Acta Polytechnica CTU Proceedings **57**:159–167, 2026
https://doi.org/10.14311/APP.2026.57.0159

Published by the Czech Technical University in Prague

COMPARISON OF METHODS TO DETERMINE PEDESTRIAN SHOULDER ORIENTATION FROM STEREO RECORDINGS

Deniz Kiliç*, Maik Boltes

Institute for Advanced Simulation 7: Civil Safety Research, Forschungszentrum Jülich, 52428 Jülich, Germany
* corresponding author: d.kilic@fz-juelich.de

Abstract. The orientation of pedestrians is used for analysis of space usage, social groups or orientation dynamics in crowds. For this purpose, it is desirable to measure the orientation for each pedestrian in the crowd. This work compares geometric and machine learning methods to determine the shoulder orientation. We make use of stereo camera systems to enable the estimation without the need for intrusive measurement equipment.

The machine learning methods consistently outperform the geometric methods. Especially in higher densities. However, high-density and low movement datasets are still challenging for machine learning techniques with automated labels. In laboratory experiments, coloured shoulder markers can be a cost-effective alternative method.

Keywords: Orientation, shoulder, stereo computer vision, machine learning, pedestrian experiments.

1. Introduction

The orientation of pedestrians can be used to analyse their space usage, detect social groups or as part of a biomechanical analysis of their gait. If we are interested in crowd dynamics, measurements of the orientation of (almost) all members of the crowd are desirable. Measurement methods such as inertial measurement units or optical motion capturing systems do not scale easily to a larger crowd. In contrast, image-based methods are applicable even to a large number of pedestrians. They often do not require instrumentation of each pedestrian, which can simplify laboratory experiments or enable field observations. Existing work in robotics has shown the superiority of depth data over colour data in the task of orientation estimation [1]. In pedestrian dynamics, depth data has been used to determine the shoulder orientation in field observations [2, 3]. In this work we want to adapt and modify existing approaches, based on geometry [2] or machine learning [3]. With a look at laboratory experiments, we also compare these to using optical markers in RGB images. We hope to provide guidance on what method to use if orientation data for the entire crowd is desired.

2. Materials and methods

2.1. Data acquisition

In this work we use two datasets acquired with stereo cameras. The first one is a corner flow experiment [4] and the second one is a bottleneck experiment [5]. For the corner experiment a Bumblebee™ XB3 [6] was used. For the bottleneck experiments a Framos D455e [7]. For the bottleneck experiments an ordinary RGB GoPro camera is available for marker-based detection.

Stereo cameras take two images of the scene at the same time. The two images are compared and the shift of the same features from one image to the other in pixel space, the disparity, is determined. From this disparity, the depth, i.e. the distance of a pixel from the camera, can be determined [8, p. 537]. With the distance and the pixel coordinates, we can calculate the 3D position of the point captured by each pixel. The collection of these 3D points is called a point cloud.

For the corner data we do the disparity calculations using the `StereoSGBM` algorithm of the OpenCV library [9]. For the bottleneck data, the calculations were performed on the camera itself using a hardware accelerator (FRAMOS D4 Visual Processing Board).

2.2. Geometric methods

The method of Brščić et al. [2] works by determining all the points belonging to a single person in the point cloud, extracting the shoulder region, projecting it onto the ground and fitting an ellipse to it. We also implemented a variant fitting an ellipsoid instead of an ellipse in an effort to make as much use as possible from the 3D data given. The flow diagram in Figure 1 describes the steps of the algorithm. First, a disparity is determined and a point cloud is calculated. Following this, a background subtraction is performed. We use the minimum background [10], where a small part of the recording with only background present is used to calibrate the background model by saving the minimal depth of each pixel in the calibration period.

A pixel is classified as foreground if the object is closer to the camera than the calibrated background, i.e. if the depth is smaller. Following this, we filter the point cloud by height to extract the upper bodies by discarding points above 2 m or below 0.9 m. After this, we downsample the point cloud via a random

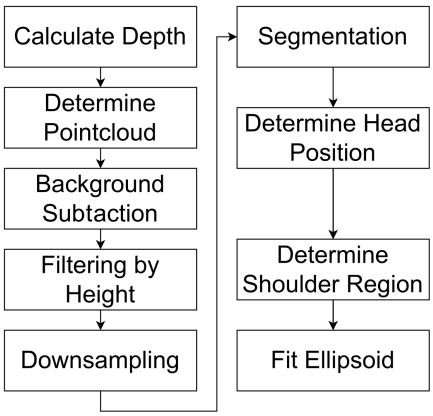

Figure 1. Algorithmic flow of the ellipsoid fitting method.

sampling. The downsampled point cloud is segmented following Brščić et al. [2] by sorting all points by their z-component. Starting from the highest point, we search for a neighbour that already has a label. If such a neighbour exists, the point assumes the label of the closest labelled neighbour. If no such neighbour exists, it gets a new label. This procedure takes advantage of the fact that pedestrians are well separated at head height and grows the segments from the heads downward. For each of these segments, we can determine the head position. Deviating from Brščić et al. [2], we implemented the method from [11] to determine the head position by using the mean of all points above the 10^{th} percentile according to z-coordinate. Given the head height, we can now determine the shoulder region by the height of points relative to the head height, ranging from 0.3 m below the head point until 0.45 m below according to the distance between body height and acromial shoulder height for the upper threshold and the distance between acromial shoulder height and height of the shoulder blade for the lower threshold [12].

