Highlights

Infrared-Based Orientation Estimation on Pololu $3\pi+$: Evaluating Mathematical and Deep Neural Network Models

2825659, 2821397, 2736690, 2768303

- Developed a low-cost infrared-based method for absolute orientation estimation.
- Designed a circular greyscale gradient enabling unique reflectance signatures.
- Implemented a mathematical decoder using filtered sinusoidal curve fitting.
- Implemented a lightweight neural network for end-to-end angle prediction.
- Compared accuracy, memory use and computational efficiency of both methods.
- Demonstrated reliable 360° orientation estimation on the Pololu $3\pi+$ robot.

Infrared-Based Orientation Estimation on Pololu $3\pi+$: Evaluating Mathematical and Deep Neural Network Models

2825659[a], 2821397[b], 2736690[c] and 2768303[d]

University of Bristol, Queens Building, University Walk, Bristol, BS8 1TR, United Kingdom

ARTICLE INFO

Keywords:
Angle Estimation
Machine Learning
Neural Network
Pololu $3\pi+$
Curve-Fitting
Digital Signal Processing
Gradient Pattern

ABSTRACT

Accurate orientation estimation is essential for mobile robots, yet common sensing methods such as encoders and IMUs accumulate drift over time. As a low-cost alternative that emulates the behaviour of an optical rotary encoder, this work employs the robot's built-in infrared (IR) reflectance sensors together with a custom circular greyscale gradient to infer orientation directly from reflectance patterns. Two approaches are evaluated: a mathematical curve-fitting method and a neural network-based model. The curve-fitting approach filters IR data using cascaded Chebyshev Type II IIR biquads and fits a sinusoidal model via the Levenberg-Marquardt algorithm, while the neural network uses a compact multilayer perceptron to perform end-to-end angle prediction. Experimental results show that both methods achieve comparable average accuracy, with the neural network offering improved performance under darker conditions, whereas the curve-fitting method remains more efficient in memory and computation.

1. Introduction

The present study investigates a low-cost approach for orientation sensing that uses the robot's built-in infrared (IR) reflectance sensors. To enable these sensors to provide orientation information, a custom circular greyscale gradient is placed beneath the robot, enabling the bottom-mounted IR sensors to capture reflectance patterns that correspond uniquely to different orientations.

Two learning-based approaches, a traditional curve-fitting model and a neural network, are evaluated for their ability to map these patterns to an estimated robot angle. The study aims to determine whether such techniques can provide reliable orientation estimates while remaining suitable for lightweight, low-cost robotic platforms.

1.1. Background

The platform used in this study is the Pololu $3\pi+$ 32U4 OLED [6], a compact and efficient differential-drive mobile robot. In addition, the robot is equipped with quadrature wheel encoders and a set of five bottom-mounted infrared (IR) reflectance sensors, as shown in Figure 1. These sensors are arranged in an arc along the front edge of the chassis and are utilised in this experiment to capture the greyscale pattern beneath the robot. For reference in this study, the sensors are labeled sequentially from S1 to S5, starting from the left most sensor when the robot is oriented facing forward.

2. Hypothesis

It is hypothesised that learning-based regression methods offer a viable alternative for estimating robot orientation from low-cost IR reflectance signals. Furthermore, we posit that a neural-network decoder is likely to demonstrate improved fidelity compared to fixed curve-fitting models, specifically in regions characterised by saturated sensor responses.

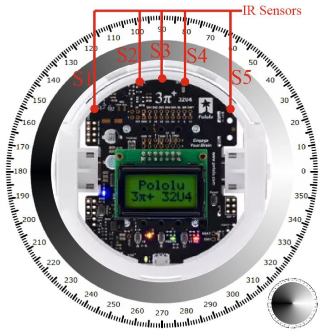

Figure 1: Pololu $3\pi+$, showing the five infra-red sensors (S1-S5), [6].

3. Implementation

3.1. Gradient Formulation

Let the printed pattern be described by a continuous, periodic gradient-generating function:

$$G : \theta \mapsto G(\theta) \in [0, 255],$$

Where $G(\theta)$ is the greyscale intensity at angular coordinate $\theta \in [0, 2\pi)$.

