A graph neural network to model disruption in human-aware robot navigation

P. Bachiller[1] · D. Rodriguez-Criado[2] · R. R. Jorvekar[3] · P. Bustos[1] · D. R. Faria[2] ·
L. J. Manso[2]

© The Author(s) 2021

Abstract

Autonomous navigation is a key skill for assistive and service robots. To be successful, robots have to minimise the disruption caused to humans while moving. This implies predicting how people will move and complying with social conventions. Avoiding disrupting personal spaces, people's paths and interactions are examples of these social conventions. This paper leverages Graph Neural Networks to model robot disruption considering the movement of the humans and the robot so that the model built can be used by path planning algorithms. Along with the model, this paper presents an evolution of the dataset Soc-Nav1 (Manso et al 2020) which considers the movement of the robot and the humans, and an updated scenario-to-graph transformation which is tested using different Graph Neural Network blocks. The model trained achieves close-to-human performance in the dataset. In addition to its accuracy, the main advantage of the approach is its scalability in terms of the number of social factors that can be considered in comparison with handcrafted models. The dataset and the model are available in a public repository (https://github.com/gnns4hri/sngnnv2).

1 Introduction

Human-aware robot navigation deals with the challenge of endowing mobile social robots with the capability of considering the emotions and safety of people nearby while moving around their surroundings. There is a wide range of works studying human-aware navigation from considerably diverse perspectives. Pioneering works such as [28] started taking into account the personal spaces of the people surrounding the robots, often referred to as proxemics. Semantic properties were also considered in [10]. In addition to proxemics, human

---

This paper is an extension of [24]. New contributions include considering the speed and previous poses of the robot and the people around it. It also provides a new dataset.

✉ L. J. Manso
l.manso@aston.ac.uk

Extended author information available on the last page of the article.

motion patterns were analysed in [15] to estimate whether humans are willing to interact with a robot. Although not directly applied to navigation, the relationships between humans and objects were used in the context of ambient intelligence in [3]. Proxemics and object affordances were jointly considered in [42] for navigation purposes. Two extensive surveys on human-aware navigation can be found in [33] and [4].

Despite the previously mentioned approaches being built on well-studied psychological models, they have limitations. Considering new factors programmatically (*i.e.*, writing additional code) involves a potentially high number of coding hours, makes systems more complex, and increases the chances of including bugs. Additionally, with every new aspect to be considered for navigation, the decisions made become less *explainable*, which is precisely one of the main advantages of handcrafted approaches over data-driven ones. In addition to the mentioned model scalability and explainability issues, handcrafted approaches have the intrinsic and rather obvious limitation that they only account for what the model explicitly considers. Given that these models are manually written by humans, they cannot account for aspects that the designers are not aware of.

Approaches leveraging machine learning have also been published. The parameters of a social force model [16] are learned in [11] and [30] to navigate in human-populated environments. Inverse reinforcement learning is used in [32] and [40] to plan navigation routes based on a list of humans in a radius. Social norms are implemented using deep reinforcement learning in [8], again, considering a set of humans. An approach modelling crowd-robot interaction and navigation control is presented in [6]. It features a two-module architecture where single interactions are modelled and then aggregated. Although its authors reported good qualitative results, the approach does not contemplate integrating additional information (*e.g.*, relations between humans and objects, structure and size of the room). The work in [26] tackles the same problem using Gaussian Mixture Models. It has the advantage of requiring less training data, but the approach is also limited in terms of the input information it can process.

All the previous works and many others not mentioned have achieved outstanding results. Some model-based approaches such as [10] or [42] can leverage structured information to take into account space affordances. Still, the data considered to make such decisions are often handcrafted features based on an arbitrary subset of the data that a robot can potentially work with. There are many reasons motivating to seek learning-based approaches not limited to a selection of handcrafted features. Their design is time-consuming and often requires a deep understanding of the particular domain (see discussion in [21]). Additionally, there is generally no guarantee that a particular hand-engineered set of features is close to being the best possible one. On the other hand, most end-to-end deep learning approaches have important limitations too. They require a large amount of data and computational resources that are often scarce and expensive, and they are hard to explain and manually fine-tune. Somewhere in the middle of the spectrum, we have proposals advocating not to choose between hand-engineered features or end-to-end learning. In particular, [2] proposes Graph Neural Networks (GNNs) as a means to perform learning that allows combining raw data with hand-engineered features, and most importantly, to learn from structured information. The relational inductive bias of GNNs is specially well-suited to learn about structured data and the relations between different types of entities, often requiring less training data than other approaches. In this line, we argue that using GNNs for human-aware navigation reduces the time and effort required to integrate new social cues.

In this work, we trained different GNN models to estimate people's comfort given a scenario and its previous states. The state of a scenario includes objects, walls, the robot,

and humans who can be interacting with other humans or objects. For moving entities (*i.e.*, humans and the robot) the network also considers not only their pose but also their linear and angular velocities. GNNs are proposed because the information that social robots can work with is not just a map and a list of people, but a more sophisticated data structure where the entities represented can have different relations among them. For example, social robots could potentially have information about who a human is talking to, where people are looking at, who is friends with whom, or who is the owner of an object in the scenario. Regardless of how this information is acquired, it can be naturally represented using a graph, and GNNs are a particularly well-suited and scalable machine learning approach to work with these graphs.

2 Graph neural networks

2.1 Graph neural networks basics

Graph Neural Networks (GNNs) are a family of machine learning approaches which extend neural networks to be able to take graph-structured data as input. They can perform classifications and regressions on graphs, nodes, edges, as well as predicting links when working with partially observable phenomena. Except for few exceptions (*e.g.*, [46]) GNNs are composed by similar stacked blocks (layers) operating on a graph whose structure remains static but the features associated to its nodes are updated in every layer of the network (see Fig. 1).

As a consequence, the features associated to the nodes of the graph in each layer become more abstract and are influenced by a wider context as layers go deeper. The features in the nodes of the last layer are frequently used to perform the final classification or regression.

The first published efforts on applying neural networks to graphs date back to works by A. Sperduti et al. [39]. GNNs were further studied and formalised by M. Gori et al. [13] and F. Scarselli et al. [37]. However, it was with the appearance of Gated Graph Neural Networks [23] and especially Graph Convolutional Networks (GCNs, [19]) that GNNs gained traction. A review and a unified notation for GNNs can be found in [2].
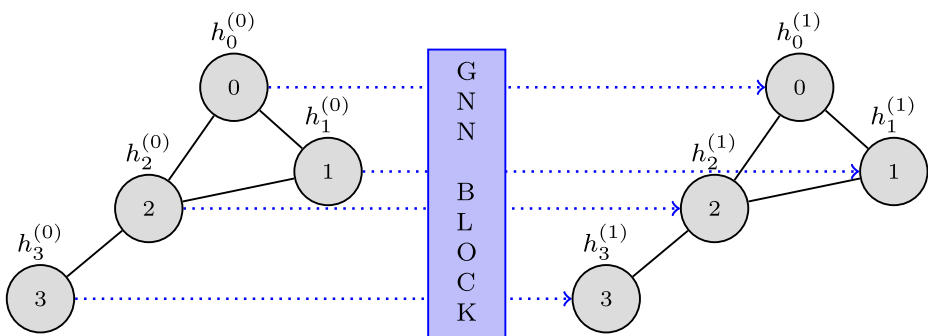

