shown; in **b**) the process of removal of a connection is shown together with the addition of a new node in place of the removed link

sections; specifically, the search of the parameter space has been done with $n_{\text{gen}} = 100$, while the convergence test has been performed with $n_{\text{gen}} = 1000$.

There are two separate fitness functions: one to evaluate the agent that did not reach the target, the other to evaluate those that were able to do so. In giving a score to agents that died before reaching the goal the main factor to consider is: how close were they?

$$f_1 = C_1 - d_g \tag{15}$$

where $C_1 = 3 \times 10^3$ is a positive constant to ensure that $f_1 > 0$, and d_g is the distance from the goal at the moment of death.

If the agent reaches the goal it gets evaluated by a different function:

$$f_2 = C_2 - k_a D - k_b T \tag{16}$$

where, D is the traveled distance, T the time alive, $k_a = 2$ and $k_b = 3$ are coefficients necessary to assign coherent weight to time and distance in the fitness function. C_2 is a parameter that grants to an agent that reached the target a higher fitness score than all those who did not; we set $C_2 = 2 \times 10^4$.

At this point the fitness function f can be expressed as follows,

$$f = \begin{cases} f_1 & \text{if target not reached} \\ f_2 & \text{if target reached} \end{cases} \tag{17}$$

This distinction is made to allow the algorithm to characterize successful agents (those which reached the goal) by considering the time and distance it took for these to reach the target, as captured by Eq. 16; at the same time, it should be noted that it is not possible to "score" non-successful agents as well with the same approach, since both D and T are not defined for non-successful agents. In this regard, Eq. 15 characterizes non-successful agents in terms of the "closeness" to target metric d_g; it is worth noting that for successful agents d_g is always zero.

3 Numerical Evaluation

In order to better characterize the performance of the approach presented in this work, we explore part of the parameter-space. This allows to create a rough preliminary map of the influence of the main parameters of the simulation. These are:

- the number n of rays r_i used by the robot;
- the PRP, i.e., how far back in terms of steps s_{lb} in the past is perceived by the NN;
- the mean speed $\bar{v}$ of the individuals in the environment;
- the number of individuals m in the environment.

Table 1 Parameters space

Parameter	Unit	Values
Number of rays n		15, **30**, 60, 90
Lookback steps s_{lb}		0, 2, 4, 6, 8, **10**
Crowd mean speed $\bar{u}$	ms^{-1}	0.25, **0.50**, 0.75, 1.00, 1.20, 1.50
Crowd size m		1, 5, 10, **15**, 20, 25
v_{wall}	ms^{-1}	0.15
L	m	18
W	m	8
R	m	0.6
R_R	m	0.2
R_{range}	m	3
n_{gen}		100
n_{pop}		600

Each run is performed by varying one parameter of the following. The other parameters are kept at their respective default value (highlighted in bold)

In the following, the findings are reported and discussed for each parameter. Table 1 summarizes the parameter space that was explored in the experimental campaign; the main other parameters are shown in the table on the right.

We recall that NEAT, as most other evolutionary algorithms, is a stochastic optimization method and hence results exhibit some randomness. In the following, in order to mitigate the effect of randomness on our conclusions, we repeated the optimization several times by varying the random seed and we analyzed the results using the proper statistical tools as, for example, box plots and descriptive statistics.

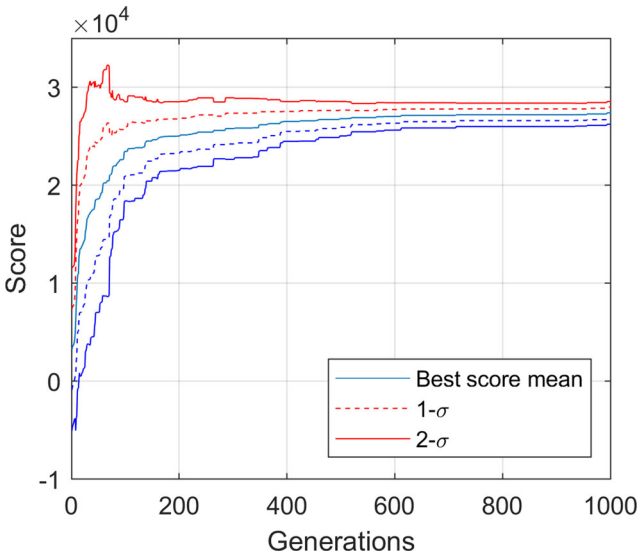

Fig. 4 Large number of generations test. Each generation averaged best score is plotted together with the 1 and 2-σ confidence bands

3.1 Convergence

Given the many unknowns related to the complexity and convergence of the NEAT evolution approach, it was elected to perform a preliminary set of tests that involve a great number of generations ($n_{\text{gen}} = 1000$).

In total, a number of 13 evolutionary runs were performed, by varying the random seed, with the default value for parameters described in Table 1. For each run the calculation of the cumulative maximum was performed, i.e., the fitness score of the best individual at the λ-th generation,

$$f_\lambda = \max\left(\left\{f_{i_\lambda} : i_h = 1, \ldots, \lambda\right\}\right) \tag{18}$$

where λ is the λ-th generation and f_λ is the fitness function score at generation λ.

Aggregate results from the 13 runs for a population of $n_{\text{pop}} = 600$ are shown in Fig. 4, in terms of the best-score development of each generation population.

It is apparent that the evolutionary process rapidly increases in the first 100 generations approximately, to finally reach a plateau around generation 500. The standard deviation bands (1, 2-σ related to the distribution of scores at each generation) show consistent thinning from generation 200. Based on these observations, and taking into account the CPU-time requested by a large number of runs of the algorithm, it was decided to consider $n_{\text{gen}} = 100$ as the threshold for the parameter exploration campaign that follows; it is a compromise between a reasonable convergence and computational resources requirements: the maximum score at the 100-th generation reaches within 10% of the score at the 1000-th generation. Furthermore, in order to contextualize these assumtions, it is important to note that these thresholds are valid with the conditions of this test; should the search difficulty increase, e.g., due to an increase in the inputs of the ANN, then the thresholds should be increased as well.

