

M. Zhassuzak^{1*} , Z. Abdirazak¹ ,
D. Dauletiya² , M. Satymbekov³ 

¹S. Nurmagambetov Military Institute of Land Forces, Almaty, Kazakhstan

²Astana IT University, Astana, Kazakhstan

³Shakarim University, Semey, Kazakhstan

*e-mail: zhassuzak.mukhtar@gmail.com

SHOOTVISION: A REAL-TIME VISION-BASED SYSTEM FOR SHOOTING DETECTION AND MARKSMANSHIP ACCURACY ANALYSIS USING YOLOV8

Abstract. Objective and reproducible scoring of marksmanship is a long-standing challenge in sport shooting and firearms training, where manual scoring is slow and subjective and dedicated electronic targets are expensive. This paper presents ShootVision, a web-based system that performs real-time detection and accuracy analysis of shots using computer vision. The core of the system is a YOLOv8n (nano) single-stage detector, fine-tuned through transfer learning to perform binary classification of detected objects into two classes, hit and miss. A custom augmentation pipeline expands each source image into six variants through horizontal flipping, rotation with analytical recomputation of bounding-box coordinates, and brightness and contrast variation. To suppress false positives in a live video stream, ShootVision introduces a three-level detection-filtering scheme (confidence thresholding, a custom area-aware non-maximum suppression, and temporal smoothing over a sliding window) together with an original temporal-stability mechanism that confirms a shot only after repeated, spatially consistent detections. Hit accuracy is quantified by the Euclidean distance from each hit to the target centre and mapped to a ten-point score over concentric scoring zones. The system is delivered through a Django web platform that streams annotated video and offers four operating modes (classic, distance, target, and duel), a trilingual interface, and interactive analytics. The deployed system operates at approximately 30 frames per second and demonstrates that lightweight object detection combined with temporal-stability logic provides a low-cost, accessible alternative to specialised scoring hardware.

Keywords: object detection; YOLOv8; computer vision; marksmanship analysis; temporal stability; non-maximum suppression; real-time video.

1. Introduction

Shooting sports and firearms training depend on accurate, repeatable measurement of where projectiles strike a target. Such measurement underpins competitive scoring, structured skill development in military, law-enforcement, and recreational settings, and the objective tracking of a shooter's progress over time, so any reduction in its cost or latency has broad practical value. Conventional approaches fall into two broad categories. Manual scoring, in which an instructor inspects the target after a shooting series, is inexpensive but slow, error-prone, and inherently subjective; it also interrupts training because shooters must wait between series. Dedicated electronic scoring targets, based on acoustic, optical, or pressure sensors, deliver precise automatic results but are costly, mechanically fragile, and tied to specific hardware, which places them out of reach for many clubs, schools, and individual practitioners.

Recent advances in deep-learning-based object detection [1] make it possible to recover spatial information about shots directly from ordinary video, using only a camera and commodity computing. Single-stage detectors such as the YOLO family [2] achieve a favourable balance between accuracy and inference speed [3], which makes them suitable for real-time analysis. However, applying an object detector to live shooting video raises practical problems that a raw detector does not solve on its own: detections flicker between frames, the same physical shot can be reported multiple times, and visually similar marks can produce spurious detections. Robust scoring therefore requires additional logic layered on top of the detector.

This paper presents ShootVision, an end-to-end, web-based system that addresses these problems. Its contributions are as follows:

- A lightweight detection model: a YOLOv8n detector fine-tuned by transfer learning to distin-

guish hits from misses, chosen for real-time operation on modest hardware.

- A custom data-augmentation pipeline that generates five augmented variants per source image, so that the augmented training set is six times the size of the source set, and that analytically recomputes bounding-box coordinates under geometric transformations.

- A three-level detection-filtering scheme and a temporal-stability mechanism that together convert noisy per-frame detections into stable, de-duplicated shot events, together with a probabilistic model that relates per-frame detector recall to the probability of confirming a physical shot.

- A quantitative evaluation of the detector, including a decomposition of the confusion matrix that separates localisation error from class-confusion error.

- A distance-based accuracy metric that maps each hit to a ten-point score over concentric scoring zones.

- A complete Django web platform with four analysis modes, a trilingual interface, real-time video streaming, and interactive analytics.

The remainder of the paper is organised as follows. Section 2 describes the system architecture, detection model, dataset and augmentation, filtering and confirmation algorithms, scoring metric, evaluation protocol, and web platform. Section 3 reports the deployed system and its quantitative evaluation. Section 4 discusses the approach and its limitations, and Section 5 concludes.

1.1 Related Work

Computer vision is now a mature tool for analysing athletic performance, and object detectors have been applied to localise players, equipment, and projectiles in both broadcast and training footage [4], [5]. In the specific domain of marksmanship, three families of solutions coexist. Manual and semi-automatic scoring relies on a human judge, or on overlaying a scoring template on a photograph of the target; it is inexpensive but slow and subjective. Sensor-based electronic targets infer the point of impact from acoustic, optical, or pressure transducers and achieve high precision, but they are expensive, mechanically complex, and difficult to recalibrate. Vision-based approaches, by contrast, treat hit localisation as an object-detection or image-difference problem and require only a camera, which lowers cost and increases portability. Early vision methods depended on classical image processing, such as background subtraction, thresholding, and contour

analysis, and were sensitive to lighting and target appearance. The advent of convolutional single-stage detectors such as YOLO [2], [3] shifted the field toward learned, appearance-robust detection.

Several recent studies address automatic scoring directly. Ali et al. [6] compared seven detectors on images of shooting cards and reported a best result of $\text{mAP}@50 = 0.967$ for YOLOv8m and 0.965 for YOLOv8s, with inference times of 5.7 ms and 2.3 ms respectively; the dataset was released publicly. Their evaluation, however, is confined to static, scanned cards photographed under controlled conditions, in which the target is planar, frontal, and free of motion, so the reported accuracy characterises a substantially easier setting than continuous live video. Lin et al. [7] moved the task into the video domain and observed that the dominant error source there is not the detector itself but the displacement of the target sheet and the camera between frames, which causes previously scored holes to be re-reported as new ones; they addressed this with the STC-YOLO architecture, introducing an SPD-Conv module and the SIOU loss to improve small-object detection. Hah et al. [8] demonstrated a bullet-hole detection system for army shooting examinations built on a Raspberry Pi 4B with a telephoto lens and IoT reporting, confirming that the low-cost hardware assumption underlying vision-based scoring is realistic in field conditions.

