25th International Congress on Modelling and Simulation, Darwin, NT, Australia, 9 to 14 July 2023
mssanz.org.au/modsim2023

# Indoor movement data analytics based on machine learning for manufacturing assembly line

**Mashud Rana, Daniel Smith, Ashfaqur Rahman and Peter Baumgartner**

*Data61, CSIRO, Australia*
*Email: ashfaqur.rahman@data61.csiro.au*

**Abstract:** The paper presents an analysis on how machine learning models can be used to extract information from indoor movement data in an assembly line. An Indoor Positioning System (IPS) is commonly used to track workers, moving objects and vehicles at indoor venues where GPS is not effective. Movements happen within an assembly line as certain sets of work activities are performed (e.g., delivering assembled units from one work bench to another, reworking at a work bench and transporting a trolley from one place to another) and anything beyond that can be considered as anomalous.

A computer assembly line was simulated and the movement trajectories of five workers were generated to perform the following activities: working at their own station, moving between different workstations, helping at another work station, working at storage and moving to a storage zone. We then developed a set of machine learning models to investigate how accurately these behaviours can be detected.

The machine learning pipeline for indoor tracking data involves:

(a) segmenting worker trajectories using Self–Supervised Learning (SSL). SSL is a machine learning process where a model is trained to learn one part of the input data from another part without the use of manually acquired labels. SSL is used to find the change points of the trajectories (i.e., used as segment boundaries) by training a model to predict the future interval of a trajectory from its temporally adjacent past window.

(b) classifying trajectory segments (i.e., sub-trajectories) into different categories of worker behaviour. We investigated different supervised machine learning models to learn from the labelled segments.

(c) estimating the state of the assembly floor based on the inferred worker behaviour. The state provides a wholistic view of the factory floor including the collective behaviour of workers. We used a clustering approach to identify the dominant states (joint activity) of the factory floor.

We received some promising results from our analysis. We present the methods and results in detail in the paper.

***Keywords:*** *Movement analytics, self-supervised learning, activity classification, state estimation*

## 1. INTRODUCTION

Moving objects leave characteristic signatures in their spatial trajectories that can be used to understand their activity or behaviour (Baumgartner et al. 2022). Such insights can be helpful in many applications: surveillance at airports, understanding worker efficiency in assembly lines and detecting illegal fishing activities in oceans. Whilst several methods have been developed for analysing different types of trajectory data, this paper focuses upon analysing the movement data acquired with an Indoor Positioning System (IPS). Technologies like IPS (Farahsari et. al. 2022) were developed to estimate the spatial location $(x, y)$ of an object in an indoor environment (at time $t$) since Global Positioning System (GPS) do not work well in such environments. The underlying technology behind IPS can be based on vision (MoCap (Delamare et al. 2020)), wireless (UWB (Delamare et al. 2023)), or magnetism ((Angelis et al. 2015; Blankenbach et. al 2010)). Irrespective of the underlying technology, IPS captures a sequence of spatial positions $(o_i, x, y, t)$ of a moving object $o_i$, which is known as its trajectory. Whilst GPS based trajectory analysis for behaviour classification (Dunne et al. 2017) and anomaly detection (Datlıca et al. 2021) is very common in the literature, IPS based trajectory analysis is a relatively new area. GPS is commonly utilised over larger spatial areas than IPS, however, when deployed, both technologies can often exhibit similar scales of measurement error. Consequently, as the impact of noise can be more significant with IPS based analysis, different methodologies may be required.

In this study, we aim at identifying how IPS based trajectories can be utilised to infer meaningful information about the assembly floor. As an example, identifying workers who are frequently searching for misplaced items or tools could be used to identity operational inefficiencies. Similarly, discovering the persistence reworking of manufactured products that fail to pass quality checks could also be used to alert of potential production issues. Therefore, inferring such activities (and others) from the movement of workers could help to identify bottlenecks within the manufacturing process and assist in developing appropriate management policies for optimizing productivity and efficiency (Gyulai et al. 2019).

A simple use case for method development was considered, an indoor computer assembly line. We simulated a scenario where a set of workers assemble computers from separate parts. Trajectories of the workers were generated as they moved between different workstations, storage areas and rework benches. An analytical pipeline is proposed to infer state information based on worker trajectories and is comprised of: (a) a segmentation of trajectories based on Self-Supervised Learning (SSL), (b) a supervised classification model to classify segments into categories of worker behaviour, and (c) a state estimation of the assembly floor based on these behaviour categories.