Given the shoulder region of a pedestrian, we get the orientation by fitting an ellipse. This is done via a principal component analysis on the 2D projection of the points from the shoulder region onto the ground. As a deviation from Brščić et al. [2], we implemented an alternative procedure that fits a 3D ellipsoid with a fixed size and fixed roll and pitch angle, only optimising on position and yaw. For fitting the ellipsoid, a cost based on Karel Zimmermann and Tomáš Svoboda [13] as well as its derivative are implemented to facilitate the use of a standard nonlinear optimizer, in our case, L-BFGS implemented via nlopt [14]. The correction in case of an occluded shoulder is not applied in either fitting procedure, since no improvements on the given dataset were observed. The authors observed no improvements by using the 3D ellipsoid. In the result section, only results for the ellipse are reported.

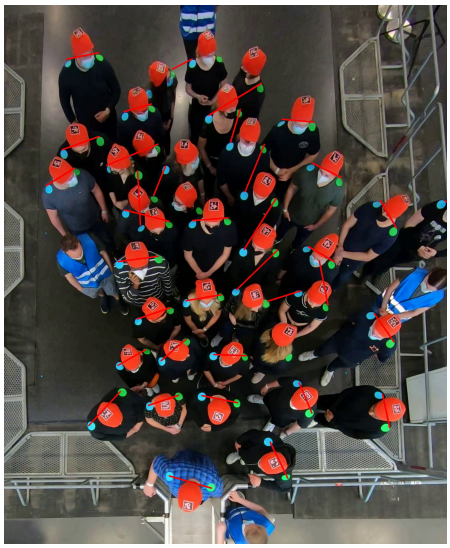

Figure 2. Shoulder matching for one frame. The occluded right shoulder at the back of the crowd and the face mask that is detected as a right shoulder in the front of the crowd lead to a chain of wrong matchings.

2.3. Marker-based method

When performing laboratory experiments, pedestrians can be equipped with markers. Additionally to head markers (e.g. coloured hats), shoulder markers can be used. Boomers and Boltes [15] have explored marker based shoulder orientation measurements. Each pedestrian gets a coloured dot glued onto each shoulder as seen in Figure 2. The colour of the dot signified whether it is the left or right shoulder, allowing for 360° resolution, compared to the 180° resolution of the other methods. The error of marker-based estimation is low with a RMSE of 7.09° and a MAE of 4.05°[1]. However, the presented method struggles with high densities. Since the dataset for shoulder markers in this study features a bottleneck with high densities, another method for matching the shoulders to pedestrians was used. Please note that on successful matching, the error remains the same between the two methods. Additionally, we note that all other methods compare their results to values that are guaranteed to be in the same coordinate system, while [15] are comparing to motion capturing data that needed to be fused with camera data first.

To enable matching in high-density scenarios, we solve the problem globally on a frame instead of solving it locally on a region of interest. This allows us to encode that a shoulder is only part of a single pedestrian and never of multiple pedestrians. We model this problem as a three-dimensional assignment problem. Given a set of left shoulder positions L, right shoulder positions R and head positions H respectively, we search for a set $S \subset L \times H \times R$ that matches heads to shoulders. We can enumerate all the tuples in $L \times H \times R$. Then, we create a boolean tensor x with

[1]Errors calculated with data shared by Boomers and Boltes

the intended semantics that x_{ijk} is 1 if and only if $(l_i, h_j, r_k) \in S$. For each such tuple we can define a cost c_{ijk}. In our case, the cost is defined as

$$\begin{aligned} c_{ijk} = &\Big(||l_i - h_j||^2 - (0.2\,\mathrm{m})^2\Big)^2 \\ &+\Big(||r_k - h_j||^2 - (0.2\,\mathrm{m})^2\Big)^2 \\ &+2\Big(||l_i - r_k||^2 - (0.35\,\mathrm{m})^2\Big)^2 . \end{aligned} \tag{1}$$

Since we use the coloured dots as the position of the shoulders, the desired distances between the shoulders and between shoulder and head are smaller than might be expected from literature on body measurements [12]. This leaves us with a minimisation problem.

There is one practical problem in this formulation: due to (self-)occlusion of pedestrians, not all shoulder points are detected. Therefore, we do not want a maximum matching. It might be desired to leave out head or shoulder points if they cannot be assembled into a credible human, e.g. because head and shoulder are multiple meters apart. We can do this by introducing variables $y^{\text{left,head,right}}$, where e.g. $y_j^{\text{head}} = 1$ if we leave out the head h_j or $y_j^{\text{head}} = 0$ otherwise. By giving a cost $c_y > 0$ to assigning a 1, we can fine-tune how high the cost of a (r_i, h_j, l_k) tuple needs to be, such that leaving the points out is preferable. We set $c_y = (0.003, 0.003, \dots)^T$. A nice feature of this formulation compared to a fixed threshold is that smaller errors due to many slightly wrong matchings build up until they eventually allow for discarding problematic points. This becomes relevant when close-by pedestrians' shoulder can be used to form a chain of using the neighbours shoulders. A failure to prevent such a case can be seen in Figure 2.

