sensors

Article

Social Robot Navigation Tasks: Combining Machine Learning Techniques and Social Force Model

Óscar Gil †, Anaís Garrell † and Alberto Sanfeliu *

Intitut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028 Barcelona, Spain; oscar.gil.viyuela@upc.edu (Ó.G.); anais.garrell@upc.edu (A.G.)
* Correspondence: alberto.sanfeliu@upc.edu
† These authors contributed equally to this work.

Abstract: Social robot navigation in public spaces, buildings or private houses is a difficult problem that is not well solved due to environmental constraints (buildings, static objects etc.), pedestrians and other mobile vehicles. Moreover, robots have to move in a human-aware manner—that is, robots have to navigate in such a way that people feel safe and comfortable. In this work, we present two navigation tasks, social robot navigation and robot accompaniment, which combine machine learning techniques with the Social Force Model (SFM) allowing human-aware social navigation. The robots in both approaches use data from different sensors to capture the environment knowledge as well as information from pedestrian motion. The two navigation tasks make use of the SFM, which is a general framework in which human motion behaviors can be expressed through a set of functions depending on the pedestrians' relative and absolute positions and velocities. Additionally, in both social navigation tasks, the robot's motion behavior is learned using machine learning techniques: in the first case using supervised deep learning techniques and, in the second case, using Reinforcement Learning (RL). The machine learning techniques are combined with the SFM to create navigation models that behave in a social manner when the robot is navigating in an environment with pedestrians or accompanying a person. The validation of the systems was performed with a large set of simulations and real-life experiments with a new humanoid robot denominated IVO and with an aerial robot. The experiments show that the combination of SFM and machine learning can solve human-aware robot navigation in complex dynamic environments.

Keywords: social robot navigation; Social Force Model; Reinforcement Learning

Citation: Gil, Ó.; Garrell, A.; Sanfeliu, A. Social Robot Navigation Tasks: Combining Machine Learning Techniques and Social Force Model. *Sensors* **2021**, *21*, 7087. https://doi.org/10.3390/s21217087

Academic Editor: Abdelaziz Benallegue

Received: 7 September 2021
Accepted: 15 October 2021
Published: 26 October 2021

1. Introduction

The introduction of robotics in our daily activities in the near future will require the navigation of humans and robots in different environments, including public spaces, buildings and private houses. The applications are huge, for example in assistive robotics tasks [1], collaborative searching [2], side-by-side navigation [3], guiding people [4] or other social navigation tasks. In any of these applications, robots have to behave in a social manner, being social-aware (that is, the robot has to plan the trajectory helping the accompaniment or guiding a person while not disturbing other pedestrian trajectories if possible) and avoiding colliding with obstacles or pedestrians.

In this work, we present a combination of SFM and machine learning techniques that, using the robot's perception systems, make the navigation of the robots easier in environments where there are static obstacles, such as walls or urban furniture and moving objects, like pedestrians or bicycles. The SFM is a general framework in which the human motion behaviors can be expressed through a function depending in the pedestrians relative and absolute positions and velocities.

In machine learning techniques, we present two different approaches: a supervised learning approach based on a neural network that is used for accompanying a person with an UAV (Unmanned Aerial Vehicle) and an RL technique that is used for robot navigation in areas where there are pedestrians moving around.

The SFM and the machine learning techniques are combined in a different ways in each of the applications. In the first case, the forces of the SFM are the inputs of a neural net, and the neural network (NN) learns the combination of repulsive forces produced by the environment and the pedestrian motions. In the second case, the SFM is used to avoid the static and motion obstacles, and the learned robot motion force is combined with the repulsive forces produced by the environment and pedestrian motions. In both cases, the resulting robot motion force is human-aware, because it modifies the robot behavior based on the pedestrian motion trajectory.

These two techniques were tested in simulation and real-life experiments. In Social Robot navigation experiments, simulation analysis has been done in complex urban scenarios and real-life experiments were performed by the IVO robot (see Figure 1). In the IVO's robot, we use two types of sensors, a 3D LiDAR to detect obstacles, pedestrian motions and to allow the self-localization of the robot as well as an RGBD RealSense camera to detect holes and ramps. Moreover, we use the odometry's robot sensors that are combined with the LiDAR localization sensor to obtain robustness in the robot localization.

In the UAV human accompaniment, the simulation analysis was performed in a simulated environment that included trees and walls, and the real-life experiments were performed between an UAV and a person, where the UAV accompanied the person in an environment with static obstacles. We used external Optitrack sensors to detect the human motion and the robot motion.

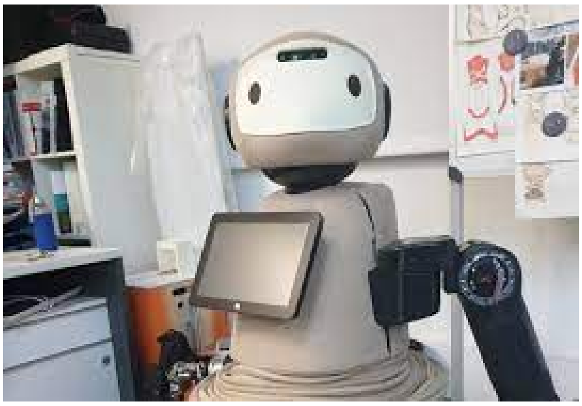


Figure 1. IVO Robot. New design of a humanoid robot for citizen assistance.

In this work, we describe in Section 2, a summary of the state of the art in social robot navigation, social robot accompaniment and deep RL in robot navigation. In Section 3, we introduce the problem definition. In Section 4, we describe the SFM and Automated Reinforcement Learning (AutoRL). In Section 5, we introduce the combination of RL with SFM, and, in Section 6, we describe the combination of supervised learning with SFM. Sections 7 and 8 describe the simulation analysis and real-life experiments of the social robot navigation and UAV human accompaniment tasks. Finally, Section 9 relates our conclusions.

2. State of the Art

The overall goal of this research is to develop robots that work cooperatively with humans while developing navigation tasks. As such, this article draws on work from diverse fields, including social robot navigation and the social robot accompaniment of people. This section introduces the data yielded by some of the most relevant research on this subject.

2.1. Social Robot Navigation

Human motion social interaction has been studied previously in several works [5–7]. Deepening on the SFM introduced by Helbing & Molnar in [8], human motion is modeled as a reaction to a set of attractive and repulsive forces generated by other people or the elements present on the environment, where people move safely and more comfortable along the minimum potential paths of the forces field.

In social navigation, a commonly used metric is the proxemic rules between humans and robots; moreover, Walter et al. [9] analyzed how the personality affects pedestrian spatial zones. Alternatively, another approach studied the person–robot proxemic distances in different interactions: verbal, physical and no interaction [10]. Furthermore, in [11,12], researchers found that people prefer robots to approach them by one of their sides; for instance, right-handed people prefer to be approached from the right.

Most social navigation models have either strict situational applicability or a pronounced reactive nature, as models defined based on the SFM stated by Helbing & Molnar [8]. Due to this, some models included predictive models on their structure [13,14] or assume the knowledge of the location of the final goal of the pedestrian or robot trajectory [15], which might demand a high computation cost and confer them a heuristic nature. The model proposed attempts to learn human behavior from a static world distribution, not relying on any knowledge inference. It aims to implement implicit prediction through the reading of the environmental characteristics.

Common walking habits of humans were investigated, as individuals, pairs or groups in [16–20]. Garrell et al. [13] focused on side-by-side pair walking, whose line of work is followed along this article. Other interesting approaches of social navigation are applied to autonomous wheelchairs [20], some of them focused on side-by-side movement [21,22].

This work goes one step forward, developing a new method based on Multi-Layer Perceptron to improve human–robot side-by-side social navigation. To tackle the introduced problem, we propose a new nonlinear-based approach that uses the Extended Social Force Model (ESFM) [23] as a means to extract meaningful information from the environment.

The field of Human Motion Prediction is similar to that of Social Navigation with the difference that only attempts to predict the future trajectories of humans in a scene, not navigate around them. Recent works in this field have focused on predicting future trajectories using Recurrent Neural Networks (RNNs) to encode given sequences of positions [24] and poses [25] and map them to predicted trajectories. These publications have improved works in the field; however, these models do not consider the Multimodality aspect in human trajectories, which is why works, such as [26,27] use Generative Adversarial Networks (GAN) in order to solve these problems. This approach to train the encoder–decoder RNN models, allows us to focus on our objective of obtaining correct socially acceptable trajectories instead of attempting to approximate the ones already in the database.

A common trait among all these methods is that, in order to obtain information and learn to account for social norms in their predictions, all these works keep a model of every person in a scene and process them together inside a self-designed pooling layer. This allows obtaining compact representations of the environment to feed into the model. However, these procedures are usually computationally expensive and difficult to scale up to a huge number of humans in a scene. Additionally, these models only take into consideration the humans present in the scene; only [27] attempted to account for static objects in the scene by feeding information on the top view of the scene to the model. However, this top view is not always available.

Seeing the limitations of these procedures, we propose the use of the SFM [8] as an alternative to obtain simple and compact representations of the environment. Given the position of static objects and humans in a region centered around the robot, the SFM allows us to describe in a simple and scalable way the environment without the need for external information.

2.2. Deep Reinforcement Learning in Robot Navigation

In recent years, deep RL algorithms [28,29] have demonstrated human-level control in some tasks, such as video games [30,31], where this is almost impossible to obtain using handcrafted features for the states or considering fully observable environments. For this reason, deep RL approaches are very common for visual navigation [32,33].

Other deep RL models are based on different environment features. In [34,35], a fully-observable environment with a discrete set of actions is considered where the robot receives all the agent positions and velocities. In [35], a Long Short-Term Memory (LSTM) cell is fed with an arbitrary number of moving agent states, and a training is performed considering all the agent experiences in a scene to increase the joint expected reward. In [34], the moving agents are simulated using the Optimal Reciprocal Collision Avoidance algorithm (ORCA) [36], and Imitation Learning is used to improve the performance. Although, as is described in [37], the performance is only high when the robot is close to the goal with few moving obstacles around.