Each IR sensor reading provides a normalised measurement of a local average of the gradient pattern around its corresponding angle θ_i:

$$R_i(\theta_i) = \text{SensorResponse}(G(\theta_i)) \in [0, 1], \qquad i = 1, \ldots, 5.$$

Since the robot rotates as a rigid body, the robot's orientation θ is defined by the centre sensor angle θ_3 . The full sensor reading vector at θ is given by: $\dot{R}(\theta) = [R_i(\theta_i)]^T, \theta \equiv \theta_3$.

The objective is to learn a mapping, $\Phi : \dot{R}(\theta) \longmapsto \theta$, such that the robot can infer its absolute orientation directly from optical signals.

3.2. Gradient Pattern Design

We designed a *symmetric clipped triangular pattern*, defined as:

$$G(\theta) = \begin{cases} 255 - \alpha \frac{255}{\pi}\theta, & 0 \le \theta < \pi, \\ 255 - \alpha \frac{255}{\pi}(2\pi - \theta), & \pi \le \theta < 2\pi, \end{cases}$$

where $\alpha = 0.875$.

This formulation ensures periodicity without discontinuities and limits the intensity range with clipping factor α.

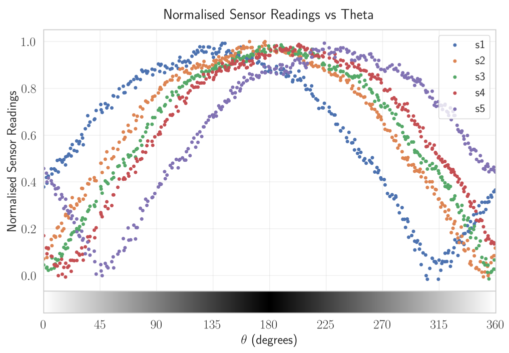

Figure 2: Normalised sensor readings ($s1$–$s5$) vs. orientation θ. The symmetric gradient strip is visualised at the bottom. The phase offsets between sensors resolve the angular ambiguity inherent in the symmetric pattern.

The design choices, specifically the symmetric geometry and the clipping factor, are empirically justified by the sensor response characteristics observed in Figure 2.

3.2.1. *Intensity Clipping*

The clipping factor α is necessary to prevent sensor saturation in the darkest regions ($\theta \approx 180°$). As shown in Figure 2, sensor $s3$ exhibits non-linear flattening even at this reduced intensity; a full black level ($\alpha = 1.0$) would cause hard saturation, creating "dead zones" with zero gradient information. Consequently, $\alpha = 0.875$ was selected to maximise contrast while maintaining the sensors within their effective dynamic range.

3.2.2. *Disambiguation via Phase Offsets*

Although the symmetric pattern implies $G(\theta) = G(2\pi - \theta)$, the physical angular spacing of sensors $s1$–$s5$ generates phase-shifted response curves (Figure 2). These offsets ensure that the aggregate state vector $\dot{R}(\theta)$ remains unique for every orientation, thereby resolving the geometric ambiguity and enabling a well-defined inverse mapping.

3.2.3. *Gradient Printing*

The final symmetric gradient is shown in Figure 3.

In this experiment, the gradient was printed on standard A4 paper. Evaluating the effects of different paper types or materials is considered outside the scope of this study.

3.3. Method I: Mathematical Approach

3.3.1. *Filtering*

The IR readings are being constantly sampled, however are intrinsically prone to slight fluctuations, abetted by ADC

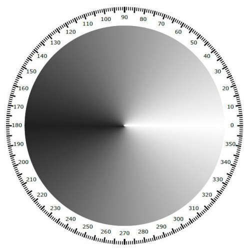

Figure 3: Final Symmetric Gradient

(Analogue to Digital Converter) quantisation. A series of filters were designed, in order to smooth the raw input data. To do this, Matlab Filter Designer was used to for computation of various Infinite Impulse Response (IIR) Low-pass Chebyshev II filters [9]. The performance was evaluated considering artefact removal, memory usage, filter speed, signal distortion, and phase shift.

