sensors

MDPI

Article

The Quadrature Method: A Novel Dipole Localisation Algorithm for Artificial Lateral Lines Compared to State of the Art

Daniël M. Bot [1,*], Ben J. Wolf [2,3] and Sietse M. van Netten [3,*]

[1] I-BioStat, Data Science Institute, Hasselt University, 3500 Hasselt, Belgium
[2] Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands; b.j.wolf@rug.nl
[3] Bernoulli Institute of Mathematics, Computer Science and Artificial Intelligence, Faculty of Science and Engineering, University of Groningen, 9747 AG Groningen, The Netherlands
* Correspondence: jelmer.bot@uhasselt.be (D.M.B.); s.m.van.netten@rug.nl (S.M.v.N.)

Abstract: The lateral line organ of fish has inspired engineers to develop flow sensor arrays—dubbed artificial lateral lines (ALLs)—capable of detecting near-field hydrodynamic events for obstacle avoidance and object detection. In this paper, we present a comprehensive review and comparison of ten localisation algorithms for ALLs. Differences in the studied domain, sensor sensitivity axes, and available data prevent a fair comparison between these algorithms from their original works. We compare them with our novel quadrature method (QM), which is based on a geometric property specific to 2D-sensitive ALLs. We show how the area in which each algorithm can accurately determine the position and orientation of a simulated dipole source is affected by (1) the amount of training and optimisation data, and (2) the sensitivity axes of the sensors. Overall, we find that each algorithm benefits from 2D-sensitive sensors, with alternating sensitivity axes as the second-best configuration. From the machine learning approaches, an MLP required an impractically large training set to approach the optimisation-based algorithms' performance. Regardless of the data set size, QM performs best with both a large area for accurate predictions and a small tail of large errors.

Keywords: hydrodynamic imaging; dipole localisation; artificial lateral line; neural networks

Citation: Bot, D.M.; Wolf, B.J.; van Netten, S.M. The Quadrature Method: A Novel Dipole Localisation Algorithm for Artificial Lateral Lines Compared to State of the Art. *Sensors* **2021**, *21*, 4558. https://doi.org/10.3390/s21134558

Academic Editor: Enrico Meli

Received: 10 May 2021
Accepted: 28 June 2021
Published: 2 July 2021

1. Introduction

Artificial lateral lines (ALLs) are sensor arrays inspired by the biological lateral line organ found in fish and amphibians. This organ enables fish to detect and locate moving objects such as prey, predators, or social partners [1]. The ability to sense one's environment using a lateral line is sometimes called hydrodynamic imaging [2,3]. Two types of hydrodynamic imaging are distinguished: active hydrodynamic imaging, where fish use their movement's flow field to detect stationary obstacles; and passive hydrodynamic imaging, where fish detect fluid flows generated externally. Both types of hydrodynamic imaging have applications for ALLs. Active hydrodynamic imaging is useful for obstacle avoidance of autonomous underwater vehicles (AUVs) [4]. Here, ALLs provide a complementary sense for AUVs because—unlike cameras—they do not rely on visibility and—unlike sonar—they do not actively emit a signal. Passive hydrodynamic imaging can be used for tracking the location of objects—for instance, ships in a harbour—or detecting disturbances near underwater installations.

Several dipole localisation algorithms have been developed for ALLs in the last 15 years [5–14]. These algorithms attempt to locate objects that move underwater using the water flow pattern their movement generates. This process is analogous to solving the inverse problem of hydrodynamic source localisation [15]. It is challenging to compare these algorithms' performance from their original works due to differences in experimental

designs and conditions. For instance, some algorithms were evaluated for 2D localisation [5–10], whereas others localised sources in 3D [11–13]. Additionally, some algorithms were evaluated in simulations [5,9], while other studies used physical sensors [6–8,10–14]. In particular, differences in the (simulated) sensor sensitivity and the areas in which sources are located prevent the comparison of average or median prediction errors between studies because the number of inaccurate predictions depends heavily on sensor sensitivity and increases with the area's size.

The present research compares ten dipole localisation algorithms via their ability to determine an object's state from velocity measurements simulated using potential flow [16]. In our study, this state refers to the 2D position and movement direction relative to an array of flow sensors. The quality of these state estimates is compared using two analyses. First, we determine the size of the area in front of the array in which these algorithms accurately estimate an object's state. This analysis allows an intuitive comparison of the effective range of each algorithm. Secondly, we determine each algorithm's distribution of prediction errors, i.e., the differences between the actual and estimated state. This analysis better indicates the reliability of each algorithm and provides a median error metric.

As our baseline, we use two localisation methods: a random predictor (RND) and an off-the-shelf least square curve fit (LSQ) algorithm. The remaining algorithms are divided into three categories. The first category contains three template-based algorithms. Template matching [6] and linear constraint minimum variance (LCMV) beamforming [11,12] use a set of velocity measurements of a single source at different known positions and movement directions to locate a source. The continuous wavelet transform (CWT)—introduced for dipole localisation by Ćurčić-Blake and van Netten [5]—is also a template-based algorithm. This algorithm is based on the observation that a set of wavelets fully describes the potential flow [16] velocity generated by a dipole source. The second category contains artificial neural networks. A multi-layer perceptron (MLP) was used by Abdulsadda and Tan [7] and Boulogne et al. [9]. The latter showed that an extreme learning machine (ELM) performed better than their MLP implementation in high signal-to-noise conditions. The third category contains model-based algorithms. The first two, Gauss–Newton (GN) and Newton–Raphson (NR), fit a potential flow model to measured velocity values to predict a dipole's position and movement direction [8]. Abdulsadda and Tan [8] showed that GN consistently performed slightly better than NR, and both algorithms outperformed LCMV beamforming.

There are several novel aspects of the present study. Firstly, all localisation algorithms are extended to use various combinations of the velocity field's parallel and perpendicular components with respect to the sensor array. Several 2D sensitive fluid flow sensors exist in the literature [17–20], which provide the orthogonal fluid flow component that is not yet used by most dipole localisation algorithms. In addition, hair cells with varying orientations in close proximity have been observed in the ear of fish [21] and on the body of the *Xenopus laevis* frog (as cited in [1]). These varying orientations are thought to contribute to the localisation of stimuli. Secondly, not only are both the position and direction of movement of the source varied, their mutual effects on the prediction error are analysed as well. Thirdly, we use a novel approach to compare the performance of the dipole localisation algorithms, quantified by the area in which they correctly predict the position and direction of movement of an object with a predefined accuracy. Fourthly, we extend the template matching algorithm to a k-nearest neighbours (KNN) generalisation, where $k = 1$ is equivalent to the template matching algorithm as referenced earlier [6]. Finally, we introduce a novel model-based dipole localisation algorithm coined the quadrature method (QM), which exploits geometric properties of a 2D-projection of a velocity field. We show that the QM has state-of-the art localisation performance and how the movement direction of a dipole can be estimated directly from velocity measurements and its estimated location.

The remainder of the present paper is structured as follows: Section 2 explains the fluid flow simulation, our methods of analysis, and the dipole localisation algorithms. Section 3 presents the results of both analysis methods, providing error distributions of the

algorithms as well as the total area with median errors below predefined levels of accuracy. Section 4 places our findings in the context of previous work. Section 5 summarises our contributions and conclusions.

2. Materials and Methods

In the following subsections, we describe the dipole flow field, the simulation environment, the conditions used for our analyses, each dipole localisation algorithm, and their hyperparameter optimisation strategy.

2.1. The Dipole Flow Field

Fluid flows were computed for a small sphere (radius $a = 1$ cm) vibrating with a fraction of its size (amplitude $A = 2$ mm), which generates a dipole field [16]. The dipole is the most informative component of a hydrodynamic stimulus for source localisation with ALLs because the higher terms of a multipole expansion decay with distance more quickly [22]. The lower term—the monopole—is measurable at larger distances. However, it is driven by changes in an object's size, so it is less relevant for localising moving objects. The dipole stimulus has become a popular source for studies with ALLs [5–9,13,14,23–28].

A potential flow model was used to simulate fluid flows produced by a sphere, usually referred to as a dipole field. This model was utilised in several other studies [9,13,25,28] and its usefulness is supported by experimental findings in fish lateral line research [5,25]. Potential flow velocity $\boldsymbol{v}$ is computed by [16]:

$$\boldsymbol{v} = \frac{a^3}{2||\boldsymbol{r}||^3}\left(-\boldsymbol{w} + 3\boldsymbol{r}\frac{(\boldsymbol{w}\times\boldsymbol{r})}{||\boldsymbol{r}||^2}\right), \tag{1}$$

where a is the radius of the sphere, $\boldsymbol{w} = \langle w_x, w_y\rangle$ are the instantaneous velocity components of the moving sphere in 2D, and $\boldsymbol{r} = \boldsymbol{s} - \boldsymbol{p}$ is the location of the sensor $\boldsymbol{s} = \langle x, y\rangle$ relative to the source $\boldsymbol{p} = \langle b, d\rangle$. The dipole's position $\boldsymbol{p}(t)$ and velocity $\boldsymbol{w}(t)$ over time were computed as:

$$\boldsymbol{p}(t) = \boldsymbol{p}_0 + A\begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix}\sin(2\pi f t), \tag{2}$$

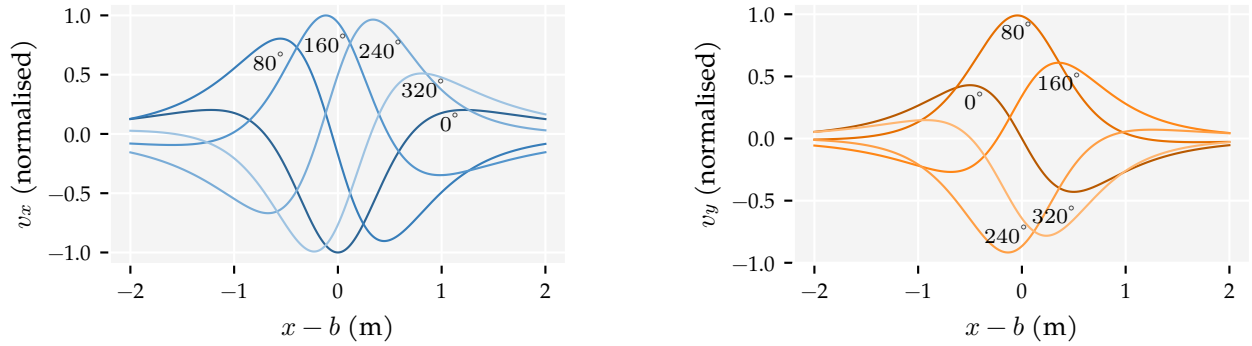
and

$$\boldsymbol{w}(t) = \frac{d\boldsymbol{p}(t)}{dt} = 2\pi f A\begin{bmatrix}\cos(\varphi)\\ \sin(\varphi)\end{bmatrix}\cos(2\pi f t), \tag{3}$$

where A is the amplitude and f the frequency of the oscillation, $\boldsymbol{p}_0$ is the average position of the source, φ is the azimuth angle of the motion, and t indicates time.

We treat localisation as recovering the source's average position $\boldsymbol{p}_0$ from ALL velocity measurements over a period of time. The source's movement during this period did not influence our results because there were an integer number of oscillations in each period. In other words, $\boldsymbol{p}_0$ corresponded precisely with the average position in the measurement segments.

Even though a unique mapping exists between source states and their velocity profiles—i.e., the patterns an infinite continuous linear array of flow sensors would measure—the inverse problem is challenging because sensor arrays only capture a discrete segment of the velocity profile, which may not contain the informative zero-crossings or peaks. Figure 1 shows the parallel (v_x) and perpendicular (v_y) velocity profiles relative to the sensor array. These velocity profiles broaden and their amplitude decays with the distance of the source, reducing the information captured by a fixed-sized ALL. 2D sensitive sensors increase the chance of capturing one of the velocity profile's more informative points because the zero-crossings of one velocity component are located near the peaks of the other velocity component.



(**a**) (**b**)

Figure 1. Normalised continuous velocity profiles for five movement directions (indicated) of a source at $d = 1$ m from the sensor array's centre: (**a**) v_x and (**b**) v_y. The sensors are located along the x axis and b is the source sphere is x position in m.

Another challenge arises in the movement direction estimation. The velocity profiles of objects in the same place but moving in opposite directions differ only in their sign. Consequently, some of the dipole localisation algorithms struggle to differentiate between these source states. We employ a post-processing step to improve these algorithms' movement direction estimation (see Section 2.5).

2.2. Simulation Environment

Eight sensors were simulated, based on the configuration of Wolf et al. [10]. The sensors were placed on the x-axis, centred around $x = 0$. The length of the sensor array was $L = 40$ cm, with 5.71 cm between sensors. The source sphere had a radius of $a = 10$ mm and moved with an amplitude of $A = 2$ mm at a fixed frequency of $f = 45$ Hz. Similar frequency values have been used in the literature: 40 Hz in [7,8,23], 45 Hz in [13,27], and 50 Hz in [26,29]. The source's radius and vibration amplitude are comparable to the work of Abdulsadda and Tan [7,8,24] ($a = 9.5$ mm and $A = 1.91$ mm). Figure 2 shows a schematic view of the present configuration. The source sphere was located between $x = \pm 0.5$ m and from $y = 0.025$ m to $y = 0.525$ m, ensuring its edge was always at least 15 mm away from the closest sensor's centre. The orientation of the source oscillation ranged from $\varphi = 0$ rad to $\varphi = 2\pi$ rad. For each measurement, the fluid velocity at the sensors was simulated for a duration of $T = 1$ s and sampled at 2048 Hz, comparable to the values used by Pandya et al. [6] ($T = 0.5$ s at 2048 Hz). The simulated sensors had a Gaussian sampled velocity-equivalent noise of $\sigma = 1.0 \times 10^{-5}$ m s^{-1}. This value was chosen to be in the top five most sensitive sensors reported by Jiang et al. [30]: 2.5×10^{-6} m s^{-1} [31], 5×10^{-6} m s^{-1} at resonance [20], 8×10^{-6} m s^{-1} [32], 8.2×10^{-6} m s^{-1} [33] and 2×10^{-4} m s^{-1} [34]. The resulting signal to noise ratios (SNRs) are shown for the fifth sensor from the left in Figure 3.

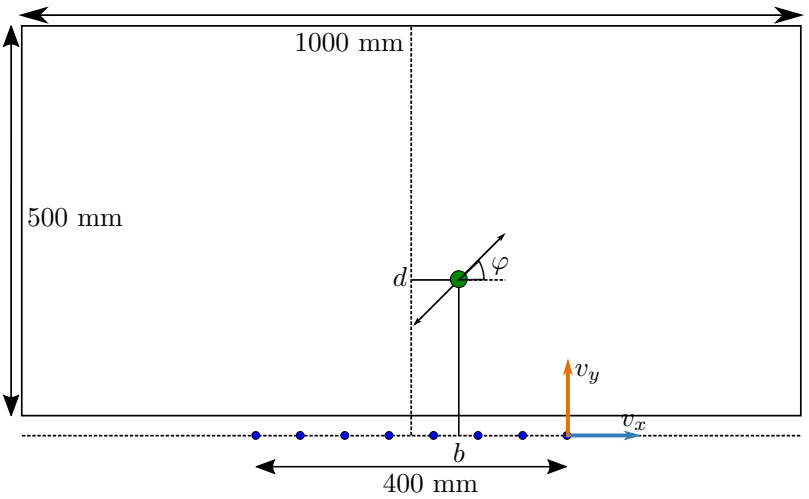


Figure 2. A schematic view of the simulated environment. The source sphere (green) has a radius of 1 cm and is shown to scale. A possible movement direction is shown by the arrow (not in scale). The sensor locations are shown in blue. Parallel v_x and perpendicular v_y velocity components are indicated at the right-most sensor (not to scale). The area in which the source sphere is positioned is offset by 25 mm from the array location, ensuring a minimal distance of 15 mm between the source's edge and closest sensor's centre.

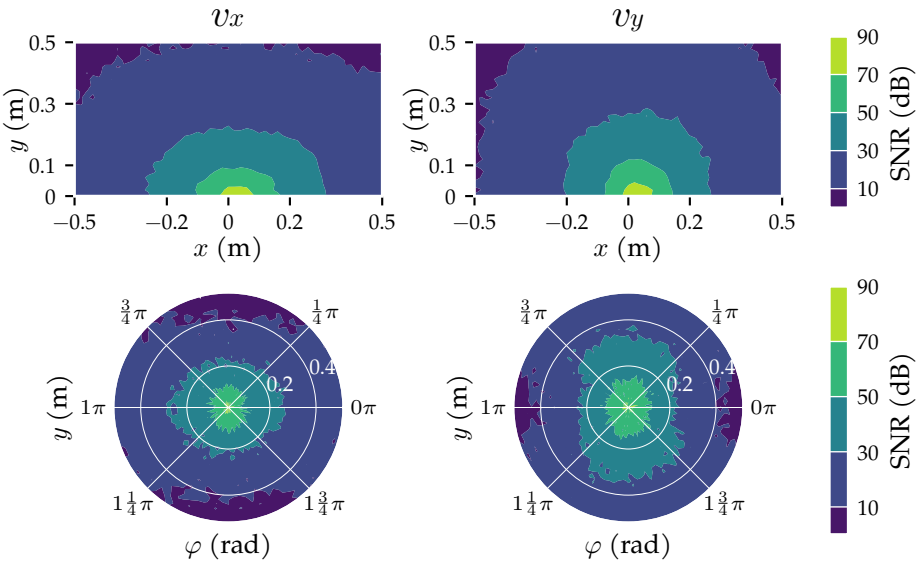


Figure 3. The signal to noise ratio (SNR) of both velocity components measured by the fifth sensor ($x = 2.86$ cm). The top row shows contours of the median SNR in cells of 2×2 cm^2. The bottom row shows the median SNR's polar contours in cells of 0.02π rad $\times$ 2 cm for source states with an x-coordinate between $x = -7.14$ cm and $x = 12.86$ cm, indicating how the movement direction of a dipole φ influences the SNR. Simulated potential flow measurements (Equation (1)) and the same measurements with additive Gaussian distributed noise values ($\sigma = 1.5 \times 10^{-5}$ m s^{-1}, $\mu = 0$ m s^{-1}) were used to compute the SNR. Specifically, the SNR was computed as the frequency power ratio between the noisy measurements and noise floor at the source frequency ($f = 45$ Hz). The frequency power was computed by a discrete Fourier transform (DFT) using a Hamming window.

2.3. Performance Analyses

Two methods of analysis were employed in this research to compare the performance of the dipole localisation algorithms. Analysis Method 1 varied the number of measurements the algorithms could use to train and optimise their hyperparameters. Not every algorithm requires training or has hyperparameters to optimise. For the algorithms that do not have a training phase, we expect a consistent performance regardless of the amount of data. The other algorithms are expected to improve as the amount of training data increases. In this analysis method, all sensors were sensitive to both the parallel and perpendicular velocity component. Analysis Method 2 varied which velocity components were measured by the sensors: (x + y) both components on all sensors, (x | y) alternating v_x and v_y for subsequent sensors, (x) only v_x by all sensors, (y) only v_y by all sensors. In this analysis method, the largest training and optimisation set was used.

The best performing algorithms based on the two analysis methods were also evaluated using higher velocity-equivalent noise levels to show how they perform on sensors with a lower SNR. The increased noise levels (σ) were: 1.0×10^{-3} m s^{-1} and 1.8×10^{-2} m s^{-1} based on [35,36], respectively, as cited in [30]. 1.0×10^{-4} m s^{-1} was added to bridge the gap with the analysis methods' noise-level, which was 1.0×10^{-5} m s^{-1}. All sensors were sensitive to both the parallel and perpendicular velocity component, and the largest training set was used in this comparison.

In all analysis methods, the algorithms' predictions were recorded for each measurement in the test set. A source state with a unique combination of position $\boldsymbol{p} = \langle b, d \rangle$ and movement direction φ was used to generate each measurement. The error of the predicted position $E_{\boldsymbol{p}}$ (m) and movement direction E_φ (rad) were computed as:

$$E_{\boldsymbol{p}}(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \sqrt{(\hat{b} - b)^2 + (\hat{d} - d)^2}, \tag{4}$$

and

$$E_\varphi(\hat{\boldsymbol{P}}, \boldsymbol{P}) = |\text{atan2}\left(\sin\left(\hat{\varphi} - \varphi\right), \cos\left(\hat{\varphi} - \varphi\right)\right)|, \tag{5}$$

where $\boldsymbol{P} = \langle b, d, \varphi \rangle$ are the actual properties of a test state and $\hat{\boldsymbol{P}} = \left\langle \hat{b}, \hat{d}, \hat{\varphi} \right\rangle$ is the prediction based on the test state's velocity measurements. The areas in which the localisation algorithms' median position errors were lower than 1 cm, 3 cm, 5 cm, and 9 cm were computed by discretising the simulated domain in 2×2 cm cells. The areas for the movement direction error were computed similarly with 0.01π rad, 0.03π rad, 0.05π rad, 0.09π rad.

The training and test sets contained randomly sampled source states. Poisson Disc sampling [37] was used to ensure an even spread of states over the simulated domain. The minimum distance between states D_s controlled the number of source states in each set. This distance was computed as the Euclidean distance in the x–y–$\varphi/2\pi$ space containing all possible source states. It can be interpreted as follows: when two states have the same position, their movement direction differs by at least $2\pi D_s$ rad; when they have the same movement direction, their position differs by at least D_s m. The orientation dimension was divided by 2π to balance the number of positions and orientations considered. Table 1 shows the minimum sampling distance, the resulting number of states, and the average minimum distances in terms of position and orientation for each data set. The values of D_s were chosen in terms of the source radius and correspond to the thresholds applied to the position and movement direction errors.

Table 1. An overview of the data sets used in this study. The minimum distance between samples D_s controls the number of source states. A source state is specified by a position $\boldsymbol{p}$ and orientation φ. The distance between two states was computed as the Euclidean distance in the combined x–y–$\varphi/2\pi$ space containing all possible combinations of source positions and movement directions. The orientation dimension was divided by 2π to balance the number of positions and orientations. The testing data set has a different number of source states than the training set with $D_s = 0.01$, due to the randomness of Poisson Disc sampling [37]. The average distance to the closest neighbour within each data set is indicated for both the position and orientation to support the interpretation of D_s.

Type	Min. Sample Distance (D_s)	Num. States	Avg. Min $D_{s,p}$ (m)	Avg. Min $\lvert 2\pi D_{s,\varphi} \rvert$ (rad)
training	0.09	169	2.76×10^{-2}	1.81×10^{-2}
training	0.05	874	1.21×10^{-2}	3.55×10^{-3}
training	0.03	3796	5.72×10^{-3}	8.28×10^{-4}
training	0.01	90,435		
testing	0.01	90,502		

2.4. Parameter Optimisation Approach

Several of the dipole localisation algorithms have hyperparameters, which can be varied to fine-tune their performance. A single error metric that combines and balances both the position and movement direction is required to optimise these parameters. Given that we compare the dipole localisation algorithms based on the area in which they can accurately predict an object's state, the hyperparameter optimisation process should prioritise perfecting accurate predictions over reducing the error of inaccurate predictions. Therefore, we used a normalised absolute error metric E_{norm}:

$$E_{norm}(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \frac{|\hat{b} - b|}{1\text{ m}} + \frac{|\hat{d} - d|}{0.5\text{ m}} + \frac{|E_{\varphi}(\hat{\boldsymbol{P}}, \boldsymbol{P})|}{2\pi\text{ rad}}, \tag{6}$$

where $\boldsymbol{P} = \langle b, d, \varphi \rangle$ are the properties of the test state and $\hat{\boldsymbol{P}} = \left\langle \hat{b}, \hat{d}, \hat{\varphi} \right\rangle$ is the prediction. Compared to an error metric based on the Euclidean distance, the minimisation of E_{norm} is less sensitive to predictions with a large error.

Algorithms that required training were optimised using 5-fold cross-validation (80% training/20% validation split). The other algorithms used the entire training set for a single validation pass. The hyperparameter values that minimised the mean validation error E_{norm} were used in the evaluation with the withheld test set.

Appendix A provides the optimal values of all hyperparameters for each condition of the first two analysis methods. In Analysis Method 3, the algorithms used the optimal values of Analysis Method 1's $D_s = 0.01$ condition.

2.5. Dipole Localisation Algorithms

Each dipole localisation algorithm had access to a training set $\mathcal{T} = \{\langle \boldsymbol{V}^i, \boldsymbol{P}^i \rangle\}_i^N$, where ${\boldsymbol{V}^i}_{(sensors \times time)}$ are the velocity measurements over time and $\boldsymbol{P}^i = \langle b^i, d^i, \varphi^i \rangle$ the state of the ith training dipole. The velocity measurements contain values for v_x and v_y depending on analysis variation. The to-be-predicted velocity profile is denoted as $\tilde{\boldsymbol{V}}_{(sensors \times time)}$.

Not every algorithm explicitly used the time component of the velocity measurements. When required, the time dimension was averaged out by computing a DFT of the signal at each sensor. The magnitude of the 45 Hz components multiplied by the sign of their phase reconstructs the velocity signal over the sensors $\boldsymbol{v}_{(sensors \times 1)}$. Abdulsadda and Tan [24] used this method—without multiplying the sign of the phase—to reduce the influence of noise. A Hamming window was used for computing the DFT.

In the following subsections, each dipole localisation algorithm is described. Table 2 provides a summary of their properties.

