Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

journal homepage: www.elsevier.com/locate/jpdc

Reliable IoT analytics at scale

Panagiotis Gkikopoulos [a,b,*], Peter Kropf [a], Valerio Schiavoni [a], Josef Spillner [b]

[a] University of Neuchâtel, Avenue du 1er-Mars 26, 2000, Neuchâtel, Switzerland
[b] Zurich University of Applied Sciences, Obere Kirchgasse 2, 8400, Winterthur, Switzerland

ARTICLE INFO

Keywords:
Consensus voting
Data reliability
Data fusion
Sensor redundancy

ABSTRACT

Societies and legislations are moving towards automated decision-making based on measured data in safety-critical environments. Over the next years, density and frequency of measurements will increase to generate more insights and get a more solid basis for decisions, including through redundant low-cost sensor deployments. The resulting data characteristics lead to large-scale system design in which small input data errors may lead to severe cascading problems including ultimately wrong decisions. To ensure internal data consistency to mitigate this risk in such IoT environments, fast-paced data fusion and consensus among redundant measurements need to be achieved. In this context, we introduce *history-aware sensor fusion* powered by *accurate voting with clustering* as a promising approach to achieve fast and informed consensus, which can converge to the output up to 4X faster than the state of the art history-based voting. Leveraging three case studies, we investigate different voting schemes and show how this approach can improve data accuracy by up to 30% and performance by up to 12% compared to state-of-the-art sensor fusion approaches. We furthermore contribute a specification format for easily deploying our methods in practice and use it to develop a pilot implementation.

1. Introduction

Automated decision-making based on interpretations of data from the real world [9,12] is a growing trend across societies, especially in digitally transformed economies and legislations. This trend entails a number of ethical concerns around mispredictions and wrong decisions, economic trade-offs related to investment into data acquisition equipment and maintenance, as well as technical challenges associated to ensuring proper quality of the acquired data. Data cleaning and cleansing techniques [37] are commonly used on a single measurement source, on a single time series to mitigate device downtimes and transmission errors, as well as and to eliminate unsuitable data records, effectively increasing the data quality. These *ex post* cleansing techniques can however not prevent wrong observations that result from combining multiple sources with potentially diverging or conflicting views. In addition to the above techniques that are applied to each individual data stream, we need to perform multi-view data fusion [43] to arrive at a common ground truth across multiple data sources beforehand, especially in the case of redundant sensors. In this context, we employ voting-based data fusion to merge observations from multiple sensors that, in ideal conditions, would produce identical outputs. In cases where independent ground truth measurements are not available or affordable, we dynamically assign the role of ground truth to the latest agreement among candidate the values. These additional steps are shown in Fig. 1, which shows where our methodology fits in a system that uses IoT devices to obtain observations.

The necessity for higher-quality sensor data applies especially to emerging digitalised systems, such as smart cities, industrial production and other cyber-physical domains. They involve large amounts of continuous data streams aggregated from a multitude of sensors, used as data sources. Sensor-based measurements are common especially in the Internet of Things (IoT) for triggering data-driven decisions. Quality issues in such distributed measurements [45] have profound adverse effects on systems leverage, and by extension, on humans and society, as shown in *irresponsible AI* community discussions [27]. Consequently, steps to improve input data quality within the data pre-processing step and in conjunction with data fusion are necessary to achieve better decisions and overall more reliable applications. Data fusion in general is a technique of merging different inputs [31] in an application-independent middleware to obtain a holistic view of physical objects. Data sources used as inputs to a data fusion system can have partial information that becomes meaningful once combined, or the complete

* Corresponding author at: University of Neuchâtel, Avenue du 1er-Mars 26, 2000, Neuchâtel, Switzerland.
E-mail addresses: panosgkikopoulos@gmail.com (P. Gkikopoulos), peter.kropf@unine.ch (P. Kropf), valerio.schiavoni@unine.ch (V. Schiavoni), josef.spillner@zhaw.ch (J. Spillner).

https://doi.org/10.1016/j.jpdc.2024.104840
Received 4 February 2023; Received in revised form 22 November 2023; Accepted 7 January 2024
Available online 15 January 2024

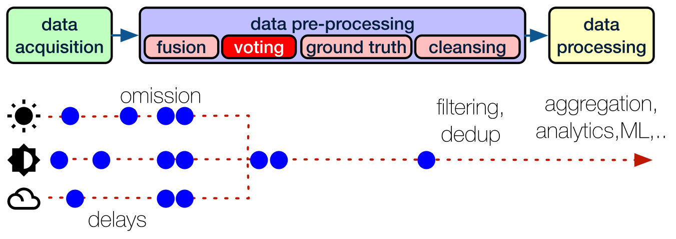


Fig. 1. Positioning of data fusion, voting and deriving the ground truth in a typical IoT data pipeline.

information but with potential conflicts between sources. Different techniques can be applied to each of these cases. Voting is an approach to fuse sensor data for the purposes of reliability and error mitigation [24,21] in safety-critical environments. For instance, in avionics, three redundant physical sensors are mandated for each logical sensor [21], and vehicles with autopilot abilities have configurations such as eight cameras (including three forward ones) to achieve redundancy on the critical data acquisition path. Thus, in the absence of external ground truth (*i.e.* a fully trusted and accurate data source, often too expensive in practice), voting is a pragmatic substitute as it leads to internal ground truth upon which critical decision-making can be based.

In this work, we study and observe state-of-the-art voting approaches for sensor data fusion, applying them to three IoT data capturing scenarios relying on redundant sensor measurements: *(i)* light sensors in a smart building setting, *(ii)* Bluetooth Low Energy (BLE) beacons to track vehicle position in a (simulated) tunnel and *(iii)* an indoor positioning experiment, also using BLE beacons, emulating a smart shopping scenario. We focus on voting algorithms used to reach *data-centric consensus*[18] on numerical values, as these are relevant when merging sensor readings and leverage historical records to factor in the reliability of individual sensors. In §7 we conduct three experiments on such IoT setups for real-time validation and pre-recorded data for the purpose of reproducibility. We exploit our findings to contribute a generic specification format that can be used to define voting schemes for several applications, particularly optimized for IoT and cyber-physical applications. We argue that such a format aids the development of distributed analytics applications by making them more needs-focused and reliable, while shielding software engineers from the voting implementation. Moreover, the increased robustness of the data, induced by multi-view data fusion implemented through the use of voting, facilitates the input data quality in data-centric artificial intelligence, a recent research direction aimed at overcoming *misprediction* due to lack of input data assurance [39]. We replicate and evaluate the state-of-the-art history-based voting algorithms, *i.e.* voting algorithms in which the weight of votes is determined by past performance of the candidate in terms of agreement with the consensus. We observe the need for a method to bootstrap the algorithms to be more accurate before the history has been established, which would also improve the number of voting rounds it would take for the weights to converge to stable values representative of the reliability of the sensors.

Our contributions are twofold: *(1)* AVOC (Accurate Voting with Clustering), a novel bootstrapping method for initializing history-based voting systems, which we fully implement and evaluate with three practical IoT scenarios; *(2)* VDX, a new voting definition specification that precisely defines application requirements and allows users to select appropriate parameters for software voters.

The article extends and refines our previous work on AVOC [17] in multiple ways. First, it adds a dimension of performance and scalability, as discussed in 7 and 8. We evaluated the performance of the voting compared to state-of-the-art data fusion approaches and examined how it affects the scalability of the sensor fusion setup in terms of increased numbers of modules. Second, it details the third scenario of reliable indoor positioning, as detailed in §6. This scenario allows us to compare with Kalman Filtering, a state-of-the-art approach for multilateration-based postitioning, in terms of performance and accuracy. Based on this experiment, we also discuss the implications on performance of scaling our system versus scaling the existing solutions. Third, it presents additional details about our effort to replicate history-aware voting algorithms and implement them within our VDX software, prior to modifying them with AVOC.

The rest of the article is structured as follows. §2 surveys state-of-the-art voting algorithms, data fusion and data quality issues relevant to IoT. §3 investigates voting algorithms used to reconcile redundant data values. §4 presents AVOC, our approach. We describe the proposed format VDX for defining the voting process in §5. In §6 we detail our use-case scenarios and how we built our hardware prototypes. §7 presents our experimental evaluation using both a reference scenario dataset and our experimental setup. Our findings are discussed in §8. We conclude in §9 by discussing our findings and prospect future research directions in redundancy-based data quality in IoT.

2. Related work

Data-driven decisions [16] are key elements of cyber-physical systems and digitalised applications of all scales and domains (*e.g.*, smart cities, mobility, industrial production, home automation, *etc.*). In many such systems, incoming data are subject to real-time analysis and subsequent decision-making. However, while systems research has allowed dealing with the volume, velocity and variety of multi-source data involved in the processing chain [36], issues still emerge regarding data quality, value and veracity. In this work, we focus on the accuracy (*i.e.*, quality) for sensor measurements [34]. Data fusion across multiple homogeneous or heterogeneous sensors has been utilised to tackle the challenges induced by problematic data, improving analytics reliability through a variety of techniques (*e.g.*, data association, state estimation, decision fusion, classification, prediction, machine learning and analytics [25]). Initial efforts exist to create standards and frameworks for data management and interoperability, for instance in the smart city space [20,22]. However, they currently lack a common framework and standardized format. Our work proposes a new interoperable format to define voting-related data fusion.