We implemented this problem using the CP-SAT [16] solver in OR-Tools [17]. As such, the solver was mostly enforcing the constraints and then brute-forcing the remaining options. To reduce the computational burden, we discard tuples with cost $c_{ijk} > 0.5$ before invoking the solver.

The resulting problem formulation is

$$\begin{aligned} \min_{x,y} \quad & c^T x + c_y^T [y^{\text{left}}; y^{\text{head}}; y^{\text{right}}] \\ \text{s.t.} \quad & \sum_{j=0}^{|H|} \sum_{k=0}^{|R|} x_{ijk} + y_i^{\text{left}} = 1 \quad \forall i \\ & \sum_{i=0}^{|L|} \sum_{k=0}^{|R|} x_{ijk} + y_j^{\text{head}} = 1 \quad \forall j \\ & \sum_{i=0}^{|L|} \sum_{j=0}^{|H|} x_{ijk} + y_k^{\text{right}} = 1 \quad \forall k \\ & x_{ijk} = 0 \quad \forall i,j,k : c_{ijk} > 0.5 \, . \end{aligned} \tag{2}$$

2.4. Machine-Learning-based method

One simple estimator for the shoulder orientation is to assume it is orthogonal to the movement direction. This estimate is noisy. However, one can train a neural network with noisy labels. This was done by Willems et al. [3], who devised their own neural network architecture for the problem of orientation estimation. Willems et al. [3] used their method on field data, which was further enriched using simulated data. With these data sources the largest training dataset they considered had $N \approx 5 \cdot 10^5$ data points. We want to look at using this methodology for experimental data, where N usually is smaller, even with the automatic labelling procedure. To extract the image patches with their annotation, trajectories extracted and manually corrected with PeTrack [18, 19] are used. The head point of the trajectory is lowered by 32 cm and re-projected onto the image to find the centre of the patch. This is done to ensure that the patch is centred around the shoulder of the pedestrian, even in cases where the angle of view is large. For the corner flow experiments a region of 80×80 px is extracted around these points; for the bottleneck experiments a region of 120×120 px. The sizes of the patches were selected such that the upper body was entirely visible in the patch given the camera resolution and position. For processing in the neural network, these image patches are downscaled to 48×48 px.

In contrast to [3], we save the image patches as grey scale images, with the grey value being determined by a min-max normalisation with user-determined values. For labelling the patches, the moving direction needs to be determined from the head trajectories. For the corner flow experiments, a central moving average with a window size of 0.8 s is used to smooth the head swaying out of the trajectories. Then the vector between the current and next point on the trajectory is used as the estimate of the moving direction. For the bottleneck experiment, movement is generally too slow for a central moving average. Instead, we determine the next point on the trajectory that is at least 0.5 m away from the current point, to remove local velocity fluctuations without forward movement, and use the vector between these two points as the estimate of the moving direction. During training, the data is augmented by random rotations and horizontal flipping.

As model architectures, we tested the architecture by Willems et al. [3] as well as other approaches from the computer vision community to orientation estimation such as the biternion net [20] and the modified biternion net [1]. The implementation of the biternion nets is based on the repository from [1] and done in TensorFlow [21] and Keras [22]. All architectures are VGG-like architectures. Willems et al. [3] add $B = 45$ bins as the final layer, encoding the value using two-bin scaled encoding. Each bin is referring to an angle in the middle of the bin. The network is supposed to estimate the probability of the angle lying inside the bin i, $P(i)$. The estimated value for the orientation is then the expected value of the distribution. For the expected value, a circular average

$\hat{\theta} = \frac{1}{2}\arctan 2\left(\sum_{i=1}^{B} P(i)\sin 2\theta_i, \sum_{i=0}^{B} P(i)\cos 2\theta\right)$ is used. As loss function, cross-entropy is used.

The biternion networks use a biternion output layer. The output is a two-dimensional normalised vector that is interpreted as $\hat{\mathbf{y}} = (\cos\hat{\theta}, \sin\hat{\theta})^T$. This output representation is combined with a loss function based on the von Mises distribution, an approximation of the normal distribution on the circle. The loss $1 - \exp(\kappa(\hat{\mathbf{y}} \cdot \mathbf{y} - 1))$ with parameter κ, characterising the variance of the distribution. While this architecture generally estimates the entire 360°, we limit ourselves by only estimating angles in $[0°; 180°]$. This is done by re-scaling the output and annotations. This simplifies the learning task for the network that does not need to learn the slight symmetry breaks that would enable it to learn the orientation with regard to all possible values and since we have movement information and pedestrians rarely walk backwards, we can infer the exact orientation later on. Also, it makes the network more comparable to the architecture from Willems et al. [3]. The modified biternion net [1] has some changes to the architecture. The largest change is the introduction of dropout layers.

Willems et al. [3] employ test time augmentation to improve the results and strengthen the symmetry of the estimator. We did not observe any systematic improvement on our data, so we opted to not apply test time augmentation.

The implementation of the image patch generation, the neural network models and the training can be found at `https://jugit.fz-juelich.de/shoulder-orientation-estimation/machine-learning`.