Table 2. Properties of the dipole localisation algorithms. The 'Limited to domain' column indicates whether the algorithm can produce predictions outside the simulated domain (see Figure 2). The 'Limited to sample' column indicates whether the algorithm is able to produce a prediction that is not present in the training set.

Algorithm	Type	Training	Hyperparameters	Limited to Domain	Limited to Sample
RND	—	no	no	yes	no
LCMV [12,13]	template-based	yes	no	yes	yes
KNN	template-based	yes	yes	yes	no
CWT [5]	template-based	yes	yes	yes	no
ELM [9,10]	neural network	yes	no	no	no
MLP [7,9]	neural network	yes	no	no	no
GN [8]	model-based	no	yes	yes	no
NR [8]	model-based	no	yes	yes	no
LSQ	model-based	no	yes	yes	no
QM	model-based	no	yes	yes	no

2.5.1. The Random Predictor (RND)

The random predictor is used as a baseline for comparing performance. The algorithm does not use the training set $\mathcal{T}$ nor the to-be-predicted velocity measurement $\tilde{\boldsymbol{V}}$. Instead, it generates a uniform random position and movement direction within the simulated domain (see Figure 2) for every test state.

2.5.2. Linear Constraint Minimum Variance (LCMV) Beamforming

LCMV beamforming was introduced for dipole localisation by Yang et al. [12] and Nguyen et al. [13]. The algorithm computes a prediction by evaluating an energy value E^i of each source state in the training set $\mathcal{T}$ [12,13]:

$$E^i = \frac{1}{\boldsymbol{v}^{iT}\boldsymbol{R}^{-1}\boldsymbol{v}^i}, \tag{7}$$

where $\boldsymbol{R} = \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T$ is the covariance of the to-be-predicted source state, and v^i is the ith training source's velocity measurement with the time dimension averaged-out. The position and movement direction of the training source with the highest energy value is used as prediction $\hat{\boldsymbol{P}} = \left\langle \hat{b}, \hat{d}, \hat{\varphi} \right\rangle$. LCMV is not able to differentiate between sources at the same position but with opposite orientations. To solve this issue, we computed the predicted state's expected potential flow values for both the predicted and opposite movement direction. The movement direction with the smallest difference to the measured velocity was used as the final estimate.

2.5.3. K-Nearest Neighbours (KNN)

The KNN algorithm generalises the template matching approach used by Pandya et al. [6]. KNN computes a prediction by finding the k training states with the most similar velocity measurements $\boldsymbol{V}^i$ compared to the to-be-predicted source's measurements $\tilde{\boldsymbol{V}}$. Before computing the Euclidean distance between the velocity measurements, the time dimension was averaged out from both the training measurements and the measurement of the to-be-predicted source (see Section 2.5). Then, all velocity measurements were normalised by their maximum absolute value. Finally, the average position and movement direction of the k closest training states were computed and used as prediction $\hat{\boldsymbol{P}} = \left\langle \hat{b}, \hat{d}, \hat{\varphi} \right\rangle$. The value of k was optimised, ranging from $k = 1$ to $k = 20$.

2.5.4. The Continuous Wavelet Transform (CWT)

The CWT was introduced for dipole localisation by Ćurčić-Blake and van Netten [5]. The algorithm is based on the observation that potential flow and the pressure gradient along a lateral line can be expressed as wavelets. As in Wolf and van Netten [14], we extend

the family of wavelets to include the perpendicular velocity component. Note, the coordinate system used here is slightly different. Deriving the wavelets from the potential flow formula (Equation (1))—using the approach of Ćurčić-Blake and van Netten [5]—finds (see Appendix B for the derivations):

$$\begin{aligned} v_x &= \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(\psi_e \cos(\varphi) + \psi_o \sin(\varphi)), \\ v_y &= \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(\psi_o \cos(\varphi) + \psi_n \sin(\varphi)), \end{aligned} \tag{8}$$

with

$$\psi_e = \frac{2\rho^2 - 1}{(\rho^2+1)^{(5/2)}}, \tag{9}$$

$$\psi_o = \frac{3\rho}{(\rho^2+1)^{(5/2)}}, \tag{10}$$

$$\psi_n = \frac{2-\rho^2}{(\rho^2+1)^{(5/2)}}, \tag{11}$$

$$\rho = \frac{r_x}{r_y} = \frac{x-b}{y-d}, \tag{12}$$

where $a = 1$ cm is the radius of the source, $\boldsymbol{w} = \langle w_x, w_y \rangle$ is the movement velocity of the source, $\boldsymbol{r} = \boldsymbol{s} - \boldsymbol{p}$ is the relative location of the sensor $\boldsymbol{s} = \langle x, y \rangle$ from the perspective of the source $\boldsymbol{p} = \langle b, d \rangle$, and φ is the azimuth angle of the motion relative to the sensor array.

To compute a prediction, the CWT uses the to-be-localised velocity measurement with the time dimension averaged out $\tilde{v}$ (see Section 2.5) and the source positions $\boldsymbol{p}^i = \langle b, d \rangle$ in the training set $\mathcal{T}$. Note that the CWT does not use the training dipoles' velocity measurements or movement directions. Instead, it computes the values of the wavelets for each position in the training set. These values were evaluated for an extended sensor array matching the simulation domain's width and normalised by their maximum absolute value. Only the values at the eight sensors were kept after the normalisation step. From these values, a vector was constructed for each position $\boldsymbol{p}^i = \langle b, d \rangle$ in the training set $\mathcal{T}$, containing four CWT coefficients: one for each combination of velocity component and wavelet. The peak of a Gaussian surface fitted to this vector's magnitude $Wv(\boldsymbol{p}^i)$ was used to estimate the sources' position $\hat{\boldsymbol{p}} = \langle b, d \rangle$. This fit was constrained to have a peak within the simulated domain (see Figure 2), and only Wv's values between a factor t_{min} and t_{max} of its maximum were used for the fit. The values of t_{min} and t_{max} were optimised, ranging from 0 to 1 under the constraint that $t_{min} < t_{max}$. The movement direction was estimated as the circular mean of:

$$\begin{aligned} \hat{\varphi}_x &= -\operatorname{atan} c_x \frac{Wv_x^o(\hat{\boldsymbol{p}})}{Wv_x^e(\hat{\boldsymbol{p}})}, \\ \hat{\varphi}_y &= -\operatorname{atan} c_y \frac{Wv_y^n(\hat{\boldsymbol{p}})}{Wv_y^o(\hat{\boldsymbol{p}})}, \end{aligned} \tag{13}$$

where $Wv_n^m(\hat{\boldsymbol{p}})$ is the CWT coefficient of $\tilde{v}_{n=(x \text{ or } y)}$ with $\psi_{m=(e,\, o, \text{ or } n)}$. Appendix C shows how these equations were derived and provides the analytical values of c_x and c_y. Unfortunately, the values of c_x and c_y can be shown to depend on the source position for our simulated finite and discontinuous sensor array. Therefore, we optimised the values of c_x (between 0.5 and 1) and c_y (between 0.3 and 0.9). Consequently, the estimation of the movement direction was optimised for the positions where the CWT is accurate.

In total, four hyperparameters were optimised for the CWT (t_{min}, t_{max}, c_x, c_y). The best combination of values was determined in 30 iterations of the Bayesian optimisation algorithm provided by MATLAB [38].

2.5.5. The Extreme Learning Machine (ELM)

The ELM is a neural network designed to provide "the best generalisation performance at extremely fast learning speed" [39]. An ELM is a single layer feed-forward network with randomly initialised weights on the hidden layer. These weights are not changed during training. To find the optimal weights for the output layer, the ELM can be treated as a linear system because the input weights and activation function are fixed. The online sequential ELM variant (OS-ELM) [40] was used to support iterative training. Random initial weights were generated, and the network was trained on the training set $\mathcal{T}$. The velocity measurements were pre-processed by averaging out the time dimension and normalising with their maximum absolute value. Rectified linear units were used as hidden nodes, as recommended in Goodfellow et al. [41] (p. 168). The layer size $\bar{n}$ of the ELM was optimised using 101 values spaced logarithmically ranging from $\bar{n} = 10$ to $\bar{n} = N$, where N is the number of training measurements in the training set $\mathcal{T}$ (see Table 1). The parameter sweep was terminated for the first value of $\bar{n}$ for which the ELM could not be trained due to a singular matrix inversion.

2.5.6. The Multi-Layer Perceptron (MLP)

An MLP was implemented to determine how much better a high capacity network performs compared to the ELM. Rectified linear units were used as hidden nodes, as recommended in Goodfellow et al. [41] (p. 168). Linear activation functions were used on the input and output nodes. The weights of the network were initialised using normalised initialisation, introduced by Glorot and Bengio [42]. Bias-weights were initialised to zero and kept constant during training, essentially disabling them. The network was trained to minimise the mean absolute error (MAE) between the predicted source states $\hat{\boldsymbol{P}}^i = \left\langle \hat{b}^i, \hat{d}^i, \hat{\varphi}^i \right\rangle$ and actual states $\boldsymbol{P}^i = \langle b^i, d^i, \varphi^i \rangle$. This error metric differs from E_{norm} (Equation (6)) because the MAE does not normalise the individual dimensions and does not consider the circular nature of φ.

As indicated earlier, 80% of the training set $\mathcal{T}$ was used for training, the remainder for validation. Each velocity measurement was pre-processed by averaging out the time dimension and normalising by its maximum absolute value. The Adam [43] optimisation algorithm was used with the recommended values for the gradient decay factor $\rho_1 = 0.9$, the squared gradient decay factor $\rho_2 = 0.999$, and denominator offset $\delta = 10^{-8}$ [41] (p. 301). Weight decay was applied with a factor of 10^{-4}. A decay factor of 0.1 was applied to the learning rate ϵ every 10 epochs. The remaining 20% of the training set was used to compute a validation error every 50 iterations. The validation set and training set were each shuffled every epoch. The training was stopped when the validation error did not reach a new minimum in the last 5 evaluations, or the number of epochs exceeded 500. The minibatch size was 2048. The network was pre-trained in three stages to improve the performance on source states close to the sensors, first using states within 20 cm, then 40 cm, and finally 60 cm of the origin.

Three hyperparameters were optimised: the learning rate ϵ ranging from $\epsilon = 10^{-4}$ to $\epsilon = 10^{-1}$, the number of layers n ranging from $n = 1$ to $n = 4$, and the layer sizes $\bar{n}$ ranging from $\bar{n} = 16$ to $\bar{n} = 1024$. Each layer had the same number of nodes to simplify the optimisation. The best combination of parameters was determined in 30 iterations of the Bayesian optimisation algorithm provided by MATLAB [38].

2.5.7. The Gauss–Newton (GN) Algorithm

GN was implemented for dipole localisation by Abdulsadda and Tan [8]. The algorithm does not use the training set $\mathcal{T}$. Instead, it iteratively fits a potential flow model to the absolute value of the to-be-predicted source state's velocity measurements with the time dimension averaged out $|\bar{v}|$. Let $\theta_0 = \langle b_0, d_0, \varphi_0 \rangle$ be an initial estimate. Then the next iteration is given by [8]:

$$\theta_{k+1} = \theta_k + \lambda \left(\nabla |\boldsymbol{v}(\theta_k)|^T \nabla |\boldsymbol{v}(\theta_k)| \right)^{-1} \nabla |\boldsymbol{v}(\theta_k)|^T (|\hat{\boldsymbol{v}}| - |\boldsymbol{v}(\theta_k)|), \tag{14}$$

where λ is a step size parameter, and $\boldsymbol{v}(\theta_k)$ is the potential flow of a source state θ_k computed using Equation (1). The algorithm terminates when the change in θ is smaller than $\epsilon = 1 \times 10^{-3}$, the number of iterations exceeds 100, or the matrix inversion could not be computed due to a singular matrix. The gradient of $|\boldsymbol{v}(\theta_k)|$ was estimated numerically using a step size of $\delta = 1 \times 10^{-3}$. The step size was $\lambda = 1$, because every iteration solves a linearised version of the fitting problem (see [8]).

The simulated domain's centre was used as the initial estimate ($b_0 = 0$ m and $\varphi_0 = \pi$ rad). However, the centre of the d-domain may not be the optimal value for d_0 because states close to the sensors are typically localised more accurately, and the distance between the actual source position and the initial estimate influences the convergence of GN [8]. Therefore, the value of d_0 was optimised and chosen from the range $d_0 = 0.025$ m to $d_0 = 0.525$ m, with a step size 0.025 m. Differently from Abdulsadda and Tan [8], we applied an upper and lower bound on θ_k. After every iteration, the values of b_k and d_k were clipped to the simulated domain (see Figure 2), and a modulo 2π operation was applied to φ_k.

A single post-processing step was performed to improve the movement direction estimation. GN is not able to differentiate between sources at the same position with opposite orientations because it uses the absolute velocity measurements to fit a potential flow. To solve this issue—as with the LCMV beamforming algorithm—we computed the predicted state's expected potential flow values for both the predicted and opposite movement direction. The movement direction with the smallest difference to the measured velocity was used as the final estimate.

2.5.8. The Newton–Raphson (NR) Algorithm

NR was also introduced for dipole localisation by Abdulsadda and Tan [8]. The algorithm is very similar to GN; only the hyperparameter optimisation and update step are different. Let $\theta_0 = \langle b_0, d_0, \varphi_0 \rangle$ be an initial estimate. Then, the next iteration is given by [8]:

$$\theta_{k+1} = \theta_k - \lambda \boldsymbol{G}(\theta_k)^{-1} \boldsymbol{g}(\theta_k), \tag{15}$$

with

$$\boldsymbol{g}(\theta) = \nabla |\boldsymbol{v}(\theta)|^T (|\hat{\boldsymbol{v}}| - |\boldsymbol{v}(\theta)|), \tag{16}$$

and

$$\boldsymbol{G}(\theta) = \frac{\partial \boldsymbol{g}(\theta)}{\partial \theta}, \tag{17}$$

where λ is a step size parameter, $\boldsymbol{v}(\theta_k)$ is the potential flow of a source θ_k computed using Equation (1), and $|\hat{\boldsymbol{v}}|$ is the absolute value of the to-be-predicted source state's velocity measurements with the time dimension averaged out. The gradient $g(\theta)$ and Hessian $G(\theta)$ were estimated numerically with a step size of $\delta = 1 \times 10^{-3}$. The step size λ, stopping conditions, bounds check, and initial estimate were the same as for GN. Different from GN, a norm limit l was applied to the change in θ in each iteration. The value of l was optimised, ranging from $l = 0.1$ to $l = 1$. The best combination of d_0 and l was determined in 30 iterations of the Bayesian optimisation algorithm provided by MATLAB [38]. The same post-processing step was applied as in GN, to improve the movement direction estimation.

2.5.9. The Least Square Curve Fit (LSQ) Algorithm

The LSQ predictor was implemented as another baseline for comparing performance. The algorithm does not require training and does not have hyperparameters to optimise. As a result, the training set $\mathcal{T}$ is not used by LSQ. Therefore, the number of training states does not influence the performance of LSQ. Consequently, the algorithm is only used in the second analysis method.

LSQ computes a prediction using the *lsqcurvefit* function from MATLAB [38]. The algorithm fits a potential flow model $\boldsymbol{v}(\theta)$ (Equation (1)) to $\hat{\boldsymbol{v}}$: the to-be-predicted source state's velocity measurement with the time dimension averaged out. The *lsqcurvefit* function implements the *trust-region-reflective* algorithm (see [44]). Lower and upper bounds were provided for all elements of $\theta = \langle b, d, \varphi \rangle$, to limit their values to the simulated domain (see Section 2.2). Gradients were estimated numerically by *lsqcurvefit*, using a step size of $\delta = 1 \times 10^{-3}$. The algorithm terminated when the number of iterations exceeded 100 or the change in θ was smaller than $\epsilon = 1 \times 10^{-3}$. The function and optimality tolerance checks were set to machine-precision. The initial estimate θ_0 is identical to GN.

2.5.10. The Quadrature Method (QM) Algorithm

QM is a newly proposed localisation algorithm that takes advantage of 2D sensitive sensors. The algorithm combines measurements of v_x and v_y to construct a curve, ψ_{quad}, that has characteristics that are nearly independent of the orientation of a source:

$$\psi_{quad} = \sqrt{v_x^2 + \frac{1}{2} v_y^2}. \tag{18}$$

Combining v_x and v_y in this way is somewhat analogous to computing the overall amplitude of two quadrature time signals. Appendix D explains the properties of the quadrature curve in more detail. Summarising, the factor $1/2$ can be shown to negate the influence of the orientation φ at the maximum of the curve, allowing for an estimate of the lateral position b that is virtually independent of the orientation φ. The distance of the source d is linearly related to a measure of the width of ψ_{quad}. Two so-called anchor points $\rho^{\pm}_{anch}$ are used to measure this width. These anchor points are located where ψ_{quad} takes a value of about 0.458 times the curve's maximum and their location is almost independent of the dipole's orientation. Note, ρ (Equation (12)) describes a source state's location along the sensor array normalised by its distance. Practically, though, the locations of these anchor points are estimated in terms of x. Given the location of the anchor points, the source distance d is computed as:

$$d = \frac{1}{1.79} (\rho^{+}_{anch} - \rho^{-}_{anch}). \tag{19}$$

In practice, the locations of the anchor points were estimated by linearly interpolating ψ_{quad} between the two sensors where ψ_{quad} intersected with 0.458 times its maximum value.

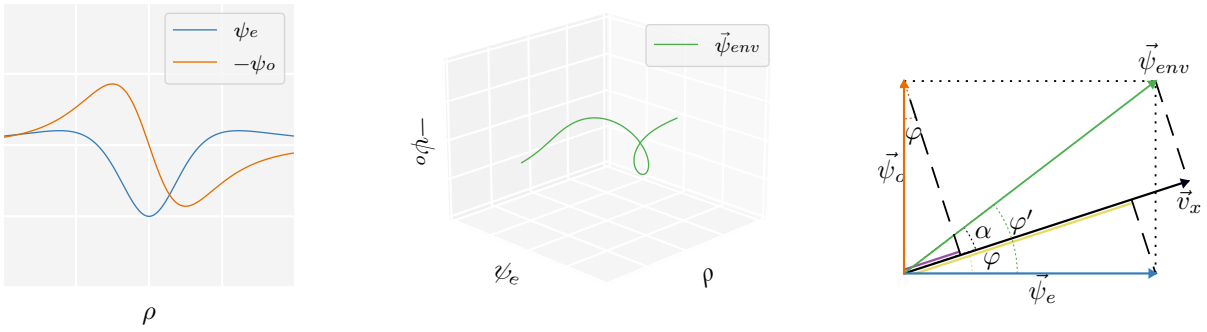
An accurate estimate of the movement direction φ can be computed directly from the measured velocity values once the source state's position $\boldsymbol{p} = \langle b, d \rangle$ is known. For this purpose, the measured velocity is analysed using the wavelets from the CWT (see Section 2.5.4). Figure 4 visualises the wavelets and their form in a 3D ψ_e–ψ_o–ρ space. The movement direction can be recovered from ψ_e–ψ_o slices of this 3D space at the sensor locations (Figure 4c). In these slices, the wavelets and the measured velocity are vectors ($\vec{\psi}_e$, $\vec{\psi}_o$, and $\vec{v}_x$) with lengths corresponding to their values at the sensor. Note that all these values are known because the wavelets can be computed from the estimated position. The angle between $\vec{v}_x$ and $\vec{\psi}_e$ corresponds to the to-be-predicted movement direction φ. To compute φ, we use the vector combination of the wavelets $\vec{\psi}_{env} = \vec{\psi}_e + \vec{\psi}_o$, which has a length $\psi_{env} = \sqrt{\psi_e^2 + \psi_o^2}$. The angle between $\vec{\psi}_{env}$ and $\vec{\psi}_e$ is $\varphi' = \operatorname{atan} \psi_o / \psi_e$. The difference between φ and φ' is $\alpha = \operatorname{acos} v_x / \psi_{env}$ because v_x is a linear combination of the two wavelets. Consequently, the movement φ can be estimated using:

$$\varphi = \varphi' \pm \alpha, \tag{20}$$

Depending on the source's position and movement direction and the sensors' location, α should be added or subtracted from φ'. Therefore, two estimates were computed for each sensor's v_x measurement. The circular median of all sixteen estimates was used as final prediction. This estimation approach also works for v_y when using ψ_o and ψ_n in place of ψ_e and ψ_o.

The predictions based on the anchor points' estimated locations are limited because the spatial resolution of the sensor array limits the accuracy of the anchor point location estimation. In addition, one or both anchor points may not be within the measurable range of the ALL.

A refinement step was introduced to improve the predictions. First, the estimate of the position was improved by fitting a potential flow model to ψ_{quad}. The fit was computed using the *lsqcurvefit* function from MATLAB [38]. The position and orientation estimates computed using the anchor points were used as starting estimate $\theta_0 = \langle b_0, d_0, \varphi_0 \rangle$. Lower and upper bounds were provided for all elements of θ. Gradients were estimated numerically by *lsqcurvefit*, with a step size of $\delta = 1 \times 10^{-3}$. The fit procedure terminated when the number of iterations exceeded 10, or the change in θ was smaller than $\epsilon = 1 \times 10^{-3}$. The function and optimality tolerance checks were set to machine precision. Then, the orientation was re-estimated using the improved position estimate. The complete refinement step was repeated four times, each iteration using the previous estimated position and orientation as the starting point.



(**a**) (**b**) (**c**)

Figure 4. Graphical illustration of the movement direction estimation from the measured velocity and the source's position $\boldsymbol{p} = \langle b, d \rangle$. (**a**) A view of $\psi_e(\rho)$ and $\psi_o(\rho)$ along the sensor array. (**b**) The values of the wavelets can be interpreted as vectors ($\vec{\psi}_e$ and $\vec{\psi}_o$) in a 3D ψ_e–ψ_e–ρ space. Their vector combination $\vec{\psi}_{env} = \vec{\psi}_e + \vec{\psi}_o$ is a fixed 3D wavelet structure that can be constructed solely from the source's previously determined position $\boldsymbol{p}$. This vector $\vec{\psi}_{env}$ has a magnitude $\psi_{env} = \sqrt{\psi_e^2 + \psi_o^2}$ and angle $\varphi' = \operatorname{atan} \psi_o / \psi_e$. The measured velocities—which are linear combinations of ψ_e and ψ_o—can be viewed as a 2D projection of this 3D wavelet. For instance, a projection on the ρ–ψ_e plane (bottom plane) yields ψ_e for $\varphi = 0$ rad. For a general angle φ, the measured velocity profile is a projection on a plane through the ρ axis subtending an angle φ with the ρ–ψ_e plane. (**c**) Diagram illustrating the geometric relation between a measured v_x, the angles α and φ' which are constrained via ψ_{env}, and the movement orientation φ. We show a slice of $\vec{\psi}_{env}$ (green) in the ψ_e–ψ_o plane for a fixed value of ρ. The velocity value at this fixed ρ is a vector $\vec{v}_x$ (black) in this space. It has a length $v_x \propto \psi_e \cos\varphi + \psi_o \sin\varphi$ and has angle φ. The contributions of $\vec{\psi}_e$ (blue) and $\vec{\psi}_o$ (orange) to v_x are shown in yellow and purple. The angle φ' of $\vec{\psi}_{env}$ can be computed directly from an estimated source position. Given that the difference between φ and φ' is $\alpha = \operatorname{acos} v_x / \psi_{env}$, we can compute an estimate of φ at every sensor using only measured velocity values and a position estimate.

3. Results

In this research, ten dipole localisation algorithms were compared by the area in which they accurately estimate the position and movement direction of an object. Specifically, the area with a median position error $E_{\boldsymbol{p}}$ below 1 cm, 3 cm, 5 cm, and 9 cm, and the area with a median orientation error E_φ below 0.01π rad, 0.03π rad, 0.05π rad, and 0.09π rad are reported. Section 3.1 presents the first analysis' results, where we varied the available data set size, and Section 3.2 presents the second analysis' results, where the sensor sensitivity direction was varied. Section 3.3 provides additional results to compare the localisation algorithms, including the performance of the best three algorithms on simulated sensors with lower SNRs.

3.1. Analysis Method 1: Amount of Training and Optimisation Data

This analysis method varied the number of measurements the algorithms could use to train and optimise their hyperparameters to show how that influences the algorithms' performance. LSQ was not included in this analysis because it does not require training nor has hyperparameters to optimise. However, LSQ's results from the comparable (x + y) condition of Analysis Method 2 are shown to allow for a comparison of the performance of all algorithms.

The results of this analysis are summarised in Figures 5 and 6. These figures visualise the total area in which the median position error and median orientation error were below their respective thresholds. There are several observations of note. Firstly, as expected, the model-based algorithms' areas did not increase with the amount of training and optimisation data. With $D_s = 0.09$, we found 0.21 m^2 for QM, 0.22 m^2 for GN, and 0.11 m^2 for NR at $E_p \leq 1$ cm. In contrast, the template-based algorithms' and the neural networks' areas did increase with the training and optimisation sets. Using the largest data set $D_s = 0.01$, these algorithms achieved a position error lower than 1 cm in areas of 0.18 m^2 for MLP, 0.14 m^2 for KNN, 0.12 m^2 for LCMV, 0.1 m^2 for ELM, and 0.00 m^2 for CWT. Only the random predictor had a median position error larger than 9 cm everywhere.

