

Article

Vision-Based Inspection of Airfield Ground Lights: Detection, Colour Classification, and Edge-Computing Performance

Jakub Suder *  and Jakub Tadeusz Jóźwiak

Division of Signal Processing and Electronic Systems, Institute of Automatic Control and Robotics, Faculty of Automatic Control, Robotics and Electrical Engineering, Poznan University of Technology, ul. Jana Pawła II 24, 60-965 Poznan, Poland; jakub.jozwiak@student.put.poznan.pl

* Correspondence: jakub.suder@put.poznan.pl

Abstract

Airfield ground lighting is safety-critical electrical infrastructure and requires reliable inspection. We benchmarked lamp detection, colour assignment and embedded execution using 59 GoPro HERO 8 recordings acquired through scientific cooperation with Poznań-Ławica Airport. The dataset contains 3612 frames and 18,141 instances in four colour classes; training-only augmentation produced 8668 images. Four YOLO variants and three classical pipelines were compared on a 362-image test subset using region-of-interest filtering and greedy one-to-one matching at an intersection-over-union threshold of 0.45. At 640 by 480 pixels, YOLO26 achieved the highest observed F1 score (0.8080) at 64.0 frames per second, followed by YOLOv12 (0.7178) at 53.8 FPS. Brightness thresholding reached 622.4 FPS with F1 score = 0.3057. Raspberry Pi 5 CPU execution reached 1.1–1.3 FPS; Hailo-8 reached 48.6 and 36.4 FPS for YOLOv8 and YOLO11, respectively, with a reduced F1 score after conversion. FPS includes working-image preparation, detector execution and post-processing, but excludes video decoding and rendering. These fixed-split measurements quantify the trade-off between detection quality, colour coverage and headless throughput for human-supervised inspection and edge-platform selection.

Keywords: airfield ground lighting; visual inspection; small-object detection; colour classification; YOLO; empirical benchmarking; embedded artificial intelligence; edge computing; Hailo-8

1. Introduction

Airfield ground lighting (AGL) provides visual guidance for runway, taxiway and approach operations, particularly at night and in reduced visibility. Inspection requires access to operational areas and must distinguish lamp degradation from appearance changes caused by exposure, pavement reflections, weather and viewing geometry. Accident reports involving runway-edge lights and night-time alignment illustrate the need for unambiguous visual guidance [1,2]. Light geometry, colour, brightness, glare and fog also affect how pilots interpret this guidance [3,4]. Computer vision offers tools for monitoring airport infrastructure and supporting inspection [5].

Maintaining AGL requires coordinated inspection of distributed electrical assets. Airport-energy research includes airfield lighting among the loads considered in infrastructure planning [6]. Image-based monitoring can help locate lamps and document their appearance. This study examines the detection quality and processing throughput that established detectors and edge platforms can provide for that task.



Academic Editor: Atsushi Mase

Received: 17 September 2026

Revised: 5 October 2026

Accepted: 12 October 2026

Published: 15 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

We consider vehicle-mounted imaging of inset or near-pavement airfield lamps. The lamps may occupy only a few pixels, their colours must be distinguished, and reflective markings can resemble light sources. Resource-constrained hardware adds a processing limit. Evaluation must therefore cover the detection and post-processing pipeline rather than rely solely on neural network validation metrics.

Acquiring airfield-light imagery requires controlled access to runway and taxiway operating areas. The recordings used here were obtained through scientific cooperation between Poznań-Ławica Airport and Poznan University of Technology. They include operationally relevant backgrounds and visibility conditions; this cooperation does not constitute operational certification of the system.

Previous airfield-light studies addressed localization, chromaticity, luminous-intensity measurement and unmanned inspection [7–10], using setups that differ from vehicle-mounted imaging. Small-object studies identify limited pixel support and scale variation as detection challenges [11,12]. Here, we compare established detectors using restricted-area imagery, explicit ROI and matching rules, branch-specific colour assignment, and headless timing on desktop and embedded platforms. The contribution of this study lies in the dataset, common evaluation framework and deployment evidence, not in a new detection algorithm.

We address three engineering questions: (1) how learned and classical pipelines differ in detection and colour-assignment quality at the reported operating point; (2) how performance varies across the available scene groups and working resolutions; and (3) what trade-offs between quality and throughput arise on Raspberry Pi 5 CPU and Hailo-8.

The contributions of this study are:

- A field dataset derived from 59 airport drive recordings, with 3612 original frames, four lamp-colour classes, explicit background examples, and 18,141 annotated instances;
- A benchmark integrating established YOLO and classical pipelines with a documented ROI policy, common box representation and geometric matching, and branch-appropriate colour assignment;
- An application-level evaluation on held-out frames using detection and colour metrics, scene-subset counts, working-resolution comparisons, and headless processing time;
- An edge-platform quality-throughput comparison covering Raspberry Pi 5 CPU execution and Hailo-8 HEF execution, with conversion treated as a distinct deployment variant.

2. Related Work

Vision-based light-source recognition appears in both road and airport applications. For road traffic lights, de Charette and Nashashibi combined spot-light detection with adaptive templates, demonstrating how appearance and geometric constraints can support recognition of small luminous objects [13]. The airport environment adds constrained access, long viewing distances, runway or taxiway geometry, and safety-related maintenance requirements. Pamuła et al. used image processing to support UAV-based airport-light inspection without GNSS positioning [7]. Their objective was inspection-path geometry rather than direct comparison of lamp detectors on a vehicle-mounted dataset. Sitompul et al. investigated a drone-mounted photometric instrument for runway-edge-light measurement, reducing dependence on fixed airport infrastructure but addressing direct photometry rather than image-based lamp detection [14].

Suder et al. evaluated lamp chromaticity using an integrated colour sensor, while Podbucki et al. measured luminous-intensity distributions of in-pavement lamps under laboratory conditions [9,10]. Podbucki and Marciniak described camera-based light-source localization for precise luminous-intensity measurement [8]. These studies distinguish

geometric detection, colour assignment and photometric measurement. We use camera intensity only as a relative screening signal, not as traceable photometry.

Table 1 summarizes how the closest groups of work relate to the scope of this study.

Table 1. Position of the present study relative to representative light-recognition and airfield-light research.

Reference	Environment and Sensor	Primary Task	Relation to This Study
de Charette and Nashashibi [13]	Road scenes; monocular camera	Real-time traffic-light recognition	Demonstrates small luminous-object recognition, but not airfield geometry, field-light classes, or edge-platform comparison.
Pamuła et al. [7]	Airport; UAV imaging	Inspection-point localization without GNSS	Addresses autonomous inspection geometry rather than detector accuracy and colour classification.
Sitompul et al. [14]	Airport runway; drone-mounted photometric instrument	Runway-edge-light photometry	Demonstrates mobile direct measurement, but not camera-based multi-class detection or embedded inference.
Suder et al. [9]	Airport lamps; colour sensor	Chromaticity verification	Provides colour-measurement context but does not evaluate image-based localization.
Podbucki et al. [10]	Laboratory; photometric sensors	Luminous-intensity distribution	Provides calibrated photometric assessment under controlled conditions.
Podbucki and Marciniak [8]	Airport-lamp measurement setup; camera	Light-source localization for photometry	Motivates brightness-based candidate extraction, but not a multi-condition detector benchmark.
This study	Vehicle recordings; action camera	Detection, colour assignment, relative screening, and edge execution	Compares learned and classical pipelines under one ROI-filtered test and timing protocol.

Classical computer-vision methods remain attractive for embedded inspection because they require no training and can be computationally inexpensive. Edge-based pipelines use gradients and contour geometry, with Canny detection and the Hough transform providing established formulations for edge localization and line suppression [15,16]. Morphological opening and White Top-Hat filtering provide a complementary means of emphasizing small bright structures against slowly varying backgrounds [17]. Such methods are transparent and tunable, but fixed thresholds can be sensitive to illumination and reflections.

Deep learning detectors provide learned localization and classification within a single inference pipeline. The original YOLO formulation established the single-stage approach [18]; subsequent versions have progressively changed feature extraction, heads, losses, and deployment paths [19]. The present comparison includes YOLOv8, YOLO11,

attention-centric YOLOv12, and end-to-end YOLO26 [20–22]. Small variants were selected to reflect the intended edge-computing context.

Airfield lamps also belong to the small-object-detection regime. Surveys identify limited pixel support, noisy representations, and scale variation as persistent causes of missed detections [11]. Feature pyramids retain multi-scale information, while augmentation can improve exposure to sparse small objects and diverse backgrounds [12,23]. Embedded operation adds a separate constraint: model export, calibration, quantization, and accelerator-specific post-processing can change predictions even when the source weights are unchanged. Related airport-area studies evaluated YOLOv5 for FOD detection at aerodromes and subsequently compared GoogLeNet and YOLOv5 using two FOD datasets [24,25].

Resolution and hardware selection affect the computational demands of embedded video processing [26]. Airport-energy research provides broader infrastructure context [6], but visual detection does not measure electrical faults or luminaire efficiency. FOD and surface-marking studies likewise address different perception tasks; this evaluation concerns AGL detection and colour assignment.

Figure 1 illustrates the field context of the present task. The daytime frame combines low-contrast lamps with pavement texture and distant infrastructure, whereas the night-time frame contains small saturated light sources, glare, and large brightness differences. These observations motivate the joint evaluation of learned detectors, classical bright-object methods, and colour post-processing.



Figure 1. Representative application context from the source recordings: a daytime runway or taxiway scene with low-contrast lamps and structured pavement (left), and a night-time scene with small luminous objects and glare (right).

3. Problem Formulation and Evaluation Scope

The system takes a camera frame and an analysis ROI covering the pavement where inset or near-pavement lamps are expected. It returns bounding boxes with lamp-colour labels and, optionally, a relative image brightness ratio and structured inspection records. The evaluation covers recorded images and computational throughput, not certified photometric compliance, end-to-end vehicle navigation or autonomous maintenance decisions.

We evaluate detection and colour classification at a fixed operating point on the held-out test subset, and then compare the available time-of-day, reflection and degraded-visibility groups. The resolution comparison measures the change in detection quality between 640×480 and 320×240 pixels. Measurements on the laptop, Raspberry Pi 5 CPU and Raspberry Pi 5 with Hailo-8 quantify throughput and the endpoint changes associated with HEF conversion.

