Multi-camera Torso Pose Estimation using Graph Neural Networks

Daniel Rodriguez-Criado[1], Pilar Bachiller[2], Pablo Bustos[2], George Vogiatzis[1], and Luis J. Manso[1]

***Abstract*— Estimating the location and orientation of humans is an essential skill for service and assistive robots. To achieve a reliable estimation in a wide area such as an apartment, multiple RGBD cameras are frequently used. Firstly, these setups are relatively expensive. Secondly, they seldom perform an effective data fusion using the multiple camera sources at an early stage of the processing pipeline. Occlusions and partial views make this second point very relevant in these scenarios. The proposal presented in this paper makes use of graph neural networks to merge the information acquired from multiple camera sources, achieving a mean absolute error below $125mm$ for the location and $10°$ for the orientation using low-resolution RGB images. The experiments, conducted in an apartment with three cameras, benchmarked two different graph neural network implementations and a third architecture based on fully connected layers. The software used has been released as open-source in a public repository[1].**

I. INTRODUCTION

Autonomous robots have a wide range of applications, including performing daily chores for an ageing population and carrying out tasks that might be dangerous for humans. To work seamlessly among humans, robots need social skills, for instance, to not to get in the way, or to understand people's intentions and communicate their own. Among other relevant information such as gestures or facial expressions, people's position and orientation are among the most important cues that can help service and assistive social robots understand humans. A common application of human localisation and orientation is predicting intentions and movements in surveillance video feeds [1], [2]. An accurate localisation and orientation estimation is also crucial for human-aware navigation [3]. For instance, the orientation of pedestrians' velocity vectors is used in [4] to make a robot navigate in crowded environments complying with constraints defined by proxemics.

Although there is a considerable number of exceptions (*e.g.*, [5]–[7]) orientation and other social cues are usually acquired using a two stage pipeline: human body parts are detected as a first step and then passed as input to a second stage algorithm. This second algorithm is frequently implemented using basic trigonometry, considering the coordinates of the shoulders or the hips [1]. For instance, in [8], it is calculated using the cross-product of the vectors going from the head to the right and left sides of the hip, respectively.

[1]Daniel Rodriguez-Criado, George Vogiatzis and Luis J. Manso are with School of Engineering and Applied Science, Computer Science Department, Aston University, United Kingdom. l.manso@aston.ac.uk

[2]Pilar Bachiller and Pablo Bustos are with Robotics and Artificial Vision Laboratory, Caceres School of Technology, Universidad de Extremadura, Extremadura.

[1]https://github.com/vangiel/WheresTheFellow

To overcome the poor behaviour that handcrafted equations tend to have when working with missing, noisy and redundant data, some works follow a machine learning-based approach. For instance, [5], and [6] use Histograms of Oriented Gradients (HOGs). In [5] RGBD HOGs were used to provide discrete angle estimations. The work presented in [7] uses both RGBD and IR images with IR trackers to train a single-camera Convolutional Neural Network (CNN). Their model provides continuous angle estimation, achieving a mean absolute error close to 6°. The accuracy of the work at hand for torso orientation is below that of [7]. However, this proposal does not only estimate the orientation but also the 3D coordinates of the torsos and it does not require the use of relatively expensive RGBD cameras. To do this, our work builds on top of a skeleton detector. To the best of our knowledge, there are not any application using GNNs for predicting pedestrian's orientation.

There are numerous works on human detection. Pioneering works such as [9] or [10] took RGB images as input. With the advent of RGBD cameras, different alternatives were made available in the early 2010s [11], [12]. The additional depth channel made these algorithms less sensitive to illumination changes and made some tasks such as segmentation more approachable. Nevertheless, they had important limitations when applied to robotics such as a low accuracy distinguishing the left and right sides of humans in specific angle ranges and poor performance on moving cameras [13]. Many works have been recently published using CNNs to address some of the limitations of previous approaches. OpenPose [14] became state-of-the-art detecting body parts using Part Affinity Fields to learn the association between body parts and humans in the image. However, its performance deteriorates as resolution decreases and does not work as well in crowded environments with occluded body parts. OpenPifPaf [15] was proposed to solve OpenPose's limitations. It uses a CNN with two heads; the first to locate the joints and the second to predict associations between them, including occluded parts.