A set of example trajectories is shown in Fig. 5. It can be seen that already at generation 40, the trained ANN allows the robot to come within reach of the target. However, 20 generations later, the robot fails; at generation 80, the robot finally is able to achieve its goal.

3.2 Number of Rays

The number n of rays $\hat{\mathbf{r}}_i$ used by the system to perceive the surroundings, directly influences the resolution of the local map used by the ANN as the basis of its decision process. In order to explore how this influences the performance of the navigation system, we performed an experiment consisting in a number of evolutionary runs for each of a set of values of n. At least 5 runs were performed for each value of n.

For each run the cumulative maximum was computed according to Eq. 18. For each generation across the run spectrum, the average was computed and plotted in Fig. 6a.

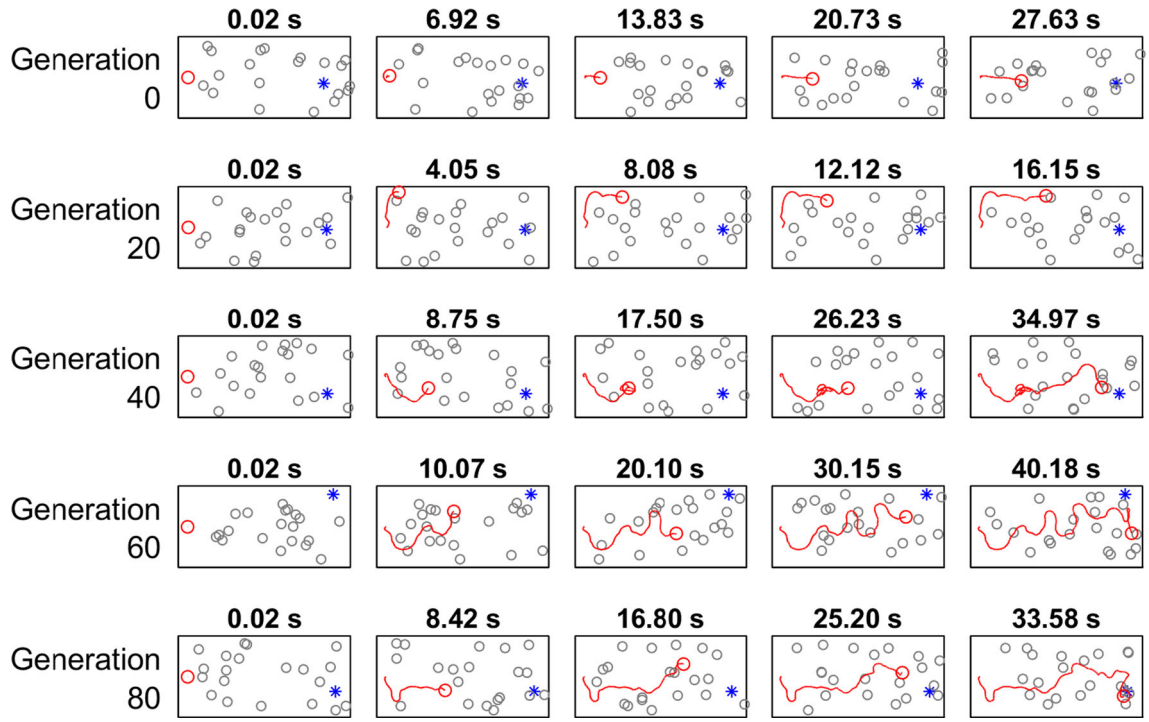

Fig. 5 Selection of best individuals for an evolution run with $\bar{u} = 1.0\text{ms}^{-1}$, $m = 20$ at 20 generation intervals. The small gray circles represent the crowd individuals $\mathbf{Q}_j$, the larger red circle represents the robot $\mathbf{P}$, while the blue asterisk indicates the target $\mathbf{T}$ location. Each row represents a different generation, while each column shows the trajectory of the best individual at different moments in time

The plot shows small substantial variation between different n runs when investigating the value at convergence. Indeed, from Fig. 6e it can be seen that most values fall within a similar interval in the last 20 generations. The lower score obtained by $n = 90$ may be due to a lower than necessary number of generations. No statistically significant effect can be detected conclusively, given the comparatively low number of runs.

However, it can be appreciated that the cumulative maximum line for the case of $n = 15$ and $n = 30$ are slower to reach convergence than the others. This will be investigated in a later section.

3.3 Lookback Parameter

The lookback parameter s_{lb} gives to the cognition model a certain knowledge of the past, hence of speed or rate of change of a certain input. From results plotted in Fig. 6b it seems that the evolution trajectory is generally indistinguishable if $s_{\text{lb}} > 0$, where the case of s_{lb} is that where there is no knowledge of past values. This last case appears substantially lower in terms of convergence speed performance. If one looks at the statistical aggregation presented in Fig. 6f, relative to the last 20 generations, it appears that s_{lb} does not influence the performance significantly when non-null. Again, the plot demonstrate the lower performance in terms of fitness score of the case of $s_{\text{lb}} = 0$.

3.4 Impact of Problem Parameters

The task of navigating the crowded environment intuitively becomes harder the more individuals there are and the more prevalent their motion is in relation to that of the robot. In