4. Materials and Methods

4.1. Video Material and Dataset Preparation

The source material comprised 59 vehicle drive recordings acquired in restricted operational areas through scientific cooperation between Poznań-Ławica Airport and Poznań University of Technology. A GoPro HERO 8 camera recorded 4K video (3840×2160 pixels) at 60 frames per second [27]. The recordings totalled approximately five hours and include day and night scenes, rain, residual snow, reflections and reduced visibility (Figure 1).

Frames were selected manually from useful video segments and extracted in the original resolution with VLC media player [28]. The published dataset contains 3612 original frames, including more than 400 background images without target airfield lamps [29]. These backgrounds include pavement, grass, buildings, elevated lamps, and empty runway or taxiway regions.

Images were annotated and exported through Roboflow [29]. Four target classes were selected from the visible lamp colours and the ICAO aerodrome-lighting context [30]: white, red, green and yellow. Preparation comprised video selection, frame extraction, annotation, splitting and training-only augmentation. Bounding boxes marked individually distinguishable lamps. Unresolved chains of distant lights were excluded because a merged light region does not represent a single lamp (Figure 2).



Figure 2. Example of the annotation policy: visually separated airfield lamps are marked as individual instances, while an unresolved chain of distant lamps is not annotated as multiple uncertain objects.

The annotated dataset contained 18,141 object instances. Table 2 gives the class distribution. White lamps dominated the recordings. Yellow lamps account for 337 instances (1.86%). Aerodrome lighting specifications assign yellow to particular guidance functions, including alternating green/yellow taxiway-centrelines near runways and yellow runway-edge caution sections, rather than uniformly across the airfield [31]. This localized use provides a plausible context for lower yellow representation in inspection imagery; its observed frequency also depends on the recorded routes, viewing directions and operating conditions.

The original-frame split contains 2528 training, 722 validation and 362 test images, approximately 70/20/10. Roboflow applied auto-orientation and stretch resizing to 1024×1024 pixels. Offline augmentation was confined to training: brightness variation from -15% to $+15\%$ and salt-and-pepper noise affecting up to 0.5% of image pixels. The export contained 7584 training images and 8668 images overall [29,32]. Validation and test images were unchanged. This frame-level split does not establish recording-grouped inde-

pendence or control near-duplicates. The held-out results therefore describe the sampled frames, not cross-recording generalization.

Table 2. Annotated airfield-light instances in the dataset.

Class	Number of Instances
White	14,751
Red	1864
Green	1189
Yellow	337
Total	18,141

4.2. Detection Methods

Seven method families were evaluated: four YOLO models and three classical image-processing pipelines. The brightness-thresholding method is denoted PP, following the project implementation. The Canny–Hough and White Top-Hat pipelines were each tested with day and night parameter presets, giving nine reported configurations. Table 3 summarizes their role in the study.

Table 3. Detection methods evaluated in the article.

Method	Main Mechanism	Role in the Comparison
YOLOv8, YOLO11, YOLOv12, YOLO26	Learned single-stage object detection; small variants trained on four lamp classes	Learned baselines for simultaneous localization and colour-class assignment.
Canny–Hough	Median filtering, Canny edges, Hough line suppression, morphological closing, and contour filtering	Classical geometry-based detector for lamp-like compact structures.
PP	Brightness thresholding, dilation, area filtering, and bright-pixel-ratio verification	Classical brightness-based method inspired by camera-based light-source localization.
White Top-Hat	Morphological opening, background subtraction, thresholding, area and aspect-ratio filtering	Classical local-contrast detector for small bright objects on nonuniform backgrounds.

The YOLO models were trained through the Ultralytics interface using transfer learning from pretrained small variants. PyTorch, CUDA, and cuDNN were used on the available GPU platforms [33–36]. Training used `imgsz = 1024`, consistent with the square Roboflow export, to preserve information from distant lamps. YOLOv8, YOLO11, and YOLO26 were trained primarily on the MSI Katana GF76 laptop. YOLOv12 was trained in Google Colab with an NVIDIA Tesla T4 because its measured epoch time on the laptop exceeded three and a half hours.

Table 4 separates the shared training configuration from model-specific parameters explicitly specified in the YOLO26s script. Script parameters describe the requested configuration, whereas settings resolved automatically by the training framework are not inferred retrospectively. The comparison therefore attributes the observed results to the evaluated checkpoints and training workflows rather than to architecture alone. The hardware specifications of the benchmark platforms are summarized in Table 5.

The three classical methods shared a preprocessing stage: image resizing, conversion to grayscale, and ROI masking. The first method used a median filter followed by Canny

edge detection, Hough-based line suppression, morphological closing, contour extraction, and area filtering. The non-maximum suppression step in Canny may be summarized by Equation (1) as follows:

$$M(x, y) = \begin{cases} G(x, y), & \text{if } G(x, y) \geq G(x_1, y_1) \text{ and } G(x, y) \geq G(x_2, y_2), \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where $G(x, y)$ is the gradient magnitude at pixel (x, y) and (x_1, y_1) and (x_2, y_2) are neighbouring pixels along the gradient direction. The Hough line representation used in the pipeline is given in Equation (2):

$$\rho = x \cos \theta + y \sin \theta, \quad (2)$$

where ρ is the distance from the image origin to a line and θ is the angle of its normal vector.

The PP method treated lamps as bright local objects. Candidate regions were obtained from thresholding and dilation, filtered by size, and then verified by measuring the proportion of sufficiently bright pixels in the original grayscale region. The White Top-Hat method used morphological opening to estimate the background and subtract it from the original image according to Equation (3):

$$T_w(I) = I - (I \circ K), \quad (3)$$

where I is the input grayscale image, K is the structuring element, $\circ$ denotes morphological opening, and T_w is the White Top-Hat response.

Table 4. Training settings shared by the four small YOLO variants and model-specific parameters specified in the YOLO26s training script.

Scope	Parameter	Specification
All four variants	Model initialization	Pretrained small variants
All four variants	Training image size	1024 × 1024 pixels
YOLO26s training script	Maximum epochs; batch size	150; 4
YOLO26s training script	Early-stopping patience; optimizer	20 epochs; auto
YOLO26s training script	Checkpoint interval	10 epochs
YOLO26s training script	Mosaic; close mosaic	1.0; final 15 epochs
YOLO26s training script	Copy-paste; mixup	0.3; 0.1
YOLO26s training script	Box-loss weight	10.0

Table 5. Test environments used for benchmark experiments.

Environment	Main Parameters	Role in the Study
MSI Katana GF76 laptop	Intel Core i5-11400H, 16 GB RAM, NVIDIA GeForce RTX 3050 Laptop GPU with 4 GB VRAM	Reference environment for full benchmark execution and comparison of YOLO and classical methods.
Raspberry Pi 5 CPU	Raspberry Pi 5, 8 GB RAM, 64-bit quad-core Arm Cortex-A76 at 2.4 GHz, Raspberry Pi OS/Linux	Embedded CPU-only platform for assessing practical computational cost without an AI accelerator.
Raspberry Pi 5 + Hailo-8	Raspberry Pi 5, 8 GB RAM, Raspberry Pi AI HAT+ with Hailo-8, 26 TOPS declared INT8 performance, PCIe interface, HailoRT 4.23.0	Embedded accelerated platform for HEF-format YOLO inference; classical methods still executed on CPU.

The benchmark used fixed classical-method parameters from the released scripts [37]. Canny–Hough used a 5×5 median filter, Canny thresholds of 127/255 for the day preset and 75/150 for the night preset, a probabilistic Hough threshold of 40, and a maximum line gap of 10 pixels. The corresponding minimum Hough line lengths were 0 and 100 pixels, line-erasure thicknesses were 20 and 10 pixels, and morphological-closing kernels were 1 and 5 pixels for the day and night presets, respectively. Accepted contour areas ranged from 5 to 1000 px². The PP pipeline used an intensity threshold of 250, a 3×3 dilation kernel with two iterations, contour areas from 10 to 1000 px², and a minimum ratio of 0.20 for pixels brighter than 200. White Top-Hat used elliptical kernels of 9 pixels for the day preset and 50 pixels for the night preset, a response threshold of 125, contour areas from 5 to 1500 px², and a maximum bounding-box aspect ratio of 3.0.

4.3. Colour Classification and Relative Image Brightness Ratio

YOLO returns colour labels directly because its four trained classes correspond to lamp colours. The evaluation maps each predicted class ID through the model's class-name metadata to a recognized colour. Missing or unrecognized names become the internal category *Inny*, reported as *unclassified*. An unclassified YOLO outcome can therefore arise during label normalization even when the network returns a class ID; it does not invoke the classical HSV rule [37]. Classical detectors return candidate positions, which are passed to a common colour-classification module. This module converts a local region around each detection centre from BGR to HSV and excludes the saturated lamp core with an annular mask. In OpenCV's 8-bit HSV representation, hue H lies in $[0, 179]$, while saturation S and value V lie in $[0, 255]$ [38]. Low-saturation pixels receive white votes:

$$\text{class} = \text{white} \quad \text{if} \quad S < S_{\text{thresh}}, \quad (4)$$

where $S_{\text{thresh}} = 50$. For $S \geq 50$, the implemented intervals are red $H \in [0, 8] \cup [160, 179]$, yellow $H \in (8, 30]$, and green $H \in (30, 85]$. Thus, both $H = 0$ and $H = 179$ receive red votes. Votes use an annulus with inner and outer radii of 3 and 12 pixels, clipped at image boundaries. The code does not apply a separate V threshold in this colour-voting step. An empty region or annulus, or a winning vote below 20% of all annular pixels, yields *unclassified* (the internal *Inny* category). The 20% denominator includes pixels outside the accepted chromatic hue intervals. Ties follow the code's class insertion order: white, red, yellow, and then green. These are implementation rules, not empirically optimized colour-confidence thresholds [37,39].