A common limitation of these studies is that accuracy is reported at the level of the individual frame or image, whereas the practical quantity of interest in a live system is the probability that a physical shot is registered exactly once and assigned the correct class. The two are not equivalent: a detector with modest per-frame recall can still confirm shots reliably if the confirmation logic aggregates evidence across frames, and conversely a detector with high per-frame recall can over-report shots if duplicates are not suppressed. The problem is structurally identical to track initiation in radar signal processing, where a tentative track is promoted to a confirmed track only after M detections within N consecutive scans, the classical M -of- N rule [9], [10]. In computer vision, ByteTrack [11] makes a related observation from the opposite direction, showing that low-confidence detections should not simply be discarded because association with an existing tracklet can distinguish genuine objects from background. ShootVision applies this reasoning to marksmanship scoring: it pairs a lightweight learned detector with an explicit spatial-temporal confirmation rule and reports both the per-frame detector

metrics and the resulting shot-level confirmation probability, an evaluation that, to the best of the authors' knowledge, has not previously been presented for vision-based scoring systems.

2. Materials and methods

2.1 System Architecture

ShootVision is organised as a layered pipeline that transforms raw video into scored shot events and interactive analytics. Video frames are captured through OpenCV and preprocessed to the detector

input size; the YOLOv8n model produces per-frame detections; the detection-filtering and shot-confirmation layers remove noise and duplicates; the scoring layer assigns a numerical accuracy value; and a Django backend persists results and streams annotated video to a web frontend that offers four operating modes and analytics. Separating detection from the downstream filtering, confirmation, and scoring stages keeps each component independently testable and replaceable, and isolates the noisy per-frame detector output from the higher-level shot logic. Figure 1 summarises this architecture.

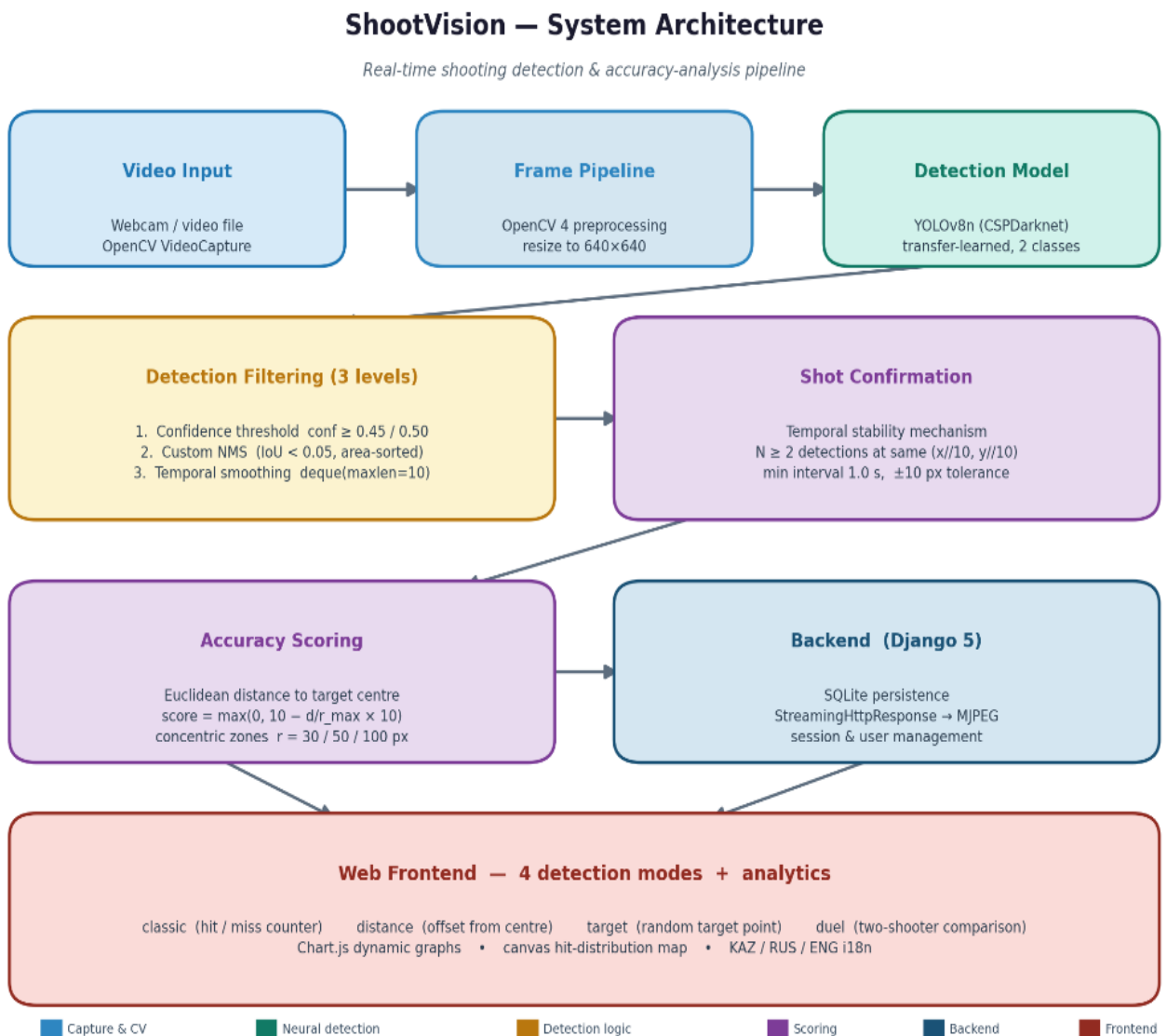


Figure 1. Layered architecture of the ShootVision system, from video capture to web-based analytics