Attentuations A_{pass} and A_{stop} were set at 0.5 dB and 60 dB respectively, 0.5 dB passband ripple was chosen as it keeps the signal almost undistorted, while a 60 dB stopband attenuation strongly suppresses unwanted noise, giving a sharp, clean filter response without excessive order or instability.

Each of the IR sensors were sampled by the ADC at a frequency of 20 Hz, giving the Nyquist Limit as 10 Hz [8], in order to avoid aliasing. The frequencies F_{pass} and F_{stop} were varied and implemented, as to measure the responses.

It was found that an F_{pass} of 4 Hz, and a F_{stop} of 7 Hz, offered the best result, in terms of the aforementioned metrics, the frequency response can be found in Figure 4. The coefficients for such a filter were generated within Matlab Filter Designer, with three, second order cascaded biquads.

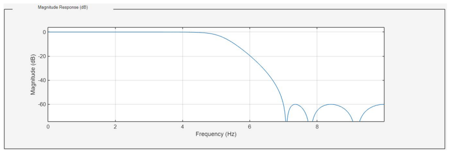

Figure 4: Frequency Response of the IIR Chebshev II Filter, at 4 Hz F_{pass} and 7 Hz F_{stop}, sampled at 20 Hz .

The following equation is the standard second-order IIR biquad transfer function [3], the coefficients from the filter designer can be used to construct the filter biquads:

$$H_i(z) = g_i \times \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{Y_i(z)}{X_i(z)}$$

As stated previously, the coefficients produce three cascading biquads. The outputs of these, feed forward into the input of the next biquad. The final output is the product of the previous difference equations:

$$H(z) = \prod_{i=1}^{N} H_i(z) = \prod_{i=1}^{N} \frac{Y_i}{X_i}$$

This can be derived from the biquad equation, wherein g_i is the gain of the system, b_i coefficients are the feed-forward elements, and the a_i coefficients are the feedback coefficients:

$$y_i[n] = g_i(b_0 x_i[n] + b_1 x_i[n-1] + b_2 x_i[n-2]) \\ - a_1 y_i[n-1] - a_2 y_i[n-2]$$

Without the a-coefficients, the filter is a Finite Impulse Response (FIR) filter. FIR filters are always stable, zero phase distortion, however IIR filters are more computationally efficients, and are able to achieve sharper roll-offs [7]. The predominant issue, fundamental to IIR filters, are that they introduce phase distortion, due to the inherent feed-forward b-coefficients.

With respect to this application, the process is:

- *Input IR Reading:* $x_1[n]$
- *Feed-Forward Process:* $x_3[n] = y_2[n]$, $x_1[n] = y_1[n]$
- *Feedback Process:* $x_i[n-(k+1)] = x_i[n-k]$ and $y_i[n-(k+1)] = y_i[n-k]$, $k \in \{0, 1\}$, $i \in \{1, 2, 3\}$.
- *Output IR Reading:* y_3[n] / R_i

3.3.2. *Curve Fitting*

Once the data has been filtered, to attenuate high-frequency noise, a function can be fitted to the data, to extrapolate an angle from the IR reading. Despite the symmetric gradient ideally outputting a triangular wave, when readings were plotted over time, the output most closely resembled the function $y = |\sin(x)|$.

The general form of a sinusoid can be fitted to this function, wherein the parameters can be tuned to the filtered data, to create a fitted curve.

$$R_i = A_i \sin(B_i \theta_i + C_i) + D_i,$$

where i = Sensor Index, θ_i = Angle, R_i = IR Reading, A_i = Amplitude, B_i = Frequency, C_i = Phase, D_i = Offset.

The general model was fitted using the Levenberg-Marquardt algorithm. This non-linear least squares method was chosen for its robust convergence properties[1].

From the Python Script, the variables of Amplitude, Frequency, Phase, and Offset were defined. The inverse of the general form of a sinusoid can then be inverted:

$$\theta_i = \frac{\sin^{-1}\left(\frac{R_i - D_i}{A_i}\right) - C_i}{B_i} = \frac{\pi - \sin^{-1}\left(\frac{R_i - D_i}{A_i}\right) - C_i}{B_i}$$

Each sensor is uniquely fitted using this method, with individual sensors having their own amplitude, frequency, phase, and offset variables.