Voting algorithms increase the reliability of measurements [24] in safety-critical domains, *e.g.* aviation [21] or self-driving cars [8]. We focus on reconciling numeric data using software voters, with either result selection or amalgamation techniques [24]. Specifically, we consider history-based voting algorithms [23] that weigh values based on the historical performance record of the candidate sensor. History-Based Weighted Average [23] weights the historical data to compute an output value. To improve the granularity of historical records, [14] uses a soft dynamic threshold. In [5], authors apply a hybrid approach using module elimination and dynamic threshold. We further detail these approaches in §3. Our approach, AVOC, extends the hybrid algorithm from [5] with a clustering component to improve performance before there is a long enough historical record, as well as to speed up convergence of the weights to stable values.

Some voting-based data fusion frameworks rely on specific description languages to define algorithmic details [6]. However, these approaches ignore history-based measurements. We also observe that modern voting algorithms are too complex to be represented in such terms. Where [6] defines voting as three-step process (reaching quorum, excluding outliers and calculating results), modern algorithms often include further steps like weighing and updating historical records, or optimising reliability metrics [26]. We argue that a customisable voting framework serves as encapsulation for sensor-fusion applications if built atop state-of-the-art approaches, as shown next, when we present our proposal for a generic definition specification for voting algorithms, VDX. A further benefit of our approach is that it can then be integrated into data-centric tooling, such as ETL (Extract-Transform-Load) tools. Such tools, *e.g.* Singer.io [3] and Node-RED [4] are gaining popularity, but still have no support for voting-related data cleansing methods,

Table 1
Comparison of state-of-the-art voting schemes. MNN: Mean Nearest Neighbour.

Algorithm	Module Elimination	Dynamic Threshold	Output Selection
History-Based Weighted Average (STANDARD - §3.1)	✗	✗	Mean
Module Elimination Weighted Average (ME- §3.2)	✓	✗	Mean
Soft-Dynamic Threshold Weighted Average (SDT- §3.3)	✗	✓	Mean
Hybrid History-Based Weighted Average (HYBRID- §3.4)	✓	✓	MNM

which we argue would be beneficial for IoT data integration with concurrent streams from redundant sensors.

3. History-aware voting algorithms

To combine values from uncalibrated redundant sensors, the history-based averaging algorithm (*i.e.*, the STANDARD algorithm [23]) either chooses a sensor output value or creates an amalgamation of these values. This approach can be optimized by temporarily ignoring values produced by modules with below average historical records. This variant, *i.e.* Module Elimination Weighted Average (ME), assigns zero-weights to the discarded values in the voting until their historical records improve by submitting better values, even if discarded in the voting itself.

The Soft Dynamic Threshold History-Based Weighted Average (SDT) introduces a finer grain definition of agreement, beyond the binary-only definition [14]. Values between 1 and 0 can be assigned if values are not in agreement based on the accepted error threshold, but are in agreement based on a multiple of it. The magnitude of the multiple is defined by a parameter of the algorithm that can be tuned according to the needs of the specific use case.

We further consider Hybrid History-Based Weighted Average (henceforth HYBRID [5]). It combines ME and SDT, while utilising agreement-based and not history-based weights. The HYBRID algorithm allows to choose a winning value rather than assigning the resulting average, using the mean nearest neighbour approach. Table 1 recaps these alternatives and their supported optimizations, which we describe in further details in the reminder of this section. We discuss our findings on the output quality of all algorithms in §7.

3.1. *History-based weighted average* - STANDARD

In the STANDARD variant [23], agreement of two values x_i, x_j of a set of N values is defined when they satisfy the inequality:

$$d_{ij} = |x_i - x_j| < \alpha \tag{1}$$

where α is a pre-selected margin.

When a value x_i is in agreement with at least $(N-1)/2$ other values, then we set $S_i = 1$ otherwise $S_i = 0$. Hence, $S_i = 1$ represents an input in agreement with the majority. When n runs are completed, the historical record of a module is defined as:

$$H_i(n) = \sum_{l=1}^{n} S_i(l) \tag{2}$$

This value is then normalised by n to obtain the state indicator P:

$$P_i(n) = \frac{H_i(n)}{n} \tag{3}$$

During a voting round, each module' weight is based on its state indicator:

$$w_i = P_i^2 \tag{4}$$

Finally, the weighted average is calculated as:

$$x_o = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} \tag{5}$$

3.2. *Module elimination weighted average* - ME

This optimization, described further in [23], eliminates modules (*i.e.* sensors in our case) that perform below the average from the vote, assigning each of them a weight of zero. Specifically, a P_{avg} is calculated as follows:

$$P_{avg} = \frac{\sum_{i=1}^{N} P_i}{N} \tag{6}$$

The weight calculation is modified by setting the weight w_i of a module to zero when it performs below average, *i.e.* its P_i is below P_{avg} as follows:

$$w_i = \begin{cases} 0 & \text{if } P_i < P_{avg} \\ P_i^2 & \text{otherwise} \end{cases} \tag{7}$$

Finally, the average calculation relies on Equation (5).

3.3. *Soft dynamic threshold weighted average* - SDT

The Soft Dynamic Threshold algorithm [14] replaces the margin α with a dynamic variant v_t based on the input x, computed as:

$$v_t = px \tag{8}$$

Where p is a proportional constant, *i.e.* the error margin is a proportion of the input value and not an absolute value for all inputs. The algorithm then replaces the Boolean definition of Si defined above. Rather than two modules i, j with a distance of d_{ij}, the parameter S_{ij} becomes:

$$S_{ij} = \begin{cases} 1 & \text{if } d_{ij} \le v_t \\ (\frac{k}{k-1})(1 - \frac{d_{ij}}{k * v_t}) & \text{if } v_t < d_{ij} < k * v_t \\ 0 & \text{if } d_{ij} \ge k * v_t \end{cases} \tag{9}$$

The parameter k is tunable, and chosen on a use-case basis to determine how permissive the algorithm is to considering values to be in partial agreement.[1]

Then S_i is calculated by normalising:

$$S_i = \frac{\sum_{j=1, j\neq i}^{N} S_{ij}}{N-1} \tag{10}$$

$H_i(n)$ and $P_i(n)$ are calculated as before in Equation (2) and Equation (3).

Hence, SDT computes the weights, where w_i is obtained as:

$$w_i = \begin{cases} 2P_i(n) & \text{if } 0.5 \le S_i \le 1 \\ P_i^2(n) & \text{if } 0 < S_i < 0.5 \\ 0 & \text{if } S_i = 0 \end{cases} \tag{11}$$

Finally, the output is calculated with Equation (5).

[1] In this context, we refer to partial agreement when the value is outside of the agreement threshold v_t but within the tunable and more tolerant $k * v_t$.

Algorithm 1: Soft-Threshold 1-D Clustering.

```
Input  : Values: V, threshold-factor: f
Output: Clusters: C
Create-cluster($C$, [])
for value in V do
    for Cluster in C do
        if $(1-f) \times avg(Cluster) \leq value \leq (1+f) \times avg(Cluster)$ then
            cluster.add(value)
        end
    end
    else
        Create-cluster($C$, [$value$])
    end
end
```

3.4. Hybrid history-based weighted average - Hybrid

The Hybrid variant combines the Module Elimination algorithm and the Soft Dynamic Threshold algorithm. S_{ij} and S_i are computed by Equation (9) and Equation (10) with the change that v_t is once again replaced with α as in the standard algorithm.

Weights are calculated similarly to the Module Elimination algorithm:

$$w_i = \begin{cases} 0 & \text{if } S_i = 0 \text{ or } P_i(n) < P_{avg}(n) \\ \frac{\sum_{i=1}^{N} S_i}{N-1} & \text{otherwise} \end{cases} \tag{12}$$

However, the final output y of the vote is not the average $\bar{y}$ but rather the input x_i closest to it.

The final difference is the calculation of the history, based on the final output y and computed in a similar manner to S_{ij}:

$$h_i(n) = \begin{cases} 1 & \text{if } d_{iy} \leq \alpha \\ (\frac{k}{k-1})(1 - \frac{d_{iy}}{k*\alpha}) & \text{if } \alpha < d_{iy} < k*\alpha \\ 0 & \text{if } d_{iy} \geq k*\alpha \end{cases} \tag{13}$$

The history of the module for n rounds is:

$$H_i(n) = \sum_{i=1}^{N} h_1(n) \tag{14}$$

Finally $P_i(n)$ is calculated as in Equation (3).

4. AVOC: accurate voting with clustering

History-based algorithms typically fall back to standard average (or a similar unweighted approach) on the first round until a historical record is established or when the weights become 0 due to severe issues with the data. Weights can drop to 0 after a series of disagreements, which results in notorious disagreers being rated as untrustworthy by the system applying the algorithm. To counteract these issues, we introduced AVOC [17], which we aim to investigate and evaluate in greater depth in this work. AVOC builds atop the Hybrid algorithm by applying a simplified clustering algorithm during the first round when the weights are all 0. The clustering step eliminates obvious outliers, improving the accuracy of that round compared to mean average, with little performance overhead and faster convergence speed. In this work, we further evaluate how AVOC performs in comparison to the state-of-the-art voting algorithm, and how it can be combined with it to achieve optimal performance in an indoor positioning scenario.