2.2 Detection Model

The detection component is built on YOLOv8n [12], the smallest (nano) variant of the YOLOv8 single-stage object detector, whose backbone follows the cross-stage-partial design of CSPDarknet [13]. The nano configuration was selected to keep inference fast enough for real-time operation on consumer hardware. YOLOv8 adopts an anchor-free, decoupled detection head that predicts object centres directly, which simplifies the output and tends to improve small-object localisation; combined with the compact nano backbone, this makes the model well suited to deployment on edge devices such as the single-board computer used in the prototype. Rather than training from scratch, the model is initialised from publicly available pretrained weights (yolov8n.pt) and fine-tuned on a custom dataset through transfer learning [14], a strategy that markedly reduces the amount of task-specific data required and accelerates convergence; the pretrained weights themselves were learned on the large-scale COCO detection benchmark [15]. The detection task is framed as binary classification of detected objects into two classes: hit (class 0) and miss (class 1). The two classes are distinguished by the position of a perforation relative to the scoring area rather than by its appearance: a perforation lying within the printed scoring rings is annotated as hit, whereas a perforation on the surrounding backing, outside the outermost ring, is annotated as miss. The perforations themselves are visually almost identical, so the class decision rests largely on local context. This is stated explicitly because it governs how the class-confusion results of Section 3.4 should be interpreted, and because it introduces a dependence on target appearance that is revisited in Section 4. Training used 100 epochs, an input image size of 640 pixels, a batch size of 8, automatic optimiser selection (optimizer='auto'), and early stopping with a patience of 20 epochs. In addition to the offline augmentation described in Section 2.3, the Ultralytics training loop applies its default online augmentation, comprising four-image mosaic composition, random scaling and translation, horizontal flipping, and HSV colour jitter; a representative training batch is shown in Figure 3.

2.3 Dataset and Augmentation

The model was trained on a shooting-target dataset obtained from Roboflow [16] (shooting2555555/dataset/v3), released under the CC BY 4.0 licence. Because the volume of annotated shooting imagery is limited, data augmentation was used to improve generalisation and reduce overfitting, a technique widely shown to benefit deep-learning models trained on small datasets [17]. A custom augmentation pipeline (augment_data.py) was implemented in which each source image yields five augmented variants. Together with the original image, this gives six images per source frame, so the augmented training set contains six times as many images as the source set, as expressed in Equation (1):

$$N_{\text{aug}} = (1 + 5) \times N_{\text{src}} = 6 N_{\text{src}} \quad (1)$$

where N_{src} is the number of source images and N_{aug} the number of images after augmentation. The wording is stated explicitly here because the five generated variants and the resulting sixfold set size are distinct quantities. Crucially, bounding-box coordinates are recomputed analytically through the rotation matrix so that annotations remain geometrically valid after transformation. The transformations are summarised in Table 1.

Table 1. Custom augmentation transformations and their parameters

Transformation	Parameters
Horizontal flip	Applied with 50 % probability
Rotation	$\pm 15^\circ$, with analytical recomputation of YOLO coordinates
Brightness	Scaling factor 0.7-1.3
Contrast	Alpha (gain) 0.7-1.3
Net effect	5 augmented variants per source image; augmented set = $6 \times$ source set

The composition of the annotated set is summarised in Figure 2, which reports the class distribution, the spatial distribution of annotation centres, and the distribution of bounding-box dimensions. The corresponding figures are collected in Table 2.

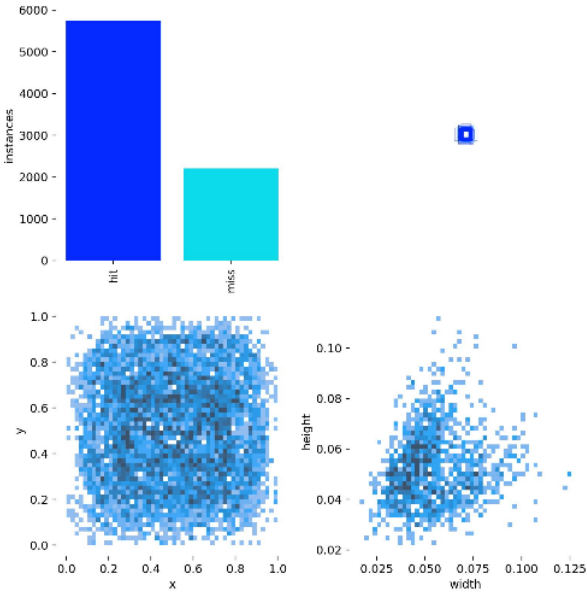


Figure 2. Composition of the annotated dataset: class distribution (top left), overlaid bounding boxes (top right), spatial distribution of annotation centres (bottom left), and joint distribution of normalised box width and height (bottom right)

Table 2. Composition of the annotated dataset and scale of the annotated objects

Property	Value
Annotated instances, class hit	5 742 (72.3 %)
Annotated instances, class miss	2 202 (27.7 %)
Class imbalance ratio (hit : miss)	2.61 : 1
Modal normalised bounding-box width	≈ 0.05
Modal normalised bounding-box height	≈ 0.05
Corresponding object size at 640 px input	$\approx 32 \times 32$ px
Object area as a fraction of the frame	≈ 0.25 %
Spatial distribution of annotation centres	Approximately uniform over the frame

Two properties of the set bear directly on the results of Section 3. First, the classes are imbalanced by a factor of 2.61 in favour of hit, which is expected because most shots in the source imagery fall inside the scoring rings; the consequence is a modest advantage in recall for the majority class, visible in Table 7. Second, and more consequentially, the annotated objects are extremely small. The modal normalised box side is approximately 0.05, which at the 640-pixel training resolution corresponds to an ob-

ject of roughly 32×32 pixels occupying about 0.25 % of the frame area, placing the task at the boundary of the small-object regime as defined in the COCO evaluation protocol [15]. The spatial distribution of annotation centres is approximately uniform, so no positional prior is available to the detector, and the joint width-height distribution shows that box dimensions are correlated with each other but not with position in the frame. The implications of this scale for the reported metrics are analysed in Section 3.3.