The optional module calculates frame-relative image brightness. It sums grayscale intensities of pixels above 150 within each detection region, using 15-pixel padding for classical detections and no padding for YOLO in this benchmark. With at least three detections, a quadratic is fitted to these sums as a function of vertical image coordinate:

$$\hat{B}(y) = ay^2 + by + c, \quad (5)$$

where $\hat{B}(y)$ is a fitted reference in camera-intensity units, not an expected luminous intensity, and a , b , and c are fitted coefficients. The implementation variable is named *Health Index* (HI); the manuscript reports the quantity as the relative image brightness ratio R_B :

$$R_B = \frac{B}{\hat{B}(y)}, \quad (6)$$

where B is the summed image intensity. The ratio is interpretable only when the fitted reference is positive and well-conditioned. It is an exploratory camera-intensity measure, not calibrated photometry or evidence of lamp condition; the implementation thresholds

are therefore excluded from maintenance and compliance interpretation. Exposure, gain, distance, viewing angle, weather, saturation, camera processing, detection-box size and the other lamps in the frame can alter the ratio. With exactly three distinct vertical coordinates, a quadratic interpolates the three brightness values and can yield $R_B = 1$ for every fitted sample regardless of their relative intensities. The code substitutes a reference of 1 for nonpositive fitted values, which can produce large ratios. The evaluation therefore reports only computational overhead.

4.4. Evaluation Metrics

Each predicted bounding box was compared with ground-truth boxes inside the ROI. Intersection over union (IoU) was used for geometric matching as defined in Equation (7):

$$\text{IoU} = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}, \quad (7)$$

where B_p is a predicted box and B_g is a ground-truth box. All prediction–ground-truth pairs with IoU of at least 0.45 were sorted by decreasing overlap and assigned greedily under a one-prediction-to-one-ground-truth constraint. Matched predictions were counted as true positives (TPs), remaining predictions as false positives (FPs), and unmatched ground-truth boxes as false negatives (FNs). Precision, recall, and F1 score were then computed using Equation (8):

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (8)$$

Colour accuracy is C/TP , where C counts geometrically matched detections assigned the correct ground-truth colour. Every reference match has an evaluable ground-truth class, and unclassified outcomes remain in the denominator. The joint correct-colour fraction, $C/(TP + FN)$, also accounts for missed lamps. Because this denominator excludes FP, joint coverage complements rather than replaces precision or a full multiclass quality measure. Matching is class-agnostic. Incorrect colour means that a recognized class differs from the ground truth; unclassified means that no four-class label is recognized after normalization. These disjoint outcomes are reported separately from FN. Ratios with zero denominators are undefined, although the benchmark writer stores zero in that case [37,40].

5. System Design and Implementation

5.1. Unified Processing Architecture

The YOLO and classical branches share input handling, ROI filtering, post-processing, visualization and logging. Frames from images, video files or a camera stream are resized, processed by the selected detector and passed to post-processing and structured export. Figure 3 shows where relative brightness screening and logging enter the pipeline.

The ROI module used a trapezoidal mask controlled by four parameters: top cut-off height, top width, bottom width, and horizontal centre shift. The parameters were expressed relative to the working resolution. Classical methods used the mask on intermediate images. YOLO inference used the full frame, and boxes were filtered afterwards by checking whether their centres lay inside the ROI. In this way, the neural detector retained image context, while all methods were evaluated in the same analysis region.

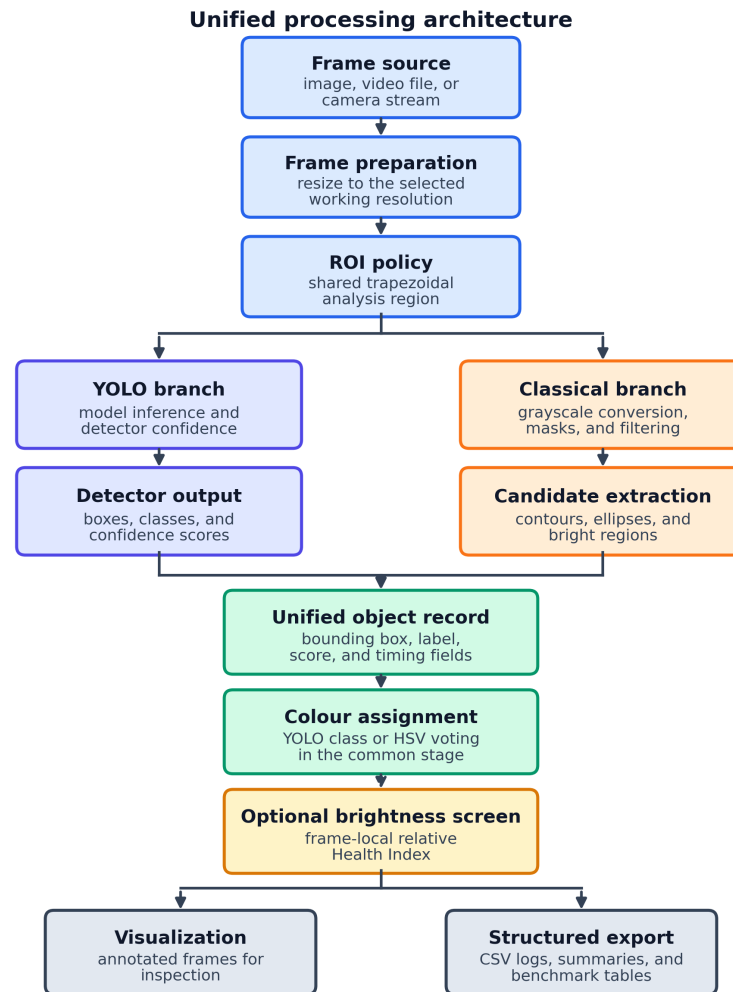


Figure 3. Processing architecture for airfield-light detection, colour classification, optional relative image brightness calculation, visualization and result logging. Colours distinguish input preparation, YOLO inference, classical image processing, shared post-processing, optional screening and exported outputs.

5.2. Classical and YOLO Branch Integration

The classical branch converted contours, ellipses, bright regions or centres into a common bounding-box representation before metric computation. Candidate boxes could be expanded to include the lamp halo used by the colour and brightness modules.

The YOLO branch loaded the selected model, ran inference at confidence 0.40, filtered detections by ROI and exported box coordinates, class labels and confidence values. The benchmark script passed no custom framework NMS threshold. Interactive operation allowed for manual selection of model weights and data sources. Benchmarking omitted rendering and timed detector execution and post-processing separately.

5.3. Output Visualization and Data Export

An interactive inspection mode was implemented with Tkinter and OpenCV [38,41]. Benchmark runs did not use graphical rendering. During development, the visualization layer was used to check detections, ROI filtering, colour assignment, and exported records. Figure 4 shows representative detection outputs.



Figure 4. Representative output visualizations from the inspection prototype: a detected runway or taxiway light in a daytime frame (**left**) and colour-coded light detections along an airfield marking under reflective conditions (**right**).

Interactive CSV records contained frame number, object identifier, confidence, bounding-box coordinates, class or colour, and processing time. Benchmark CSV records aggregated entire configurations and included TP, FP, FN, precision, recall, F1 score, average IoU, timing breakdowns, FPS, 1% Low FPS, and hardware-resource measurements.

5.4. Embedded Execution and Hailo Conversion

The embedded target was Raspberry Pi 5, optionally equipped with a Raspberry Pi AI HAT+ using a Hailo-8 accelerator [42,43]. Figure 5 shows the hardware, and Table 5 lists the test environments. Classical methods ran on the CPU in all environments. YOLO models ran from PyTorch/Ultralytics weights on the laptop and Raspberry Pi 5 CPU. Hailo execution required export and compilation to Hailo Executable Format (HEF).

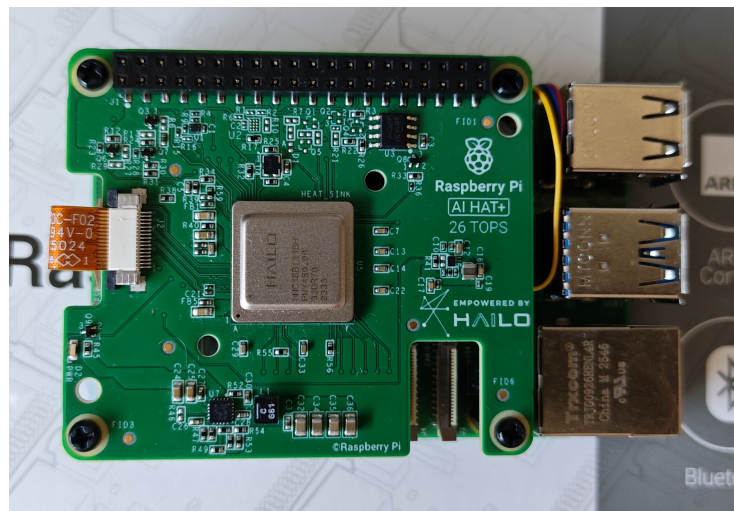


Figure 5. Raspberry Pi 5 with the Hailo-8 acceleration module used in the embedded execution experiments.

HEF evaluation covers YOLOv8 and YOLO11 at the benchmark input sizes, using validation images for INT8 calibration. These models were selected because the Hailo Model Zoo workflow used here did not provide dedicated YOLOv12 or YOLO26 architecture configurations. This limits the evaluated workflow, not the general compatibility of either model family with Hailo-8. Extending it requires a configuration specifying the export graph, output layers and accelerator post-processing, followed by compilation and quality evaluation [44–47].

The public runtime adapter converts BGR to RGB, applies aspect-preserving letterboxing with padding value 114 and supplies float32 input without explicit division by 255. It decodes class-wise NMS output, then removes coordinate padding, rounds and clips coordinates, and filters boxes by ROI centre. The decoder applies no additional Python NMS.

Inspection of this adapter does not verify the normalization or NMS settings embedded in a HEF file. Ultralytics and Hailo also round coordinates differently. These differences warrant staged comparison but do not establish the cause of the measured loss [37].

6. Experimental Setup

6.1. Benchmark Variables

The benchmark iterated over the variables in Table 6. Separate scripts were used for the laptop, Raspberry Pi 5 CPU, and Raspberry Pi 5 with Hailo-8, with the same evaluation logic in each case [37]. Figure 6 shows the per-configuration procedure: fixed test image list, detector and post-processing execution, and aggregation of detection quality, colour classification, timing, and resource-use indicators.

Table 6. Variables included in the automated benchmark procedure.

Variable	Variants	Purpose
Detection algorithm	YOLO, Canny–Hough, PP, White Top-Hat	Selection of the main detection branch.
YOLO model	YOLOv8, YOLO11, YOLOv12, YOLO26	Model variant for the YOLO branch; Hailo tests used only converted YOLOv8 and YOLO11 variants.
Working resolution	640 × 480, 320 × 240 pixels	Assessment of the influence of downscaling on small-object detection and speed.
Classical preset	Day, night	Parameter presets for classical methods in different illumination regimes.
Relative brightness module	Off, on	Measurement of post-processing overhead.
Scene condition	Tags from the configuration file	Grouping by time of day and visibility/reflection condition.

