Graph Neural Networks for Human-aware Social Navigation

Luis J. Manso[1], Ronit R. Jorvekar[2], Diego R. Faria[1],
Pablo Bustos[3], and Pilar Bachiller[3]

[1] Dept. of Computer Science, College of Engineering and Physical Sciences,
Aston University, United Kingdom
[2] Dept. of Computer Engineering, Pune Institute of Computer Technology, India
[3] Robotics and Artificial Vision Laboratory, University of Extremadura, Spain
`l.manso@aston.ac.uk`,

Abstract. Autonomous navigation is a key skill for assistive and service robots. To be successful, robots have to comply with social rules, such as avoiding the personal spaces of the people surrounding them, or not getting in the way of human-to-human and human-to-object interactions. This paper suggests using Graph Neural Networks to model how inconvenient the presence of a robot would be in a particular scenario according to learned human conventions so that it can be used by path planning algorithms. To do so, we propose two automated scenario-to-graph transformations and benchmark them with different Graph Neural Networks using the SocNav1 dataset [1]. We achieve close-to-human performance in the dataset and argue that, in addition to its promising results, the main advantage of the approach is its scalability in terms of the number of social factors that can be considered and easily embedded in code in comparison with model-based approaches. The code used to train and test the resulting graph neural network is available in a public repository.

Keywords: social navigation, graph neural networks, human-robot interaction

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 a considerably diverse set of perspectives. Pioneering works such as [2] started taking into account the personal spaces of the people surrounding the robots, often referred to as proxemics. In addition to proxemics, human motion patterns were analysed in [3] to estimate whether humans are willing to interact. Semantic properties were also considered in [4]. Although not directly applied to navigation, the relationships between humans and objects were used in the context of ambient intelligence in [5]. Proxemics and object affordances were jointly considered in [6] for navigation purposes. Two extensive surveys on human-aware navigation can be found in [7] and [8].

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 model-based approaches over data-driven ones. Besides the mentioned model scalability and explainability issues, model-based 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 (see [9]) are learned in [10] and [11] to navigate in human-populated environments. Inverse reinforcement learning is used in [12] and [13] to plan navigation routes based on a list of humans in a radius. Social norms are implemented using deep reinforcement learning in [14], again, considering a set of humans. An approach modelling crowd-robot interaction and navigation control is presented in [15]. 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 [16] 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 used.
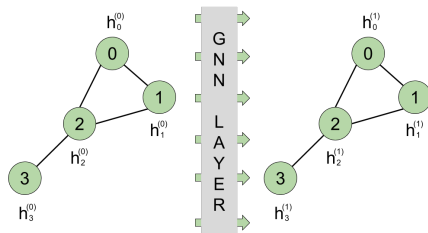
All the previous works and many others not mentioned have achieved outstanding results. Some model-based approaches such as [4] or [6] can leverage structured information to take into account space affordances. Still, the data considered to make such decisions are often handcrafted features based on an arbitrary subset of the data that a robot would be able to work with. There are many reasons motivating to seek a learning-based approach not requiring feature handcrafting or manual selection. Their design is time-consuming and often requires a deep understanding of the particular domain (see discussion in [17]). 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 big 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, [18] proposes Graph Neural Networks (GNNs) as a means to perform learning that allows combining raw data with hand-engineered features, and most importantly, 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 makes possible

integrating new social cues in a straightforward fashion, by including more data in the graphs that they are fed.

In this paper, we use different GNN models to estimate social navigation compliance, *i.e.*, given a robot pose and a scenario where humans and objects can be interacting, estimating to what extent a robot would be disturbing the humans if it was located in such a pose. 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 have different relations among them. For example, social robots can have information about who a human is talking to, where people are looking at, who is friends with who, 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

Graph Neural Networks (GNNs) are a family of machine learning approaches based on neural networks that take graph-structured data as input. They allow classifying and making regressions on graphs, nodes, edges, as well as predicting link existence when working with partially observable phenomena. Except for few exceptions (*e.g.*, [19]) 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).

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, to that the output of any of the nodes in the last layer can used for classification or regression purposes.

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 [20]. GNNs were further studied and formalised in [21] and [22]. However, it was with the appearance of Gated Graph Neural Networks (GG-NNs, [23]) and especially Graph Convolutional Networks (GCNs, [24]) that GNNs gained traction. The work presented in [18] reviewed and unified the notation used in the GNNs existing to the date.

