Detailed Mathematical Documentation
Adaptive Parameter Feedback Tuning

Team Flare Net

October 2025

Abstract

This document provides a rigorous, end-to-end mathematical description of the adaptive parameter system used in an anomaly-detection pipeline. It covers the detection maths (anomaly-map thresholding used in the model core), the mapping between the user-facing `percent_threshold` and internal parameter k, per-parameter discrete update rules driven by feedback types (false positive / false negative / bbox resize / severity / category change), continuous-time approximations, convergence and bounds, geometric calculations (area / overlap / aspect ratio), confidence computations, and worked numerical examples. Each mathematical block is followed by a plain-language explanation and a small numeric example.

1 Notation

X	image (pixel intensities)
W, H	image width, height
$A_{\text{image}} = W \cdot H$	image area (pixels)
$\mathcal{M}(x, y)$	anomaly map value at pixel (x, y) (model output)
$\mu_{\mathcal{M}}, \sigma_{\mathcal{M}}$	mean and std of anomaly map $\mathcal{M}$
T_{percent}	user-facing percent threshold (0–100)
k	internal multiplier used for map thresholding (mapped from T_{percent})
f_{area}	`min_area_factor` (fraction of image area)
$A_{\min} = f_{\text{area}} \cdot A_{\text{image}}$	absolute min area (pixels)
$B = (x, y, w, h)$	bounding box with width w, height h
$A_{bbox} = w \cdot h$	bbox area (pixels)
A_{blob}	connected component (blob) area (pixels)
$A_{\text{frac}} = A_{blob}/A_{bbox}$	fraction of bbox covered by blob
R_{frac}	red/orange fraction inside box (for severity)
C	confidence score output for a detection

2 Detection thresholding in the model core

The model produces an anomaly map $\mathcal{M}$ (real-valued). The code uses a statistical threshold of the form:

$$\text{thresh} = \mu_{\mathcal{M}} + k \cdot \sigma_{\mathcal{M}}$$

and a binary mask:

$$\text{mask}(x,y) = \begin{cases} 1 & \text{if } \mathcal{M}(x,y) > \text{thresh}, \\ 0 & \text{otherwise.} \end{cases}$$

Explanation: this is a mean-plus-k-sigma rule; larger k makes the mask stricter (fewer pixels pass).

Example: if $\mu_{\mathcal{M}} = 0.1$, $\sigma_{\mathcal{M}} = 0.05$, and $k = 1.5$, then thresh $= 0.1 + 1.5 \cdot 0.05 = 0.175$. Pixels above 0.175 are marked anomalies.

2.1 Mapping user percent to internal k

Your code defines a mapping:

$$k = 1.1 + \left(\frac{T_{\text{percent}}}{100}\right) \cdot (2.1 - 1.1) = 1.1 + 1.0 \cdot \frac{T_{\text{percent}}}{100}.$$

So equivalently:

$$k = 1.1 + 0.01 \cdot T_{\text{percent}}.$$

Interpretation: $T_{\text{percent}} = 0 \Rightarrow k = 1.1$ (most sensitive), $T_{\text{percent}} = 100 \Rightarrow k = 2.1$ (least sensitive).

Numeric example: If $T_{\text{percent}} = 29$, then $k = 1.1 + 0.29 = 1.39$; the threshold becomes $\mu + 1.39\sigma$.

3 Connected components and minimum area

After mask creation, connected components are found. A component passes only if:

$$A_{blob} \geq A_{\min} = f_{\text{area}} \cdot A_{\text{image}}.$$

Explanation: this removes tiny noisy islands. The factor f_{area} is adapted by feedback.

Numeric example: For $W = 1024, H = 768$ (area $= 786432$ px) and $f_{\text{area}} = 0.0005$, we have $A_{\min} = 393.216$ px. Any blob smaller than 393 px is discarded.