The following are the key contributions of this paper: (a) a computer assembly line was simulated and synthetic trajectory data was generated based on probabilistic logic programming, (b) a self-supervised learning (SSL) method was adapted to segment IPS trajectory data, (c) an ML framework was proposed to infer worker behaviour, and (d) an approach to infer the factory floor state from (b) and (c) was demonstrated.

Trajectories are time sequences of spatial co-ordinates; we use SSL to train a short time interval of a sequence to predict the next contiguous interval. Larger prediction errors can be associated with statistical changes in the trajectory that correspond to the segment boundaries used for trajectory segmentation. Each segment is then classified as a particular behaviour using a supervised behaviour classification model that has been trained on a set of labelled segments. The collective behaviour of the workers represents a state of the assembly floor, and we use a clustering method to find the dominant states of the floor. We present these methods and the results obtained on the synthetic data set in the following sections.

## 2. DATA SET

We simulated an indoor assembly line scenario where computers are assembled from different parts. There are six workbenches and one storage area in the computer assembly line. Figure 1 presents different spatial zones representing the work benches and storage area on the work area grid. Five workers with dedicated duties are assigned to different work benches. The computer assembly workflow is presented in Figure 2. Worker 1 assembles the power unit and casing at WB 1 and delivers to worker 2 at WB 2. Worker 2 attaches the mother board and other necessary cards (e.g., graphics, sound card etc.) to the component prepared by Worker 1 and delivers them to worker 3 at WB 3. Worker 3 installs the software at WB 3 and passes it on to worker 4 at WB 4. Worker 4 tests the system as part of a quality check at WB 4. Worker 4 fixes any issues at WB 5 arising from the testing and delivers the final product to worker 5 at WB 6 who does the packaging. Workers occasionally go to a storage area to bring in extra parts. The assembly process is sequential in nature. Hence, the workers at later workbenches help other workers who commence their work earlier in the workflow. Similarly, workers who commence working earlier, finish their assembly tasks earlier. They help workers who are responsible for assembly activities at the later workbenches. A total of three computers were assembled and one of them was made to rework.

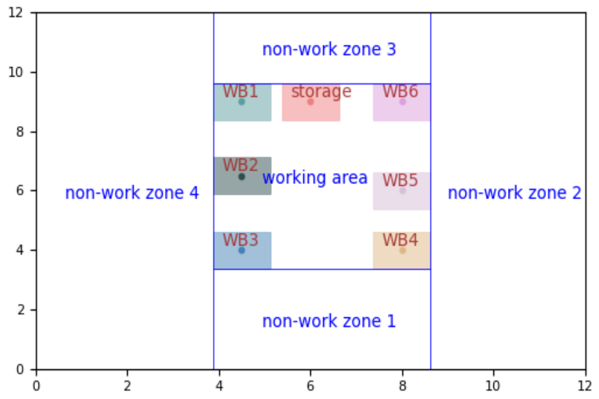
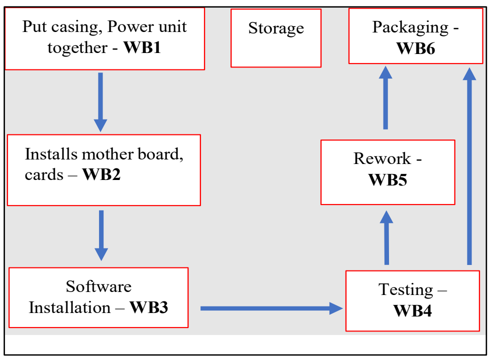


**Figure 1.** Different work areas/zones of the synthetic computer assembly data set where workers spend most of their time during the work shift



Put casing, Power unit together - WB1
Storage
Packaging - WB6
Installs mother board, cards – WB2
Rework - WB5
Software Installation – WB3
Testing – WB4

**Figure 2.** Workflow in the computer assembly line. Arrows indicate how partially assembled components move between work benches