Two equations are required due to the sin^{-1} function not able to disambiguate either side of the *sine* curve. To solve this, the trigonometric identity of $\sin(\theta) = \sin(\pi - \theta)$ is introduced, with region-based sensor selction able to perform the disambiguation.

Figure 5 shows comparison between the outputs of each stage of the mathematical approach signal processing, including the raw data, filtered data, and fitted curve. The IIR filter response indicates an introduction of phase distortion around the angles 0°-125°, and 260°-360°. As a consequence of the signal warping, the curve fitting functionality

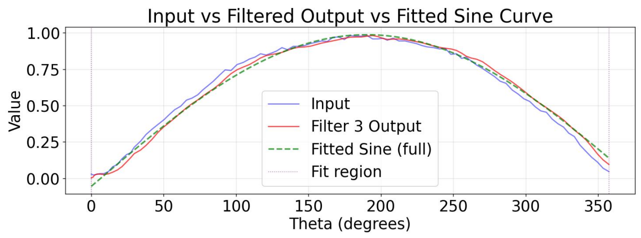

Figure 5: Comparison between the raw sensor input, the filtered signal (third-stage output), and the fitted sinusoidal model. The dashed lines indicate the angular region used for curve-fitting.

is shifted within these regions. Conversely, at the peak of the curve, the fitted region closely follows the filtered data, without being effected by phase distortion.

3.3.3. *Region-Based Sensor Selection*

To resolve the angular ambiguity inherent in the inverse sinusoidal mapping, the system leverages the spatial distribution of the sensor array. Since the gradient pattern is symmetric, a single reflectance value R_i corresponds to two possible orientations. We differentiate between these by comparing the readings from the outermost sensors, S1 and S5. If the reading from S1 exceeds S5, the robot is determined to be in the first half of the gradient (0–π), allowing the system to select the correct branch of the inverse sine functions.

Once the half-rotation is established, the specific angle is computed using the sensor that offers the highest linearity for that sector. As illustrated in Figure 6, the rotational range is divided into four regions, assigned to S3, S4, S5, and S1 respectively. This ensures that the final estimation relies solely on the sensor with the steepest gradient response, avoiding the noise-prone peaks and troughs of the sinusoidal signal.

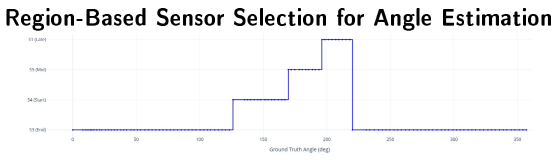

Figure 6: Region-based mapping of the circular greyscale gradient showing which sensor (S3, S4, S5, or S1) provides the most reliable reading for angle estimation across different segments of the rotation range.

S2 was omitted from this selection pool as it provided no distinct accuracy advantage over the other sensors.

The full pipeline process of online training, to onboard angle estimation can be seen in Figure 7.

3.4. Method II: Neural Network Approach

To approximate the mapping $\Phi : \dot{R}(\theta) \longmapsto \theta$, a compact Multilayer Perceptron (MLP) was adopted, designed to be both expressive and suitable for deployment on a resource-constrained microcontroller.

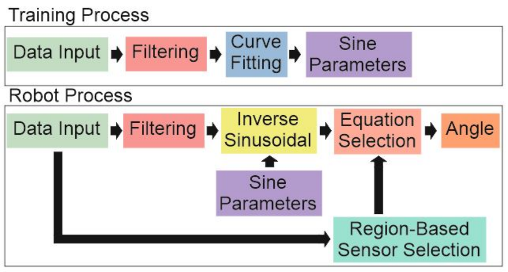

Figure 7: Complete process of curve-fit online training, with onboard angle estimation, including subprocesses.

3.4.1. Input and Output Representation

The model directly adapts the normalised readings from the five IR sensors as input features, $x = \dot{R}(\theta) \in \mathbb{R}^5$.