Some works consider prediction information to improve the algorithm, for example in [38], where people are tracked, and the next frame for a person is predicted. This prediction is used as an additional observation in the Proximal Policy Optimization algorithm (PPO). This prediction reduces the freezing robot problem in dense crowds.

3. Problem Definition

As has been mentioned previously, in this work, we investigate two different social navigation tasks, social robot navigation and robot accompaniment, and both combine machine learning techniques with the Social Force Model (SFM).

In this section, we proceed to describe the problem definition of both social tasks.

3.1. Reinforcement Learning with Social Force Model

Given a dynamic environment, which consists of static objects and pedestrians, the objective is to navigate being human-aware. The robot actions, linear and angular velocities, are computed by a combination of robot velocities learned by a reinforcement learning model (AutoRL [39]) and robot velocities computed using a SFM. The SFM is used to take into account the moving pedestrians, which allows human-aware navigation. In this model, we use robot Lidar measurements to acquire the environment and pedestrian observations.

We consider the robot navigation task as a Partially Observable Markov Decision Process (POMDP) defined by the tuple (O, A, D, R, γ) where the action space, A, and the observation space, O, is considered continuous. D are the dynamics, provided by the simulation environment or the real world, R is the reward described in Equation (13), and $\gamma \in (0, 1)$ is the discount factor.

The robot model is a circle of radius 0.3 m, and the actions are the typical velocity commands: linear and angular velocities, $\mathbf{a} = (a_l, a_\phi)$. The observations, $\mathbf{o} = (\mathbf{o}_l, \mathbf{o}_g)^{\theta_n} \in O$, are divided in two parts: $\mathbf{o}_l$ are 64 1-D LiDAR distance lectures taken during the last θ_n simulation steps and $\mathbf{o}_g$ are the goal polar coordinates during the last θ_n steps. Each simulation step is about 0.2 s, and the LiDAR field of view is 220°. The distance range for the LiDAR lectures goes from 0 to 5 m. The action space for the linear velocity is $v_l \in [-0.2, 1]$ m/s and $v_\phi \in [-1, 1]$ rad/s for the angular velocity.

3.2. Supervised Learning with Social Force Model

Given a dynamic environment, composed of both static and mobile/aerial social entities, the objective is to accompany a determined actor in the environment in a socially acceptable manner, both on task and social navigation perspectives. As introduced before, this problem is approached from an egocentric perspective, entailing a strict restriction on world perception. It is important to remark on mobile social actors in the environment may change their behavior influenced by the robot movement.

As a result, all robot interactions with the different elements of the environment are integrated into four different forces. These forces represent the features that the learning model uses as input to predict the acceleration commands that must be applied to the robot, which are defined, in detail, in the following subsection. The presented research can be applied to any type of robots, nevertheless, for the current article, we will focus on aerial robots.

4. Background

In the present section, we proceed to describe the background methods used in this article, which are the SFM and the AutoRL approach for Robot Navigation.

4.1. Social Force Model

In the late 1950s, pedestrian behaviors were modeled by the first time. At the beginning, these models were concentrated on the dynamics of macroscopic theories, and pedestrian dynamics were studied as fluids [40]. As the development advanced, scientist focused more on microscopic description, here, the motion of each pedestrian is described independently [41].

In contrast, the SFM [8] reproduces pedestrian dynamics using interaction forces. Thus, the human motion behavior can be described as a function depending on the pedestrians' positions and velocities. Furthermore, authors presented in [19] combined the time of collision with the SFM. Nevertheless, these works do not take into account the interactions between robots and people. For the representation of these interactions, we were influenced by the works of [8,19].

Formally, this approach considers each pedestrian p_i in the environment with mass m_{pi} as a particle following the laws of Newtonian mechanics:

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{t} + \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{1}$$

where (x, y) is person's position, (v_x, v_y) is his/her velocity, and (a_x, y_x) is the acceleration.

Assuming that pedestrian attempts to adapt his or her velocity within a *relaxation time* k_i^{-1}, $\boldsymbol{f}_i^{goal}$ is given by:

$$\boldsymbol{f}_i^{goal} = k_i(\boldsymbol{v}_i^0 - \boldsymbol{v}_i) \tag{2}$$

In order to reach a particular desired velocity and direction a relaxation time is required. Moreover, the force $\boldsymbol{F}_i^{int}$ represents the repulsive effects caused by obstacles, pedestrians and robots in the environment. This force is conformed as an addition of forces introduced by people p_j , by static obstacles in the environment o or by a particular robot R.

$$\boldsymbol{F}_i^{int} = \sum_{p_j \in \mathcal{P}} \boldsymbol{f}_{ij} + \sum_{o \in \mathcal{O}} \boldsymbol{f}_{io} + \boldsymbol{f}_{iR} \tag{3}$$

where, $\mathcal{P}$ is the set of pedestrians in the environment, and $\mathcal{O}$ is the set of static obstacles.

Usually, people maintain certain distances from other humans in the environment. Pedestrians feel uncomfortable the closer they are to an unknown person.

$$\boldsymbol{f}_{ij} = Ae^{(d-d_{ij})/B} \frac{\boldsymbol{r}_{i,j}(t)}{d_{i,j}(t)} \tag{4}$$

where the set of parameters {A,B,d} denotes the strength, and the range on interaction force, respectively. $\boldsymbol{r}_{i,j} = \boldsymbol{r}_{p_i} - \boldsymbol{r}_{p_j}$ is the sum of radius of the two pedestrians involved in an interaction. See [3] for further details.

In addition, an obstacle o creates a virtual repulsive effect, as people feel less comfortable the closer an obstacle is navigated, and this can be expressed as:

$$f_{i,o} = \nabla_{r_io} U^0_{io} e^{||r_{io}||/C} \tag{5}$$

where $r_{io} = r_{p_i} - r_{io}$ represents the location of the point of the obstacle o that is closest to pedestrian p_i [42] for a more detailed explanation.

Lastly, people maintain a security distance from robots. Thus, a robot R creates a repulsive effect if the distance to the human is lower than a particular threshold. This effect is expressed as:

$$f_{iR} = A_{iR} e^{(d_r - d_{iR})/B_{iR}} \frac{\boldsymbol{r}_{iR}(t)}{d_{iR}(t)} \left(\lambda_{iR}(1 - \lambda_{iR}) \frac{1 + cos(\phi_{iR})}{2} \right) \tag{6}$$

The described theory is used to build a social robot navigation tasks framework; here, the robot moves naturally in human environments following the SFM, and thus we achieve a higher acceptance from pedestrians.

4.2. Social Force Model with Collision Prediction

The SFM with Collision Prediction, described in [19], is a model that takes into account the velocities of all the agents in an environment and makes a prediction of their future collisions. These future possible collisions and these agent's velocities are used to compute the repulsive forces applied in all the agents.

To estimate the future collisions, the agents are propagated in time with a constant velocity model. Then, for each pair of pedestrians, p, q, the time in which the difference between agent positions, $\boldsymbol{d}_{q,p}$ becomes minimal is computed using this formula:

$$t_{q,p} = \begin{cases} -\dfrac{\boldsymbol{d}^T_{q,p} \cdot \mathbf{v_{q,p}}}{\|\mathbf{v_{q,p}}\|^2} & |\theta_{q,p}| < \pi/4 \\ \infty & |\theta_{q,p}| > \pi/4 \end{cases} \tag{7}$$

where $\mathbf{v_{q,p}}$ is the relative velocity between p and q, and $\theta_{q,p}$ is the angle between $\mathbf{v_{q,p}}$ and $\boldsymbol{d}_{q,p}$.

For each pedestrian p, there is a set of times in which a possible collision can occur, $\{t_{q,p}\}$. To estimate a possible collision, the most important time of the set to be considered is the minimal one, t_p:

$$t_p = min_q\{t_{q,p}\} \tag{8}$$

Using this time estimation , the repulsive force expression applied in agent p due to the agent q is:

$$\mathbf{f}^{\mathbf{int}}_{\mathbf{q,p}}(\{\mathbf{v_{q,p}}\}, \{\mathbf{d_{q,p}}\}, \mathbf{v_p}) = A_q \frac{v_p}{t_p} e^{-d_{q,p}/B_q} \frac{\mathbf{d'_{q,p}}(\mathbf{t_p})}{d'_{q,p}(t_p)} \tag{9}$$

where $\{\mathbf{v_{q,p}}\}$ is the set of relative velocities between p and other agents q. $\{\mathbf{d_{q,p}}\}$ is the set of vectors with all relative distances between p and other agents, p. v_p is the velocity module for p. $\mathbf{d'_{q,p}}(\mathbf{t_p})$ is the relative position of p regarding q, in t = t_p. A_q and B_q are parameters that can be adjusted and $d_{q,p}$ is the Euclidean distance between agents.

This repulsive force can be applied to static obstacles considering zero velocity for each obstacle q. In this model, there is an attractive force, given by Equation (2), to encourage the agent to achieve a goal.

The resultant force for each agent is the sum of the repulsive forces from other agents, from static obstacles and the attractive force to the goal:

$$\mathbf{F_p} = \mathbf{f^{goal}_p} + \sum_{q \in P} \mathbf{f^{int}_{q,p}} + \sum_{o \in O} \mathbf{f^{int}_{o,p}} \tag{10}$$

where P is the set of agents and O is the set of static obstacles. This model is very useful to simulate the pedestrian behavior near the obstacles and to avoid collisions, maintaining the direction toward the goal. However, this model is not so good when faces large static obstacles.

4.3. AutoRL for Robot Navigation

AutoRL [39] is an algorithm that searches the optimal hyperparameters for a parametrized reward and for the neural network in deep RL. For a feed-forward fully connected network, the parameters are the number of hidden layers and the number of neurons for each hidden layer. In [39], these optimal hyperparameters are searched and used through the DDPG algorithm for robot navigation in two navigation tasks: path following (PF) and point-to-point (P2P).