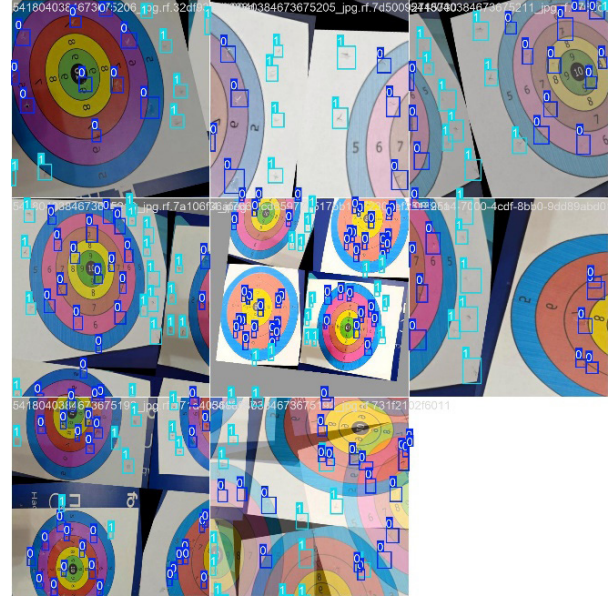


Figure 3. Representative training batch, showing four-image mosaic composition, geometric augmentation, and hit (class 0) and miss (class 1) annotations

2.4 Detection-Filtering Algorithm

A live video stream produces many unreliable detections; ShootVision therefore applies a three-level false-positive suppression scheme (`yolo_detector.py`). First, confidence thresholding retains detections only above a confidence threshold of 0.45 in classic mode and 0.50 in the other modes. Second, a custom non-maximum suppression stage [18] sorts candidate boxes by bounding-box area and filters them using an intersection-over-union criterion ($\text{IoU} < 0.05$), with hit and miss detections processed separately to avoid cross-class suppression. Third, temporal smoothing stabilises per-frame counters through a moving average computed over a sliding window of the last 10 frames, implemented as a fixed-length deque ($\text{maxlen} = 10$). The choice of operating threshold is examined quantitatively in Section 3.2.

2.5 Shot-Confirmation Algorithm and Its Probabilistic Model

Beyond per-frame filtering, ShootVision applies a temporal-stability mechanism that decides when a stable detection should be registered as a genuine shot. Each detection position is quantised to rounded coordinates ($x // 10, y // 10$), which encodes position compactly and enables fast hash-based matching across frames; two detections are treated as spatially consistent when they fall in the same quantised cell, corresponding to a tolerance of approximately ± 10 pixels. A shot is confirmed when at least $k = 2$ spatially consistent detections occur in consecutive frames. A minimum interval of 1.0 second between successive shots (`min_shot_interval`) prevents a single physical event from being counted multiple times.

This rule is a special case of the M-of-N confirmation logic used for track initiation in radar processing [9], [10], in which a tentative track is promoted only after M detections are associated with it within N consecutive looks. Formalising the rule makes it possible to predict system-level behaviour from the measured per-frame detector performance. Let p denote the probability that the detector localises a given mark in a single frame, that is, the per-frame recall, and assume for analysis that detections in different frames are conditionally independent. Under the generalised rule requiring M detections in the same spatial cell within a window of N frames, the probability of confirming a genuine shot is given by Equation (2):

$$P_{\text{conf}} = \sum_{k=M}^N C(N, k) \cdot p^k \cdot (1-p)^{N-k} \quad (2)$$

where $C(N, k)$ is the binomial coefficient. For the stricter rule actually implemented, which requires k detections in consecutive frames rather than anywhere in the window, the probability is obtained from the recursion in Equation (3):

$$\begin{aligned} f(i, r) &= p \cdot f(i+1, r+1) + (1-p) \cdot f(i+1, 0), \\ f(i, r \geq k) &= 1, f(N, r < k) = 0 \end{aligned} \quad (3)$$

where i indexes the frame within the window and r is the length of the current run of consecutive detections, so that $P_{\text{conf}} = f(0, 0)$. The corresponding false-confirmation probability is obtained from the same expressions by substituting q , the per-frame probability that a spurious detection appears in one particular spatial cell, for p . Both quantities are evaluated against the measured detector performance in Section 3.5.

The independence assumption in Equations (2) and (3) is an idealisation. In practice, detections are

correlated across frames, because a mark that is difficult to detect because of illumination, viewing angle, or occlusion tends to be difficult in every frame of the sequence. The resulting values are therefore upper bounds on the achievable confirmation probability, and they are presented as such.

2.6 Accuracy-Scoring Metric

Once a hit is confirmed, its accuracy is quantified geometrically. The Euclidean distance between the hit location and the target centre is computed as defined in Equation (4):

$$\text{distance} = \sqrt{[(x_{\text{hit}} - x_{\text{centre}})^2 + (y_{\text{hit}} - y_{\text{centre}})^2]} \quad (4)$$

This distance is mapped to a ten-point score using a linear penalty bounded below by zero, where r_{max} denotes the outermost (maximum) scoring radius, as defined in Equation (5):

$$\text{score} = \max(0, 10 - (\text{distance} / r_{\text{max}}) \times 10) \quad (5)$$

Scoring is visualised as concentric circles with radii of 30, 50, and 100 pixels, which act as analogues of the scoring rings on a physical target: hits nearer the centre receive higher scores, and hits beyond the outermost radius receive a score of zero. Because the penalty is linear in distance, a hit half-way to the outermost radius receives roughly half of the maximum score, which yields an intuitive, continuous accuracy signal rather than a coarse ring count and makes small improvements in precision visible. It should be emphasised that the radii are expressed in pixels rather than in physical units; the consequences of this choice are discussed in Section 4.