Fig. 1 A basic GNN block/layer. GNN layers output updated versions of the input graph. These updated graphs have the same nodes and links, but the feature vectors of the nodes will generally differ in size and content depending on the feature vectors of their neighbours and their own vectors in the input graph. A GNN is usually composed of several stacked GNN layers. Higher level features are learnt in the deeper layers, so that the output of any of the nodes in the last layer can be used for classification or regression purposes

Graph Convolutional Networks (GCN [19]) is one of the most common GNN blocks. Because of its simplicity, we build on the GCN block to provide the reader with an intuition of how GNNs work. Following the notation proposed in [2], GCN blocks operate over a graph $G = (V, E)$, where $V = \{v_i\}_{i=1:N^v}$ is a set of nodes, being v_i the feature vector of node i and N^v the number of vertices in the graph. $E = \{(s_k, r_k)\}_{k=1:N^e}$ is a set of edges where s_k and r_k are the source and destination indices of edge k and N^e is the number of edges in the graph. Each GCN layer generates an updated representation v_i' for each node v_i using two functions:

$$\rho^{e \to v}(E_i) = \sum_{\{k:r_k=i\}} e_k,$$

$$\phi^v(\overline{e}_i, v_i) = NN_v([\overline{e}_i, v_i]).$$

For every node v_i, the first function ($\rho^{e \to v}(E_i)$) is used to aggregate the feature vectors of other nodes with an edge towards v_i and generates a temporary aggregated feature $\overline{e}_i$ which is used by the second function:

$$\overline{e}_i = \rho^{e \to v}(E_i).$$

The function $\phi^v(\overline{e}_i, v_i)$ is then used to generate an updated v_i' feature vector for each node i from the aggregated feature vector $\overline{e}_i$ using a neural network (usually a multi-layer perceptron, but the framework does not make any assumption on this):

$$v_i' = \phi^v(\overline{e}_i, v_i).$$

Such a learnable function is generally the same for all the nodes. By stacking several blocks where features are aggregated and updated, the feature vectors can carry information from nodes far away in the graph and convey higher level features that can be finally used for classification or regressions.

Several means of improving GCNs have been proposed. Relational Graph Convolutional Networks (R-GCNs [38]) extend GCNs by considering different types of edges separately. They are applied to vertex classification and link prediction in [38]. Graph Attention Networks (GATs [43]) extend GCNs by adding self-attention mechanisms (see [41]). They are applied to vertex classification in [43]. In Message Passing Graph Neural Networks (MPNNs [12]), the messages which are aggregated are not only composed of node features but also edge features. This allows MPNNs to account for both vertex and edge features. For a more detailed review of GNNs and the generalised framework, please refer to [2].

2.2 Graph neural networks applied to human-aware navigation

A number of recent machine learning-based approaches leveraging structured data for social navigation have been recently published. A GNN model integrated with a Deep Reinforcement Learning (DRL) algorithm based in Monte Carlo Tree Search was presented in [5]. It utilises a graph-based model to detect the implicit relations between the humans in a room. Interactions are useful to predict future human trajectories. For instance, interacting pedestrians generally behave differently than those who do not interact. This phenomena is also exploited in [7], where a GCN-based DRL leverages the gaze of humans to estimate interactions and predict their trajectories. These works consider human-robot and human-human relations but disregard interactions with objects or obstacles that could be exploited. Moreover, the DRL algorithms in [5] and [7] use a simple handcrafted reward function based on the minimum distance between the robot and the humans that disregards any other information including the orientation and velocity of the humans or how densely populated the

room is (distance restrictions are usually eased in crowded spaces). Due to the variety of different scenarios and factors to consider, handcrafting a reward function that complies with social rules seems prohibitively complex and time-consuming.

A model combining a Convolutional Neural Network (CNN) and a GNN to learn an action policy for multi-robot navigation is presented in [22]. The CNN extracts features from local observations of the environment, and the action policy for the robot swarm is computed from those features using GNN. Although safety and collision avoidance are considered, the approach only considers humans as obstacles.

Other works use GNNs for reasoning and perception in the domain of social navigation. A significant amount of them directs their focus to the prediction of pedestrians' paths as exemplified in works undertaken by [17, 44] or [14]. The use of GNNs for these tasks allows extracting additional information from the crowd such as relations between people. However, none of the previous works tackle the problem of modelling discomfort.

GNNs have been used to model and estimate discomfort in our previous works, [24] and [35]. Both works generate discomfort estimations on a scale from 0 to 100 and consider human-human, human-robot and human-object interactions, as well as walls and other objects. While [24] generates a single value a given scenario, [35] generates a two-dimensional cost map using a combination of GNNs and CNNs, in that order. The main limitation of these models is that the scenarios they consider are static (*i.e.*, they disregard human and robot motion).

The work at hand follows a similar approach to [24] with a number of enhancements. Firstly, we consider two different scores to measure two aspects of social navigation (see Section 3). Secondly, the model is trained using dynamic scenes where humans and the robot move, which was the main limitation of [24].

3 SocNav2 dataset

SocNav1 [25], was designed to learn and benchmark estimation functions for social navigation conventions. *SocNav2* -presented in this paper- has the same goal as its predecessor but unlike SocNav1, it considers the velocity and trajectory of the robots and the humans around them. As SocNav2, SocNav1 contains scenarios with a robot in a room, a number of objects and a number of people that can potentially be interacting with other objects or people. In case any human-human or human-object interaction exists it is also noted in the scenarios. Each sample in the dataset is given a score between 0 and 100, depending on the extent that the subjects consider that the robot is disturbing the people in the scenario. The main limitation of SocNav1 is that samples do not consider velocity information or the trajectory of the humans.

SocNav2 overcomes such limitation and provides 13406 scored samples of dynamic scene sequences. Each sample consists of 35 "snapshots" of a scene of a room with a moving robot, objects and potentially moving humans, taken during a time interval of 4 seconds. In SocNav2 the room also includes a landmark that constitutes a goal position to be reached by the robot.

Each SocNav2 sample includes scores for two social navigation-related statements: "*the robot does not cause any disturbance to the humans in the room*" (*Q1*) and "*the robot is moving towards the goal efficiently, not causing any disturbance to the humans in the room*" (*Q2*). The scores range from 0 to 100, considering the following reference values:

–   0: unacceptable

- – 20: undesirable
- – 40: acceptable
- – 60: good
- – 80: very good
- – 100: perfect

The scenarios compiled in SocNav2 have been generated using SONATA [1]. SONATA is a toolkit built on top of PyRep [18] and CoppeliaSim [36] designed to simulate human-populated navigation scenarios and to generate datasets. It provides an API to generate random scenarios including humans, objects, interactions, the robot and its goals. The walls delimiting a room are also randomly generated considering rectangular and L-shaped rooms. Despite SONATA only provides simulated scenarios, the use of synthetic data is essential in the context of social navigation. Firstly, because it would not be feasible to generate as many situations using only real-world data. Secondly, because situations endangering humans' integrity, such as human-robot collisions, could not be generated in real scenarios.