The PF task consists on following a guidance path obtained through the method described in [43]. The P2P task consists on navigating from an initial position to a goal. To find the best hyperparameters, AutoRL maximizes an objective function G that is evaluated for each training with different sets of hyperparameters. This function depends on the task:

$$G_{P2P}(s) = \mathbb{I}(\|s - s_g\| < d_{P2P}) \tag{11}$$

$$G_{PF}(s) = \frac{\sum_{\omega \in \mathcal{P}} \mathbb{I}(\|s - \omega\| < d_{wr})}{\|\mathcal{P}\|} \tag{12}$$

where s is the robot position and $\mathbb{I}$ is the indicator function. For P2P, s_g is the goal position and d_{P2P} is the goal radius. For PF, ω are the way point positions of the guidance path $\mathcal{P}$ and d_{wr} is the way point radius. The objective in P2P is to maximize the probability of reaching the goal during an episode. The objective in PF is to reach as many way points as possible in each episode.

For the P2P task, the parametrized reward proposed in [39] is:

$$R_{\theta_{r_{\mathrm{P2P}}}} = \boldsymbol{\theta}^T_{r_{P2P}}[r_{step}\, r_{goalDist}\, r_{col}\, r_{turn}\, r_{clear}\, r_{goal}] \tag{13}$$

where $\theta_{r_{\mathrm{P2P}}}$ are the reward hyperparameters for the P2P task, r_{step} is the penalty constant at each step in the episode with value 1, $r_{goalDist}$ is the negative Euclidean distance to the goal, r_{col} is 1 in collisions and zero otherwise. r_{tur} is the negative angular speed, r_{clear} is the distance to the closest obstacle and r_{goal} is 1 when the agent reaches the goal and zero in the remaining cases. To know more details of Equations (11)–(13), see the work of [39].

In [44], AutoRL was demonstrated to be able to achieve very good results in other tasks compared to hand-tuned and hyperparameter-tuning methods, such Vizier [45]. However, the time required and the computational cost to apply AutoRL can make it difficult to use in cases where there are not enough computational resources.

To extend the navigation distance in large indoor environments, AutoRL has been implemented as a local planner for a PRM [46] or RRT [47]. The results in real environments show a high performance and robustness to noise.

5. Combining Reinforcement Learning with Social Force Model

The approach described in this section presents a hybrid policy for social robot navigation, which combines a RL model with the SFM to improve social navigation in environments that contain pedestrians and obstacles. Specifically, the SFM version used for this approach is the one described in [19].

Hybrid Model Description

This section explains the details of the hybrid model based on AutoRL work for P2P navigation.

First, the model is trained using the DDPG algorithm with the same optimal hyperparameters, obtained with the AutoRL described in [46] for a differential drive robot model. The noise parameters are the same too. Although the task can be seen as a POMDP, it is very common to approximate it as a Markov Decision Process (MDP) to use deep RL algorithms, such as DDPG where the observations are treated as states. This algorithm uses feed-forward networks for both the actor and the critic. The actor network is the policy, with network parameters θ_a. The actor takes θ_n observations and gives an action:

$$\pi(\mathbf{o};\theta_a) = (a_l, a_\phi) \tag{14}$$

Secondly, in the evaluation phase, the SFM with collision prediction is applied to each point detected using the LiDAR. Only the repulsive force described in Equation (9) is considered. The attractive force in (2) is not considered in the model, because the trained model has learned to efficiently navigate toward the goal when there are not large obstacles close to the robot in the goal's direction. The objective of using the SFM in this work is to use it only when the robot is close to the obstacles to avoid collisions. The repulsive force is almost zero far from the obstacles.

In simulation, the agent's velocities used to calculate the repulsive forces are known if the agents are detected with the robot LiDAR. In a real implementation, these velocities can be provided by an efficient tracker, for example, the ones described in [48,49], that can provide agent velocities in real time using the LiDAR observations.

There is a repulsive force for each LiDAR point, $\mathbf{f_i}$. These set of repulsive forces can be sorted by module from highest to lowest, $\{\mathbf{f_i^{sort}}\}_{i=1}^{n}$. To capture the most important interactions, the five largest forces are considered each time-step to calculate a resultant force:

$$\mathbf{F}_{res} = \sum_{i=1}^{5} \mathbf{f_i^{sort}} \tag{15}$$

To avoid very large forces, the resultant force is normalized and multiplied by the biggest force module, f_1^{sort}, to obtain the final repulsive force:

$$\mathbf{F}_{rep} = f_1^{sort} \frac{\mathbf{F}_{res}}{||\mathbf{F}_{res}||} \tag{16}$$

The module of $\mathbf{F}_{rep}$ represents the nearest obstacle influence, and the force direction represents the best direction to avoid the nearest obstacles around the robot.

The force $\mathbf{F}_{rep}$ is used to compute the robot's velocity. Since we consider in the SFM that the robot has mass = 1, then the robot acceleration is the force, and we use this acceleration to compute the changes of velocity $(\mathbf{\Delta v})_{rep}$ in each time-step. This velocity change can be decomposed in 2 components, $(\mathbf{\Delta v})_{rep} = (\Delta v_l, \Delta v_\phi)$. Δv_l is the component in the current velocity direction, and Δv_ϕ is oriented in the orthogonal direction. These velocity changes or SFM actions are added to the DDPG actions to calculate the actions, which are provided by the hybrid model:

$$a_l' = a_l + \Delta v_l \tag{17}$$

$$a_\phi' = a_\phi + \Delta v_\phi \tag{18}$$

The weights of the SFM and DDPG's actions in the Equations (17) and (18) are the same, although the SFM only acts when the robot is near to obstacles. The SFM action is almost zero if the distance is more than 1 m, but if the robot is very close to the obstacles, the SFM action is much bigger than the DDPG action.

The complete scheme of the hybrid model is shown in Figure 2.

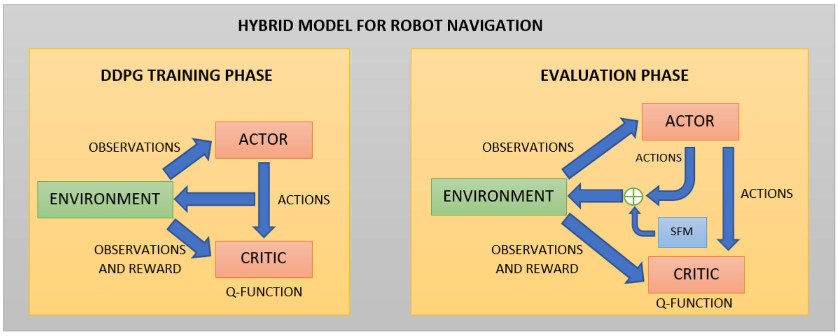


Figure 2. Hybrid Model Scheme. The model is trained using the DDPG algorithm without SFM actions. Once the model is trained in the evaluation phase, the SFM actions are added to move the robot in the environment.

6. Combining Supervised Learning with Social Force Model

The ESFM can be considered as a means to extract a simple representation of the environment useful in social navigation problems. Therefore, given a set of social forces, our goal is to predict—through supervised learning techniques—the velocity vector that will allow the robot to follow the necessary motion to obtain a proactive human-like side-by-side navigation with humans.

For that reason, we present a nonlinear-based approach related to the ESFM. Ahead, we proceed to describe the combination of the supervised learning with the well-known SFM. The proposed system aims to allow human–robot side-by-side social navigation in real environments, concretely, drone navigation, and this approach is based on the extension of the SFM presented in [50].

6.1. Feature Definition

SFM has demonstrated that changes in behavior/intentions of humans can be explained in terms of social forces or social fields. The total force applied to the robot comes from the following components: (i) and (ii) the robot-humans and robot-objects interaction forces, which are repulsive, and (iii) the goal attraction force, which is a predicted position that makes the robot stay closer to the human being accompanied. All these forces are incorporated in the non-linear SFM to the set of input features, which also includes instantaneous drone velocity.

6.1.1. Repulsive Features

First, the static object repulsive feature provides information about the relative location of static objects with respect to the robot, and it also offers information about their proximity. It takes the form of a global force that aggregates all the individual repulsive forces:

$$\mathbf{F_o} = \sum_{o=1}^{O} \mathbf{f_o} \tag{19}$$

$$\mathbf{f_o} = \frac{\mathbf{P_o} - \mathbf{P_R}}{\|\mathbf{P_o} - \mathbf{P_R}\|} A_{R,o} e^{(\frac{d_{R_o} - d_{R,o}}{B_{R,o}})} \tag{20}$$

where O is the set of detected objects; $\mathbf{f_o}$ is defined as a non-linear function of the distance of each detected static objects in the environment; $\mathbf{P_o}$ and $\mathbf{P_R}$ are the positions of the object and the drone, respectively; and d_{R_o} is the distance between the robot and the object. $A_{R,o}$, $B_{R,o}$ and $d_{R,o}$ are fixed parameters; the study of the definition of the parameters is described in [50].

Secondly, pedestrians can be considered as dynamic obstacles; nevertheless, bystanders have their own personal space, and intrusions into this area by external objects may cause discomfort to the person. Thus, we define a feature to model: the repulsion exerted on the drone from pedestrians. The pedestrian repulsive feature is again a force defined similarly to the static object repulsive force,

$$\mathbf{F_h} = \sum_{h=1}^{H} \mathbf{f_h} \tag{21}$$

where H is the set of detected pedestrians and $\mathbf{f_h}$ is defined as

$$\mathbf{f_h} = \frac{\mathbf{P_h} - \mathbf{P_R}}{\|\mathbf{P_h} - \mathbf{P_R}\|} A_{R,h} \frac{v_h}{t_h} e^{\left(\frac{d_{R_h} - d_{R,h}}{B_{R,h}}\right)} \tag{22}$$

$\mathbf{P_h}$ and $\mathbf{P_R}$ are the positions of the pedestrian and the drone, respectively, and d_{R_h} is the distance between the robot and the object. Moreover, the term v_h/t_h is introduced to model the force in such a way that the pedestrian h is able to stop in time t_h [19].