II. MULTI-CAMERA TORSO POSE ESTIMATION

The problem we deal with is that of estimating the pose of a person from a set of cameras. The pose is defined as their position on the floor plane and their orientation with respect to the vertical axis: (x, y, α). The system should be able to cover spaces wide enough to require several cameras attached to the walls with overlapping fields of view. The setup used in the experiments is composed of 3 Intel RealSense 415 depth cameras whose extrinsic parameters are calibrated with respect to a common reference system on the floor (RGBD

cameras are used to allow comparing results using RGB and RGBD images). This paper uses both real and synthetic training data to generate a large dataset in a very short time, saving a great amount of resources. Figure 1 shows a representation in CoppeliaSim [16] of three views of the environment where the system has been tested. As shown below, once the model is trained it can estimate 3D poses using only the joints' image coordinates, not requiring depth data.

The processing pipeline has three main stages: first, images are acquired and processed using OpenPifPaf [15]. The output data of this stage -a set of detected skeletons from the different cameras- is passed to the next stage, where skeletons corresponding to the same person are matched and grouped. These groups are then provided to a Graph Neural Network (GNN) which provides the final output. The remainder of this section explains the stages in more detail.

Image acquisition and skeleton detection: The images acquired are provided to OpenPifPaf to get the skeleton data. For each frame, an observation $\Psi\left\{p_i, r_i, t_i\right\}$ is generated and provided to the next stage, where:

- p_i is the people detected in that frame, each of which holds a list of up to 16 joint's coordinates. If using RGBD cameras, each joint's depth from the camera is computed using the depth plane.
- r_i is the RGB Region of Interest (ROI) corresponding to the bounding box of the skeleton.
- t_i is the acquisition time of the frame.

Match observations to people: a stream of $\Psi_t,\ t \in \mathbb{N}$, observations are generated from the skeleton detectors. A state machine manages the creation, update and removal of a set of data objects representing people. Each observation can either create a new person, update an existing one or be dismissed as noise. Before a new person is accepted, it has to receive successful matches for at least 2 seconds. An observation matches an existing person if their distance $d(o_i, p_j)$ is lower than a certain threshold d_{max}, taken here as 0.65 in the $[0, 1]$ range. The distance is defined as the median of the distances between the observation and the recent history of the person: $d = med_t\left\{B\left(h(o_i), h_t(p_j)\right)\right\}$ where B is the Bhattacharyya distance [17], $h(o_i)$ is the observation's 2D histogram computed over the hue and saturation planes of the person's ROI and $h_t(p_j)$ is the 2D histogram of person p_j at time t, where t goes from the last observation to Q samples in the past. Other distances have been tested with no better results. The removal of unseen people occurs after 2 seconds without receiving any matches.

In the next stage, a set S of observations from a person is fed to the GNN to obtain a tuple of target coordinates (x, y, α) representing the pose for each torso. This set S is extracted from the person's history as:

$$S \coloneqq \left\{\, o \mid o_i^k \in O^p \ \wedge k \in \{1, 2, 3\} \,\right\}$$

where o_i^k is the past matched observation i by camera k and O^p is the set of past time-ordered observations of person p. S is thus the set of the most recent matched observations obtained from different cameras.

GNN processing: The models are designed to use the information obtained from each camera to estimate the position and orientation of a human even with a partial view, obtaining more accurate results when more data is available. As can be observed in Fig. 2, the total number of visible joints is limited in some cases, which makes it hard to estimate the position and orientation of the person using analytical methods.