The movements of the robot were generated through two different strategies to increase the diversity of its behaviour. The first strategy uses a machine learning model (see [1]) that outputs the control actions of the robot according to a graph representation of the scenario. This model was trained using supervised learning (*i.e.*, it only contains examples of appropriate behaviours), so it has unexpected behaviours in situations that would not usually happen when controlled by humans. Nevertheless, for the creation of SocNav2, these behaviours allow to generate a wide variety of good and bad situations that would not have been obtained from random actions. In addition to the samples where the movement of the robot was controlled by the machine learning approach, a second set of samples was generated using a joystick to control the robot manually. This second set was created to include infrequent situations in the first set, such as the robot moving backwards to avoid getting blocked or stopping to let people pass.

The subjects providing the scores for SocNav2 were shown sequences of 4 seconds, and were asked to give their answers for the behaviour of the robot in the last second. During the three previous seconds, the video was shaded to make easier to know what time slice had to be evaluated (see Fig. 2). The geometrical and relational data of the sequences were stored in JSON files. Subjects were asked to provide a score for *Q1* and *Q2* after watching the video (as many times as necessary) according to the aforementioned reference values. Despite some guidelines were given, subjects were requested to feel free to express their opinions. Some of the guidelines were the following:

- – The goal should be disregarded when answering Q1. It should only be considered when answering Q2.
- – The closer the robot gets to people, the more it can be deemed disturbing.
- – In small rooms with a high number of people, closer distances are acceptable in comparison to big rooms with fewer people.
- – The robot is required not to collide with objects or walls. If it collides it should have a score of 0.

Six subjects participated in the scoring of the dataset, producing 13406 scored samples. This initial set of samples has been extended using data augmentation. Specifically, each scenario has been mirrored in the vertical axis assuming the same scores as in the original scenario. In addition, each normal and mirrored scenario has been rotated 180°, changing also the sign of the advance speed of the robot. This extension assumes that the human
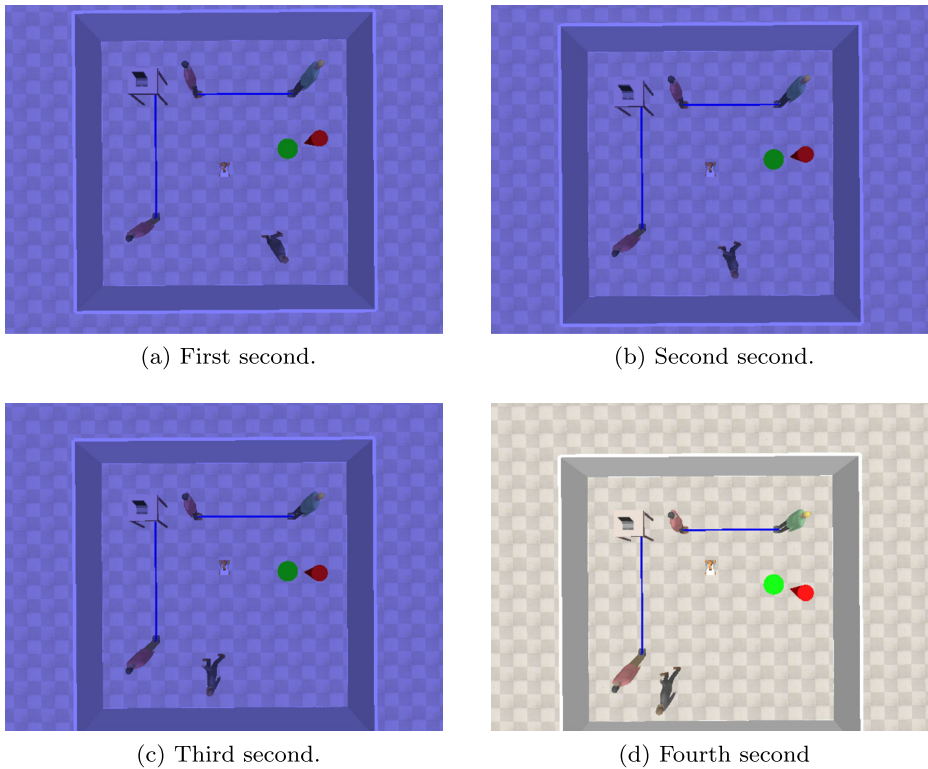
(a) First second.

(b) Second second.

(c) Third second.

(d) Fourth second

Fig. 2 A SocNav2 sequence. The shaded images correspond to the first 3 seconds of the sequence, which are also shown to subjects to provide context. The last image, in Fig. 2d, corresponds to the second that the users score. During the whole sequence the robot is moving forwards

discomfort does not change whether the robot is moving forwards or backwards. As a result of this data augmentation process, the final dataset is composed of 53600 samples.

In order to analyse the consistency of the scoring of the dataset, the inter-rater and intra-rater agreements have been computed for 4 subjects using the linearly weighted kappa coefficient [9]. For the inter-rater consistency, common samples scored by each pair of subjects were considered. The minimum number of common samples for which this coefficient was obtained is 609. For measuring the intra-rater reliability, each user scored 200 duplicate samples. Tables 1 and 2 show the inter-rater and intra-rater consistency for the scores of Q1 and Q2, respectively (intra-rater in the diagonal cells, inter-rater in the remaining cells).

As shown in Table 1, the intra-rater consistency for Q1 is "*almost perfect*" in the scale defined in [20]. The inter-rater agreement for Q1 can be considered substantial in most of the cases. Only subjects 1 and 4 have a low agreement, but they fall in the high *moderate* bracket. Table 2 shows that the consistency for Q2 is generally lower than for Q1. This reduction can be due to the very nature of the question, since subjects may broadly differ about how the robot should move to *efficiently* reach the goal position. Nevertheless, the inter-rater and intra-rater consistency is still *substantial* excepting for subjects 1 and 4, which is high *moderate*.

Table 1 Inter-rater and intra-rater consistency of four subjects for Q1

	Subject1	Subject2	Subject3	Subject4
Subject1	0.83	0.75	0.80	0.56
Subject2	0.75	0.88	0.85	0.63
Subject3	0.80	0.85	0.88	0.62
Subject4	0.56	0.63	0.62	0.81

4 Scenario to graph transformation

This paper follows the strategy developed in [24] and includes a number of modifications to account for velocity and trajectory information. To leverage the properties of GNNs (see Section 2) the input data from *SocNav2* has to be transformed into a graph. This section describes the scenario-to-graph transformation process.

4.1 Graph structure

The graphs inputted to the GNN models are composed of a sequence of 3 sub-graphs for 3 *snapshots* of the videos shown to the subjects. Each sub-graph (*frame graph*) is separated by 1 second, being the last one the graph which the users scored. The graph creation process has two steps. First, each *snapshot* is transformed into a separate *frame graph*. Once the 3 frame graphs in the sequence have been generated, they are merged into a single graph representing the sequence (see Fig. 3). This temporal connection is done with an edge linking the node in each frame graph with the same node in the next frame graph.

The nodes in the graphs have five types:

–   **room:** There is one room node per frame graph. It acts as a global node [2] and it is connected to any other node of the graph for that frame. Using a global node favours communication across the graph and reduces the number of layers required.
–   **wall:** A node for each of the segments defining the room.
–   **goal:** Used to represent the position that the robot must reach.
–   **object:** A node for each object in the scenario.
–   **human:** A node for each human. Humans might be interacting with objects or other humans.