Algorithmic approach.

For the clustering step, we leverage a similar logic to the agreement calculation in voting algorithms: we check for values within a given scaling threshold of each other (which is selected to mirror the parameters of the given algorithm), and group the values in agreement. Then, we select as output value the average (or its closest real value) of the largest group (if we are using the mean nearest-neighbour approach to output selection). Algorithm 1 formalizes this approach. This grouping

```
1  {
2   "algorithm_name": "AVOC",
3   "quorum": "UNTIL",
4   "quorum_percentage": 100,
5   "exclusion": "NONE",
6   "exclusion_threshold": 0,
7   "history": "HYBRID",
8   "params":{
9       "error": 0.05,
10      "soft_threshold": 2
11  },
12  "collation": "MEAN_NEAREST_NEIGHBOR",
13  "bootstrapping": true,
14 }
```

Listing 1: Vote definition sample in VDX JSON format.

logic is similar to DBSCAN [15]; AVOC opts for self-calibration, rather than requiring costly parameters tuning. This is achieved through a majority vote with a soft-dynamic error margin (as the margin depends on a reference value). The clustering step is used for bootstrapping a new set of modules, or as a fallback in cases of issues. To reach this goal, we use the historical record value for each module, and declare that the clustering approach should be used when all records are 1 (indicating a new set) or 0 (indicating a failure of the system or an extreme data spike). Fig. 2 depicts this logic.

Generalisation. Generalising this approach for multi-dimensional data, an unsupervised clustering algorithm can be used such as Mean-shift [13] or X-Means [32]. The logic of choosing an output value would be similar. However, in such scenarios, choosing a single output vector for multiple dimensions is non-trivial as the complexity of data and correlation of errors considerably increases. To mitigate, the voting approach can be applied for each dimension separately, leaving other data fusion techniques to process the multi-dimensional results. In AVOC, we follow the approach of voting on each dimension separately, without incorporating the clustering itself.

5. VDX: voting definition specification

To enable reliable implementations and improve the usability of software voters, we contribute a new voting specification scheme and parsing logic. The scheme can define any of the algorithms described above, as well as simpler ones without history.

Software-defined voting schemes exist, *e.g.* Voting Definition Language (VDL) [6]. Those predate more complex history-based voting approaches, and extending VDL for finer-grain algorithmic definitions is challenging. However, our specification VDX supports the relevant parameters of VDL, enabling our definition to describe a superset of VDL-scoped algorithms. The full schema, as well as a sample implementation and usage examples can be found at: https://doi.org/10.5281/zenodo.8069916. The repository also includes an interactive application that allows users to compare the algorithms presented with the state of the art (Fig. 4) as well as experimental extensions still under development (e.g. for handling multi-dimensional data as mentioned in §4).

Capabilities. Listing 1 shows our AVOC algorithm using this scheme as example. VDX allows the specification of several parameters, including quorum (*i.e.*, how many candidates need to submit values for a vote to be triggered), exclusions (to automatically prune outliers) collation techniques similar to VDL, *e.g.* “mean nearest neighbour” (Listing 1, line 12). It extends VDL by allowing the selection of a history algorithm (Listing 1, line 7), additional parameters (Listing 1, lines 8-11), and whether to enable clustering algorithm as a bootstrap/-fallback mechanism (Listing 1, line 13). Another extension VDX adds over VDL is the ability to vote on categorical *i.e.* non-numeric values, such as character strings and JSON blobs. In such cases however, several features are disabled. Value-based exclusion cannot be applied, as there can be no mean or standard deviation calculation. The “standard” and “module-elimination” algorithms for deriving module history are available, however the “hybrid” algorithm is not, as the fine-grained

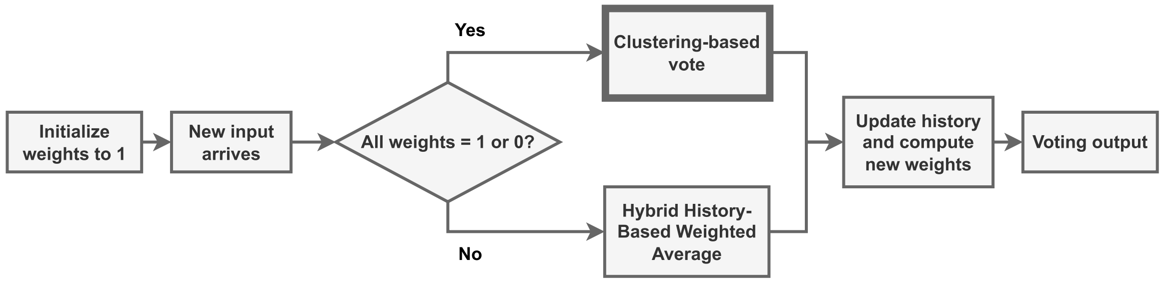


Fig. 2. Flowchart of our extension to the state of the art voting algorithm, resulting in AVOC. The clustering version of the vote is triggered when weights are all 1 or 0, indicating a freshly initialized system or a large amount of flaws that made the weights unusable. After the clustering vote concludes the weights are calculated as in the unmodified algorithm, but using the output of our custom vote, therefore reducing the transient errors.

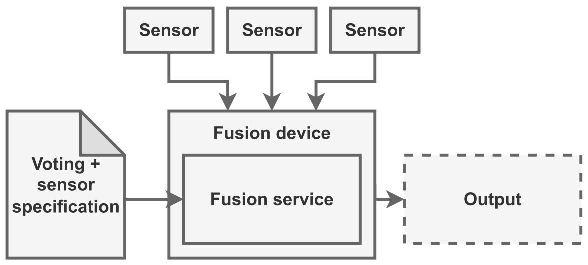


Fig. 3. Overview of the voting specification usage. The same fusion service can be applied to a variety of setups (*e.g.* edge/fog nodes, robots, IoT devices *etc.*) and a custom definition can be supplied during setup to select the appropriate behaviour.

agreement definition cannot be applied to non-numeric values. Finally, clustering-based bootstrapping cannot be applied to categorical values and the only collation method is the weighted majority vote. Software voting implementers may re-introduce some of these features by supplying a custom distance metric for categorical values.

Fig. 3 depicts the proposed workflow leveraging the voting specification. The voting format description is submitted to the voting system by the user to initialize the system. Sensor reading can then be submitted to the same system, directly to the exposed sensor reading fusion service. These will trigger voting rounds. The system needs to maintain a record of the success rate of each sensor (defined as P_i in Section 3) and the parameters each voter group uses. In principle, it is not required to retain the sensor readings from previous submission rounds (though some algorithms require it to retain the last output value to be used in the next round).

Limitations and assumptions. VDX currently cannot define algorithms that use parameters for the candidate values, *e.g.* MLV [26], or genetic voting algorithms [35], but assists already by introducing voting into further IoT software for reliable input data and analytics. It should also be noted that VDX itself has no security features that protect against malicious actors, so this is left up to the client code to implement as needed.

6. Multi-sensor application scenarios

We devise three use-case scenarios to validate our approach. The scenarios represent sensor-reliant deployments where redundancy proves valuable. Specifically, we draw on the smart building and self-driving vehicle domains, both showcasing cyber-physical systems relying on sensor measurements to control other critical systems.

Fig. 5 depicts our first use-case, a sunlight detection system in a hypothetical smart building. Such a setup could be used to control automated window blinds, for example. This example was selected to showcase a scenario with redundant sensors that are expected to produce identical results. The hub is connected to a sink node to record 10'000 rounds of concurrent measurements from 5 sensors, polling at 8 samples/s, to create a reference dataset representing 1250 seconds of data collection. Each round produces 5 float values per sensor. The reference dataset consists of the raw readings from all sensors and is used to compare all voting algorithms on the same set of values. We built a portable demonstrator that executes them and our proposed voting algorithm, AVOC (detailed in §4), leveraging a Raspberry Pi 4B unit (see Fig. 6). It uses a Phidget Wifi hub [40] (VINT) connected to a set of 5 LUX1000 Light Phidget sensors [1]. Input, weights and results are shown on an LCD screen [19] connected to the hub. The portable version let us confirm the feasibility to execute on constrained hardware, combining both redundant measurements and voting.

