# GPR-Former and ROOFER360

A technical whitepaper on context-aware rooftop moisture detection and end-to-end 360° roof defect inspection.

**Building Diagnostic Robotics, Inc.**

Document version: 1.0
Date: 2026-01-06
For technical evaluation and partner discussions.

| | |
|---|---|
| **What this paper covers** | 1) GPR-Former: subsurface moisture detection on low-slope roofs using GPR data and spatial-context transformers.<br>2) ROOFER360: visual defect detection and best-view extraction from 360° video + LiDAR using LLM-driven detection and optional segmentation.<br>3) How these technologies combine into a multi-modal roof inspection workflow. |

# Table of Contents

# 1. Executive summary

Low-slope roofing systems present two complementary diagnostic challenges: (1) subsurface moisture ingress that is not reliably visible in RGB imagery, and (2) surface-level defects and anomalies that can be visually observed but are difficult to localize, track across long videos, and summarize consistently.

This paper describes two core technologies used in our inspection stack:

- **GPR-Former** - Context-aware moisture detection AI that ingests GPR radargrams and outputs per-location moisture probabilities (and moisture maps). It uses transformer encoders with 2D coordinate-based positional encodings and spatial grouping/windowing to capture non-linear moisture propagation patterns [1]. The current implementation adds a configurable pipeline, deterministic Morton space-filling-curve windowing, and production-oriented evaluation/inference tooling.
- **ROOFER360** - An end-to-end 360-degree roof defect and best-view extraction pipeline that fuses video frames with LiDAR geometry. It uses OpenAI's gpt-5 model for structured bounding-box detection, optional SAM 2 segmentation for masks, 3D projection and clustering across frames, and an automated report packager (HTML + COCO + run summaries).

Together, these systems enable a multi-modal roof inspection workflow that produces moisture maps, visual defect annotations, and inspection-ready best views for reporting and remediation planning.

# 2. Platform context: multi-modal roof inspection

Roof diagnostics is inherently multi-modal. Moisture-related damage can be subsurface and detectable via GPR, while many actionable defects are surface-visible in RGB imagery (and in some cases thermal imagery). Our platform aligns these sensing modalities into a single data-to-report workflow.

Figure 1 summarizes the end-to-end inspection pipeline: data acquisition (GPR, LiDAR, RGB, thermal), AI-based processing, and report generation.