Graph Convolutional Networks (GCN) [24] are 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 in general. Following the notation proposed in [18], 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 is a set of edges $E = \{(s_k, r_k)\}_{k=1:N^e}$, 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:

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

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

For every node v_i, the first function ($\rho^{e \to v}(E)$) aggregates the feature vectors of other nodes with an edge towards it and generates a temporary aggregated feature $\bar{e}_i$ which is used by the second function. In a second pass, the function $\phi^v(\bar{e}_i, v_i)$ is used to generate updated v_i' feature vectors from the aggregated feature vectors using a neural network (usually a multi-layer perceptron, but the framework does not make any assumption on this). Such a learnable function is 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 (RGCNs [25]) extends GCNs by considering different types of edges separately and applies the resulting model to vertex classification and link prediction. Graph Attention Networks (GATs [26]) extend GCNs by adding self-attention mechanisms (see [27]) and applies the resulting model to vertex classification. For a more detailed review of GNNs and the generalised framework, please refer to [18].

3   Formalisation of the problem

The aim of this work is to analyse the scope of GNNs in the field of human-aware social navigation. Our study has been set up using the *SocNav1* dataset (see [1]) which provides social compliance labels for specific scenarios. It 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. Each sample is labelled with a value from 0 to 100 depending on to what extent the subjects that labelled the scenarios considered that the robot is disturbing the people in the scenario. The dataset provides 16336 labelled samples to be used for training purposes, 556 scenarios as the development dataset and additional 556 for final testing purposes.

As previously noted, GNNs are a flexible framework that allows working somewhere in the middle of end-to-end and feature engineered learning. Developers can use as many data features as desired and are free to structure the input graph data as they please. The only limitations are those of the particular GNN layer blocks used. In particular, while GCN and GAT do not support labelled edges, RGCN and GG-NN do. To account for this limitation, two automated scenario-to-graph transformations were used in the experiments, depending on the GNN block to be tested: one without edge labels and one with them.

The first version of the scenario-to-graph transformation used to represent the scenarios does not use labelled edges. It uses 6 node types (the features associated to each of the types are detailed later in this section):
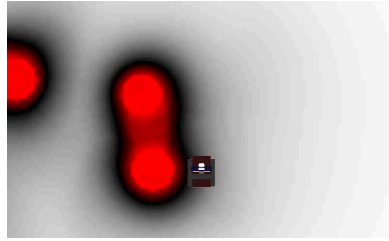
- **robot:** The dataset only includes one robot in each scenario, so there is just one robot symbol in each of the graphs. However, GNNs do not have such restriction.
- **wall:** A node for each of the segments defining the room. They are connected to the room node.
- **room:** Used to represent the room where the robot is located. It is connected to the robot.
- **object:** A node for each object in the scenario.
- **human:** A node for each human. Humans might be interacting with objects or other humans.
- **interaction:** An interaction node is created for every human-to-human or human-to-object interaction.

Figure 2 depicts two areas of a scenario where four humans are shown in a room with several objects. Two of the humans are interacting with each other, another human is interacting with an object, and the remaining human is not engaging in interaction with any human or object. The structure of the resulting non-labelled graph is shown in Fig.3a.
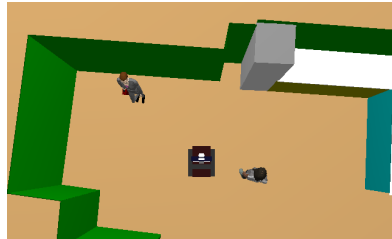
The features used for **human** and **object** nodes are: distance, the relative angle from the robot's point of view, and its orientation, from the robot's point of view too. For **room** symbols the features are: the distance to the closest human and the number of humans. For the **wall** segments and the **interaction** symbols, the features are the distance and orientation from the robot's frame of reference. For wall segments, the position is the centre of the segment and the orientation is the tangent. These features geometrically define the room, which is relevant to characterise the density of people in the room and the distance from the robot to each of the walls. For interactions, the position is the midpoint between the interacting symbols, and the orientation is the tangent of the line connecting the endpoints. Features related to distances are expressed in meters,
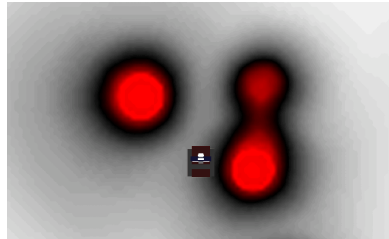
(a) An area of a scenario where two humans interacting in a room are depicted.