Instead of regressing θ directly, the network predicts the *sine* and *cosine* of the angle: $\hat{y} = [\hat{s}, \hat{c}]^\top \approx [\sin\theta, \cos\theta]^\top$.

This choice is motivated by:

- *Periodicity and continuity:* θ is periodic; jumping from 2π to 0 introduces artificial discontinuities in a direct regression. The pair $(\sin\theta, \cos\theta)$ lies on the unit circle and varies smoothly.
- *Symmetry handling:* small changes in θ near the seam do not cause large changes in $(\sin\theta, \cos\theta)$, making the learning problem easier.
- *Geometric consistency:* enforcing $\hat{s}^2 + \hat{c}^2 \approx 1$ encourages physically meaningful predictions.

At inference time, the recovered angle is:

$$\hat{\theta} = \operatorname{atan2}(\hat{s}, \hat{c}) \in [-\pi, \pi),$$

3.4.2. Loss Function Design

Given a dataset of pairs $(x^{(n)}, y^{(n)})$, where

$$y^{(n)} = [\sin\theta^{(n)}, \cos\theta^{(n)}]^\top.$$

The model is trained to minimise a weighted composite loss function $\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{unit}}\mathcal{L}_{\text{unit}}$, where $\mathcal{L}_{\text{MSE}}$ is the Mean Squared Error between the predicted and ground-truth vectors:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{n=1}^{N}\left\|\hat{y}^{(n)} - y^{(n)}\right\|_2^2,$$

and $\mathcal{L}_{\text{unit}}$ is a regularisation term that penalises deviations from the unit circle:

$$\mathcal{L}_{\text{unit}} = \frac{1}{N}\sum_{n=1}^{N}\left(\|\hat{y}^{(n)}\|_2^2 - 1\right)^2.$$

The unit-circle term encourages $\hat{s}^2 + \hat{c}^2 \approx 1$, stabilising the angle reconstruction via atan2.

3.4.3. Network Topology

The final architecture is a fully-connected *Feed Forward Neural Network* with the following structure, shown in Figure 8. Concretely, the network computes:

$$\begin{aligned} h_1 &= \mathrm{ReLU}(W_1 x + b_1), & W_1 &\in \mathbb{R}^{h_1\times 5},\ b_1 \in \mathbb{R}^{h_1}, \\ h_2 &= \mathrm{ReLU}(W_2 h_1 + b_2), & W_2 &\in \mathbb{R}^{h_2\times h_1},\ b_2 \in \mathbb{R}^{h_2}, \\ \hat{y} &= W_3 h_2 + b_3, & W_3 &\in \mathbb{R}^{2\times h_2},\ b_3 \in \mathbb{R}^{2}. \end{aligned}$$

Where h_1, h_2 are neuron counts of the two hidden layers, and $\mathrm{ReLU}(z) = \max(0, z)$ is applied elementwise [5].

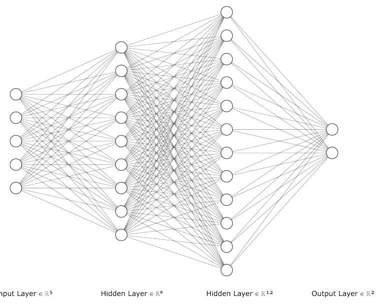

Figure 8: Neural network implemented within the robot.

Design considerations:

- *Depth:* two hidden layers provide greater expressive power than a single layer at similar parameter count, which is useful for capturing the nonlinear response of the sensors and the gradient.
- *Width:* a small width keeps the total number of parameters low, making it feasible to store in flash memory and evaluate in real time on the microcontroller.
- *Activation:* ReLU is chosen for simplicity and good empirical performance on low-dimensional regression; its piecewise-linear nature matches well with the gradual gradient pattern.

3.4.4. Hyperparameter Search

Before fixing the network architecture, a small-scale hyperparameter search was conducted to evaluate the effect of hidden-layer widths, and the regularisation coefficient λ_{unit} on prediction accuracy. Figure 9 summarises the results, reported as $\log_{10}(\text{MSE})$.