Secondly, only QM, GN, and NR had a median orientation error below 0.01π rad in some part of the simulated domain with $D_s = 0.09$, with areas of 0.06 m^2, 0.06 m^2, and 0.02 m^2, respectively. With the largest training and optimisation set ($D_s = 0.01$), the MLP and KNN reached the 0.03π rad threshold in 0.09 m^2 and 0.02 m^2, respectively, whereas LCMV only reached 0.05π rad in an area of 0.04 m^2. The other predictors had a median orientation error larger than 0.09π rad everywhere.

The predictors' performance is not only characterised by the area in which they work well; it is also important to show how accurate predictions were in areas where they did not work well. To provide a complete picture of the predictor performance, Figures 7 and 8 visualise the distributions of the position and orientation errors, respectively.

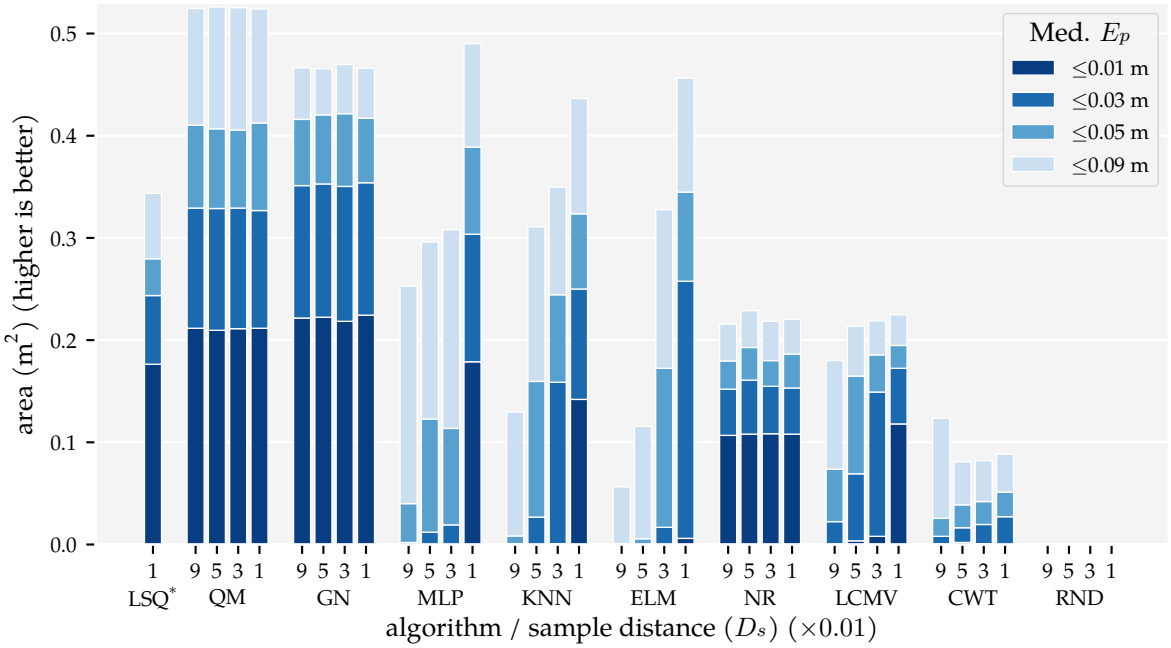


Figure 5. Total areas with a median position error E_p (Equation (4)) below 1 cm, 3 cm, 5 cm, and 9 cm for the training and optimisation sets with a varying minimum distance between source states D_s (see Section 2.3 and Table 1) and the (x + y) sensor configuration at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The median position error was computed for 2×2 cm^2 cells. Note, the bar for LSQ* is based the (x + y) condition in Analysis Method 2.

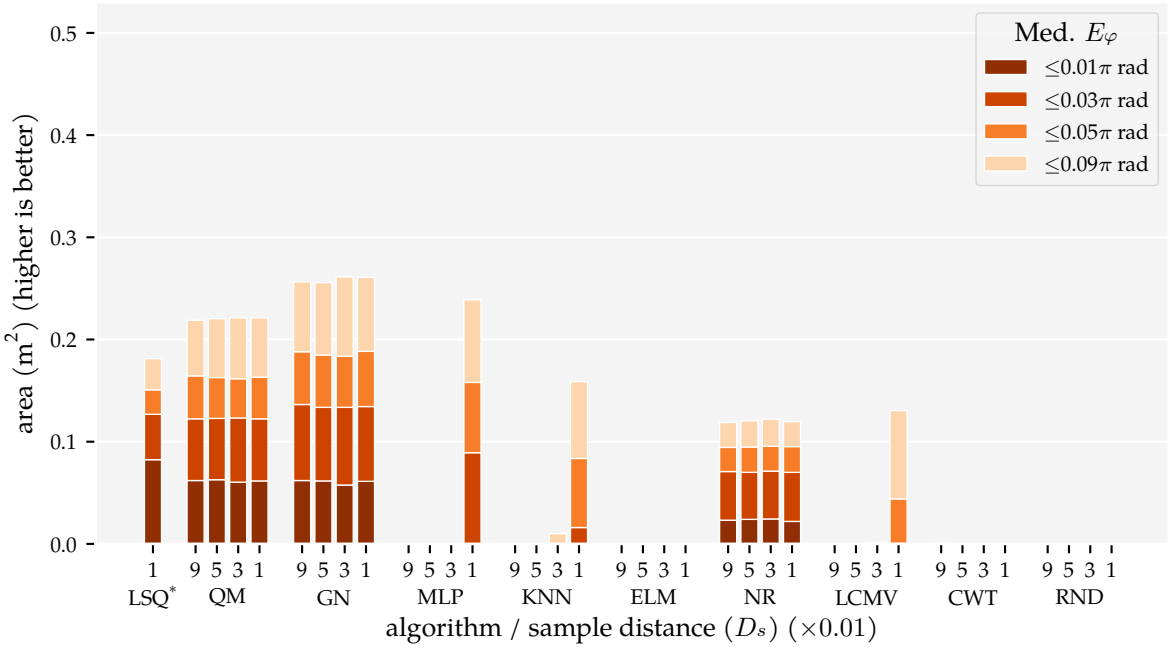


Figure 6. Total areas with a median movement direction error E_φ (Equation (5)) below 0.01π rad, 0.03π rad, 0.05π rad for the training and optimisation sets with a varying minimum distance between source states D_s (see Section 2.3 and Table 1) and the (x + y) sensor configuration at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The median movement direction error was computed for 2 × 2 cm^2 cells. Note, the bar for LSQ* is based on the (x + y) condition in Analysis Method 2.

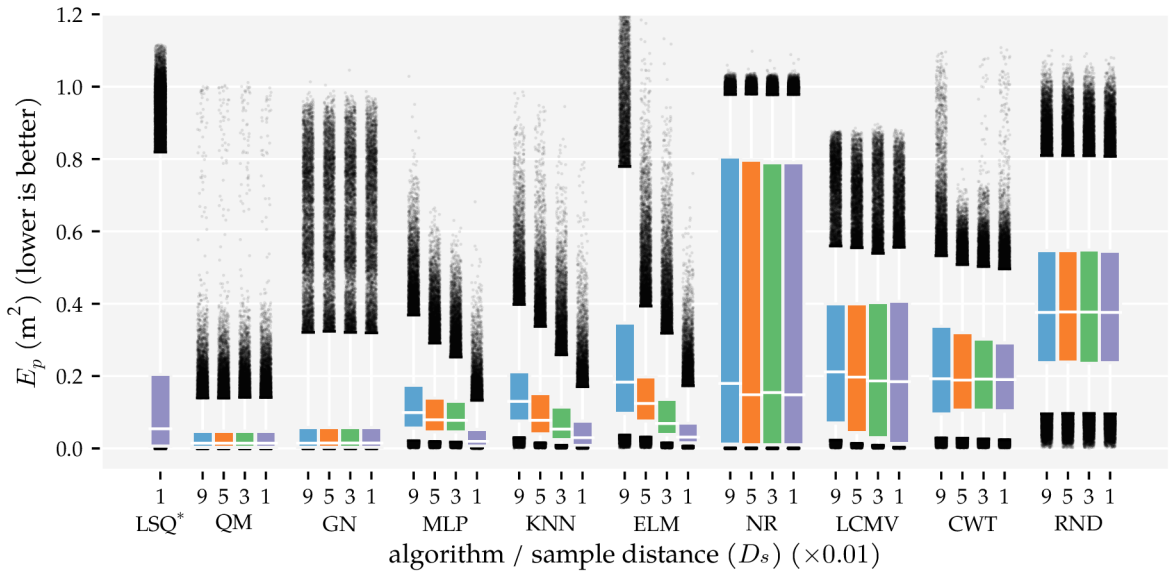


Figure 7. Boxplots of the position error distributions for all dipole localisation algorithms in each condition of first analysis method. This analysis varied the minimum distance D_s between source states in the training and optimisation set (see Section 2.3 and Table 1) and used the (x + y) sensor configuration at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The whiskers of the boxplots indicate the 5th and 95th percentiles of the distributions. Predictions with errors outside these percentiles are shown individually. LSQ* is based on the (x + y) condition in Analysis Method 2.

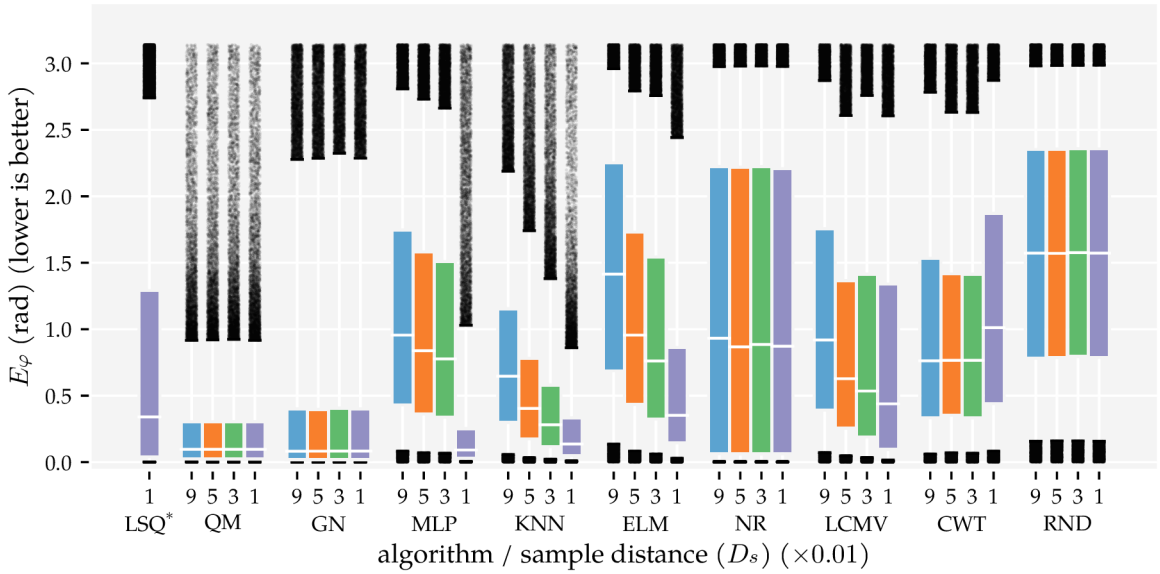


Figure 8. Boxplots of the movement direction error distributions for all dipole localisation algorithms in each condition of first analysis method. This analysis varied the minimum distance D_s between source states in the training and optimisation set (see Section 2.3 and Table 1) and used the (x + y) sensor configuration at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The whiskers of the boxplots indicate the 5th and 95th percentiles of the distributions. Predictions with errors outside these percentiles are shown individually. LSQ* is based on the (x + y) condition in Analysis Method 2.

Firstly, note that all dipole localisation algorithms had lower median position errors than the random predictor with all training and optimisation sets. This difference was smaller for the median orientation error. In particular, ELM's orientation error distribution with the smallest training and optimisation set was similar to that of the random predictor. Secondly, the position error distribution of NR had a large tail; the 75th and 95th percentiles were 0.81 m and 0.98 m with $D_s = 0.09$, respectively. The GN predictor also had a higher 95th percentile (0.32 m) than the otherwise similarly performing QM predictor (0.14 m) with $D_s = 0.09$. The same relation was also present in the orientation error distributions. The 95th percentile of GN (2.28 rad) was larger than that of QM (0.92 rad) with $D_s = 0.09$.

Another interesting observation is the difference between the development of the orientation error distributions of MLP and KNN. The errors of KNN decreased gradually with each increase in training and optimisation data, whereas the errors of the MLP decreased drastically between $D_s = 0.03$ and $D_s = 0.01$. Note that the MLP used four layers with $D_s = 0.01$ and one layer with all the other training and optimisation sets (see Table A1). Finally—unlike the position error—the CWT's orientation errors were larger with the largest optimisation set compared to the smaller sets.

3.2. Analysis Method 2: Sensor Sensitivity Axes

This analysis method varied which velocity components were measured by the sensors to determine how that influences the algorithms' performance. The QM predictor was not included in this analysis because it requires both velocity components to be measured by all sensors. However, QM's results from the comparable $D_s = 0.01$ condition of Analysis Method 1 are shown to allow for a comparison of the performance of all algorithms.

The results of this analysis are summarised in Figures 9 and 10. These figures visualise the total area in which the median position error and median movement direction error were below their respective thresholds. In configuration (x + y)—which measured both velocity components at all sensors—the GN predictor had the largest area with median position error lower or equal to 1 cm (0.22 m^2). The MLP and LSQ predictors followed with 0.19 m^2 and 0.18 m^2, respectively. The NR predictor (0.11 m^2) performed worse than KNN (0.14 m^2) and LCMV (0.12 m^2). The ELM did not perform well at the 1 cm threshold (0.01 m^2). However, the areas at the higher thresholds were similar to those of the MLP.

The CWT and RND predictors had a median position error larger than 1 cm in the entire simulated domain.

In the other configurations, LSQ performed the best; the median position error was lower or equal to 1 cm in 0.12 m^2 in configuration (x), and 0.15 m^2 in configuration (y) and (x | y). In general, the areas with median position errors lower or equal to 3 cm were larger with the alternating configuration (x | y) compared to configurations (x) or (y). The CWT predictor is an exception; it performed the best in configuration (x) at all position error thresholds. For GN and NR, the larger area at the 3 cm threshold in configuration (x | y) went along with a smaller area at the 1 cm threshold compared to configuration (x).

The movement direction errors, shown in Figure 10, indicate a slightly different pattern. The LSQ predictor had the largest areas with a median movement direction error below 0.01π rad in all configurations (0.05 m^2 in (x) and (x | y), 0.07 m^2 in (y), and 0.08 m^2 in (x + y)). The GN predictor performed better than LSQ at the higher thresholds in configuration (x + y) (0.19 m^2 to 0.15 m^2 at $E_\varphi \leq 0.05\pi$ rad and 0.26 m^2 to 0.18 m^2 at $E_\varphi \leq 0.09\pi$ rad). Interestingly, the MLP performed better than KNN, especially in configuration (x | y) and (x + y) (see Figure 10).

Similar to the first analysis (Section 3.1), we also show the error distributions in Figures 11 and 12. The benefit of using both v_x and v_y (x + y) is visible in the lower median, 75th, and 95th percentiles. The alternating configuration (x | y) also resulted in lower median, 75th, and 95th percentile values compared to configurations (x) and (y), except for the NR, CWT and RND predictors. Similar to the results in Analysis Method 1, the MLP, KNN, and ELM predictors have lower 95th percentiles in the position error distribution in configuration (x + y) (0.13 m, 0.17 m, and 0.16 m, respectively) than the model-based algorithms (0.32 m for GN, 0.82 m for LSQ, 0.98 m for NR).

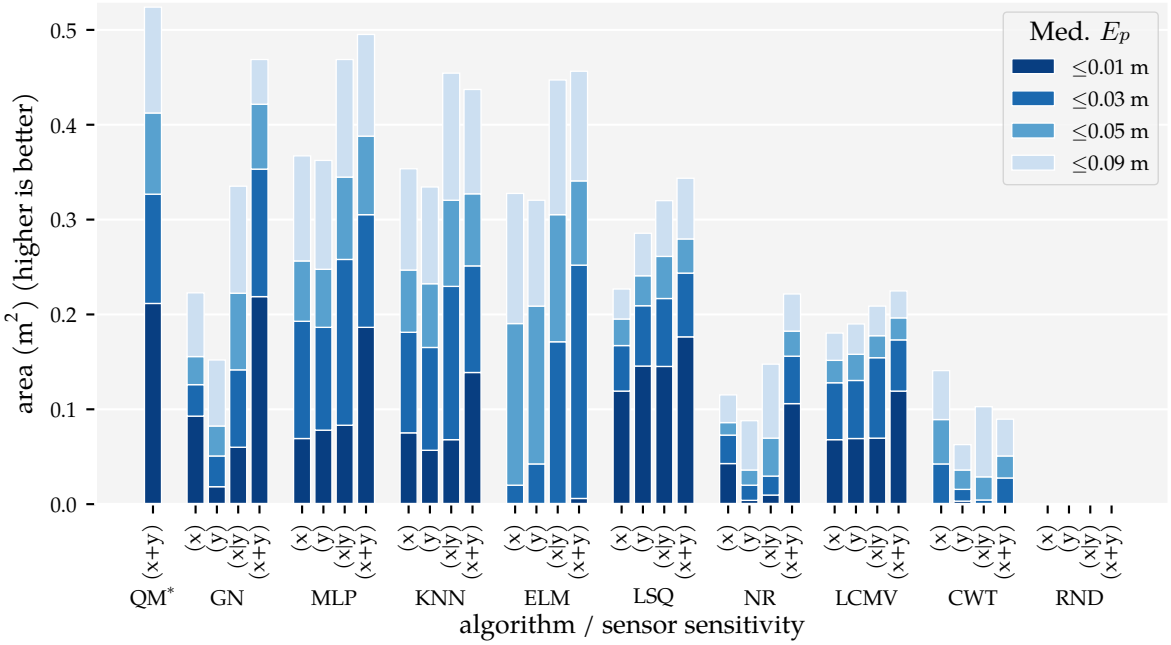


Figure 9. Total area with a median position error E_p (Equation (4)) below 1 cm, 3 cm, 5 cm, and 9 cm for varying sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. This analysis method used the $D_s = 0.01$ training and optimisation set and the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The median position error was computed in 2×2 cm^2 cells. Note, the bar for QM* is based on the $D_s = 0.01$ condition in Analysis Method 1.

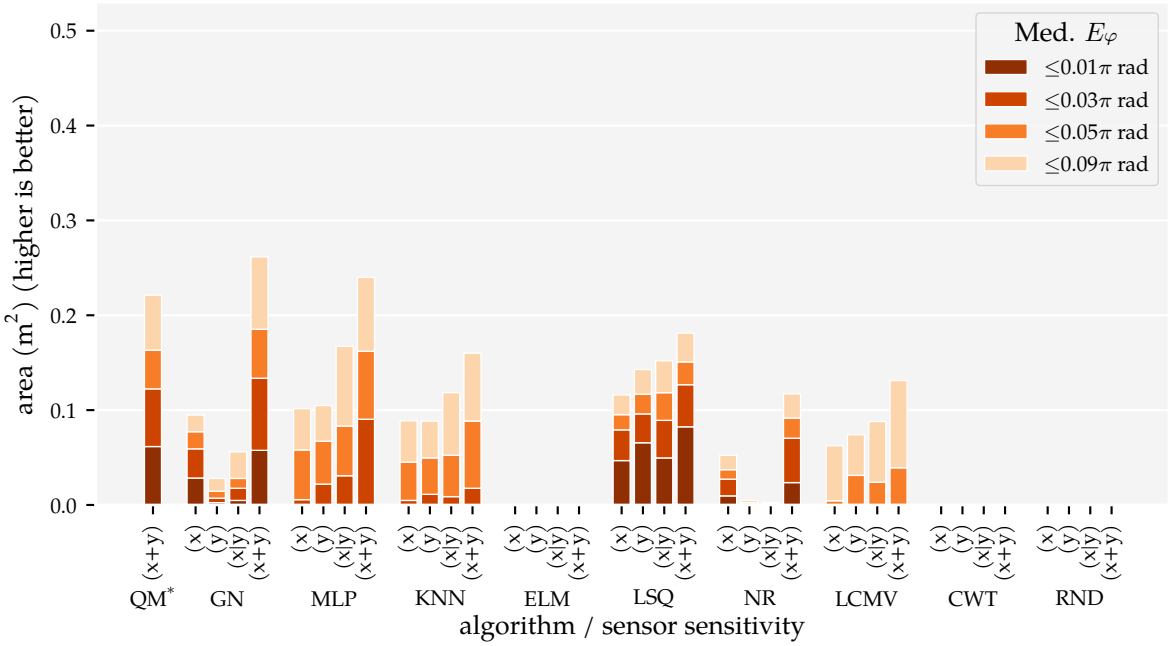


Figure 10. Total area with a median movement direction error E_p (Equation (4)) below 1 cm, 3 cm, 5 cm, and 9 cm for varying sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. This analysis method used the $D_s = 0.01$ training and optimisation set and the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The median movement direction error was computed in 2×2 cm^2 cells. Note, the bar for QM* is based on the $D_s = 0.01$ condition in Analysis Method 1.

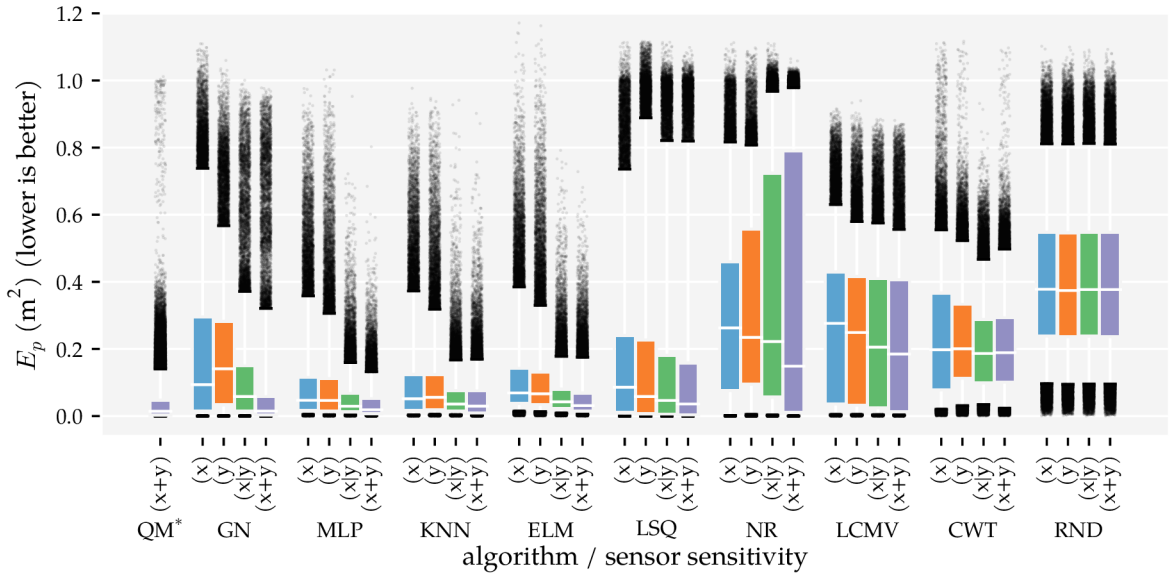


Figure 11. Boxplots of the position error distributions for all algorithms in the second analysis method. This analysis varied the sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. The $D_s = 0.01$ training and optimisation set and $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level were used. The whiskers of the boxplots indicate the 5th and 95th percentiles of the distributions. Predictions with errors outside these percentiles are shown individually. QM* is based on the $D_s = 0.01$ condition in Analysis Method 1.

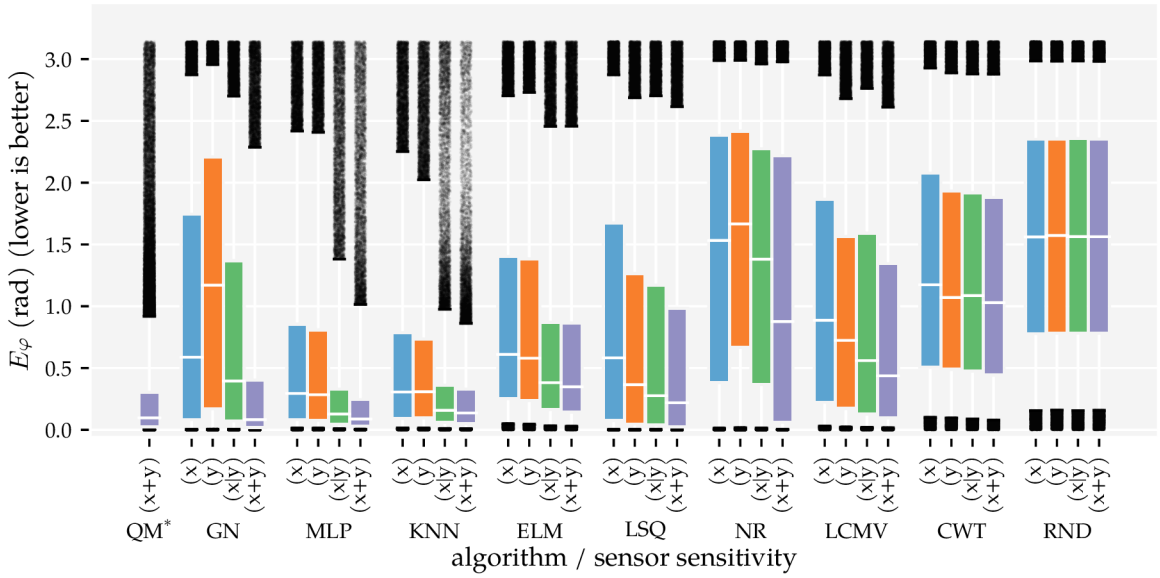


Figure 12. Boxplots of the movement direction error distributions for all algorithms in the second analysis method. This analysis varied the sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. The $D_s = 0.01$ training and optimisation set and $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level were used. The whiskers of the boxplots indicate the 5th and 95th percentiles of the distributions. Predictions with errors outside these percentiles are shown individually. QM* is based on the $D_s = 0.01$ condition in Analysis Method 1.