There is no node explicitly representing the robot because all node features are in the reference frame of the robot (further explained in Section 4.2). For every human engaging in interactions, two new edges are added between the human and the entity (human or object) they interact with, one in each direction. The graphs also include self-edges for all nodes, and the room node is connected in both directions to the rest of the nodes in the graph. As an example, Fig. 2 depicts four frames of a sequence where four humans are in a room

Table 2 Inter-rater and intra-rater consistency of four subjects for Q2

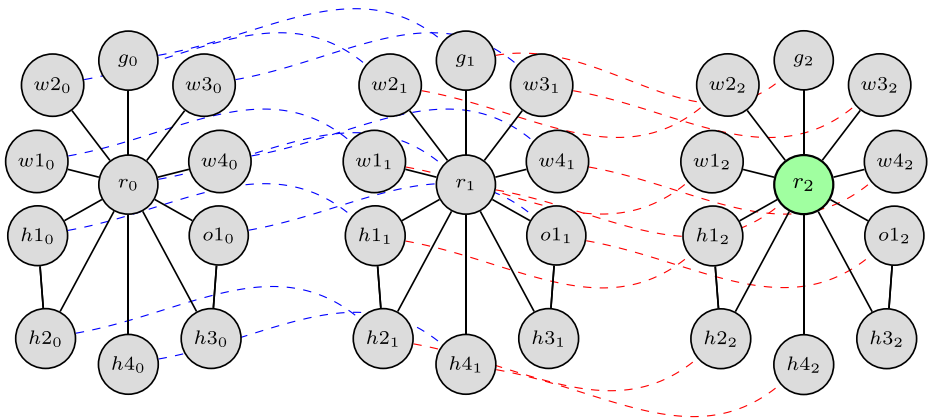|	Subject1	Subject2	Subject3	Subject4
Subject1	0.74	0.68	0.72	0.57
Subject2	0.68	0.71	0.74	0.63
Subject3	0.72	0.74	0.76	0.64
Subject4	0.57	0.63	0.64	0.73

Fig. 3 Example of how the scenario-to-graph transformation works, based on the scenario depicted in Fig.2

with several objects. Two of the humans are interacting with each other, another human is interacting with an object, and the remaining human is moving without interacting with other human or object. Figure 3 shows the structure of the resulting graph.

4.2 Node and edge features

Node feature vectors are built by concatenating different sections. The first section is a one-hot encoding for the type of node. The remaining sections are type specific and are only filled if the node is of the corresponding type, filled with zeros otherwise. The features used in the sections for *human*, *wall* and *object* nodes are: position, distance to the robot, speed and orientation, all from the robot's frame of reference. Position and distance are represented in decametres for normalization purposes. Similarly, the orientation is split into sine and cosine, instead of including the angle itself. For wall segments, the position is the centre of the segment and the orientation is the tangent. Object sections also contain width and height features defining the object's bounding box. The section corresponding to the *room* symbol is composed of the number of humans in the room and the velocity command given to the robot. Table 3 depicts this layout.

Table 3 Structure of the feature vectors of nodes

n. one-hot	5 elements (one per node type)			
f. one-hot	3 elements (one per frame graph)			
room	number of humans		adv. speed	rot. speed
human	pos. (2D)	speed (3D)	orientation (2D)	distance
object	pos. (2D)	speed (3D)	orientation (2D)	distance	shape (2D)
wall	pos. (2D)	orientation (2D)		distance
goal	pos. (2D)		distance	

The first two sections refer to the one-hot encodings that specify the node types and the frame they belong to. Positions (*pos.*) are defined by 2D euclidean coordinates. Speeds are expressed using 3 dimensions for the linear and angular velocities in the plane. Orientations are given by the corresponding sine and cosine values. All metric values are in the robot's reference frame

Table 4 Ranges of the hyperparameter values sampled

hyperparameter	min	max
max. epochs	1000	
patience	4	
batch size	25	70
hidden units	20	90
attention heads	3	10
number of bases	4	24
learning rate	1e-4	5e-4
weight decay	0.0	1e-6
layers	3	9
dropout	0.0	1e-6
alpha	0.1	0.3

Attention heads is only applicable to GAT blocks. *Number of bases* is only applicable to R-GCN blocks

Edge features were implemented differently for the experiments depending on the blocks used. Some GNN blocks such as GAT or GCN, do not support edge features or labels, so no edge information is provided when they are used. R-GCN blocks support edge labels, so a different label is used for each possible type of relation (*e.g.*, human-human, human-room, wall-room). MPNN blocks treat edge information as features not limiting it to identifiers. Therefore, besides containing values identifying the kind of relationship as a one-hot encoding, edge features also include the distance between the two entities being linked when using MPNN blocks.

5 Experimental results

Based on the assumption that in real life scenarios we can build on top of third party body trackers (*e.g.*, [31, 34]) and path planning systems, we proceed with the evaluation of the approach against the dataset presented in Section 3. Because all nodes are connected to their corresponding room node, the GNNs were trained to perform backpropagation based on the feature vector of the *room* node in the last layer.

Three GNN blocks were considered in the experiments: the two best-performing GNN blocks in [24] (*i.e.*, R-GCN [38], GAT [43]) and MPNN [12]. The implementations tested are based on the Deep Graph Learning library (DGL [45]), using PyTorch [29] as backend.

To benchmark the different architectures, 341 training sessions were launched using the SocNav2 dataset with a split of 47598 samples for training, 643 for evaluation and 643 for testing. Given the variability of scenarios, 643 was considered a representative sample set size. The hyperparameters were randomly sampled from the range values shown in Table 4. Table 5 summarises the results obtained for the best model of each architecture, providing the performance on the different splits of the dataset.

The training results obtained (see Table 5) show that MPNN blocks delivered the best results, with a Mean Squared Error (MSE) of 0.036821 for the evaluation dataset. The best model, which was selected based on the MSE on the evaluation split, yielded an MSE of 0.035192 for the test split. The best performing model was trained with a batch size of 57, a learning rate of 2.5e-4, weight decay regularisation of 1.0e-6 and no dropout. Its **network architecture** is a sequence of **6 MPNN blocks** with 40, 30, 21, 12 and 3 hidden units.

GNN block	training loss	development loss	test loss
	(MSE)	(MSE)	(MSE)
R-GCN	0.017347	0.040098	0.040607
GAT	0.015188	0.037838	0.035818
MPNN	0.025020	**0.036821**	0.035192

Table 5 The 3 GNN blocks tested along with their MSE for SocNav2

To provide an intuition of the output of the network, the scenarios of Figs. 4, 5 and 2 have been tested considering the output of the model for all the different positions of the robot in the room. As a result, a heatmap representation of the network's response has been obtained for each tested scenario. To ease the interpretation of each heatmap, the elements presented in the scenarios have been drawn over the image with the following representation: oriented blue circles for humans, small green circles for objects, a wider green circle for the goal position and red lines for interactions. The horizontal and vertical axis of the room's frame of reference have also been depicted using black discontinuous lines to help distinguish the differences among the heat maps.

