TÉCNICO LISBOA

Intelligent Systems

Mechanical Engineering

End-point position estimation of a soft continuum robot joint using magnetic sensors

---

Group 11:

Diogo João Mendes Pereira (ISTID 87172)
diogojmpereira@tecnico.ulisboa.pt

Margarida do Rio Filipe Nabais (ISTID 113219)
margarida.filipe.nabais@tecnico.ulisboa.pt

2025/2026 – 1º Semester, P1

1 Introduction

Soft continuum robotic joints have increasingly gained attention in recent years due to their high dexterity, smooth motion, and inherent safety in physical interaction. These characteristics make them particularly suitable for applications such as minimally invasive surgery, human-robot collaboration, and inspection tasks in constrained environments. However, their structure and nonlinear deformation behavior present themselves as significant challenges for accurate position estimation and control. Conventional calibration and modeling techniques normally fail in this aspect due to hysteresis effect, material elasticity, and the complex coupling between embedded magnetic sensors and actuated components.

To overcome these limitations, data-driven and intelligent modeling approaches have emerged as alternatives. By using machine learning and soft computing techniques such as artificial neural networks, fuzzy inference systems, and neuro-fuzzy hybrid models it becomes possible to predict the end-point position of the robot joint from sensor data, while capturing nonlinearities and uncertainties inherent in the system. The objective of this project is to design and evaluate a data-driven model capable of estimating the 3D end-point position of a soft continuum robot joint based on twelve hall effect sensor signals. The corresponding ground-truth positions, measured by an optical 3D positioning system , serve as reference targets for supervised learning. The project explores different intelligent system methodologies, comparing their performance in terms of accuracy, robustness and interpretability. For each modeling approach, it was systematically tracked training and validation metrics, including mean squared error and mean absolute error and reported the R^2 when relevant to provide further insight into each model's behavior and generalization capability.

2 Takagi sugeno model

One of the models developed for this project was a Takagi-sugeno model to predict end-point position. The TSK model is a useful approach that combines fuzzy logic systems with the precision of mathematical regression models, providing a way to describe how input conditions contribute to the predicted output through a set of if rules. The model divides the input space into several fuzzy regions, each with their own rule that defines a local linear model.

2.1 Feature engineering

Feature engineering is the process of transforming a raw dataset into meaningful inputs improving their efficiency to develop an efficient predictive model. For the tsk model two methods of feature engineering were explored: Feature selection and Dimensionality reduction.

2.1.1 Feature Selection

To perform feature selection a gridsearch was developed to determine which combination of sensors provided the most informative input for predicting each coordinate. For this the code generates all possible subsets of 10 sensors from the total of 12, trained the model for each subset and evaluated performance by using metrics like MSE, RMSE and R^2. The value of

10 was chosen to find a balance. Using 12 sensors would include all available information but increase redundancy and overfitting risk, on the other hand using few sensors might result in losing critical information reducing predictive performance, for this reason it is important to choose a value that provides balance and retains most of the information while still exploring combinations to remove redundancy. It was concluded that for each coordinate the best subset of sensors was composed of sensors 1 to 10 equally.

2.1.2 Dimensionality Reduction

Dimensionality reduction was performed through an autoencoder to compress the original sensor data into a smaller set of latent features (In this case 10 as well), to capture the most relevant information. The autoencoder improves model generalization and reduces overfitting.

The autoencoder was trained for 50 Epochs with the Adam optimizer and MSE loss. During training, it can be verified the training and validation losses decreased steadily, this is important because it indicates that the model learned to reconstruct the input data efficiently. At the end of the process the final validation loss was of 0.000512 and training loss of 0.000014, meaning that the autoencoder was able to capture the most relevant features and produce a compressed dataset.

2.2 TSK model training

The takagi sugeno models were trained for each coordinate (x,y,z), to achieve the best performing models the following methods were applied:

2.2.1 Validation set and standardization

A validation set is a subset of the dataset that is separate from the training and test sets, it is used to monitor model performance during training and perform hyperparameter selection (which will be discussed on the next section) without interfering with the test set, so it is the validation performance that is taken into account, this prevents overfitting and guarantees that the final evaluation on the test set is a viable generalization. Besides validation splitting, the sensor input was standardized to have zero mean and unit variance, ensuring that each sensor contributes equally to the learning process, this way features with higher numerical ranges wont dominate the training, this provides stability and convergence.