2.7 Evaluation Protocol

Detector quality is assessed on the held-out validation split of the augmented dataset using the standard detector log (`results.csv`) and a plotting utility (`plot_metrics.py`). The reported metrics are mean average precision at IoU 0.50 (`mAP@50`) and averaged over IoU 0.50-0.95 (`mAP@50-95`), following standard object-detection evaluation practice [15], [19], together with precision, recall, and the F1 score. The F1 score is computed as the harmonic mean $2PR / (P + R)$, because the training log does not contain a dedicated F1 column. Precision-confidence and F1-confidence curves are used to select the operating threshold, and the confusion matrix, reported both in absolute counts and column-normalised form, is used to separate two qualitatively different failure modes: localisation

error, in which an annotated object is not detected at all or a background region is reported as an object, and class-confusion error, in which an object is detected but assigned the wrong class. This distinction is central to the present application, because a class-confusion error changes the score awarded to the shooter, whereas a localisation error in a video stream primarily changes the latency with which a shot is confirmed. Confusion matrices are computed at the default operating point of confidence 0.25 and IoU 0.45.

The confirmation logic is evaluated separately from the detector, by substituting the measured per-frame recall into Equations (2) and (3) and computing the resulting shot-level confirmation probability and latency for a range of rule parameters. This analytical evaluation is reported in Section 3.5; its limitations, and the controlled experiment required to complement it, are stated in Section 4.

2.8 Web Platform and Operating Modes

The system is served by a Django 5 backend [20] with an SQLite database. Annotated video is delivered to the browser as a Motion-JPEG stream through StreamingHttpResponse, with frames produced by OpenCV [21]. The frontend provides interactive analytics using Chart.js for dynamic charts and an HTML canvas for a spatial map of the hit distribution. The interface is available in three languages (Kazakh, Russian, and English). Four detec-

tion modes, each with its own analytical purpose, are offered, as summarised in Table 3.

Table 3

The four operating modes of ShootVision and their analytical functions

Mode	Purpose
classic	Real-time counting of hits and misses on the target.
distance	Measurement of the Euclidean offset of each hit from the centre.
target	Aiming at a randomly generated target point.
duel	Comparative accuracy analysis of two shooters.

3. Results

3.1 Hardware Prototype Stand

In addition to the software pipeline, a functional physical prototype of an automated shooting-target stand was designed and built to host the camera and demonstrate end-to-end operation. The stand is constructed from an aluminium-profile frame, which provides structural stability and modularity while remaining lightweight and easy to reconfigure. The overall geometry and principal dimensions of the frame are shown in Figure 4, and a three-dimensional model of the assembled stand with its mounted target board is shown in Figure 5.

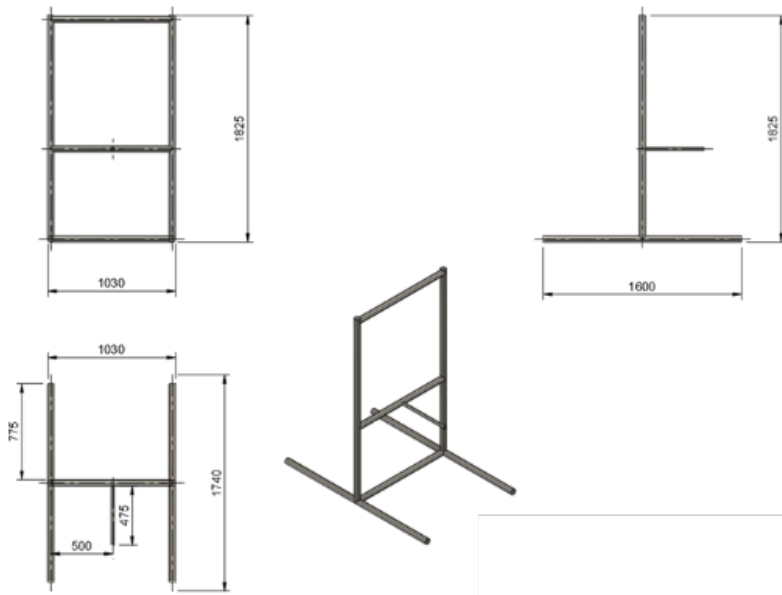


Figure 4. Orthographic engineering drawing of the aluminium-profile target stand with principal dimensions (mm)

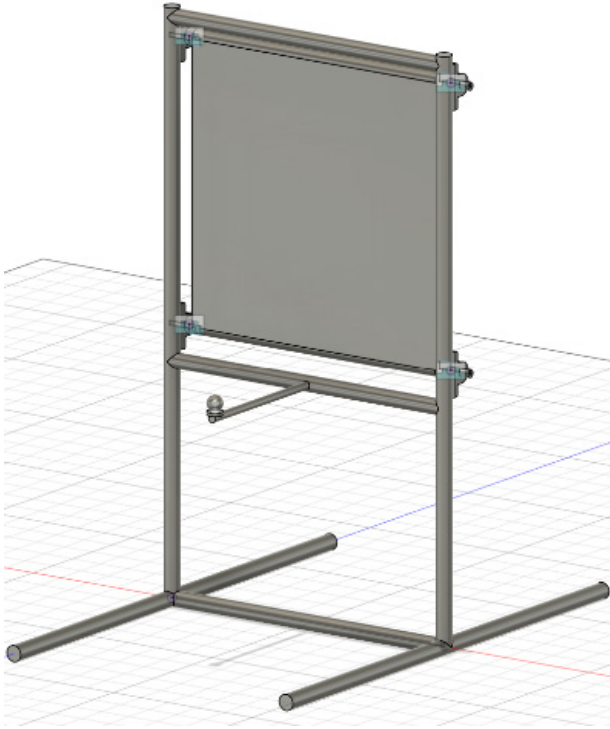


Figure 5. Three-dimensional CAD model of the assembled target stand with mounted target board

The stand integrates NEMA 23 stepper-motor actuators, a Raspberry Pi single-board computer, and a camera-based vision module. The mechanical subsystem enables automatic target switching after each shooting cycle, improving throughput and reducing manual intervention. The assembled prototype in operation, with a printed target mounted on the board, is shown in Figure 6.

Operation of the automated stand is controlled remotely through a user interface (a remote controller) and comprises two principal functions, target switching and shooting-result evaluation, organised as the following cycle. On initialisation, the system powers on, initialises all components, and enters standby mode. For target switching, the user presses the “Next Target” button; the stepper motor rotates the target; a position sensor detects the correct position; and the motor stops. For result evaluation, the user presses the “Start Evaluation” button; the camera captures the target image; the detection algorithm localises hits and computes the score; and the result is saved and displayed. The system then repeats the cycle in response to subsequent user commands.



