HydroSure

Water Analysis API: Real-Time Calibration Architecture

Technical Report

Author: HydroSure Engineering Team
Date: October 30, 2025

Contents

1 Executive summary

This report describes the design and implementation of the **HydroSure Water Analysis API**. The API converts two smartphone photographs — a reference color chart and a reacted test strip — into a validated, human-readable water quality report. The system combines deterministic computer-vision processing with lightweight LLM validation and interpretation to balance accuracy, robustness, and explainability.

Key properties of the system:

- Per-sample, real-time color calibration using a photographed chart to correct for camera and lighting differences.
- A sequential two-step verification (CV segmentation followed by LLM validation) that prevents incorrect matches and reduces false positives.
- A modular LangGraph workflow that isolates responsibilities into testable nodes and enables clear error handling.

2 Goals and scope

2.1 Primary goals

- Produce accurate water parameter estimates from consumer-grade smartphone photos.
- Guarantee that only high-quality pad crops proceed to color matching, avoiding misleading results.
- Provide concise, actionable summaries for end users while exposing raw results for diagnostics and audit.

2.2 Scope and assumptions

The current design assumes a fixed 16-parameter strip layout and a corresponding 16-row reference chart. The pipeline expects the chart and strip to be photographed separately under reasonable lighting; robust handling for extreme lighting or novel chart formats is part of planned enhancements.

3 Technology stack

Table 1 lists the principal components and their responsibilities.

4 High-level pipeline

A concise description of the pipeline executed on each `POST /analyze` call:

1. Client uploads two images: a reference *chart* and a reacted *strip*.
2. The FastAPI server converts uploads to Base64 and starts the LangGraph workflow.
3. The `chart_processor_node` builds a per-sample color map in L∗a∗b space.
4. The `segment_strip_node` proposes 16 pad crops and computes their L∗a∗b means.
5. The `validate_segments_node` sends the crops to GPT-4o for a boolean validity verdict.
6. If all pads are valid, `match_colors_node` uses Delta E 2000 to find the closest chart swatch for each pad.

Technology	Category	Role
FastAPI	Backend framework	Exposes the `/analyze` endpoint and handles multipart uploads.
Uvicorn	ASGI server	Hosts the FastAPI app for asynchronous I/O.
LangGraph	Orchestration	Sequences the pipeline as a stateful graph with conditional branching.
OpenCV	Computer vision	Image decoding, thresholding, contour detection, cropping and sampling.
scikit-image	Color science	Implements CIEDE2000 (Delta E 2000) for perceptual color comparison.
NumPy	Numerical computing	Efficient array processing used across CV operations.
OpenAI API	LLMs	GPT-4o for image-quality validation; GPT-3.5-Turbo for result interpretation.
Pydantic	Data validation	DTOs for requests and responses, ensuring type safety.
python-dotenv	Configuration	Loads environment variables like API keys in development.

Table 1: Primary technologies used

7. The `interpret_results_node` formats the numerical results into a short, actionable summary using GPT-3.5-Turbo.

8. The API returns a structured `AnalysisResponse` JSON object with both the summary and raw results.

5 Node specifications

Each pipeline step is implemented as an independent node with clear inputs, outputs and error semantics. The following subsections describe implementation details and operational notes.

5.1 Chart processor (`chart_processor_node`)

Inputs: Base64-encoded chart image.

- Decode the image and normalize orientation.
- Partition the image into 16 logical rows (one per parameter) and isolate the swatch region (configurable; typical: right 50–70% of the row).
- For each swatch, sample the interior (avoid borders) and compute the mean BGR; convert to $L*a*b$.
- Output: a map from parameter $\rightarrow$ ordered list of calibrated swatch $L*a*b$ vectors.

5.2 Strip segmentation (`segment_strip_node`)

Inputs: Base64-encoded strip image.

- Convert to grayscale and apply Otsu thresholding to create a foreground mask.

- Detect contours; select the largest rectangular contour as the strip and perform a perspective warp if needed.
- Vertically slice the cropped strip into 16 equal regions and sample centers for color measurement.
- Return: 16 Base64 pad crops plus their L*a*b means for matching.

5.3 Pad validation (`validate_segments_node`)

Purpose: Stop bad inputs early.

- Send the 16 pad images (Base64) to GPT-4o with a short, deterministic instruction to return a JSON array of boolean flags.
- The node maps the response to either `ALL_VALID` or `INVALID_FOUND` and provides per-pad diagnostics to the caller.

5.4 Color matching (`match_colors_node`)

- For each pad L*a*b, compute Delta E 2000 against the calibrated swatches for that parameter.
- Select the swatch with the smallest Delta E as the match and report the numeric value along with the Delta E as the `confidence_score` (smaller is better).
- Optional: client-side can convert Delta E to a normalized confidence percentage for display (e.g., using empirical mapping).