3. Results

Three runs of the corner flow were partially manually annotated. These are the experiments `300-060-300`, `300-100-300` and `300-300-300` in increasing order of density [4]. To evaluate annotation accuracy, one frame per experiment was annotated twice by the same person, resulting in $3.79 \pm 3.19°$ disagreement between annotations. With the annotated data, we can see how the different methods perform under different densities. The average densities for these three runs are $0.54 \pm 0.14\,\mathrm{ped\,m^{-2}}$, $0.83 \pm 0.4\,\mathrm{ped\,m^{-2}}$ and $1.84 \pm 0.66\,\mathrm{ped\,m^{-2}}$. For the bottleneck experiments, the GoPro camera was used to extract shoulder markers as described in Section 2.3. These are used instead of manual labels for evaluation of the other methods. The bottleneck experiment has an average density of $2.86 \pm 2.10\,\mathrm{ped\,m^{-2}}$ [5]. The participants all wear similar, mono-coloured shirts. This makes stereo matching more complicated, especially in high densities.

The simplest method to determine the shoulder orientation is using the movement direction. In Figure 3 we can see that the movement direction is a good indicator of the shoulder orientation in the case of the corner flow (MAE moving average: 7.42°, 6.54° and 9.14°). In the bottleneck where pedestrians often stand still or move slowly, the movement direction performs significantly worse (MAE moving average: 31.92°. Distance method: 15.92°). While the moving average performs better in the corner experiments, the distance based method performs better in the bottleneck experiment. Accordingly, we use the moving average to automatically label our corner flow data for the ML methods, while we use the distance based method on the bottleneck experiment.

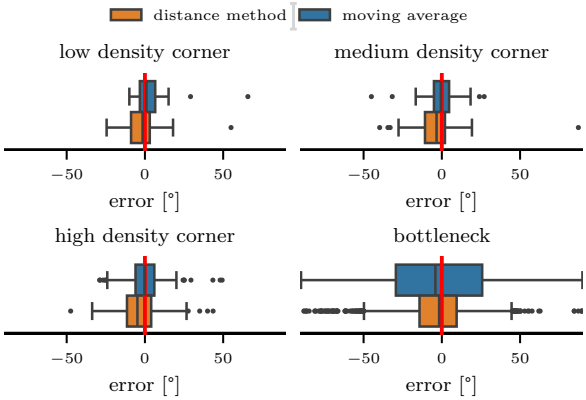

Figure 3. Errors for different methods using the movement direction to determine the shoulder orientation. The shoulder orientation is always assumed to be orthogonal to the movement direction.

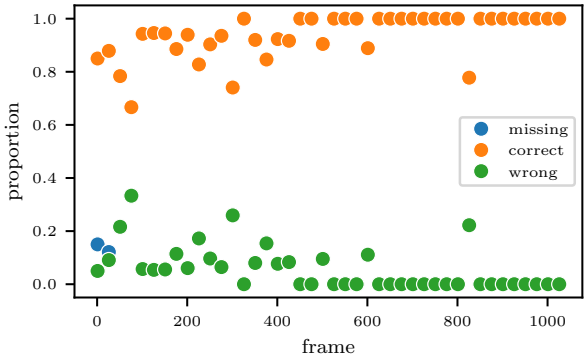

Figure 4. Proportion of pedestrians with correct head-shoulder matching, wrong matching or missing matching. All pedestrians that are part of a wrong matching are also counted as missing, since their correct matching is not recognised. Data for each second (each 25th frame).

The test dataset for the bottleneck experiments is labelled using the shoulder marker method. If the detection and matching is correct, the error is ca. 4° [15]. To evaluate the method, we look at how many head-shoulder-triplets are correct, missing or wrong respectively. A wrong pair cannot easily be assigned to a single person, thus a person with e.g. a head in a wrong triplet is counted as missing as well. The relative occurrence of these cases w.r.t. the total population over time can be seen in Figure 4.

The large errors in the beginning of the video are due to face masks falsely being detected as shoulder

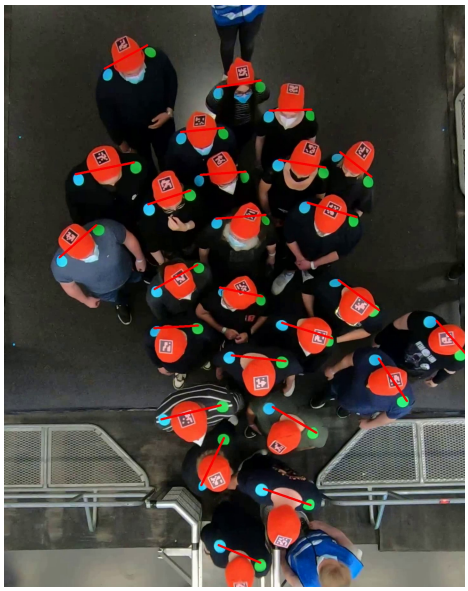

FIGURE 5. Shoulder matching for frame 410.