GNNs adapt particularly well to structured data of varying size and missing nodes (body parts in this case) [18]. Among the different GNNs variations, Graph Convolutional Networks (GCNs) [19] are one of the easiest to understand, and many other build on top of GCNs. They generalise the concept of learned convolutions to graphs. They are similar to CNNs in the sense that they learn convolutions, but instead of working on images, GNNs work on graphs. Equation 1 describes how the output feature vector $h_i^{(l+1)}$ of a node i on layer $l+1$ is computed.

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in IN(i)} \frac{1}{C_{ij}} W^{(l)} h_j^{(l)}\right) \tag{1}$$

In equation 1, $IN(i)$ is the set of nodes j so that an edge (j, i) exists in the graph, $W^{(l)}$ is the trainable weight matrix for the layer l, $\sigma\left(\cdot\right)$ is the activation function and C_{ij} is as normalisation parameter.

Relational Graph Convolutional Networks (RGCNs) [20] build on top of GCNs, allowing labelled edges by using a different learnable weight matrix for each label type. The propagation model for the feature vector of the node i is shown in equation 2.

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{r \in N_i^r} \frac{1}{C_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right) \tag{2}$$

where $W_r^{(l)}$ and $W_0^{(l)}$ are the learnable matrices for r-labelled edges and self-edges, respectively.

Graph Attention Networks (GATs) [21] introduce self-attention to GNNs. A simplified node propagation function for the single-headed version of GAT can be seen in equation 3.

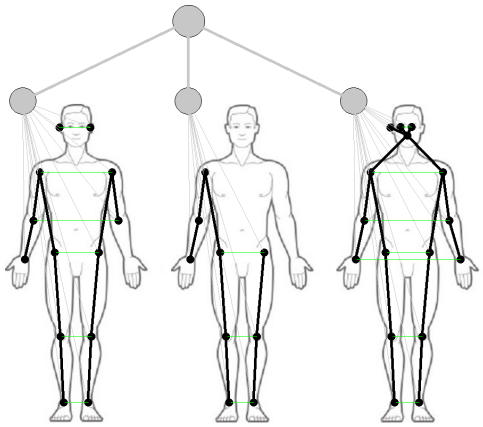
$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N_i} \alpha_{ij}^{(l)} W^{(l)} h_j^{(l)}\right) \tag{3}$$

In this case, the feature vector of node i is updated from the neighbouring nodes weighted by a learnable attention parameter α.

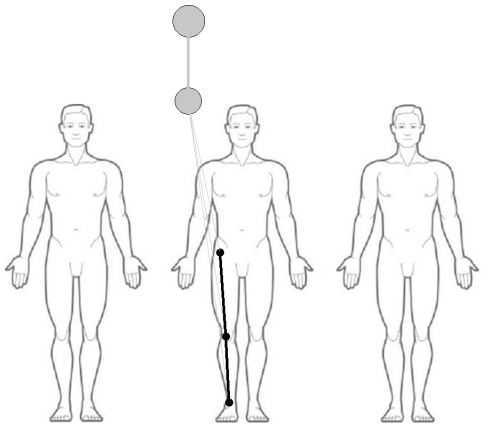
An example of the extraction of information of body parts by a GNN can be seen in [22], where the features of the hand are encoded in a graph of different points and a GCN yields the hand gesture. Similarly, [23] uses the coordinates of the human skeleton joints as the input of a GCN to recognise actions performed on videos.

In the present work, an input graph is created for each set of skeletons belonging to the same person. First, the joints detected by each camera are used to create a separate graph corresponding to a different view of the same

Fig. 1. Example images obtained from the simulator. In the experiments, the resolution used was 640x480.



(a) Person detected by the three cameras.



(b) Person partially detected by a single camera.

Fig. 2. Examples of graph representation of the information obtained by the multi-camera system.

skeleton. These graphs have a node representing the body and additional nodes for all the body parts available (as provided by OpenPifPaf), connecting each part not only to its kinematic parent but also to its mirrored body part. The nodes representing the body are referred to as *body nodes* in this paper. Finally, all body nodes available (one per view) are connected to an additional node aggregating the information from the previous nodes; we call that node the *superbody* node. Fig. 2 depicts two graphs with the body parts captured by our three-camera setup.