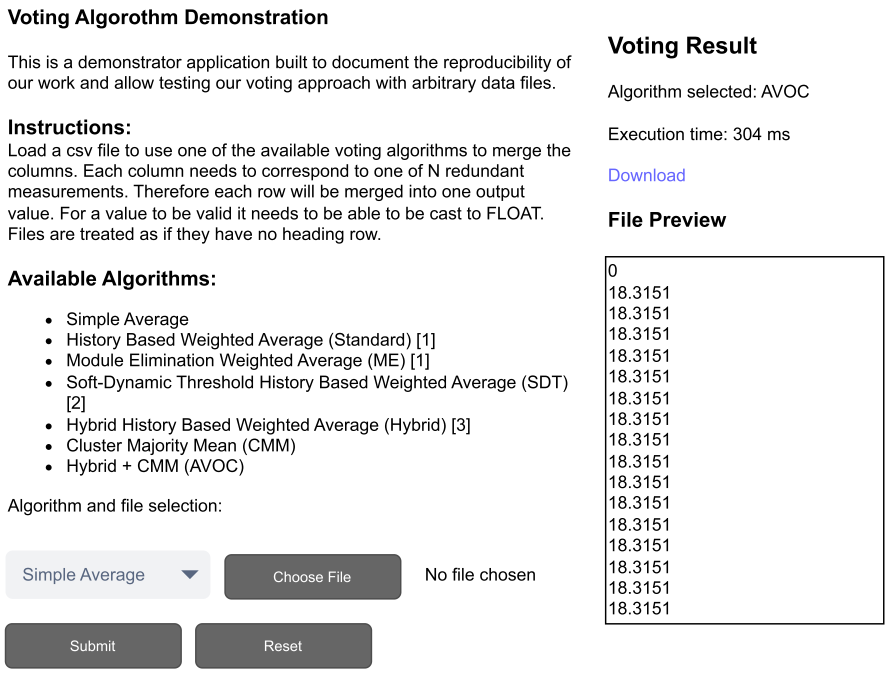
In the second use-case, we mimic a typical smart-city/self-driving vehicle application that relies on indoor-positioning to track the position of a moving unit (*e.g.*, a robot). We simulate the operations to track a cargo vehicle traversing a tunnel, as shown in Fig. 7. Such vehicles use Bluetooth (BLE [2]) beacons as *milestones* to locate the position of the truck, as done in emerging country-wide systems (*i.e.* CST – Cargo Sous Terrain [28,11]). We deploy two stacks of nine redundant beacons and a cargo vehicle using a Lego Mindstorms EV3 [29] robot. As the Bluetooth receiver on the robot was incompatible with the beacons due to lack of BLE support, we installed a laptop on top of it acting as pragmatic substitute receiver with edge processing capabilities, and without affecting the generality of the findings other than affecting the speed of the robot (which has no effect on what we are measuring). Fig. 8 shows the prototype.

The robot drives slowly in a straight line with no line-of-sight obstacles from one beacon stack to the other, across a distance of 15 meters. The speed of the robot was set to 7% of its specified top speed (0.09 m/s). We collected as many data points as possible along the route, resulting in 297 measurements per beacon, noting that autonomous cargo systems like CST proceed at around 8.3 m/s, thus having 99% less measurement samples available for voting. We elected not to apply filtering and to vote on the raw values in this experiment, to evaluate the performance of our voting system in the presence of the typical unpredictability of BLE singal strength measurements.

The third use-case is a positioning experiment in a real-world environment, as shown in Fig. 9. The experiment context is meant to emulate a smart shopping scenario, recreated in our office building. Here, we have placed 4 stacks of 5 beacons in a room of dimensions 7.18 m x 1.98 m. Each stack represents a store shelf in a hypothetical store. A user with a bluetooth-enabled device walks around the room to collect measurements. This scenario allows us to compare our approach to other sensor fusion methods for indoor localisation, namely Kalman filtering.

7. Evaluation

This section presents the experimental evaluation of AVOC and the VDX system in general. With VDX, we fully implemented the three use-case scenarios from §6, and compare the error-correction performance of the various voting schemes. More concretely, we use our light sensor



Voting Algorothm Demonstration

This is a demonstrator application built to document the reproducibility of our work and allow testing our voting approach with arbitrary data files.

Instructions:
Load a csv file to use one of the available voting algorithms to merge the columns. Each column needs to correspond to one of N redundant measurements. Therefore each row will be merged into one output value. For a value to be valid it needs to be able to be cast to FLOAT. Files are treated as if they have no heading row.

Available Algorithms:

- Simple Average
- History Based Weighted Average (Standard) [1]
- Module Elimination Weighted Average (ME) [1]
- Soft-Dynamic Threshold History Based Weighted Average (SDT) [2]
- Hybrid History Based Weighted Average (Hybrid) [3]
- Cluster Majority Mean (CMM)
- Hybrid + CMM (AVOC)

Algorithm and file selection:

Voting Result

Algorithm selected: AVOC

Execution time: 304 ms

File Preview

```
0
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
18.3151
```

Fig. 4. Algorithm comparison application.

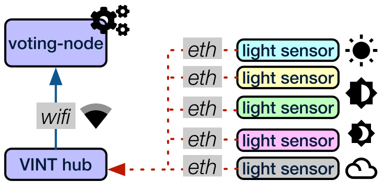


Fig. 5. Light sensor use-case: the sensors are wired via ethernet to a hub, streaming data via WiFi to the voting sink-node.

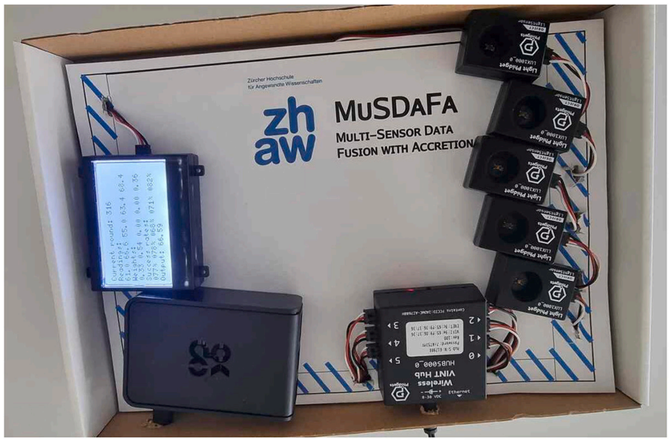


Fig. 6. Portable 'shoe-box' testbed for our light sensor setup (Fig. 5). A Raspberry Pi 4 runs the fusion script. An LCD display shows the voting results and weight values.

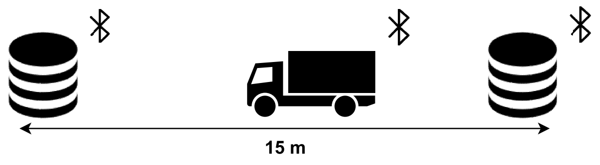


Fig. 7. BLE beacon use-case. 2 stacks of beacons 15 meters apart with a robot driving between them, taking signal strength measurements along the way.

Table 2
Notation used in our evaluation.

Symbol	Description
E1-E5	Light sensor labels
Lumen (lm)	Unit of measurement of light quantity
avg	Distance-weighted mean
strd	Standard History-based weighted average
ME	Module elimination history-based weighted average
Hybrid	Hybrid history-based weighted average
Clustering	Clustering-only voting
AVOC	Accurate Voting with Clustering (our method)
RSSI	Received Signal Strength Indicator
LSML	Least Squares Multilateration

experiment as a model for sensor-based time-series data, which can be voted upon using history-based algorithms. We use the BLE beacon experiment to test various algorithms on their ability to function on low quality data. Finally, we use the localisation experiment to compare our voting approach to Kalman filtering, a state-of-the-art approach for sensor fusion, as well as test how we can effectively combine the two approaches to improve accuracy and performance. Our evaluation answers the following questions: *(Q1)* Can AVOC improve the output quality of our use-case scenarios? *(Q2)* Can it mitigate injected errors and reach the same output? *(Q3)* Which algorithm fits which scenario better, and *(Q4)* How can we leverage VDX to customise the voting behaviour for each scenario?

Due to the variety of experiments, we provide Table 2, to ease understanding of our notation in this section.

Implementation details. We implemented AVOC and the approaches from §3 in Python 3.9, for a total of 490 LOC. We note that though the evaluation was done with pre-recorded data for reproducibility purposes, the system can execute a history-aware voting round in 1 millisecond and a stateless vote in 50 microseconds (datastore reads and writes being the bottleneck). Thus, the system can operate under soft real-time constraints, as in the case of our "shoebox" demonstrator in Fig. 6 as well as many critical cyber-physical applications in practice.

UC-1: Light sensors. We used the 10'000 value dataset recorded using our light sensor setup (§6) to gather the raw data and reference

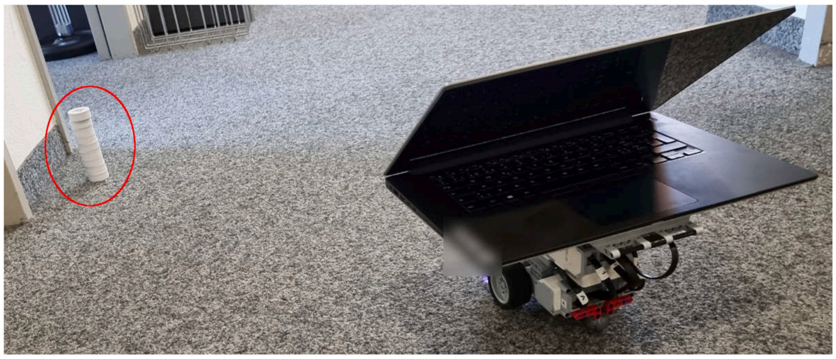


Fig. 8. Robot driving to the circled beacon stack destination. The laptop acts as bluetooth receiver and edge voter.