Hidden layer sizes. A grid search over $h_1, h_2 \in \{1, \dots, 16\}$ revealed a clear low-error basin for moderately sized networks. Very small networks underfit, whereas excessively large ones provide no meaningful benefit. The best-performing region lies around: $h_1 \approx 9$–16, $h_2 \approx 12$–16, indicating that a compact, but sufficiently wide, architecture is needed to capture the nonlinear sensor–angle relationship.

Regularisation strength. A logarithmic sweep over $\lambda_{\text{unit}} \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ showed that performance is stable for small λ, with a minimum near $\lambda_{\text{unit}} = 10^{-1}$. Larger values ($\lambda_{\text{unit}} \geq 1$) degrade performance sharply due to over-regularisation.

Selected configuration. Based on these trends, the model adopts the following hyperparameters: $h_1 = 9, h_2 = 12, \lambda_{\text{unit}} = 0.1$.

This configuration sits inside the low-error basin while remaining lightweight enough for deployment on the microcontroller.

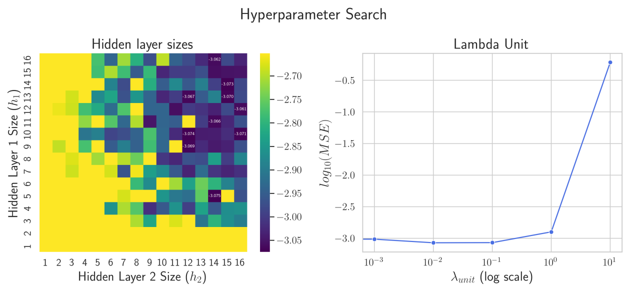

Figure 9: Hyperparameter results for neural network model.

3.4.5. Training Procedure

The collected dataset was randomly partitioned into an 80% training set and a 20% validation set. The network was trained using standard backpropagation with the Adam optimiser [2] to regress the target vector $[\sin\theta, \cos\theta]$.

We utilised a batch size of 20 and a learning rate of 1×10^{-3}. To prevent overfitting, early stopping was enabled with a patience of 30 epochs [4], monitoring the angular error on the validation set. Under these conditions, the model consistently converged within 200 epochs ($\approx$ 4 seconds).

3.4.6. Deployment on the Pololu 3π+

After training in Python, the network weights $\{W_1, b_1, W_2, b_2, W_3, b_3\}$ were exported to a C header file as constant arrays. The network weights contains 200 stored parameters totalling to 800 bytes, which accounts for 31.25% of the robot's Random Access Memory (RAM). The forward pass is reimplemented in C++ on the Pololu 3π+ 32U4, mirroring the equations in section 3.4.3. At runtime, the robot:

1. Reads 5 IR sensor values and normalises them, obtaining $x \in \mathbb{R}^5$
2. Evaluates the MLP to obtain $[\hat{s}, \hat{c}]$
3. Recovers $\hat{\theta} = \text{atan2}(\hat{s}, \hat{c})$

4. Experiment Method

4.1. Data Acquisition

Before the data acquisition process, IR readings were normalised to remove sensor-specific biases and place all five sensors on a comparable scale. For both modelling approaches, the normalised sensor readings (S1-S5) were paired with the corresponding orientation angles obtained from the encoders. The encoder measurements were considered reliable within 1° for a single rotation; however, accuracy degrades over time as noise and wheel slip accumulate.

To mitigate drift and ensure that the instantaneous centre of rotation remained fixed during data collection, an external jig was used to hold the robot in place. This setup prevented unintended motion and stabilised the robot, enabling more reliable sensor and encoder measurements. Any error or bias that would occur without this support is not considered in this study. Additionally, due to memory limitations, data were recorded only once per full rotation, resulting in approximately 80 data points per sensor.