6.1.2. Attractive Features

In order to obtain a flying behavior in which the drone is moving in a side-by-side formation with the human, some notion of the path the human and is following is required. This module is responsible for inferring future human's position.

$$\mathbf{f_g} = \mathbf{P_g} - \mathbf{P_R} \tag{23}$$

where P_R is the drone's position and P_g is the estimated human position for one second into the future. This feature is not expressed as a force, but still provides the necessary information to allow the drone to fly next to the main human.

Moreover, a new feature is defined, it provides information about the current position of the human. This feature combined with the goal feature encodes information about the current position of the human and the expected direction of movement. The human feature is defined similarly to the goal feature, as a vector from the drone position P_R to the main human position P_h.

$$\mathbf{f_c} = \mathbf{P_c} - \mathbf{P_R} \tag{24}$$

As can be seen in Figure 3, we plotted all the forces used as features in our supervised learning approaches, moreover, we defined an additional destination to the robot approach. The robot aims to the target person in order to accompany him/her.

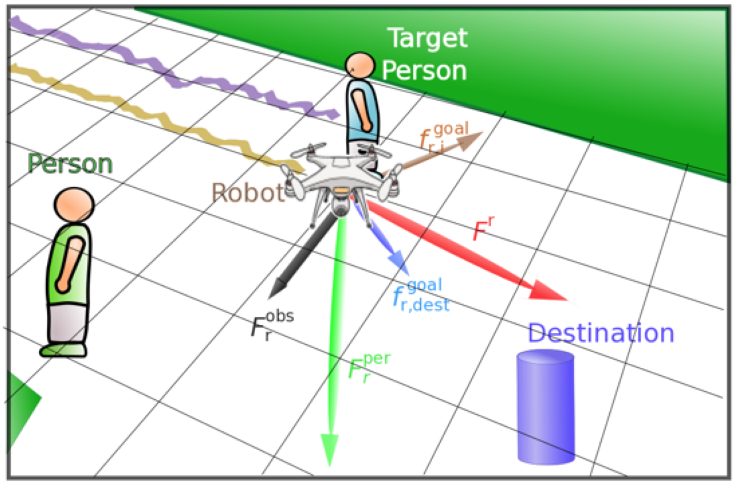


Figure 3. SFM, forces applied to the drone while accompanying a person.

6.2. Non-Linear Regressor

Social Navigation based on Artificial Potential Fields was presented previously [50], and these works extract forces from the elements in the environment, and combine them linearly obtaining a resulting force. This force is then applied to the robot, which determines its acceleration. The aim of this research is to substitute the linear approach by a non-linear model learned from data, the data is composed by recordings of robot's trajectories teleoperated by an expert user. We desire to train a dense neural network to imitate the flying controls of a human expert. By learning the model from data, we also avoid the tuning of the presented parameters as in previous approaches.

The non-linear model used to learn the expert flying policy is a fully connected five-layer dense neural network, where the first hidden layer contains 18 neurons, and the second hidden layer is made of 23 neurons. The input layer has 15 neurons, one for each input dimension, and the output layer has three neurons, one for each component of a linear acceleration in a 3D space. This adds up to a total of 797 trainable parameters.

The probability values defined for the network are, from input to output layer, 0.1 and 0.1. Regarding the parameters used during the training phase of the networks, the mean squared error was the chosen loss function and Adam was used as the gradient descend optimizer, and it was trained with batches of size 512 [51].

Validation data, consisting of an episode of each situation, is extracted and separated from the rest. All windows from each episode from the rest of the dataset are then pooled together, scaled, shuffled and split into training and test datasets. Then, the model is trained and tested. Moreover, validation is done in two phases. First, we compare the Root Mean Square Error (RMSE) of the estimated force with the results obtained following previous works [52]. Secondly, we assess the performance of our model by comparing its RMSE with previous works in specific environments using the data from a specific episode of the validation data.

FreeGLUT toolkit, and the joysticks used for user control are PlayStation 3 DualShock 3 Wireless Controllers.

On the knowledge transfer to real-world applicability, it has been chosen to implement over ROS (Robot Operating System). Two steps were taken in this process, first a model testing phase over a simulated *Gazebo* world and, secondly, an execution phase on a *Drone* base. On both, all world data were collected through the processing of two laser inputs *-Hokuyo UTM-30LX Scanning Laser Rangefinder-*.

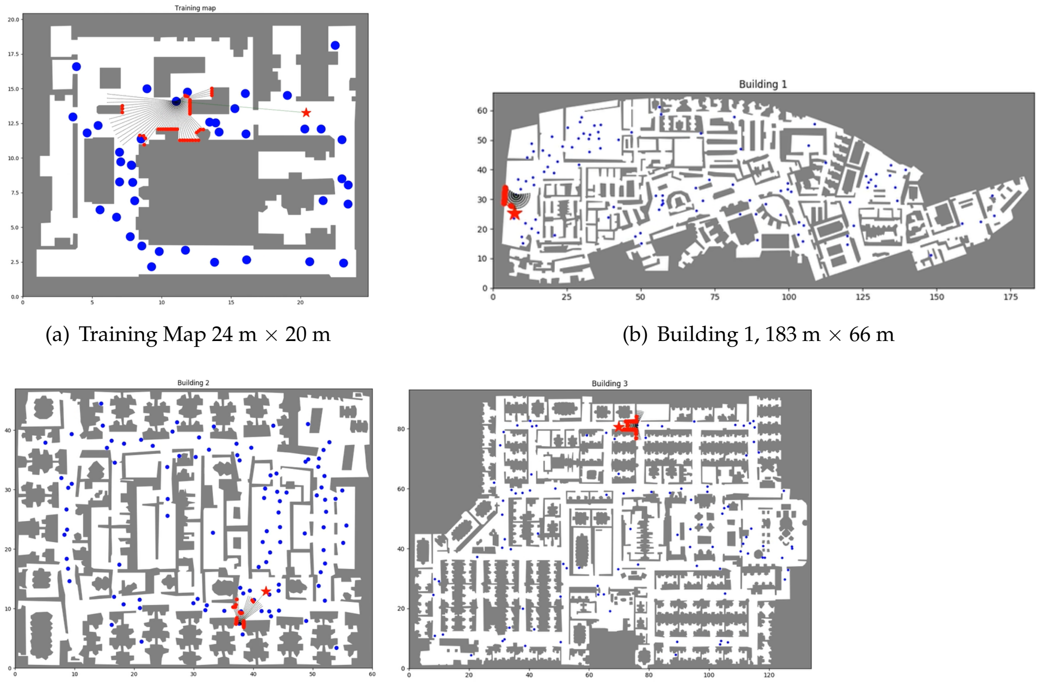
6.3. Quantitative Metric of Performance

To properly evaluate the learned control policies, the Non-linear Aerial SFM defines a new quantitative metric of performance inspired by the metric presented in [23], which is based on the idea of proxemics [6].The new quantitative metric defines three distances, d_A, d_B and d_C, which define three concentric spherical volumes around a human. Please refer to [23] to check the definition of the used metrics.

7. Experiments: Social Robot Navigation

In this section, we explain the simulation and real-life experiments in social robot navigation, using the combination of a Reinforcement Learning technique with the SFM.

For the simulations, we used the maps shown in Figure 4: one of them for training and the rest for evaluation. The training and the evaluation described in this section were performed using the OP-DDPG described in this article with the optimized parameters of the AutoRL work [39].



(a) Training Map 24 m × 20 m

(b) Building 1, 183 m × 66 m

(c) Building 2, 60 m × 47 m

(d) Building 3, 134 m × 93 m

Figure 4. Floor maps. These are the floor maps for training and evaluation with their sizes in meters.

7.1. Metrics

We have used the following three metrics to evaluate method in the maps:

- **Success Rate (SR)**: Percent of episodes in which the robot achieves the goal without collision. It is a measure of the robot behavior to move closer to the goal and gives an idea of the overall performance. When the robot reaches the goal, the episode is finalized.
- **Collision Rate (CR)**: Percent of episodes that ends in a collision. A collision marks the end of an episode.
- **Timeout Rate (TOUT)**: Percent of episodes in which the episode time limit is reached without a collision or success. This metric gives information in complex environments or very uncommon situations for the robot.

During the execution, other metrics are calculated at each episode as for example, the number of steps and the goal distance to obtain a more local description of each episode.

7.2. Evaluation Results

For all the evaluations, we used the same trained OP-DDPG, which was trained with the map of Figure 4a. The hybrid model is introduced in the evaluation phase, where the combination of RL and SFM is applied.

For the evaluation, we used two different types of distances: the Euclidean distance (ED) and the approximate path distance (PD). The main reason to use the approximate path distance is that it provides more precise results of the policy performance when the path to the goal is obtained using PRM [46] or RRT [47] planners. When it is used the ED, the goal is sampled between 5 and 10 m; and when the PD is used, the goal is obtained using a RRT's path whose length is less than 10 m. The results shown in Table 1, obtained with 5–10 m of Euclidean distance and path distance, show that the hybrid model can increase the success rate in the three evaluation maps without moving obstacles. All the evaluation results were obtained computing 100 episodes as in [39].

The results obtained using Euclidean distance were worse than the obtained ones in [39] as the implementations are different. This means that, when the AutoRL's optimal hyperparameters are applied in the OP-DDPG model, suboptimal results were obtained.

In this work, the results are focused in the advantages of applying the SFM, regardless the AutoRL optimization.

Table 1. OP-DDPG and Hybrid Model comparison in static environments for Euclidean and path distance cases.