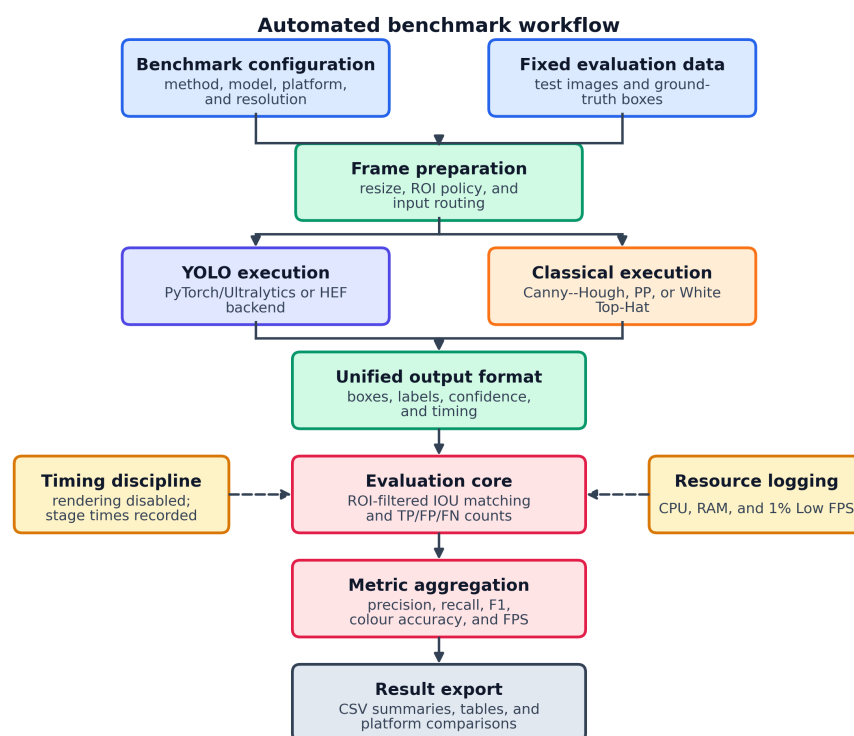


Figure 6. Benchmark workflow used for detector and hardware configurations. The same initial resize, ROI policy, output normalization, IoU matching, resource logging, and metric aggregation logic was applied across configurations; backend-specific inference preparation was retained where required.

For each image, the benchmark loaded the exported frame, resized it directly with OpenCV to 640×480 or 320×240 pixels, applied the ROI logic, ran the selected detector, converted outputs to the common bounding-box format, matched detections to ground truth using IoU, and recorded timing data. This direct resize did not preserve the square export aspect ratio. Ground-truth boxes whose centres lay outside the ROI were excluded from FN counting. Before each timed YOLO configuration, three predictions on a zero-valued dummy image were executed as warm-up. The YOLO confidence threshold was 0.40 and the IoU threshold for TP assignment was 0.45.

6.2. Test Set and Scene Categories

The held-out test subset contained 362 images. The ROI-filtered ground truth comprised 909 lamps at 640×480 pixels and 908 at 320×240 . The one-instance difference occurs in the day/reflections group, which contains 328 and 327 eligible lamps, respectively, after integer coordinate scaling and ROI-mask testing. Table 7 lists image and eligible-lamp counts, shared across configurations at each resolution. Scene categories describe recording conditions, not a controlled single-factor experiment.

Table 7. Test-set scene distribution and ROI-filtered ground-truth lamp counts at 640×480 pixels.

Scene Condition	Images	GT Lamps in ROI	Lamps/Image
Day, good weather	194	430	2.22
Day, reflections	111	328	2.95
Twilight, good weather	15	59	3.93
Twilight, rain/fog	24	64	2.67
Night, good weather	18	28	1.56
Total	362	909	2.51

6.3. Evaluation Protocol and Software Environment

Unless stated otherwise, the results use 640×480 pixels with the exploratory brightness module disabled. FPS is $1000/\bar{t}$, where $t = t_{\text{prep}} + t_{\text{det}} + t_{\text{post}}$ is the measured processing time in milliseconds. Preparation includes working-resolution resizing and ROI-mask construction. Detector execution and post-processing also cover branch-internal input preparation, colour assignment and optional brightness calculation. Image reading, annotation loading, metric calculation, video decoding and rendering are excluded. The Ultralytics timer surrounds the prediction call. The Hailo timer covers input preparation, accelerator transfer and execution, with decoding and ROI filtering timed separately. Stage boundaries thus differ by backend. The scripts use call-boundary timing without an explicit CUDA synchronization call. FPS measures headless computational throughput, not capture-to-decision latency [37].

The 1% Low FPS statistic is the reciprocal of the mean processing time for the slowest $\max(1, \lfloor 0.01N \rfloor)$ images in a configuration. For $N = 362$, it uses three images and quantifies the within-session timing tail. On the laptop, 1% Low FPS was 27.7–37.3% below mean FPS for the four YOLO models and 60.5% below mean FPS for PP.

For the available fixed-threshold counts, we quantified F1 score uncertainty with a multinomial parametric bootstrap of the TP/FP/FN triplet (200,000 replicates; seed 4,594,883). The 2.5th and 97.5th percentiles form the reported fixed-split 95% intervals. These intervals quantify count uncertainty at the evaluated operating point; they do not represent repeated training, recording-grouped sampling or independent hardware-timing sessions.

The evaluation reports pooled fixed-threshold detection and colour outcomes. $\text{mAP}_{50:95}$ requires score-ranked predictions at IoU thresholds from 0.50 to 0.95 in 0.05 incre-

ments, with the class definition and ROI policy specified; filtering at confidence 0.40 truncates a precision–recall curve [40]. The retained benchmark export contains thresholded aggregate counters; so, this manuscript reports the verified operating-point metrics rather than reconstructing AP from insufficient statistics. The software versions recorded for the experimental workflow are summarized in Table 8 and documented through the corresponding component references [33,45,46,48–55].

Table 8. Software versions used in the experimental workflow, assigned to the stage or execution platform for which they were reported.

Component	Version	Experimental Stage/Platform
Python	3.11	Training orchestration, benchmark scripts and application code; patch version not specified
Ultralytics	8.4.58	YOLO training and source-model inference on the laptop, Google Colab and Raspberry Pi CPU paths
OpenCV	4.13.0.92	Image preparation, classical detection, ROI filtering and post-processing on all benchmark platforms
NumPy	2.4.6	Array processing and aggregate metric computation
PyTorch	2.12.0	YOLO training and source-model inference; not the Hailo runtime
CUDA Toolkit	13.3.0	GPU-enabled YOLO training and laptop source-model inference
cuDNN	9.22.0	GPU backend for training and source-model inference
ONNX	1.21.0	Intermediate model export in the HEF conversion workflow
Hailo Dataflow Compiler	3.33.1	HEF optimization and compilation in the WSL/Ubuntu conversion environment
HailoRT	4.23.0	HEF execution on Raspberry Pi 5 with Hailo-8
Hailo Model Zoo	2.17.0	YOLOv8/YOLO11 architecture configurations used during model conversion

7. Results

7.1. Reference Detection and Colour Classification Results

Table 9 gives the reference results on the MSI Katana platform at 640×480 pixels with the relative image brightness ratio disabled. TP, FP, and FN counts are included to show whether errors are dominated by missed lamps or false detections.

Table 9. Detection and colour-classification quality in the reference configuration: MSI Katana, 640×480 pixels, relative image brightness ratio disabled. Conditional colour accuracy is Correct/TP; joint correct-colour coverage is Correct/(TP + FN).

Method	TP	FP	FN	Prec.	Recall	F1	Avg. IoU	Cond. Colour	Joint Coverage
YOLO26	644	41	265	0.9401	0.7085	0.8080	0.7239	0.9907	0.7019
YOLOv12	576	120	333	0.8276	0.6337	0.7178	0.6880	0.9913	0.6282
YOLO11	341	9	568	0.9743	0.3751	0.5417	0.7287	0.9941	0.3729
YOLOv8	329	10	580	0.9705	0.3619	0.5272	0.7355	0.9909	0.3586
PP	221	316	688	0.4115	0.2431	0.3057	0.6605	0.7466	0.1815
Top-Hat–night	98	730	811	0.1184	0.1078	0.1128	0.5704	0.5408	0.0583
Canny–day	75	367	834	0.1697	0.0825	0.1110	0.5951	0.6800	0.0561
Canny–night	77	1116	832	0.0645	0.0847	0.0733	0.6229	0.7143	0.0605
Top-Hat–day	14	255	895	0.0520	0.0154	0.0238	0.5205	0.4286	0.0066

Figure 7 shows detection metrics and error counts in a common method order. YOLO26 reached the highest F1 score, 0.8080, with a fixed-split 95% bootstrap interval of 0.7867–0.8286.

YOLOv12 followed at $F1 = 0.7178$ (0.6925–0.7421). YOLOv8 and YOLO11 combined high precision with low recall, indicating that missed lamps dominated their errors. PP had the highest classical method F1 score, 0.3057 (0.2746–0.3367), below all YOLO variants. The intervals in Table 10 quantify count uncertainty under the evaluated operating point.