The movement between benches are generated by simulation. We implemented a general method that allowed us to automatically generate trajectories from a list of worker activities of given durations and locations as specified in a workflow. The activities are high-level properties such as "working at WB *i*", and "restock from storage area". These activities are operationalized by means of probabilistic logic programs. In these programs, probabilities are used to express, broadly speaking, a rate of change between locations (somewhat reminiscent of a Poisson process). This mimics the occasional worker movements for helping a colleague. The logic programs are generative and can be sampled to obtain, in a first step, the coarse waypoints of the result trajectory. In a second step, the coarse waypoint planning is refined into absolute coordinates with an event rate of 1Hz. To ensure realism, locations are changed at a typical walking pace of 1.3m/sec, and random perturbations are added for small worker movements at workstations and for slightly non-optimal movements along paths which simulates avoiding obstacles and collisions with other workers. We use probabilistic logic programs for that as well. All programs are executed in a probabilistic extension of our logic programming system Fusemate (Baumgartner, 2021).

## 3. MODELLING

### 3.1. Trajectory segmentation

To model the spatial trajectories of workers, which represent worker's movement across the assembly line, they must be segmented into a sequence of sub-trajectories. Time Series Change Point Detection (TS–CP$^2$) (Deldari et al. 2021), a self-supervised learning method, was used for segmentation. First, a sliding window analysis was performed on trajectories to form pairs of temporally adjacent windows at each time step. The pairs of short-time windows were then passed through the encoders and projected into a low dimension embedding.

TS–CP$^2$ (Figure 3) is a Siamese network architecture comprised of a pair of identical encoders. The encoder used by TS–CP$^2$ is a neural based sequence model, the Temporal Convolutional Neural Network (TCNN). The architecture is trained using a contrastive learning cost function, infoNCE, where the cosine similarity between a positive sample pair is contrasted against the mean cosine similarity of a set of negative sample pairs. The positive sample pair is composed of the embedding of an anchor window and the embedding of a temporally adjacent future window. Whilst the negative sample pairs are comprised of the same anchor embeddings (as in the positive pair) and a set of embeddings from randomly selected windows that are temporally distant to the anchor. By minimising the infoNCE cost function, the network is trained to push together the pairs of positive samples (consecutive time intervals), whilst simultaneously, push apart the pairs of negative samples (temporally distant intervals) within latent space.

Once the network architecture is trained, an equivalent sliding window analysis is performed with the test data. Positive sample pairs are constructed and then passed through the encoder pair. A detection function is formed by computing the cosine similarity between positive sample embeddings at each time step. The local minima within the detection function are found with a valley picking algorithm and those with a cosine similarity less than a specified threshold are estimated as the change points. These change points are then used to partition trajectories by leveraging them as the segment boundaries.

## 3.2. Behaviour classification

We formulate the behaviour classification task as a supervised learning problem and apply ML classifiers to model the relationship between features of trajectory segments and behaviour of the worker pertaining to each segment. We first prepared the input-output pairs for the segments belonging to each trajectory. The input data consisted of 10 features that reflect the worker's homogeneous behaviour within the segment. These features included the duration of the segment, the cumulative distance between consecutive spatial points within the segment, the two end points of the segment, the number of unique spatial points within the segment, the spatial area where the work spent the highest proportion of time and the duration of stay in that area. Moreover, the output of each segment was a class label manually annotated with the worker activity during the segment.

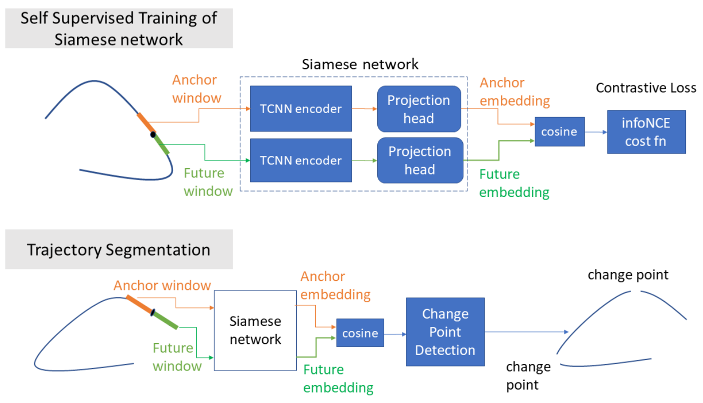


**Figure 3.** Using the TS–CP$^2$ architecture for trajectory segmentation. The Siamese network is trained with self-supervised learning. Trajectories are segmented by passing temporally adjacent windows through the Siamese network and computing the cosine similarity function. Change point detection is applied to extract segments (by finding local minima in the cosine similarity function).