Figure 6 shows the resulting generated maps for the first and last situations of the scenario of Fig.4 considering the network output for Q1. The different colours represent the output of the network. A red colour is used to show a value near to 0 (unacceptable situation). Grey tones represent the remaining range of values, where dark grey levels indicate lower values (high degree of discomfort) and a light one a high value (socially acceptable). This test shows how the network adapts to differently populated environments. For crowded spaces such as the one in Fig. 4a, the discomfort area of the humans narrows in relation to scenarios with less dense spaces. For instance, the *unacceptable* area of the humans in the bottom left of the room is wider in Fig. 6b than in Fig. 6a. In addition, the response of the network increases in the positions near the walls if the number of people in the room is high (see the goal marked by a green circle in the right top corner of the images as a reference point). This means that the positions near the limits of the room are considered more suitable for crowded environments.

The scenario in Fig. 5 has been used to test how the actions of the robot have influence in the behaviour of the network. Figures 7 and 8 show the response of the network for Q1 and Q2, respectively. From bottom to top, left to right, the actions of the robot for each image

(a) Fourteen humans.   (b) Two humans.
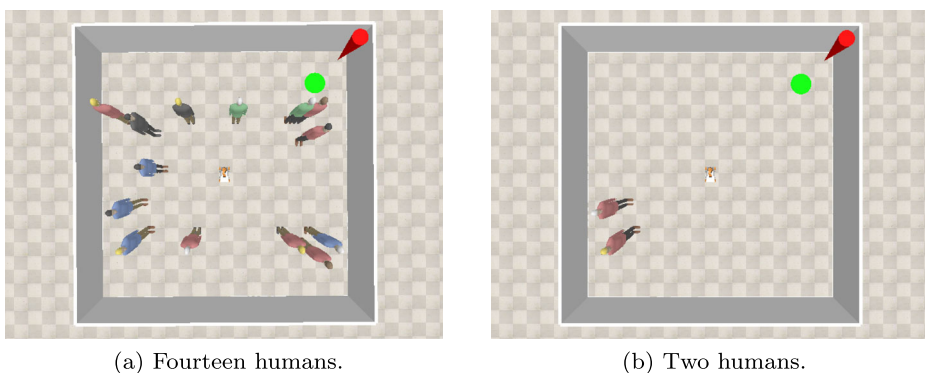
Fig. 4 Two scenarios containing a different number of people. Results for these scenarios are shown in Fig. 6

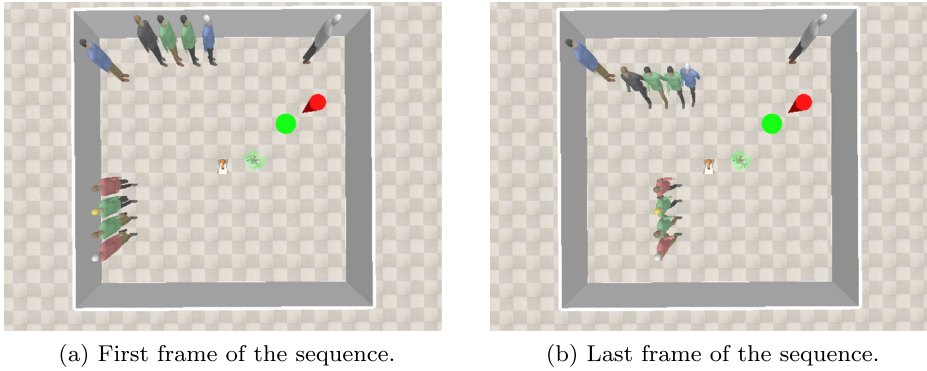(a) First frame of the sequence.   (b) Last frame of the sequence.

Fig. 5 Scenario with two groups of people walking. Results for these scenarios are shown in Figs. 7 and 8

are the following: turning left, stopped, turning right, moving forward to the left, moving forward, moving forward to the right. As shown in Fig. 7, the *unacceptable* area (red area) of moving people changes according to their motion direction and the robot actions, while for standing humans such an area remains almost unalterable. In this way, when the robot moves forward (Fig. 7b) the red area of the group of people moving in the opposite direction extends towards the direction of the movement. However, for the same action of the robot, the red area of the group of people moving in the horizontal direction keeps centered in the vertical-axis' position of the humans. For this second group, the unacceptable area extends forwards or backwards when the robot moves to the left (Fig. 7a) or to the right (Fig. 7c). When the robot is stopped or turning without translation, the positions with the lowest scores elongate towards the opposite direction of the movement of the humans (Fig. 7d, e and v). These positions correspond to the trajectory followed by the humans during the sequence, therefore the network response can be considered consistent with the situation.
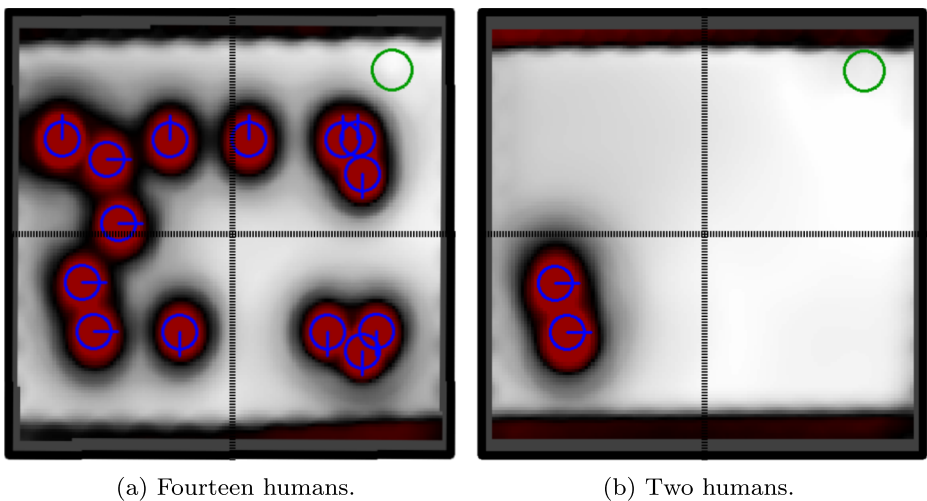

(a) Fourteen humans.   (b) Two humans.

Fig. 6 Output of the model for the two scenarios in Fig. 4. The response of the model is more strict for the case with fewer people

(a) Moving forward to the left.   (b) Moving forward.   (c) Moving forward to the right.

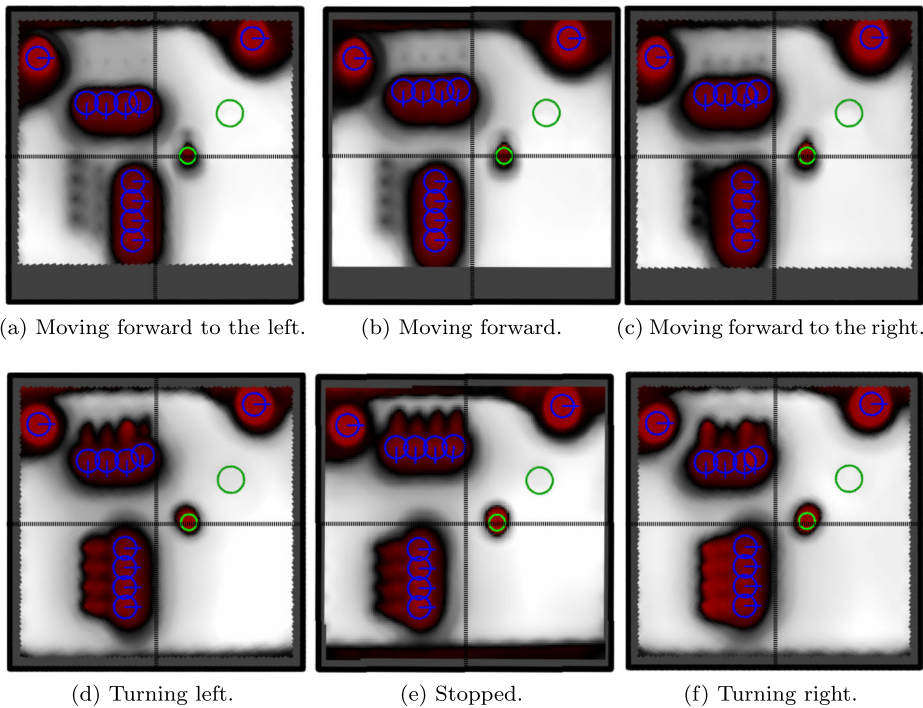(d) Turning left.   (e) Stopped.   (f) Turning right.