3.3. Additional Results

This section presents additional results of the two analysis methods and the performance of the three best algorithms on simulated sensors with lower SNRs. The algorithms' training and prediction times are reported, and spatial and polar maps of the median errors are shown, visualising the areas in which the predictors performed well and the effect of the source's orientation on the prediction accuracy.

Table 3 shows each localisation algorithms' average prediction time using both velocity components and total training time for the largest training set ($D_s = 0.01$). All algorithms were evaluated on a high performance computing cluster, using 12 cores of an Intel Xeon 2.6 GHz processor and 64 GB RAM. The run-time performance of the implementations was optimised for computing many predictions simultaneously. As a result, the average prediction times may not be representative of the single-source prediction time. The random predictor had the shortest prediction time, followed by the MLP and ELM. KNN was also one of the quicker algorithms, about three times slower than the MLP. The model-based predictors were slower: the MLP could compute roughly four, eight and nine predictions in the time it took for GN, LSQ and QM to compute a prediction. The remaining algorithms were considerably slower than the MLP (36 times for LCMV, 175 times for NR, and 300 times for CWT). Most of the CWT's prediction time came from fitting the Gaussian to the coefficients. It should be noted that the computational aspects of our implementations have not been extensively optimised, and the degree of optimisation between algorithms may have varied.

Table 3. Training and prediction time measurements of all dipole localisation algorithms. The (x + y) sensor configuration was used combined with the largest training and optimisation set $D_s = 0.01$.

Algorithm	Avg. Prediction Time	Relative to MLP	Total Training Time
RND	3.2×10^{-4} s	0.9	
MLP	3.6×10^{-4} s	1.0	12 min 0 s
ELM	4.3×10^{-4} s	1.2	1 min 52 s
KNN	9.7×10^{-4} s	2.7	
GN	1.4×10^{-3} s	3.9	
LSQ	2.8×10^{-3} s	7.8	
QM	3.4×10^{-3} s	9.6	
LCMV	1.3×10^{-2} s	37.1	
NR	6.3×10^{-2} s	176.3	
CWT	1.1×10^{-1} s	311.1	

Figure 13 shows the spatial contours of the median position error E_p and median movement direction error E_φ. These figures show in which areas the algorithms were able to compute an accurate prediction. Both QM and GN had a median position error within 1 cm up to roughly 35 cm from the sensor array. The GN predictor was better at locating source states in the lower corners of the simulated domain than LSQ and QM. However, its predictions directly in front of the sensors were worse. The MLP and KNN also performed well in the lower corners of the simulated domain. However, their median position error was lower than 1 cm up to only roughly 25 cm. For the LSQ and LCMV predictors, the median position errors were smaller than 1 cm up to 30 cm and 20 cm, respectively. These algorithms showed a quick drop in performance as the distance of a source state increases. Their median position errors were larger than 9 cm from 30 cm and 45 cm of the sensor array, respectively. QM, GN and the MLP had a median position error lower than 9 cm at least up to 50 cm. The contours in the orientation errors were similar. Comparing the two neural networks, the MLP had more accurate movement direction predictions than the ELM and also covered a larger area of the simulated domain.

Polar contours of the median position and median movement direction errors are shown in Figure 14, indicating the effect of source movement direction φ and distance d on the predictions. QM and GN performed similarly in the (x + y) configuration; both algorithms had accurate position and movement directions predictions for all source orientations φ. In configuration (x + y), QM's movement direction prediction was slightly more accurate at longer distances for parallel and perpendicular movement directions. Closer to the sensors, QM's movement direction prediction was more accurate for source states with a parallel orientation. GN had accurate movement direction predictions for parallel source states at all distances in configuration (x + y). The difference in movement direction estimation between QM and GN shown in Figure 14 can be explained by GN's wider range of accurate predictions visible in Figure 13. LSQ had trouble predicting the position and movement direction of source states with orientations between $\varphi = \pm\frac{1}{4}\pi$ rad—except for parallel sources $\varphi = 0$ rad—in configurations (x + y) and (y). GN's localisation performance in configurations (x) and (y) is interesting as well, as it favoured perpendicular source states. On the other hand, GN's movement direction predictions in configuration (y) were inaccurate regardless of the source state's orientation φ and distance d.

Figure 15 shows how QM, GN, and the MLP perform using sensors with lower SNRs. Only the position error is shown. Appendix E contains a similar figure for the movement direction errors, as well as the spatial and polar projections. QM has the largest area with median position errors below 1 cm at the higher noise levels. However, from $\sigma = 1 \times 10^{-4}$ m s^{-1} and up, the MLP has lower median and 75 percentile values.

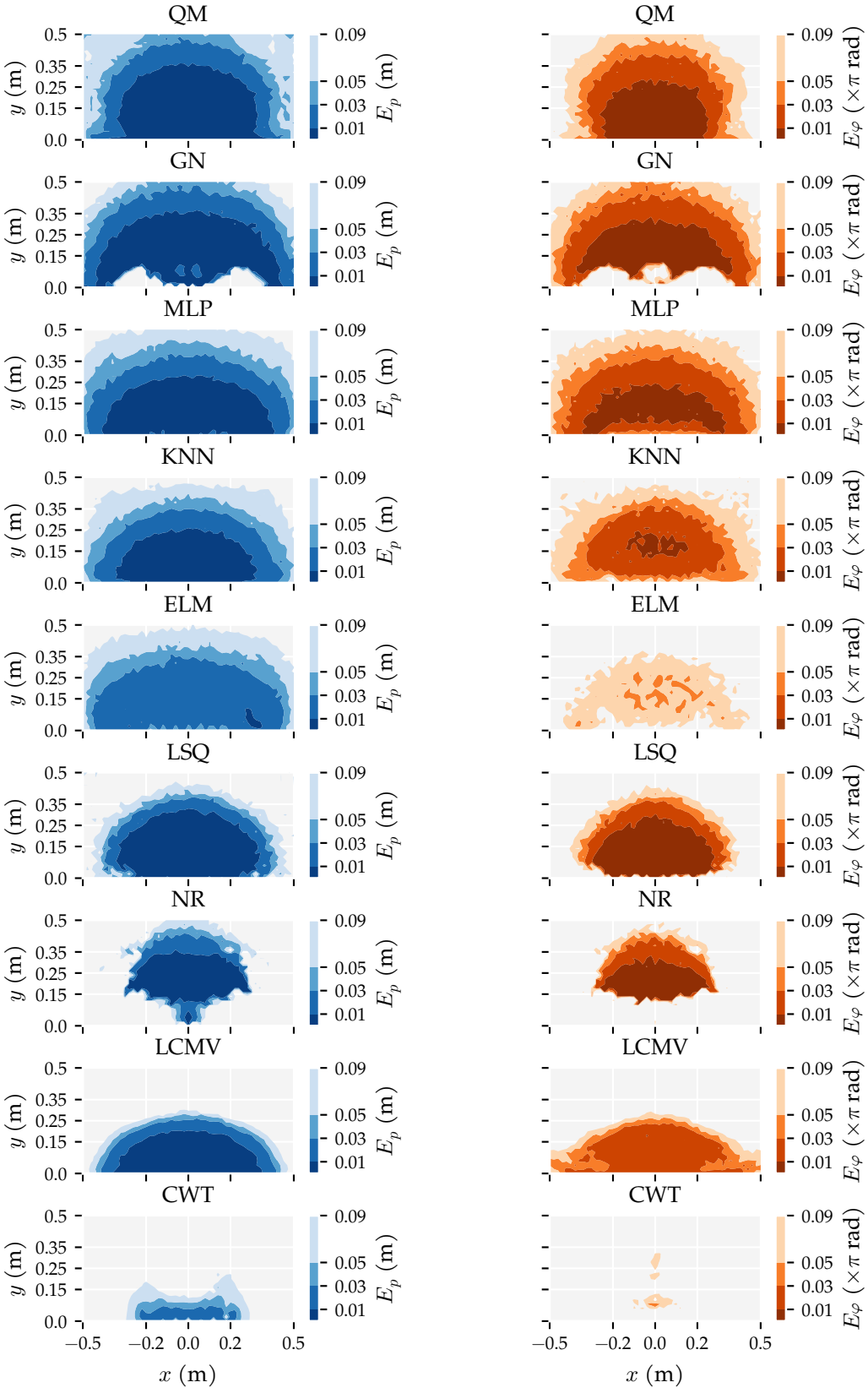


Figure 13. Spatial contours of the median position error E_p (blue) (Equation (4)) and median movement direction error E_φ (orange) (Equation (5)) of the predictors using the largest training and optimisation set ($D_s = 0.01$) and 2D sensitive sensors (x + y) at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The algorithms are ordered with an increasing overall median position error. The median errors were computed in 2×2 cm^2 cells.

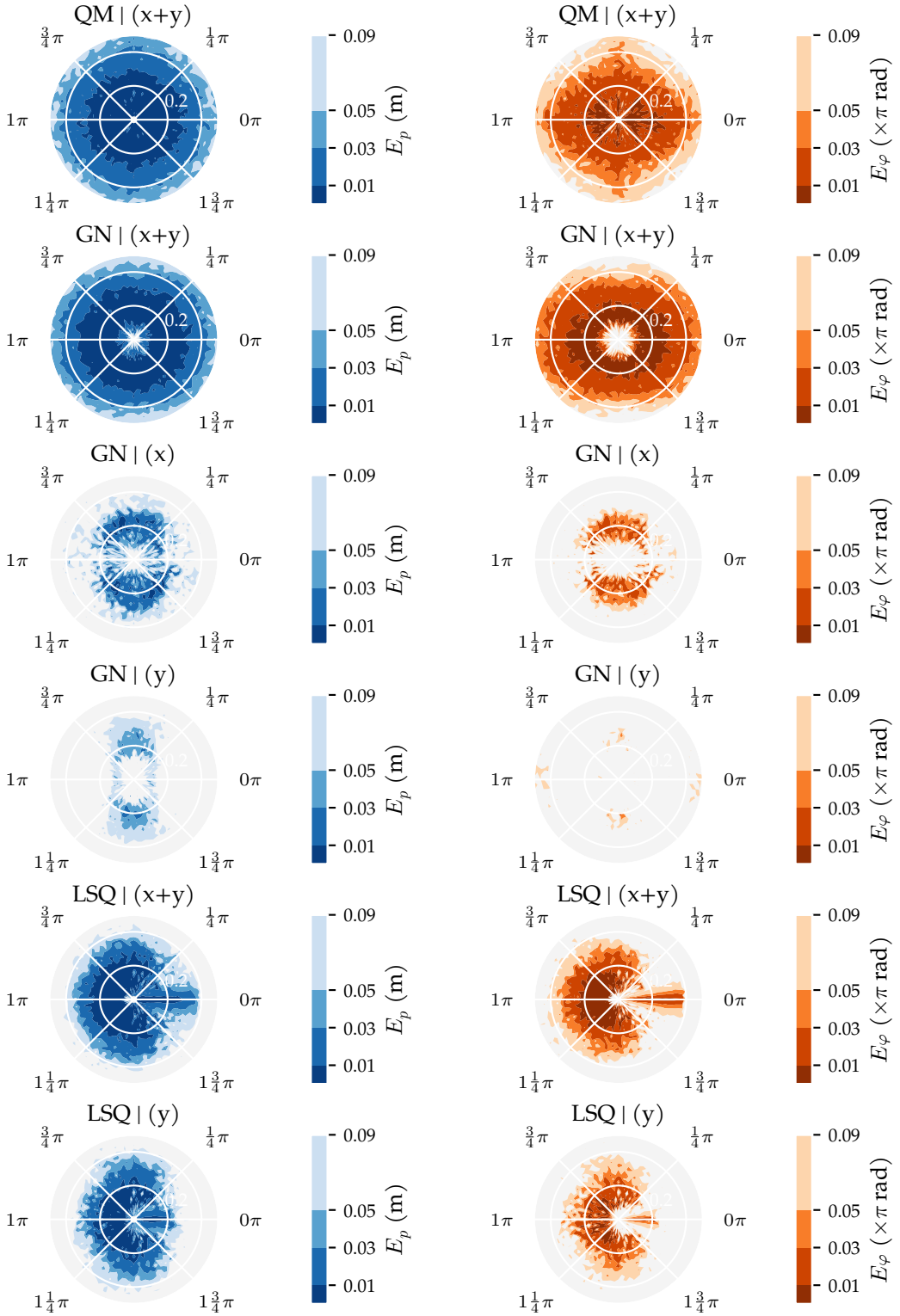
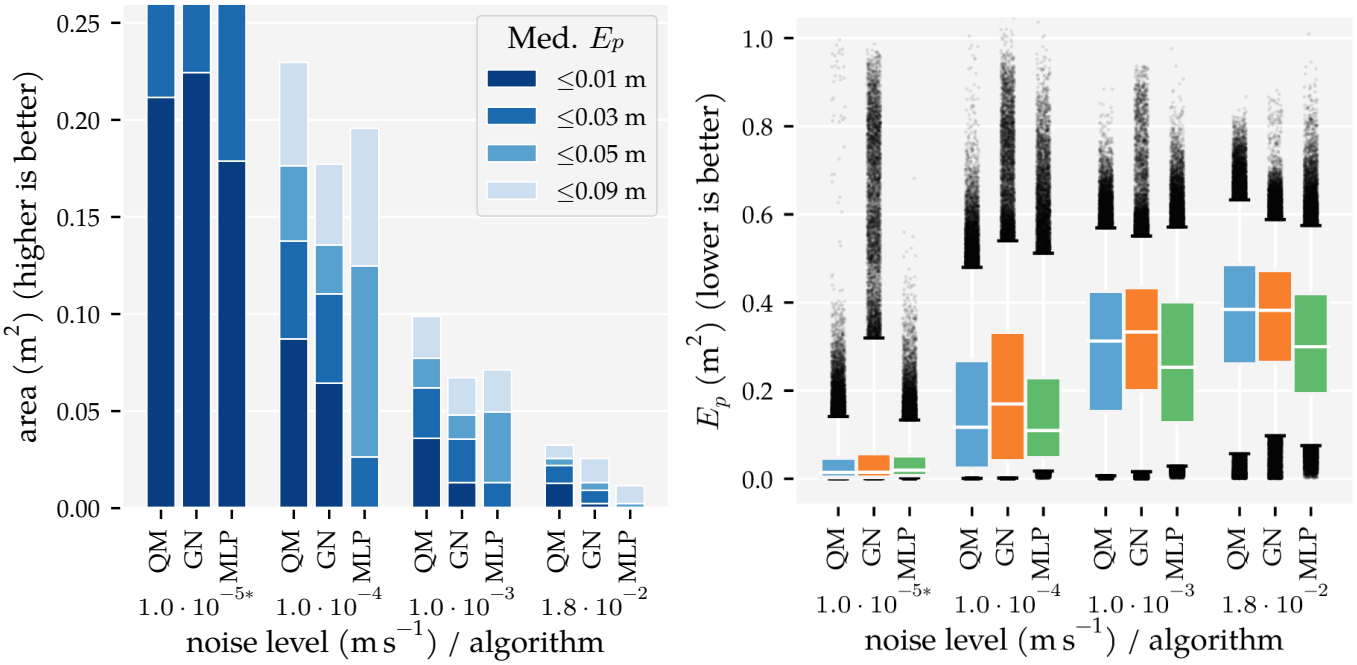


Figure 14. These figures indicate how the movement direction φ and distance d of a source state influence the median position error E_p (blue) (Equation (4)) and median movement direction error E_φ (orange) (Equation (5)). The quadrature method (QM), Gauss–Newton (GN), and least square curve fit (LSQ) predictors were used with three sensor configurations: (x + y), (x), (y) at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level. The median errors were computed in cells of 0.01π rad $\times$ 1 cm.



(**a**) (**b**)

Figure 15. An overview of the position error E_p (Equation (4)) of QM, GN, and MLP using simulated sensors with higher velocity equivalent noise levels. (**a**) Total areas with a median position error E_p below 1 cm, 3 cm, 5 cm, and 9 cm. (**b**) Boxplots of the position error distributions, whiskers indicate the 5th and 95th percentiles of the distributions. Predictions with errors outside these percentiles are shown individually. The values for $\sigma = 1.0 \times 10^{-5}$ m s^{-1} are based on the $D_s = 0.01$ condition in Analysis Method 1. The (x + y) sensor configuration was used. The MLP was re-trained for each noise level. Both the MLP and GN used the optimal hyperparameter values from the $D_s = 0.01$ condition of Analysis Method 1.

4. Discussion

In Analysis Method 1, we confirmed that (1) the total area in which the model-based algorithms produce accurate predictions does not depend on the amount of training and optimisation data, and (2) the template-based algorithms' and neural networks' areas with accurate predictions increase with the amount of training and optimisation data (Figures 5 and 6). The MLP, in particular, benefits from large amounts of training data. However, only with our largest training and optimisation set (90,435 states) are the MLP and KNN able to approach QM's and GN's performance. Even in that case, their movement direction predictions are no match for those of QM and GN. Whether the MLP or KNN provides the better performance depends on the amount of training and optimisation data. The MLP performed better with our smallest and largest sets (Section 3.1). In the other cases, KNN performed better.

In Analysis Method 2, we demonstrated the benefit of 2D sensitive sensors compared to 1D sensitive sensors. Improved position and movement direction estimation of ELMs using 2D sensors compared to using only v_x was previously shown by Wolf and van Netten [45]. In the present study, we showed that other localisation algorithms also benefit from using both velocity components. In addition, we demonstrated that the alternating sensor configuration (x | y) used by Yang et al. [12] and Nguyen et al. [13] is not an adequate substitute for 2D sensitive sensors at the used spatial resolution. However, it does improve performance compared to measuring a single velocity component along the entire sensor array. We speculate that the performance difference between the alternating sensor configuration and 2D sensitive sensors diminishes as the sensor array's spatial resolution increases.

Finally, we showed that the newly introduced QM provided the best overall performance with 2D sensors, regardless of the data set size. Its areas with a median position error below 1 cm and median movement direction error below 0.01π rad were as large

as that of GN (Figures 5 and 6). However, the tails of the error distributions of QM were shorter than those of GN (Figures 7 and 8). In other words, the less-accurate predictions of QM were better than those of GN. With 1D sensors, LSQ performed the best, except for source states with an orientation close to $\varphi = 0$ rad.

It should be noted that the model-based algorithms depend on an accurate forward model. Fitting a potential flow model only works when it accurately describes the actual velocity measurements. When more complicated hydrodynamic phenomena are present in the measurements, a forward model may become quite complex. In that case, the MLP and KNN may be good alternatives. However, these algorithms require a lot of training and optimisation data to reach a similar performance as the model-based algorithms.

We continue the discussion with remarks specific to the used algorithms in the following subsections.

4.1. The Gauss–Newton (GN) Algorithm

GN's performance in this work was mostly in line with the results presented by Abdulsadda and Tan [8]. They showed that GN has a superior localisation performance compared to LCMV beamforming and template matching. In addition, they highlighted that the convergence behaviour of GN heavily depends on the initial estimate. In their simulation, the largest region of convergence had a radius of 1.5 cm, depending on the source's position [8]. In the current study, we did not estimate the region of convergence. Instead, we showed that—based on the median position and movement direction errors in the simulated domain—GN can be used in a larger area, especially when 2D sensitive sensors are used. However, there is no guarantee of convergence in that larger area, as is evident from the large tail in the error distributions (Figures 11 and 12). To improve the convergence rate, one could potentially use the estimated position and movement direction of another prediction method as the initial estimate of GN.

A difference in our results compared to Abdulsadda and Tan [8], is the effect of the source state's orientation on the prediction accuracy when only v_x or only v_y are used (Figure 14). We found that parallel source states are localised less accurately than other orientations when only v_x is used. Abdulsadda and Tan [8] did not report such an effect. In their results, of the 19 states reported, four had a roughly parallel orientation. Due to those state's positions it is difficult to determine how much their movement direction influenced the prediction accuracy. So, this effect may not have been observable with their experimental setup.

Another difference compared to Abdulsadda and Tan [8] were our step tolerance and maximum number of iteration hyperparameters. Both parameters determine the accuracy and computational cost of the algorithm. For our experiments, a prediction within 1 mm and 0.001π rad was sufficient. The maximum number of iterations was reduced to 100 compared to the 2500 used by Abdulsadda and Tan [8]. We found that the limit of 100 iterations did not prevent a prediction from reaching the outer edges of the simulated domain. Depending on the use case, these parameters can be tuned to provide the best performance.

Finally, unlike Abdulsadda and Tan [8], we applied a bounds check on each iteration's estimated position and movement direction, to keep the estimates within the simulated domain (Figure 2). Since we did not compare GN's performance with and without the bounds check, we cannot draw conclusions about its effect. However, we expect that the bounds check has two advantageous effects. Firstly, it may cause non-converging predictions to terminate earlier because—depending on their trajectory—the difference between estimates in subsequent iterations may become smaller than $\epsilon = 1.0 \times 10^{-3}$. Secondly, it may cause some otherwise non-converging predictions to converge. Instead of continuing on their trajectory, these estimates move along the simulated domains boundaries and find a path towards the solution.

4.2. The Newton–Raphson (NR) Algorithm

Unlike the results of Abdulsadda and Tan [8], NR did not perform similar to GN in our analyses. Computing NR's predictions took more than 40 times as long as for GN (Table 3). The increased run-time cost of NR in our implementation is probably due to the numerical estimation of the Hessian. The difference in localisation performance may also be due this numerical estimation. The Newton method (Equation (15)) requires that the Hessian is positive definite; otherwise, it may move the prediction towards a saddle point instead of a minimum [41] (p. 279,304). Manual inspection of the Hessian found negative eigenvalues in all iterations for the inspected source states, regardless of the prediction accuracy. A preliminary hyperparameter validation showed that the regularisation strategy suggested by Goodfellow et al. [41] (p. 304) to solve this issue did not improve the performance of NR. From their publication, it is not clear whether Abdulsadda and Tan [8] computed the Hessian analytically. Alternatively, this difference in performance may be explained if their initial estimates were within NR's region of convergence.

In our implementation of NR, we applied the same bounds check as for GN. Since we did not compare the performance of NR with and without the bounds check, we cannot draw conclusions about its effect. However, we expect the same improvements as with GN.

4.3. The Multi-Layer Perceptron (MLP) and Extreme Learning Machine (ELM)

We showed that an MLP performs quite a lot better than an ELM when a large amount of training data is available (90,435 source states) (Figures 9 and 10). The main differences between these neural networks are the training procedure and the higher capacity of the MLP. The MLP was trained to minimise the mean absolute error (MAE), whereas the ELM minimises the mean squared error (MSE). The difference in capacity was especially large when the largest training and optimisation set was used with 2D sensitive sensors. In that case, the MLP had roughly ten times more weights than the ELM and roughly sixty times more trainable weights. When the smaller training and optimisation sets were used, the MLP performed more similar to the ELM and used only a single layer.

The comparison of QM, GN and the MLP on simulated sensors with lower SNRs showed the robustness of the MLP against noise (Figure 15). This finding is in line with Boulogne et al. [9], who showed that the MLP was more robust to noise than ELMs and echo state networks (ESNs).

The performance of the MLP may be improved further by also training bias-weights. Bias-weights serve as an activation-offset for each node in a layer, providing additional flexibility in the activation function. Bias-weights increase the capacity of the network and enable nodes to output non-zero values when their input is zero. We expect that the network's prediction of, in particular, the source distance benefits from bias-weights because the distances are not centred around the same value as the input.

4.4. The Quadrature Method (QM) Algorithm

In this study, we introduced the QM dipole localisation algorithm. The algorithm is designed for 2D sensitive sensors and provides state-of-the-art performance. Compared to GN, the algorithm produces more accurate predictions close to the sensor array (Figure 13) and has less skewed error distributions (Figures 7 and 8). These results indicate that the iterative refinement procedure of QM converges better than the non-linear optimisation of GN. In addition, the effective area of QM was larger than that of GN when using simulated sensors with lower SNRs (Figure 15).

Aside from its performance, QM has several attractive attributes. For instance, the orientation estimation procedure is very quick—with a computational complexity in the order of sensors—and only depends on the measured velocity and an estimate of the source's position. As a result, any algorithm that estimates a source's position can use QM's orientation estimation. In addition, QM is able to compute an initial position estimate directly from the measured velocity (also with a computational complexity in the order of sensors). Consequently, QM does not require a hyperparameter specifying an arbitrary initial esti-

mate. It should be noted that this initial estimate is accurate only in a limited area, as it depends on anchor-points on ψ_{quad} that have to fall within the sensor array (Appendix D). Finally, the run-time of QM is easily tuned using two simple hyperparameters: the number of refinement iterations and the number of iterations used to fit the potential flow model. The hyperparameters values used in this study resulted in an average prediction time that was roughly two and a half times slower than GN.