Environment	OP-DDPG			Hybrid Model		
	SR	CR	TOUT	SR	CR	TOUT
Building 1 (ED)	60	34	6	66	11	23
Building 2 (ED)	50	46	4	74	15	11
Building 3 (ED)	65	35	0	81	11	8
Building 1 (PD)	75	11	14	88	3	9
Building 2 (PD)	87	13	0	96	1	3
Building 3 (PD)	87	12	1	97	1	2

The evaluation with moving agents was also performed using the Euclidean and path distance. There are two cases that represent a low-crowded environment with 20 moving agents (pedestrians) and a high-crowded scene with 100 moving agents.

The results in Table 2 show that the number of moving agents does not greatly affect the hybrid model performance. Only in environments with a very high density of moving obstacles, such as Building 2 with 100 obstacles, is the success rate significantly reduced. The differences in the success rate between the Euclidean and path distance cases increase in the high-crowded cases because the number of encounters with moving agents is augmented.

Table 2. OP-DDPG and Hybrid Model comparison for Euclidean and path distance cases in dynamic environments with 20 and 100 moving agents.

Environment-Agents	OP-DDPG			Hybrid Model		
	SR	CR	TOUT	SR	CR	TOUT
Building 1 ED-20	58	37	5	67	22	11
Building 2 ED-20	47	44	9	66	22	12
Building 3 ED-20	65	34	1	77	18	5
Building 1 PD-20	74	11	15	84	4	12
Building 2 PD-20	80	20	0	89	3	8
Building 3 PD-20	84	16	0	88	2	10
Building 1 ED-100	52	44	4	61	27	12
Building 2 ED-100	39	50	11	53	37	10
Building 3 ED-100	57	41	2	71	22	7
Building 1 PD-100	67	16	17	84	3	13
Building 2 PD-100	61	39	0	76	16	8
Building 3 PD-100	83	17	0	92	0	8

Another evaluation (refer to Figure 5) was performed increasing the goal path distance in the range from 10 to 15 m. In this case, the hybrid model shows less degradation in the success rate compared with the OP-DDPG model.

Robustness to noise was evaluated in the maps with 20 moving agents. In these cases, such as in the AutoRL work, Gaussian noise is considered, $\mathcal{N}(0, \sigma_{lidar})$, for four different standard deviations between 0.1 and 0.8 m. The results obtained show that the OP-DDPG model and the hybrid model are very robust to noise. The success rate decreases in both models less than 5%.

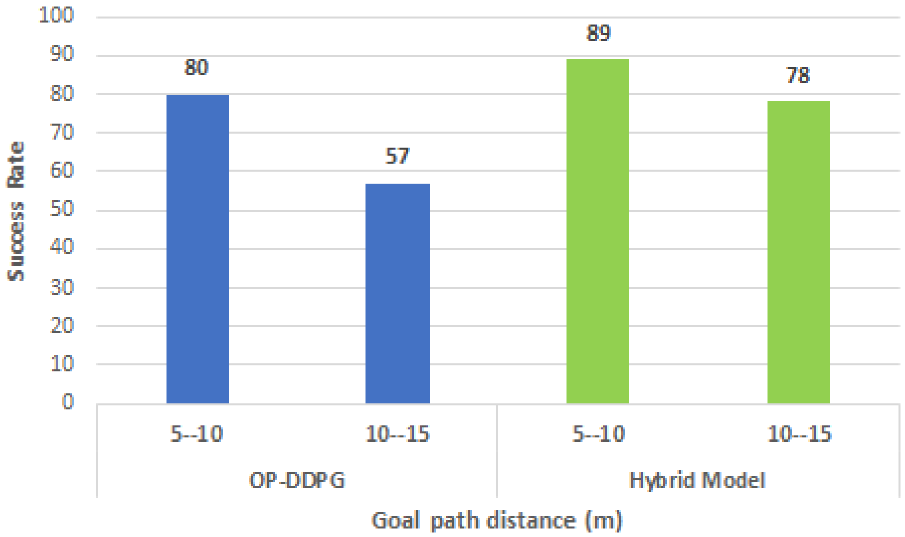


Figure 5. Success rate variation with distance in Building 2. The graphic shows less rate decay with path distance in the hybrid model (green bars) with 20 moving agents.

The difference between the OP-DDPG model and the hybrid model lies mostly in the collision rate reduction. On the contrary, the timeout rate can increase, because a percentage of the episodes that lead to collisions in OP-DDPG model become episodes that end in timeouts in the hybrid model. In the hybrid model, the repulsive forces have only influence when the robot is close to the obstacles, due to the exponential decay of those forces. For this reason, the majority of actions are taken by the OP-DDPG policy. This causes the hybrid model to avoid collisions, but the repulsive force of SFM is not enough to escape from complex obstacles if the OP-DDPG's trained policy has not learned to avoid those situations properly.

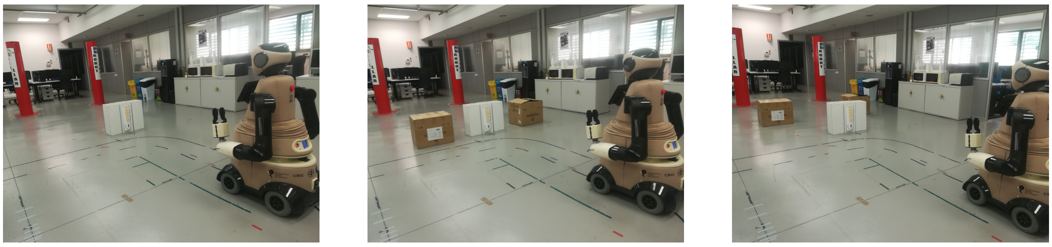
The timeout episodes are associate to large scale local minima, for example, a room with a small exit where the robot moves in a loop. The timeouts give information about difficult situations that have not been learned, for example when the robot has to move in the big open spaces in the map building 1. These spaces have not been observed in the training map, leading to worse behavior to reach the goal. That is the reason why timeouts are larger in building 1. Despite timeout increasing cases, the overall performance is better with the hybrid model.

7.3. Real-Life Experiments

Real indoor experiments were performed with the IVO robot owned by the Institut de Robòtica i Informàtica Industrial [53]. IVO is an urban land-based robot designed to perform navigation tasks that involve object manipulation for grasping and human–robot interaction. The robot is a modified version of the Tiago robot (https://pal-robotics.com/es/robots/tiago/). In the experiments, for the navigation task, IVO uses a platform with four wheels, a 3D LiDAR where information is filtered to obtain the same observation size as is done in simulations, and a RealSense stereo camera to detect holes and ramps. The implementation for the real experiments was implemented in ROS Melodic.

The real-life experiments were evaluated in three scenarios with static obstacles (see Figure 6). One of the scenarios was used with people to evaluate the robot behavior in case of an unexpected occlusion and how the robot can anticipate a possible collision.

The environment map is used in RVIZ to see the LiDAR lectures, which are used to track people and detect the static obstacles (see Figure 7).



(a) 1 centered obstacle

(b) 3 aligned obstacles

(c) 3 obstacles in V-configuration

Figure 6. Real-life environments for experiments. In all the cases the robot goal is between the red columns.

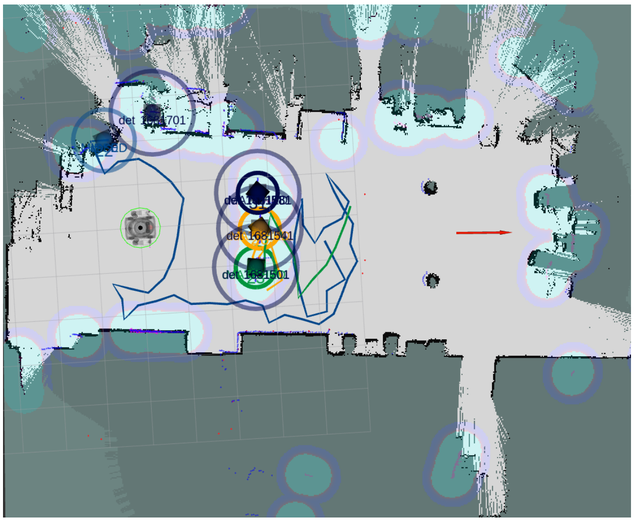
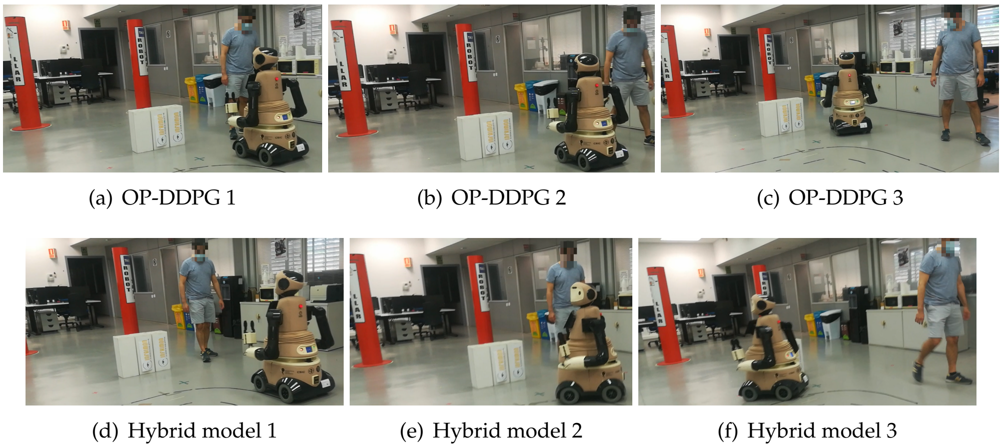


Figure 7. RVIZ representation. In this real time representation we can see the LiDAR observations (small red points), the tracks for moving obstacles or people (big circles) and the goal position (red arrow).

The OP-DDPG model and the hybrid model were tested in the three scenarios.