4.2. Performance Metrics

The following metrics are designed for a comprehensive comparison of the performance of NN and CF models:

- *Qualitative Analysis:*
  - Global Error Distribution: Histogram of angular errors across the entire dataset.
  - Angular Dependence and Crossover Saturation: Error trends as a function of ground-truth angle, with focus on the 150°–210° region where sensor saturation occurs.
- *Quantitative Metrics:*
  - *Dynamic Memory Usage:* The amount of RAM required by each method.
  - *Mean Absolute Angular Error:* The average absolute deviation between the estimated angle and the ground truth across all samples. Calculated by $\bar{x} = \frac{1}{N}\sum_{i=1}^{N}|\theta_i - \theta_{ground}|$, where N = total number of samples.
  - *Maximum Error:* Captures the worst-case estimation failure, indicating robustness under challenging sensing conditions.
  - *Training & Calibration Time:* Evaluates how long each method takes to configure or learn its model before deployment.

5. Results

5.1. Angular Error Analysis

The comparative analysis of the Neural Network (NN) and the baseline Curve-Fitting (CF) models reveals a distinct performance advantage for the learning-based approach, particularly in certain challenging regions of the gradient.

5.1.1. Global Error Distribution

As shown in the Angular Error Histogram (Fig. 10), the NN achieves superior global precision with a sharp distribution peak in the 0°–1° range. In contrast, the CF model exhibits a flatter distribution with a "heavier tail" accumulating between 2°–5°, indicating that the learning-based approach provides greater consistency across the majority of the dataset.

5.1.2. Angular Dependence and Crossover Saturation

Figure 11 highlights that while both models perform comparably in linear regions (e.g., 0°–120°), performance diverges significantly in the 150°–210° gradient crossover area, where sensor readings saturate, as discussed in section 3.2.1.

- **Curve-Fitting (CF):** The CF error spikes severely (mean peak $\approx$ 8°) due to reliance on *empirically thresholded sensor selection*. The algorithm fails to cleanly switch sensors or extract valid data when the primary sensors saturate.

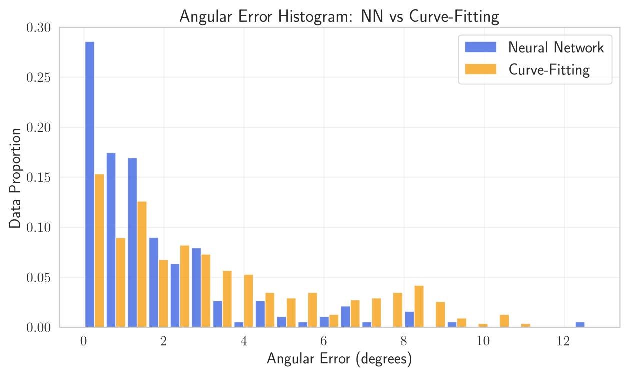

Figure 10: Comparison of the angular estimation errors for neural network model and curve-fitting baseline.

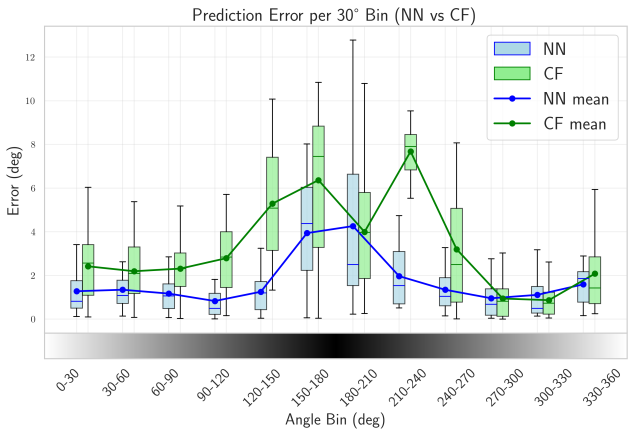

Figure 11: Comparison of the errors between the two methods, in relation to ground truth, with 30 degree bins.

- **Neural Network (NN):** The NN mitigates this degradation (mean error ≈ 4°) through *automatic sensor fusion*. By using all five sensor readings as input features, the NN aggregates residual information to resolve orientation, avoiding the pitfalls of binary sensor selection.