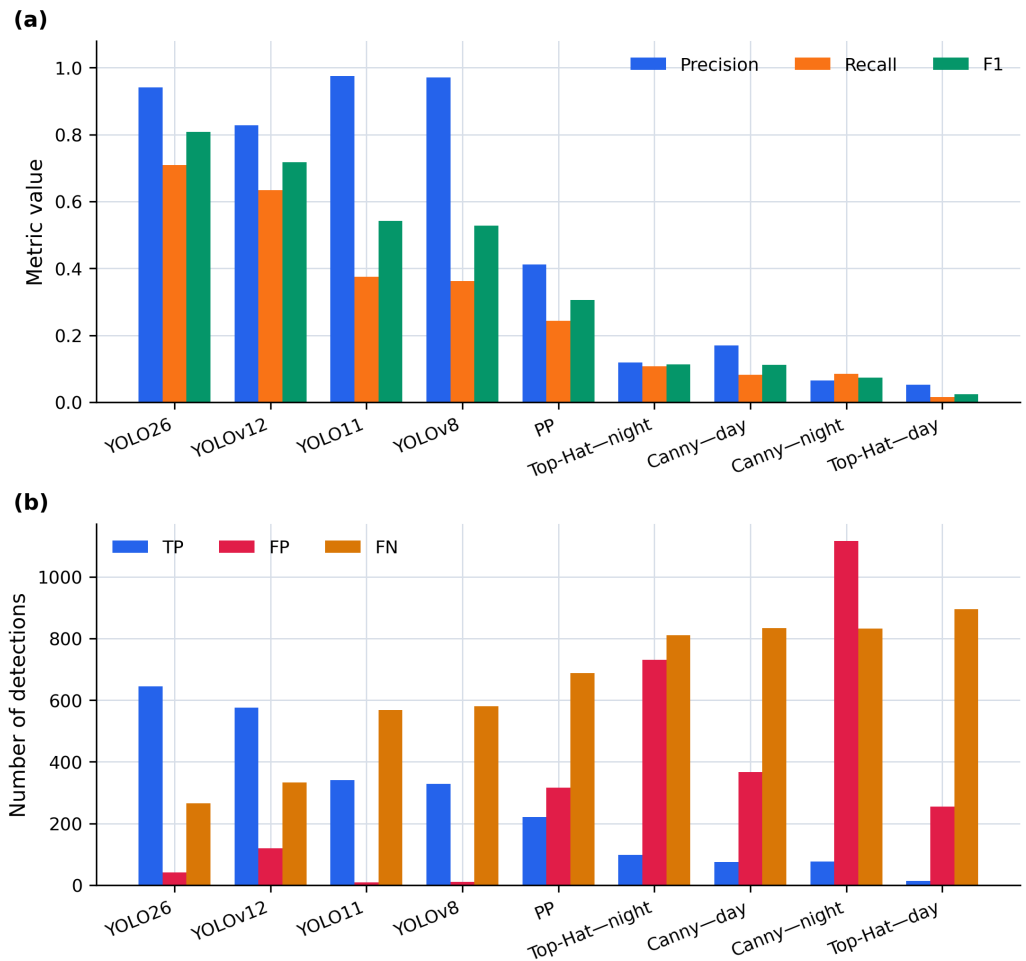


Figure 7. Reference detection results at 640×480 pixels on the MSI Katana platform: (a) precision, recall, and F1 score; (b) true-positive, false-positive, and false-negative counts. The common method order facilitates comparison between aggregate quality and error structure.

Table 10. Fixed -split F1 score estimates and 95% parametric-bootstrap intervals for the reference configuration. The bootstrap resamples the reported TP/FP/FN triplet as a multinomial outcome (200,000 replicates; seed 4,594,883). These intervals quantify count uncertainty at the fixed operating point; they do not represent repeated training or hardware-timing variation.

Method	F1	95% Lower	95% Upper
YOLO26	0.8080	0.7867	0.8286
YOLOv12	0.7178	0.6925	0.7421
YOLO11	0.5417	0.5073	0.5745
YOLOv8	0.5272	0.4934	0.5607
PP	0.3057	0.2746	0.3367
Top-Hat-night	0.1128	0.0931	0.1333
Canny-day	0.1110	0.0884	0.1345
Canny-night	0.0733	0.0576	0.0887
Top-Hat-day	0.0238	0.0120	0.0371

Conditional colour accuracy for matched YOLO detections ranged from 0.9907 to 0.9941, whereas joint correct-colour coverage was lower because it includes missed lamps. For YOLO26, 638 of 644 matches had the correct colour, three had an incorrect recognized colour and three were unclassified; 265 lamps were missed. Its conditional accuracy was 99.07% (Wilson 95% interval 97.98–99.57%), while joint coverage was 70.19% (67.13–73.07%). YOLOv12 had corresponding values of 99.13% (97.98–99.63%) and 62.82% (59.63–65.90%). Table 11 verifies the four-way decomposition. Among classical methods, only Top-Hat-night produced an unclassified annular-voting outcome: one of 98 matches (1.02%); the other listed classical configurations produced zero. Yellow comprises 337 of 18,141 original annotations (1.86%), giving a white-to-yellow imbalance of 43.8:1.

Table 11. Colour outcomes for geometrically matched detections (TP) on MSI Katana at 640×480 , with the relative image brightness ratio disabled. Incorrect colour denotes a recognized label different from the ground truth; unclassified denotes no recognized four-colour label after normalization. Each row satisfies Correct + Incorrect + Unclassified = TP and TP + FN = 909 eligible ground-truth lamps.

Method	TP	Correct Colour	Incorrect Colour	Unclassified	Unclassified [% of TP]	FN
YOLO26	644	638	3	3	0.47	265
YOLOv12	576	571	2	3	0.52	333
YOLO11	341	339	2	0	0.00	568
YOLOv8	329	326	3	0	0.00	580
PP	221	165	56	0	0.00	688
Top-Hat-night	98	53	44	1	1.02	811
Canny-day	75	51	24	0	0.00	834
Canny-night	77	55	22	0	0.00	832
Top-Hat-day	14	6	8	0	0.00	895

7.2. Influence of Scene Conditions

Table 12 reports F1 score values by scene condition for the MSI Katana reference platform at 640×480 pixels.

Table 12. F1 score by scene condition on MSI Katana at 640×480 pixels, brightness module disabled. Headers give images/eligible GT lamps for every method in that column. Comparisons are descriptive, not controlled estimates of illumination or weather effects.

Method	Day Good 194/430	Day Reflections 111/328	Twilight Good 15/59	Twilight Rain-Fog 24/64	Night Good 18/28
YOLO26	0.7852	0.8633	0.8571	0.5660	0.8846
YOLOv12	0.6640	0.7759	0.8598	0.5472	0.8679
YOLO11	0.4754	0.6134	0.6667	0.2400	0.8400
YOLOv8	0.4476	0.6089	0.6364	0.2703	0.8400
PP	0.1562	0.5354	0.3467	0.0000	0.4348
Canny-day	0.0459	0.1538	0.2025	0.0000	0.4918
Canny-night	0.0340	0.1195	0.0600	0.0000	0.3939
Top-Hat-day	0.0157	0.0113	0.1707	0.0000	0.0000
Top-Hat-night	0.0345	0.2089	0.2745	0.0000	0.3881

Figure 8 shows the lowest F1 score in the twilight/rain-fog group for all YOLO models, PP, Canny-Hough and Top-Hat-night; Top-Hat-day ties at zero for twilight/rain-fog and night/good. The twilight/rain-fog group contains 24 images and 64 eligible lamps. YOLO26 yielded TP/FP/FN of 30/12/34 there, compared with 23/1/5 among 18 night/good images containing 28 lamps. The two groups differ in sample count and may also differ in lamp size, distance, colour composition, geometry and recording content. Higher night/good F1 score is compatible with greater contrast, but neither contrast nor

weather was isolated experimentally. The paired counts are descriptive observations, not evidence of a causal weather effect.

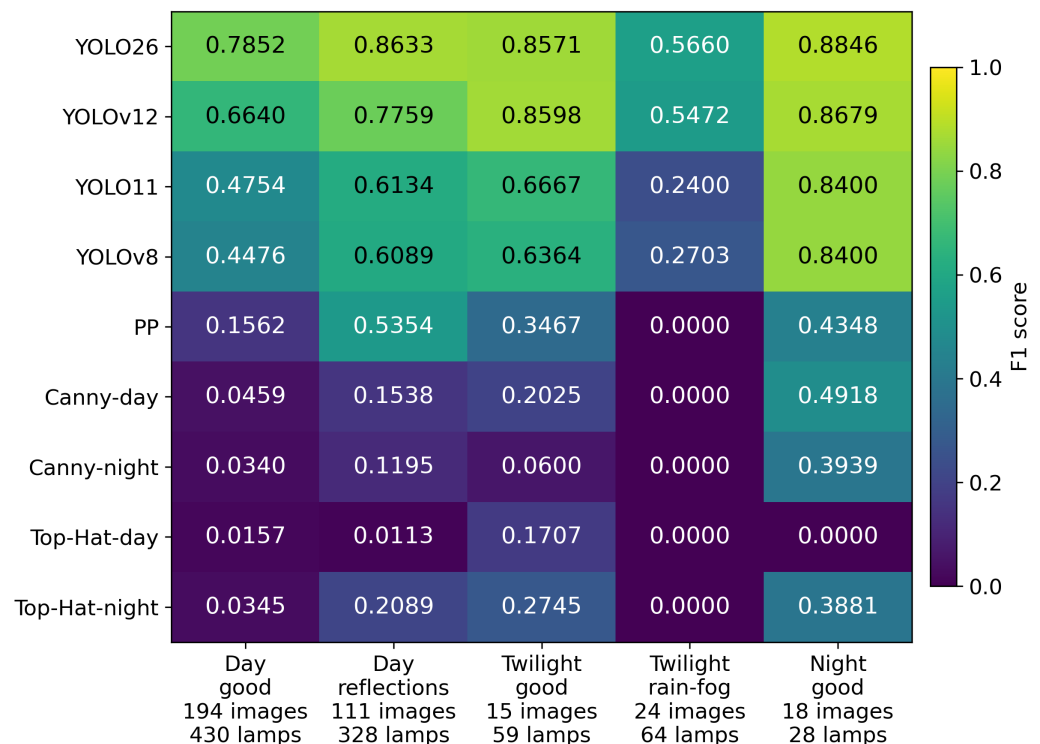


Figure 8. F1 score by method and scene condition at 640×480 pixels on MSI Katana. Below each scene name, the image count and number of eligible ground-truth lamps in the ROI are shown on separate lines. These counts apply to every method in a column. Scene groups are unequal and confounded; so, comparisons are descriptive rather than controlled weather or illumination effects.

7.3. Influence of Working Resolution

Table 13 compares 640×480 and 320×240 pixels on the MSI Katana platform.

Table 13. Effect of reducing the working resolution from 640×480 to 320×240 pixels on the MSI Katana platform. Metrics are rounded to three decimals; $\Delta F1$ was calculated from unrounded benchmark outputs.