The combined set of feature-label pairs from the trajectories were then fed to the classifier models. We adopted a set of complementary Machine Learning (ML) models with different pattern learning capabilities that include *k*–Nearest Neighbour (kNN), Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Neural Networks (NN), Random Forest (RF), and Gradient Boosting (GB). The final two classifiers (RF and GB) are based on an ensemble technique that trains multiple base learners and combines their outputs to improve the generalisation capability of the classifiers. In addition, we applied a voting–based ensemble classifier (Voting) which included all eight classifiers as its members and combined their outputs based on the consensus amongst the members. This was done to attempt to reduce the over-fitting problem and subjective bias associated with individual classifiers.

## 3.3. State estimation

State represents a wholistic view of the factory floor – a representation of some form that conveys what's happening on the floor. A floor state called worker activity is defined i.e., who is doing what at a given time segment. It is represented as a vector $\boldsymbol{a}_t = (a_{1,t}, a_{2,t}, \dots, a_{n,t})$ where $a_{i,t}$ represents the activity of worker $i$ at time segment $t$. $\boldsymbol{a}_t$ represents the state of worker activities in the floor at time $t$. Given, activity vectors over time $\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_T$, we can cluster the vectors into $k$ groups. The centroids of the $k$ groups will tell us the most observed worker behaviour patterns (states) on the floor. We have adopted *k*–mode clustering algorithm (Miguel et al. 2013) to identify the clusters. *k*–mode clustering is designed to work on categorical attributes (activity class label in this case). The cluster centroids are randomly initialized, and an iterative process is followed to update them until convergence.

## 4. RESULTS AND DISCUSSION

### 4.1. Trajectory segmentation

The TCNN used by the TS-CP$^2$ architecture (detailed in Section 3.1) is composed of four convolutional neural networks (CNN) layers with dilation rates of 1, 2, 4 and 8, respectively. Each CNN layer is composed of 32 filters with a kernel size of 2. The projection head (i.e., mapping the TCNN output into a lower dimension space) is a multi-layer perceptron network with a feature dimension of 16. The architecture is trained using the Adam learning algorithm (initialized with a learning rate of 0.001) with a window size of 16 seconds with 50%

overlap and a batch size of 32. The cosine similarity function is computed between the embeddings of consecutive time windows. The threshold used to detect change points from the metric function was set to 0.9.

A 5-fold cross–validation was performed to segment worker trajectories. The TS–CP$^2$ architecture was self-trained upon the trajectories of four workers and then used to segment the trajectory of the remaining worker. This process was repeated five times so that each of the worker's trajectories were segmented once. Figure 4 shows an example of the segmentation results obtained with the trajectory of Worker 5.

Table 1 presents a summary of the segmentation results of the five workers. The TS–CP$^2$ algorithm identified 277 segments from the trajectories of the five workers. These segments were manually labelled with five activity classes that reflected the basic working and collaborative behaviour of workers; such behaviours may assist in analysing the performance of a manufacturing system. For example, the segments associated with the 'work at storage' and 'moving between storage and WS' classes can be utilised to identify the proportion of time that workers spent acquiring new supplies or finding tools. Likewise, segments with labels 'work at own WS' and 'work at other WS' can be used to identify the time (or proportion of time) that a worker spends completing their assigned tasks and collaborating with other workers. This information may help increase the production efficiency of a system by identifying resourcing issues and bottlenecks within production.

A comparison of the segments associated with the activity classes (and their cumulative duration) indicate that whilst workers spent a reasonable proportion of their working day completing assigned tasks at their workstation (35%), they spent approximately half of their working hours (50.6%) at the workstations of other workers. This is expected, given substantial worker collaboration was necessary due to the interdependencies between assembly tasks (Figure 2). Workers were required to interact with other workers during assembly and were also expected to assist other workers with tasks once their own tasks were completed. Moreover, workers spent approximately 11% of their working hours moving between a WS and storage area to either restock workstation supplies or to find missing tools. With respect to the five activity classes, the workers spent the lowest proportion of their time (3%) moving between workstations.