Figure 6. Assembled hardware prototype of the ShootVision stand with a mounted paper target

The embedded camera captures the target after each shooting cycle, and the detection pipeline of Section 2 analyses the image and computes the score automatically, enabling performance evaluation without human involvement. Overall, the prototype demonstrates a working integration of mechanical, electronic, and software components, and provides a platform for automated shooting training and scoring.

3.2 Detection Performance

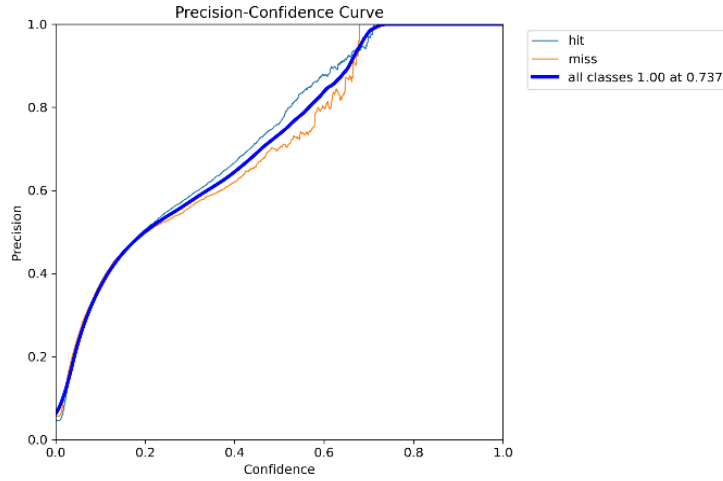
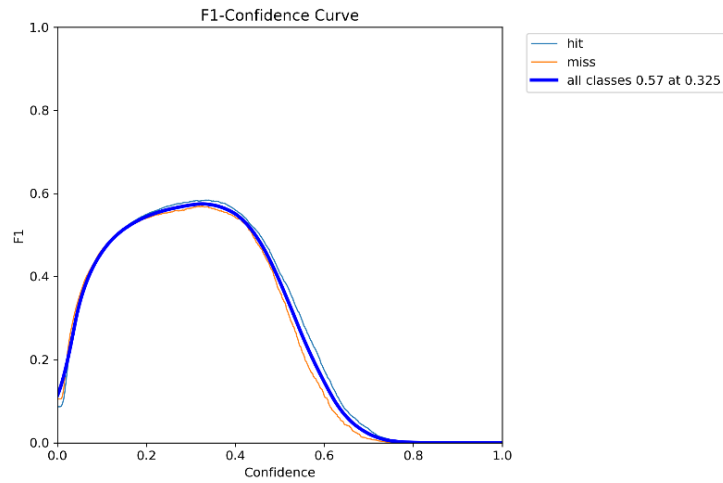
The detector was fine-tuned for up to 100 epochs with early stopping (patience = 20), so training halts automatically once validation performance ceases to improve, which mitigates overfitting on the augmented dataset. The resulting performance on the validation set is reported in Table 4.

The F1 score is computed as $2PR / (P + R)$; the training log contains no dedicated F1 column. Precision and recall are reported at the operating point of the trained model, whereas the confusion matrices below are computed at confidence 0.25, which accounts for small differences in the third decimal place.

Table 4. Detection-quality metrics on the validation set (YOLOv8n, image size 640, IoU 0.45)

Metric	Value
mAP@50	0.469
mAP@50-95	0.194
Precision (macro)	0.595
Recall (macro)	0.579
F1 score	0.587
Maximum F1 / corresponding confidence	0.57 at confidence 0.325
Confidence at which precision reaches 1.00	0.737

The precision-confidence and F1-confidence curves are shown in Figures 7 and 8. The F1 curve peaks at 0.57 at a confidence of 0.325, which identifies the threshold that balances precision and recall most favourably; the operating thresholds of 0.45 and 0.50 used in the deployed modes therefore trade some recall for higher precision, a deliberate choice given that the confirmation logic of Section 2.5 can recover missed detections across frames but cannot remove a false shot once it has been confirmed. The precision curve reaches 1.00 at a confidence of 0.737, showing that the false-positive rate is controllable through the threshold rather than being an irreducible property of the model.

**Figure 7.** Precision-confidence curve for the hit and miss classes and for all classes combined**Figure 8.** F1-confidence curve for the hit and miss classes and for all classes combined

3.3 Object Scale and Localisation Error

The gap between $\text{mAP}@50=0.469$ and $\text{mAP}@50-95=0.194$ is large, and the object scale reported in Section 2.3 accounts for most of it. Average precision averaged over IoU thresholds from 0.50 to 0.95 rewards localisation that is accurate to a fixed fraction of the object size, so for a 32-pixel object the tolerance in absolute terms is a small number of pixels. Table 5 reports the intersection over union between a square box of side a and an identical box displaced diagonally by d pixels, for the object size present in this dataset and for two larger sizes given for reference.

Table 5. Intersection over union between a square annotation and an identical box displaced diagonally by d pixels, as a function of object size

Displacement d (px)	$a = 32$ px (this dataset)	$a = 64$ px	$a = 128$ px
1	0.884	0.940	0.969
2	0.784	0.884	0.940
3	0.697	0.832	0.911
4	0.620	0.784	0.884
6	0.493	0.697	0.832
8	0.391	0.620	0.784

For the 32-pixel objects of this dataset, an IoU of 0.50 is lost once the predicted box is displaced by about 6 pixels, an IoU of 0.75 once it is displaced by about 2.4 pixels, and an IoU of 0.90 once it is

displaced by less than one pixel. The same displacements leave a 128-pixel object at IoU 0.83, 0.94, and 0.97 respectively. Since $\text{mAP}@50-95$ averages ten thresholds of which the upper half require sub-three-pixel accuracy, the metric is structurally bounded for objects of this size regardless of how well the detector performs semantically. The observed ratio $\text{mAP}@50-95 / \text{mAP}@50 = 0.414$ is consistent with this account: for objects at the small-object boundary a ratio in the range 0.40-0.45 is typical, whereas detectors operating on larger objects commonly reach 0.65 or above. The figure should therefore be read as a statement about object scale rather than as evidence of a defective model, and comparisons with published results obtained on larger targets are not meaningful unless object scale is matched.