4 Geometric rules: overlap / aspect ratio / area fraction

4.1 Overlap with center region

Given two center coordinates defining a central region (used in code):

$$\text{center box } C = [x_0, y_0, x_1, y_1].$$

Overlap area between detection bbox B and C is:

$$\text{overlap} = \max(0, \min(x + w, x_1) - \max(x, x_0)) \cdot \max(0, \min(y + h, y_1) - \max(y, y_0)).$$

Overlap fraction relative to bbox:

$$\text{overlap\_frac} = \frac{\text{overlap}}{A_{bbox}}.$$

4.2 Aspect ratio

$$\text{aspect} = \frac{\max(w, h)}{\max(1.0, \min(w, h))}.$$

The code checks if aspect $\geq r_{\text{wire}}$ to classify wires.

4.3 Area fraction

$$A_{\text{frac}} = \frac{A_{blob}}{A_{\text{image}}} \quad \text{or} \quad \frac{A_{blob}}{A_{bbox}},$$

depending on rule in use.

5 Confidence scoring

A typical linear confidence model used in the code:

$$C = \min(1.0,\ B + \alpha \cdot F)$$

where

- B is the base confidence for category (e.g., 0.6),
- α is a tunable factor (e.g., 0.8),
- F is a normalized feature (e.g., area fraction or brightness, in $[0, 1]$).

Example: Loose joint confidence:

$$C_{\text{loose}} = \min(1.0, 0.6 + 0.8 \cdot A_{\text{frac}}).$$

If $A_{\text{frac}} = 0.2$, then $C_{\text{loose}} = \min(1.0, 0.6 + 0.16) = 0.76$.

6 Feedback-driven discrete parameter updates (exact rules coded)

Below are the exact discrete update rules implemented in `AdaptiveParams`.

6.1 False Negative

$$T_{n+1} = \max(10,\ T_n - 3).$$

$$f_{n+1} = \max(0.0005,\ 0.8 \cdot f_n).$$

Explanation: each false-negative event reduces T by 3 percentage points and reduces minimum-area factor by 20% but not below $5 \cdot 10^{-4}$.

Numeric example:

$$T : 47 \to 44 \quad (47 - 3)$$

$$f : 0.0008 \to 0.00064 \quad (0.0008 \times 0.8)$$

6.2 False Positive

$$T_{n+1} = \min(90,\ T_n + 3).$$

$$f_{n+1} = \min(0.005,\ 1.2 \cdot f_n).$$

Explanation: increases threshold and min area to reduce sensitivity; clamped at given bounds.

Numeric example:

$$T : 35 \to 38 \quad (35 + 3)$$

$$f : 0.0005 \to 0.0006 \quad (0.0005 \times 1.2)$$

6.3 Bounding box resize (bbox_resize)

Let the user resize produce area ratio:

$$r = \frac{A_{\text{corrected}}}{A_{\text{original}}}.$$

If the category contains `loose_joint`:

$$\text{if } r < 0.8 : \quad L_{n+1} = \min(0.20,\ 1.1 \cdot L_n).$$

$$\text{if } r > 1.2 : \quad L_{n+1} = \max(0.05,\ 0.9 \cdot L_n).$$

Explanation: user shrinking box means original box was too loose — tighten the rule by increasing minimal required area; expanding the box relaxes the rule.

Numeric example: if $L_n = 0.10$ and user shrinks box to 60% ($r = 0.6 < 0.8$):

$$L_{n+1} = 1.1 \times 0.10 = 0.11.$$

6.4 Severity change

If user changes severity:

$$S_{n+1} = \begin{cases} \min(0.8,\ S_n + 0.05), & \text{if Faulty} \to \text{Potentially Faulty}, \\ \max(0.2,\ S_n - 0.05), & \text{if Potentially Faulty} \to \text{Faulty}. \end{cases}$$

Here S denotes the red/orange pixel fraction threshold for “Faulty”.

Explanation: adjust the severity threshold up or down in small steps (5%) with clamping to [0.2,0.8].