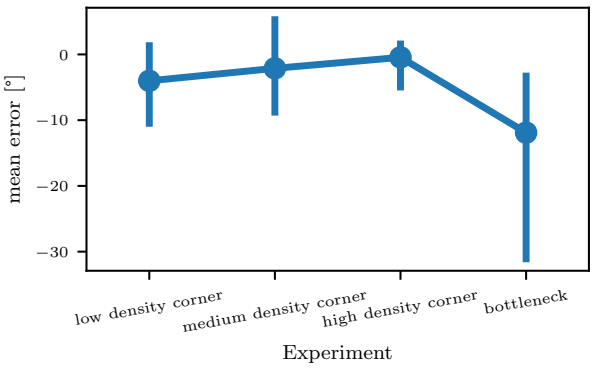

FIGURE 6. Mean error of 10 training runs of a modified biternion net with the same hyperparameters. Point is the median, bar shows span of reached values.

markers, leading to a chain of misattributions along the crowd, see Figure 2. This highlights the need for a low false-positive rate in the detection of shoulder markers.

For the test dataset on the bottleneck, only frames after frame 400 are taken, since for these the shoulder markers give the most accurate annotation (cf. Figure 4 and Figure 5).

When comparing the different neural net architectures with each other in Figure 7, we can see that there is not much of a difference between the different architectures in the corner flow experiments. In the bottleneck scenario, there is more of a difference between the models. It should be noted that the figure only shows results of one trained model. The variance of model performance is higher in the bottleneck case (cf. Figure 6) and due to the nature of our hyperparameter optimisation, no equal number of training runs has been used for all the models.

Nonetheless, we select this best modified biternion net for the comparison with the other methods for the bottleneck flow (MAE: 12.34°). For the corner flow experiments, we select the ordinary biternion net (MAE: 8.67°, 6.40° and 8.19°).

The geometric method performs worse with increasing densities (MAE: 11.38°, 12.70° and 18.47° for corner; 22.07° for bottleneck). In Figure 8 we see that all methods, except the ML methods, degrade with higher density. It should be noted that the ML methods dataset has more examples from higher densities due to the density being controlled by the number of pedestrians participating in the experiment. Throughout all scenarios the ML method performs best among the tested methods. However, in the bottleneck scenario, no method achieves satisfactory performance. A visual comparison of all methods can be seen in Figure 9.

4. Conclusions

We have evaluated multiple different methods for estimating the shoulder orientation from depth data. Throughout all our scenarios, the machine learning based methods performed best. This was the case despite being trained only on small laboratory datasets that were automatically labelled. However, for the bottleneck scenario with little to no movement the automated labelling and thus the machine learning method perform poorly. Still, due to the high densities in the bottleneck experiment, the geometric approach performs even worse. Interestingly, the movement direction outperforms the geometric method. Of course, the movement direction has systematic errors. But going purely by summary statistics, it is more accurate than the geometric approach, even for the bottleneck scenario. The comparison of different neural network architectures did not reveal any meaningful differences. When in a controlled laboratory setting, using coloured shoulder markers is a well-working alternative to depth-based estimation of the shoulder orientation. This holds true even for the challenging case of high densities.

Another avenue not explored in this work is the usage of supervised machine learning with human labels. Especially the results concerning the degradation of performance of the ML methods on the bottleneck dataset indicate that higher quality data could increase the accuracy in scenarios with movement like the corner flow, while it could make estimation of shoulder orientation possible for scenarios with little movement like the bottleneck.

Acknowledgements

This study uses data from experiments that were funded by the European Unions Horizon 2020 research and innovation program within the project CrowdDNA [grant number 899739] and the project Hermes funded by the Federal Ministry of Education and Research (BMBF) Program on “Research for Civil Security – Protecting and Saving Human Life”.