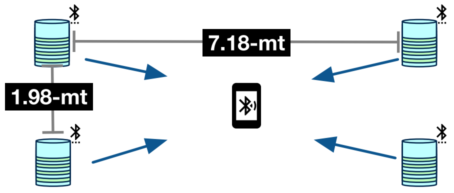


Fig. 9. Indoor positioning experiment setup. 4 stacks of beacons are deployed in the 4 "store shelfs". The user device is placed in the middle of the room and the user walks around the room to collect measurements.

values for the baseline algorithms (Fig. 10-a). The dataset contains light quantity in Lumen, measured by our 5 light sensors, labelled E1 to E5.[2] Then, we injected an artificial outlier sensor, by adding +6'000 lumen to sensor E4 and repeated the voting experiment. We compare the performance of the different algorithms according to the following metrics: *(a)* voting rounds required to converge back to the baseline, and by extension how quickly outliers are eliminated; and *(b)* how far the new stable value is from the original.

We make a first comparison using the raw reference data. In this scenario (Fig. 10-b), all 6 variants performed equally well, with outputs matching almost completely. We present the 6 different horizontal plots, with the light quantity in Lumen on the Y axis, separately, as they would overlap if plotted together. The error injection case (shown in Fig. 10-c) exhibits some interesting facts. First, the Standard algorithm exhibits high initial skew, as can be seen on the left side of Fig. 10-e, which is then slowly mitigated as the *faulty* sensor (E4) is deemphasised in the later part of the experiment, as shown on the right side of Fig. 10-e. However, even after 10'000 voting rounds of voting (*e.g.*, 20 minutes in our experiments), the skew is not eliminated completely, as can be seen at the end of Fig. 10-e. This is where the module elimination feature of Me is beneficial, as the faulty sensor is quickly eliminated in round 2, as performing below average compared to the rest. However, as seen in Fig. 10-c, the gap created by skewing E4 indicates how E3 is also now tagged as outlier, and its result is skewed upwards by 200 lm.

The Hybrid algorithm also uses a granular definition of agreement score, but combines it with the aggressive elimination of modules from Me. For our experiment, this is the *best of both worlds* result (also shown by the differentials in Fig. 10-e) where, minus few spikes, the value is identical to the pre-error output.

For completeness, we show clustering-based voting on its own without combining it with Hybrid (which remains ideal for this scenario). This can be seen in Fig. 10-e. We observe similar behaviour to Me, with E4 being excluded from the output immediately. Differently from Me, E4 was also excluded from the first round. In contrast, E3 was not always excluded, due to the lack of history-based elimination, indicating higher variations in the output. Regarding clustering-only voting (COV), it significantly outperforms other stateless approach, *i.e.* weighted average without history. This result indicates that the COV approach fits well scenarios where maintaining historical result records is impractical: short-lived sensor measurements, one-time comparisons of datasets, *etc.*

One consistent observation in the error injection experiment is that history-based algorithms experience a spike on startup, as the artificially modified value is skewing the output but is not yet mitigated by the history. This is the phase where in principle the clustering step detailed above has higher chances to affect the results. Indeed, although the clustering algorithm alone is not as accurate as Me or Hybrid, it overcomes the initial data spikes. Thus, a system capable of fallback to it when history is not available or suspected unreliable, can benefit from its inclusion.

Next, we run AVOC, which concretely combines the clustering step with Hybrid. We observe how the initial spike is quickly pruned (Fig. 10-f) within the initial rounds. The bootstrap boost can also be noticed: due to the better history adjustment in round 1, the voter already *learns* to exclude E3 from round 2, returning to its pre-error output almost instantly, despite the clustering is only used once. As AVOC converges within 200 ms, the experiment confirms its utility for fast accurate voting and provides a positive response to our questions (Q1) and (Q2), as we improve the reliability of the output even in the presence of the injected errors.

UC-2: BLE beacons. We leverage this second use-case with BLE beacons and a Lego Mindstorms EV3 robot to study a scenario with more anomalies and faults. We set up two stacks of 9 beacons each 15 meters apart in an indoor corridor with no obstacles. The robot drives at 7% of its full speed in a straight line, from one stack to the other. The robot takes continuous RSSI (Received Signal Strength Indicator) measurements for each beacon along the way. This experiment examines if the high redundancy and voting actually are beneficial. The scenario simulates the operations to locate a vehicle traversing a tunnel using beacon stacks as *milestones* along the way, determining the closest stack to the vehicle. The state of the art in leveraging RSSI for positioning relies on filtering [42] or collaborative positioning [41] to improve stability and output quality. However, since the scope of our experiment is to evaluate the effects of voting on sensor measurements, we kept the raw RSSI values from the sensors, to keep the values as close to the original measurement as possible for the voting, before applying other techniques to improve positioning performance. The resulting data, which we plan to publicly release, lacks several values as well as mismatched readings in each stack, providing a more challenging fusion scenario. Such missing values allowed us to identify several fault scenarios, which we describe next.

Fault scenario: missing values. Due to some beacons not being reachable from the BLE receiver (*i.e.*, the laptop in our experiment). Missing values can reduce the reliability of the output measurement, since fewer

[2] The lumen (symbol: lm) is the unit of luminous flux, a measure of the total quantity of visible light emitted by a source per unit of time, in the International System of Units (SI). [7].

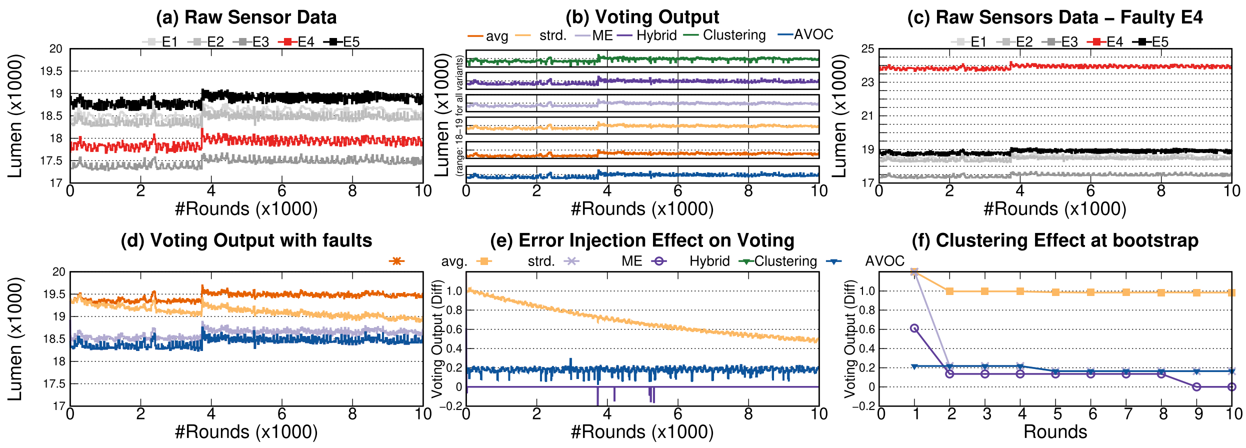


Fig. 10. Comparison between our voting approach AVOC and the state-of-the-art approaches. Our 5 sensors, labelled E1 to E5, measure light quantity in Lumen. (a) Reference data, captured by our light sensor setup for 20 min. (b) voting output using AVOC on the reference data. (c) Reference data with injected errors (1 faulty sensor). (d) Output of HYBRID, Clustering and AVOC under these errors. (e) Output difference between voting on the raw values and voting on the error-injected values. (f) Zoom on the first 10 rounds. All Y-axis units are in 1000 s of Lumen.

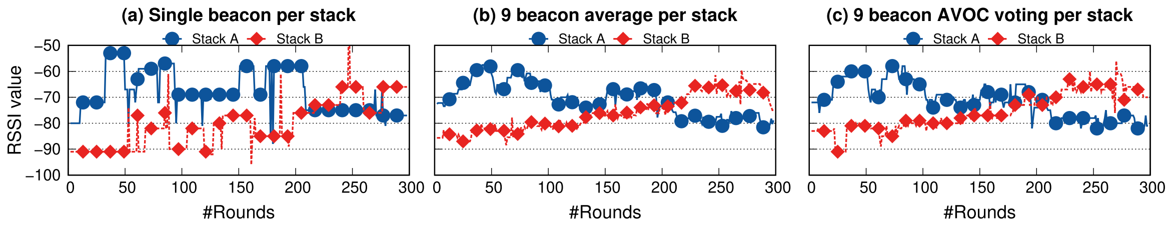


Fig. 11. Results of the BLE beacon experiment. We measure the signal strength from each of 2 stacks of BLE beacons as the robot moves from Stack A to Stack B. (a) shows the RSSI output when only one beacon from each stack is used. (b) shows the average RSSI value of all 9 beacons in the stack for each round. (c) shows the AVOC voting output for all 9 beacons in the stack. Averaging provides visibly less ambiguity in determining which stack is closest to the robot, when compared with the mean nearest neighbour selection used for HYBRID.

candidate values are being considered, and potentially prevent from reaching a consensus value if most or all values are missing. A small amount of missing values, *i.e.* less than the majority, does not prevent the system from converging to a common result, though it reduces the redundancy as well as the number of candidates considered, and as consequence the trustworthiness of the outcome. If the majority or all values are missing, the result would no longer be trustworthy, and the system should either revert to the last accepted result, or raise an error. Obviously, these failure scenarios should be accounted for when the voting behaviour is defined. Due to the complexity possible and the variability by scenario, these behaviours are currently not modelled by VDX itself, and are instead left up to the client code to define. In our test-bed implementation, the error handling procedure was programmed into the implementation of each algorithm. It is possible to make this behaviour customisable, including the custom error-handling in the voting parameters of the schema definition.