(b) Heat map of the social compliance estimation for the area shown in Fig. 2a in the different positions in the environment.

(c) An area of the same scenario with two humans. The human on the left is not engaged in any interaction. The human on the right is interacting with the object in front of her.
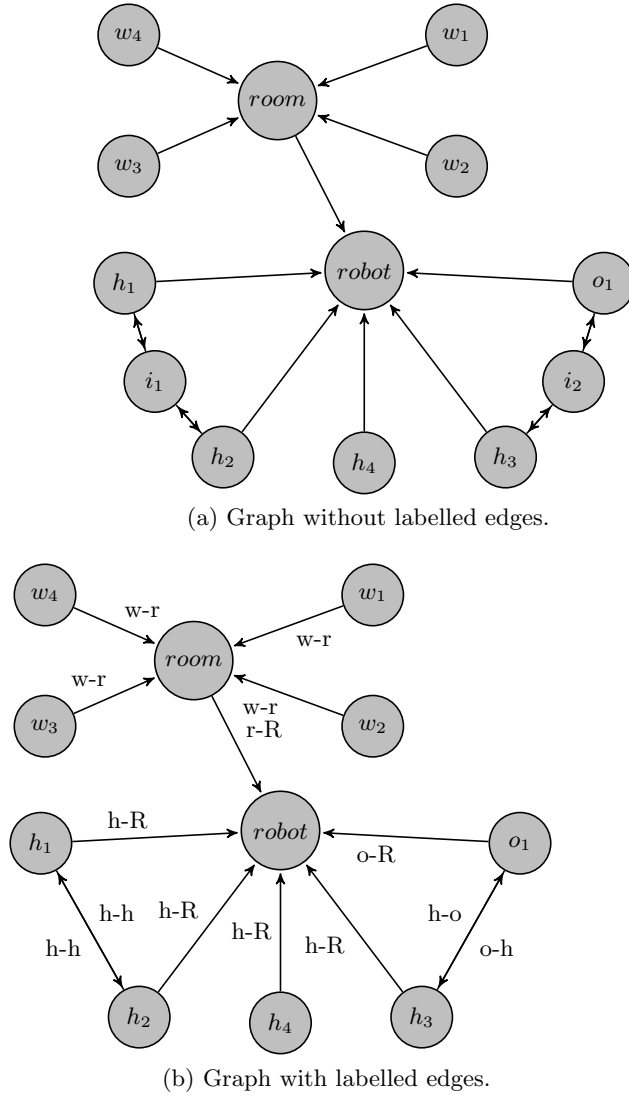

(d) Heat map of the social compliance estimation for the area shown in Fig. 2c in the different positions in the environment.

Fig. 2. Different areas of a scenario where social interactions are being held and their corresponding estimated heat map of *"social inconvenience"*.

whereas those related to angles are actually expressed as two different numbers, $sin(\alpha)$ and $cos(\alpha)$. The final features of the nodes are built by concatenating the one-hot encoding that determines the type of the symbol and the features for the different node types. It is worth noting that by building feature vectors this way, their size increases with every new type. This limitation is currently being studied by the GNN scientific community.

For the GNN blocks that can work with labelled edges, a slightly different version of the scenario-to-graph transformation is used. The first difference is that in this version of the scenario-to-graph model there are no interaction nodes. The elements interacting are linked to each other directly. Robot, room, wall, human and object nodes are attributed with the same features as in the previous model. The second difference is related to the labelling of the edges. In this domain, the semantics of the edges can be inferred from the types of the nodes being connected. For example, *wall* and *room* nodes are always connected by the same kind of relation "composes". Similarly, *humans* and *object* nodes are always connected by the relation "interacts_with_human". The same holds the other

(a) Graph without labelled edges.

(b) Graph with labelled edges.