Several interesting research questions about QM remain unanswered. For instance, we did not analyse the QM's performance using only its initial estimate. Especially for higher resolution sensor arrays, these estimates may perform quite well compared to the other algorithms. In addition, other algorithms may benefit from using QM's initial estimate. For example, GN's performance close to the sensor array may improve when QM's initial estimate is used as the starting point for fitting the potential flow model. Other two-stage combinations of algorithms may also be interesting to develop.

4.5. Future Research Directions and Possible Applications

In the present research, ten dipole localisation algorithms were compared using a stationary ALL in a 2D environment. Applications of ALLs typically operate in more complex environments that are embedded in 3D and may include self-motion. The shape and movement of a sensing platform itself would have a significant impact on the flow fields which were omitted in this analysis (see, for instance, Windsor et al. [46] for an analysis of flow fields around gliding blind cave fish). Localising sources in 3D requires different sensor configurations. Yang et al. [12] used a sensor array consisting of two orthogonal lines on a cylinder to demonstrate LCMV beamforming's 3D localisation performance. Wolf et al. [15] used two parallel ALLs to localise multiple simultaneous sources in 3D. Analysing QM's performance on 3D localisation tasks would be an interesting future research project.

Other interesting future research directions include using more realistic fluid flow simulations. For instance, Lin et al. [47] used a Navier–Stokes equation solver to predict the ratio of a source's distance and size based on the measured velocity amplitude range. Additionally, it remains unclear how well the algorithms' performances transfer to localising differently shaped or self-propelled objects. Finally, the algorithms could be compared on their performance locating moving objects instead of stationary dipoles.

5. Conclusions

In the present study, we compared a wide range of algorithms for determining a dipole's position and orientation in the vicinity of a flow sensor array. To demonstrate the effect of the amount of available data to optimise or tune each algorithm, we sampled a bounded domain with four levels of granularity. To demonstrate the effects of sensitivity directions of the flow sensors, we extended the implementation of existing algorithms to support data from four different sensor configurations: sensitivity parallel to the array, sensitivity at a right angle to the array, sensors with alternating sensitivity directions, and finally an array of 2D-sensitive sensors. These effects on the algorithms were quantified by the area in which an algorithm can correctly determine a dipole's position and orientation relative to the array with predefined degrees of accuracy. For further comparisons, we also disclosed box plots to indicate the distribution of errors, as well as visual representations of these errors in the 2D spatial domain and source orientation domain.

We demonstrated the benefit of 2D-sensitive sensors compared to 1D sensitive sensors for a dipole localisation task. All considered algorithms benefited from using information from 2D-sensitive sensors, although the amount of improvement varies. The extension to 2D-sensitive sensors allowed the introduction of a novel dipole localisation algorithm coined the quadrature method (QM). This algorithm is designed to take advantage of geometric properties that result from 2D-sensitive flow measurements. We showed that QM provides state-of-the-art performance and produces more accurate predictions than the Gauss–Newton (GN) algorithm [8], especially for source positions close to the sensor array.

Finally, we analysed how dipole localisation algorithms' performance depends on the amount of training and optimisation data. We find that template-based algorithms and neural-networks require large amounts of training data to approach the performance of model-based algorithms that require only a small training and optimisation set in a simulation setting.

Since the simulation's assumptions are all based on the potential flow of a vibrating sphere, the resulting flow fields can be used for all dipole fields. Such fields are not restricted to those generated by submerged moving objects. Exactly the same dipole field equations are associated with acoustic, electric, and magnetic phenomena. Therefore, the comparisons made in the present work may also be of interest in other applications than hydrodynamic imaging.

Author Contributions: Conceptualisation, D.M.B., B.J.W. and S.M.v.N.; Data curation, D.M.B.; Formal analysis, D.M.B. and S.M.v.N.; Funding acquisition, S.M.v.N.; Investigation, D.M.B.; Methodology, D.M.B. and B.J.W.; Project administration, B.J.W. and S.M.v.N.; Resources, B.J.W.; Software, D.M.B.; Supervision, B.J.W. and S.M.v.N.; Validation, B.J.W. and S.M.v.N.; Visualisation, D.M.B.; Writing—original draft, D.M.B.; Writing—review & editing, B.J.W. and S.M.v.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partly supported by the Lakhsmi project (B.J.W., S.M.v.N.) that has received funding from (1) the European Union's Horizon 2020 research and innovation programme under grant agreement No 635568 and (2) the SeaClear project (B.J.W.) that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871295 and (3) the Flemish Government programme "Onderzoeksprogramma Artificiële Intelligentie (AI)" (D.M.B.).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data generated in this study, including the algorithms' predictions and the training and test source states, are available from Zenodo [48]. Our implementation of the algorithms in MATLAB R2018a [38] as used in this publication are also available from Zenodo [49].

Acknowledgments: We would like to thank the Center for Information Technology of the University of Groningen for providing access to the Peregrine high performance computing cluster.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ALL	artificial lateral line
AUV	autonomous underwater vehicle
CNN	convolutional neural network
CWT	continuous wavelet transform
DFT	discrete Fourier transform
ELM	extreme learning machine
ESN	echo state network
GN	Gauss–Newton
KNN	k-nearest neighbours
LCMV	linear constraint minimum variance
LSQ	least square curve fit
MAE	mean absolute error
MSE	mean squared error
MLP	multi-layer perceptron
NR	Newton–Raphson
OS-ELM	online sequential extreme learning machine

QM	quadrature method
RND	random
SLFN	single layer feed-forward network
SNR	signal to noise ratio

Appendix A. Final Hyperparameter Values

This appendix lists the final hyperparameter values as used in the analyses. Tables A1 and A2 provide the optimal hyperparameter values for Analysis Methods 1 and 2, respectively. Analysis Method 3 used the optimal values of Analysis Method 1's $D_s = 0.01$ condition. Section 2.4 explains the general approach used to find the hyperparameters' optimal values and Section 2.5 explains how each hyperparameter influences the dipole localisation algorithms.

Table A1. All hyperparameter values for the first analysis optimised using the training set. This analysis varied the minimum distance D_s between source states in the training and optimisation sets to show how the amount of training and optimisation data influences the dipole localisation algorithms' performance. The (x + y) sensor configuration at the $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level was used in this analysis. Table 1 describes the respective data sets in more detail.

KNN	D_s	k neighbours			
	0.09	3			
	0.05	3			
	0.03	3			
	0.01	5			
CWT	D_s	threshold t_{min}	threshold t_{max}	φ factor c_x	φ factor c_y
	0.09	0.36	1.0	1.0	0.30
	0.05	0.18	0.92	0.89	0.89
	0.03	0.21	0.74	0.98	0.35
	0.01	0.26	0.57	0.82	0.38
ELM	D_s	$\bar{n}$ nodes		Analytical $c_x = 0.6$ (higher value tunes	
	0.09	120		estimation for longer distances)	
	0.05	75		Analytical $c_y \approx 0.366$ (lower value tunes	
	0.03	1383		estimation for longer distances)	
	0.01	11,169			
MLP	D_s	learning rate ϵ	n layers	$\bar{n}$ nodes per layer	
	0.09	3.5×10^{-3}	1	990	
	0.05	3.4×10^{-3}	1	798	
	0.03	3.7×10^{-3}	1	1015	
	0.01	1.2×10^{-3}	4	993	
GN	D_s	initial distance d_0 (cm)			
	0.09	5.0			
	0.05	5.0			
	0.03	2.5			
	0.01	5.0			
NR	D_s	initial distance d_0 (cm)	norm limit l		
	0.09	2.5	0.14		
	0.05	2.9	0.10		
	0.03	2.5	0.10		
	0.01	2.5	0.10		

Table A2. All hyperparameters for the second analysis optimised using the training set. This analysis varied the sensor sensitivity axes to show how the velocity components contribute to the predictions while keeping D_s constant. The configurations were: (x + y) both components on all sensors, (x | y) alternating v_x and v_y for subsequent sensors, (x) only v_x on all sensors, (y) only v_y on all sensors. The $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level was used and with the $D_s = 0.01$ training and optimisation set.

KNN	sensor	k neighbours			
	(x + y)	5			
	(x | y)	5			
	(x)	6			
	(y)	6			
CWT	sensor	threshold t_{min}	threshold t_{max}	φ factor c_x	φ factor c_y
	(x + y)	0.23	0.60	0.71	0.33
	(x | y)	0.22	0.40	0.76	0.64
	(x)	0.21	0.53	0.61	0.40
	(y)	0.12	0.79	0.58	0.87
ELM	sensor	$\bar{n}$ nodes		Analytical $c_x = 0.6$ (higher value tunes	
	(x + y)	11,169		estimation for longer distances)	
	(x | y)	5165		Analytical $c_y \approx 0.366$ (lower value tunes	
	(x)	5579		estimation for longer distances)	
	(y)	5579			
MLP	sensor	learning rate ϵ	n layers	$\bar{n}$ nodes per layer	
	(x + y)	1.4×10^{-3}	4	1014	
	(x | y)	2.3×10^{-3}	4	982	
	(x)	2.0×10^{-3}	4	1024	
	(y)	2.0×10^{-3}	4	1016	
GN	sensor	initial distance d_0 (cm)			
	(x + y)	5.0			
	(x | y)	2.5			
	(x)	7.5			
	(y)	2.5			
NR	sensor	initial distance d_0 (cm)	norm limit l		
	(x + y)	2.5	0.10		
	(x | y)	7.8	0.13		
	(x)	9.3	0.10		
	(y)	2.5	0.10		

Appendix B. Potential Flow Wavelets

In this appendix, we derive the three wavelets used in Wolf and van Netten [14] from the potential flow formula [16] (Equation (1)):

$$\boldsymbol{v} = \frac{a^3}{2||\boldsymbol{r}||^3}\left(-\boldsymbol{w} + 3\boldsymbol{r}\frac{(\boldsymbol{w}\times\boldsymbol{r})}{||\boldsymbol{r}||^2}\right), \tag{A1}$$

where a is the radius of the sphere, $\boldsymbol{w} = \langle w_x, w_y \rangle$ are the velocity components of the moving sphere in 2D, and $\boldsymbol{r} = \boldsymbol{s} - \boldsymbol{p}$ is the position of the sensor $\boldsymbol{s} = \langle x, y \rangle$ as seen from the source $\boldsymbol{p} = \langle b, d \rangle$.

Appendix B.1. The Even Wavelet

The even wavelet ψ_e occurs in v_x when a source moves parallel to the sensor array [5,14]. Let $\boldsymbol{w} = \langle w_x, 0 \rangle$, then the parallel velocity component is given by:

$$v_x = \frac{a^3}{2||\boldsymbol{r}||^3}\left(-w_x + \frac{3w_x(x-b)^2}{||\boldsymbol{r}||^2}\right), \tag{A2}$$

$$v_x = \frac{a^3 w_x}{2||\boldsymbol{r}||^3}\left(-1 + \frac{3(x-b)^2}{||\boldsymbol{r}||^2}\right), \tag{A3}$$

$$v_x = \frac{a^3 w_x}{2||\boldsymbol{r}||^3}\left(\frac{3(x-b)^2 - ||\boldsymbol{r}||^2}{||\boldsymbol{r}||^2}\right). \tag{A4}$$

To find the wavelet, $||\boldsymbol{r}||$ is rewritten as:

$$||\boldsymbol{r}|| = \sqrt{(x-b)^2 + (y-d)^2} = \sqrt{(y-d)^2}\sqrt{\rho^2+1}, \tag{A5}$$

with

$$\rho = \frac{x-b}{y-d}. \tag{A6}$$

Substituting this in the formula for v_x results in:

$$v_x = \frac{a^3 w_x}{2\left(\sqrt{(y-d)^2}\sqrt{\rho^2+1}\right)^3}\left(\frac{3(x-b)^2 - (y-d)^2(\rho^2+1)}{\left(\sqrt{(y-d)^2}\sqrt{\rho^2+1}\right)^2}\right), \tag{A7}$$

$$v_x = \frac{a^3 w_x}{2\left(\sqrt{(y-d)^2}\right)^3}\left(\frac{3\frac{(x-b)^2}{(y-d)^2} - \frac{(y-d)^2(\rho^2+1)}{(y-d)^2}}{\left(\sqrt{\rho^2+1}\right)^5}\right), \tag{A8}$$

$$v_x = \frac{a^3 w_x}{2|y-d|^3}\left(\frac{3\rho^2 - \rho^2 - 1}{(\rho^2+1)^{(5/2)}}\right). \tag{A9}$$

So, finally:

$$\psi_e = \frac{2\rho^2 - 1}{(\rho^2+1)^{(5/2)}}. \tag{A10}$$

Appendix B.2. The Odd Wavelet

The odd wavelet ψ_o occurs in v_x when a source moves perpendicular to the sensor array [5,14]. Let $\boldsymbol{w} = \langle 0, w_y \rangle$, then the parallel velocity component is given by:

$$v_x = \frac{a^3}{2||\boldsymbol{r}||^3}\left(-0 + \frac{3w_y(x-b)(y-d)}{||\boldsymbol{r}||^2}\right), \tag{A11}$$

$$v_x = \frac{a^3 w_y}{2||\boldsymbol{r}||^3}\left(\frac{3(x-b)(y-d)}{||\boldsymbol{r}||^2}\right). \tag{A12}$$

Rewriting and substituting $||\boldsymbol{r}||$ in the same way as in Appendix B.1, yields:

$$v_x = \frac{a^3 w_y}{2\left(\sqrt{(y-d)^2}\sqrt{\rho^2+1}\right)^3}\left(\frac{3(x-b)(y-d)}{\left(\sqrt{(y-d)^2}\sqrt{\rho^2+1}\right)^2}\right), \tag{A13}$$

$$v_x = \frac{a^3 w_y}{2\left(\sqrt{(y-d)^2}\right)^3}\left(\frac{3\frac{(x-b)(y-d)}{(y-d)^2}}{\left(\sqrt{\rho^2+1}\right)^5}\right), \tag{A14}$$

$$v_x = \frac{a^3 w_x}{2|y-d|^3}\left(\frac{3\rho}{(\rho^2+1)^{(5/2)}}\right). \tag{A15}$$

So, finally:

$$\psi_o = \frac{3\rho}{(\rho^2+1)^{(5/2)}}. \tag{A16}$$

Appendix B.3. The Navelet

The navelet (Not-A-waVELET) ψ_n occurs in v_y when a source moves perpendicular to the sensor array [14]. Let $\boldsymbol{w} = \langle 0, w_y \rangle$, then the perpendicular velocity component is given by:

$$v_y = \frac{a^3}{2||\boldsymbol{r}||^3}\left(-w_y + \frac{3w_y(y-d)^2}{||\boldsymbol{r}||^2}\right), \tag{A17}$$

$$v_y = \frac{a^3 w_y}{2||\boldsymbol{r}||^3}\left(-1 + \frac{3(y-d)^2}{||\boldsymbol{r}||^2}\right), \tag{A18}$$

$$v_y = \frac{a^3 w_y}{2||\boldsymbol{r}||^3}\left(\frac{3(y-d)^2 - ||\boldsymbol{r}||^2}{||\boldsymbol{r}||^2}\right). \tag{A19}$$

Rewriting and substituting $||\boldsymbol{r}||$ in the same way as in Appendix B.1, results in:

$$v_y = \frac{a^3 w_y}{2\left(\sqrt{(y-d)^2}\sqrt{\rho^2+1}\right)^3}\left(\frac{3(y-d)^2 - (y-d)^2(\rho^2+1)}{\left(\sqrt{(y-d)^2}\sqrt{\rho^2+1}\right)^2}\right), \tag{A20}$$

$$v_y = \frac{a^3 w_y}{2\left(\sqrt{(y-d)^2}\right)^3}\left(\frac{3\frac{(y-d)^2}{(y-d)^2} - \frac{(y-d)^2(\rho^2+1)}{(y-d)^2}}{\left(\sqrt{\rho^2+1}\right)^5}\right), \tag{A21}$$

$$v_y = \frac{a^3 w_y}{2|y-d|^3}\left(\frac{3-\rho^2-1}{(\rho^2+1)^{(5/2)}}\right). \tag{A22}$$

So, finally:

$$\psi_n = \frac{2-\rho^2}{(\rho^2+1)^{(5/2)}}. \tag{A23}$$

Appendix C. Movement Direction Estimation with the Continuous Wavelet Transform (CWT)

In this appendix, we show how the CWT can be used to estimate the direction of movement of a dipole source. In Section 2.5.4, we showed that potential flow (Equation (1)) can be expressed in terms of wavelets:

$$\begin{aligned} v_x &= \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(\psi_e \cos(\varphi) + \psi_o \sin(\varphi)), \\ v_y &= \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(\psi_o \cos(\varphi) + \psi_n \sin(\varphi)), \end{aligned} \tag{A24}$$

with

$$\psi_e = \frac{2\rho^2 - 1}{(\rho^2 + 1)^{(5/2)}}, \tag{A25}$$

$$\psi_o = \frac{3\rho}{(\rho^2 + 1)^{(5/2)}}, \tag{A26}$$

$$\psi_n = \frac{2 - \rho^2}{(\rho^2 + 1)^{(5/2)}}, \tag{A27}$$

$$\rho = \frac{r_x}{r_y} = \frac{x - b}{y - d}, \tag{A28}$$

where $a = 1$ cm is the radius of the source, $\boldsymbol{w} = \langle w_x, w_y \rangle$ is the movement velocity of the source, $\boldsymbol{r} = \boldsymbol{s} - \boldsymbol{p}$ is the relative position of the sensor $\boldsymbol{s} = \langle x, y \rangle$ as seen from the source $\boldsymbol{p} = \langle b, d \rangle$, and φ is the azimuth angle of the motion. In general, the CWT coefficients $Wf(u, s)$ are computed using [50]:

$$Wf(u, s) = \int_{-\infty}^{+\infty} f(t)\psi_{u,s}(t)dt, \tag{A29}$$

where $f(t)$ is the signal that is analysed and $\psi_{u,s}(t)$ is a scaled s and translated u version of a mother wavelet ψ [50]:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t - u}{s}\right). \tag{A30}$$

In the case of dipole localisation, four sets of CWT coefficients are computed:

$$Wv_x^e(\boldsymbol{p}) = \frac{1}{\sqrt{|y - d|}} \int_{-\infty}^{+\infty} v_x(x)\psi_e(\boldsymbol{s} - \boldsymbol{p})dx, \tag{A31}$$

$$Wv_x^o(\boldsymbol{p}) = \frac{1}{\sqrt{|y - d|}} \int_{-\infty}^{+\infty} v_x(x)\psi_o(\boldsymbol{s} - \boldsymbol{p})dx, \tag{A32}$$

$$Wv_y^n(\boldsymbol{p}) = \frac{1}{\sqrt{|y - d|}} \int_{-\infty}^{+\infty} v_y(x)\psi_n(\boldsymbol{s} - \boldsymbol{p})dx, \tag{A33}$$

$$Wv_y^o(\boldsymbol{p}) = \frac{1}{\sqrt{|y - d|}} \int_{-\infty}^{+\infty} v_y(x)\psi_o(\boldsymbol{s} - \boldsymbol{p})dx, \tag{A34}$$

where $v_x(x)$ and $v_y(x)$ are the velocity components measured at a sensor $\boldsymbol{s} = \langle x, y \rangle$ for a source at $\boldsymbol{p} = \langle b, d \rangle$. Since the sensor array has a finite length, the information used by the CWT is in practice limited to sources that have most part of their significant nonzero excitation profile projected on the array. An important property of the wavelets is that they do not form a global orthonormal or orthogonal set of base functions. This renders the method less vulnerable to possible noise contributions on the measurements of the sensor array. On the other hand, different scales and shifts of the mother wavelets are mixed, so that especially in the case of closely spaced multiple sources, crosstalk over the reconstructed wavelets will occur. Consequently, the estimated spatial map formed by the maxima of the reconstruction functions may be distorted in such cases. In addition, the contributions of the individual wavelets to the measured velocity are only separated at the source's position because the CWT of ψ_e with ψ_o and ψ_n with ψ_o produces zero coefficients only at that position. Consequently, the movement direction of a dipole source depends on an accurate position estimate. In the following subsections, the movement direction estimation is discussed for v_x and v_y, respectively.

Appendix C.1. Movement Direction Estimation with the Parallel Velocity Component

The formulas for the CWT coefficients using the measurements of v_x (Equations (A31) and (A32)) can be rewritten to show the influence of both wavelets on the coefficients, using Equation (A24):

$$Wv_x^e(\boldsymbol{p}) = \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(W_{xee}(\boldsymbol{p})\cos(\varphi) + W_{xeo}(\boldsymbol{p})\sin(\varphi)), \tag{A35}$$

$$Wv_x^o(\boldsymbol{p}) = \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(W_{xoe}(\boldsymbol{p})\cos(\varphi) + W_{xoo}(\boldsymbol{p})\sin(\varphi)), \tag{A36}$$

with

$$W_{xee}(\boldsymbol{p}) = \frac{1}{\sqrt{|y-d|}}\int_{-\infty}^{+\infty}\psi_e(\boldsymbol{s}-\bar{\boldsymbol{p}})\;\psi_e(\boldsymbol{s}-\boldsymbol{p})dx, \tag{A37}$$

$$W_{xeo}(\boldsymbol{p}) = \frac{1}{\sqrt{|y-d|}}\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})\;\psi_e(\boldsymbol{s}-\boldsymbol{p})dx, \tag{A38}$$

$$W_{xoe}(\boldsymbol{p}) = \frac{1}{\sqrt{|y-d|}}\int_{-\infty}^{+\infty}\psi_e(\boldsymbol{s}-\bar{\boldsymbol{p}})\;\psi_o(\boldsymbol{s}-\boldsymbol{p})dx, \tag{A39}$$

$$W_{xoo}(\boldsymbol{p}) = \frac{1}{\sqrt{|y-d|}}\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})\;\psi_o(\boldsymbol{s}-\boldsymbol{p})dx, \tag{A40}$$

where $a = 1$ cm is the radius of the source sphere, $\boldsymbol{w}$ is the velocity vector of the source, φ is the azimuth angle of $\boldsymbol{w}$ with respect to the sensor array, and $\boldsymbol{s} = \langle x, y\rangle$ is the location of the sensor. In these equations, we differentiate between the position of the source which generated the measured velocity ($\bar{\boldsymbol{p}}$) and the position of a source for which the CWT is evaluated ($\boldsymbol{p}$).