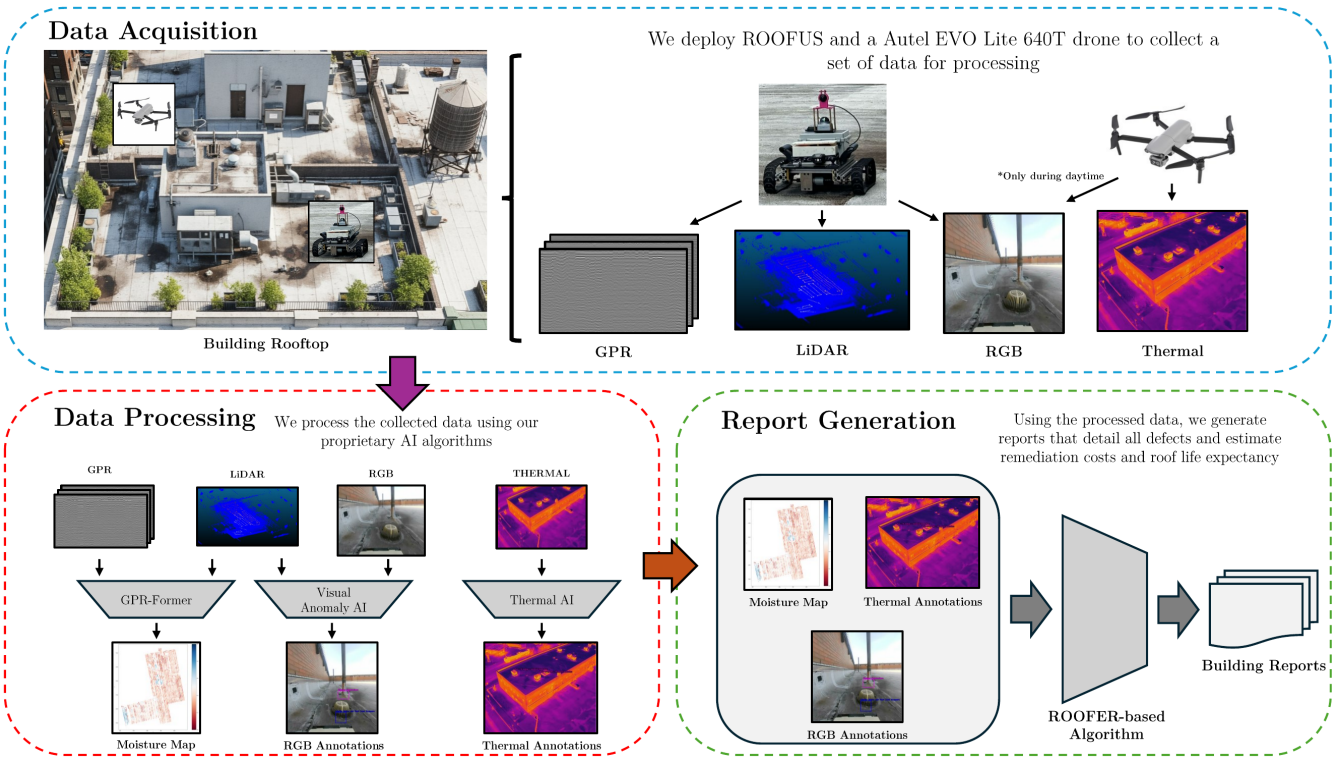



*Figure 1. Multi-modal inspection workflow: data acquisition using a robotic GPR platform and a drone, processing via GPR-Former and visual/thermal AI modules, and report generation.*

In this whitepaper we focus on the two modules that underpin subsurface and surface diagnostics: GPR-Former (moisture detection from GPR) and ROOFER360 (visual defect detection and view extraction).

# 3. Technology whitepaper: GPR-Former

GPR-Former is our proprietary AI algorithm for low-slope rooftop moisture detection. It operates on ground-penetrating radar (GPR) measurements collected across the roof surface and produces a per-location probability of moisture-related damage that can be rendered as a moisture map.

## 3.1 Problem statement

Moisture damage in roof assemblies can drive higher energy use, accelerate structural degradation, and increase lifecycle costs. From a sensing perspective, GPR signals are affected by environmental noise and material heterogeneity, and moisture propagation patterns are often non-linear rather than isolated points. This makes single-scan interpretation brittle and motivates context-aware learning approaches.

## 3.2 Inputs and outputs

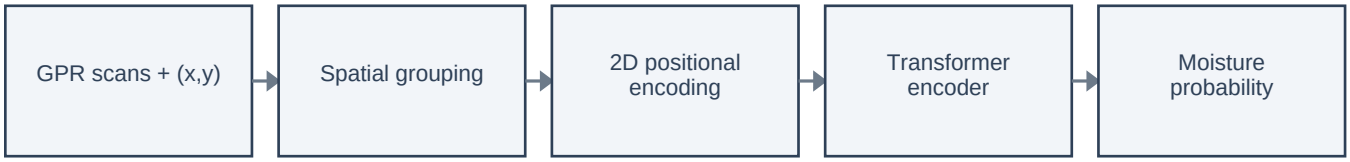
| Inputs | Typical format | Notes |
|---|---|---|
| GPR radargrams | A-scan vectors and/or B-scan stacks (e.g., .npy) | A-scans are 1D traces; B-scans are sequences of A-scans along a path. |
| Spatial metadata | 2D (x,y) coordinates per trace or derived from mapping | Used to build spatial groupings/windows and positional encodings. |
| Annotations (training) | Binary labels / masks | Binary classification is the operational baseline: moisture vs. no moisture [1]. |
| Outputs | Per-trace probabilities + optional thresholded labels | Aggregated into roof-level moisture maps for reporting. |

## 3.3 Core idea: context-aware moisture detection

The published GPR-Former research frames the task as binary classification of A-scans, but crucially does not treat A-scans as independent. Instead, it aggregates scans into spatially coherent groups and applies a transformer encoder with coordinate-based positional encodings so that attention can model spatial dependencies [1].