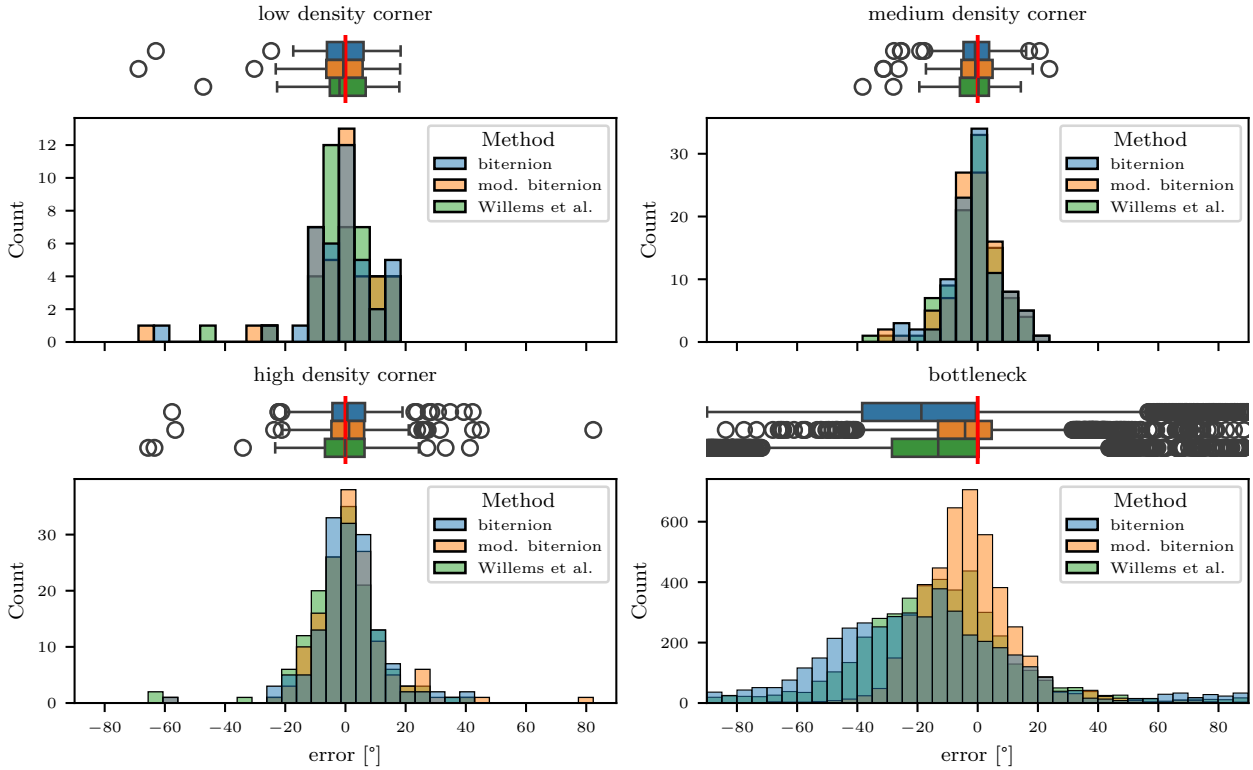

Figure 7. Histogram of errors of the orientation estimate for the different scenarios and machine learning models.

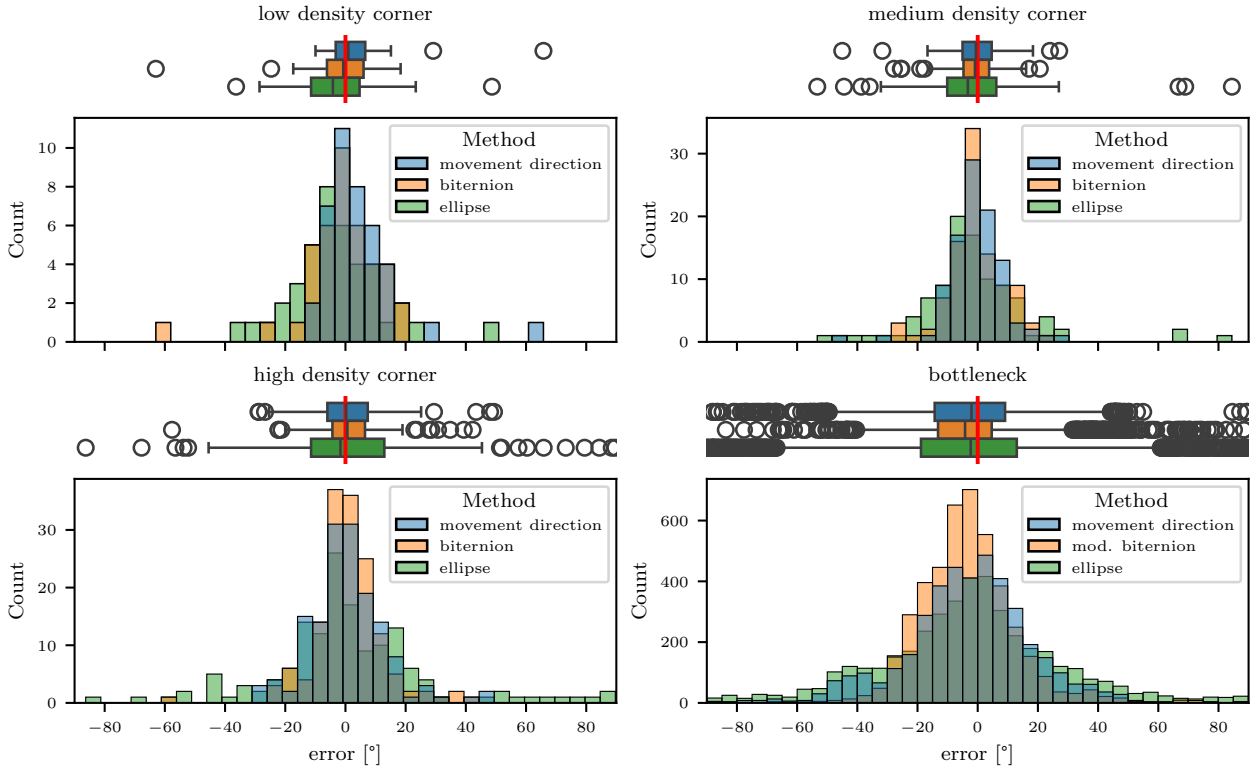

Figure 8. Histogram of errors of the orientation estimate for the different scenarios.