Method	Precision		Recall		F1		$\Delta F1$ [%]
	640	320	640	320	640	320	
YOLO26	0.940	0.776	0.709	0.291	0.808	0.423	−47.6
YOLOv12	0.828	0.603	0.634	0.203	0.718	0.303	−57.7
YOLO11	0.974	0.809	0.375	0.117	0.542	0.204	−62.3
YOLOv8	0.971	0.851	0.362	0.126	0.527	0.219	−58.5
PP	0.411	0.379	0.243	0.106	0.306	0.165	−45.9
Canny-day	0.170	0.146	0.083	0.032	0.111	0.052	−52.8
Canny-night	0.065	0.059	0.085	0.036	0.073	0.045	−38.6
Top-Hat-day	0.052	0.071	0.015	0.015	0.024	0.025	6.3
Top-Hat-night	0.118	0.115	0.108	0.046	0.113	0.066	−41.5

Every YOLO model had a lower F1 score at 320×240 , with relative decreases of 47.6% to 62.3% (Figure 9). For YOLO26, TP decreased from 644 to 264 and recall from 0.7085 to 0.2907; for YOLOv12, TP decreased from 576 to 184 and recall from 0.6337 to 0.2026. Halving both image dimensions quarters nominal box area before rounding; so, this controlled scale perturbation confirms strong sensitivity to reduced pixel support.

The small increase for Top-Hat-day occurs at very low F1 score and does not demonstrate improved practical reliability.

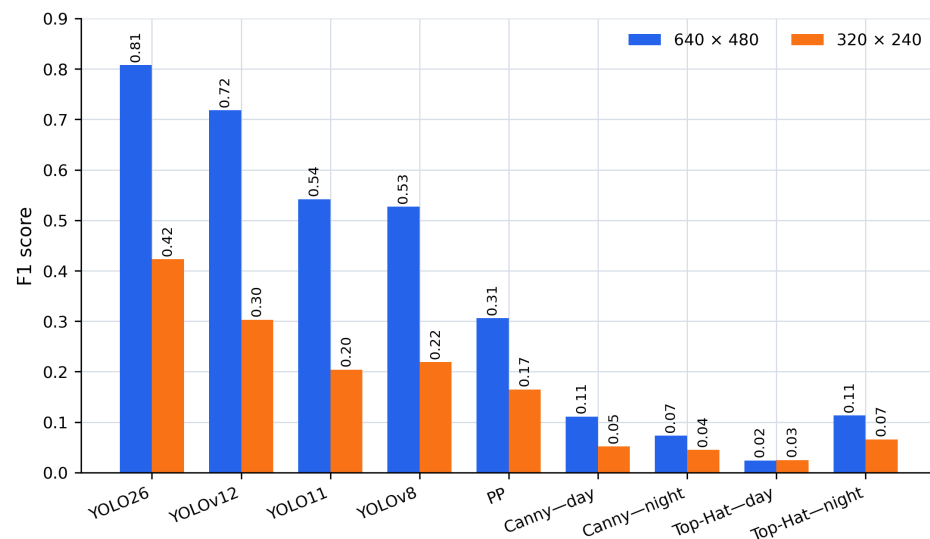


Figure 9. Influence of working resolution on F1 score for the MSI Katana benchmark. The reduction from 640×480 to 320×240 pixels substantially reduced recall and aggregate detection quality for the learned detectors.

7.4. Platform Throughput and HEF Conversion

Table 14 compares F1 score and FPS across platforms at 640×480 pixels with the relative brightness module disabled.

Table 14. Detection quality and throughput by platform at 640×480 pixels with the relative brightness module disabled.

Method	MSI Katana		Raspberry Pi 5 CPU		Raspberry Pi 5 + Hailo-8	
	F1	FPS	F1	FPS	F1	FPS
YOLO26	0.8080	64.0	0.8080	1.3	—	—
YOLOv12	0.7178	53.8	0.7165	1.1	—	—
YOLO11	0.5417	75.4	0.5417	1.3	0.2842	36.4
YOLOv8	0.5272	82.3	0.5272	1.1	0.2732	48.6
PP	0.3057	622.4	0.3057	393.5	0.3057	364.5
Canny-day	0.1110	196.0	0.1089	227.2	0.1110	155.8
Canny-night	0.0733	187.0	0.0755	193.2	0.0733	140.0
Top-Hat-day	0.0238	538.4	0.0238	284.5	0.0238	269.4
Top-Hat-night	0.1128	73.1	0.1128	20.8	0.1128	20.6

For non-converted methods, platform-level F1 score values were identical or differed by, at most, 0.0022, while speed changed substantially. YOLO models that reached 53.8–82.3 FPS on the laptop reached only 1.1–1.3 FPS on Raspberry Pi 5 CPU. Classical methods remained much faster on the embedded CPU; PP reached 393.5 FPS but only 0.3057 F1 score.

Hailo-8 increased throughput for converted YOLOv8 and YOLO11, but the HEF variants had a lower F1 score than their PyTorch/Ultralytics counterparts. Table 15 gives the size of the change. YOLOv8 increased from about 1.1 FPS on Raspberry Pi 5 CPU to 48.6 FPS on Hailo-8, while the F1 score decreased from 0.5272 to 0.2732. YOLO11 increased from about 1.3 FPS to 36.4 FPS, while the F1 score decreased from 0.5417 to 0.2842.

The conversion chain substantially reduced the inference-related throughput limitation for these models but reduced detection quality in the tested setup.

Table 15. Comparison of YOLO models executed on Raspberry Pi 5 CPU as PyTorch weights and on Raspberry Pi 5 with Hailo-8 as HEF models.

Model	F1			Recall			Throughput	
	CPU (.pt)	Hailo (.hef)	Δ [%]	CPU (.pt)	Hailo (.hef)	Δ [%]	Hailo FPS	Speed-Up
YOLOv8	0.5272	0.2732	−48.2%	0.3619	0.1617	−55.3%	48.6	$\approx 44\times$
YOLO11	0.5417	0.2842	−47.5%	0.3751	0.1771	−52.8%	36.4	$\approx 28\times$

Figure 10 shows the trade-off between throughput and detection quality. Classical pipelines occupy the high-throughput, low-F1 score region. Laptop YOLO configurations combine a higher F1 score with tens of frames per second; Raspberry Pi 5 CPU retains a similar F1 score near one frame per second. Hailo-8 increases YOLOv8 and YOLO11 throughput, but both F1 score and recall decrease (panel (b)). Table 16 shows that YOLOv8 predictions decreased by 50.7% and TP by 55.3%, while FP doubled; YOLO11 predictions decreased by 36.0% and TP by 52.8%, while FP increased from 9 to 63. The common pattern is reduced detection coverage after HEF deployment, with an additional precision loss for YOLO11.

Table 16. Detection counts at the source-model and Hailo deployment endpoints at 640×480 , with brightness screening disabled. Source-model counts match the CPU reference; Hailo counts describe HEF execution. Each endpoint has 909 eligible ground-truth lamps.

Model	Endpoint	Predictions	TP	FP	FN	F1
YOLOv8	Source (.pt)	339	329	10	580	0.5272
YOLOv8	Hailo (.hef)	167	147	20	762	0.2732
YOLO11	Source (.pt)	350	341	9	568	0.5417
YOLO11	Hailo (.hef)	224	161	63	748	0.2842

7.5. Relative Image Brightness Ratio Overhead

We tested the relative image brightness ratio as an additional post-processing step, not as a calibrated photometric estimator. Table 17 reports its timing effects. Post-processing time increased slightly, with small changes in mean FPS relative to the dominant detector cost.

Table 17. Representative effect of enabling the relative image brightness ratio at 640×480 pixels.

Platform	Method	FPS Off	FPS On	Δ FPS	Post. Off [ms]	Post. On [ms]
MSI Katana	YOLO26	64.0	61.4	−4.06%	0.42	0.60
MSI Katana	PP	622.4	585.2	−5.98%	0.25	0.33
MSI Katana	Top-Hat-day	538.4	518.2	−3.75%	0.08	0.12
Raspberry Pi 5 CPU	YOLO26	1.3	1.3	0.00%	0.29	0.52
Raspberry Pi 5 CPU	PP	393.5	375.2	−4.65%	0.42	0.54
Raspberry Pi 5 + Hailo-8	YOLOv8	48.6	49.1	1.03%	0.13	0.16
Raspberry Pi 5 + Hailo-8	PP	364.5	341.2	−6.39%	0.45	0.58

The positive FPS difference for Hailo YOLOv8 is smaller than the observed within-session timing-tail spread and is not interpreted as acceleration. This comparison measures implementation overhead only.