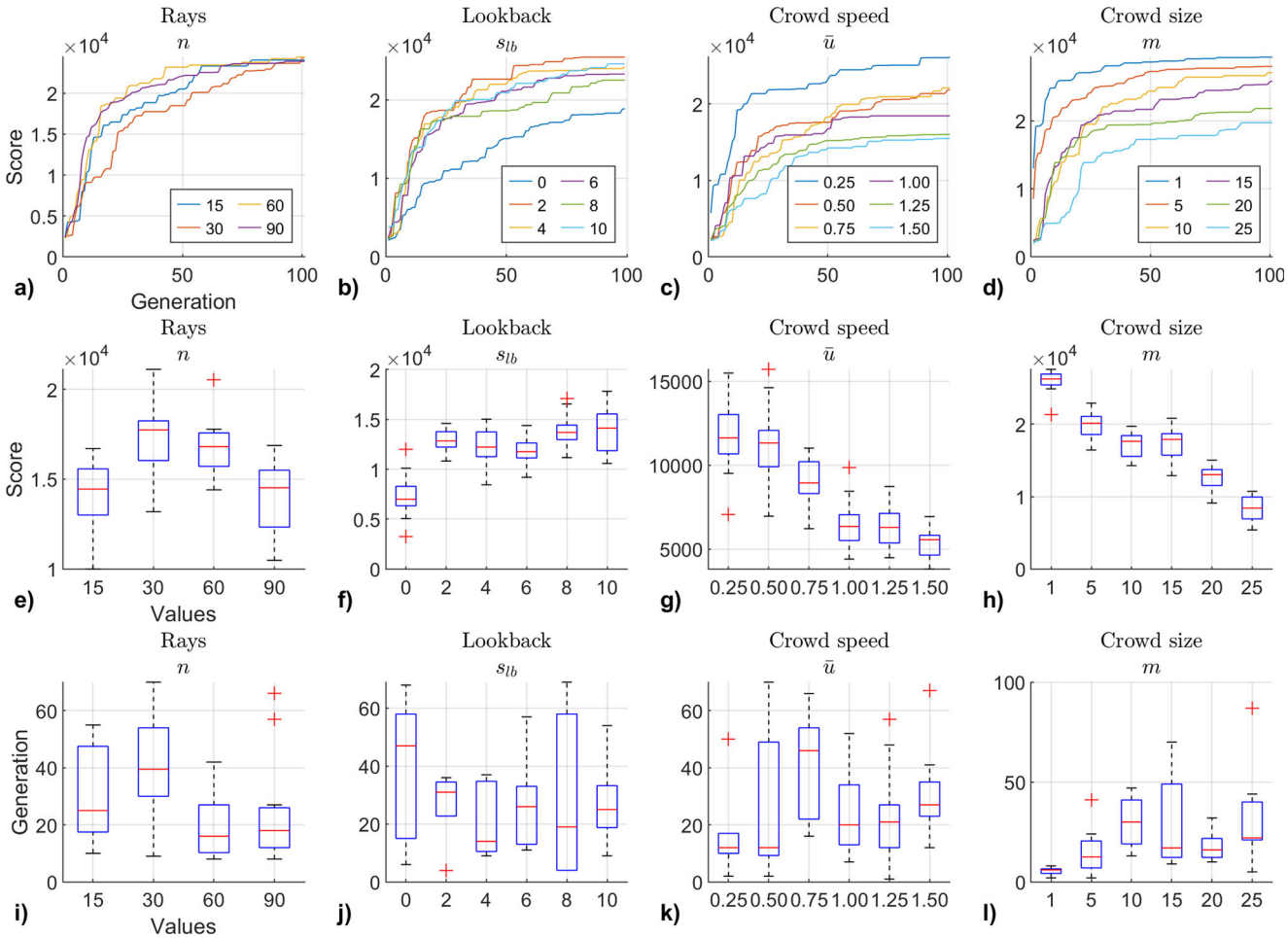

Fig. 6 Absolute best score over time (**a-d**), the distribution of the final 20 generations scores (**e-h**) and distribution of the speed of convergence (**i-l**). In the first row, each line represents the mean value of the overall best score of each underlying evolution run, averaged at each generation, grouped based on the explored parameter value; in **a**) the explored parameter is n, in **b**) is s_{lb}, in **c**) the speed $\bar{u}$, and in **d**) the size of the crowd m. In the second row, the scores distribution of runs at different values of the parameters are shown; in **e**) varying n, in **f**) s_{lb}, in **g**) $\bar{u}$, and in **h**) the crowd size m. In the last row, the speed of convergence to 75% of the best score (defined as that at generation 100) is shown; in **i**, **j**, **k**, **l** varying respectively parameters n, s_{lb}, $\bar{u}$ and m

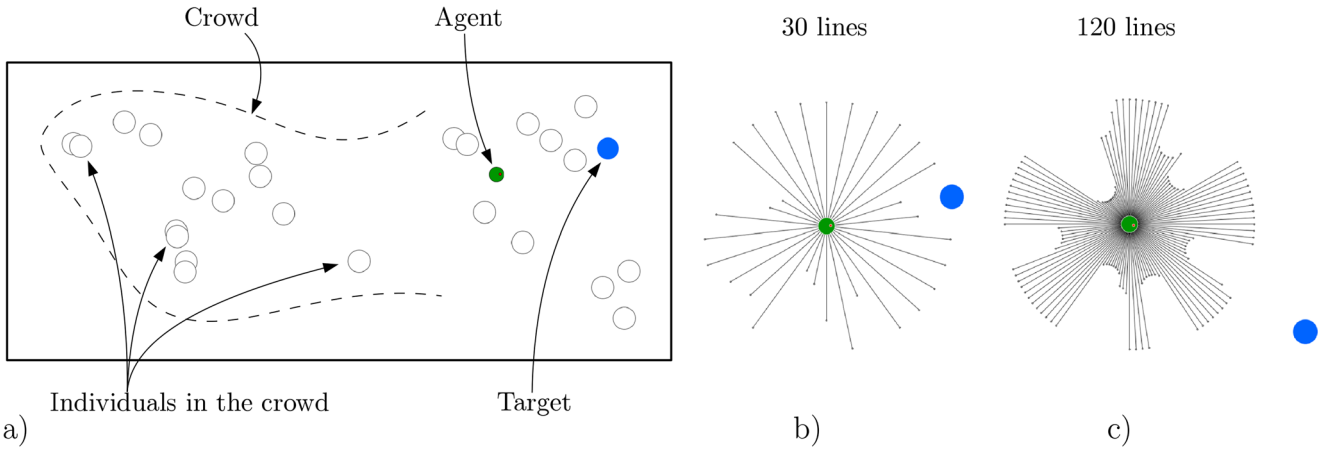

Fig. 7 The environment display for the user. In **a**) the full view is shown, along with the main components; the dashed line, here used to indicate the crowd, is not visible by the user. In **b**) a 30-lines case is visible to the user, along with the target (in blue), and in **c**) the same is shown for the 120-lines raycasting. Note that **a**) and **b**) are partial views of the environment

the simulated environment, each individual is characterized by a mean speed parameter $\bar{u}$ described in Eq. 2. The number of individuals in the crowd is m.