- Spatial context grouping: group A-scans using 2D proximity so each prediction is informed by neighboring traces, reflecting how moisture propagates across a roof surface [1].
- Coordinate-based positional encoding: encode (x,y) information into token embeddings so the transformer can learn spatial relationships within each group [1].
- Transformer encoder + classification head: a standard encoder stack followed by a sigmoid output for moisture probability [1].

**High-level conceptual flow (published)**



Published results (snapshot) show improved recall, AUROC, and calibration for GPR-Former-H relative to Roofus on a held-out test section [1].

| Model (paper) | Average Recall (AR) ↑ | F2-score ↑ | AUROC ↑ | NLL loss ↓ |
|---|---|---|---|---|
| Roofus | 0.4563 | 0.7158 | 0.8240 | 0.4940 |
| GPR-Former-H | 0.4834 | 0.7277 | 0.8436 | 0.4376 |

## 3.4 Current production implementation

The current GPR-Former project codebase operationalizes the same high-level concept (transformers + spatial context) with practical engineering improvements for repeatability, large-scale processing, and production inference.

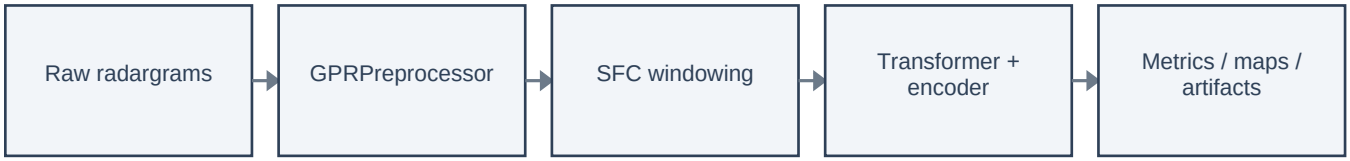
- Deterministic spatial windowing using a Morton space-filling curve (SFC), with configurable overlap and guardrails to prevent windows from spanning large spatial jumps.
- A positional encoding registry with pluggable 2D encoders (rope2d, sincos2d, learned2d, or none) configurable via YAML or CLI flags.
- Survey discovery tooling that expands parent directories into deterministic per-line jobs, enabling reproducible visualization, evaluation, and inference runs.
- A modular training/evaluation stack with shared configuration schema, CLI override support, and optional experiment tracking.

## 3.5 Architecture and data flow

At a system level, GPR-Former is implemented as a pipeline that standardizes preprocessing, builds spatially coherent training examples, runs transformer inference, and emits both metrics and visualization artifacts. The repository separates modeling, preprocessing, data loading, and CLI orchestration.

- **Models**: transformer architectures and positional encoders live under *src/gpr_former/models/*.
- **Preprocessing**: reusable transforms (normalization, augmentation, etc.) live under *src/gpr_former/preprocessing/* and can be chained via a GPRPreprocessor.
- **Data**: dataset builders and window strategies construct roof-bounded SFC windows for training and evaluation.
- **CLI**: standardized entry points (gpr-train, gpr-evaluate, gpr-visualize, gpr-visualize-preprocessing) share common runtime plumbing.

**Current pipeline (implementation view)**




## 3.6 Spatial windowing and positional encodings

Spatial context is injected through two complementary mechanisms: (1) how traces are grouped into fixed-length examples, and (2) how spatial coordinates are encoded into the transformer tokens.

**Windowing (group construction).** The current codebase uses Morton SFC windowing: 2D coordinates are ordered along a space-filling curve so that neighboring points in 2D tend to remain near each other in the 1D sequence. Fixed-length windows are sampled along the SFC with configurable overlap, and a jump threshold guards against discontinuities that would mix unrelated regions.

**Positional encodings.** Encoders are registered and selected via configuration. Supported options include 2D sinusoidal encodings, learned 2D encodings, and 2D rotary encodings (RoPE). Encoder parameters can be overridden from YAML or CLI flags, enabling controlled ablations and fast iteration.

The published paper reports that (for its dataset) sinusoidal absolute encoding and an intermediate group size ($\omega$=1024) provided the best average recall, while overly small or overly large groups reduced performance due to insufficient context or grouping artifacts [1].