2.2.2 Parameter tuning

To identify the optimal hyperparameters that lead to the most efficient model performance a gridsearch was developed to choose the best performing model for each one of the coordinates. These parameters were:

- Number of clusters (Determines the number of fuzzy rules)
- Learning rate (controls step size)

- Gradient descent epochs (number of iterations for the update of rule parameters)
- Maximum iterations (controls the number of iteratios in optimization)

The models were evaluated in the following metrics:

- Mean Squared Error (MSE) - averaged squared deviation
- Root Mean Squared Error
- Mean Absolute Error (MAE) - robust to outliers
- R2- proportion of variance
- Normalized rmse (NRMSE)- percentage error relative to the output range

2.2.3 Cross validation

During the process of building the code a 5-fold cross-validation was considered to be implemented to provide a more comprehensive performance evaluation. However, due to the high computational cost associated with raining tsk models for each coordinate and parameter configuration, a hold-out validation approach was adopted instead, this method still guaranteed a fair evaluation while reducing training time.

2.3 Results for the tsk model

Table 1: Performance metrics of the TSK fuzzy model for each coordinate (with autoencoder)

Coordinate	Best Feature Subset	Clusters	MSE	RMSE	MAE	R^2	NRMSE (%)
1	[0, 1, 2, 3, 4, 5, 7, 8, 9]	5	2.41	1.55	1.23	0.9987	0.89
2	[0, 1, 2, 3, 4, 5, 7, 8, 9]	5	3.04	1.74	1.39	0.9983	0.98
3	[0, 1, 2, 3, 4, 5, 7, 8, 9]	5	3.68	1.92	1.42	0.9888	2.08

Table 2: Performance of test metrics of the TSK fuzzy model for each coordinate (no autoencoder)

Coordinate	Best Feature Subset	Clusters	MSE	RMSE	MAE	R^2	NRMSE (%)
1	[0, 1, 2, 3, 5, 7, 8, 9, 10, 11]	5	2.05	1.43	1.14	0.9989	0.82
2	[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11]	5	2.52	1.59	1.24	0.9986	0.89
3	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]	5	6.72	2.59	1.98	0.9796	2.81

These tables present the performance metrics of the tsk models for each coordinate, one with the use of an autoencoder and the other without. For the models that do not possess an autoencoder coordinate 1(x) presents the lowest prediction error with an MSE of 2.05, RMSE of 1.43, and NRMSE of 0.82%, indicating good predictive performance. Coordinate 3 (z) seems to be the most challenging coordinate to predict, with higher MSE (6.72), RMSE (2.59), and NRMSE (2.81%). This might be due to the fact that the sensors provide more information for the horizontal (x and y) displacement than the vertical ones (z) possibly because of sensor

placement. The autoencoder slightly improves the prediction for coordinate 3 (z) reducing MSE from 6.72 to 3.68 and NRMSE from 2.81% to 2.08% while showing similar performances for the other coordinates. Overall, the models seem to work efficiently and there is the possibility that for coordinate 3 we achieved better results with the use of the autoencoder due to the fact that the sensors provide richer information for estimating x and y coordinates but provides more limited data for determining z, as a result the introduction of an autoencoder addresses this limitation by learning a more compact informative representation of the sensor data, filtering out redundant information, there is also the possibility that maybe there is a different sensor placement could improve predictions if a different placement is able to provide more meaningful information for z displacements.

3 Anfis

The adaptive Neuro-fuzzy Inference System (ANFIS) is a hybrid modeling approach that combines neural networks with fuzzy logic, it enables to model complex nonlinear relationships between inputs and outputs. It is composed by an fuzzification, rule, normalization, defuzzification and learning layers. The construction of the anfis model was done by the same methodology as the tsk model, with the same methods of data treatment, feature engineering and gridsearch tuning etc... for this reason, as these concepts have already been explored in the tsk subsection the anfis subsection will consist only on presenting and discussing it's results.