A series of runs was performed varying both of these parameters in order to investigate their relation on the performance of NEAT.

It is immediately apparent by looking at plots c), d), g) and h) of Fig. 6 that these parameters have a great degree of influence on both the speed of convergence and the ultimate fitness score of the NEAT algorithm.

3.5 Speed of Convergence

In order to analyze in detail the speed at which convergence is reached while exploring the parameter space, we define a threshold $f_{\text{th}} = 0.75 f_c$, where $f_c = f_{n_{\text{gen}}}$, which corresponds to the best score of each run. At this point, for each run, we can determine the first generation λ_c for which $f_h \geq f_{\text{th}}$. The plot in Fig. 6i-l shows the distributions of λ_c for different values of the usual parameters.

It is apparent (Fig. 6i) that for what regards the number of rays n, it appears that a higher number helps in reaching convergence sooner, as opposed to the case of $n = 15$ and especially $n = 30$ where convergence is slow. When the lookback depth s_{lb} is considered, the main difference is between $s_{\text{lb}} = 0$ and $s_{\text{lb}} > 0$, with the former delivering a slower climb towards convergence. The influence of the mean speed $\bar{u}$ of the crowd is hard to analyze. It seems that a peak exist at around $\bar{u} = 0.75\text{ms}^{-1}$, where convergence is especially slow; this probably indicates that there is a certain ratio between the robot maneuverability and the crowd speed for which best results are achieved. This finding could be useful for deciding the speed of the robot as a function of the average speed of the crowd. However, we remark that other factors, namely the acceptance of a robot by humans, might be considered while setting a value for the robot speed. Investigating these factors is beyond the scope of this paper. Finally, the number m of individuals in the crowds has a rather large and predictable effect on the speed of convergence of NEAT; a lower count of persons allows convergence to be reached sooner, however starting from 5–10 individuals, the variability increases and results appear less consistent. This is nevertheless expected behavior.

4 Experimental Evaluations

In this section, two distinct experiments are presented. In order to give an additional measure to contextualise the performance of the NEAT-controlled mobile robots, in the first experiment, a set of runs were performed by an untrained human. The second experiment, instead, implements the perception method described in Section 2.4, in a real physical environment.

4.1 Human Driver

The main purpose of this campaign is to develop an experimental environment; as such, what follows has no presumption to be an accurate and statistically relevant analysis of the performance of a human driver. This being said, in order to lay the foundation for a more accurate campaign, 61 runs were executed by a single untrained individual with each of the following configurations:

- $n = 30$ lines of sight,
- $n = 120$ lines of sight,
- full top view.

In particular, in the first two modes, the user can see a top-view of the environment with the location of the target T and of the robot $\mathbf{P}$. The perception of the environment is conveyed solely through the rays r_i and the points $\mathbf{S}_i$. The latter mode, i.e., full top view, shows the user the entire environment as illustrated in Fig. 1. The environment display is shown in Fig. 7. The subject can give the robot the same inputs of the neural net and it does it through the arrow keys while viewing the domain and the current score on a 13" full-HD screen. Each playing session consisted of 61 trials, the duration was dependant from the ability of the player to stay alive (every attempt could last from just a couple seconds up to about 20s) and each session begun as the player felt ready in order to maximize his comfort. The subject was 26 years old and had experience in gaming but he had never tried this simulation before the test.

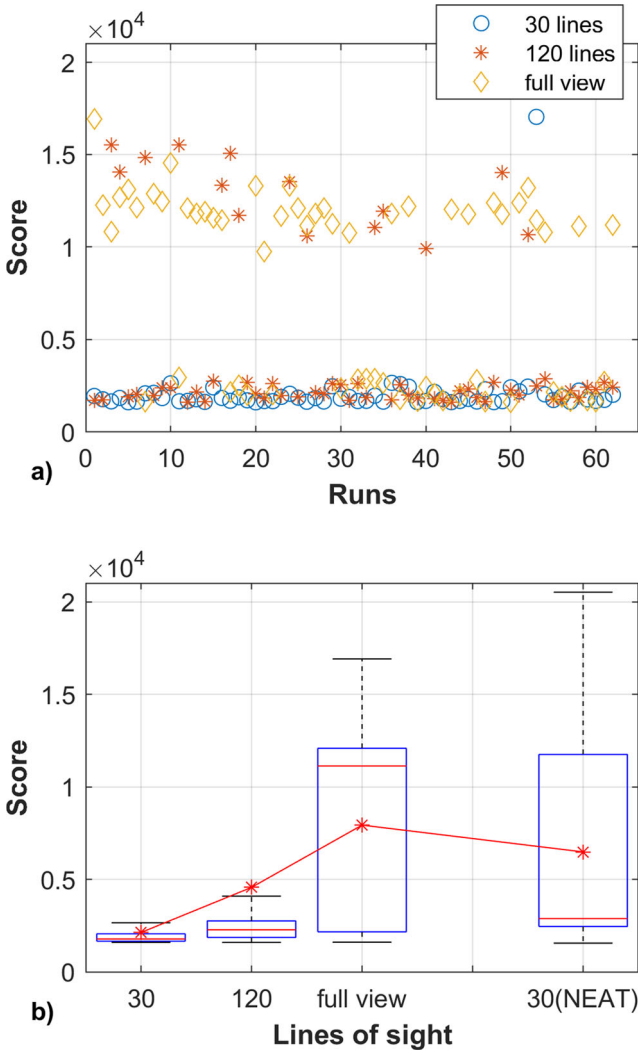

Fig. 8 Distribution of the scores of the human driver. In **a**) a temporal view of the distribution of performance results is shown for the human-driven system; in **b**) a statistical aggregation is shown, compared to the results from a large number of NEAT-controlled runs

In Fig. 8 the results of this preliminary experimental campaign are shown, both in temporal and in aggregate view.

The temporal view, in a), shows the full number of runs performed by the human driver. While it does appear to be a certain downward trend in the performance (i.e., the score), especially with the *120 lines* and *full view* cases, this appears rather shallow and should, if confirmed, be ascribed mainly to fatigue and boredom, given the large number of runs. Furthermore, no learning appears to happen, since in neither of the three cases an upward trend can be determined.