Fig. 3. Examples of how the scenario-to-graph transformation work based on the scenario depicted in Fig.2.

way around: "composes" relations only occur between *wall* and *room* nodes, and "interacts_with_human" relations only occur with *humans* and *object* nodes. Therefore, for simplicity, the label used for the edges is the concatenation of the types involved. The structure of the resulting labelled graph for the scenario depicted in Fig.2 is shown in Fig.3b.

Because all nodes are connected to the robot, the GNNs were trained to perform the regression on the feature vector of the *robot* node in the last layer.

Using the previously mentioned dataset and the two proposed scenario-to-graph transformations, different architectures using different GNN blocks were compared.

4   Experimental results

The experimental results are aligned with the contributions of the paper, which deal with modelling social conventions. Therefore, we assume that we can build on top of a third party body tracker and a path planning system and proceed with the evaluation of the algorithm against the dataset.

The four GNN blocks used in the experiments were:

 – Graph Convolutional Networks (GCN) [24].
 – Gated Graph Neural Networks (GG-NN) [23].
 – Relational Graph Convolutional Networks (RGCNs) [25].
 – Graph Attention Networks (GAT) [26].

If available, each of these GNN blocks was benchmarked using Deep Graph Library (DGL) [28] and PyTorch-Geometric (PyG) [29]. In addition to using several layers of the same GNN building blocks, alternative architectures combining RGCN and GAT layers were also tested:

1. A sequential combination of n RGCN layers followed by n GAT layers with the same number of hidden units (alternative 8 in table 1).
2. An interleaved combination of n RGCN and n GAT layers with the same number of hidden units (alternative 9 in table 1).
3. A sequential combination of n RGCN layers followed by n GAT layers with a linearly decreasing number of hidden units (alternative 10 in table 1).

As a result, 10 framework-architecture combinations were benchmarked. Table 1 describes them and provides their corresponding performance on the development dataset. To benchmark the different architectures, 5000 training sessions were launched using the SocNav1 training dataset and evaluated using the SocNav1 development dataset. The hyperparameters were randomly sampled from the range values shown in Table 2.

The results obtained (see table 1) show that, for the dataset and the framework/architecture combinations benchmarked, DGL/GAT delivered the best results, with a training loss of 0.01701 for the development dataset. The parameters used by the DGL/GAT combination were: batch size: 273, number of hidden units: 129, number of attention heads: 2, number of attention heads in the last layer: 3, learning rate: 5e-05, weight decay regularisation: 1e-05, number of layers: 4, no dropout, alpha parameter of the ReLU non-linearity 0.2114. After selecting the best set of hyperparameters, the network was compared with a third *test* dataset, **obtaining an MSE of 0.03173**. Figures 2b and 2d provide an intuition of the output of the network for the scenarios depicted in figures 2a

Alt. #	Framework	Network architecture	Training Loss (MSE)
1	DGL	GCN	0.02283
2	**DGL**	**GAT**	**0.01701**
3	DGL	GAT2	0.01740
4	PyG	GCN	0.29778
5	PyG	GAT	0.01804
6	PyG	RGCN	0.02827
7	PyG	GG-NN	0.02718
8	PyG	RGCN‖GAT 1	0.02238
9	PyG	RGCN‖GAT 2	0.02147
10	PyG	RGCN‖GAT 3	0.0182

Table 1. A description of the different framework/architecture combinations and the experimental results obtained from their benchmark for the SocNav1 dataset.
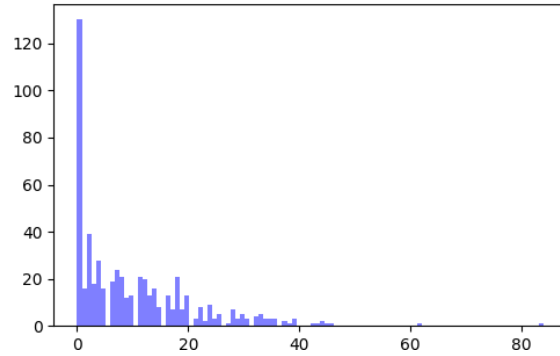
Hyperparameter	Min	Max
epochs	1000	
patience	5	
batch size	100	1500
hidden units	50	320
attention heads	2	9
learning rate	1e-6	1e-4
weight decay	0.0	1e-6
layers	2	8
dropout	0.0	1e-6
alpha	0.1	0.3

Table 2. Hyperparameter values. Only applicable to Graph Attention Network blocks.

and 2c considering all the different positions of the robot in the environment when looking along the y axis.