Fault scenario: conflicting results. In case of conflicts, a majority agreement on outputs using automated voting is less likely to be reached. It is possible that a relative majority agrees on an output, but they are an overall minority, and no absolute majority exists. Especially in systems with small number of votes, ties might occur more easily and tie-breaking mechanisms kick in, such as proximity to the previous output. These fault scenarios clearly show that **setting the constraints of a voting system is non-trivial, and that voting algorithm implementations in a generic data fusion platform should be parametric**. In addition, there should be a voting specification declared for the target application, to take the desired error-handling behaviour into account. Accounting for such fault scenarios allows to implement more robust versions of the algorithms. It is also possible to extend VDX in a future revision to support high-level descriptions of the desired fault handling policy. Examples of such policies include rejecting a round of measurements if there is no majority quorum or majority agreement.

We then tested our voting algorithms on the BLE experiment data. We used the same recorded values for each algorithm. We run voting between the 9 sensors of each stack to create 2 output values per round (*i.e.* 1 per stack). We observed that the method for computing the history of each sensor has no effect. The output of all history-based algorithms overlaps completely. (They are not all plotted due to space constraints.) This is because the chaotic nature of the measurements meant the history values were all very low, as there were few agreements between the sensors. We observe however that the value collation method has impact, *e.g.*, averaging the weighted values, a mean-nearest-neighbour selection, *etc.* This created 2 algorithm groups, those averaging and those choosing the mean-nearest neighbour value, with every algorithm in each group performing identically to each other. In order to determine the best results, we study the number of rounds while it is ambiguous which stack of sensors is closest to the robot at any given time. Fig. 11 presents these results.

Fig. 11-a is the reference: if each stack only had one sensor, it is not possible to identify the closest stack to the robot for most of the duration of the experiment. Simply averaging the values of the 9 sensors (Fig. 11-b) produces a less ambiguous result. We present the results using AVOC in Fig. 11-c. In spite of the method used to create a historical record for each sensor, what had the most impact on the output was whether or not the last step was to average the values or to select a value (with averaging being the better option in our experiment). In scenarios with high degree of noisy data, such as the BLE one, relying on historical record has no practical value. This confirms our initial assumption that **there is no optimal voting method for all applications**, despite common elements shared by our scenarios (*e.g.*, having to reconcile numerical values from a group of sensors). Thus we conclude

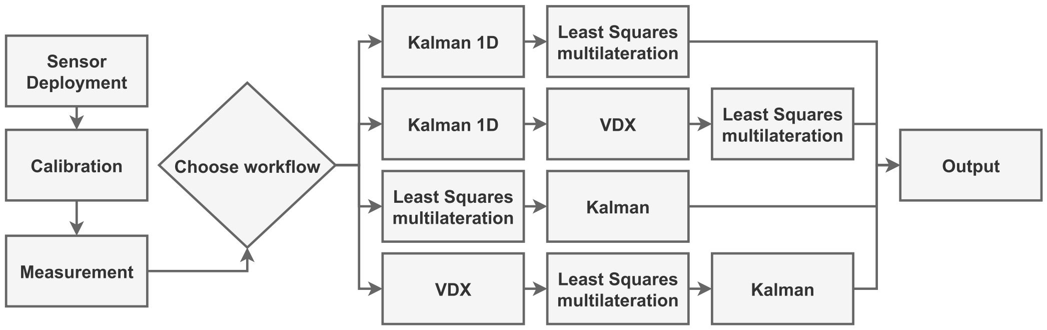

Sensor Deployment
Calibration
Measurement
Choose workflow
Kalman 1D
Least Squares multilateration
Kalman 1D
VDX
Least Squares multilateration
Least Squares multilateration
Kalman
VDX
Least Squares multilateration
Kalman
Output

Fig. 12. Overview of the 4 workflows we used for data cleaning. 2 workflows are from the state of the art, and employ different versions of the Kalman filter to clean the data. The 2 other workflows use voting between the 5 sensors in each stack rather than treat each sensor independently.

that the answer to (Q3) depends of the specifics of the use case, and that the customisation offered by our specification allows us to address (Q4).

UC-3: Indoor positioning.

Our last experiment utilized the same BLE beacons as the previous one, but this time in a standard indoor positioning scenario. The scenario was based on a smart shopping context, in which we placed 4 "store shelves" in the corners of a rectangular area in our lab, and stacked 5 beacons on each shelf (to simulate the 5 levels of a supermarket shelf). The dimensions of the experiment area are 7.18 m x 1.98 m.

To calculate the position using BLE beacons, we used the following formula for range estimation, based on the path loss model [33]

$$d = 10^{\frac{rssi_0 - rssi}{10\times n}} \tag{15}$$

where $rssi_0$ is the RSSI value at 1 m, $rssi$ is the RSSI value measured by the receiver, and n is the path loss exponent. As the beacons used do not advertise their TX power (a pre-calculated value for the RSSI at 1 m), we needed to measure the RSSI at 1 m for each beacon, as well as the path loss exponent. To obtain the path loss exponent, we measured the RSSI at 2 m, and solved for n using Eq. (15).

To calculate the position, we used the ranges obtained from the beacons, and the position of the beacons, to define circles which we used to solve the multilateraion problem using the least squares method. The system of circle equations is:

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2 \tag{16}$$

where x_i and y_i are the coordinates of the beacon, and d_i is the range to the beacon. The solution to this system of equations is the position of the user. We used the `scipy.optimize.least_squares` function in Python to solve the system of equations. This uses the Levenberg-Marquardt algorithm [30] to solve the non-linear least squares problem.

These two parts of the position estimation remained the same for all the algorithms we tested. We then recorded RSSI values from the beacons at different known locations in the experiment area, and used these values to test the algorithms. As before, we used the pre-recorded data to ensure all 4 algorithms were tested on the same inputs.

The 4 workflows we used are presented in Fig. 12. All methods used variations of the Kalman filter [38] for smoothing, according to the state of the art. First, we used the 1D Kalman filter to clean the data, and then used the Least Squares Multilateration to derive the output. This means we applied the filter directly to each individual range estimation, and then used the filtered values to calculate the position. This method of data cleaning is similar to the one used in [10].

The second workflow adds a voting step between the filter and the multilateraion. As with the tunnel experiment, we used a simple averaging voter here, as the data is limited in history and also quite noisy.

The third workflow uses the 2D Kalman filter after the multilateration. This means we applied the filter to calculated position to smooth it. This method of data cleaning is similar to the one used in [44].

Finally, the fourth workflow adds a voting step before the multilateration to the third one. The raw range data was voted on, before the multilateration was calculated.

A visual representation of the two Kalman filter workflows is shown in Fig. 13a and Fig. 13b. The aggregated results of the 4 workflows are presented in Table 3. The table presents the results per measured user location, averaged across all positions used in the experiment. In each position we took 3 measurements, 5 seconds apart, from each of the 20 beacons (5 beacons per stack, 4 stacks). Here we can see that voting improved the result of both methods. Perhaps surprisingly, the 2D Kalman filter did not improve the result, and in fact made it worse. This can be explained by the fact that the measurements were only taken from static positions, and not from a moving user. This means that the 2D Kalman filter was not able to take advantage of the temporal information and velocity.

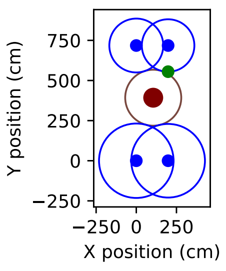
The performance results from the same table are also of interest. We observe that even though the 2D Kalman filter takes more time to compute, the difference is negligible compared to the total runtime. Voting reduces the amount of time that the Least Squares method takes to compute, which is always the most time-consuming step.

To further test the effect of voting on performance, we artificially increased the number of beacon groups by duplicating the real measurements. As expected the filtering time for the Kalman 1D filter scaled linearly, as the number of measurements in each filtering step remained the same, and only the number of steps scaled with the number of beacons. Each filtering step takes approximately 1.33 μs (assuming again 3 measurements per beacon).

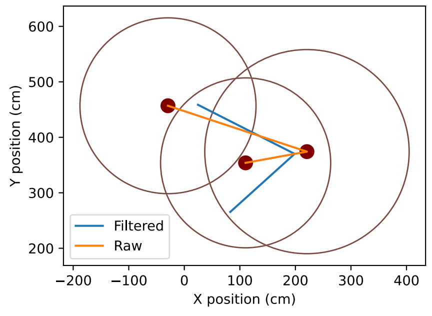
Similar behaviour was observed for the voting with each voting round between 5 values taking approximately 1.5 μs. It should be noted here that enabling the history aware features of VDX would increase the time of a voting round depending on the implemented storage method for the history. In that case, reading and writing the histories into a backend component would take the majority of the voting time, depending on the efficiency of the implementation.