The feature vectors of the nodes have 25 dimensions (28 if using RGBD cameras). The feature vector for each node i is built by concatenating one-hot encodings and metric information:

$$h_i^{(0)} = (t_i|c_i|p_i|s_i)$$

- Node type one-hot encoding (t_i): It encodes the node type. It has a length of 19 because OpenPifPaf can detect 17 body parts and there are two additional types for the *body* and *superbody* nodes.
- Camera one-hot encoding (c_i): This one-hot encoding represents the camera that captures the skeleton. In our experiments it has a length of 3, as that is the number of cameras used. It is zero-filled for the *superbody* node.
- Coordinates vector (p_i): It has a length of 2 (5 if using RGBD cameras) and stores the coordinates of each body part. *Body* and *superbody* nodes have zeros in all the elements of this vector. The image coordinates provided by OpenPifPaf are normalised so that they are within the range $[-1, 1]$. If using 640x480 cameras, the normalisation would be:

$$p'_x = (x - 320)/320$$
$$p'_y = (240 - y)/240$$

  The 3D coordinates are only provided if using RGBD cameras. They are also normalised to be in the range $[-1, 1]$, based on the size of the room.
- Score (s_i): This is a single element field that provides the certainty of the measure gathered by OpenPifPaf. It is only used for body part nodes, zero otherwise.

The model is trained so that the output feature vectors are 4-dimensional and correspond to x, y, $sin(\alpha)$ and $cos(\alpha)$ for the *superbody* node in the last layer. The actual angle α is then reconstructed from its sine and cosine.

III. Dataset generation

As the models are scenario-specific, they have to be trained with simulations run with the camera calibration information. Datasets can be built using any simulator that can animate avatars, provide their ground-truth positions, and provide RGB(D) streams so that OpenPifPaf or a similar software can be used to detect people and their skeletons. To create a proper virtual replica, the intrinsic and extrinsic parameters must be estimated. The software released uses an Augmented Reality (AR) tag placed on the floor (so that it can be detected by multiple cameras) and guides users through the calibration process.

Once the cameras are calibrated, new datasets can be easily produced generating paths for the simulated avatars in the virtual model. Using this procedure, a big amount of data can be gathered with a limited effort. Nevertheless, simulated data can be insufficient depending on the accuracy of the calibration because small calibration errors can lead to a significant reduction of accuracy in the estimation. For this reason, the dataset generated from the simulations can be extended with real data, recording an actual human moving around the environment. In our experiments, to store ground-truth information, the human was equipped with an Intel RealSense tracking camera in their chest, which provides under 1% closed loop drift. The camera pose at the start coincides with the global reference frame of the room. Thus, the pose of the camera directly corresponds to the ground truth pose Combining both, simulated and real data, a final training dataset with more than 20,000 samples was created. Specifically, 19833 samples of the dataset are simulated and 631 are real. The final dataset is provided in JSON format (available in the public repository[2]).

It is worth mentioning that calibration is only necessary to build a replica of the real world in the simulator. If the dataset is composed only of real data, calibration can be avoided.

IV. EXPERIMENTAL RESULTS

To evaluate the proposed multi-camera human torso pose estimation system several experiments were conducted. Each experiment was carried out using three different architectures: 1) a sequence of GAT layers, 2) a sequence of RGCN layers, and 3) a sequence of per-camera fully connected layers (FC) (shared across cameras) followed by concatenation and a further sequence of FC layers (MLP). For the MLP architecture, a parallel input vector of 0s and 1s was provided alongside the normal input, to indicate missing data.