- **With one centered obstacle:** In this simple case, the two models (OP-DDPG and hybrid model) avoid the obstacle, although the hybrid model generates a trajectory far off the obstacle with lower velocities, when the robot is close to the obstacle.
- **With three aligned obstacles:** This case is probably the most complex one for the robot, due to the confusing space between the obstacles. The robot with the OP-DDPG model attempts to navigate between the static obstacles or navigate in the correct direction, but finally it touches one of the static obstacles. Using the hybrid model, the robot avoids to enter in the reduced space between the static obstacles and navigates toward the left or right space to avoid the obstacles and achieve the goal without collisions.
- **With three obstacles in V-configuration:** In this case, the robot with the OP-DDPG model sometimes touches one of the static obstacles. The hybrid model achieves the goal without collisions, but it has to reduce its velocity in the space between the three obstacles.
- **With one centered obstacle and a person:** In this case, a person walks in the opposite direction of the robot, starting from the goal, toward the side initially chosen by the robot to avoid the centered obstacle. When the robot uses the OP-DDPG model, it suddenly detects the person and slightly reduces its velocity. The person has to change its direction to let the robot pass. Using the hybrid model, when the robot detects the person several meters before, the robot changes its direction choosing the other side to avoid collisions and achieves the goal as can be seen in Figure 8.


(a) OP-DDPG 1

(b) OP-DDPG 2

(c) OP-DDPG 3

(d) Hybrid model 1

(e) Hybrid model 2

(f) Hybrid model 3

Figure 8. Experiment with a person. From 1 to 3 shows the robot behavior when the robot uses the OP-DDPG model and the hybrid model. In the first case, the person has to adapt to the robot motion, while, in the second case, it is the robot who adapts to the person.

The quantitative results obtained in the three environments for the success rate and the navigation time (NT) are shown in Table 3. The results are the average values obtained during 10 navigation episodes where the goal distance is 5 m. The hybrid model increases the success rate, but the NT increases as well because the robot reduces its velocity and sometimes oscillates when an obstacle is very close.

Table 3. OP-DDPG and Hybrid Model comparison for real-life experiments in the three environments.

Environment	OP-DDPG		Hybrid Model	
	SR	NT(s)	SR	NT(s)
1 centered obstacle	90	18.40	100	29.80
3 aligned obstacle	20	34.50	80	40.80
3 obstacles in V-configuration	50	17.85	100	36.20
1 centered obstacle and 1 person	80	28.45	100	29.01

7.4. Implementation Details

In this subsection, the parameters and features used in our implementation are detailed. All the hyperparameters used in the OP-DDPG model and the hybrid model were taken from [46]:

- **Network hyperparameters**: Critic joint, critic observation and actor layer widths are $(607, 242) \times (84) \times (241, 12, 20)$ with ReLU as the activation function.
- **Reward hyperparameters**: As they are ordered in Equation (3), the reward hyperparameters, $\boldsymbol{\theta}^T_{r_{P2P}}$, are $(-0.43, 0.38, -57.90, 0.415, 0.67, 62.0)$.

The number of time-step observations taken for training and evaluation at each step is $\theta_n = 1$. Noise parameters are the described ones in [46]. All results were obtained with the same trained model during two million steps and 500 steps as the maximum episode length.

The implementation was performed in PyTorch using the Gym library for RL. For the simulation environment, the Shapely library was used to manage the floor maps, the robot, and the agent models. The Huber loss was used as the critic loss and Adam as the optimizer. The batch size is 512, and the replay buffer has a capacity of 0.5 million. The remaining training parameters are described in [46]. The moving agents were simulated as circles with a radius of 0.3 m. Agents move to a goal using the SFM with collision prediction [19] to avoid static and moving obstacles in the environment. When an agent goal is reached, a new goal is sampled.

For the real-life experiments, the implementation was done in ROS Melodic, and we used two nodes for the OP-DDPG model: A node is used to filter the frontal LiDAR observations of IVO and the second node is used to calculate the OP-DDPG actions. For the hybrid model, three additional nodes were used: the first node uses the tracker explained in [49] to obtain the pedestrians' velocities, the second node calculates the SFM forces, and the last node combines the OP-DDPG action with the SFM action. All the nodes were implemented in C++, except the node that computes the OP-DDPG action, that was implemented in Python, using PyTorch and the Gym library. To obtain better results in the real-life experiments, a new model was trained with a radius of 0.45 m for the robot.

8. Experiment: UAV Human Accompaniment

In this section, we describe the experimentation process to demonstrate the proper functionality of a UAV accompanying a pedestrian.

8.1. Database Generation

Data used for this work was gathered in a simulated environment. The simulation was developed focusing on intuitive controls and visualization, both characteristics tested through a qualitative questionnaire answered by the participants in a first test run.

Therefore, in order to teach our model to behave similarly to a human, we require a significant number of expert demonstrations that will be later used to build a dataset by extracting the relevant features. This dataset will be then used to train the nonlinear model, and hopefully the trained model will fulfill our requirements. Thus, the first step to learn the aerial control policy is to obtain trajectories through human expert demonstrations. The software used to build and run the simulator is FreeGLUT (http://freeglut.sourceforge.net).

From the created expert trajectories we extracted the following features: (i) the companion attractive force, (ii) the goal attractive force, (iii) the pedestrian repulsive force, (iv) the drone velocity, and (v) the y component of the static object repulsive force. The two first attractive forces are required to achieve a side-by-side formation with the companion. In order to respect the pedestrian's personal space the pedestrian repulsive force is necessary. The system also needs the drone velocity to correct motion control. To avoid obstacles, y feature is used to keep a proper distance to the floor.

As a result, we obtained a dataset composed by matrices **X** and **Y**. **X** matrix consist of the feature vectors of samples from the expert recorded trajectories. **Y** matrix are the target vectors of the samples from the expert trajectories. In Figure 9, some environments to train our model are plotted.

$$\mathbf{y_i} = \begin{bmatrix} a_{R_x} & a_{R_y} & a_{R_z} \end{bmatrix} \tag{25}$$

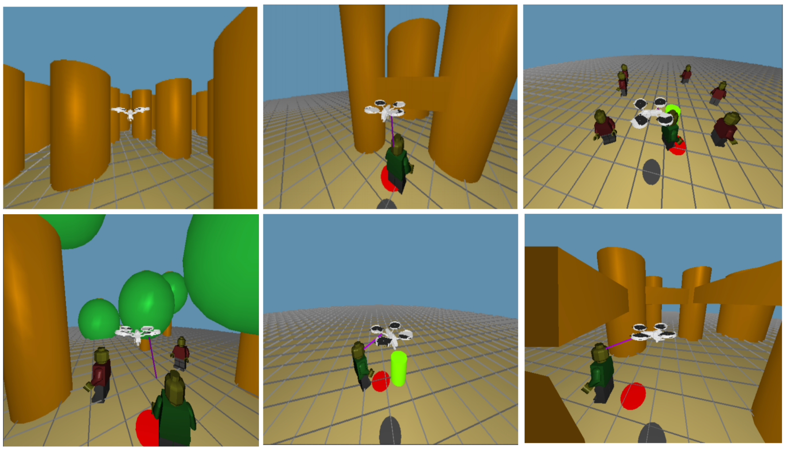


Figure 9. Simulated environments: Frames from the simulator's view for different environments.

8.2. Simulation Results

The learned flying policies were tested on different test environments. To compute the performance of a model in a test environment, the test environment is simulated, and the trained model is used to compute the acceleration command that must be applied to the drone. The performance metric is computed in each cycle of a test simulation, and thus, the final performance of a certain model in a certain environment is obtained by averaging the instantaneous measurements of performance along a test simulation run on that environment.

In order to evaluate the model through an episode, we use a quantitative metric of performance presented in [50], which is based on the idea of proxemics [6]. We measure the distances of the controlled agent to the human companion and to all other pedestrians and static obstacles at each frame. Then we draw three circles of increasing radius (0.5, 1.5 and 2 m) around the center of the human companion. The smallest represents the area occupied by the robot, the second would be the region we are interested to be in and the third concentric area represents the limits of the zone we want the automated companion to be, being these the limits defined in [6] as the intimate/personal, social and public distances.

We reproduce this evaluation five times for every test environment, Figure 10. We measure the area below the curve for each of them, calculate the average proxemics score per frame through the episode and extract the mean proxemics score per frame and their standard deviation throughout the 10 episodes performed Table 4.

Table 4. Nonlinear model performance against the previous approach, linear model. Evaluation done through the proxemics social distance metric. Bold shows the best value.

	Cluttered Env.		**Crowded Env.**		**Cluttered & Crowded Env.**	
Models	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
Linear	0.480	0.066	0.515	0.097	0.533	0.084
Non-Linear	**0.616**	0.119	**0.586**	0.0123	0.542	0.137

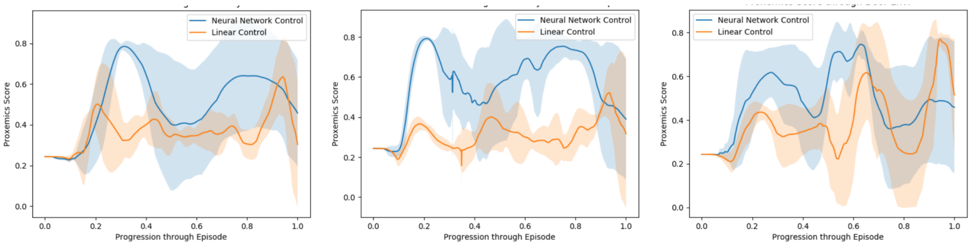


Figure 10. Evaluation Simulation. Evaluation of the model through proxemics social distances as presented in Section 6. From left to right: Cluttered environment, crowded scenarios and cluttered and crowded scenarios.

First, as we can observe in the results, the non-linear model score in the simulated environments is clearly higher in comparison to the Linear model. While the Linear model behaves in a reactive way and ends up following the human companion, the Non-Linear is able to learn from human behavior data, recognize such situations, accelerate and reach the human's side while in motion.

Secondly, it is shown in Table 4 how our method performs better in average than the Linear Model, especially when pedestrians are included in the environment. Throughout every training episode, among the present forces, we were able to register the companion's repulsive and attractive forces at every frame. This means a great amount of data regarding how to behave around other pedestrians and react to their repulsive forces, which allows the model to better behave in front of other agents.