Increasing the number of sensors does not increase the time the 2D Kalman filter takes to compute, as it is applied to the final positioning result, so only 2D coordinates are applied as a measurement. Therefore only the amount of measurements per sensor increasing would affect it, but that would require a prolonged measurement period, which would not fit the indoor navigation scenario.

The Least Squares Multilateration seemingly scales differently in the real experiments, though when inflating the number of sensors we observed that for very high sensor counts (we tested between 100 and 20000 circles) the time is quite consistently linear. The average time divided by the number of circles was 400 μs per circle, above 5000 circles, as below that the overhead still makes the duration less predictable. It is consistently the most time consuming part of the workflow, so voting, which reduces the amount of circles it needs to solve, is a good way to improve performance (provided the scenario allows the sensors to be grouped).



(a) 1D Kalman filter



(b) 2D Kalman filter

Fig. 13. Sample visual output of the localisation program using the 2 different Kalman filters. (a) 1D Kalman filter. Here the filter is applied to each range measurement individually and multilateration is performed at the end. The 4 blue circles represent the raw range measurements. The red circle represents the filtered position estimate. The green dot represents the true position of the user. (b) 2D Kalman filter. Here the filter is applied to the position estimate after multilateration. The 3 red circles represent the raw position estimates. The blue line represents the filtered position estimate.

Table 3
Execution breakdown of the 4 data cleaning methodologies in the indoor positioning experiment.

Method	Filter calculations	Filter time	LSML calculations	LSML time	Voting rounds	Voting time	Mean Positioning Error	Time
Kalman 1D	20 x 3 measurements	80 µs	1 x 20 circles	20 ms	-	-	156.75 cm	502 ms
Kalman	1 x 3 measurements	236 µs	3 x 20 circles	44 ms	-	-	208.25 cm	567 ms
Kalman 1D + Voting	20 x 3 measurements	80 µs	1 x 4 circles	12 ms	4 x 5 values	6 µs	145.75 cm	496 ms
Kalman + Voting	1 x 3 measurements	319 µs	3 x 4 circles	31 ms	12 x 5 values	18 µs	159.25 cm	557 ms

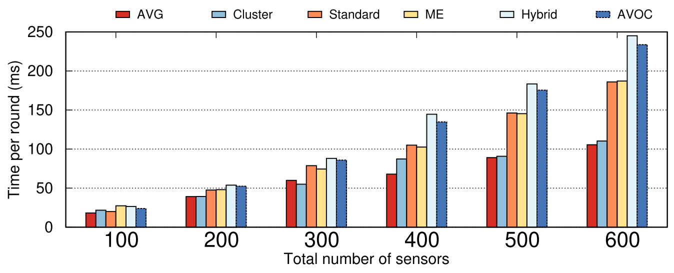


Fig. 14. Scaling experiment with 100-600 simulated light sensors. We compare the 6 algorithms from our light sensor experiment on the time taken to complete each voting round.

Scalability evaluation.

We conducted a scaling experiment by creating simulated sensors. Each simulated sensor in the dataset uses a randomly selected duplicate of the data of one of our light sensors from the first experiment. We varied the number of simulated sensors between 100 and 600 and compared the time per voting round of the six algorithms compared in the first experiment. As before, all algorithms were implemented using our VDX system. (See Fig. 14.)

We observe that the 2 non-history algorithms exhibit similar performance, further strengthening the Clustering algorithm in scenarios without history, as it does not degrade performance and, as shown before, significantly increases accuracy. The two versions of History-Based Weighted Average, Srtd and ME, perform similarly, as expected. Hybrid and AVOC also perform similarly, as they use the same history derivation method. However, it can be seen that AVOC is consistently slightly faster than Hybrid, due to using the clustering step and skipping the history-based calculation when the weights are equal.

It is notable that up to 300 sensors, which we believe to be a high number of redundant sensors for any sensor fusion setup, AVOC is faster than the 100 ms refresh time of our sensors, indicating that it can be used in real time.

8. Findings and discussion

Evaluating AVOC and VDX yielded some interesting results. First, the light sensor experiment to evaluate the time-series performance of history-based voting. We observe that both the clustering component of AVOC and the history component from the Hybrid algorithm (that is part of AVOC) is important is removing faulty modules and improving the accuracy of results. It is also an important finding that clustering alone is a better voter than distance-based weighted averaging, the typical method for reconciling numeric data which was used as one of the points of comparison in the experiment. This is because clustering is able to remove faulty modules, while distance-based weighted averaging is not. This is useful in scenarios where history is not utilized, due to the system being short-lived or memory constraints. The extra computation in any case is not significant, as shown above, due to the small amount of data involved in each round of voting, even for a large number of modules.

The first BLE experiment showed that noisy and unstable data is unsuitable for clustering-based methods or history-based methods. In such examples averaging works best, though as we examined in the third experiment, the correct approach would be to filter the data or use other smoothing algorithms first. Our latest positioning experiment presents both a comparison against the state of the art, as well as an approach for combining our methods with the state of the art in data fusion. We observe that our approach can speed up the more inefficient part of the process, which is optimising the multilateration, as well as slightly improving the accuracy of the results. We argue that with enough ease of deployment, this method can be applied in real world scenarios without much effort, and can be used to improve the accuracy of existing systems, with only software changes. This ease of deployment is exactly why we developed VDX, which was used in all our experiments to implement the different voting methods used.

With our simulated scaling experiment we also showed that our AVOC algorithm is usable in real-time systems, even when hundreds of sensors need to be voted on.

9. Conclusion and future work

We have conducted an experimental study of history-aware voting in IoT and smart city/smart building applications. We demonstrated the performance of different algorithms on three different scenarios with different needs in terms of sensor fusion and presented out findings in terms of selecting an optimal algorithm for each scenario. Our findings show that inherently reliable systems can benefit more from history-aware voting as it can more easily root out more nuanced quality issues. On the other end of the spectrum, inherently unstable setups benefit more from smoothing and averaging techniques due to the unpredictability of the values rendering historical records ineffective. We also presented AVOC, our clustering-based voting approach, and demonstrated its effectiveness in bootstrapping a new group of sensors in a voting system. We showed how it can improve the accuracy of the voting result in the early rounds by eliminating outliers in-place, rather than discovering them based on past performance. We then proposed voting definition format VDX that can be used to describe a voting procedure to a compatible voter service running on an edge node. We further consolidated our work by conducting an indoor positioning experiment and both comparing our voting-based solution with the state-of-the-art approach as well as combining both approaches to find an optimal solution. We discuss the benefits of the combined approach for maintaining performance when increasing the number of sensors. We plan to explore deeper and more realistic scenarios and the applicability of our voting methodology to providing improved data quality, as well as further develop the voting specification in order to field test a voter service prototype with a variety of compute-power-restricted setups.

CRediT authorship contribution statement

Panagiotis Gkikopoulos: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing, Funding acquisition. **Peter Kropf:** Supervision, Writing – original draft, Writing – review & editing. **Valerio Schiavoni:** Supervision, Visualization, Writing – original draft, Writing – review & editing. **Josef Spillner:** Supervision, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Panagiotis Gkikopoulos reports financial support was provided by Innosuisse - Swiss Innovation Agency, grant number: 63912.1 INNO-ICT.

References

[1] Light Phidget, https://www.phidgets.com/?tier=3&catid=8&pcid=6&prodid=707, jun 2022.

[2] Bluetooth Technology Overview, https://www.bluetooth.com/learn-about-bluetooth/tech-overview/, jun 2022.

[3] Singer.io, https://www.singer.io/, jun 2023.

[4] Node-RED, https://nodered.org/, jun 2023.

[5] A. Alahmadi, B. Soh, A hybrid history based weighted voting algorithm for ultra-critical systems, in: 2012 International Symposium on Communications and Information Technologies (ISCIT), 2012, pp. 1122–1127.

[6] D. Bakken, Z. Zhan, C. Jones, D. Karr, Middleware support for voting and data fusion, in: 2001 International Conference on Dependable Systems and Networks, 2001, pp. 453–462.

[7] BIPM, Le Système international d'unités / The International System of Units ('The SI Brochure'), ninth edition, Bureau international des poids et mesures, 2019, http://www.bipm.org/en/si/si_brochure/.

[8] M.R. Boukhari, A. Chaibet, M. Boukhnifer, S. Glaser, Voting algorithm approach for autonomous vehicle safe driving, in: 2018 IEEE International Conference on Industrial Technology (ICIT), 2018, pp. 327–332.

[9] C. Candrian, A. Scherer, Rise of the machines: delegating decisions to autonomous AI, Comput. Hum. Behav. 134 (2022) 107308, https://doi.org/10.1016/j.chb.2022.107308, https://www.sciencedirect.com/science/article/pii/S0747563222001303.

[10] V. Cantón Paterna, A. Calveras Augé, J. Paradells Aspas, M.A. Pérez, Bullones, a bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and Kalman filtering, Sensors 17 (12) (2017), https://doi.org/10.3390/s17122927, https://www.mdpi.com/1424-8220/17/12/2927.

[11] Cargo Sous Terrain, https://www.cst.ch/, jun 2022.