7 Feedback analysis / matching logic

The feedback handler compares `original_detections` and `user_corrections` by generating detection IDs:

$$\text{id} = x\_y\_w\_h\_index,$$

then:

- If id present only in original: classify as **false positive**.
- If id present only in corrected: classify as **false negative** (user added).
- If present in both: check bbox area ratio, severity change, category change.

Important note: Using x, y, w, h, index may fail to match two boxes that have the same coordinates but different indices or boxes that moved slightly; the code also computes position-change by pixel difference:

$$\Delta_{pos} = |x_{\text{orig}} - x_{\text{corr}}| + |y_{\text{orig}} - y_{\text{corr}}|$$

and treats movements > 10 px as significant.

8 Continuous-time / differential approximation

The discrete update rules can be approximated as a continuous-time ODE when feedback arrives at a rate:

$$\frac{dT}{dt} = -k_{\text{FN}} \cdot \text{FN}(t) + k_{\text{FP}} \cdot \text{FP}(t)$$
$$\frac{df}{dt} = -\lambda_{\text{FN}} \cdot \text{FN}(t) + \lambda_{\text{FP}} \cdot \text{FP}(t)$$

where FN(t) and FP(t) are instantaneous feedback rates (counts per unit time). Discrete steps correspond to impulses of these ODEs.

Interpretation: the system acts like a control loop: false negatives integrate negative error (push T down), false positives integrate positive error (push T up). The discrete steps use fixed increments $\Delta T = 3$ and multiplicative factor 0.8/1.2 for area.

9 Stability, convergence and clamping

- Percent threshold T is clamped to $[10, 90]$. This prevents runaway sensitivity or insensitivity.
- Min area factor f is clamped to $[5 \times 10^{-4}, 5 \times 10^{-3}]$.
- Severity threshold clamped to [0.2,0.8].

Convergence remarks: under repeated identical feedback (e.g., persistent false negatives), T decreases linearly by 3 each event until lower bound is reached; f decreases geometrically by factor 0.8 each event until lower bound is hit. Geometric decay has faster relative reduction early, then limited by clamp.

10 Worked full numeric trace (your CSV summarized)

Initial (example) parameters:

$$T_0 = 47, \quad f_0 = 0.0008.$$

Sequence of **false negative** events (6 events):

$$T : 47 \xrightarrow{-3} 44 \xrightarrow{-3} 41 \xrightarrow{-3} 38 \xrightarrow{-3} 35 \xrightarrow{-3} 32 \xrightarrow{-3} 29.$$

$$f : 0.0008 \xrightarrow{\times 0.8} 0.00064 \xrightarrow{\times 0.8} 0.000512 \xrightarrow{\times 0.8 \text{ (clamp)}} 0.0005 \quad \text{(clamped)}.$$

One final mixed event (false_positive + false_negative) produced no net T change because both update rules applied (increase by +3 and decrease by -3) or code logic prevented further change once at desired point.

11 Practical recommendations

- Use IoU-based matching (Intersection-over-Union) instead of exact-coordinate+index matching to robustly pair original and corrected boxes:

$$\mathrm{IoU}(B_1, B_2) = \frac{A_{\text{inter}}}{A_{\text{union}}}$$

  and treat IoU > 0.5 as same detection.
- Consider smoothing updates: replace fixed step ± 3 with $T_{n+1} = T_n \pm \eta$ where η is adaptive (e.g., proportional to confidence of feedback or number of consistent feedbacks).
- Track per-user reliability and weight their feedback accordingly (reduces noise from incorrect edits).

12 Summary

- The system uses thresh $= \mu + k\sigma$ to produce initial masks and geometric/color heuristics for classification.
- User feedback maps to small, interpretable discrete parameter updates: additive steps for percent thresholds and multiplicative steps for area factors.
- The percent threshold is mapped to internal k via $k = 1.1 + 0.01 \cdot T_{\text{percent}}$.
- Area thresholds and severity thresholds are bounded to avoid pathological behavior.
- The adaptation loop is simple, explainable, and effective for incremental tuning. Use IoU matching and optional smoothing for more robust production behavior.