3.1 Results and discussion

Table 3: Performance of test metrics of the ANFIS model for each coordinate

Coordinate	Best Feature Subset	Clusters	MSE	RMSE	MAE	R^2	NRMSE (%)
1	(0, 2, 3, 4, 5, 6, 7, 8, 9)	4	0.01498	0.1224	0.0937	0.9850	2.97
2	(0, 1, 2, 3, 4, 5, 6, 8, 9)	5	0.01421	0.1192	0.0929	0.9865	2.76
3	(0, 1, 2, 3, 4, 5, 6, 7, 9)	5	0.10911	0.3303	0.2601	0.8880	6.58

The table presents performance methods for the test set of the anfis model for each coordinate with the inclusion of an autoencoder. Overall, the results demonstrate that the anfis model is capable of efficient predictive performance. For coordinates 1 (x) and 2 (y), the models achieve very low prediction errors, with MSE values below 0.015 and NRMSE below 3%, besides that the models also seem to capture almost all variance in the data (R^2 ¿ 0.98). Just like in the case of the tsk model coordinate 3(z), remains the most difficult coordinate to predict, the model for z still remains reasonable accuracy ($R^2 = 0.888$). Overall, the anfis models provide a robust approach for end point estimation.

4 Model Exploration and Methodology

4.1 Feature Selection Experiments

Some quick tests were made to evaluate the effect of reducing the number of input variables. A few feature subsets were tried, comparing the use of raw sensor data with filtered or engineered

inputs. However, this part was not explored extensively, and the results did not show clear improvements over using all available features.

4.2 Grid Search with Cross-Validation

A grid search with cross-validation was also tested to obtain a more robust estimate of model performance. However, this approach introduced a significant computational cost, as each configuration had to be trained multiple times on different data splits. In practice, the improvement over a single well-chosen validation split was minimal, so this method was not used extensively in later experiments.

4.3 Multilayer Perceptron (MLP)

We started by exploring Multilayer Perceptrons as a baseline approach for the regression problem. A grid search was carried out to test different network sizes, learning rates, and batch sizes, aiming to find a balance between model complexity and accuracy. The models were trained both with the original sensor inputs and with additional engineered features such as the average and difference between redundant sensors. These experiments helped us understand how input representation and architecture depth influence the network's ability to learn the nonlinear mapping between magnetic signals and the robot tip coordinates.

Results with Standard Features

Table 4: Best Model Configuration (MLP)

Parameter	**Value**
Architecture	256–ReLU–128–ReLU–64–ReLU
Learning rate	0.001
Batch size	16

Table 5: Performance metrics for the 3-Expert Model

MSE [Unitless]			
0.002446			
Metric	**X [mm]**	**Y [mm]**	**Z [mm]**
RMSE	1.527	1.447	1.240
MAE	1.230	1.078	0.872
Overall metrics			
Mean residual distance	2.113 mm		
Mean 3D deviation	2.113 ± 1.225 mm		

Results with Computed Features

Table 6: Best Model Configuration (MLP)

Parameter	**Value**
Architecture	256–ReLU–128–ReLU–64–ReLU
Learning rate	0.0005
Batch size	32
Validation loss	0.002048

Table 7: Performance Metrics for the 3-Expert Model

MSE [Unitless]			
0.002048			
Metric	**X [mm]**	**Y [mm]**	**Z [mm]**
RMSE	1.400	1.263	1.093
MAE	1.092	0.941	0.733
Overall metrics			
Mean residual distance	1.853 mm		
Mean 3D deviation	1.853 ± 1.147 mm		

4.4 Autoencoder–Assisted Regression

An autoencoder was tested as a way to compress the sensor inputs into a smaller latent space before performing regression. The idea was to reduce redundancy and highlight the most relevant signal patterns for position estimation. Although the approach worked in principle, the results were not as consistent as with the direct MLP models, and this technique was not explored in depth in later stages of the project.

4.5 Hierarchical Models

Some initial experiments were made with hierarchical model structures, where separate subnetworks would estimate each coordinate or learn intermediate representations before the final prediction. These attempts were only explored briefly and did not show clear advantages over the simpler direct MLP approach, often resulting in unstable training and worse overall accuracy.

