Professional Technical Report

Project Overview

This project is a real-time railway pantograph monitoring system.

It combines camera vision, depth sensing, GPS, and alert logging to measure:

- Contact wire height
- Stagger
- Setting distance (implantation)
- Gradient
- Double contact

Core runtime modules:

- main.py
- old_models_realtimeprocessor.py
- zed_implantation.py
- driver.py
- mpeg_writer.py

End-to-End Runtime Flow

1. Startup launches two parallel processes:
   - ZED sender from start.bat -> driver_launcher.py
   - GUI + analytics from start.bat -> main_launcher.py
2. ZED sender grabs color and depth frames and writes both to shared memory.
3. Main thread reads webcam frames + ZED shared frames and runs YOLO detection.
4. Webcam stream is processed for pantograph geometry measurements.
5. ZED stream is processed for setting distance measurement.
6. Alerts are visualized in GUI tables and exported to CSV/Excel reports.
7. Output videos are encoded through FFmpeg NVENC.

Math Formulas, Why They Are Used, and Where

1) GPS DDM to Decimal Degrees

Formula:

- degrees = floor(coord / 100)
- minutes = coord - (degrees * 100)
- DD = degrees + (minutes / 60)

Reason:

- Convert NMEA style GPS values into decimal coordinates for geospatial operations.

Where:

- old_models_realtimeprocessor.py (convert_coordinates)
- tools/plot_map.py (convert_ddm_to_dd)

2) Decimal Degrees to DDM

Formula:

- minutes = (abs(dd) - abs(degrees)) * 60
- DDM = degrees * 100 + minutes

Reason:

- Convert map-ready decimal values back to GPS style format when needed.

Where:

- tools/gps_format_change.py
- old_models_realtimeprocessor.py (to_ddmm_mmmm)

3) Haversine Distance

Formula:

- a = sin(dlat/2)^2 + cos(lat1)*cos(lat2)*sin(dlon/2)^2
- c = 2 * atan2(sqrt(a), sqrt(1-a))
- distance = R * c

Reason:

- Compute nearest mast from current GPS location reliably.

Where:

- old_models_realtimeprocessor.py (haversine_distance, find_nearest_mast)

4) Speed Conversion

Formula:

- speed_kmph = speed_knots * 1.852

Reason:

- Display speed in km/h.

Where:

- old_models_realtimeprocessor.py (collect_and_interpolate_gps_data)

5) Pixel to Millimeter Calibration

Formula:

- pixel_to_mm_ratio = pantograph_width_mm / strip_width_pixels

Reason:

- Convert image-space measurements to physical units.

Where:

- old_models_realtimeprocessor.py (calculate_pixel_to_mm_ratio)

6) Contact Wire Height

Formula:

- height_mm = (frame_height - strip_top_y) * pixel_to_mm_ratio + train_height
- corrected_height = height_mm + height_correction + roof_to_frame_height

Reason:

- Estimate real contact wire height from image geometry.

Where:

- old_models_realtimeprocessor.py (measure_contact_wire_height + correction usage)

7) Stagger

Formula:

- stagger = abs(strip_x - contact_x) * pixel_to_mm_ratio
- corrected_stagger = stagger + stagger_distance_correction

Reason:

- Measure lateral contact offset.

Where:

- old_models_realtimeprocessor.py (measure_stagger + process_frame)

8) Setting Distance from ZED Depth

Formula:

- final_distance_m = (depth + camera_offset + setting_distance_correction) * 0.0254

Reason:

- Convert depth value from inch-based stream to meters and apply calibration offsets.

Where:

- zed_implantation.py (process_frame)
- driver.py sets ZED coordinate units to inch

9) Setting Distance Aggregation

Formula:

- setting_distance = mean(current_mast_distances)

Reason:

- Use frame aggregation for stable final value per mast event.

Where:

- zed_implantation.py (process_frame)

10) Gradient

Formula:

- gradient_mm_per_m = (curr_height - prev_height) / distance_between_mast

Reason:

- Detect sharp rise/fall trend between masts.

Where:

- old_models_realtimeprocessor.py (calculate_gradient)

11) Relative Gradient (Windowed)

Formula:

- relative_gradient = (height_last - height_first) / sum(distance_window)

Reason:

- Smooth trend across recent mast window.

Where:

- old_models_realtimeprocessor.py (calculate_relative_gradient)

12) Double Contact

Formula:

- double_contact = abs(cp1_x - cp2_x) * pixel_to_mm_ratio

Reason:

- Detect abnormal dual contact condition.