The three architectures were trained using the dataset described in section III (**DS1**), applying different combinations of hyperparameters to select the best ones. In addition, a second training dataset (**DS2**) including only simulated data was generated. The three architectures were also trained using this second dataset following the same process of hyperparameter tuning. For GNNs, different values for the number of layers, number of hidden units, attention heads (for GAT only), number of bases (for RGCN only) and activation function of each layer were tested. These values were randomly generated to cover a wide range of combinations Similarly, for the MLP, we explored various depths and widths of the hidden layers.

Besides the two training datasets, two additional datasets were generated from real data: one for development composed of 225 samples and another one with 283 samples for testing purposes. Larger datasets would have been collected, but it was not possible due to the COVID-19 pandemic.

Each sample of the datasets corresponds to a graph representing the view of a human from the cameras at a given instant. Since RGBD cameras were available, for each perceived joint both 2D image coordinates and 3D positions were available. Nevertheless, in order to make RGBD cameras optional, each architecture was trained using two versions of the data (with and without 3D information). The first version includes the 3D and image coordinates of the body parts in the feature vectors. The second one considers only the image coordinates, so that it can be applied to multi-camera systems composed of RGB cameras.

Table III shows the Mean Squared Error (MSE) of the test dataset for the best model of each architecture using the 2D-only and 3D versions of data and the two training datasets. As can be observed, the two GNN architectures provide better results than MLP for all the combinations of training datasets and types of features. As expected, the use of a training dataset including only simulated data (DS2) produces a loss of accuracy in all the cases, with a more significant effect in MLP. This fact becomes more evident with the 2D version of the data, affecting the estimation of both, orientation (see table I: MSE considering only the orientation-related outputs of the network) and position (see table II: MSE for position-related outputs). Nevertheless, for the training dataset combining simulated and real data (DS1), the use of 2D-only features does not have a big impact on the results.

TABLE I
ORIENTATION MSE FOR THE TWO TRAINING DATASETS

	MLP	**RGCN**	**GAT**
DS1 - 3D features	0.024	0.018	0.019
DS1 - 2D features	0.026	0.02	0.021
DS2 - 3D features	0.083	0.080	0.038
DS2 - 2D features	0.077	0.058	0.066

TABLE II
POSITION MSE FOR THE TWO TRAINING DATASETS

	MLP	**RGCN**	**GAT**
DS1 - 3D features	0.0011	0.00079	0.00092
DS1 - 2D features	0.0012	0.0016	0.0011
DS2 - 3D features	0.0057	0.0021	0.0013
DS2 - 2D features	0.010	0.0064	0.0049

TABLE III
GLOBAL MSE FOR THE TWO TRAINING DATASETS

	MLP	**RGCN**	**GAT**
DS1 - 3D features	0.010	0.0076	0.0083
DS1 - 2D features	0.011	0.009	0.009
DS2 - 3D features	0.037	0.033	0.016
DS2 - 2D features	0.037	0.027	0.029

To test the accuracy of the solutions, the output was compared with an analytical estimation of the human pose based on the depth data. The position and orientation of the human in the analytical estimation are computed as follows:

- Estimated position: for each camera, the position is individually estimated as the median of the positions of the joints. The final position is computed by the average

[2]https://github.com/vangiel/WheresTheFellow

of the estimations of all the cameras perceiving at least three joints of the human.
- Estimated orientation: for each camera, the orientation is computed from the positions of pairs of symmetric joints. Specifically, the shoulders and hips are used. The final orientation is obtained as the average of the estimations of the different cameras. If none of the cameras perceives a symmetric pair of joints, the estimation of the previous instant is maintained.

The comparison between the results obtained from the analytical and the learnt estimators shows how the learning-based solutions outperform the analytical methods, more notably if they have access to the depth channel of the images too, especially for the orientation. This can be observed in figure 3, which depicts the estimation of the position and the orientation for every sample of the test dataset using the analytical method (red dotted line) and the RGCN-based architecture trained using 3D data with the dataset DS1 (blue dashed line). As can be seen, the difference with the ground-truth (green solid line) is smaller in the estimation obtained by the GNN, which is especially remarkable in the angle prediction.