## 3.7 Training, evaluation, and operating points

GPR-Former is packaged as a research-to-production toolkit with repeatable training and evaluation. Configuration is expressed as a nested schema and hydrated from YAML, with CLI flags overriding specific fields for experiments.

- Training orchestration is modular: the trainer coordinates step execution, metrics accumulation, and logger hooks, making it easier to add workflows like cross-validation folds.
- Evaluation is decomposed into batch collection, metric computation, and report writing, with support for ROC/PR/FROC artifacts, calibration plots, and bootstrapped summaries when enabled.
- Operating points (threshold selection) are configurable: fixed thresholds, maximization of a target metric (default f2), or FP-budget constrained threshold search.
- Metric reporting supports Recall@FP-per-1k targets and FROC sweeps for deployment-relevant trade-offs.

This design is intended to keep a clear separation between model improvement work (architecture/encodings/windowing) and operational work (reproducible runs, artifacts, and comparable operating points).

## 3.8 Production inference

For deployment and batch processing, the codebase exposes a production-ready inference CLI that mirrors evaluation wiring while emitting duplicate-aware inference artifacts and diagnostics. This reduces the gap between offline validation and field usage by keeping data discovery, preprocessing, and windowing consistent across phases.

## 3.9 Visualization and artifacts

GPR-Former includes reproducible visualization pipelines for B-scans and C-scans. Visualization entry points can ingest either a single scan line or a parent survey directory; survey discovery expands the parent into deterministic jobs and records all decisions in a run manifest.

- B-scan visualization renders line-level radargrams with optional overlays and manifest logging.

- C-scan visualization aggregates scans at the survey level and can auto-generate a manifest when one is not supplied.
- Preprocessing visualization tools help validate normalization and augmentation choices before training.

These artifacts are valuable for communicating model behavior to operators and for auditing failure modes (e.g., reflective objects and structural interference) that can resemble moisture in raw GPR [1].

## 3.10 Integration into reporting

Operationally, GPR-Former outputs are rendered into roof-level moisture maps. These maps can be aligned with other modalities (e.g., RGB defect annotations) and ingested by downstream reporting systems to support remediation planning.

Within the broader multi-modal workflow (Figure 1), the moisture map generated by GPR-Former becomes a primary subsurface diagnostic layer that complements ROOFER360’s surface defect layer.

# 4. Technology whitepaper: ROOFER360

ROOFER360 is an end-to-end pipeline that detects visual rooftop anomalies and defects, maps them into 3D, and selects the best inspection views for reporting. It is designed for 360° roof walkthrough videos with corresponding LiDAR reconstruction outputs, but also supports an image-only mode for photo sets or 360° videos without point-cloud context.

## 4.1 Design goals

- Detect roof distress in a structured format (JSON bounding boxes, optional segmentation masks) suitable for downstream QA and reporting.
- Fuse detections with 3D geometry so that repeated observations of the same defect can be clustered across frames and localized.
- Automatically extract best views (well-framed, low-distortion crops) that can be used directly in inspection reports.
- Provide practical operational controls: CPU-only fallbacks, concurrency tuning, caching, and an optional quality-control (QC) review loop.

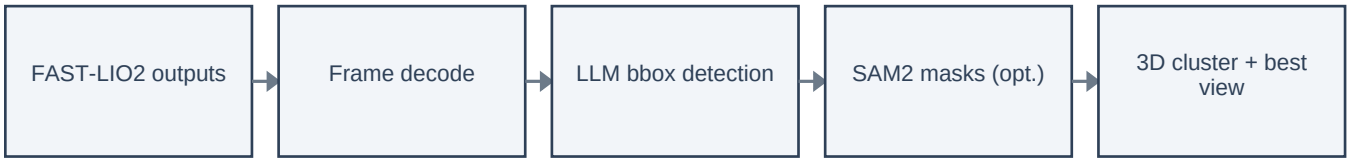
## 4.2 Inputs and outputs

| Mode | Required inputs | Key outputs |
|---|---|---|
| Video + LiDAR | MP4 inspection video; coloured point cloud (PCD); camera poses; extrinsics | Textured mesh; defect images + JSON; HTML report; clustered defect database |
| Image-only | Folder of images or 360° video | Per-image defect JSON (boxes and optionally masks) + packaged results bundle |

## 4.3 System architecture

ROOFER360 assumes that the upstream mapping stack (FAST-LIO2) has already produced a coloured point cloud and time-synchronized camera poses. ROOFER360 then performs defect detection, optional segmentation, 3D projection and clustering, and best-view extraction.