Fig. 7 Response of the model for Q1 for the scenario in Fig. 5 considering different actions of the robot

As expected, the response of the network regarding humans is maintained for Q2 (Fig. 8), but in this case the positions with low values increase according to the goal position and the action of the robot. For instance, moving forward leaving the goal behind has a very low score. Thus, when the goal is situated behind the robot, the best scoring actions are turning right or left according to the relative position of the goal.

To test the network response to potential interactions between humans or humans and objects, the scenario of Fig. 2 has been used with the robot moving forward. Results for this scenario with and without interactions for Q1 are shown in Fig. 9. As can be observed in Fig. 9a, the interaction between the human and the object produces lower values than the interaction between the two humans. This is consistent with the action of the robot, since the human-object interaction is taking place in the direction of the movement of the robot. As a consequence, the interruption caused by the robot action is more intense than the one that is produced in the human-human interaction. If no interactions are taking place (Fig. 9b), the areas between the two humans in the top of the image and the human and the object in the left are considered socially acceptable positions. Thus, the network is properly generalising the different kinds of situations. Another interesting result that can be seen in Fig. 9b is the different treatment of humans and objects when objects are not being used by humans. Specifically, being close to an object has a high response, while being close to a human is not considered acceptable.

Due to the subjective nature of the scores in the dataset (human feelings are utterly subjective), there is some level of disagreement even among humans. To compare the performance of the network with human performance, we used a subset of the samples in SocNav2 which was labelled twice by each of the subjects (the same subset used to obtain
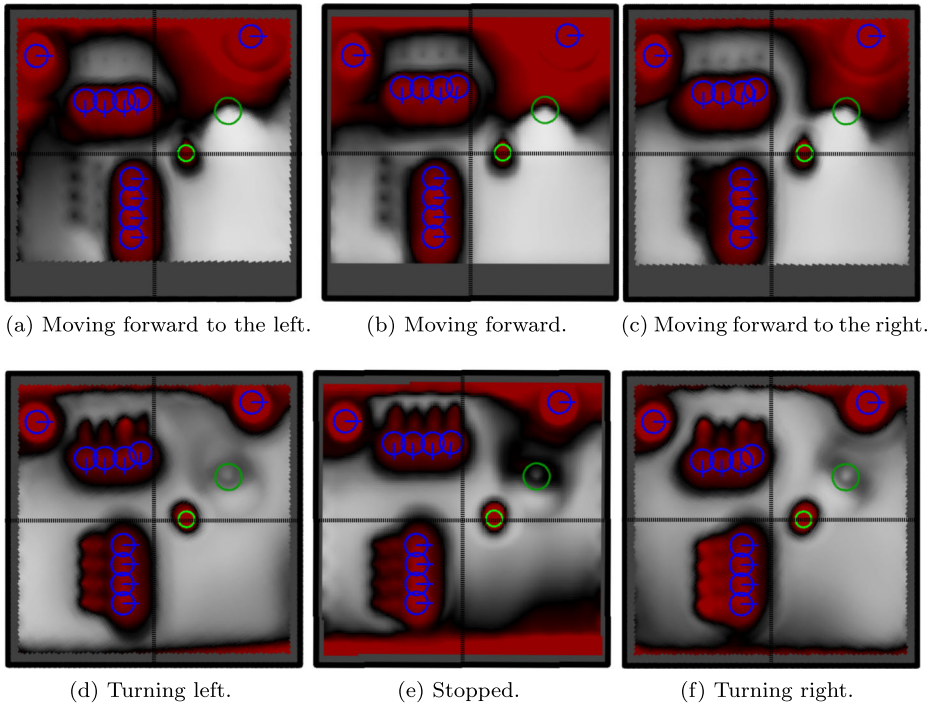
(a) Moving forward to the left.   (b) Moving forward.   (c) Moving forward to the right.

(d) Turning left.   (e) Stopped.   (f) Turning right.

Fig. 8 Output of the model for Q2 for the scenario depicted in Fig. 5 considering different actions of the robot

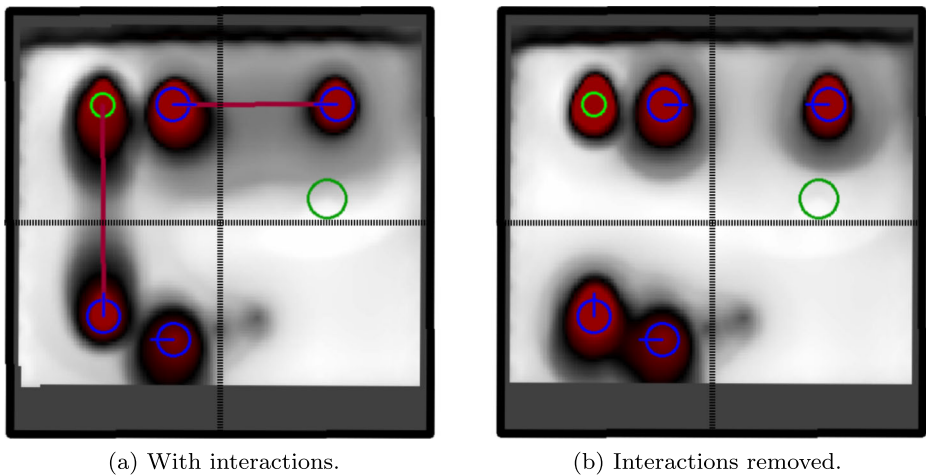(a) With interactions.   (b) Interactions removed.