What is however immediately apparent from the box-plot in b) is that the human driver performs rather poorly when provided with low-n feedback, while its performance increases considerably with higher resolution perception (*120-lines*) and even more so with a "realistic" perception of the environment (*full view*). Indeed, the human driver could not reach the target in all but one case in the *30-lines* case (win ratio 1.61%). Conversely, the *full view* mode allowed the driver to complete the task most of the times (win ratio 58.1%). The *120-lines* saw an intermediate win ratio of 22.6%.

4.2 Perception Experimental Implementation

In order to conduct the experimental tests, the TurtleBot3 Waffle Pi mobile robot was selected. The robot is a small differential drive platform which is equipped with a LDS-01 LiDAR; the scanner main properties are summarized in Table 2.

The mobile robot has been positioned in a indoor room surrounded by a simulated crowd; a representation of the mock crowd is shown in Fig. 9a. Due to the small scale of the experiment, the LiDAR maximum range has been set to 2m.

The test was conducted by making use of the ROS (Robot Operating System) network and its toolboxes available in MATLAB in the Robotics Toolbox. The process has been synthesized by developing a specialized application, whose layout can be seen in Fig. 10. This enables the communications process, and the analysis steps that take place between getting the scan readings from the mobile

Table 2 Characteristics of the LDS-01 LiDAR range sensor

Properties	Value	Unit
Distance range	120–3500	mm
Angular range	2π	rad
Accuracy on d.120499mm	± 15	mm
Accuracy on d.5003500mm	± 3.5	%
Angular speed	300 ± 10	min^{-1}
Angular resolution	0.0175	rad

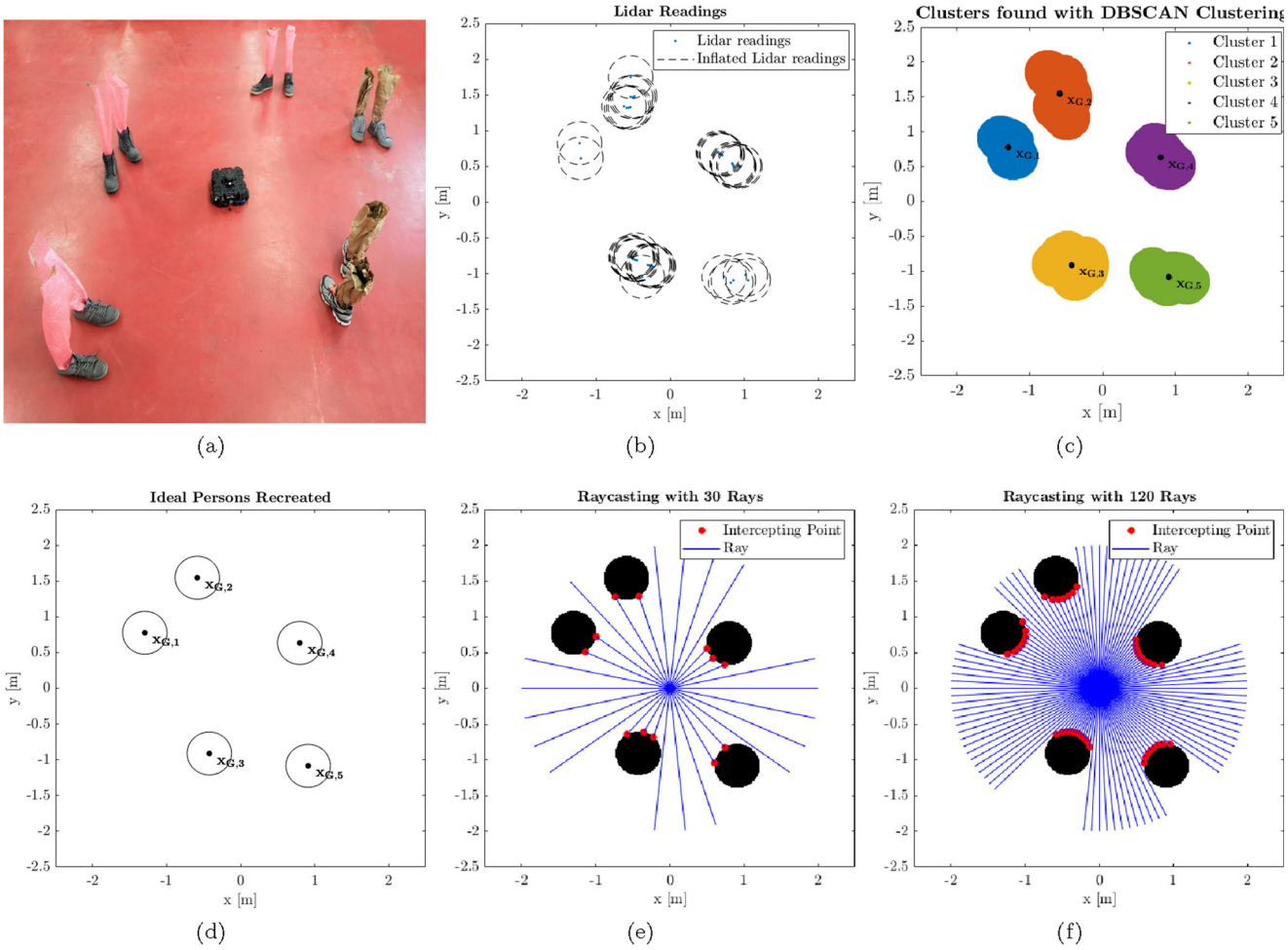

Fig. 9 In **a** the environment setup with simulated legs for the experimental raycasting; in **b** the Lidar Readings and its next enlargement in order to create a logical pixel map; in **c** the clusters, representing the single person, and their centroids as result of the clustering process; in **d** the ideal persons recreated around the clusters centroids; in **e** the raycasting done with 30 rays on the map representing the ideal persons; in **f** the same done with 120 rays

robot and the final ray casting. A brief description of the workflow is given below:

- (Positioning) The robot reaches its intended position by driving,
- (Scanning) A number of scan readings from the LiDAR sensor are taken. The data obtained for each reading in the form of distances and angles, is encoded in a 2D occupancy matrix, which represents the map of the environment. An inflation operation is performed to the matrix in order to represent the robot safety radius (Fig. 9b). Note that the map coordinate system and the one of the mobile robot are coincident.
- (Clustering) In order to classify pixels based on their association with the objects in the environment, the occupied points of the map have been clustered by using a DBSCAN algorithm (Density-Based Spatial Clustering of Applications with Noise). As Fig. 9c shows, in our experiment five different clusters have been successfully found that represent the five simulated persons.
- (Obstacle representation) The centroid is computed for each cluster. A synthetic circular obstacle is generated at each centroid, with a radius representing the ideal person as seen by the robot, including a safety distance. The result of this operation is shown in Fig. 9d.
- (Raycasting) The raycasting is performed following the method presented in Section 2.4; two representative sets are reported, 30 and 120 rays, respectively in Fig. 9e and f.

As can be seen the mobile robot is capable of detecting the obstacles around it, subsequently recreate the ideal representation of the person and then finally compute the raycasting with 30 and 120 rays. In the tests done with both rays densities, the robot was capable to find each

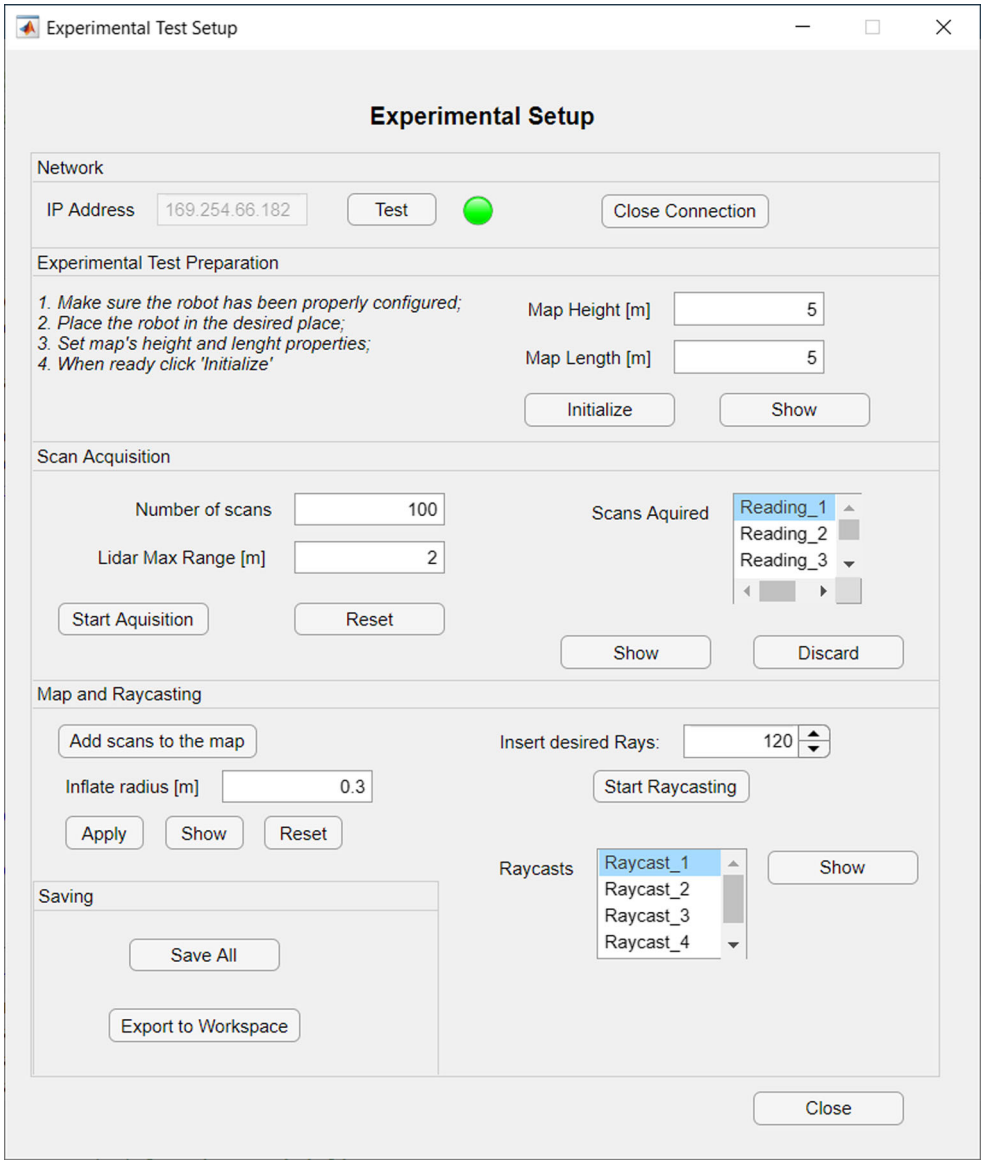

Fig. 10 Screenshot of the MATLAB application developed to get the LiDAR Readings of the Turtlebot3 Waffle Pi, build the environment map, recreate the ideal person and compute the raycasting

person with at least 2 rays. It is clear that with 30 rays, information about a certain obstacle is low; on the other hand it is noteworthy that the closer an obstacle is to the robot, the more rays detect it, thus increasing it is relative importance. Escalating the number of rays (e.g., 120) greatly increases resolution and obstacle representation potential, as expected.

5 Concluding Remarks

The task of navigating promiscuous environment where robots coexist with people is one which is coming ever more into focus in recent times. With door-to-door robotic delivery, autonomous cars and tele-presence, this field is rapidly evolving.

In this paper we build on the state-of-the-art with the implementation of a controller for a mobile robot, based on artificial neural networks that are optimized with NEAT. This approach is especially appropriate to confront this problem due to the fact that NEAT implements successfully the complexification paradigm to ANNs; the procedure starts with simple and small ANNs, which are grown when the need arises. This promises to provide good performance in tasks which involve complex perception capabilities for navigating a crowded environment to reach a target.