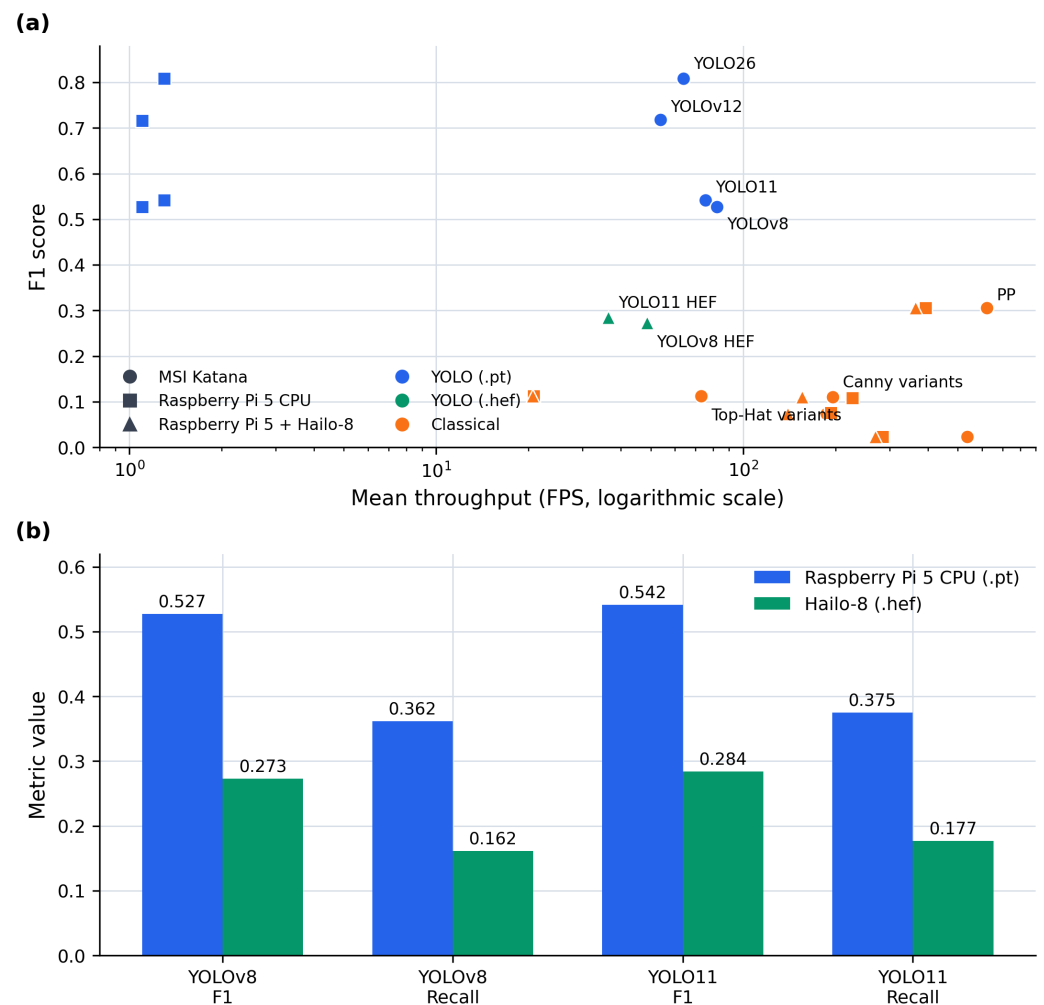


Figure 10. Platform and conversion trade-offs at 640×480 pixels: (a) F1 score versus measured headless throughput on the laptop, Raspberry Pi 5 CPU, and Raspberry Pi 5 with Hailo-8; (b) F1 score and recall before conversion on Raspberry Pi 5 CPU and after HEF conversion on Hailo-8. The horizontal axis in panel (a) is logarithmic.

8. Discussion

The measurements reveal a clear trade-off between detection quality and headless throughput. YOLO26 had the highest aggregate F1 score, followed by YOLOv12; their fixed-split bootstrap intervals did not overlap. YOLOv8 and YOLO11 combined high precision with lower recall at confidence 0.40. For the learned detectors, missed lamps account for more unrecovered ground-truth instances than incorrect recognized colours (Table 11). These rankings apply to the evaluated checkpoints, split and operating point. Maintenance or compliance decisions require qualified human review of the inspection outputs.

PP was the fastest method and remained lightweight on Raspberry Pi 5, making it a possible auxiliary detector or candidate generator. Its low F1 score limits standalone use in mixed airport scenes. Canny–Hough and White Top-Hat were more sensitive to parameter choice and scene variability, particularly reflections, twilight, weather changes and structured backgrounds.

The twilight/rain–fog group produced the lowest or tied-lowest F1 score across configurations in this test subset. Lower contrast, reflections, scattering and glare may contribute, but the comparison does not establish their causal effects. Image and lamp counts, colour, distance, apparent size, geometry and recording membership may covary. Further acquisi-

tion should balance these factors and retain recording identifiers for grouped evaluation and paired uncertainty estimates.

Reducing the working resolution from 640×480 to 320×240 can increase speed, but sharply reduced recall and F1 score. Pixel support becomes a limiting factor for small lamps. Lower-resolution deployment would require retraining, retuning or a different model design rather than reuse of the same operating point.

Raspberry Pi 5 CPU retained approximately the source-model detection quality at much lower throughput than the laptop. Hailo-8 reached 36.4–48.6 FPS for the converted models, but both F1 score and recall decreased. The endpoint counts show a common loss of TP and increase in FN, together with a model-dependent FP change. This indicates that deployment validation must assess output fidelity as well as throughput.

Local processing on inspection vehicles can reduce dependence on continuous video transfer to centralized computing. The reported FPS values describe capacity only within the measured processing boundary. Platform power, energy per frame, thermal constraints and duty cycles would extend this comparison [26].

The relative image brightness ratio fits its reference to the same detections it assesses. Exactly three observations can be fitted without residual error, limiting independent abnormality detection. It is not used as evidence of a fault or regulatory compliance.

8.1. Synthesis of the Evaluation Findings

At the fixed operating point, the evaluated YOLO checkpoints had higher aggregate F1 score than the classical pipelines. Lower resolution reduced detection quality, and scene groups differed descriptively. Hailo-8 increased throughput for two converted models while changing their detection output. Joint correct-colour coverage accounts for misses that conditional colour accuracy excludes, making it useful alongside F1 score when selecting detectors and platforms for inspection support.

8.2. Limitations and Further Validation

The dataset consists of manually selected frames from one airport and uses a frame-level split. The fixed-split bootstrap intervals quantify uncertainty in the reported TP/FP/FN counts but not recording dependence, repeated training or independent timing sessions. A complete extension should use recording-grouped splits and repeated seeds/sessions; export score-ranked predictions for $mAP_{50:95}$ and confidence/IoU sensitivity; retain per-image boxes for predefined lamp-area bins and class-wise confusion including yellow support and unclassified outcomes; and compare the same checkpoint and samples through PyTorch, ONNX and HEF with aligned tensors, NMS, coordinates, ROI rules and class mapping.

Augmentation analysis should also preserve original/augmented pairs and compare augmentation-on/off training across independent seeds. Inspired by the cosine-similarity-guided comparison of state representations used by Zeng et al. for visual reinforcement-learning attention [56], object-region and global embeddings from a fixed detector layer could be evaluated together with generalization, robustness and training variance. Calibrated lamp measurements are separately required before the relative image brightness ratio can be interpreted photometrically.

Operational evaluation would require calibrated acquisition geometry and exposure, cross-airport recordings and repeated vehicle trajectories. The system remains a human-supervised inspection prototype, not a certified aeronautical safety system.

9. Conclusions

We compared established detectors for vision-based airfield-light inspection under a documented application-level protocol. The dataset comprises 59 restricted-area recordings acquired through scientific cooperation with Poznań-Ławica Airport, 3612 original frames and 18,141 lamp annotations, with 362 frames held out for testing. The benchmark covers detection, branch-specific colour assignment and embedded execution.

YOLO26 had the highest reference F1 score, reaching $F1 = 0.8080$ (fixed-split 95% interval 0.7867–0.8286) at 64.0 FPS on the laptop at 640×480 pixels. YOLOv12 reached $F1 = 0.7178$ (0.6925–0.7421) at 53.8 FPS. PP was faster, at 622.4 FPS on the laptop and 393.5 FPS on Raspberry Pi 5 CPU, but its F1 score was 0.3057. Raspberry Pi 5 CPU YOLO execution reached only 1.1–1.3 FPS. Hailo-8 increased converted YOLOv8 and YOLO11 throughput to 48.6 and 36.4 FPS, respectively, with lower F1 score after HEF conversion.

The twilight/rain–fog subset produced the lowest or tied-lowest observed F1 score across configurations, although unequal and confounded groups allow for only descriptive interpretation. Downscaling to 320×240 reduced YOLO detection quality. Separating incorrect colour, unclassified outcomes and missed lamps makes joint inspection coverage explicit. Detector and platform selection for human-supervised maintenance planning should assess deployment quality together with computational throughput.

Author Contributions: Conceptualization, J.S. and J.T.J.; methodology, J.T.J. and J.S.; software, J.T.J.; validation, J.T.J.; formal analysis, J.T.J.; investigation, J.T.J.; resources, J.S.; data curation, J.T.J.; writing—original draft preparation, J.S.; writing—review and editing, J.S.; supervision, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: Funded by Poznan University of Technology, grant 0211/SBAD/0225.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Annotated frames and labels are provided through the identified Roboflow Universe dataset [29], whose project page specifies CC BY 4.0. The public project repository contains benchmark and training source code, model-checkpoint files for the four YOLO variants, and configuration-level aggregate CSV results [37]. The original airport video recordings are restricted and cannot be publicly released.

Acknowledgments: The authors thank Poznań-Ławica Airport for scientific cooperation and for enabling controlled acquisition of the source recordings in restricted airfield operating areas.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

AI	Artificial intelligence
CPU	Central processing unit
CSV	Comma-separated values
F1	Harmonic mean of precision and recall
FN	False negative
FP	False positive
FPS	Frames per second
GPU	Graphics processing unit
HEF	Hailo Executable Format
HI	Implementation variable name for the relative image brightness ratio R_B
HSV	Hue, saturation, value colour space
ICAO	International Civil Aviation Organization
INT8	8-bit integer representation

IoU	Intersection over Union
NMS	Non-maximum suppression
ONNX	Open Neural Network Exchange
PP	Project designation for the brightness-thresholding classical pipeline
ROI	Region of interest
TP	True positive
YOLO	You Only Look Once