Despite superior mean accuracy, the NN displays larger outliers in Fig. 11. This suggests it is slightly less robust to random sensor noise than the CF model. Without the strict physical constraints inherent to the curve-fitting algorithm, the high-dimensional function approximated by the NN is more susceptible to stochastic noise in the input vector.

5.2. Quantitative Metrics Comparison

Table 1 quantifies the trade-offs between the two decoding approaches. The Neural Network provides superior accuracy, reducing the Mean Absolute Error by nearly half (1.79° vs. 3.32°) and significantly improving performance in the gradient crossover region (60° vs. 120°).

However, this precision incurs higher computational and operational costs. The NN requires substantially more dynamic memory (52% vs. 12%) and demands double the calibration and training time (1 hour vs. 30 mins). Additionally, the slightly higher maximum error for the NN (12.78°) corroborates the earlier observation regarding its sensitivity to outliers compared to the curve-fitting baseline.

Table 1
Performance comparison of the two decoders.

Metric	Curve-Fit	Neural Net
Dyn. Memory (kB)	308	1334
Dyn. Memory (%)	12%	52%
Mean Abs. Error (deg)	3.32	1.79
Max Error (deg)	11.4	12.78
Grad. Crossover Error (deg)	120	60
Train & Calib. Time	≈ 30 (mins)	≈ 1 (hrs)

6. Conclusion

The experimental results largely validate the primary hypothesis, confirming that the Neural Network (NN) provides superior tracking fidelity, particularly within the challenging saturation regions. By leveraging *automatic sensor fusion*, the NN effectively halves the global Mean Absolute Error (1.79° vs. 3.32°) and prevents the massive error spikes observed in the CF model when sensor gradients flatten.

However, while the hypothesis regarding accuracy holds true, the results also highlight necessary trade-offs. The precision of the NN incurs a 4× increase in dynamic memory usage (52% vs. 12%) and exhibits higher sensitivity to stochastic noise. Consequently, while the NN is the superior choice for handling non-linearities as hypothesised, the CF approach remains a relevant, lightweight alternative for strictly memory-constrained applications where extreme precision in crossover regions is secondary.

References

[1] Gavin, H.P., 2020. The Levenberg–Marquardt Algorithm for Nonlinear Least-Squares Curve-Fitting Problems. Technical Report. Duke University, Department of Civil and Environmental Engineering. URL: https://people.duke.edu/~hpgavin/lm.pdf. available online at https://people.duke.edu/~hpgavin/lm.pdf.

[2] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. CoRR abs/1412.6980. URL: https://api.semanticscholar.org/CorpusID:6628106.

[3] Lee, F.M.S., Antoniou, A., 2000. Digital Filters: Analysis, Design and Applications. McGraw-Hill, New York. URL: https://users.dimi.uniud.it/~antonio.dangelo/MMS/materials/DigitalFilters.pdf. accessed: 2025-11-28.

[4] Morgan, N., Bourlard, H., 1989. Generalization and parameter estimation in feedforward nets: Some experiments. Advances in neural information processing systems 2.

[5] Nair, V., Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814.

[6] Pololu Corporation, 2025. 3pi+ 32u4 oled robot - standard edition (30:1 mp motors), assembled. https://www.pololu.com/product/4975. Accessed: 2025-11-27.

[7] Proakis, J.G., Manolakis, D.G., 1996. Digital Signal Processing: Principles, Algorithms, and Applications. 3 ed., Prentice Hall, Upper Saddle River, NJ. URL: https://uvceee.wordpress.com/wp-content/uploads/2016/09/digital_signal_processing_principles_algorithms_and_applications_third_edition.pdf. accessed: 28 November 2025.

[8] Shannon, C.E., 1949. Communication in the presence of noise. Proceedings of the IRE 37, 10–21. doi:10.1109/JRPROC.1949.232969.

[9] Smith, S.W., 1999. The Scientist and Engineer's Guide to Digital Signal Processing. 2 ed., California Technical Publishing, San Diego. URL: https://ia801301.us.archive.org/23/items/

GuideToDigitalSignalProcessing/Guide%20To%20Digital%20Signal%20Processing.pdf. accessed via Internet Archive. See p. 347.