Raspi5-Hailo Bubble Detection Project Presentation

TTL AI :
Landon Campbell - Team lead - integration, performance testing
Thomas Keyes - Physical design, camera integration, CV pipeline
Tiger Zhang - CV pipeline, UNET and CNN training and deployment

Problem & Motivation

- Hydraulic MTB brakes can trap microscopic air bubbles during bleeding.
- Bubbles compress, causing inconsistent lever feel and reduced braking performance.
- Manual bubble detection is visual and subjective; our long-term goal is an automated, quantitative tool that integrates with an automation tool.
- We target on-device processing so the system can run in a small, portable, bike-shop friendly tool with no cloud dependency.
- Additional applications: medical devices where current ultrasonic sensing is inconsistent.

Fig 1. Mountain bike brake bleed with bubble release

Edge Bubble Detection: Overview & Current Hardware

- Goal: Edge-deployed vision system on a Raspberry Pi 5 + Hailo-8L that detects, sizes, and counts bubbles in MTB brake lines in real time, fully on-device for low latency and privacy while maintain above 90% accuracy and greater than 60fps.

- Current model implementation:
  - Classical OpenCV pipeline developed and debugged on a laptop using synthetic and downloaded bubble videos.
  - Pipeline outputs: binary masks, bounding boxes, approximate bubble diameters, and simple void-fraction estimates per frame.
  - Code is structured as reusable modules (BubbleCVModel, BubbleCNNModel) that can be dropped onto the Pi once hardware access is available.

- Planned edge-hardware step:
  - Port the existing pipeline to Raspberry Pi 5, then swap the CPU mask step with a small CNN running on Hailo-8L for lower latency and power.

Fig 2. Single frame of CV pipeline detection and mask

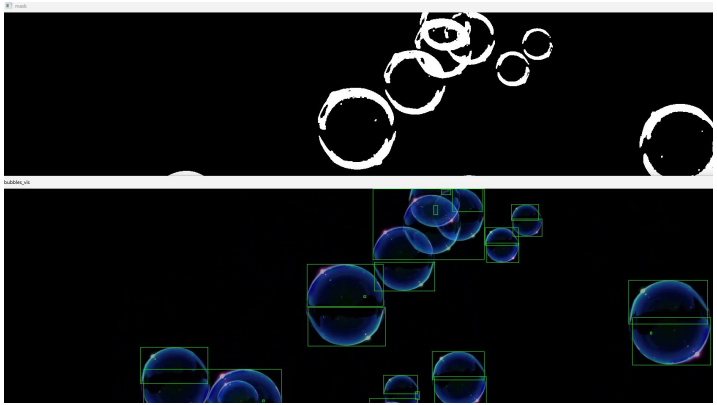

Fig 3. Snippet of video processing of detection and mask

Data Collection and Classical Computer Vision Results

- Dataset so far:
  - 1 synthetic “fake bubble” video and 1 additional bubble video downloaded from the web.
  - Frames from the synthetic video were processed with the classical CV pipeline to generate pseudo-label masks.
  - From these, we created a small dataset of 256×256 image–mask pairs for training a segmentation model.
- Classical CV pipeline implemented:
  - Grayscale → Otsu threshold → morphology → connected components.
  - Outputs per-bubble bounding boxes (green), approximate diameters in pixels, and a binary mask.
  - Works well on clean synthetic data; on the second video it still detects bubbles but shows some over/under-segmentation.
- CNN segmentation model (BubbleCNNModel):
  - Small UNet-like network trained in PyTorch on the synthetic image–mask pairs.
  - Training on 256×256 crops; validation Dice ≈ 0.93 when compared to the CV-generated masks (as expected, since it’s learning to mimic that baseline).
  - On synthetic data, CNN masks closely match the label masks; on the second video, CNN and CV masks generally agree but diverge in some noisy regions.
- Takeaway:
  - We now have both a classical CV baseline and a trained CNN for bubble segmentation running on a laptop, with APIs designed so they can be swapped on an edge device later..

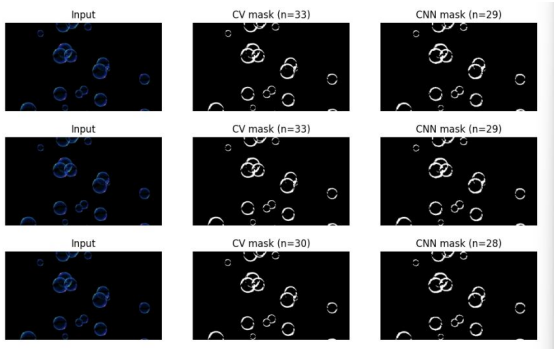

Fig 4. CV vs. CNN mask on trained video

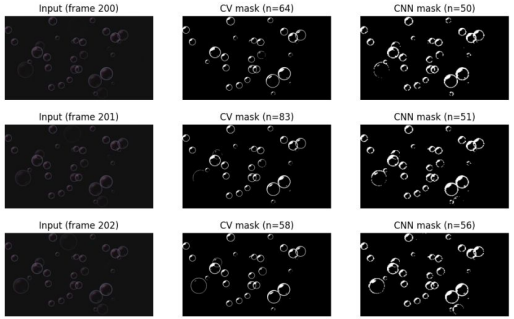

Fig 5. CV vs. CNN mask on untrained video

Path to Edge-AI Hardware (Planned Hailo Deployment)

- Current AI model status:
  - Trained small UNet-style CNN (SmallUNet) for bubble vs background segmentation on CPU.
  - Wrapped in BubbleCNNModel with the same predict(frame) interface as the classical CV model.
  - Checkpoint saved and re-used in notebooks for evaluation and qualitative comparison.
  - Working on integration with raspi 5
- Planned edge deployment on Raspberry Pi 5 + Hailo-8L:
  - Export the trained PyTorch model to ONNX and quantize to INT8.
  - Use Hailo's toolchain to compile the model for the Hailo-8L NPU.
  - Run segmentation on the NPU while the Pi CPU handles tracking, metrics, and UI overlay.
- Expected hardware-level benefits (to be validated):
  - Lower per-frame latency for segmentation compared to CPU-only CV, especially at higher resolutions or ROIs.
  - Lower energy per frame by offloading conv layers to the NPU.
- Remaining work before final presentation:
  - Collect more training data and use manual dataset labeling
  - Hailo integration and real-time camera capture of bubble release