**Table 1.** Summary of trajectory segmentation results

| Activity class | Segment Count | Duration (sec) |
|---|---|---|
| work at storage | 28 | 2673 |
| work at own WS | 60 | 9253 |
| work at other WS | 133 | 13225 |
| moving between WS | 44 | 771 |
| moving between storage and WS | 12 | 175 |

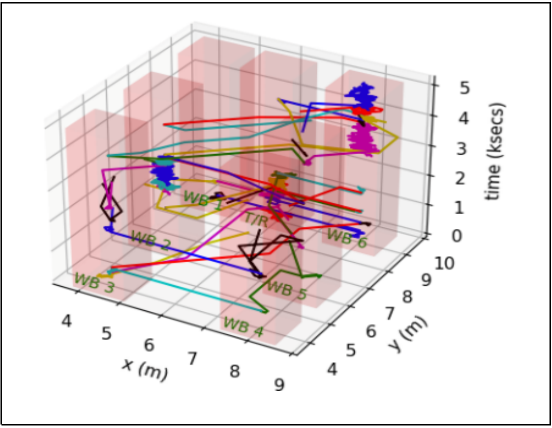


**Figure 4.** Segmentation of the movement trajectory of worker 5 using the TS-CP$^2$ method (segments represented with different colours)

## 4.2. Behaviour classification

To evaluate the performance of the ML classifiers for activity classification, weed consider two standard metrics: the *accuracy* and *weighted f-score*. Accuracy refers to the percentage of correctly classified examples. F–score is a suitable measure for classification problems with imbalanced classes as ours and defined as the harmonic mean of *precision* and *recall*. Taking the class imbalance into account, we computed the weighted f–score, i.e., computed the F–score for each class separately, and the computed the average weighted by the number of instances in each class. Both metrics have ranges 0 to 1, where 0 and 1 indicate the least accurate and the most accurate models respectively.

The accuracy of the classifiers was computed using 5–fold cross validation, where the dataset was split into 5 non-overlapping subsets (called folds) of equal size. Each classifier was then built five times – each time the training was performed upon four subsets of the dataset and testing was performed on the remaining subset. The evaluation showed that the GB, RF, and NN classifiers achieved the highest classification accuracy in terms of both metrics, with GB being the most accurate of all the classifiers. The accuracy and f–score for the three top performing classifiers were in the range of 0.77-0.83 and 0.76-0.83, respectively. The Voting and SVC classifiers were next in the ranking with similar accuracies of 0.75 and 0.73, and f-scores of 0.73 and 0.71. The *k*–NN and NB offered the lowest classification accuracies of 0.62 and 0.49, respectively. In addition, a comparison of the classification performance of individual worker trajectories (Figure 5) showed a similar trend; in most cases, the GB and RF were the top performers, followed by the Voting, SVC, and NN classifiers.

The experiment indicated that 3 out of the 5 top performing classifiers were based on ensemble techniques. For example, the RF combined multiple base learners of the same type (DTs) – each trained separately with a different subset of input data based on bootstrap sampling. GB trained multiple base learners sequentially to make them complement each other (each base learner attempts to minimize the errors of the previous leaner), whilst the Voting model combined several types of base learners based on soft or hard voting. Hence, ensemble classifiers help to diversify the learning process compared to standalone classifiers. The performance variance of the ensembles across different trajectories were also relatively lower compared to the standalone classifiers (Figure 5(f)). The superior performance of these ensemble-based classifiers can be attributed to their greater robustness to over-fitting and better generalisation performance.