This also explains part of the localisation error quantified in the following section. A perforation of 32 pixels differs from surrounding paper texture, scanning artefacts, and shadow by a small number of pixel intensities, so the boundary between an annotated object and background is intrinsically less well defined than for the larger objects on which the pretrained weights were learned.

3.4 Decomposition of the Confusion Matrix

The absolute and column-normalised confusion matrices are shown in Figures 9 and 10. Their decomposition, following the protocol of Section 2.7, is reported in Table 6 and is the most informative result of the evaluation.

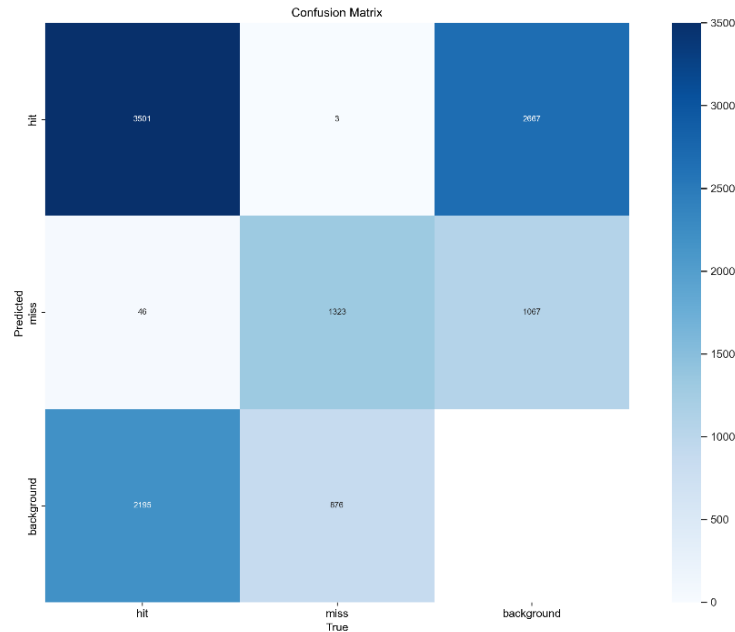


Figure 9. Confusion matrix in absolute counts (rows: predicted class; columns: true class)

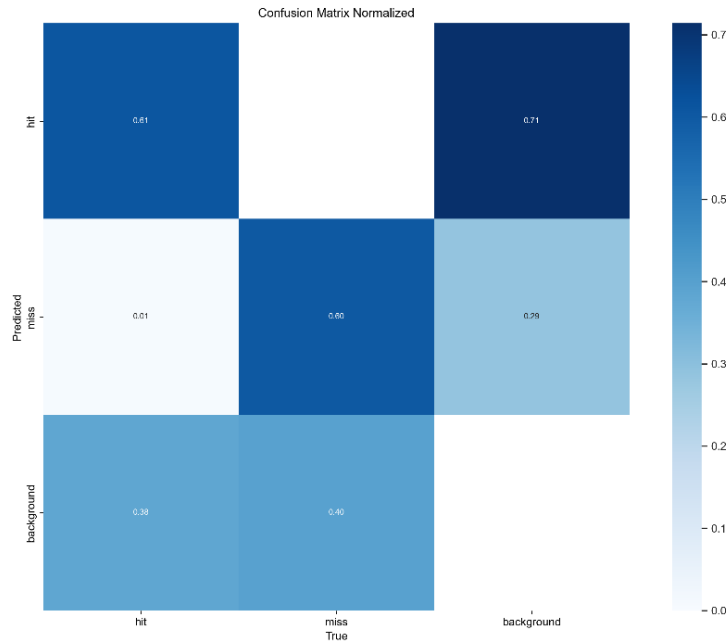


Figure 10. Column-normalised confusion matrix

Table 6. Decomposition of the confusion matrix into localisation and class-confusion error

Quantity	Value
Annotated objects in the validation set	7 944 (5 742 hit, 2 202 miss)
Objects not localised (assigned to background)	3 071 (38.7 % of all objects)
Background regions reported as objects	3 734
Objects localised by the detector	4 873
Localised objects assigned the correct class	4 824 (99.0 %)
hit reported as miss	46 (1.30 % of localised hits)
miss reported as hit	3 (0.23 % of localised misses)

The decomposition shows that the moderate aggregate metrics of Table 4 arise almost entirely from localisation rather than from class confusion. Once the detector has localised a mark, it assigns the correct class in 99.0 % of cases, and the asymmetry is favourable for the application: only 3 of 1 326 localised misses are reported as hits, so the system almost never credits a shooter with a hit that did not occur. The remaining error is concentrated in the

38.7 % of annotated objects that are not localised at all and in the 3 734 background regions reported as objects. This distinction matters because the two failure modes have different consequences in a live system. A class-confusion error changes the score awarded and cannot be recovered downstream; a missed localisation, by contrast, merely delays confirmation, because the same physical mark is presented to the detector in every subsequent frame. Per-class metrics derived from the same matrix are given in Table 7.

Table 7. Per-class precision, recall, and F1 derived from the confusion matrix

Class	Precision	Recall	F1
hit	0.567	0.610	0.588
miss	0.543	0.601	0.571

3.5 Analysis of the Shot-Confirmation Logic

The per-frame recall measured in Section 3.4 is 0.610 for the hit class and 0.601 for the miss class. Substituting these values into Equations (2) and (3) gives the confirmation probabilities and latencies reported in Table 8, computed for a capture rate of 30 frames per second.