Finally, these results issuing from successfully navigating through different environments, demonstrate that our method is capable of following a companion operated by an expert, while navigating in environments with pedestrians and static obstacles. These results also demonstrate that our model is able to perform this activity at a human level when it comes to proxemics rules and clearly outperforms the previous method.

8.3. Real-Life Experiments

During the real experiments, the learned policy was tested on a real-life environment, which requires the use of a real quadrotor. The selected drone to fill this role is the AR.Drone 2.0 owned by the Institut de Robòtica i Informàtica Industrial [53]. We also made use of the Optitrack Motion Capture system, created by NaturalPoint Inc, which provides all the necessary information about absolute positions of all the elements in the environment that are properly marked. Everything was combined with ROS, and the controller of the drone was implemented as ROS node. With the information provided by Optitrack, the controller node can build the inputs of the model to obtain the predicted acceleration commands for the drone. The drawback of using Optitrack to accurately locate all the elements in the environment is that the working area is limited to 5 × 5 m, which imposes severe constraints on the movement of the drone and the main human, see Figure 11.

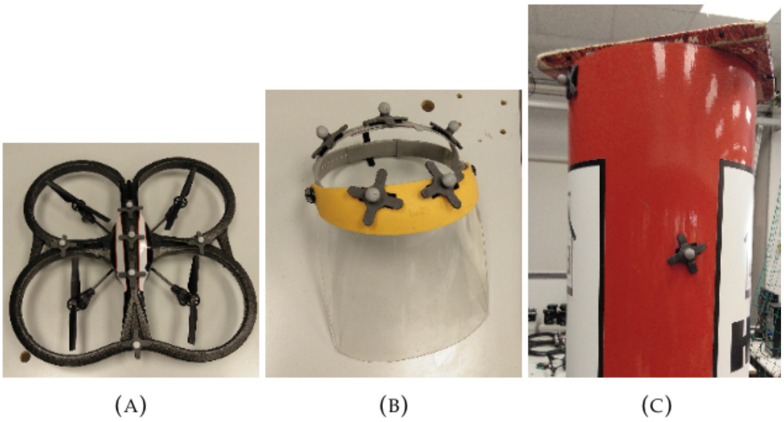


Figure 11. Hardware involved: SMarkers applied to (**A**) the AR.Drone, (**B**) the main human protective gear and (**C**) the obstacle.

First, we built a Optitrack node to detect and localize each element of the environment: the volunteer protective gear, the autonomous Drone and the obstacle of the scene; see Figure 11. With this information, the Optitrack can accurately compute the position of each required element. Figure 12 provides images of the real-life experiments conducted in our laboratory.

When we run several experiments in a simple scenario with the accompanied person and the drone, the behavior of the drone observed in the real-life experiments was correct. During the experiments, we observed that the drone was able to approach the human, and once it reaches a certain distance to the human, it stops and maintains the position in the air. If the human starts moving, the drone is able to follow the person while keeping a safe distance.

In the second set of experiments, a pillar obstacle was introduced into the scenario. We observed that the drone was still able to follow the main human maintaining a safe distance, and additionally it successfully avoided the pillar if it was found in the drone's way. It is important to mention, however, that, due to the limitations of the experiment setup, it was not possible to fully replicate the simulated results in reality, and we had to run simplified experiments. Figure 12 shows the real-life experiments where the drone moved and was able to accompany a person.

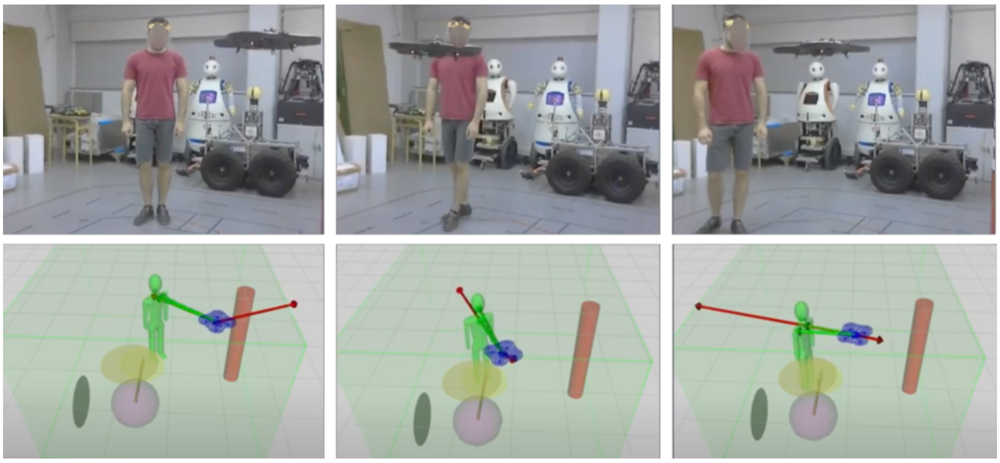


Figure 12. Real-life experiments: (**Top**): Validation of the model in a real-world environment. (**Bottom**): Gazebo simulation replicating the conditions found in the laboratory. The drone, the main human and an obstacle are the only elements in the environment, which is also limited by the green volume.

Finally, in Figure 13, we present the proxemics performance results of the real-life experiments. As in the previous section, we measure the area below the curve for each experiment, and we calculate the average proxemics score per frame through different episodes performed. If there are obstacles on the environment the performance decreases; here, the robot cannot achieve high values as the object creates repulsive forces that makes difficult to approach the accompanied person. In contrast, we obtain values over 0.5 if the there are no obstacles.

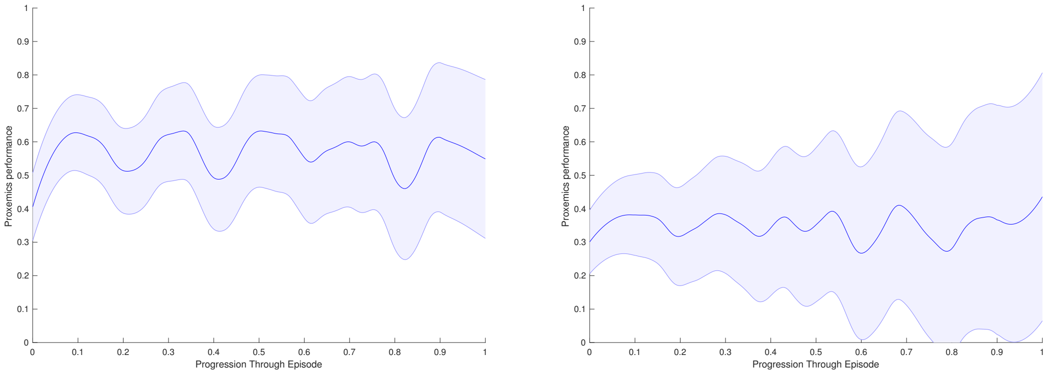


Figure 13. Real-life experiment with a person: Evaluation Results of the model through proxemics social distances as presented in Section 6. (**Left**): Environment with no obstacles. (**Right**): Environment with one obstacle.

9. Conclusions

The overall goal of the present work is to demonstrate that the combination of learning techniques and Social Force Model, achieve good human-aware (social) navigation trajectories in complex scenarios, where there are environmental constraints (buildings, static objects, etc.), pedestrians and other mobile vehicles. This work presents two new techniques that combine machine learning and SFM: (i) social robot navigation using deep RL and SFM; and (ii) accompaniment of a pedestrian using supervised learning and 3D SFM. The first method was tested with a ground robot and the second one with an aerial robot.

Sometimes, in deep RL, it is difficult to obtain a good performance in complex situations that require a great deal of parameter tuning, reward shaping and other methods that do not guarantee an improvement in the results. The use of SFM was demonstrated to be very useful to simulate people's behavior in many situations. The results obtained in simulations and experiments with the IVO robot demonstrate that the SFM is a useful technique to improve RL for robot navigation.

The ability for the robot to avoid obstacles was increased in all cases when the SFM was added during simulations and generates safer policies in real-life environments with people. The hybrid model provides a robust policy against noise in simulations and real experiments with very little information about the environment. Moreover, regarding the teaching model for an aerial robot, the learning by demonstration architecture obtained in simulated environments by an expert user can be tested in the same type of simulated environments, and the trained architecture provided good results when it was applied on a real drone in real-life experiments.

This research reveals that it is possible to learn flying behaviors by the use of recorded aerial trajectories performed by an expert pilot. Furthermore, we may resolve that the system learns better polices with the use of simple scenarios instead of complex environments. Thus, our drone was capable of learning a control policy and navigating close to a person in the working environment. Finally, the authors believe that the combination of SFM with different machine learning techniques enhance the performance of the presented frameworks, achieving a human-aware (social) navigation of both ground robots and aerial robots.

Author Contributions: Ó.G. and A.G. has contributed equally to this work. Conceptualization Ó.G., A.G. and A.S.; methodology, Ó.G., A.G. and A.S.; software, Ó.G., and A.G.; validation, Ó.G. and A.G.; formal analysis, Ó.G., A.G. and A.S.; investigation, Ó.G., A.G. and A.S.; resources, Ó.G., A.G. and A.S.; data curation, Ó.G., A.G. and A.S.; writing—original draft preparation, Ó.G., A.G. and A.S.; writing—review and editing, Ó.G., A.G. and A.S.; visualization, Ó.G., A.G. and A.S.; supervision, A.S.; project administration, A.S.; funding acquisition, A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the grant MDM-2016-0656 funded by MCIN/AEI/10.13039/501100011033, the grant ROCOTRANSP PID2019-106702RB-C21 funded by MCIN/AEI/10.13039/501100011033 and the grant CANOPIES H2020-ICT-2020-2-101016906 funded by the European Union.