[12] Z. Chen, X. Feng, S. Zhang, Emotion detection and face recognition of drivers in autonomous vehicles in IoT platform, Image Vis. Comput. 128 (2022) 104569, https://doi.org/10.1016/j.imavis.2022.104569, https://www.sciencedirect.com/science/article/pii/S0262885622001986.

[13] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 603–619, https://doi.org/10.1109/34.1000236.

[14] M. Das, S. Bhattacharya, A modified history based weighted average voting with soft-dynamic threshold, in: 2010 International Conference on Advances in Computer Engineering, 2010, pp. 217–222.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, AAAI Press, 1996, pp. 226–231.

[16] Y. Fang, Z. Shan, W. Wang, Modeling and key technologies of a data-driven smart city system, IEEE Access 9 (2021) 91244–91258, https://doi.org/10.1109/ACCESS.2021.3091716.

[17] P. Gkikopoulos, P.G. Kropf, V. Schiavoni, J. Spillner, AVOC: history-aware data fusion for reliable IoT analytics, in: K. Zhang, A. Gherbi, P. Bellavista (Eds.), Proceedings of the 23rd International Middleware Conference: Industrial Track, Middleware 2022, Quebec, Quebec City, Canada, November 7-11, 2022, ACM, 2022, pp. 1–7.

[18] P. Gkikopoulos, V. Schiavoni, J. Spillner, Decentralised data quality control in ground truth production for autonomic decisions, IEEE Trans. Parallel Distrib. Syst. 33 (10) (2022) 2416–2427, https://doi.org/10.1109/TPDS.2022.3142967.

[19] Graphic LCD Phidget, https://www.phidgets.com/?tier=3&catid=48&pcid=41&prodid=963, jun 2022.

[20] S. Jeong, S. Kim, J. Kim, City data hub: implementation of standard-based smart city data platform for interoperability, Sensors 20 (23) (2020), https://doi.org/10.3390/s20237000, https://www.mdpi.com/1424-8220/20/23/7000.

[21] M.A. Kassab, H.S. Taha, S.A. Shedied, A. Maher, A novel voting algorithm for redundant aircraft sensors, in: Proceeding of the 11th World Congress on Intelligent Control and Automation, 2014, pp. 3741–3746.

[22] S. Kolozali, M. Bermudez-Edo, N. Farajidavar, P. Barnaghi, F. Gao, M. Intizar Ali, A. Mileo, M. Fischer, T. Iggena, D. Kuemper, R. Tonjes, Observing the pulse of a city: a smart city framework for real-time discovery, federation, and aggregation of data streams, IEEE Int. Things J. 6 (2) (2019) 2651–2668, https://doi.org/10.1109/JIOT.2018.2872606.

[23] G. Latif-Shabgahi, J. Bass, S. Bennett, History-based weighted average voter: a novel software voting algorithm for fault-tolerant computer systems, in: Proceedings Ninth Euromicro Workshop on Parallel and Distributed Processing, 2001, pp. 402–409.

[24] G. Latif-Shabgahi, J. Bass, S. Bennett, A taxonomy for software voting algorithms used in safety-critical systems, IEEE Trans. Reliab. 53 (3) (2004) 319–328, https://doi.org/10.1109/TR.2004.832819.

[25] B.P.L. Lau, S.H. Marakkalage, Y. Zhou, N.U. Hassan, C. Yuen, M. Zhang, U.-X. Tan, A survey of data fusion in smart city applications, Inf. Fusion 52 (2019) 357–374, https://doi.org/10.1016/j.inffus.2019.05.004, https://www.sciencedirect.com/science/article/pii/S1566253519300326.

[26] Y.-W. Leung, Maximum likelihood voting for fault-tolerant software with finite output-space, IEEE Trans. Reliab. 44 (3) (1995) 419–427, https://doi.org/10.1109/24.406576.

[27] D. Lupton, 'Flawed', 'Cruel' and 'Irresponsible': The Framing of Automated Decision-Making Technologies in the, Australian Press, April 2021.

[28] N. Minaei, Future transport and logistics in smart cities: safety and privacy, in: Smart Cities, CRC Press, 2022, pp. 113–142.

[29] Mindstorms Lego EV3, https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-intelligent-brick/45500, jun 2022.

[30] J.J. Moré, The Levenberg-Marquardt algorithm: implementation and theory, in: G.A. Watson (Ed.), Numerical Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg, 1978, pp. 105–116.

[31] N.U. Okafor, Y. Alghorani, D.T. Delaney, Improving data quality of low-cost IoT sensors in environmental monitoring networks using data fusion and machine learning approach, ICT Express 6 (3) (2020) 220–228, https://doi.org/10.1016/j.icte.2020.06.004, https://www.sciencedirect.com/science/article/pii/S2405959520300862.

[32] D. Pelleg, A. Moore, X-means: extending k-means with efficient estimation of the number of clusters, in: Proceedings of the 17th International Conf. on Machine Learning, Morgan Kaufmann, 2000, pp. 727–734.

[33] T.T.S. Rappaport, Wireless communications - principles and practice, 1996.

[34] O.E. Rhazal, M. Tomader, Study of smart city data: categories and quality challenges, in: Proceedings of the 4th International Conference on Smart City Applications, SCA '19, Association for Computing Machinery, New York, NY, USA, 2019.

[35] A. Torres-Echeverría, S. Martorell, H. Thompson, Multi-objective optimization of design and testing of safety instrumented systems with MooN voting architectures using a genetic algorithm, Reliab. Eng. Syst. Saf. 106 (2012) 45–60, https://doi.org/10.1016/j.ress.2012.03.010, https://www.sciencedirect.com/science/article/pii/S0951832012000440.

[36] J. Wang, C. Xu, J. Zhang, R. Zhong, Big data analytics for intelligent manufacturing systems: a review, J. Manuf. Syst. 62 (2022) 738–752, https://doi.org/10.1016/j.jmsy.2021.03.005, https://www.sciencedirect.com/science/article/pii/S0278612521000601.

[37] X. Wang, C. Wang, Time series data cleaning: a survey, IEEE Access 8 (2020) 1866–1881, https://doi.org/10.1109/ACCESS.2019.2962152.

[38] G. Welch, G. Bishop, Welch & Bishop, an Introduction to the Kalman Filter 2 1 the Discrete Kalman Filter in 1960, 1994.

[39] S.E. Whang, Y. Roh, H. Song, J. Lee, Data collection and quality challenges in deep learning: a data-centric AI perspective, CoRR, arXiv:2112.06409, https://arxiv.org/abs/2112.06409.

[40] Wireless VINT Hub, https://www.phidgets.com/?tier=3&catid=64&pcid=57&prodid=1143, jun 2022.

[41] J. Wisanmongkol, L. Klinkusoom, T. Sanpechuda, L.-o. Kovavisaruch, K. Kaemarungsi, Multipath mitigation for rssi-based bluetooth low energy localization, in: 2019 19th International Symposium on Communications and Information Technologies (ISCIT), 2019, pp. 47–51.

[42] Y. Wu, S. Cheng, X. Yan, Study on improved algorithm of RSSI correction and location in mine-well based on bluetooth positioning information, in: Proceedings of the 4th International Conference on Computer Science and Application Engineering, CSAE 2020, Association for Computing Machinery, New York, NY, USA, 2020.

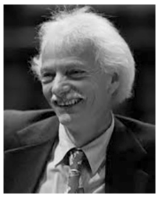
[43] Y. Zheng, Methodologies for cross-domain data fusion: an overview, IEEE Trans. Big Data 1 (1) (2015) 16–34, https://doi.org/10.1109/TBDATA.2015.2465959.

[44] C. Zhou, J. Yuan, H. Liu, J. Qiu, Bluetooth indoor positioning based on RSSI and Kalman filter, Wirel. Pers. Commun. 96 (3) (2017) 4115–4130, https://doi.org/10.1007/s11277-017-4371-4.

[45] N. Zubair, N. A, K. Hebbar, Y. Simmhan, Characterizing IoT data and its quality for use, CoRR, arXiv:1906.10497, http://arxiv.org/abs/1906.10497.

Dr.-Ing. habil. Josef Spillner is a senior lecturer / associate professor for computer science at Zurich University of Applied Sciences, Switzerland. His research activity focuses on distributed application computing paradigms. Particular emphasis is on technological support for emerging digitalisation needs of industry and society, such as smart cities and mobility.

Dr Panagiotis Gkikopoulos is a distributed systems and data science researcher, having completed his PhD at the University of Neuchâtel, Switzerland. His research focuses on data quality in data-intensive systems, with a particular interest in cyber-physical systems and cloud-native technologies.



Peter Kropf was a full Professor of Computer Science at the University of Neuchâtel, Switzerland since 2003 and is now a professor emeritus since 2021. From 1994 to 2003, he was a professor at Laval University, Quebec, and University of Montreal, Canada. He was appointed Dean of the Faculty of Sciences of the University of Neuchâtel from 2011 to 2014. He has published numerous research papers in the fields of parallel and distributed systems, simulation an optimization and was also an active member in multiple international research projects.

Valerio Schiavoni, PhD, Member of IEEE, is Maître-assistant (Lecturer) at the University of Neuchâtel, Switzerland. His research interests lie at the intersection of systems broadly conceived, security, and data management. Contact him at valerio.schiavoni@unine.ch.