Suppose that the position of the dipole is known $\boldsymbol{p} = \bar{\boldsymbol{p}}$. Then, $W_{xeo}(\boldsymbol{p})$ and $W_{xoe}(\boldsymbol{p})$ are both zero, and Equations (A35) and (A36) reduce to:

$$Wv_x^e(\boldsymbol{p}) = \frac{a^3||\boldsymbol{w}||}{2|y-d|^{(7/2)}}\left(\cos(\varphi)\int_{-\infty}^{+\infty}\psi_e(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx\right), \tag{A41}$$

$$Wv_x^o(\boldsymbol{p}) = \frac{a^3||\boldsymbol{w}||}{2|y-d|^{(7/2)}}\left(\sin(\varphi)\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx\right). \tag{A42}$$

So, $Wv_x^e(\boldsymbol{p})$ and $Wv_x^o(\boldsymbol{p})$ have to be normalised to estimate the direction of movement:

$$\varphi = -\operatorname{atan}\left(c_x\frac{\sin\varphi\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx}{\cos\varphi\int_{-\infty}^{+\infty}\psi_e(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx}\right), \tag{A43}$$

with:

$$c_x = \frac{\int_{-\infty}^{+\infty}\psi_e(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx}{\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx} = \frac{\int_{-\infty}^{+\infty}\psi_e(\rho)^2d\rho}{\int_{-\infty}^{+\infty}\psi_o(\rho)^2d\rho} = \frac{\frac{27}{128}\pi}{\frac{45}{128}\pi} = \frac{3}{5}. \tag{A44}$$

To conclude, in order to estimate the movement direction, a scaling as to be applied to the CWT coefficients at the estimated source position ($\hat{\boldsymbol{p}}$):

$$\hat{\varphi}_x = -\operatorname{atan} c_x\frac{Wv_x^o(\hat{\boldsymbol{p}})}{Wv_y^e(\hat{\boldsymbol{p}})}. \tag{A45}$$

Appendix C.2. Movement Direction Estimation with the Perpendicular Velocity Component

Using the same steps as for v_x, we find:

$$Wv_y^o(\boldsymbol{p}) = \frac{a^3||\boldsymbol{w}||}{2|y-d|^{(7/2)}}\left(\cos(\varphi)\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx\right), \tag{A46}$$

$$Wv_y^n(\boldsymbol{p}) = \frac{a^3||\boldsymbol{w}||}{2|y-d|^{(7/2)}}\left(\sin(\varphi)\int_{-\infty}^{+\infty}\psi_n(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx\right). \tag{A47}$$

So, $Wv_y^n(\boldsymbol{p})$ and $Wv_y^o(\boldsymbol{p})$ have to be normalised to estimate the direction of movement:

$$\varphi = -\operatorname{atan}\left(c_y\frac{\sin\varphi\int_{-\infty}^{+\infty}\psi_n(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx}{\cos\varphi\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx}\right), \tag{A48}$$

with:

$$c_y = \frac{\int_{-\infty}^{+\infty}\psi_o(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx}{\int_{-\infty}^{+\infty}\psi_n(\boldsymbol{s}-\bar{\boldsymbol{p}})^2dx} = \frac{\int_{-\infty}^{+\infty}\psi_o(\rho)^2d\rho}{\int_{-\infty}^{+\infty}\psi_n(\rho)^2d\rho} = \frac{\frac{45}{128}\pi}{\frac{123}{128}\pi} = \frac{15}{41}. \tag{A49}$$

To conclude, in order to estimate the movement direction, a scaling as to be applied to the CWT coefficients at the estimated source position ($\hat{\boldsymbol{p}}$):

$$\hat{\varphi}_y = -\operatorname{atan} c_y\frac{Wv_y^n(\hat{\boldsymbol{p}})}{Wv_y^o(\hat{\boldsymbol{p}})}. \tag{A50}$$

Appendix D. The Quadrature Method (QM)

Here, we provide some more details about the quadrature method (QM). Some previous attempts have been made to analytically determine a 2D position $\boldsymbol{p} = \langle b, d\rangle$ and movement direction φ of a dipole source from its generated fluid flows [5,25]. This task is called the inverse problem of hydrodynamic source localisation [15]. An intrinsic difficulty of the inverse problem using the dipole field is that both v_x and v_y consist of a directional-dependent combination of two of the three basis wavelets (see Section 2.5.4):

$$\begin{aligned} v_x &= \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(\psi_e\cos(\varphi)+\psi_o\sin(\varphi)),\\ v_y &= \frac{a^3||\boldsymbol{w}||}{2|y-d|^3}(\psi_o\cos(\varphi)+\psi_n\sin(\varphi)), \end{aligned} \tag{A51}$$

with

$$\psi_e = \frac{2\rho^2-1}{(\rho^2+1)^{(5/2)}}, \tag{A52}$$

$$\psi_o = \frac{3\rho}{(\rho^2+1)^{(5/2)}}, \tag{A53}$$

$$\psi_n = \frac{2-\rho^2}{(\rho^2+1)^{(5/2)}}, \tag{A54}$$

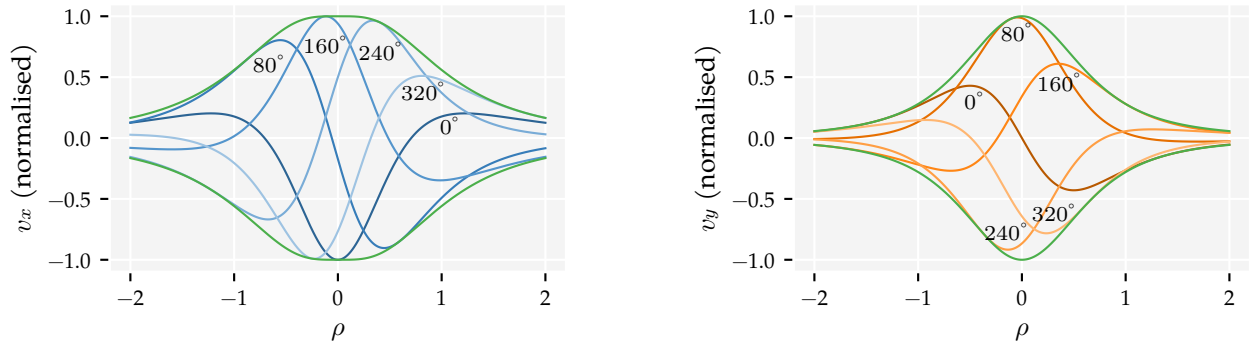
$$\rho = \frac{r_x}{r_y} = \frac{x-b}{y-d}, \tag{A55}$$

where $a = 1$ cm is the radius of the source, $\boldsymbol{w} = \langle w_x, w_y\rangle$ is the movement velocity of the source, $\boldsymbol{r} = \boldsymbol{s} - \boldsymbol{p}$ is the relative position of the sensor $\boldsymbol{s} = \langle x, y\rangle$ from the perspective of the source $\boldsymbol{p} = \langle b, d\rangle$, and φ is the azimuth angle of the motion.

The contribution of the basis functions depends on this φ. A series of velocity profiles—simultaneous measurements of flow velocity by a series of sensors arranged in a linear array—from a source at the same instantaneous position ($b = 0$, $d = 1$, so $\rho = -x$) are shown in Figure A1 for five different source directions ($\varphi = 0°$, 80°, 160°, 240° and 320°). A continuous φ-sequence of such profiles, presented in a time-lapse series, can be interpreted as a travelling wave moving within the envelopes indicated in green in Figure A1:

$$\psi_{x,env}(\rho) = \sqrt{\psi_e^2(\rho) + \psi_o^2(\rho)},$$
$$\psi_{y,env}(\rho) = \sqrt{\psi_o^2(\rho) + \psi_n^2(\rho)}. \tag{A56}$$

This travelling wave may be thought of as a 2D projection of a single 3D line structure, which rotates along the x-axis in synchrony with the angle φ (see Figure 4b). Different snapshots of such a projected travelling wave correspond to a velocity profile produced by a source at the same position but moving in different directions. Obviously, the green envelope is independent from the angle φ and is solely and completely determined by the source's position $\boldsymbol{p} = \langle b, d \rangle$.



(**a**) (**b**)

Figure A1. Velocity profiles of a continuous sensor array for five source movement directions φ: (**a**) v_x and (**b**) v_y. The envelopes of the velocity profiles $\psi_{x,env}(\rho)$ and $\psi_{y,env}(\rho)$ are shown in green.

When it could be distilled from the measurements, these envelopes' shape would indicate the source's x-position b with its top and yield the source distance d through its width, according to the definitions given in Equations (A52)–(A55). A necessary final step then would be to recover the movement angle φ from a measured velocity profile.

Reconstructing such an angle-independent envelope directly from a (measured) velocity profile has so far been proven impractical because it is difficult to separate the contributions of the wavelets in a single velocity profile without prior knowledge about the angle φ. However, a practical approach is provided by measurement of the two orthogonal velocity components v_x and v_y directed in a single plane through the array. This allows for the reconstruction of a virtually orientation-independent curve somewhat equivalent to the envelopes discussed above. This newly proposed method uses both v_x and v_y. As can be seen in comparing Figure A1a,b, v_x and v_y share a useful property: where one velocity profile has a local maximum, the other is close to a zero-crossing and vice versa, as has been previously noted in Wolf and van Netten [45]. We can effectively utilise this property to construct a curve, similar to the envelope discussed above, and show that it is only slightly dependent on the direction angle φ. The method is somewhat analogous to constructing an overall amplitude from two time-signals, as f.i. a sine and a cosine, which are 90° out of phase by taking a 'quadrature' combination of the two velocity components.

Here, we will construct the quadrature curve by using a specific weighting of the two measured quadratic velocity components, which will later be argued to minimise its dependency on the angle φ:

$$\psi_{quad}(\rho, \varphi) = \sqrt{v_x^2(\rho, \varphi) + \frac{1}{2} v_y^2(\rho, \varphi)}, \tag{A57}$$

with (using $2 \sin\varphi \cos\varphi = \sin 2\varphi$):

$$v_x^2(\rho, \varphi) = \frac{(4\rho^4 - 13\rho^2 + 1)\cos(\varphi)^2 + (6\rho^3 - 3\rho)\sin(2\varphi) + 9\rho^2}{(1 + \rho^2)^5}, \tag{A58}$$

$$v_y^2(\rho,\varphi) = \frac{(6\rho - 3\rho^3)\sin(2\varphi) + \left(\frac{13}{2}\rho^2 - \frac{1}{2}\rho^4 - 2\right)\cos(2\,\varphi) + \frac{5}{2}\rho^2 + \frac{1}{2}\rho^4 + 2}{(1+\rho^2)^5}. \tag{A59}$$

Arranging the terms per angle dependence and then normalising with $1+\sin(\varphi)^2$ at $\rho = 0$ yields the 'normalised quadrature' $\psi_{quad,norm}(\rho,\varphi)$:

$$\psi_{quad,norm}(\rho,\varphi) = \sqrt{\Phi_{sym}(\rho,\varphi) + \Phi_{skew}(\rho,\varphi)}, \tag{A60}$$

with an even (symmetric) and odd (skewed) component:

$$\Phi_{sym}(\rho,\varphi) = \frac{\left(\frac{7}{4}\rho^4 - \frac{13}{4}\rho^2 - \frac{1}{2}\right)\cos(2\varphi) + \frac{9}{4}\rho^4 + \frac{15}{4}\rho^2 + \frac{3}{2}}{(1+\rho^2)^5(1+\sin(\varphi)^2)}, \tag{A61}$$

$$\Phi_{skew}(\rho,\varphi) = \frac{\frac{9}{2}\rho^3\sin(2\varphi)}{(1+\rho^2)^5(1+\sin(\varphi)^2)}. \tag{A62}$$

Figure A2 gives examples of $\psi_{quad,norm}$ for the same values of φ as shown in Figure A1. Note that Φ_{sym} is even with respect to ρ irrespective of the angle φ, while Φ_{skew} is odd only with a third-order term (ρ^3) rather than also with a first-order term. This is a result of the factor $1/2$ applied to v_y^2 in Equation (A57), which effectively cancels the first-order terms. This cancellation has the desirable property that Φ_{skew} is flat around $\rho = 0$, which leads to $\psi_{quad,norm}$ having a single maximum at $\rho = 0$, rather than two possible local maxima. Consequently, the maximum of $\psi_{quad,norm}$ acts as a prime 'anchor point', pointing exactly to the source's x-position (b).

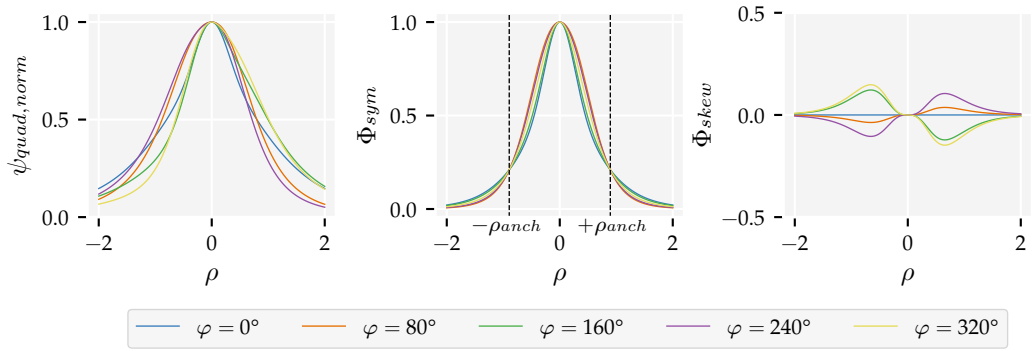


Figure A2. Quadrature profiles $\psi_{quad,norm}$ (left panel) of a continuous sensor array for five source directions φ (same as Figure A1). Furthermore, the two constituting functions $\Phi_{sym}(\rho,\varphi)$ (middle panel) and $\Phi_{skew}(\rho,\varphi)$ (right panel) are shown. These functions are, respectively, even and odd in ρ. At the secondary anchor points $\pm\rho_{anch}$ the function $\Phi_{sym}(\rho,\varphi)$ provides angle independent values which may be employed to determine the source distance d before the source direction of motion φ is known.

It should be noted from Figure A2 and Equations (A61) and (A62) that—apart from $\rho = 0$—both Φ_{sym} and Φ_{skew} still have a dependency on the source direction φ. However, as suggested by Figure A2, Φ_{sym} has two additional 'secondary anchor points' at $\pm\rho_{anch}$, which have no dependencies on angle φ and therefore provide convenient locations for a determination of an exact and invariable width measure of Φ_{sym}. This width measurement provides a good estimate of a 'representative width' of $\psi_{quad,norm}$. The addition of Φ_{skew} shifts the overall curve almost equally at these secondary anchor points because of its odd property. As a result, the relative distance between the secondary anchor points is hardly dependent on the angle φ and therefore almost linearly scales with the distance d of the source.

The exact locations of the secondary anchor points $\pm\rho_{anch}$ of Φ_{sym} can be calculated analytically. Because the anchor points should be independent of the angle φ, solving for the condition $\frac{\partial\Phi_{sym}}{\partial\varphi} = 0$ defines their location. For this, we may treat the factor $\left(1+\rho^2\right)^5$ in the denominator as a constant C:

$$0 = \frac{\partial\Phi_{sym}}{\partial\varphi} = \frac{-3\rho^2\left(5\rho^2-4\right)\sin(2\varphi)}{2C\left(\cos(\varphi)^2-2\right)^2}. \tag{A63}$$

The right-hand side of Equation (A63) has three solutions, corresponding to all three anchor points:

$$\rho = 0 \text{ and } \rho = \pm\frac{2}{\sqrt{5}} \approx \pm 0.894. \tag{A64}$$

The width between $\pm\rho_{anch}$ equals $4/\sqrt{5} \approx 1.789$. Determination of the resulting function value at the secondary anchor points $\Phi_{sym}(\pm\rho_{anch}, \varphi)$ leads via substitution to the angle-independent value:

$$\begin{aligned}\Phi_{sym}(\pm\rho_{anch}, \varphi) &= \frac{\left(\frac{7}{4}\rho_{anch}^4 - \frac{13}{4}\rho_{anch}^2 - \frac{1}{2}\right)\cos(2\varphi) + \frac{9}{4}\rho_{anch}^4 + \frac{15}{4}\rho_{anch}^2 + \frac{3}{2}}{\left(1+\rho_{anch}^2\right)^5\left(1+\sin(\varphi)^2\right)} \\ &\approx 0.210.\end{aligned} \tag{A65}$$

The distance between the two values of ρ where $\psi_{quad,norm}$ reaches $\sqrt{(0.210)}$ has been numerically determined for a range of angles φ, and indeed shows a slight variation around 1.78, as shown in Figure A3. This confirms the almost φ-independent way for determining the width and therefore the distance (d) of the source to the array.

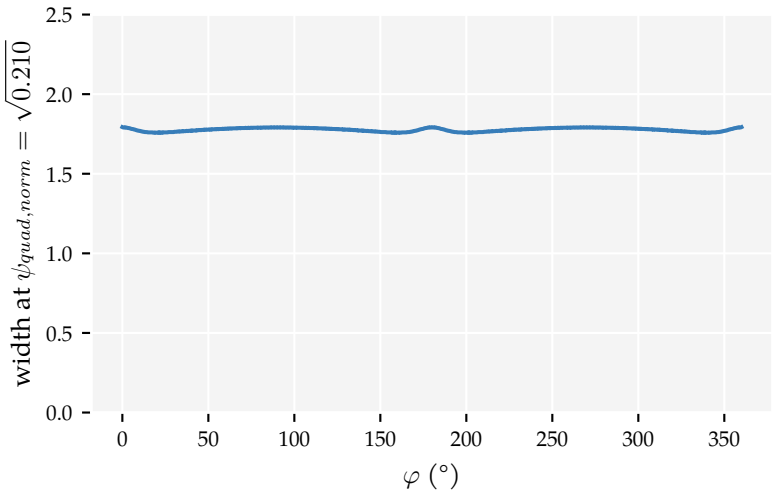


Figure A3. The width between the secondary anchor points $\pm\rho_{anch}$ of the normalised quadrature curve $\psi_{quad,norm}$ are almost constant 1.78 ± 0.011 with respect to the movement direction angle φ and is well approximated by the analytically determined value $4/\sqrt{5}$.

Appendix E. Additional Figures

This appendix provides additional figures for both analyses. For the first analysis method (Section 3.1), Figures A4 and A5 show the spatial contours of the median position error (Equation (4)) and median movement direction error (Equation (5)), respectively. Each column corresponds to the minimum sampling distance D_s of the training and optimisation set (Table 1). Figures A6 and A7 show the polar contours, indicating how a source's movement direction influences the errors.

For the second analysis method (Section 3.2), Figures A8 and A9 provide the spatial contours and Figures A10 and A11 the polar contours for this analysis. Each column corresponds to a configuration of sensor sensitivity axes: (x + y) both velocity components are measured by all sensors, (x | y) subsequent sensors measure v_x and v_y alternatingly, (x) all sensors measure only v_x, (y) all sensors measure only v_y.

Finally, Figures A12–A16 show the spatial and polar contours of the position and movement direction error for the comparison of the quadrature method (QM), the Gauss–Newton (GN) algorithm, and the multi-layer perceptron (MLP) using simulated sensors with lower signal to noise ratios (SNRs) as well as the movement direction error distributions and the total area with accurate movement direction predictions.

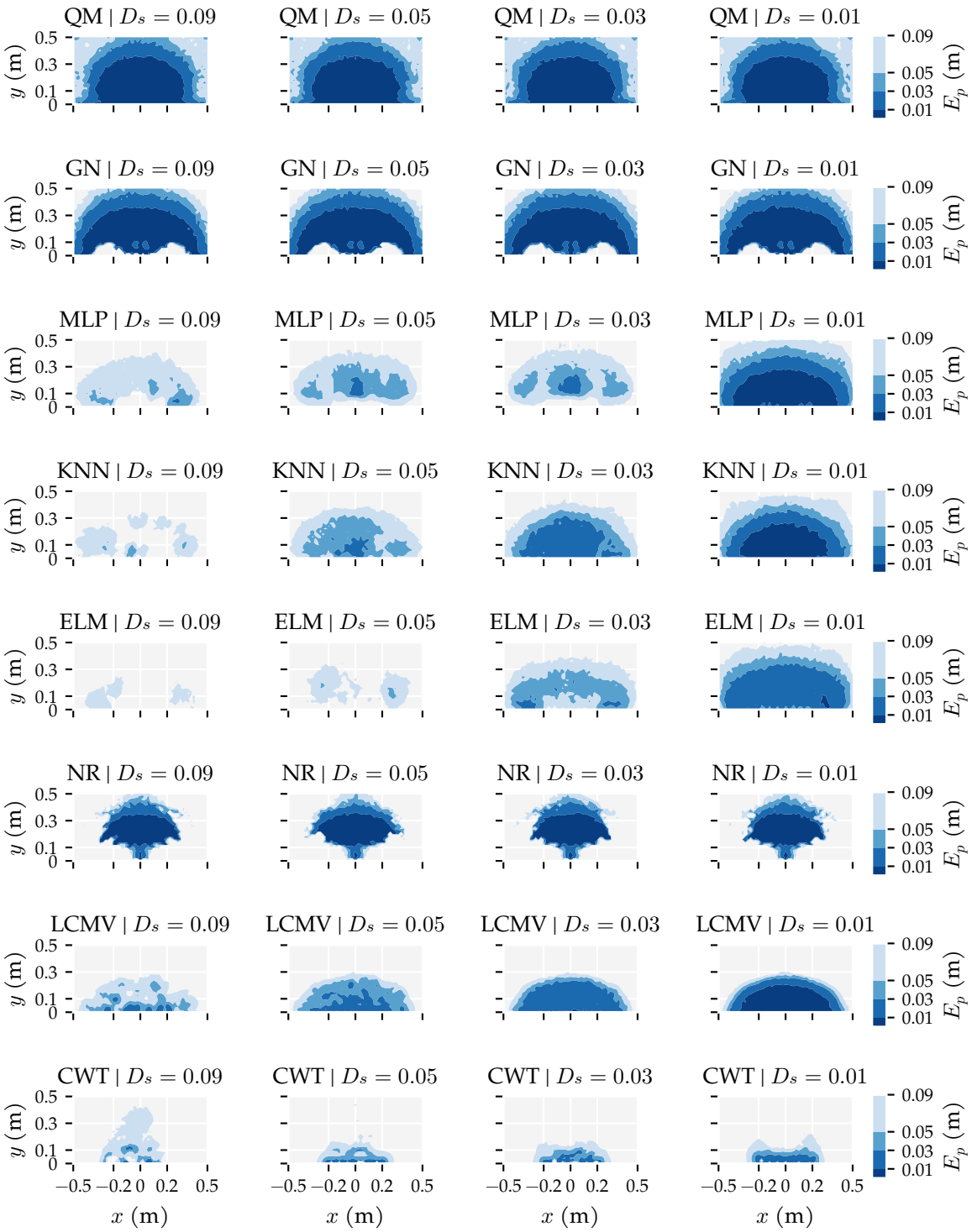


Figure A4. Spatial contours of the median position error (Equation (4)) for each localisation algorithm in all conditions of Analysis Method 1. This analysis varied the minimum distance D_s between sources in the training and optimisation sets (Table 1) while using the (x + y) sensor configuration at the $\sigma = 1 \times 10^{-5}$ m s^{-1} noise level. The median position error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

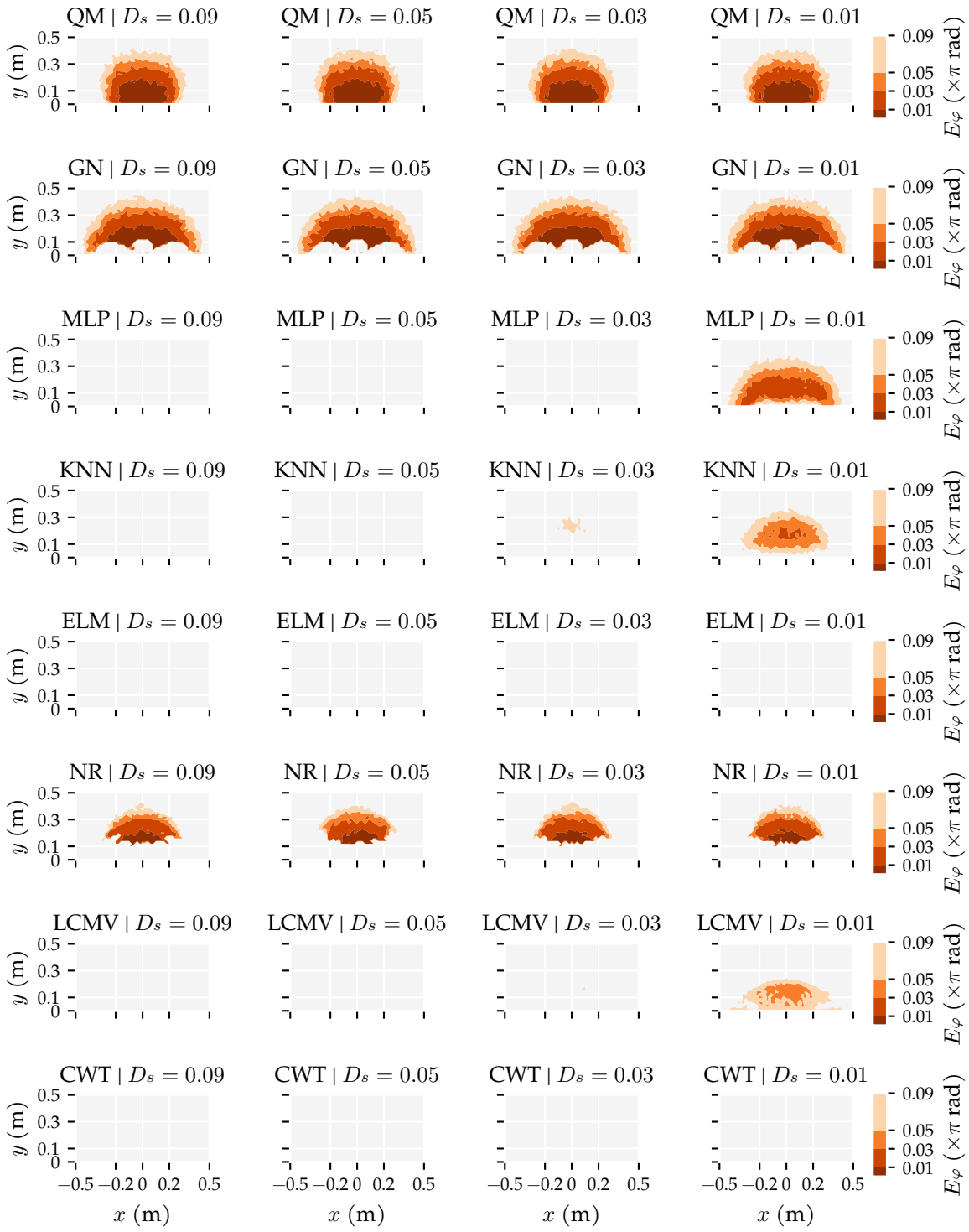


Figure A5. Spatial contours of the median movement direction error (Equation (5)) for each localisation algorithm in all conditions of Analysis Method 1. This analysis varied the minimum distance D_s between sources in the training and optimisation sets (Table 1) while using the (x + y) sensor configuration at the $\sigma = 1 \times 10^{-5}$ m s^{-1} noise level. The median movement direction error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