It is worth noting that, due to the subjective nature of the labels 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 asked 4 subjects to label all the scenarios of the development dataset. The mean MSE obtained for the different **subjects was 0.022**. This means that the network performs close to human accuracy. Figure 4 shows an histogram comparing the error made by the GNN-based regression in comparison to humans.

Most algorithms presented in section 1 deal with modelling human intimate, personal, social and interaction spaces instead of social inconvenience, which seems to be a more general term. Keeping that in mind, the algorithm proposed in [6] was tested against the test dataset and got a MSE of 0.12965. The relative bad performance can be explained by the fact that other algorithms do not take

Fig. 4. Histogram of the absolute error in the test dataset for the network performing best in the development dataset.

into account walls and that their actual goal is to model personal spaces instead of feelings in general.

Regarding the effect of the presence of humans, we can see from Fig. 2d that the learnt function is slightly skewed to the front of the humans, but not as much as modelled in other works such as [30] or [6]. One of the possible reasons why the "personal space" is close to being circular is the fact that, in the dataset, humans appear to be standing still. It is still yet to be studied, probably using more detailed and realistic datasets, how would the personal space look like if humans were moving.

The results obtained using GNN blocks supporting edge labels were inferior to those obtained using GAT, which does not support edge labels. Two reasons might be the cause of this phenomena: **a)** as mentioned in section 3 the labels edges can be inferred from the types of the nodes, so that information is to some extent redundant; **b)** the inductive bias of GATs is strong and appropriate for the problem at hand. This does not mean that the same results would be obtained in other problems where the label of the edges cannot be inferred.

5   Conclusions

Although there have been attempts to predict people's trajectories using graph neural networks, to our knowledge, this paper presented the first graph neural network modelling human-aware navigation conventions. The scenario-to-graph transformation model and the graph neural network developed as a result of the work presented in this paper achieved a performance comparable to that of humans. The final model can be used as part of a social navigation system to predict the degree of acceptance of the robot position according to social conventions. Thus, given the graph representation of a scenario, the GNN provides a quantification of how good a specific robot location is for humans. Even though the results achieved are remarkable, the key fact is that this approach allows to

include more relational information. This will allow to include more sources of information in our decisions without a big impact in the development.

There is room for improvement, particularly related to: **a)** personalisation (different people generally feel different about robots), and **b)** movement (the inconvenience of the presence of a robot is probably influenced by the movement of the people and the robot). Still, we include interactions and walls, features which are seldom considered in other works. As far as we know, interactions have only been considered in [6] and [31].

The code to test the resulting GNN has been published in a public repository as open-source software [4], as well as the code implementing the scenario-to-graph transformation and the code to train the models suggested.

References

1. Manso, L.J., Nuez, P., Calderita, L.V., Faria, D.R., Bachiller, P.: Socnav1: A dataset to benchmark and learn social navigation conventions. Data **5**(1) (2020). URL https://www.mdpi.com/2306-5729/5/1/7
2. Pacchierotti, E., Christensen, H.I., Jensfelt, P.: Human-robot embodied interaction in hallway settings: A pilot user study. In: IEEE International Workshop on Robot and Human Interactive Communication, vol. 2005, pp. 164–171. IEEE (2005). DOI 10.1109/ROMAN.2005.1513774
3. Hansen, S.T., Svenstrup, M., Andersen, H.J., Bak, T.: Adaptive human aware navigation based on motion pattern analysis. Proceedings - IEEE International Workshop on Robot and Human Interactive Communication pp. 927–932 (2009). DOI 10.1109/ROMAN.2009.5326212
4. Cosley, D., Baxter, J., Lee, S., Alson, B., Nomura, S., Adams, P., Sarabu, C., Gay, G.: A Tag in the Hand: Supporting Semantic, Social, and Spatial Navigation in Museums. Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI'09) pp. 1953–1962 (2009). DOI 10.1145/1518701.1518999
5. Bhatt, M., Dylla, F.: A Qualitative Model of Dynamic Scene Analysis and Interpretation in Ambient Intelligence Systems. International Journal of Robotics and Automation **24**(3), 1–18 (2010). DOI 10.2316/journal.206.2009.3.206-3274
6. Vega, A., Manso, L.J., Macharet, D.G., Bustos, P., Núñez, P.: Socially aware robot navigation system in human-populated and interactive environments based on an adaptive spatial density function and space affordances. Pattern Recognition Letters **118**, 72–84 (2019). DOI 10.1016/j.patrec.2018.07.015
7. Rios-Martinez, J., Spalanzani, A., Laugier, C.: From Proxemics Theory to Socially-Aware Navigation: A Survey. International Journal of Social Robotics **7**(2), 137–153 (2015). DOI 10.1007/s12369-014-0251-1
8. Charalampous, K., Kostavelis, I., Gasteratos, A.: Recent trends in social aware robot navigation: A survey, vol. 93, pp. 85–104. Elsevier B.V. (2017). DOI 10.1016/j.robot.2017.03.002
9. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Review E **51**(5), 4282–4286 (1995). DOI 10.1103/PhysRevE.51.4282