Fig. 9 The output of the model for the sequence depicted in Fig. 2. Figure 9a is the output of the model with the sequence in its original form. Figure 9b is the output of the model with the interactions removed. It is apparent that the response of the model for the perpendicular interaction is lower than that of the parallel one. This aligns with the intuition that the robot would be less disturbing if crossing perpendicularly than along the interaction line
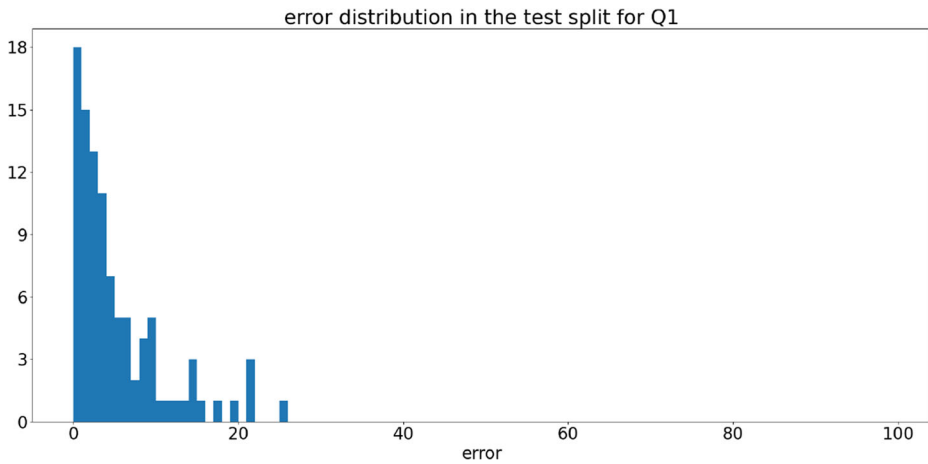
Fig. 10 Histogram of the absolute error in the test dataset for Q1

Tables 1 and 2). Using the mean of the 8 scores that were provided for each scenario as a reference, the MSE for each of the participants was computed. The average MSE was 0.036981, so we use that value as an indicative of human accuracy. This means that the network performs close to human accuracy (even slightly better 0.035192). Figures 10 and 11 show the histograms of the error of the model in the test split of the dataset for Q1 and Q2. In [24] we compared our results disregarding speed with [42] and achieved a considerably lower mean squared error (0.022 versus 0.12965). Although the comparison was favourable, it is not entirely fair as the approaches have slightly different goals. We are aware of other researchers currently working with the dataset used in this paper and SocNav1 [25], but there are no published works to compare with at the time of writing.
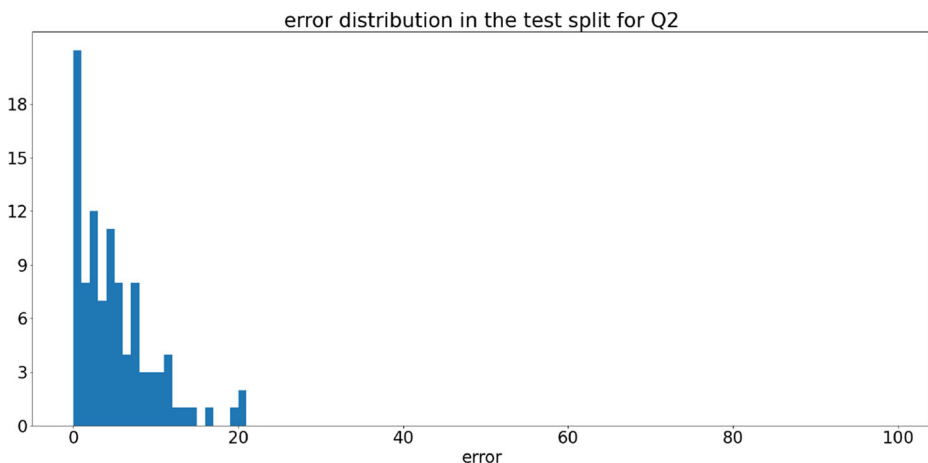

Fig. 11 Histogram of the absolute error in the test dataset for Q2

6 Conclusions

Most approaches introduced in Section 1 deal with modelling human intimate, personal, social and interaction spaces instead of social inconvenience, which is a more general term. To the best of our knowledge, all papers modelling discomfort around robots disregard information such as explicit interactions, trajectories or speed. This paper tackled these issues with a specific scenario-to-graph transformation and a graph neural network architecture composed of 6 MPNN blocks.

The results obtained are close to human accuracy and improve those in [24] not only in terms of MSE but also in terms of the features considered (*i.e.*, the trajectory and speed of the robot and the humans). The results confirm that the discomfort is only skewed to the front of the humans when there is movement involved, which was initially hypothesised in [24]. The results also show that: **a)** the model adapts to a variable density of humans (see Fig. 6); **b)** static humans are considered more carefully; and **c)** the model is able to consider the interactions which have been given explicitly.

Future works point to user profiling and personalisation, as well as considering the activity of the humans and their gaze as done in works such as [7]. Also, an ongoing line of research explores ways of linking the output of the GNN (questions Q1 and Q2) to driving control of the robot. An end-to-end solution is a possibility but complicates the acquisition of labelled examples and the modulation of the final control action. An interesting alternative would be to use the output of the GNN as an additional restriction to be fulfilled by a Model Predictive Controller [27].

The code to test the resulting GNN model, including the code implementing the scenario-to-graph transformation and the code to train the model suggested, has been published in a public repository as open-source software: https://github.com/gnns4hri/sngnnv2.

Declarations

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

References

1. Baghel R, Kapoor A, Bachiller P, Jorvekar RR, Rodriguez-Criado D, Manso LJ (2020) A toolkit to generate social navigation datasets. In: Workshop of physical agents. Springer, pp 180–193
2. Battaglia P. W., Hamrick J. B., Bapst V., Sanchez-Gonzalez A., Zambaldi V., Malinowski M., Tacchetti A., Raposo D., Santoro A., Faulkner R., Gulcehre C., Song F., Ballard A., Gilmer J., Dahl G., Vaswani A., Allen K., Nash C., Langston V., Dyer C., Heess N., Wierstra D., Kohli P., Botvinick M., Vinyals O., Li Y., Pascanu R (2018) Relational inductive biases, deep learning, and graph networks. 1–40. https://doi.org/10.1017/S0031182005008516. arXiv:1806.01261

3. Bhatt M, Dylla F (2010) A qualitative model of dynamic scene analysis and interpretation in ambient intelligence systems. Int J Robot Autom 24(3):1–18. https://doi.org/10.2316/journal.206.2009.3.206-3274

4. Charalampous K, Kostavelis I, Gasteratos A (2017) Recent trends in social aware robot navigation: A survey, vol 93. Elsevier B.V, Amsterdam, pp 85–104. https://doi.org/10.1016/j.robot.2017.03.002

5. Chen C., Hu S., Nikdel P., Mori G., Savva M (2019) Relational Graph Learning for Crowd Navigation. arXiv:1909.13165

6. Chen C., Liu Y., Kreiss S., Alahi A (2019) Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In: International Conference on Robotics and Automation (ICRA). IEEE, pp 6015–6022. arXiv:1809.08835

7. Chen Y, Liu C, Shi BE, Liu M (2020) Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. IEEE Robot Autom Lett 5(2):2754–2761. https://doi.org/10.1109/LRA.2020.2972868

8. Chen YF, Everett M, Liu M, How JP (2017) Socially aware motion planning with deep reinforcement learning. IEEE Int Conf Intell Robot Syst 2017-Septe:1343–1350. https://doi.org/10.1109/IROS.2017.8202312

9. Cohen J (1968) Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychol Bull 70(4):213

10. Cosley D, Baxter J, Lee S, Alson B, Nomura S, Adams P, Sarabu C, Gay G (2009) A tag in the hand: Supporting semantic, social, and spatial navigation in museums. In: Proceedings of the 27th international conference on human factors in computing systems (CHI'09), pp 1953–1962. https://doi.org/10.1145/1518701.1518999