Institutional Review Board Statement: This research work was approved by the Universitat Politecnica de Catalunya Ethical Committee with protocol code 2021-11.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Urdiales, C.; Peula, J.M.; Fdez-Carmona, M.; Barrué, C.; Pérez, E.J.; Sánchez-Tato, I.; Del Toro, J.; Galluppi, F.; Cortés, U.; Annichiaricco, R.; et al. A new multi-criteria optimization strategy for shared control in wheelchair assisted navigation. *Auton. Robot.* **2011**, *30*, 179–197. [CrossRef]
2. Dalmasso, M.; Garrell, A.; Dominguez, J.; Jimenez, P.; Sanfeliu, A. Human-Robot Collaborative Multi-Agent Path Planning using Monte Carlo Tree Search and Social Reward Sources. In Proceedings of the ICRA2021 IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May–5 June 2021; pp. 1–6.
3. Repiso, E.; Garrell, A.; Sanfeliu, A. Adaptive side-by-side social robot navigation to approach and interact with people. *Int. J. Soc. Robot.* **2020**, *12*, 909–930. [CrossRef]
4. Kuderer, M.; Burgard, W. An approach to socially compliant leader following for mobile robots. In Proceedings of the International Conference on Social Robotics, Sydney, NSW, Australia, 27–29 October 2014; pp. 239–248.
5. Helbing, D.; Farkas, I.J.; Molnar, P.; Vicsek, T. Simulation of pedestrian crowds in normal and evacuation situations. *Pedestr. Evacuation Dyn.* **2002**, *21*, 21–58.
6. Hall, E.T. Hidden Dimension. In *TRANS-ACTION 4 (2)*; Christian Publications: Missisauga, AY, Canada, 1966; p. 50.
7. Kendon, A. *Conducting Interaction: Patterns of Behavior in Focused Encounters*; CUP Archive: Cambridge, UK 1990; Volume 7.
8. Helbing, D.; Molnar, P. Social force model for pedestrian dynamics. *Phys. Rev. E* **1995**, *51*, 4282–4286. [CrossRef]
9. Walters, M.L.; Dautenhahn, K.; Te Boekhorst, R.; Koay, K.L.; Kaouri, C.; Woods, S.; Nehaniv, C.; Lee, D.; Werry, I. The influence of subjects' personality traits on personal spatial zones in a human–robot interaction experiment. In Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, Nashville, TN, USA, 13–15 August 2005; pp. 347–352.

10. Syrdal, D.S.; Koay, K.L.; Walters, M.L.; Dautenhahn, K. A personalized robot companion?-The role of individual differences on spatial preferences in HRI scenarios. In Proceedings of the 16th IEEE International Symposium on Robot and Human Interactive Communication, Jeju, Korea, 26–29 August 2007; pp. 1143–1148.
11. Dautenhahn, K.; Walters, M.; Woods, S.; Koay, K.L.; Nehaniv, C.L.; Sisbot, A.; Alami, R.; Siméon, T. How may I serve you?: A robot companion approaching a seated person in a helping context. In Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction, Salt Lak,UT, USA, 2–3 March 2006; pp. 172–179.
12. Woods, S.; Walters, M.; Koay, K.L.; Dautenhahn, K. Comparing human robot interaction scenarios using live and video based methods: Toward a novel methodological approach. In Proceedings of the 9th IEEE International Workshop on Advanced Motion Control, Istanbul, Turkey, 27–29 March 2006; pp. 750–755.
13. Garrell, A.; Sanfeliu, A. Cooperative social robots to accompany groups of people. *Int. J. Robot. Res.* **2012**, *31*, 1675–1701. [CrossRef]
14. Repiso, E.; Garrell, A.; Sanfeliu, A. On-line adaptive side-by-side human robot companion to approach a moving person to interact. In *Robot 2017: Third Iberian Robotics Conference*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 113–125.
15. Ferrer, G.; Garrell, A.; Herrero, F.; Sanfeliu, A. Robot social-aware navigation framework to accompany people walking side-by-side. *Auton. Robot.* **2016**, *41*, 775–793.
16. Huang, W.H.; Fajen, B.R.; Fink, J.R.; Warren, W.H. Visual navigation and obstacle avoidance using a steering potential function. *Robot. Auton. Syst.* **2006**, *54*, 288–299. [CrossRef]
17. Costa, M. Interpersonal distances in group walking. *J. Nonverbal Behav.* **2010**, *34*, 15–26. [CrossRef]
18. Moussaïd, M.; Perozo, N.; Garnier, S.; Helbing, D.; Theraulaz, G. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS ONE* **2010**, *5*, e10047. [CrossRef] [PubMed]
19. Zanlungo, F.; Ikeda, T.; Kanda, T. Social force model with explicit collision prediction. *EPL (Europhys. Lett.)* **2011**, *93*, 68005. [CrossRef]
20. Morales, Y.; Kanda, T.; Hagita, N. Walking together: Side-by-side walking model for an interacting robot. *J. Hum.-Robot. Interact.* **2014**, *3*, 50–73. [CrossRef]
21. The, V.N.; Jayawardena, C. A decision making model for optimizing social relationship for side-by-side robotic wheelchairs in active mode. In Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob), Singapore, 6–29 June 2016; pp. 735–740.
22. Nguyen, VT.; Jayawardena, C.; Ardekani, I. A navigation model for side-by-side robotic wheelchairs for optimizing social comfort in crossing situations. *Robot. Auton. Syst.* **2018**, *100*, 27–40. [CrossRef]
23. Garrell, A.; Villamizar, M.; Moreno-Noguer, F.; Sanfeliu, A. Teaching Robot's Proactive Behavior Using Human Assistance. *Int. J. Soc. Robot.* **2017**, *2*, 231–249. [CrossRef]
24. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 961–971. [CrossRef]
25. Hasan, I.; Setti, F.; Tsesmelis, T.; Del Bue, A.; Galasso, F.; Cristani, M. MX-LSTM: Mixing tracklets and vislets to jointly forecast trajectories and head poses. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 8–23 June 2018; pp. 6067–6076.
26. Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S.; Alahi, A. Social GAN: Socially acceptable trajectories with generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2255–2264.
27. Sadeghian, A.; Kosaraju, V.; Sadeghian, A.; Hirose, N.; Rezatofighi, H.; Savarese, S. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA , 15–20 June 2019; pp. 1349–1358.
28. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
29. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
30. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
31. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
32. Shi, H.; Shi, L.; Xu, M.; Hwang, K.S. End-to-end navigation strategy with deep reinforcement learning for mobile robots. *IEEE Trans. Ind. Inform.* **2019**, *16*, 2393–2402. [CrossRef]
33. Hirose, N.; Xia, F.; Martín-Martín, R.; Sadeghian, A.; Savarese, S. Deep visual MPC-policy learning for navigation. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3184–3191. [CrossRef]
34. Chen, C.; Liu, Y.; Kreiss, S.; Alahi, A. Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning. In Proceedings of the 2019 International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 6015–6022. [CrossRef]

35. Everett, M.; Chen, Y.F.; How, J.P. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3052–3059.
36. Van Den Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal n-body collision avoidance. In *Robotics Research*; Springer Tracts in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2011; pp. 3–19.
37. Gil, Ó.; Sanfeliu, A. Effects of a Social Force Model reward in Robot Navigation based on Deep Reinforcement Learning. In *Robot 2019: Fourth Iberian Robotics Conference*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 213–224.
38. Sathyamoorthy, A.J.; Liang, J.; Patel, U.; Guan, T.; Chandra, R.; Manocha, D. Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 11345–11352.
39. Chiang, H.T.L.; Faust, A.; Fiser, M.; Francis, A. Learning navigation behaviors end-to-end with autorl. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2007–2014. [CrossRef]
40. Henderson, L. The statistics of crowd fluids. *Nature* **1971**, *229*, 381–383. [CrossRef] [PubMed]
41. Burstedde, C.; Klauck, K.; Schadschneider, A.; Zittartz, J. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Phys. A Stat. Mech. Appl.* **2001**, *295*, 507–525. [CrossRef]
42. Garrell, A.; Sanfeliu, A.; Moreno-Noguer, F. Discrete time motion model for guiding people in urban areas using multiple robots. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 486–491.
43. Chiang, H.T.; Malone, N.; Lesser, K.; Oishi, M.; Tapia, L. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2347–2354.
44. Faust, A.; Francis, A.; Mehta, D. Evolving rewards to automate reinforcement learning. *arXiv* **2019**, arXiv:1905.07628.
45. Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; Sculley, D. Google vizier: A service for black-box optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1487–1495.
46. Francis, A.; Faust, A.; Chiang, H.T.L.; Hsu, J.; Kew, J.C.; Fiser, M.; Lee, T.W.E. Long-Range Indoor Navigation with PRM-RL. *IEEE Trans. Robot.* **2020**, *36*, 1115–1134. [CrossRef]
47. Chiang, H.T.L.; Hsu, J.; Fiser, M.; Tapia, L.; Faust, A. RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4298–4305. [CrossRef]
48. Vaquero, V.; Repiso, E.; Sanfeliu, A. Robust and Real-Time Detection and Tracking of Moving Objects with Minimum 2D LiDAR Information to Advance Autonomous Cargo Handling in Ports. *Sensors* **2019**, *19*, 107. [CrossRef] [PubMed]
49. Linder, T.; Breuers, S.; Leibe, B.; Arras, K.O. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 5512–5519.
50. Garrell, A.; Garza-Elizondo, L.; Villamizar, M.; Herrero, F.; Sanfeliu, A. Aerial social force model: A new framework to accompany people using autonomous flying robots. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 7011–7017.
51. Coll Gomila, C. Learning Aerial Social Force Model for Drone Navigation. Master's Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2018.
52. Repiso, E.; Garrell, A.; Sanfeliu, A. People's Adaptive Side-by-Side Model Evolved to Accompany Groups of People by Social Robots. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2387–2394. [CrossRef]
53. Institut de Robòtica i Informàtica Industrial. Available online: https://www.iri.upc.edu/ (accessed on 1 October 2021).