Perception in the robot is implemented with a raycasting model that mimicks and can be easily implemented in an industrial LiDAR-equipped robot. The system perceives both current time objects and has memory of past perception (*lookback*).

In order to better characterize this approach, we performed a thorough exploration of the parameter space, investigating the influence of crowd size, its average speed and the characteristics of the perception model both in time and in resolution.

Results show that the controller is able to converge to a feasible solution in a comparatively low number of generations.

Finally, the difference between the simulation and a physical implementation (reality-gap) has been analyzed through an experiment consisting in the detection of persons via LiDAR, in a mock environment. Results show that the methodology is feasible.

Future developments will see the implementation of the NEAT controller in a physical mobile robot; in this context, the performance of the approach will be evaluated in a real crowd environment, in controlled conditions. Results will be compared to those presented in this work.

Author Contributions Conceptualization: L.M., S.S., E.M., P.G.; Methodology: L.M., S.S., E.M.; Software: L.M., S.S; Validation: S.S., M.C.; Formal analysis: S.S.; Investigation: S.S., L.M.; Resources: S.S.; Data Curation: S.S.; Writing - Original Draft: S.S., E.M., L.M., M.C.; Writing - Review & Editing: S.S., P.G., E.M.; Visualization: S.S., M.C.; Supervision: S.S., P.G.; Project administration: S.S., P.G., E.M.; Funding acquisition: S.S., P.G.

Funding Open Access funding provided by Università degli Studi di Trieste. This work has been partially supported by the PRIN project779 SEDUCE n. 2017TWRCNB and by the University of Trieste - University funding for scientific research projects - FRA 2018.

Availability of data and material The data that support the findings of this study are available from the corresponding author, S.S., upon reasonable request.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

Code availability The code used to generate the results presented in this study is available from the corresponding author, S.S., upon reasonable request.

References

1. Jeon, S., Lee, J.: Performance analysis of scheduling multiple robots for hospital logistics. pp 937–940 (2017)
2. Bays, M., Wettergren, T.: Service agent–transport agent task planning incorporating robust scheduling techniques. Robot. Auton. Syst. **89**, 15–26 (2017)
3. Hernandez, K., Bacca, B., Posso, B.: Multi-goal path planning autonomous system for picking up and delivery tasks in mobile robotics. IEEE Lat. Am. Trans. **15**(2), 232–238 (2017)
4. Baker, M., Yanco, H.: Automated street crossing for assistive robots. **2005**, 187–192 (2005)
5. Bauer, A., Klasing, K., Lidoris, G., Mühlbauer, Q., Rohrmüller, F., Sosnowski, S., Xu, T., Kühnlenz, K., Wollherr, D., Buss, M.: The autonomous city explorer: Towards natural human-robot interaction in urban environments. Int. J. Soc. Robot. **1**(2), 127–140 (2009)
6. Chung, M.Y., Pronobis, A., Cakmak, M., Fox, D., Rao, R.: Autonomous question answering with mobile robots in human-populated environments. vol. 2016-November, pp 823–830 (2016)
7. Radwan, N., Winterhalter, W., Dornhege, C., Burgard, W.: Why did the robot cross the road? - learning from multi-modal sensor data for autonomous road crossing. vol. 2017-September, 4737–4742 (2017)
8. Lutin, J., Kornhauser, A., Lerner-Lam, E.: The revolutionary development of self-driving vehicles and implications for the transportation engineering profession. ITE J (Inst Transport Eng) **83**(7), 28–32 (2013)
9. Lombardi, G., Medvet, E., Bartoli, A.: A language for Uav traffic rules in an urban environment and decentralized scenario. In: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1139–1143. IEEE (2017)
10. Burgoon, J.K., Jones, S.B.: Toward a theory of personal space expectations and their violations. Hum. Commun. Res. **2**(2), 131–146 (1976)
11. Guzzi, J., Giusti, A., Gambardella, L.M., Theraulaz, G., Caro, G.A.D.: Human-friendly robot navigation in dynamic environments. In: 2013 IEEE international conference on robotics and automation, pp. 423–430 (2013)
12. Hebesberger, D., Koertner, T., Gisinger, C., Pripfl, J.: A long-term autonomous robot at a care hospital: A mixed methods study on social acceptance and experiences of staff and older adults. Int. J. Soc. Robot. **9**, 417–429 (2017)
13. Trautman, P., Krause, A.: Unfreezing the robot navigation in dense, interacting crowds. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 797–803 (2010)
14. Luber, M., Spinello, L., Silva, J., Arras, K.O.: Socially-aware robot navigation: A learning approach. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 902–907 (2012)
15. Trautman, P.: Robot Navigation in Dense Crowds Statistical Models and Experimental Studies of Human Robot Cooperation. Doctoral Thesis, California Institute of Technology,Pasadena, California (2013)
16. Trautman, P., Ma, J., Murray, R.M., Krause, A.: Robot navigation in dense human crowds: The case for cooperation. In: 2013 IEEE International Conference on Robotics and Automation, pp. 2153–2160 (2013)
17. Trautman, P., Ma, J., Murray, R.M., Krause, A.: Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. Int. J. Robot. Res. **34**(3), 335–356 (2015). https://doi.org/10.1177/0278364914557874