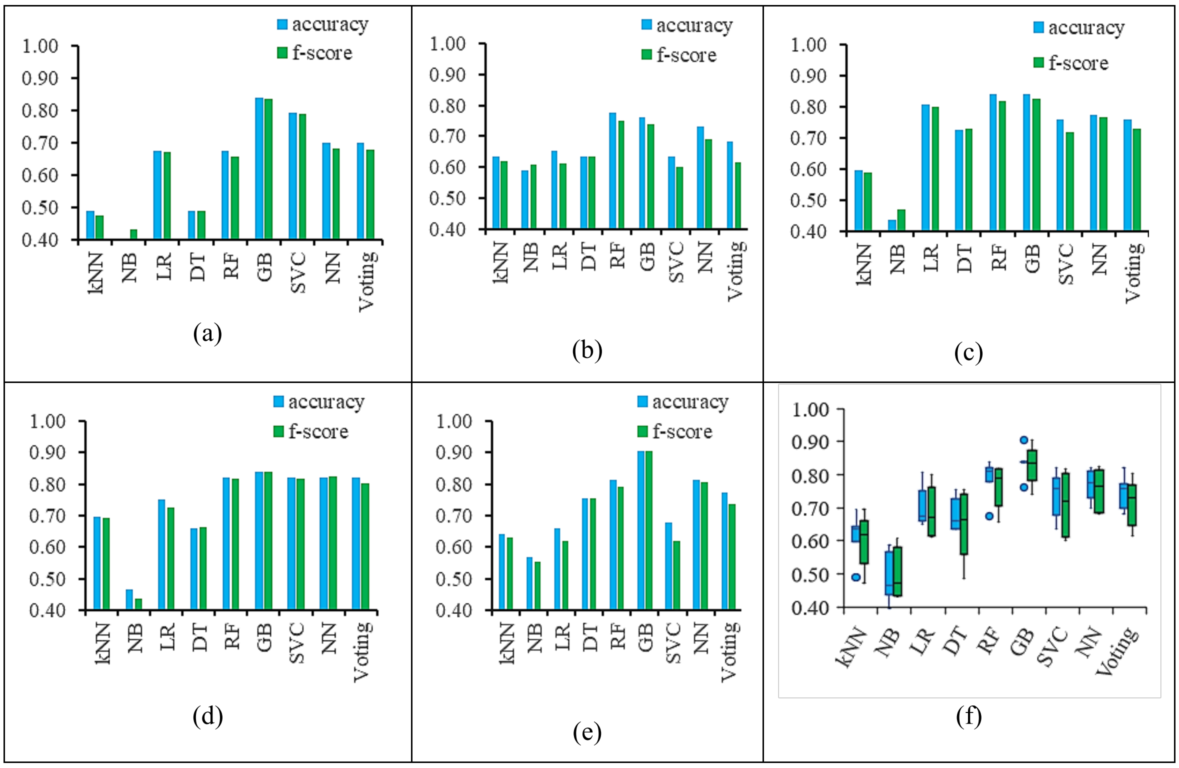


**Figure 5.** Performance of the classifiers on all trajectories: (a) to (e) accuracy and f-score for workers 1 to 5, respectively, (f) variations of accuracy and f-score for classifiers across the trajectories

## 4.3. State estimation

State represents a wholistic view of the factory floor at a given point in time. We consider the joint activities of all the workers at a given time to present the state of the factory floor. We use the activities predicted by the ML model for this purpose. To understand the dominant states, we cluster the joint activities of workers. The input to the clustering algorithm was a two-dimensional matrix where each row represents the activities of the five workers at a given time. We used the *k*–mode clustering algorithm (Miguel et al. 2013) to find the centres of the ten clusters from the data. Each cluster centre represents the most commonly occurring joint activity in that group. Table 2 presents the ten cluster centres (ten most observed states or dominant states) in the computer assemble data set and the frequency of their presence as percentage. In the simulated data set, the workers either work at their workstation or help someone at another workstation. Workers also go to the storage areas to pick up component parts. Worker 5 spent more of their time in storage areas during the simulation. The states presented in Table 2 clearly reflects how things were defined in the simulation. This is a validation of the results produced by the trajectory segmentation and activity classification methods.

## 5. CONCLUSIONS

We presented a pipeline of ML methods to generate insights from indoor movement trajectories in the context of a manufacturing assembly line. We simulated a scenario to mimic the operations of a computer assembly line and generated indoor trajectories of the workers. A self-supervised learning method was adopted to segment the trajectories. Several ML classifiers were then applied to the segments to infer the worker activity. Finally, a *k*–mode clustering algorithm was applied to generate a joint activity state of the factory floor. The state estimation results conformed to the underlying storyline of the assembly process. In the future, we aim to develop machine learning models to infer activity classes at a higher semantic level (e.g., searching, collaborating behaviour etc.).