11. Ferrer G, Garrell A, Sanfeliu A (2013) Social-aware robot navigation in urban environments. In: 2013 European Conference on Mobile Robots ECMR 2013 - Conference Proceedings, pp 331–336. https://doi.org/10.1109/ECMR.2013.6698863

12. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: Proceedings of the 34th international conference on machine learning, vol 70, pp 1263–1272. JMLR.org

13. Gori M, Monfardini G, Scarselli F (2005) A new Model for Learning in Graph domains. Proc Int Joint Conf Neural Netw 2:729–734. https://doi.org/10.1109/IJCNN.2005.1555942

14. Haddad S, Lam SK (2020) Self-growing spatial graph networks for pedestrian trajectory prediction. Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision WACV 2020:1140–1148. https://doi.org/10.1109/WACV45572.2020.9093456

15. Hansen ST, Svenstrup M, Andersen HJ, Bak T (2009) Adaptive human aware navigation based on motion pattern analysis. In: Proceedings - IEEE international workshop on robot and human interactive communication, pp 927–932. https://doi.org/10.1109/ROMAN.2009.5326212

16. Helbing D, Molnár P (1995) Social force model for pedestrian dynamics. Phys Rev E 51(5):4282–4286. https://doi.org/10.1103/PhysRevE.51.4282

17. Huang Y, Bi H, Li Z, Mao T, Wang Z (2019) STGAT: Modeling Spatial-temporal interactions for human trajectory prediction. In: Proceedings of the IEEE international conference on computer vision 2019-october, pp 6271–6280. https://doi.org/10.1109/ICCV.2019.00637

18. James S, Freese M, Davison AJ (2019) Pyrep: Bringing v-rep to deep robot learning. arXiv preprint

19. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. 1–14. arXiv:1609.02907

20. Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1)

21. Lecun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539

22. Li Q, Gama F, Ribeiro A, Prorok A (2019) Graph Neural Networks for Decentralized Multi-Robot Path Planning. arXiv preprint

23. Li Y, Tarlow D, Brockschmidt M, Zemel R (2015) Gated Graph Sequence Neural Networks. 1–20. arXiv:1511.05493

24. Manso LJ, Jorvekar RR, Faria DR, Bustos P, Bachiller P (2020) Graph neural networks for human-aware social navigation. In: Workshop of physical agents. Springer, pp 167–179

25. Manso LJ, Nuñez P, Calderita LV, Faria DR, Bachiller P (2020) Socnav1: A dataset to benchmark and learn social navigation conventions. Data 5(1). https://www.mdpi.com/2306-5729/5/1/7

26. Martins GS, Rocha RP, Pais FJ, Menezes P (2019) Clusternav: Learning-based robust navigation operating in cluttered environments. In: 2019 International conference on robotics and automation (ICRA). IEEE, pp 9624–9630

27. Neunert M, De Crousaz C, Furrer F, Kamel M, Farshidian F, Siegwart R, Buchli J (2016) Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. In: Proceedings - IEEE International Conference on Robotics and Automation, ICRA, pp 1398–1404. https://doi.org/10.1109/ICRA.2016.7487274

28. Pacchierotti E, Christensen HI, Jensfelt P (2005) Human-robot embodied interaction in hallway settings: a pilot user study. In: IEEE International workshop on robot and human interactive communication, vol 2005. IEEE, pp 164–171. https://doi.org/10.1109/ROMAN.2005.1513774

29. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: An imperative style, high-performance deep learning library. arXiv:1912.01703

30. Patompak P, Jeong S, Nilkhamhang I, Chong NY (2019) Learning proxemics for personalized Human-Robot social interaction. International Journal of Social Robotics. https://doi.org/10.1007/s12369-019-00560-9

31. Qi S, Wang W, Jia B, Shen J, Zhu SC (2018) Learning human-object interactions by graph parsing neural networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11213 LNCS, 407–423. https://doi.org/10.1007/978-3-030-01240-3_25

32. Ramon-Vigo R, Perez-Higueras N, Caballero F, Merino L (2014) Transferring human navigation behaviors into a robot local planner. In: IEEE RO-MAN 2014 - 23rd IEEE International Symposium on Robot and Human Interactive communication: Human-Robot co-existence: Adaptive Interfaces and Systems for Daily Life, Therapy, Assistance and Socially Engaging Interactions, pp 774–779. https://doi.org/10.1109/ROMAN.2014.6926347

33. Rios-Martinez J, Spalanzani A, Laugier C (2015) From proxemics theory to Socially-Aware navigation: a survey. Int J Soc Robot 7(2):137–153. https://doi.org/10.1007/s12369-014-0251-1

34. Rodriguez-Criado D, Bachiller P, Bustos P, Vogiatzis G, Manso LJ (2020) Multi-camera torso pose estimation using graph neural networks

35. Rodriguez-Criado D, Bachiller P, Manso LJ (2020) Generation of human-aware navigation maps using graph neural networks. arXiv:2011.05180

36. Rohmer E, Singh SP, Freese M (2013) Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: Proc. Int. Conf. on intelligent robots and systems, pp 1321–1326

37. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80. https://doi.org/10.1109/TNN.2008.2005605

38. Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M (2018) Modeling Relational Data with Graph Convolutional Networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10843 LNCS(1), 593–607. https://doi.org/10.1007/978-3-319-93417-4_38

39. Sperduti A, Starita A (1997) Supervised neural networks for the classification of structures. IEEE Trans Neural Netw 8(3):1–22

40. Vasquez D, Okal B, Arras KO (2014) Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In: 2014 IEEE/RSJ International conference on intelligent robots and systems. IEEE, pp 1341–1346

41. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008

42. Vega A, Manso LJ, Macharet DG, Bustos P, Núñez P (2019) Socially aware robot navigation system in human-populated and interactive environments based on an adaptive spatial density function and space affordances. Pattern Recogn Lett 118:72–84. https://doi.org/10.1016/j.patrec.2018.07.015

43. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: Proceedings of the international conference on learning representations 2018, 2015, pp 1–11. arXiv:1710.10903

44. Vemula A, Muelling K, Oh J (2017) Social attention: Modeling attention in human crowds. arXiv pp 4601–4607

45. Wang M, Zheng D, Ye Z, Gan Q, Li M, Song X, Zhou J, Ma C, Yu L, Gai Y et al (2019) Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv:1909.01315

46. Ying Z, You J, Morris C, Ren X, Hamilton W, Leskovec J (2018) Hierarchical graph representation learning with differentiable pooling. In: Advances in neural information processing systems, pp 4800–4810

Affiliations

P. Bachiller[1] · D. Rodriguez-Criado[2] · R. R. Jorvekar[3] · P. Bustos[1] · D. R. Faria[2] · L. J. Manso[2] (iD)

P. Bachiller
pilarb@unex.es

D.Rodriguez-Criado
190229717@aston.ac.uk

R. R. Jorvekar
ronitjorvekar007@gmail.com

P. Bustos
pbustos@unex.es

D. R. Faria
d.faria@aston.ac.uk

[1]   Robotics and Artificial Vision Laboratory, University of Extremadura, Extremadura, Spain

[2]   College of Engineering and Physical Sciences, Aston University, B4 7ET, Birmingham, UK

[3]   Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India