This observation can be extended to the remaining architectures trained with DS1 and 3D features (figure 4). Thus, the mean absolute error (MAE) of the neural estimation indicates a considerable improvement regarding the analytical method, providing a mean absolute angle error under 10° in the best GNN architecture.

The exclusive use of simulated data for training produces a deterioration of the results as can be seen in figure 5. However, the application of the 3D version of the data still outperforms the analytical estimation for one of the GNN architectures in position and orientation.

V. CONCLUSIONS

Our human pose estimation system has been designed for spaces that are covered by multiple cameras. It is assumed that people are visible by at least one of the cameras, although some parts of their bodies can be occluded or outside of the field of vision of some of the cameras.

In comparison to other works, our approach outperforms [5] and it is -under good conditions- outperformed by [7]. Although [7] reports better results, it requires RGBD cameras, which are an order of magnitude more expensive than low-resolution RGB cameras. Additionally, as reported in [7] their dataset does not consider occlusions or partial views, so their results will likely deteriorate in real-life conditions.

A mean absolute error of $125mm$ in the pose's coordinates seems reasonable for most human-robot interaction tasks such as human-aware navigation, considering: **a)** the size of the average human *i.e.*, the percentile 50 of the forearm-forearm breadth of an adult is about $492mm$ (female) and $579mm$ (male) [24] and **b)** proxemics studies have reported personal spaces to approximate to a circle of about $1200mm$ [25].

Given the accuracy achieved using regular RGB images and the limited improvements obtained from the use of the depth channel, regular low-end webcams seem sufficient for most HRI scenarios.

Camera calibration can be avoided if the dataset is exclusively generated from real scenarios. Nevertheless, the use of a realistic simulator to generate most of the data used for training, drastically reduces the time and resources needed to obtain a valid solution to the pose estimation problem. A fraction of data from the real set up seems necessary to account for some calibration and modelling errors but future works aims at reducing this ratio even more.

REFERENCES

[1] G. Glonek and A. Wojciechowski, "Hybrid orientation based human limbs motion tracking method," *Sensors*, vol. 17, no. 12, 2017.

[2] D. N. Thang et al., "Deep Learning-based Multiple Objects Detection and Tracking System for Socially Aware Mobile Robot Navigation Framework," *NICS 2018 - Proceedings of 2018 5th NAFOSTED Conference on Information and Computer Science*, pp. 436–441, 2019.

[3] L. J. Manso, R. R. Jorvekar, D. R. Faria, P. Bustos, and P. Bachiller, "Graph Neural Networks for Human-aware Social Navigation," *arXiv preprint arXiv:1909.09003*, 2019.

[4] A. Mateus, D. Ribeiro, P. Miraldo, and J. C. Nascimento, "Efficient and robust Pedestrian Detection using Deep Learning for Human-Aware Navigation," *Robotics and Autonomous Systems*, vol. 113, pp. 23–37, 2019. [Online]. Available: https://doi.org/10.1016/j.robot.2018.12.007

[5] F. Shinmura, D. Deguchi, I. Ide, H. Murase, and H. Fujiyoshi, "Estimation of human orientation using coaxial RGB-depth images," *VISAPP 2015 - 10th International Conference on Computer Vision Theory and Applications; VISIGRAPP, Proceedings*, vol. 2, pp. 113–120, 2015.

[6] R. Wahyu and A. Saputra, "Human Body ' s Orientation Estimation Based On Depth Image," *2019 International Electronics Symposium (IES)*, pp. 266–271, 2019.

[7] B. Lewandowski, D. Seichter, T. Wengefeld, L. Pfennig, H. Drumm, and H.-M. Gross, "Deep orientation: Fast and Robust Upper Body orientation Estimation for Mobile Robotic Applications," vol. 2, no. 03, pp. 441–448, 2020.