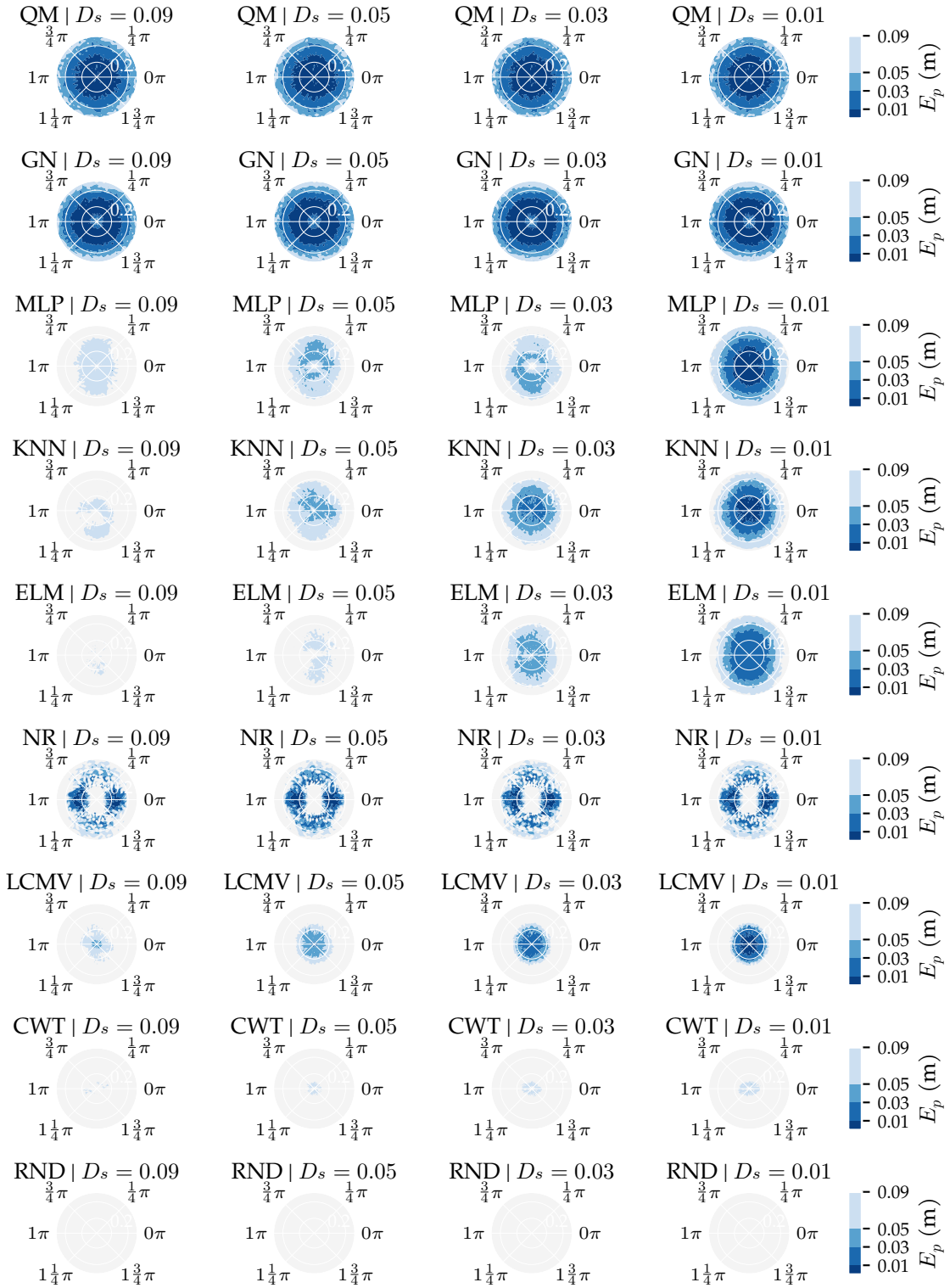


Figure A6. Polar contours of the median position error (Equation (4)) for each localisation algorithm in all conditions of Analysis Method 1. These figures indicate how the movement direction φ and distance d of a source influence the error. This analysis varied the minimum distance D_s between sources in the training and optimisation sets (Table 1) while using the (x + y) sensor configuration at the $\sigma = 1 \times 10^{-5}$ m s^{-1} noise level. The median position error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

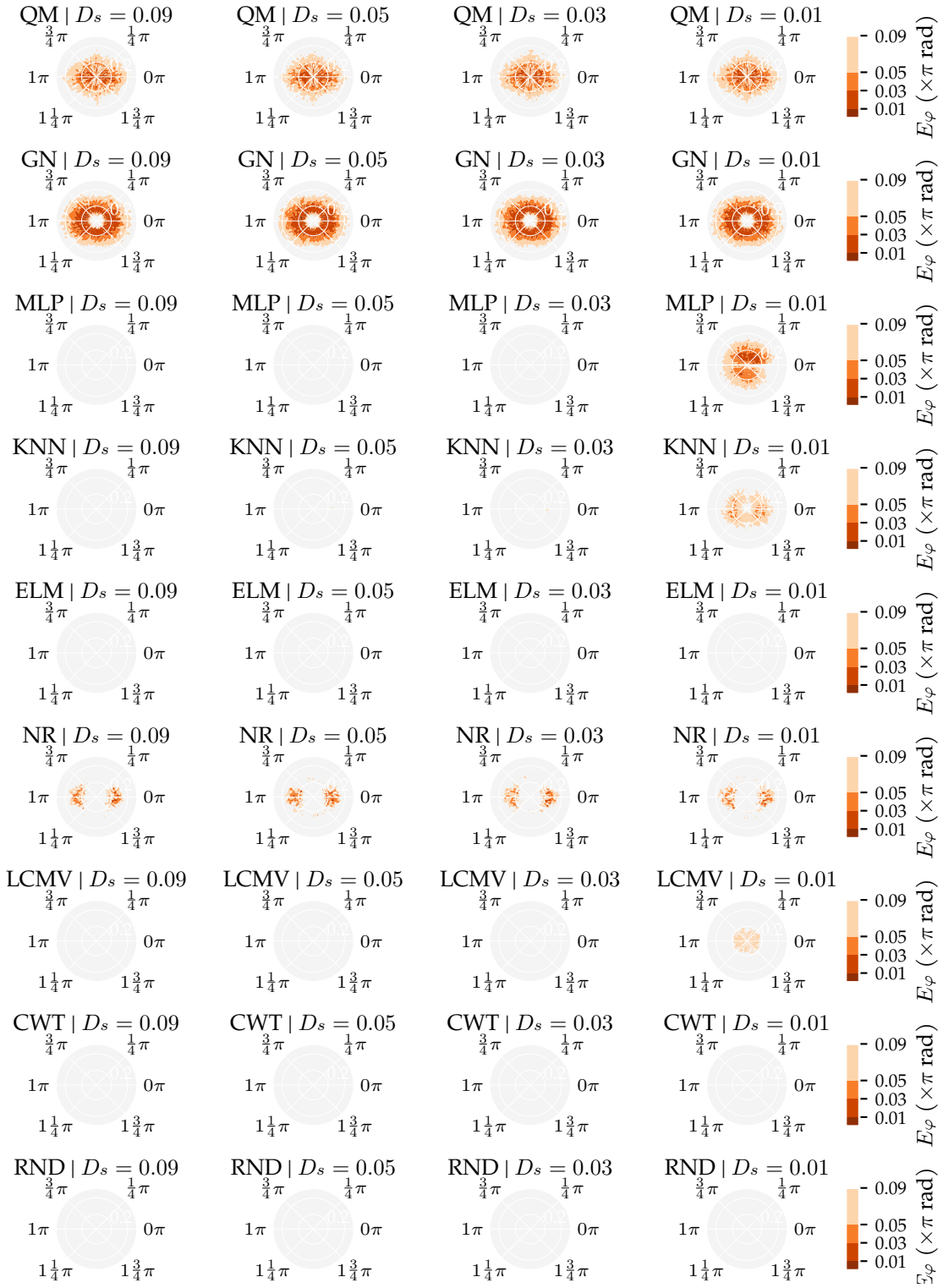


Figure A7. Polar contours of the median movement direction error (Equation (4)) for each localisation algorithm in all conditions of Analysis Method 1. These figures indicate how the movement direction φ and distance d of a source influence the error. This analysis varied the minimum distance D_s between sources in the training and optimisation sets (Table 1) while using the (x + y) sensor configuration at the $\sigma = 1 \times 10^{-5}$ m s^{-1} noise level. The median movement direction error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

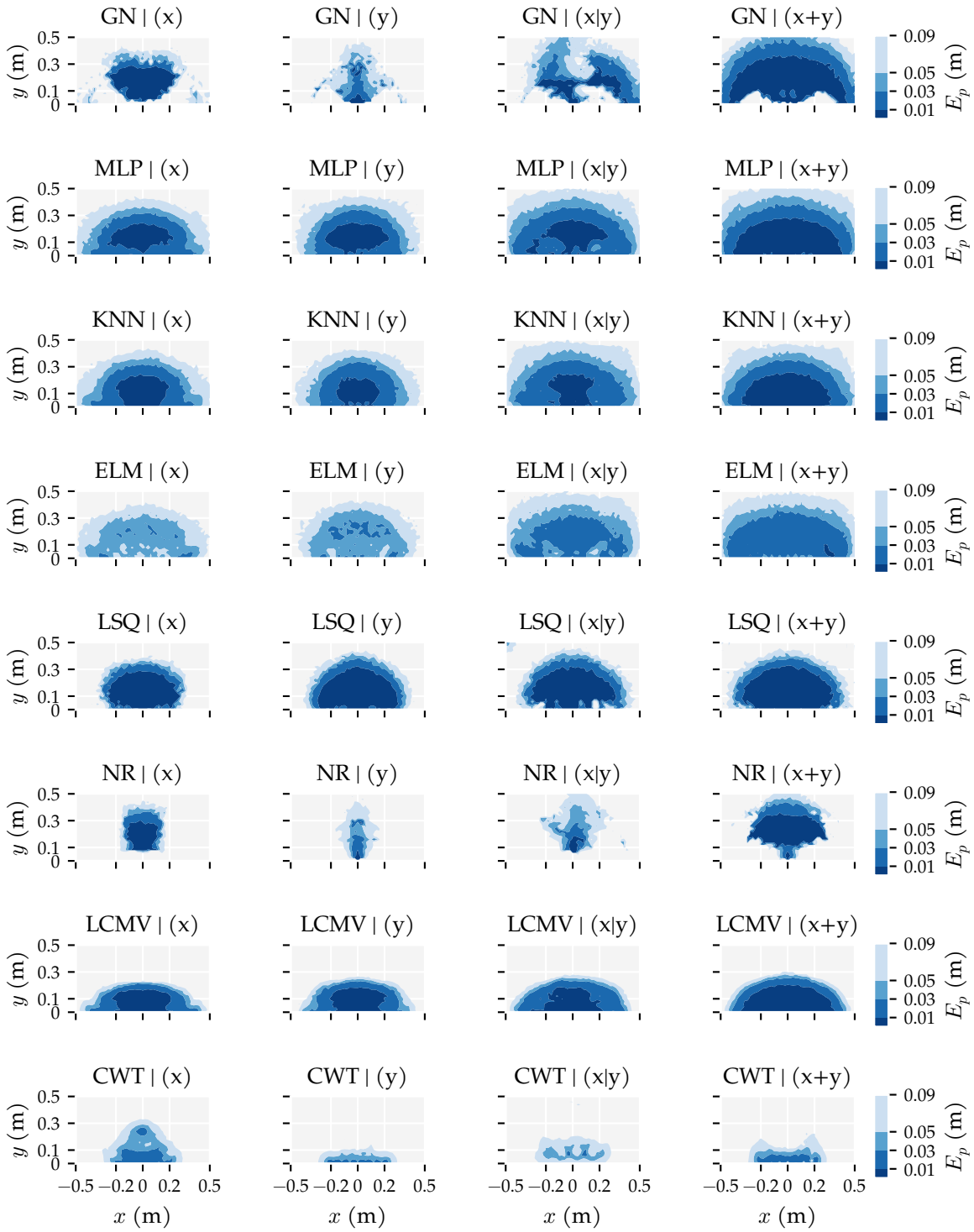


Figure A8. Spatial contours of the median position error (Equation (4)) for each localisation algorithm in all conditions of Analysis Method 2. This analysis varied the sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. The $D_s = 0.01$ training and optimisation set and $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level were used. The median position error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

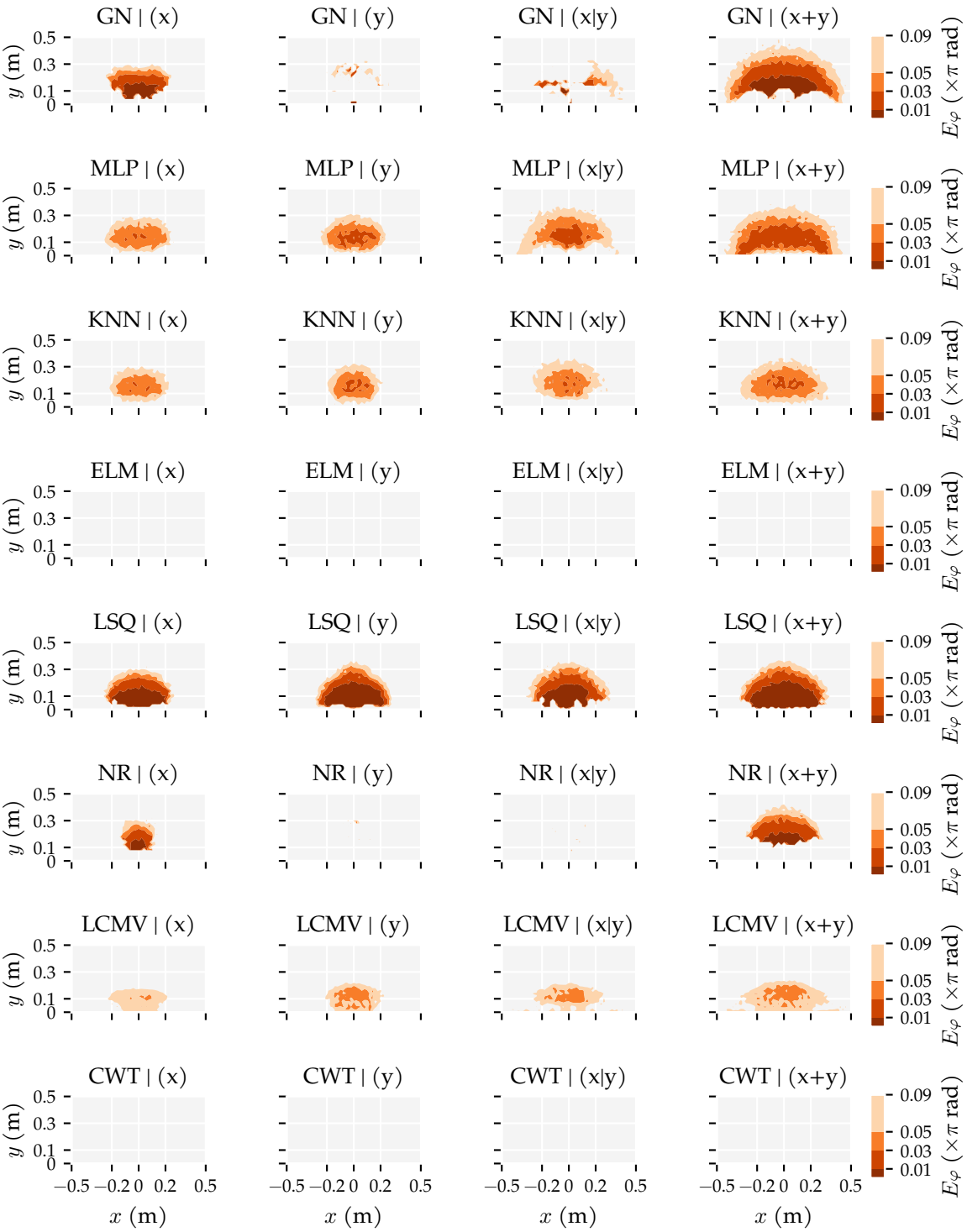


Figure A9. Spatial contours of the median movement direction error (Equation (5)) for each localisation algorithm in all conditions of Analysis Method 2. This analysis varied the sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. The $D_s = 0.01$ training and optimisation set and $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level were used. The median movement direction error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

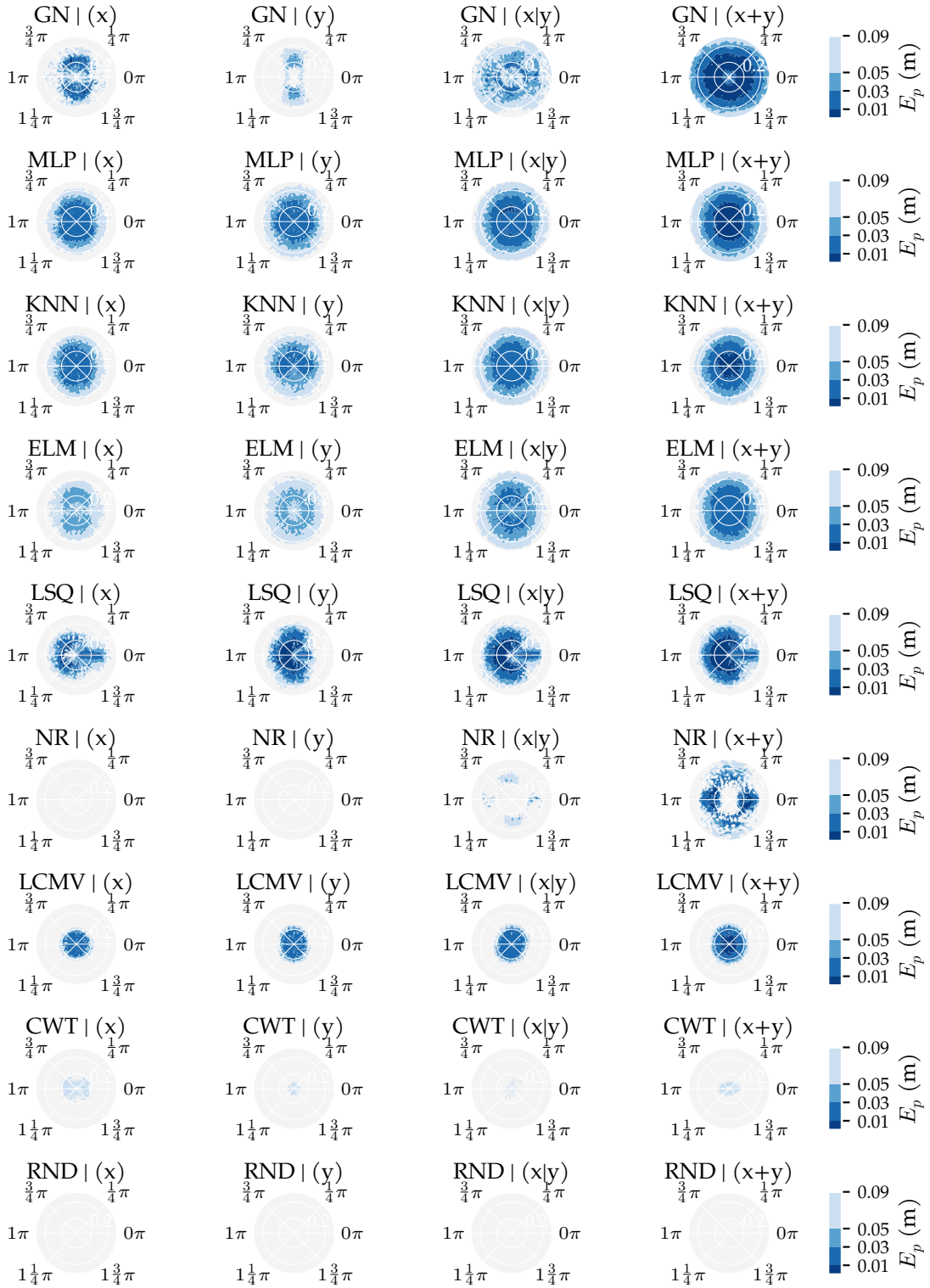


Figure A10. Polar contours of the median position error (Equation (4)) for each localisation algorithm in all conditions of Analysis Method 2. These figures indicate how the movement direction φ and distance d of a source influence the error. This analysis varied the sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. The $D_s = 0.01$ training and optimisation set and $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level were used. The median position error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

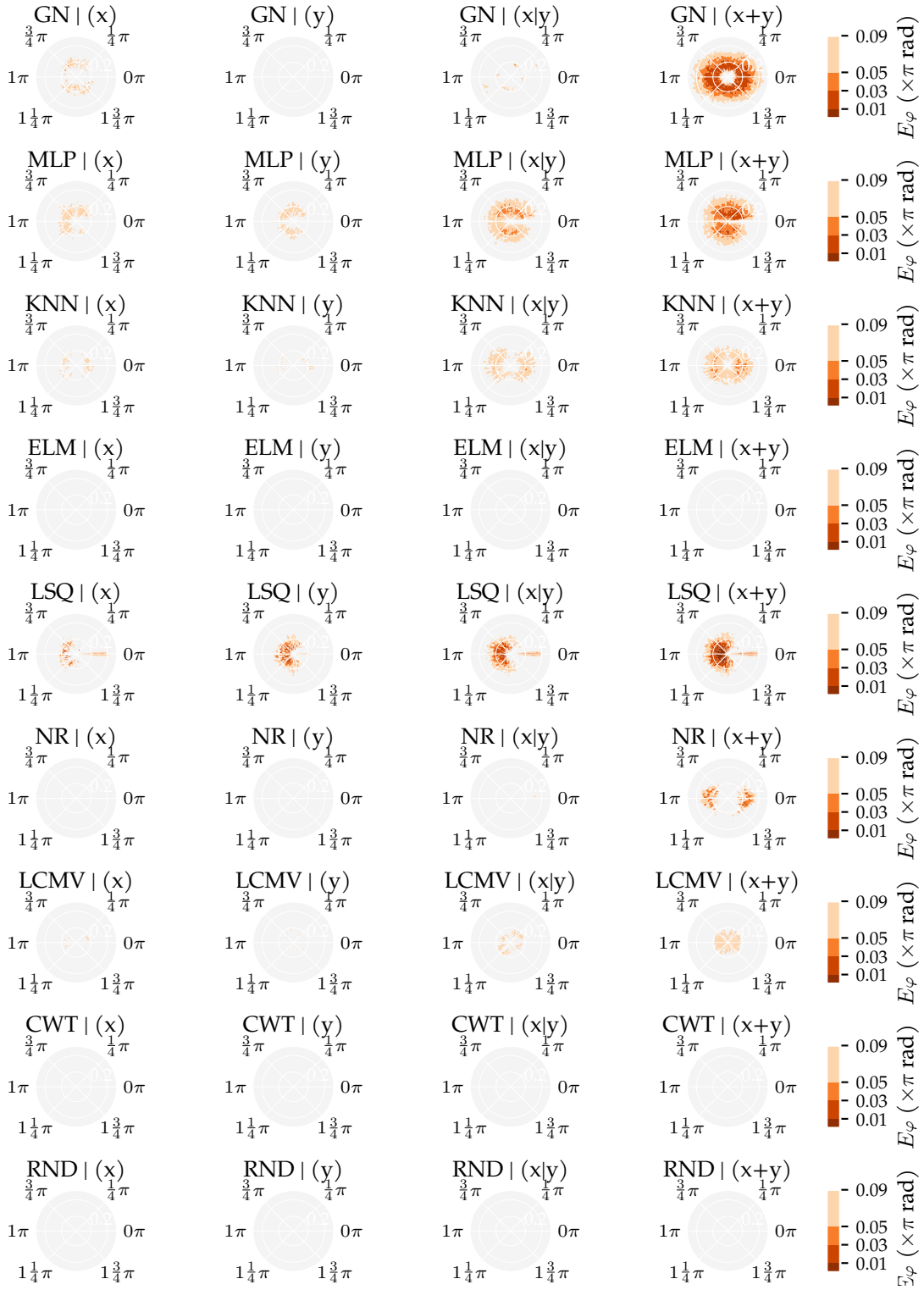
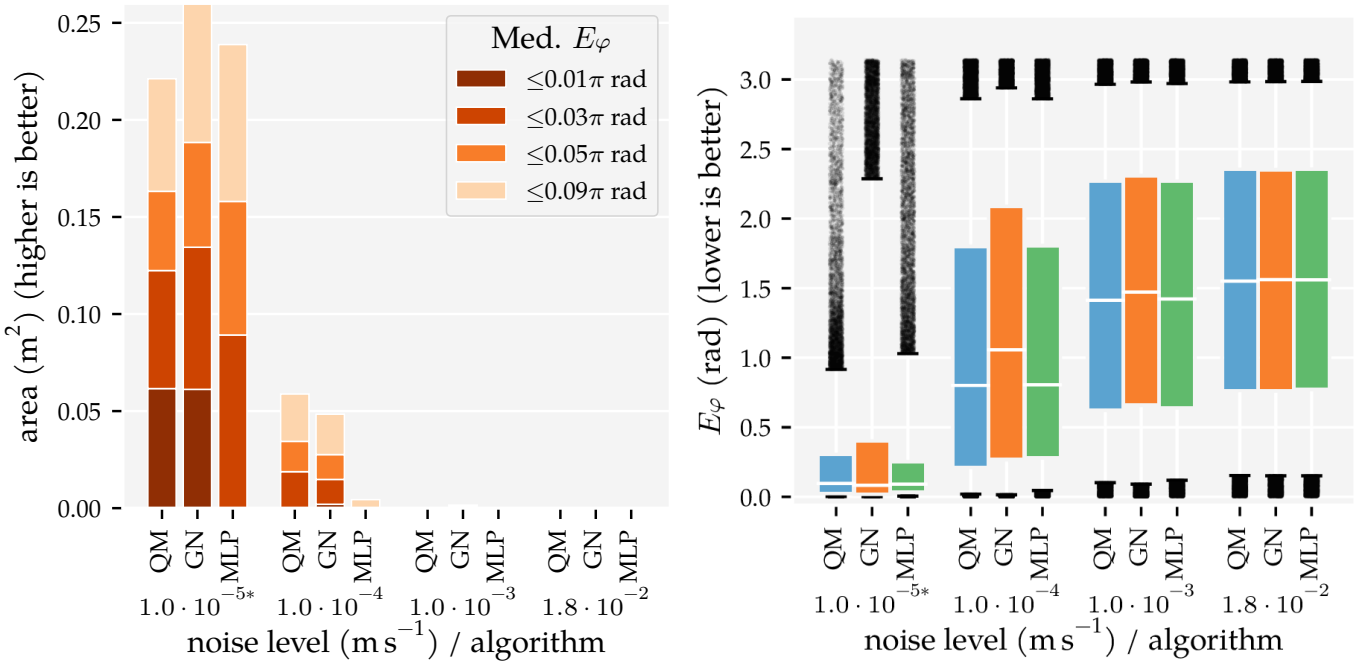


Figure A11. Polar contours of the median movement direction error (Equation (5)) for each localisation algorithm in all conditions of Analysis Method 2. These figures indicate how the movement direction φ and distance d of a source influence the error. This analysis varied the sensitivity axes of the sensors: (x + y) measured both velocity components at all sensors, (x | y) alternated measuring v_x and v_y for subsequent sensors, (x) measured only v_x at all sensors, (y) measured only v_y at all sensors. The $D_s = 0.01$ training and optimisation set and $\sigma = 1.0 \times 10^{-5}$ m s^{-1} noise level were used. The median movement direction error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.