**High-level pipeline (video + LiDAR mode)**




All defect detections use OpenAI’s gpt-5 model with medium reasoning by default, with the option to escalate to high reasoning when additional deliberation is needed.

## 4.4 Processing stages

ROOFER360 is organized as a sequence of stages. In video + LiDAR mode, the stages progress from frame selection through 3D localization and reporting; in image-only mode, the pipeline stops after per-image detection and packaging.

### Stage 1: Data ingestion and keyframe selection

- Video decoding (OpenCV) and, for 360° sources, conversion from equirectangular frames to cubemap faces.
- Pose ingestion from FAST-LIO2 trajectory files; keyframe selection can be uniform based on spatial distance (e.g., every 0.5 m) rather than purely time-based sampling.
- Image-only mode supports either a folder of images or downsampling a 360° video to 1 fps with configurable start buffers and keyframe intervals.

### Stage 2: Mesh reconstruction (video + LiDAR mode)

The coloured point cloud is converted into a textured mesh using Open3D's Poisson or TSDF reconstruction, producing a mesh artifact suitable for ray casting and projection.

### Stage 3: Defect detection and optional segmentation

- Bounding-box prediction: gpt-5 is prompted with the active distress taxonomy and returns structured JSON boxes with {label, confidence, xmin, ymin, xmax, ymax}.
- Post-processing: confidence and area filtering, plus Non-Maximum Suppression (IoU >= 0.5) to prune overlaps.
- Optional mask generation: for each bounding box, SAM 2 produces a binary segmentation mask (skipped entirely when running CPU-only).
- Overlays: annotated images and overlay styling are controlled by configuration (separate overlay scopes for distress and thermal subsystems).

### Stage 4: Mask projection and cross-frame clustering (video + LiDAR mode)

To consolidate repeated observations of the same physical defect, ROOFER360 projects mask pixels into the reconstructed mesh by ray casting. Projected hits mark mesh faces, and clustering (DBSCAN) merges detections across frames into a roof-level defect database that records per-defect geometry such as centroid and area.

### Stage 5: Best-view selection and undistortion (video + LiDAR mode)

For each clustered defect, candidate frames are scored based on viewing geometry (angle and distance). The top-scoring frame is selected and undistorted around the defect region (gnomonic projection), producing inspection-ready crops with associated metadata.

### Stage 6: Packaging and reporting

- Run outputs are copied into a structured output directory that includes the textured mesh, defect assets, and a human-readable HTML report.
- The pipeline exports COCO-format instance JSON for interoperability, including stable image identifiers derived from frame number and cubemap face index.
- Utilities exist to aggregate run results into a defect summary JSON for downstream ingestion.

## 4.5 Quality control and severity quantification

ROOFER360 includes an optional post-processing review loop that uses an OpenAI-powered assistant to audit detections and improve consistency. When enabled, overlay images and metadata are submitted for review; the assistant can approve, modify, or remove detections and the pipeline writes traceable before/after artifacts.

- QC runs in multiple passes (bounded by a configured limit) and stops early when no further edits are suggested.
- Raw responses are cached to avoid duplicate billing, and subsequent passes can reference a previous response identifier so the assistant can reuse prior reasoning (configurable).
- Before edits, baseline overlays and metadata are saved, and side-by-side comparison images are generated for auditability.

After QC, each approved detection can be assigned a severity level (Low / Medium / High) along with an optional justification. Severity is computed by querying gpt-5 with the distress label, roof type, and relevant severity guidance snippets from project documentation.

## 4.6 Configuration, concurrency, and operations

Operational robustness is addressed via a centralized configuration system and a bounded concurrency model. Defaults are stored in a single config.yaml, validated via a Pydantic settings model, and overridden by environment variables (ROOF360_*) or CLI flags.