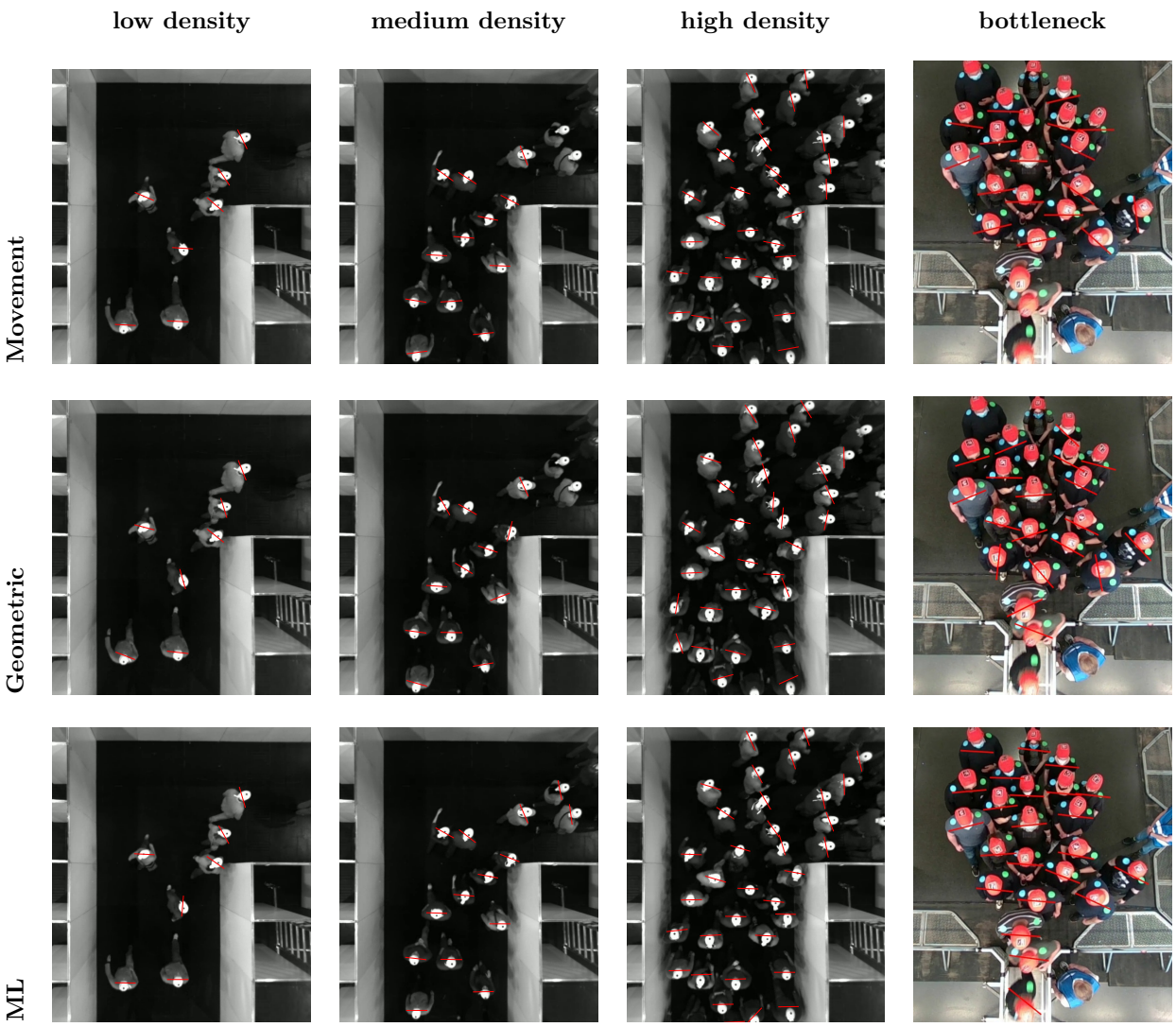

FIGURE 9. Example frames for each method on each dataset.

REFERENCES

[1] B. Lewandowski, D. Seichter, T. Wengefeld, et al. Deep orientation: Fast and robust upper body orientation estimation for mobile robotic applications. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 441–448. 2019. ISSN: 2153-0866. https://doi.org/10.1109/IROS40897.2019.8968506

[2] D. Brščić, T. Kanda, T. Ikeda, T. Miyashita. Person tracking in large public spaces using 3-D range sensors. *IEEE Transactions on Human-Machine Systems* **43**(6):522–534, 2013. https://doi.org/10.1109/THMS.2013.2283945

[3] J. Willems, A. Corbetta, V. Menkovski, F. Toschi. Pedestrian orientation dynamics from high-fidelity measurements. *Scientific Reports* **10**(1):11653, 2020. https://doi.org/10.1038/s41598-020-68287-6

[4] Forschungszentrum Jülich, University Of Wuppertal, Cologne University, et al. Corner, 2009. https://doi.org/10.34735/PED.2009.8

[5] H. Lügering, J. Adrian, M. Boltes, et al. Propagation of information in waiting crowds, 2024. https://doi.org/10.34735/PED.2022.5

[6] Bumblebee™ XB3 FireWire | Teledyne FLIR.

[7] FRAMOS Depth Camera D455e Starter Kit.

[8] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer London, 2011. https://doi.org/10.1007/978-1-84882-935-0

[9] G. Bradski. The OpenCV library. *Dr Dobb's Journal of Software Tools* **25**(11):120, 122–125, 2000.