Where:

- old_models_realtimeprocessor.py (process_frame)

13) FPS and Moving Average

Formula:

- measured_fps = frame_count / max(elapsed, 0.001)
- avg_fps = sum(fps_history)/len(fps_history)

Reason:

- Real-time performance monitoring.

Where:

- main.py (VideoThread.run)

14) Frame Rate Pacing

Formula:

- sleep_time = max(0, 1/nominal_fps - frame_processing_duration)

Reason:

- Keep runtime pacing stable.

Where:

- main.py (VideoThread.run)

15) Shared Memory Buffer Size

Formula:

- bytes = prod(shape) * dtype.itemsize

Reason:

- Allocate exact shared buffer size for inter-process frame transfer.

Where:

- driver.py

16) 3D to 2D Projection (Tool Script)

Formula:

- u = (X * fx) / Z + cx
- v = (Y * fy) / Z + cy

Reason:

- Draw projected 3D points/boxes on 2D frame.

Where:

- tools/3d_test.py

Core Algorithms and Concepts

1) Dual-Process Architecture with Shared Memory

- Producer process: ZED capture and publish.
- Consumer process: GUI, AI inference, measurements.

Files:

- driver.py

- main.py

2) File Flag Synchronization

- color.flag and depth.flag indicate frame readiness.
- all_shutdown.flag triggers graceful stop.

Files:

- driver.py
- main.py
- driver_launcher.py

3) YOLO Detection Pipeline

- Main model: best_81_l.engine
- Arm model: arm_medium_90.engine

Files:

- main.py
- old_models_realtimeprocessor.py

4) Confidence Thresholding

- strip class uses threshold ~0.4
- mast/contact_point use threshold ~0.25
- arm uses threshold ~0.50

File:

- old_models_realtimeprocessor.py

5) ROI Filtering and Robust Depth Selection

- Filter invalid depth values.
- Use mode-like representative depth after rounding.

File:

- zed_implantation.py

6) Nearest Mast Matching

- GPS + haversine vectorized distance to mast catalog.

File:

- old_models_realtimeprocessor.py

7) Event/Cooldown Based Logging

- Avoid duplicate logs each frame.

- Log on event transitions.

Files:

- old_models_realtimeprocessor.py
- zed_implantation.py

8) Alert Rules

- Stagger alert: abs(stagger) > 300
- Height alert: height > 7800 or < 4500
- Gradient alert: abs(gradient) > 3
- Double contact alert: abs(double_contact) > 500
- Setting distance alert by safe range bounds

Files:

- old_models_realtimeprocessor.py
- zed_implantation.py

9) Async Video Writer

- Queue + background writer thread + FFmpeg subprocess.

File:

- mpeg_writer.py

10) Automated Combined Report Generation

- Read per-alert CSV logs.
- Build combined Excel with branding.

File:

- old_models_realtimeprocessor.py

Technologies and Tools Used

Programming and Runtime

- Python
- Windows batch launcher (start.bat)
- Conda environment

Vision and AI

- OpenCV
- Ultralytics YOLO
- TensorRT export workflow

GPU and Depth

- ZED SDK (pyzed)
- CuPy
- NVIDIA CUDA/TensorRT stack

GUI and Threading

- PyQt6
- QThread + Qt signals

GPS and Geo

- pyserial
- pynmea2
- geopy

Data and Reporting

- pandas
- openpyxl
- CSV + Excel report generation

Video Encoding

- FFmpeg
- h264_nvenc and hevc_nvenc

Build/Protection Utilities

- Cython compile scripts (compile_all.py variants)

File-by-File Purpose Summary

- main.py: GUI, threading, stream orchestration, FPS display, alerts table updates.
- old_models_realtimeprocessor.py: main measurement math and business logic.
- zed_implantation.py: ZED depth processing and setting distance logic.
- driver.py: ZED frame producer to shared memory.
- mpeg_writer.py: async FFmpeg video writer.
- start.bat: starts sender + main app.
- driver_launcher.py: flag cleanup + sender start.
- main_launcher.py: app entry wrapper.
- compile_all.py and tool variants: Python to pyd compilation utility.
- tools/*.py: camera tests, GPS tests, map plotting, depth experiments, TensorRT export helper.

Final Notes

- The system is production-oriented for railway pantograph monitoring with calibrated physical measurements from camera and depth data.
- Unit conversions and threshold rules are central to safety alerts.
- The architecture prioritizes real-time performance, robustness, and operational reporting.