Results :

Table 8: Best Model Configuration (Hierarchical MLP)

Parameter	**Value**
Pair architecture	512–ReLU–256–ReLU–128–ReLU
Mid architecture	128–ReLU–64–ReLU
Final architecture	64–ReLU–32–ReLU
Learning rate	0.001
Batch size	16
Epochs	100
Validation loss	0.003906

Table 9: Performance Metrics for the Hierarchical Model

MSE [Unitless]			
0.003906			
Metric	**X [mm]**	**Y [mm]**	**Z [mm]**
RMSE	2.222	2.295	1.518
MAE	1.724	1.841	1.107
Overall metrics			
Mean residual distance	3.158 mm		
Mean 3D deviation	3.158 ± 1.593 mm		

4.6 Expert Models per Coordinate

An alternative setup was tested where three independent expert networks were trained, one for each output coordinate (x, y, and z). Each expert followed the same architecture and tuning procedure as the best-performing MLP. The results were very similar to those of the single multi-output model, with no significant accuracy gain, indicating that it is unknown whether splitting the task by coordinate bring clear benefits for this dataset.

Results :

Table 10: Best Model Configuration (Specialist Models)

Parameter	**Value**
Architecture	512–ReLU–256–ReLU–128–ReLU–64–ReLU–32–ReLU–16–ReLU–8–ReLU
Learning rate	0.0005
Batch size	32

Table 11: Performance Metrics for the Specialist Models

MSE [Unitless]			
0.000714 (Best X Model)			
Metric	**X [mm]**	**Y [mm]**	**Z [mm]**
RMSE	1.206	1.258	1.056
MAE	0.968	0.939	0.768
Overall metrics			
Mean residual distance	1.800 mm		
Mean 3D deviation	1.800 ± 0.956 mm		

4.6.1 Specialist Models with Shared Inputs

A variant of the specialist approach was also tested, where three separate networks were trained for each coordinate, but the Z specialist received, in addition to its own inputs, the predicted outputs from the X and Y specialists. This configuration achieved results very similar to the best models obtained previously. Although the improvement was not clear, these experiments suggest that incorporating information between coordinate estimators could be beneficial for this type of dataset.

Results :

Table 12: Best Model Configuration (Specialist Models with Cross-Expert Input)

Parameter	**Value**
Architecture	512–ReLU–256–ReLU–128–ReLU–64–ReLU–32–ReLU–16–ReLU
Learning rate	0.0005
Batch size	32
Epochs	100

Table 13: Performance Metrics for the Specialist Models (Cross-Expert Setup)

MSE [Unitless]			
0.000638 (Best X Model)			
Metric	**X [mm]**	**Y [mm]**	**Z [mm]**
RMSE	1.128	1.302	1.096
MAE	0.886	0.983	0.805
Overall metrics			
Mean residual distance	1.787 mm		
Mean 3D deviation	1.787 ± 0.988 mm		

4.7 Knowledge Distillation (Teacher → Student)

A smaller student MLP was trained to mimic the predictions of the larger teacher model, with the goal of reducing computational cost for possible deployment on a microcontroller.

Although the compressed model was much lighter, its accuracy dropped noticeably compared to the teacher. We did not directly test how large of an MLP could realistically run on devices like the ESP32 or Arduino Nano, so it is still unclear whether a better balance between size and accuracy could be achieved. For practical use, it may be more reasonable to target slightly more powerful hardware, such as a Raspberry Pi Zero 2 or higher, which can handle larger models while remaining compact and low power. One better suited setup to develop these models could be integrating something like a Raspberry Pi 5 running a JupyterLab Docker Container that could train models during operation of the robot.

4.7.1 Quantization and Deployment Readiness

After training the student model, post-training quantization was applied to convert it into an 8-bit TensorFlow Lite version. Despite the student network performing noticeably worse than the teacher, the quantized model retained almost the same accuracy as its float version while achieving a much smaller file size. This shows that quantization itself did not significantly degrade performance and can be an effective step for deployment, even if the overall results are still limited by the capacity of the student model.