[10] K. Greff, A. Brandão, S. Krauß, et al. A comparison between background subtraction algorithms using a consumer depth camera. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISIGRAPP 2012) – Volume 2: VISAPP*, pp. 431–436. 2012. https://doi.org/10.5220/0003849104310436

[11] A. Corbetta, L. Bruno, A. Muntean, F. Toschi. High statistics measurements of pedestrian dynamics. *Transportation Research Procedia* **2**:96–104, 2014. https://doi.org/10.1016/j.trpro.2014.09.013

[12] H. Greil. Körpermaße 2000 : aktuelle Perzentilwerte der deutschen Bevölkerung im jungen Erwachsenenalter. *Brandenburgische Umwelt-Berichte : BUB ;*

Schriftenreihe der Mathematisch-Naturwissenschaftlichen Fakultät der Universität Potsdam **10**:23–53, 2001.

[13] K. Zimmermann, T. Svoboda. Approximation of Euclidean distance of point from ellipse. Tech. rep., Research Reports of CMP, Czech Technical University in Prague, 2005. CTU-CMP-2005-23, ISSN 1213-2365. [2024-03-28]. http://cmp.felk.cvut.cz/ftp/articles/zimmerk/Zimmermann-TR-2005-23.pdf

[14] S. G. Johnson. The NLopt nonlinear-optimization package, 2007. [2025-05-12]. https://github.com/stevengj/nlopt

[15] A. K. Boomers, M. Boltes. Shoulder rotation measurement in camera and 3D motion capturing data. In K. R. Rao, A. Seyfried, A. Schadschneider (eds.), *Traffic and Granular Flow '22*, pp. 35–42. Springer Nature, Singapore, 2024. https://doi.org/10.1007/978-981-99-7976-9_5

[16] L. Perron, F. Didier. CP-SAT. [2024-05-07]. https://developers.google.com/optimization/cp/cp_solver/

[17] L. Perron, V. Furnon. OR-Tools. [2024-05-07]. https://developers.google.com/optimization/

[18] M. Boltes, A. Seyfried. Collecting pedestrian trajectories. *Neurocomputing* **100**:127–133, 2013. https://doi.org/10.1016/j.neucom.2012.01.036

[19] M. Boltes, D. Kilic, T. Schrödter, et al. PeTrack, 2025. https://doi.org/10.5281/ZENODO.5078176

[20] L. Beyer, A. Hermans, B. Leibe. Biternion nets: Continuous head pose regression from discrete training labels. In J. Gall, P. Gehler, B. Leibe (eds.), *Pattern Recognition*, pp. 157–168. Springer International Publishing, Cham, 2015. https://doi.org/10.1007/978-3-319-24947-6_13

[21] TensorFlow. An end-to-end platform for machine learning. [2025-11-10]. https://www.tensorflow.org/

[22] Keras. A superpower for ML developers. [2025-11-10]. https://keras.io/

A. Appendices

A.1. Shoulder Marker Matching

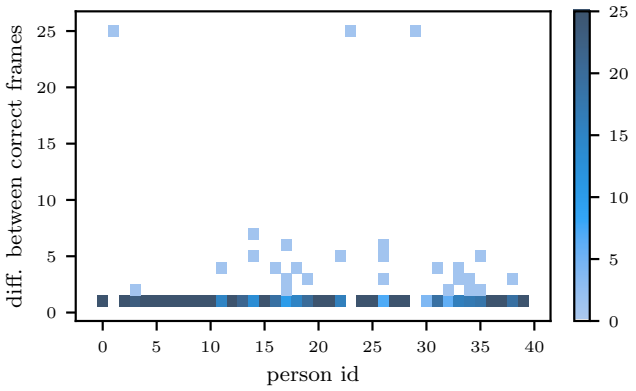

Figure 10. Histogram of differences between consecutive correct detections of head-shoulder-triplets. Perfect detections would mean a difference of 1. Measured over one second at 25 FPS. Data is from frames 76 to 101.

Systematic errors can persist over a longer time period, if the spatial configuration that leads to them is stable. In a moving crowd, errors can also be spurious. In Figure 10, we see how many frames are between two correct detections for each pedestrian over one second (25 frames). We see that most pedestrians have no wrong detections or only small frame distances between correct detections. This allows for the correction of errors via smoothing. Especially in slow moving scenarios like here, where the orientation does generally not change quickly. Three pedestrians have no correct detection in the entire second. This is due to at least one of their shoulder markers not being visible due to (self-)occlusion in all 25 frames.

A.2. Machine Learning Hyperparameters

Hyperparameters were tuned via a naive grid search, leading to more runs for the biternion net, since it has more parameters. The best model according to validation error was selected. There were additional runs on the modified biternion net to determine the variance of results, see Figure 6. One of these runs was selected for the bottleneck experiments. The hyperparameters with the best validation error that were selected for the discussion of results in the main part can be seen in Table 1

	Corner			**Bottleneck**		
	biternion	**mod. biternion**	**Willems et al. [3]**	**biternion**	**mod. biternion**	**Willems et al. [3]**
κ	0.5	1.0		0.5	0.5	
dropout		True			True	
epochs	50	50	50	50	50	50
optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
lr	0.001	0.001	0.001	0.001	0.001	0.001

Table 1. Hyperparameters used in the training runs.