References

1. Air Accident Investigation Sector. *Aircraft Struck Runway Edge Lights on Landing*; Incident Final Report AIFN/0008/2019; General Civil Aviation Authority of the United Arab Emirates: Abu Dhabi, United Arab Emirates, 2020. Available online: <https://skybrary.aero/sites/default/files/bookshelf/5914.pdf> (accessed on 17 September 2026).
2. Todd, M.A. *Factors Influencing Misaligned Take-Off Occurrences at Night*; Aviation Research and Analysis Report AR-2009-033; Australian Transport Safety Bureau: Canberra, Australia, 2010. Available online: <https://www.atsb.gov.au/sites/default/files/media/1543486/ar2009033.pdf> (accessed on 17 September 2026).
3. Jentsch, F. The Effects of Taxiway Light Geometry, Color, and Location on Position Determination by Pilots. *Proc. Hum. Factors Ergon. Soc. Annu. Meet.* **1994**, *38*, 76–80. [\[CrossRef\]](#)
4. Bullough, J.D. Human Factors Impacts of Light-Emitting Diode Airfield Lighting. *Transp. Res. Rec. J. Transp. Res. Board* **2017**, *2626*, 51–57. [\[CrossRef\]](#)
5. Talbi, A.; Abdellaoui, M.; Pons-Prats, J.; Kuljanin, J. An Overview on Computer Vision Analysis in the Airport Applications. *IEEE Access* **2025**, *13*, 120446–120471. [\[CrossRef\]](#)
6. Ortega Alba, S.; Manana, M. Energy Research in Airports: A Review. *Energies* **2016**, *9*, 349. [\[CrossRef\]](#)
7. Pamuła, W.; Pamuła, T.; Stenzel, T.; Sajkowski, M.; Rytter, A.; Żuchowska, D. UAV-Based Airport Lights Inspection Without GNSS Positioning Support. *Remote Sens.* **2025**, *17*, 1013. [\[CrossRef\]](#)
8. Podbucki, K.; Marciniak, T. Camera-Based Light Source Localization for Precise Luminous Intensity Measurement of Airport Lamps. In Proceedings of the Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznań, Poland, 25–27 September 2024; pp. 72–78. [\[CrossRef\]](#)
9. Suder, J.; Podbucki, K.; Marciniak, T. Chromaticity measurement of airport navigation lighting using integrated colour sensor. *Opto-Electron. Rev.* **2023**, *31*, e147040. [\[CrossRef\]](#)
10. Podbucki, K.; Marciniak, T.; Suder, J. Laboratory Assessment of In-Pavement Airport Lamp's Luminous Intensity Distribution. *Appl. Sci.* **2023**, *13*, 13242. [\[CrossRef\]](#)
11. Cheng, G.; Yuan, X.; Yao, X.; Yan, K.; Zeng, Q.; Xie, X.; Han, J. Towards Large-Scale Small Object Detection: Survey and Benchmarks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13467–13488. [\[CrossRef\]](#)
12. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 936–944. [\[CrossRef\]](#)
13. de Charette, R.; Nashashibi, F. Real-Time Visual Traffic Lights Recognition Based on Spot Light Detection and Adaptive Traffic Lights Templates. In Proceedings of the 2009 IEEE Intelligent Vehicles Symposium, Xi'an, China, 3–5 June 2009; pp. 358–363. [\[CrossRef\]](#)
14. Sitompul, D.S.D.; Surya, F.E.; Suhandi, F.P.; Zakaria, H. Runway Edge Light Photometry System by Using Drone-Mounted Instrument. In Proceedings of the 2019 International Symposium on Electronics and Smart Devices (ISESD), Badung-Bali, Indonesia, 8–9 October 2019; pp. 1–5. [\[CrossRef\]](#)
15. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [\[CrossRef\]](#)
16. Duda, R.O.; Hart, P.E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM* **1972**, *15*, 11–15. [\[CrossRef\]](#)
17. Serra, J. *Image Analysis and Mathematical Morphology*; Academic Press: London, UK, 1982.
18. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [\[CrossRef\]](#)
19. Ramos, L.T.; Sappa, A.D. A Decade of You Only Look Once (YOLO) for Object Detection: A Review. *IEEE Access* **2025**, *13*, 192747–192794. [\[CrossRef\]](#)
20. Ultralytics. Ultralytics YOLO Model Documentation. Available online: <https://docs.ultralytics.com/models/> (accessed on 17 August 2026).
21. Tian, Y.; Ye, Q.; Doermann, D. YOLOv12: Attention-Centric Real-Time Object Detectors. In Proceedings of the Advances in Neural Information Processing Systems 38, San Diego, CA, USA, 2–7 December 2025; Mexico City, Mexico, 30 November–5 December 2025; pp. 87151–87175. [\[CrossRef\]](#)

22. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**, arXiv:2606.03748. [[CrossRef](#)]
23. Kisantal, M.; Wojna, Z.; Murawski, J.; Naruniec, J.; Cho, K. Augmentation for Small Object Detection. *arXiv* **2019**, arXiv:1902.07296. [[CrossRef](#)]
24. Suder, J.; Marciniak, T. Foreign Object Debris Detection at Aerodromes Using YOLOv5. In Proceedings of the 2024 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznań, Poland, 25–27 September 2024; pp. 66–71. [[CrossRef](#)]
25. Suder, J.; Marciniak, T.; Dąbrowski, A. Vision System for Foreign Object Debris (FOD) Detection Using Artificial Intelligence. In Proceedings of the IEEE EUROCON 2025—21st International Conference on Smart Technologies, Gdynia, Poland, 4–6 June 2025. [[CrossRef](#)]
26. Suder, J.; Podbucki, K.; Marciniak, T. Power Requirements Evaluation of Embedded Devices for Real-Time Video Line Detection. *Energies* **2023**, *16*, 6677. [[CrossRef](#)]
27. GoPro, Inc. *HERO8 Black User Manual*; GoPro, Inc.: San Mateo, CA, USA, 2019. Available online: https://gopro.com/content/dam/help/hero8-black/manuals/HERO8Black_UM_ENG_REVB.pdf (accessed on 17 August 2026).
28. VideoLAN Organization. VLC Media Player (Version 3.0.23). Available online: <https://www.videolan.org> (accessed on 17 August 2026).
29. Józwiak, J.T. *Magisterka Airfield-Light Dataset, Version 2*; Roboflow Universe: Des Moines, IA, USA, 2026. Available online: <https://universe.roboflow.com/magisterka-a1kbw/magisterka-ezzug/dataset/2> (accessed on 17 August 2026).
30. International Civil Aviation Organization (ICAO). *Annex 14 to the Convention on International Civil Aviation: Aerodromes, Volume I—Aerodrome Design and Operations*, 9th ed.; ICAO: Montreal, QC, Canada, 2022. Available online: <https://store.icao.int/en/annex-14-aerodromes> (accessed on 17 August 2026).
31. European Union Aviation Safety Agency (EASA). Easy Access Rules for Aerodromes (Regulation (EU) No 139/2014), Revision from March 2026; CS ADR-DSN.M.675 and CS ADR-DSN.M.710. Available online: <https://www.easa.europa.eu/en/document-library/easy-access-rules/online-publications/easy-access-rules-aerodromes-regulation-eu?erules-id=ERULES-1963177438-2481> (accessed on 4 October 2026).
32. Zoph, B.; Cubuk, E.D.; Ghiasi, G.; Lin, T.-Y.; Shlens, J.; Le, Q.V. Learning Data Augmentation Strategies for Object Detection. *arXiv* **2019**, arXiv:1906.11172. [[CrossRef](#)]
33. PyTorch Foundation. PyTorch 2.12 Release. Available online: <https://pytorch.org/blog/pytorch-2-12-release-blog/> (accessed on 4 October 2026).
34. NVIDIA Corporation. CUDA Toolkit Documentation. Available online: <https://docs.nvidia.com/cuda/> (accessed on 17 August 2026).
35. NVIDIA Corporation. NVIDIA cuDNN. Available online: <https://docs.nvidia.com/deeplearning/cudnn/latest/> (accessed on 17 August 2026).
36. Ultralytics. Python Usage. Available online: <https://docs.ultralytics.com/usage/python> (accessed on 17 August 2026).
37. Józwiak, J.T. *Praca-Magisterska: Airfield-Light Detection and Benchmark Source Code*. Software Repository. 2026. Available online: <https://github.com/jakubjozwiak002/Praca-Magisterska> (accessed on 17 August 2026).
38. OpenCV. OpenCV 4.x Documentation. Available online: <https://docs.opencv.org/4.x/> (accessed on 17 August 2026).
39. Smith, A.R. Color Gamut Transform Pairs. *ACM SIGGRAPH Comput. Graph.* **1978**, *12*, 12–19. [[CrossRef](#)]
40. Padilla, R.; Netto, S.L.; da Silva, E.A.B. A Survey on Performance Metrics for Object-Detection Algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing, Niterói, Brazil, 1–3 July 2020; pp. 237–242. [[CrossRef](#)]
41. Python Software Foundation. tkinter—Python Interface to Tcl/Tk. Available online: <https://docs.python.org/3/library/tkinter.html> (accessed on 17 August 2026).
42. Raspberry Pi Ltd. Raspberry Pi Computer Hardware. Available online: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html> (accessed on 17 August 2026).
43. Raspberry Pi Ltd. AI HATs. Available online: <https://www.raspberrypi.com/documentation/accessories/ai-hat-plus.html> (accessed on 17 August 2026).
44. ONNX Community. ONNX Documentation. Available online: <https://onnx.ai/onnx/> (accessed on 17 August 2026).
45. Hailo Technologies Ltd. Dataflow Compiler v3.33.1. Available online: <https://hailo.ai/developer-zone/documentation/dataflow-compiler-v3-33-1/> (accessed on 17 August 2026).
46. Hailo Technologies Ltd. HailoRT v4.23.0. Available online: <https://hailo.ai/developer-zone/documentation/hailort-v4-23-0/> (accessed on 17 August 2026).
47. Tapuhi, T.; Gluska, S.; Klinger, A.; Dharan, N.; Sholev, O.; Zelig, Y.; Vosco, N. Hailo Model Zoo. Software Repository. 2021. Available online: https://github.com/hailo-ai/hailo_model_zoo (accessed on 17 August 2026).
48. Hailo Technologies Ltd. Hailo-8 Software Compatibility Configuration: HailoRT 4.23.0 and Hailo Model Zoo 2.17.0. Available online: https://github.com/hailo-ai/hailo-apps/blob/main/hailo_apps/config/config.yaml (accessed on 4 October 2026).

49. Python Software Foundation. Python 3.11 Documentation. Available online: <https://docs.python.org/3.11/> (accessed on 5 October 2026).
50. Ultralytics. Ultralytics 8.4.58 Release. Available online: <https://pypi.org/project/ultralytics/8.4.58/> (accessed on 4 October 2026).
51. OpenCV. opencv-python 4.13.0.92 Release. Available online: <https://pypi.org/project/opencv-python/4.13.0.92/> (accessed on 4 October 2026).
52. NumPy Developers. NumPy 2.4.6 Release. Available online: <https://pypi.org/project/numpy/2.4.6/> (accessed on 4 October 2026).
53. NVIDIA Corporation. CUDA Toolkit 13.3. Available online: <https://developer.nvidia.com/blog/nvidia-cuda-13-3-enhances-gpu-development-with-tile-programming-in-c-compiler-autotuning-and-python-updates> (accessed on 4 October 2026).
54. NVIDIA Corporation. cuDNN 9.22.0 Release Notes. Available online: <https://docs.nvidia.com/deeplearning/cudnn/backend/v9.22.0/release-notes.html> (accessed on 4 October 2026).
55. ONNX Community. ONNX 1.21.0 Release. Available online: <https://github.com/onnx/onnx/releases/tag/v1.21.0> (accessed on 4 October 2026).
56. Zeng, Y.; Wang, X.; Cheng, Y. Similarity-Guided State Attention for Visual Reinforcement Learning. *Neural Netw.* **2027**, *205*, 109560. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.