---

[4] https://github.com/robocomp/sngnn

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

11. Patompak, P., Jeong, S., Nilkhamhang, I., Chong, N.Y.: Learning Proxemics for Personalized Human-Robot Social Interaction. International Journal of Social Robotics (2019). DOI 10.1007/s12369-019-00560-9

12. Ramon-Vigo, R., Perez-Higueras, N., Caballero, F., Merino, L.: Transferring human navigation behaviors into a robot local planner. 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 (2014). DOI 10.1109/ROMAN.2014.6926347

13. Vasquez, D., Okal, B., Arras, K.O.: Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison. IEEE International Conference on Intelligent Robots and Systems (Iros), 1341–1346 (2014). DOI 10.1109/IROS.2014.6942731

14. Chen, Y.F., Everett, M., Liu, M., How, J.P.: Socially aware motion planning with deep reinforcement learning. IEEE International Conference on Intelligent Robots and Systems **2017-Septe**, 1343–1350 (2017). DOI 10.1109/IROS.2017.8202312

15. Chen, C., Liu, Y., Kreiss, S., Alahi, A.: Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning. In: International Conference on Robotics and Automation (ICRA), pp. 6015–6022. IEEE (2019). URL http://arxiv.org/abs/1809.08835

16. Martins, G.S., Rocha, R.P., Pais, F.J., Menezes, P.: Clusternav: Learning-based robust navigation operating in cluttered environments. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 9624–9630. IEEE (2019)

17. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). DOI 10.1038/nature14539

18. 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.: Relational inductive biases, deep learning, and graph networks. arXiv: 1806.01261 pp. 1–40 (2018). DOI 10.1017/S0031182005008516. URL http://arxiv.org/abs/1806.01261

19. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems, pp. 4800–4810 (2018)

20. Sperduti, A., Starita, A.: Supervised Neural Networks for the Classification of Structures. IEEE Transactions on Neural Networks **8**(3), 1–22 (1997)

21. Gori, M., Monfardini, G., Scarselli, F.: A new Model for Learning in Graph domains. Proceedings of the International Joint Conference on Neural Networks **2**, 729–734 (2005). DOI 10.1109/IJCNN.2005.1555942

22. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The Graph Neural Network Model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009). DOI 10.1109/TNN.2008.2005605

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

24. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 pp. 1–14 (2016). URL http://arxiv.org/abs/1609.02907

25. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: 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 (2018). DOI 10.1007/978-3-319-93417-4_38

26. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. In: Proceedings of the International Conference on Learning Representations 2018, 2015, pp. 1–11 (2018). URL http://arxiv.org/abs/1710.10903

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

28. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Ma, C., Huang, Z., Guo, Q., Zhang, H., Lin, H., Zhao, J., Li, J., Smola, A., Zhang, Z.: Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds (RLGM) pp. 1–7 (2019)

29. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

30. Kirby, R., Simmons, R., Forlizzi, J.: Companion: A constraint-optimizing method for person-acceptable navigation. In: RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication, pp. 607–612. IEEE (2009)

31. Cruz-maya, A.: Enabling Socially Competent navigation through incorporating HRI. arXiv: 1904.09116v1 pp. 9–12 (2019)