[8] J. Choi, B.-J. Lee, and B.-T. Zhang, "Human Body Orientation Estimation using Convolutional Neural Network," 2016. [Online]. Available: http://arxiv.org/abs/1609.01984

[9] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, July 1997.

[10] P. Constantine and P. Tomaso, "A Trainable System for Object Detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[11] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR 2011*. Ieee, 2011, pp. 1297–1304.

[12] J. Liu, Y. Liu, G. Zhang, P. Zhu, and Y. Q. Chen, "Detecting and tracking people in real time with RGB-D camera," *Pattern Recognition Letters*, vol. 53, pp. 16–23, 2015.

[13] L. V. Calderita, J. P. Bandera, P. Bustos, and A. Skiadopoulos, "Model-based reinforcement of Kinect depth data for human motion capture applications," *Sensors*, vol. 13, no. 7, pp. 8835–8855, 2013.

[14] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[15] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] E. Rohmer, S. P. Singh, and M. Freese, "Coppeliasim: A versatile and scalable robot simulation framework," in *Proc. Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[17] T. Kailath, "The divergence and Bhattacharyya distance measures in signal selection," *IEEE transactions on communication technology*, vol. 15, no. 1, pp. 52–60, 1967.

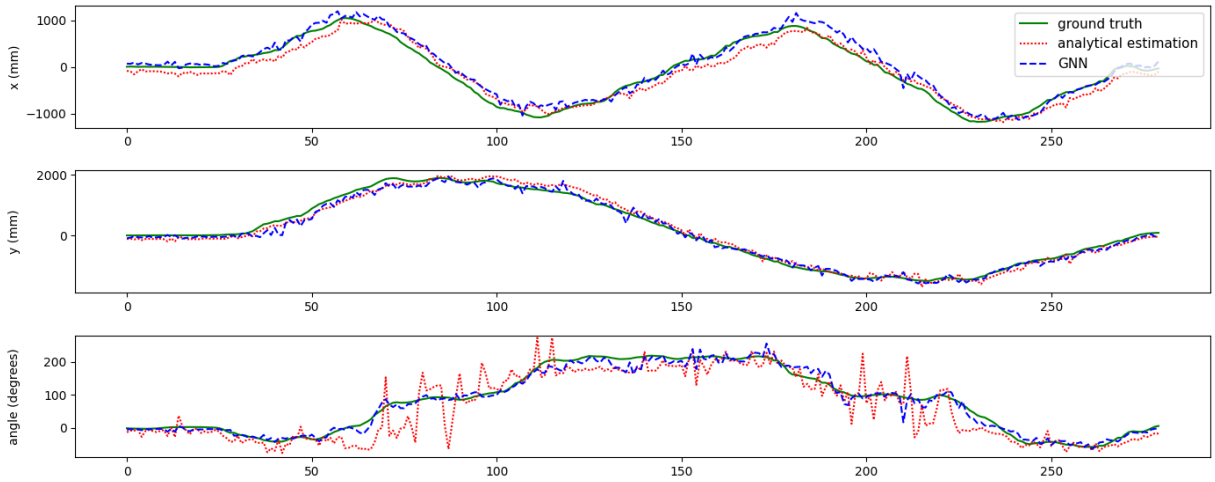


Fig. 3. Comparison of the network prediction and the analytical estimation of the pose with the ground-truth for every sample of the test datasets.