(**a**) (**b**)

Figure A12. An overview of the movement direction error E_{φ} (Equation (5)) of QM, GN, and MLP using simulated sensors with higher velocity equivalent noise levels. (**a**) Total areas with a median movement direction error E_{φ} below 1 cm, 3 cm, 5 cm, and 9 cm. (**b**) Boxplots of the movement direction error distributions, whiskers indicate the 5th and 95th percentiles of the distributions. Predictions with errors outside these percentiles are shown individually. The values for $\sigma = 1.0 \times 10^{-5}$ m s^{-1} are based on the $D_s = 0.01$ condition in Analysis Method 1. The (x + y) sensor configuration was used. The MLP was re-trained for each noise level. Both the MLP and GN used the optimal hyperparameter values from the $D_s = 0.01$ condition of Analysis Method 1.

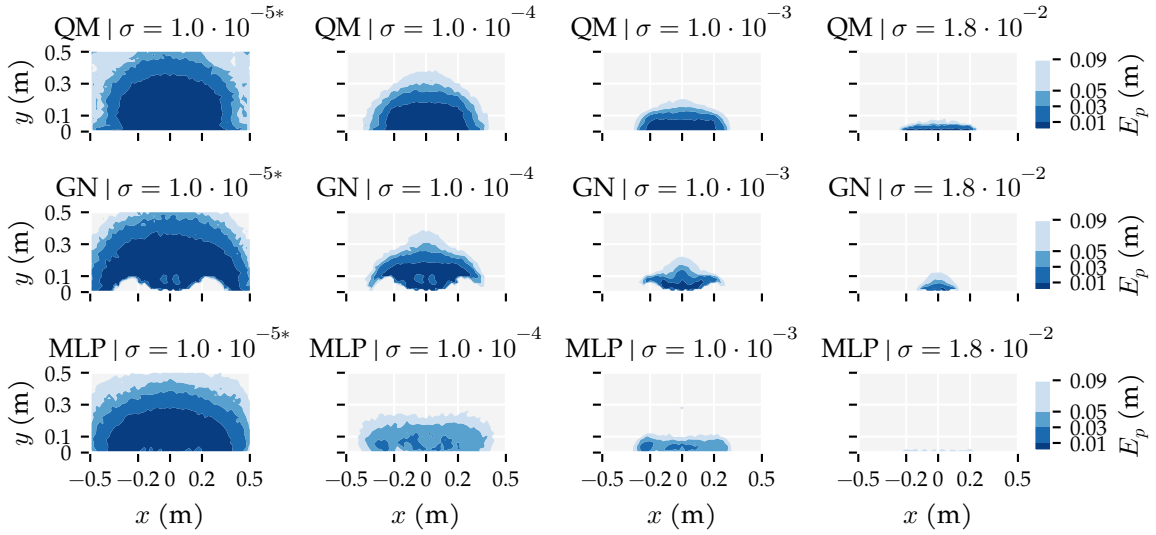


Figure A13. Spatial contours of the median position error (Equation (4)) for QM, GN, and MLP using simulated sensors with higher velocity equivalent noise levels. The $D_s = 0.01$ training and optimisation set and (x + y) sensor configuration were used. The values for $\sigma = 1.0 \times 10^{-5}$ m s^{-1} are based on the $D_s = 0.01$ condition in Analysis Method 1. The MLP was re-trained for each noise level. Both the MLP and GN used the optimal hyperparameter values from the $D_s = 0.01$ condition of Analysis Method 1. The median position error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

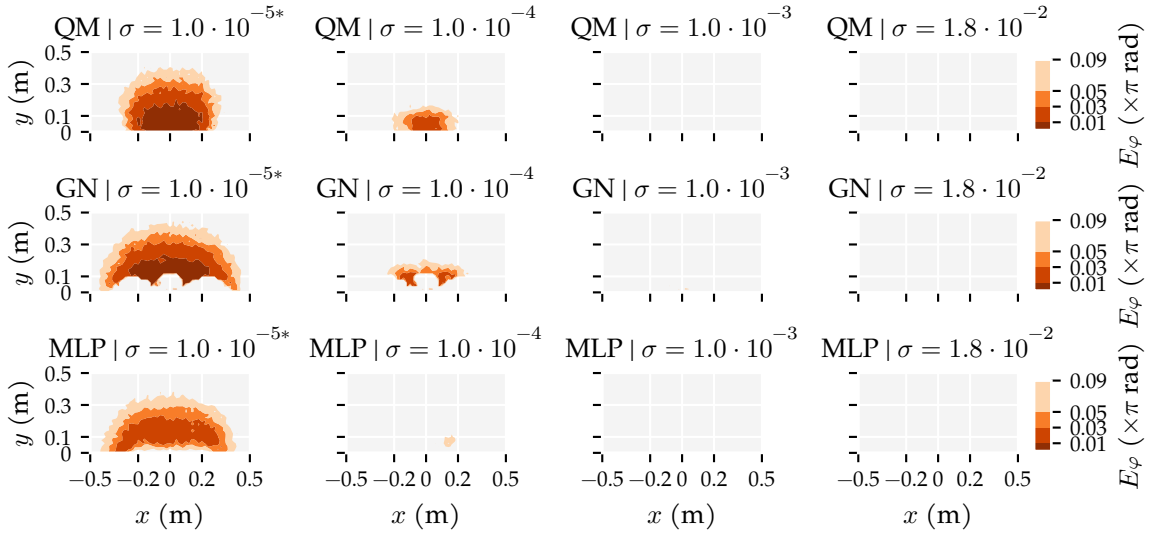


Figure A14. Spatial contours of the median movement direction error (Equation (5)) for QM, GN, and MLP using simulated sensors with higher velocity equivalent noise levels. The $D_s = 0.01$ training and optimisation set and (x + y) sensor configuration were used. The values for $\sigma = 1.0 \times 10^{-5}$ m s^{-1} are based on the $D_s = 0.01$ condition in Analysis Method 1. The MLP was re-trained for each noise level. Both the MLP and GN used the optimal hyperparameter values from the $D_s = 0.01$ condition of Analysis Method 1. The median movement direction error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

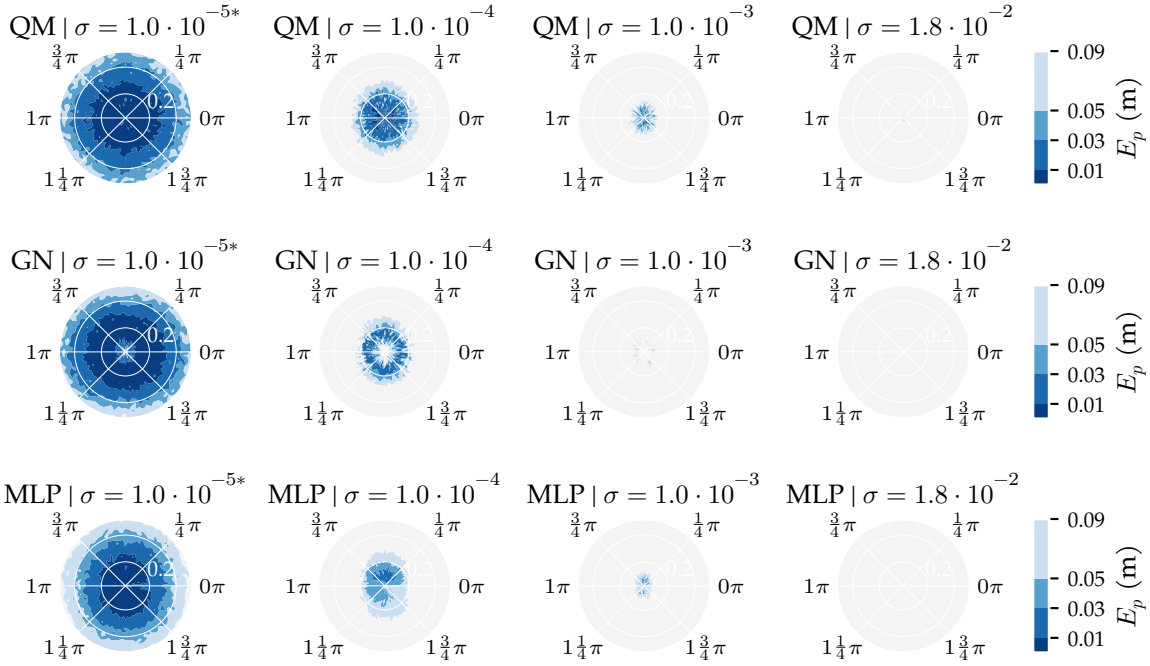


Figure A15. Polar contours of the median position error (Equation (4)) for QM, GN, and MLP using simulated sensors with higher velocity equivalent noise levels. These figures indicate how the movement direction φ and distance d of a source influence the error. The $D_s = 0.01$ training and optimisation set and (x + y) sensor configuration were used. The values for $\sigma = 1.0 \times 10^{-5}$ m s^{-1} are based on the $D_s = 0.01$ condition in Analysis Method 1. The MLP was re-trained for each noise level. Both the MLP and GN used the optimal hyperparameter values from the $D_s = 0.01$ condition of Analysis Method 1. The median position error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

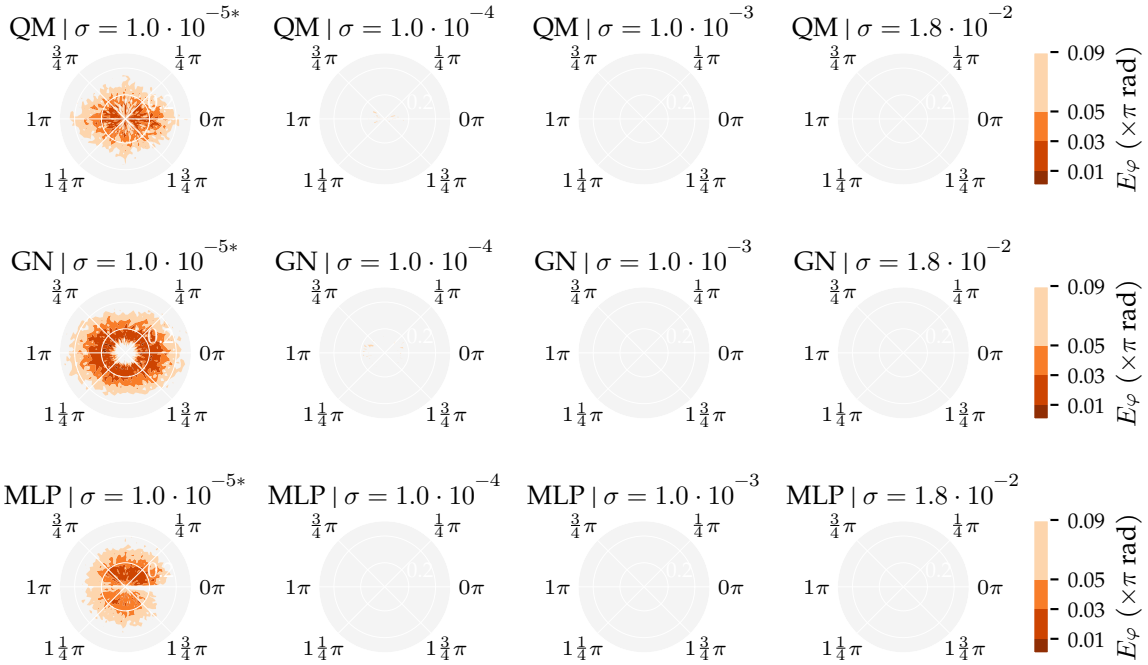


Figure A16. Polar contours of the median movement direction error (Equation (5)) for each localisation algorithm in all conditions of Analysis Method 2. These figures indicate how the movement direction φ and distance d of a source influence the error. The $D_s = 0.01$ training and optimisation set and (x + y) sensor configuration were used. The values for $\sigma = 1.0 \times 10^{-5}$ m s^{-1} are based on the $D_s = 0.01$ condition in Analysis Method 1. The MLP was re-trained for each noise level. Both the MLP and GN used the optimal hyperparameter values from the $D_s = 0.01$ condition of Analysis Method 1. The median movement direction error was computed in 2×2 cm^2 cells. The sensors were equidistantly placed between $x = \pm 0.2$ m.

References

1. Dijkgraaf, S. The Functioning and Significance of the Lateral-Line Organs. *Biol. Rev.* **1963**, *38*, 51–105. [CrossRef] [PubMed]
2. Coombs, S.; van Netten, S. The Hydrodynamics and Structural Mechanics of the Lateral Line System. In *Fish Physiology*; Elsevier: Amsterdam, The Netherlands, 2005; Volume 23, pp. 103–139. [CrossRef]
3. Yang, Y.; Chen, J.; Engel, J.; Pandya, S.; Chen, N.; Tucker, C.; Coombs, S.; Jones, D.L.; Liu, C. Distant touch hydrodynamic imaging with an artificial lateral line. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 18891–18895. [CrossRef] [PubMed]
4. Vollmayr, A.N.; Sosnowski, S.; Urban, S.; Hirche, S.; van Hemmen, J.L. Snookie: An Autonomous Underwater Vehicle with Artificial Lateral-Line System. In *Flow Sensing in Air and Water*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 521–562. [CrossRef]
5. Ćurčić-Blake, B.; van Netten, S.M. Source location encoding in the fish lateral line canal. *J. Exp. Biol.* **2006**, *209*, 1548–1559. [CrossRef] [PubMed]
6. Pandya, S.; Yang, Y.; Jones, D.L.; Engel, J.; Liu, C. Multisensor Processing Algorithms for Underwater Dipole Localization and Tracking Using MEMS Artificial Lateral-Line Sensors. *EURASIP J. Adv. Signal Process.* **2006**, *2006*, 076593. [CrossRef]
7. Abdulsadda, A.T.; Tan, X. An artificial lateral line system using IPMC sensor arrays. *Int. J. Smart Nano Mater.* **2012**, *3*, 226–242. [CrossRef]
8. Abdulsadda, A.T.; Tan, X. Nonlinear estimation-based dipole source localization for artificial lateral line systems. *Bioinspir. Biomim.* **2013**, *8*, 026005. [CrossRef] [PubMed]
9. Boulogne, L.H.; Wolf, B.J.; Wiering, M.A.; van Netten, S.M. Performance of neural networks for localizing moving objects with an artificial lateral line. *Bioinspir. Biomim.* **2017**, *12*, 56009. [CrossRef] [PubMed]
10. Wolf, B.J.; Warmelink, S.; van Netten, S.M. Recurrent neural networks for hydrodynamic imaging using a 2D-sensitive artificial lateral line. *Bioinspir. Biomim.* **2019**, *14*, 055001. [CrossRef]
11. Nguyen, N.; Jones, D.; Pandya, S.; Yang, Y.; Chen, N.; Tucker, C.; Liu, C. Biomimetic Flow Imaging with an Artificial Fish Lateral Line. In Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSIGNALS), Madeira, Portugal, 19–21 January 2018; pp. 269–276. [CrossRef]
12. Yang, Y.; Nguyen, N.; Chen, N.; Lockwood, M.; Tucker, C.; Hu, H.; Bleckmann, H.; Liu, C.; Jones, D.L. Artificial lateral line with biomimetic neuromasts to emulate fish sensing. *Bioinspir. Biomim.* **2010**, *5*, 016001. [CrossRef]

13. Nguyen, N.; Jones, D.L.; Yang, Y.; Liu, C. Flow Vision for Autonomous Underwater Vehicles via an Artificial Lateral Line. *EURASIP J. Adv. Signal Process.* **2011**, *2011*, 806406. [CrossRef]
14. Wolf, B.J.; van Netten, S.M. Hydrodynamic Imaging using an all-optical 2D Artificial Lateral Line. In Proceedings of the 2019 IEEE Sensors Applications Symposium, Sophia Antipolis, France, 11–13 March 2019; pp. 1–6. [CrossRef]
15. Wolf, B.J.; van de Wolfshaar, J.; van Netten, S.M. Three-dimensional multi-source localization of underwater objects using convolutional neural networks for artificial lateral lines. *J. R. Soc. Interface* **2020**, *17*, 20190616. [CrossRef]
16. Lamb, H. *Hydrodynamics.* Cambridge University Press: Cambridge, UK, 1924; p. 687.
17. Que, R.; Zhu, R. A Two-Dimensional Flow Sensor with Integrated Micro Thermal Sensing Elements and a Back Propagation Neural Network. *Sensors* **2013**, *14*, 564–574. [CrossRef]
18. Pjetri, O.; Wiegerink, R.J.; Krijnen, G.J.M. A 2D particle velocity sensor with minimal flow-disturbance. *IEEE Sens. J.* **2015**, *16*, 8706–8714. [CrossRef]
19. Lei, H.; Sharif, M.A.; Tan, X. Dynamics of Omnidirectional IPMC Sensor: Experimental Characterization and Physical Modeling. *IEEE/ASME Trans. Mechatron.* **2016**, *21*, 601–612. [CrossRef]
20. Wolf, B.J.; Morton, J.A.S.; MacPherson, W.N.; van Netten, S.M. Bio-inspired all-optical artificial neuromast for 2D flow sensing. *Bioinspir. Biomim.* **2018**, *13*, 026013. [CrossRef]
21. Lu, Z.; Popper, A. Neural response directionality correlates of hair cell orientation in a teleost fish. *J. Comp. Physiol. A Sens. Neural Behav. Physiol.* **2001**, *187*, 453–465. [CrossRef]
22. Kalmijn, A.J. Hydrodynamic and Acoustic Field Detection. In *Sensory Biology of Aquatic Animals*; Springer: New York, NY, USA, 1988; pp. 83–130. [CrossRef]
23. Abdulsadda, A.T.; Tan, X. Underwater Tracking and Size-Estimation of a Moving Object Using an IPMC Artificial Lateral Line. In *Smart Materials, Adaptive Structures and Intelligent Systems*; American Society of Mechanical Engineers: New York, NY, USA, 2012; pp. 657–665. [CrossRef]
24. Abdulsadda, A.T.; Tan, X. Underwater tracking of a moving dipole source using an artificial lateral line: Algorithm and experimental validation with ionic polymer–metal composite flow sensors. *Smart Mater. Struct.* **2013**, *22*, 045010. [CrossRef]
25. Franosch, J.M.P.; Sichert, A.B.; Suttner, M.D.; Van Hemmen, J.L. Estimating position and velocity of a submerged moving object by the clawed frog Xenopus and by fish—A cybernetic approach. *Biol. Cybern.* **2005**, *93*, 231–238. [CrossRef]
26. Goulet, J.; Engelmann, J.; Chagnaud, B.P.; Franosch, J.M.P.; Suttner, M.D.; van Hemmen, J.L. Object localization through the lateral line system of fish: Theory and experiment. *J. Comp. Physiol. A* **2008**, *194*, 1–17. [CrossRef]
27. Pandya, S.; Yang, Y.; Liu, C.; Jones, D.L. Biomimetic Imaging of Flow Phenomena. In Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing—ICASSP '07, Honolulu, HI, USA, 15–20 April 2007; Volume 2, pp. II-933–II-936. [CrossRef]
28. Sichert, A.B.; Bamler, R.; van Hemmen, J.L. Hydrodynamic Object Recognition: When Multipoles Count. *Phys. Rev. Lett.* **2009**, *102*, 058104. [CrossRef]
29. Coombs, S.; Hastings, M.; Finneran, J. Modeling and measuring lateral line excitation patterns to changing dipole source locations. *J. Comp. Physiol. A* **1996**, *178*, 359–371. [CrossRef]
30. Jiang, Y.; Ma, Z.; Zhang, D. Flow field perception based on the fish lateral line system. *Bioinspir. Biomim.* **2019**, *14*, 041001. [CrossRef]
31. McConney, M.E.; Chen, N.; Lu, D.; Hu, H.A.; Coombs, S.; Liu, C.; Tsukruk, V.V. Biologically inspired design of hydrogel-capped hair sensors for enhanced underwater flow detection. *Soft Matter* **2009**, *5*, 292–295. [CrossRef]
32. Asadnia, M.; Kottapalli, A.G.P.; Karavitaki, K.D.; Warkiani, M.E.; Miao, J.; Corey, D.P.; Triantafyllou, M. From Biological Cilia to Artificial Flow Sensors: Biomimetic Soft Polymer Nanosensors with High Sensing Performance. *Sci. Rep.* **2016**, *6*, 32955. [CrossRef] [PubMed]
33. Asadnia, M.; Kottapalli, A.G.P.; Miao, J.; Warkiani, M.E.; Triantafyllou, M.S. Artificial fish skin of self-powered micro-electromechanical systems hair cells for sensing hydrodynamic flow phenomena. *J. R. Soc. Interface* **2015**, *12*, 20150322. [CrossRef] [PubMed]
34. Yang, Y.; Pandya, S.; Chen, J.; Engel, J.; Chen, N.; Liu, C. Micromachined Hot-Wire Boundary Layer Flow Imaging Array. In *CANEUS: MNT for Aerospace Applications*; American Society of Mechanical Engineers Digital Collection (ASMEDC): New York City, NY, USA, 2006; pp. 213–218. [CrossRef]
35. Abdulsadda, A.T.; Tan, X. Underwater source localization using an IPMC-based artificial lateral line. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2719–2724. [CrossRef]
36. Kottapalli, A.G.P.; Bora, M.; Asadnia, M.; Miao, J.; Venkatraman, S.S.; Triantafyllou, M. Nanofibril scaffold assisted MEMS artificial hydrogel neuromasts for enhanced sensitivity flow sensing. *Sci. Rep.* **2016**, *6*, 19336. [CrossRef] [PubMed]
37. Dunbar, D.; Humphreys, G. A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. Graph.* **2006**, *25*, 503–508. [CrossRef]
38. MATLAB. *Version 9.4.0.813654 (R2018a)*; The MathWorks Inc.: Natick, MA, USA, 2018.
39. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: A new learning scheme of feedforward neural networks. In Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), Budapest, Hungary, 25–29 July 2004; Volume 2, pp. 985–990. [CrossRef]

40. Liang, N.-Y.; Huang, G.-B.; Saratchandran, P.; Sundararajan, N. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Trans. Neural Netw.* **2006**, *17*, 1411–1423. [CrossRef]
41. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
42. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; Volume 9, pp. 249–256.
43. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
44. Constrained Nonlinear Optimization Algorithms—MATLAB & Simulink. Available online: https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html (accessed on 13 April 2020).
45. Wolf, B.J.; van Netten, S.M. Training submerged source detection for a 2D fluid flow sensor array with extreme learning machines. In Proceedings of the Eleventh International Conference on Machine Vision (ICMV 2018), Munich, Germany, 1–3 November 2018; Nikolaev, D.P., Radeva, P., Verikas, A., Zhou, J., Eds.; International Society for Optics and Photonics (SPIE): Bellingham, WA, USA, 2019; Volume 11041, p. 2. [CrossRef]
46. Windsor, S.P.; Norris, S.E.; Cameron, S.M.; Mallinson, G.D.; Montgomery, J.C. The flow fields involved in hydrodynamic imaging by blind Mexican cave fish (Astyanax fasciatus). Part I: Open water and heading towards a wall. *J. Exp. Biol.* **2010**, *213*, 3819–3831. [CrossRef]
47. Lin, X.; Wu, J.; Qin, Q. A novel obstacle localization method for an underwater robot based on the flow field. *J. Mar. Sci. Eng.* **2019**, *7*, 437. [CrossRef]
48. Bot, D.; Wolf, B.; van Netten, S. Dipole Localisation Predictions Data Set. 2021. Available online: https://doi.org/10.5281/zenodo.4973492 (accessed on 30 June 2021).
49. Bot, D.; Wolf, B.; van Netten, S. Dipole Localisation Algorithms for Simulated Artificial Lateral Line. 2021. Available online: https://doi.org/10.5281/zenodo.4973515 (accessed on 30 June 2021).
50. Mallat, S. *A Wavelet Tour of Signal Processing*; Elsevier: Amsterdam, The Netherlands, 1999.