18. May, A.D., Dondrup, C., Hanheide, M.: Show Me Your Moves! Conveying navigation intention of a mobile robot to humans. In: 2015 European Conference on Mobile Robots (ECMR), pp. 1–6 (Sept 2015)
19. Shrestha, M.C., Kobayashi, A., Onishi, T., Yanagawa, H., Yokoyama, Y., Uno, E., Schmitz, A., Kamezaki, M., Sugano, S.: Exploring the use of light and display indicators for communicating directional intent. In: 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1651–1656 (2016)
20. Shrestha, M.C., Kobayashi, A., Onishi, T., Uno, E., Yanagawa, H., Yokoyama, Y., Kamezaki, M., Schmitz, A., Sugano, S.: Intent communication in navigation through the use of light and screen indicators. In: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 523–524 (2016)
21. Chik, S., Yeong, C., Su, E., Lim, T., Subramaniam, Y., Chin, P.: A review of social-aware navigation frameworks for service robot in dynamic human environments. J. Telecommun. Electron. Comput. Eng. **8**(11), 41–50 (2016)
22. Narayanan, V.K., Spalanzani, A., Luo, R.C., Babel, M.: Analysis of an adaptive strategy for equitably approaching and joining human interactions. In: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 341–346 (2016)
23. Park, C., Ondřej, J., Gilbert, M., Freeman, K., O'Sullivan, C.: Hi Robot: Human intention-aware robot planning for safe and efficient navigation in crowds. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3320–3326 (2016)
24. Chatterjee, I., Steinfeld, A.: Performance of a low-cost, human-inspired perception approach for dense moving crowd navigation. In: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 578–585 (2016)
25. Bohorquez, N., Sherikov, A., Dimitrov, D., Wieber, P.B.: Safe navigation strategies for a biped robot walking in a crowd. In: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pp. 379–386 (2016)
26. Kim, B., Pineau, J.: Socially adaptive path planning in human environments using inverse reinforcement learning. Int. J. Soc. Robot. **8**, 51–66 (2016)
27. Vemula, A., Mülling, K., Oh, J.: Modeling cooperative navigation in dense human crowds. arXiv:1705.06201 (2017)
28. Bera, A., Randhavane, T., Prinja, R., Manocha, D.: Sociosense: Robot navigation amongst pedestrians with social and psychological constraints. arXiv:1706.01102 (2017)
29. Patle, B., Parhi, D., Jagadeesh, A., Kashyap, S.: Application of probability to enhance the performance of fuzzy based mobile robot navigation. Appl Soft Comput J **75**, 265–283 (2019)
30. Al-Araji, A., Ibraheem, B.: A comparative study of various intelligent optimization algorithms based on path planning and neural controller for mobile robot. Univ. Baghdad Eng. J. **25**(07), 80–99 (2019)
31. Das, S., Mohanty, S., Behera, A., Parhi, D., Pradhan, S.: Navigational control analysis of mobile robot in cluttered unknown environment using novel neural-gsa technique. Lecture Notes Mechan. Eng. 551–563 (2020)
32. Chen, Y., Liu, C., Shi, B., Liu, M.: Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. IEEE Robotics Autom. Lett. **5**(2), 2754–2761 (2020)
33. Rizk, Y., Awad, M., Tunstel, E.: Decision making in multiagent systems: A survey. IEEE Trans. Cognitive Develop. Syst. **10**(3), 514–529 (2018)
34. Abbas, H., Saha, I., Shoukry, Y., Ehlers, R., Fainekos, G., Gupta, R., Majumdar, R., Ulus, D.: Special session: Embedded software for robotics: Challenges and future directions (2018)
35. Chen, Y., Gan, W., Zhang, L., Liu, C., Wang, X.: A survey on visual place recognition for mobile robots localization. vol. 2018-January, pp. 187–192 (2018)
36. Dudarenko, D., Kovalev, A., Tolstoy, I., Vatamaniuk, I.: Robot navigation system in stochastic environment based on reinforcement learning on lidar data. Smart Innov. Syst. Technol. **154**, 537–547 (2020)
37. Gordon, V.S., Whitley, L.D.: Serial and parallel genetic algorithms as function optimizers. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 177–183. Morgan Kaufmann Publishers Inc, San Francisco (1993)
38. Gomez, F., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. **2**, 1356–1361 (1999)
39. Moriarty, D., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution. Evol. Comput. **5**(4), 373–399 (1997)
40. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: A survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
41. Stanley, K., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
42. Buk, Z., Koutník, J., Šnorek, M.: Neat in hyperneat substituted with genetic programming. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5495 LNCS, pp. 243–252 (2009)
43. Caceres, C., Rosario, J., Amaya, D.: Approach of kinematic control for a nonholonomic wheeled robot using artificial neural networks and genetic algorithms (2017)
44. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
45. Ying, X.: An overview of overfitting and its solutions. vol. 1168 (2019)
46. Antonio, F.: Iv.6 - Faster line segment intersection. In: Kirk, D. (ed.) Graphics Gems III (IBM Version), pp. 199–202. Morgan Kaufmann, San Francisco (1992)

Stefano Seriani received his B.E. (2010) and his M.Sc in mechanical engineering (2012) from the University of Trieste, Italy. He received his PhD in April 2016 at the University of Trieste, Italy. In 2016-2017 he was research fellow at the Institute of Robotics and Mechatronics of the German space agency (DLR), working mainly on compliant mechanisms for rovers and modular robotics for space exploration. He is now research fellow at University of Trieste. His research interests include space robotics, applied mechanics, and mobile robotics.

Luca Marcini received his M.Sc. in Mechanical Engineering from the University of Trieste. His interests are in mobile robotics, neural networks and differential evolution algorithms.

Matteo Caruso received his bachelor degree in 2016 and his master degree in mechanical engineering in 2019 from the University of Trieste, Italy, with a thesis on rovers for space exploration. He is currently a PhD student in the Department of Engineering and Architecture, University of Trieste, Italy. His research interests are mobile robotics and applied mechanics.

Paolo Gallina is currently full professor of Applied Mechanics at the Department of Engineering and Architecture, University of Trieste, Trieste (Italy). He was visiting professor at the Ohio University in 2000/1. In 2002 he implemented a hands-on Mechatronics Laboratory for students in Engineering. In 2003 he implemented a Robotics Laboratory where he carries out his main research in robotics. He was head of the Council for Students in Mechanical Engineering Degree from 2004 to 2008. He was head of the Council for Students in Industrial Engineering. He was the Director of the Master in Robotics at the University of Trieste. His interests are in vibrations, human-machine interfaces and robotics.

Eric Medvet received the degree in Electronic Engineering cum laude in 2004 and the PhD degree in Computer Engineering in 2008, both from the University of Trieste, Italy. He is currently an Assistant Professor in Computer Engineering at the Department of Engineering and Architecture of University of Trieste, Italy, where he leads the Evolutionary Robotics and Artificial Life lab (ERALlab) and is the co-head of the Machine Learning Lab. His research interests include Genetic Programming and Machine Learning applications.