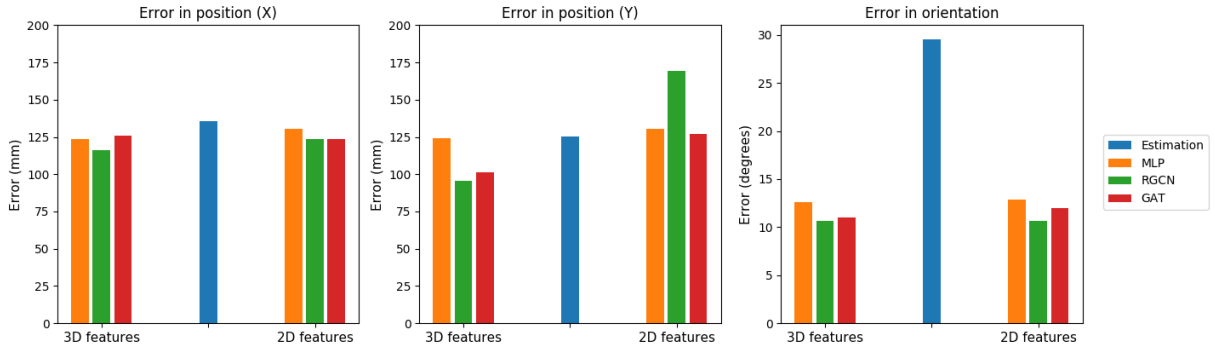


Fig. 4. MAE of the position (x and y) and orientation for the test datasets using the training dataset DS1, composed of simulated and real data.

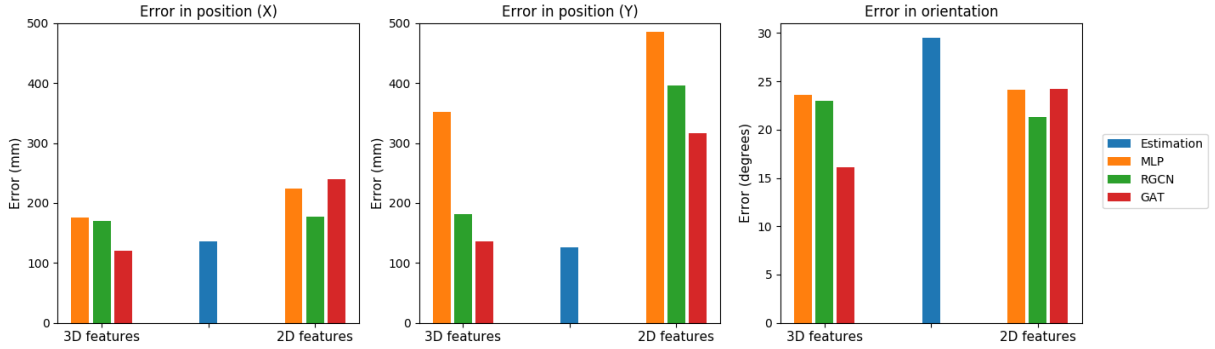


Fig. 5. MAE of the position (x and y) and orientation for the test datasets using the training dataset DS2 composed entirely of simulated data.

[18] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," pp. 1–40, 2018. [Online]. Available: http://arxiv.org/abs/1806.01261

[19] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," pp. 1–14, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[20] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling Relational Data with Graph Convolutional Networks," *Lecture Notes in Computer Science*, vol. 10843 LNCS, no. 1, pp. 593–607, 2018.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *Proceedings of the International Conference on Learning Representations 2018*, no. 2015, 2018, pp. 1–11. [Online]. Available: http://arxiv.org/abs/1710.10903

[22] Y. Li, Z. He, X. Ye, Z. He, and K. Han, "Spatial temporal graph convolutional networks for skeleton-based dynamic hand gesture recognition," *Eurasip Journal on Image and Video Processing*, vol. 2019, no. 1, 2019.

[23] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[24] C. C. Gordon, C. L. Blackwell, B. Bradtmiller, J. L. Parham, P. Barrientos, S. P. Paquette, B. D. Corner, J. M. Carson, J. C. Venezia, B. M. Rockwell, *et al.*, "2012 anthropometric survey of us army personnel: Methods and summary statistics," Army Natick Soldier Research Development and Engineering Center MA, Tech. Rep., 2014.

[25] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Human-robot embodied interaction in hallway settings: A pilot user study," in *IEEE International Workshop on Robot and Human Interactive Communication*, vol. 2005. IEEE, 2005, pp. 164–171.