- BoundedExecutor: a thread-pool wrapper with a bounded queue (queue_size default 8) to prevent unbounded memory growth when processing long videos.
- Dedicated worker pools for detection, QC, and thermal helpers; CPU-bound stages can switch to process-based execution via a flag.
- Thread-safety mechanisms include per-file locks for cache writes, async JSONL append utilities for streaming logs, and thread-local OpenAI clients.
- Structured logging emits per-run concurrency metrics (submitted/completed/failed/avg_duration) and maintains JSONL streams for detections and errors.

## 4.7 Deployment modes and fallbacks

- CPU-only mode skips SAM 2 segmentation and returns bounding boxes only, enabling execution on lightweight hardware at the cost of pixel-accurate masks.
- Image-only mode runs without Open3D or point-cloud context and can process either a folder of images or a downsampled 360° video.
- Video + LiDAR mode produces 3D-aware clustering and best-view extraction outputs that are suited for full report generation workflows.

## 4.8 Thermal defect detection (planned / under development)

A thermal subsystem is scaffolded to ingest 640×512 thermal videos and drone flight logs, downsample to keyframes, run gpt-5 detections, and apply a thermal-specific QC pass. This module mirrors the main pipeline’s caching and auditability patterns and is expected to be expanded in future releases.

# 5. System-level integration and reporting

While GPR-Former and ROOFER360 can run independently, they are designed to be complementary layers within a single roof inspection workflow.

## 5.1 Complementary diagnostics

- **GPR-Former** provides a subsurface moisture probability layer that can identify hidden saturation patterns even when the roof surface appears intact [1].
- **ROOFER360** provides a surface defect layer with localized imagery and view-selected evidence that supports actionable repair scoping.
- When combined, the two layers help prioritize remediation (e.g., visual damage over dry substrate vs. visually subtle but moisture-positive areas) and reduce the likelihood of missed issues.

## 5.2 Data products for downstream systems

The platform produces machine-readable artifacts (JSON, COCO) and human-readable artifacts (HTML reports, annotated imagery, moisture maps) to support both automation and inspector review. A typical downstream consumer is a report generator that collates evidence, estimates remediation scope/costs, and tracks roof condition over time.

## 5.3 Practical integration patterns

- Use consistent site identifiers and timestamped run manifests so that moisture maps and visual defects can be joined at the roof and survey level.
- Align coordinate frames when available (e.g., LiDAR-derived (x,y) for GPR traces and FAST-LIO2 reconstruction for ROOFER360) to support map overlays and cross-modality localization.
- Standardize taxonomies (defect labels, severity schema) so outputs can be aggregated into portfolio-level dashboards.
- Maintain audit trails (before/after QC artifacts, cached model responses, and run logs) to support quality assurance and regulator/partner review.

# 6. Roadmap and future work

Both technologies are active areas of development. The current implementations are designed to support iterative improvement through modular components and configurable operating points.

## 6.1 GPR-Former roadmap themes

- Enhanced grouping/windowing: overlapping groups or clustering-based sampling to reduce discontinuity artifacts while preserving spatial locality [1].
- Data expansion: broader coverage of roof material types and environmental conditions to improve generalization and unlock more expressive positional encodings [1].
- Additional targets beyond binary moisture: research directions include moisture depth or severity estimation for richer decision support [1].
- Operational calibration: continued focus on calibrated probabilities and deployment-relevant recall-vs-false-positive trade-offs via FROC and FP-budget operating points.

## 6.2 ROOFER360 roadmap themes

- Thermal defect detection: expand the under-development thermal pipeline (video downsampling, flight-log alignment, QC) to provide an additional modality for moisture-related indicators and heat-loss anomalies.
- Latency and cost optimization: increase cache reuse, tune keyframe selection, and selectively escalate model reasoning effort only when needed.
- Robustness on challenging geometry: improve handling of 360° distortions and refine segmentation and projection steps for edge cases.
- Data integrity: strengthen pose-to-frame alignment checks and error handling to prevent mismatches between video streams and trajectory exports.

# 7. References


- [1] K. Lee and C. Feng, "GPR-Former: Context-Aware Moisture Detection," in *Proceedings of the 42nd International Symposium on Automation and Robotics in Construction (ISARC)* , Montreal, Canada, Jul. 2025, pp. 1-8. International Association for Automation and Robotics in Construction (IAARC), ISBN 978-0-6458322-2-8, ISSN 2413-5844, doi: 10.22260/ISARC2025/0002.