5 Conclusion

The experimental work carried out throughout this project demonstrated that data-driven intelligent modeling techniques can accurately estimate the end-point position of a soft continuum robotic joint from magnetic sensor data. Among all models tested, the deep Multilayer Perceptron architectures showed the most consistent and precise performance, achieving sub-millimetre-level errors for the horizontal coordinates and less than 2 mm of mean 3D deviation overall.

While classical fuzzy approaches such as TSK and ANFIS provided interpretable models with reasonable accuracy, the neural-based methods clearly outperformed them in terms of precision and generalization capability. The use of feature selection and normalization proved essential to stabilize training and reduce redundancy, whereas hierarchical and autoencoder-based approaches did not yield significant gains compared with direct MLP regression.

The final specialist model with cross-expert inputs achieved the best overall accuracy (mean residual distance of 1.79 mm, RMSE $\approx$ [1.13, 1.30, 1.10] mm), capturing nonlinear sensor relationships and inter-coordinate dependencies effectively. These results validate the potential of neural models as reliable estimators.

Future work could focus on developing a fuzzy control system based on the trained data-driven models. Such a controller would allow not only accurate position estimation but also real-time adjustment of the robot's actuation through interpretable fuzzy rules, enhancing stability, adaptability, and safety during operation. Integrating the learned mappings into a fuzzy control framework would therefore represent a natural continuation of this work toward achieving closed-loop control of soft continuum robots.

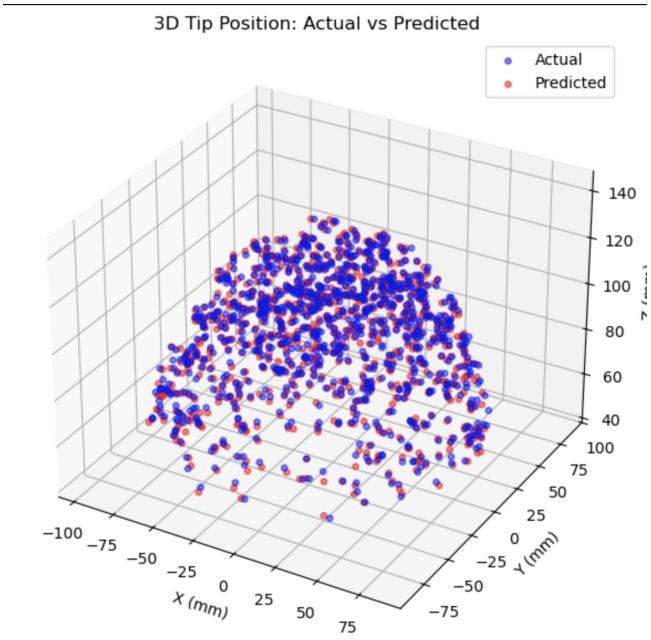

Figure 1: 3D Plot of Actual manipulator coordinates vs Predicted

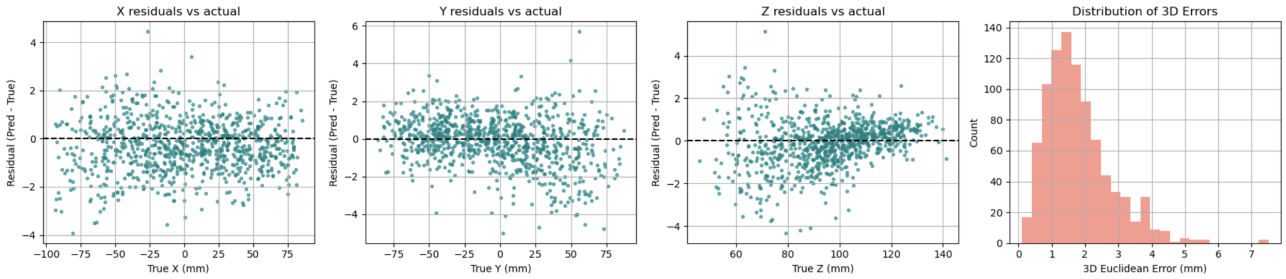

Figure 2: Residuals and Distribution of 3D Euclidean Error

6 GitHub Repo

https://github.com/Pereira98/IS_87172_113219