**Table 2.** Centroids of the 10 clusters representing joint activities of workers

| Cluster ID | Occurrence Percentage (%) | Cluster Centroid [Concurrent activities of 5 workers] | | | | |
|---|---|---|---|---|---|---|
| | | Worker 1 | Worker 2 | Worker 3 | Worker 4 | Worker 5 |
| 1 | 18.37 | work at own WS | work at other WS | work at own WS | work at other WS | work at storage |
| 2 | 15.2 | work at other WS | work at other WS | work at other WS | work at other WS | work at other WS |
| 3 | 14.75 | work at other WS | work at own WS | work at own WS | work at other WS | work at other WS |
| 4 | 11.66 | work at other WS | work at other WS | work at other WS | work at own WS | work at other WS |
| 5 | 14.4 | work at other WS | work at own WS | work at other WS | work at other WS | work at other WS |
| 6 | 9.07 | work at own WS | work at other WS | work at other WS | work at other WS | work at other WS |
| 7 | 1.48 | work at storage | work at other WS | work at other WS | work at other WS | work at other WS |
| 8 | 8.39 | work at own WS | work at storage | work at other WS | work at storage | work at other WS |
| 9 | 4.87 | work at other WS | work at other WS | work at own WS | work at own WS | work at own WS |
| 10 | 1.81 | work at own WS | work at storage | work at other WS | work at other WS | work at storage |

## REFERENCES

Angelis, G. De, Pasku, V., Angelis, A. De, Dionigi, M., Mongiardo, M., Moschitta, A., Carbone, P., 2015. An Indoor AC Magnetic Positioning System, IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 5, pp. 1267–1275, doi: 10.1109/TIM.2014.2381353.

Baumgartner, P. The Fusemate Logic Programming System. In: Platzer, A., Sutcliffe, 2021. G. (eds) Automated Deduction – CADE 28. CADE 2021. Lecture Notes in Computer Science, vol 12699. Springer, Cham. https://bitbucket.csiro.au/projects/MOVEMENTANALYTICS/repos/fusemate-distrib/

Baumgartner, P., Smith, D., Rana, M., Kapoor, R., Tartaglia, E., Schutt, A., Rahman, A., Taylor, J., Dunstall, S., 2022. Movement Analytics: Current Status, Application to Manufacturing, and Future Prospects from an AI Perspective, arXiv:2210.01344, doi: 10.48550/arXiv.2210.01344.

Blankenbach, J., Norrdine, A., 2010. Position estimation using artificial generated magnetic fields, International Conference on Indoor Positioning and Indoor Navigation, pp. 1–5, doi: 10.1109/IPIN.2010.5646739.

Datlıca, M.T., Demir, E., 2021. Anomaly Detection in Location and Trajectory Datasets, Signal Processing and Communications Applications Conference (SIU), Istanbul, Turkey, pp. 1–4, doi: 10.1109/SIU53274.2021.9477872.

Delamare, M., Duval, F., Boutteau, R., 2020. New Dataset of People Flow in an Industrial Site with UWB and Motion Capture Systems, Sensors (Basel). 12;20(16):4511. doi: 10.3390/s20164511.

Delamare, M., Duva, F., Boutteau, R., 2023. Dataset of person flows during an assembly phase in an industrial site with an UWB system in NLOS and a motion capture system, [Online]. Available: https://github.com/vauchey/IndoorInsdustrialLocalisationDataset/, Last accessed March 2023.

Deldari, S. Smith, D., Xue, H., Salim, F., 2021. Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding, In Proceedings of the Web Conference 2021, ACM.

Dunne, R., Henry, D., Rawnsley, R., Rahman, A. 2017. Behavior Classification of Dairy Cows Fitted with GPS Collars, In: Kang, U., Lim, EP., Yu, J., Moon, YS. (eds) Trends and Applications in Knowledge Discovery and Data Mining (PAKDD). Lecture Notes in Computer Science, vol 10526. Springer, Cham.

Farahsari, P. S., Farahzadi, A., Rezazadeh, J., Bagheri, A., 2022. A Survey on Indoor Positioning Systems for IoT-Based Applications. IEEE Internet of Things Journal, 9(10), 7680–7699.

Gyulai, D., Pfeiffer, A., Bergmann, J., 2019. Analysis of asset location data to support decisions in production management and control, CIRP Conference on Intelligent Computation in Manufacturing Engineering, Italy, doi: 10.1016/j.procir.2020.05.035.

Miguel, W.W., Perpinan, A.C., 2013. The K-modes algorithm for clustering, arXiv:1304.6478, doi: 10.48550/arXiv.1304.6478.