5.5 LLM interpretation (`interpret_results_node`)

- Compile the matched numeric values into a compact structure and send to GPT-3.5-Turbo.
- Request a 3–5 sentence summary highlighting abnormal or actionable findings and recommended next steps.
- Preserve the raw matched data for auditability; the LLM summary is meant for human consumption only.

6 Two-step verification: rationale and thresholds

Combining a deterministic CV pass followed by an LLM sanity check reduces both false positives and edge-case failures:

- CV quickly identifies candidate pads; it has predictable failure modes (e.g., blur, occlusion).
- GPT-4o evaluates subjective quality attributes that are expensive or brittle to encode as rules.

Suggested pragmatic thresholds (tunable):

- Laplacian variance threshold for acceptable sharpness: e.g., 100 for typical smartphone images.
- Delta E acceptance guidance: $\Delta E 5$ is a strong match; $5 \leq \Delta E 12$ is acceptable with lower confidence; larger values require manual review.

7 Computer vision and color science notes

7.1 Why L∗a∗b and Delta E

Human color perception is not linear in RGB. Converting to L∗a∗b and using CIEDE2000 (Delta E 2000) better reflects perceived differences and leads to more reliable matching across devices.

7.2 Robustness techniques

- Preprocessing: apply adaptive histogram equalization for uneven lighting.
- Outlier handling: shrink sample region if the pad contains speckling or artifacts.
- Failure modes: provide clear client errors (e.g., "Re-take photo with plain background") and render small illustrative hints in the mobile UI.

8 AI prompts and deterministic behavior

Keep prompts short, explicit, and factual to reduce variability. Example prompts (cleaned and concise):

```
GPT-4o (validation):
"You will receive 16 base64-encoded images. For each image, return true if
it is a clear, single-color test pad suitable for color sampling; otherwise
return false. Output exactly a JSON array of 16 booleans."

GPT-3.5-Turbo (interpretation):
"You are an experienced water-quality analyst. Given the following list of
parameters and numeric values, produce a 3-5 sentence summary that highlights
any values outside typical ranges and suggests next steps. Keep language
concise and actionable."
```

9 API schemas and example response

9.1 Request

```
POST /analyze
multipart/form-data: strip_file (file), chart_file (file)
```

9.2 Example response

```
{
  "strip_id": "6b052cc5-8a05-40fe-be92-df1e1e930d31",
  "timestamp": "2025-10-30T07:47:00.289263Z",
  "ai_summary": "Overall, your water is generally suitable for household use. pH
is slightly acidic at 6.0 and nitrate levels are elevated; we recommend a
confirmatory laboratory test for nitrate and checking for corrosion in
plumbing.",
  "results": [
    {
      "parameter": "pH",
      "matched_value": "6.0",
      "unit": "pH",
      "confidence_score": 4.2,
      "matched_hex": "#53524f"
    },
    {
      "parameter": "HARDNESS",
      "matched_value": "50",
      "unit": "ppm",
      "confidence_score": 2.1,
      "matched_hex": "#626362"
    }
    // ... additional results ...
  ]
}
```

10 Demo flow and UX guidance

For a smooth demo and pilot experience:

1. Instruct users to photograph the reference chart flat, centered and in good light.
2. Photograph the strip after the reaction time, placed on a plain, neutral background.
3. Provide immediate feedback: display the LLM summary first and stream the detailed results as they become available.

11 Deployment and operations

- Host the FastAPI app with Uvicorn behind NGINX for TLS and load balancing.
- Store secrets (OpenAI keys, service credentials) in a secure vault and rotate periodically.
- Capacity planning: scale horizontally on request concurrency; LLM latency will dominate user-perceived response time.

12 Testing and monitoring

- Unit tests for each CV node using a labeled corpus that includes edge cases (blur, occlusion, nonstandard charts).
- Integration tests that mock OpenAI responses and assert correct branching and error codes.
- Monitoring: track per-parameter Delta E distributions, validation failure rates and LLM latency. Alert on sudden shifts in these metrics.

13 Roadmap and enhancements

Immediate and medium-term improvements:

- Convert Delta E scores to calibrated confidence probabilities using empirical data.
- Implement on-device pre-checks (blur detection and background masking) to improve UX and reduce server workload.
- Add automatic detection for alternative chart layouts and dynamic mapping to parameters.
- Store anonymized pad crops (with user consent) for periodic human review and model retraining.

14 Appendix A: implementation notes

14.1 Strip segmentation (pseudocode)

```
# decode base64 -> cv2 image
# convert to grayscale
# apply Otsu threshold
# find contours -> select largest rectangular contour
# warp/crop and slice vertically into 16 pads
# compute mean in lab space for each pad
```

14.2 Prompt examples

See the "AI prompts and deterministic behavior" section for concise prompt text.

Document version: 1.0
Prepared by: HydroSure Engineering Team