Table 8. Probability of confirming a genuine shot as a function of the confirmation rule, at per-frame recall $p = 0.610$ (hit) and $p = 0.601$ (miss)

Rule	Window N (frames)	Latency (s)	P(confirm), hit	P(confirm), miss
2 consecutive (implemented)	5	0.17	75.3 %	74.1 %
2 consecutive (implemented)	10	0.33	95.2 %	94.7 %
2 consecutive (implemented)	15	0.50	99.1 %	98.9 %
3 of N (generalised)	8	0.27	95.6 %	95.1 %
3 of N (generalised)	10	0.33	99.0 %	98.8 %
4 of N (generalised)	10	0.33	95.2 %	94.6 %

Two observations follow. First, the mechanism accomplishes what the aggregate detector metrics do not: with a per-frame recall of only 0.61, a shot is confirmed with probability 0.95 within one third of a second under the currently implemented rule, and with probability 0.99 within half a second. The purpose of the confirmation stage is precisely to convert a detector that is unreliable on any single frame into a system that is reliable on a physical event, and the figures quantify how much is gained. Second, relaxing the requirement from two strictly consecutive detections to any three detections within a ten-frame window raises the confirmation probability from 95.2 % to 99.0 % at identical latency, because the relaxed rule tolerates an isolated missed frame in the middle of a run. This suggests a concrete refinement of the implemented logic at no cost in responsiveness.

The same expressions bound the false-confirmation rate. If a spurious detection appears in one particular spatial cell with per-frame probability q , the probability that the implemented rule confirms a non-existent shot within a ten-frame window is 0.09 % for $q = 0.01$, 0.35 % for $q = 0.02$, 2.1 % for $q = 0.05$, and 8.0 % for $q = 0.10$. Because a false confirmation requires the spurious detection to recur in the same quantised cell, and because raising the confidence threshold towards 0.737 drives precision to 1.00, the spatial-consistency requirement and the threshold together provide two independent means of keeping q small. The values in Table 8 are upper bounds obtained under the independence assumption stated in Section 2.5, and Section 4 specifies the controlled experiment needed to measure the corresponding quantities directly.

3.6 Qualitative Detection Results

Figures 11 and 12 show the same validation batch with, respectively, the ground-truth annotations and the predictions of the trained detector at the default evaluation threshold. The comparison makes the quantitative results of Sections 3.2 to 3.4 visible directly and illustrates each of the failure modes they describe.

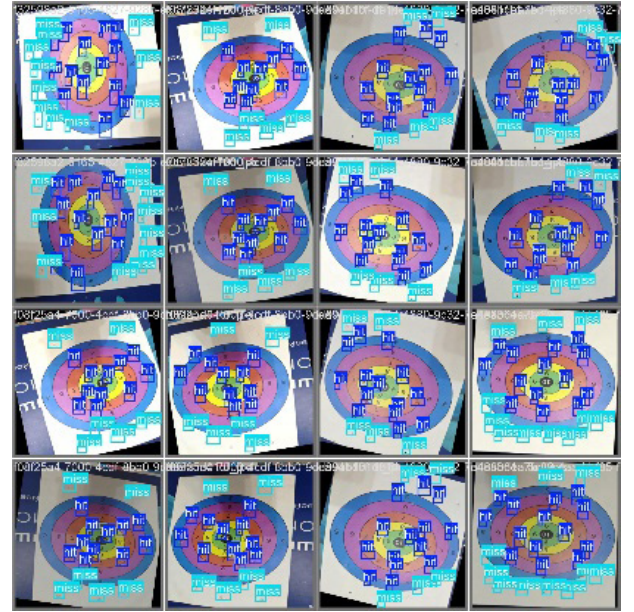


Figure 11. Ground-truth annotations on a validation batch. Perforations within the scoring rings are annotated hit; perforations on the surrounding backing are annotated miss

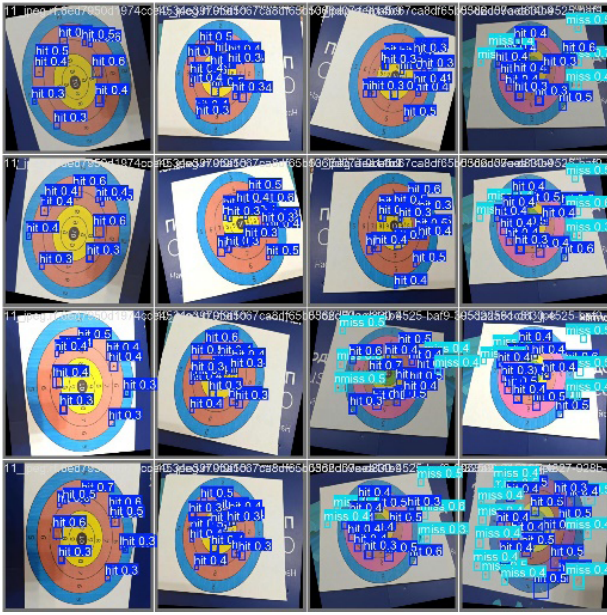


Figure 12. Predictions of the trained detector on the same validation batch, with class labels and confidence values

Three observations follow. First, class assignment is visibly reliable: predictions inside the scoring rings are labelled hit and those on the surrounding backing are labelled miss, with no systematic exchange between the two, which is the qualitative counterpart of the 99.0 % conditional

class accuracy in Table 6. Second, the predictions are sparser than the annotations, particularly for perforations on the plain backing and near the frame edges, which is where the 38.7 % of unlocalised objects is concentrated; marks on unstructured light backgrounds have less local contrast than marks on the printed rings. Third, the reported confidence values lie predominantly between 0.3 and 0.7 and rarely approach 0.9, consistent with the precision-confidence curve of Figure 7, which reaches unit precision only at 0.737. Overlapping boxes on closely spaced perforations are also visible and are the condition that the area-aware non-maximum suppression of Section 2.4 is designed to resolve.

These are validation images drawn from the same dataset as the training data, so they demonstrate that the pipeline produces coherent annotated output but do not evidence generalisation to unfamiliar targets; that limitation is stated in Section 4.

3.7 Deployed Web Interface

The fully functional system was deployed and runs as a Django web application. Figure 13 shows the public landing page, which summarises the technology stack (YOLOv8, OpenCV, Django), advertises the four operating modes, real-time 30-fps processing, and the trilingual interface, and provides entry points for authentication and registration.

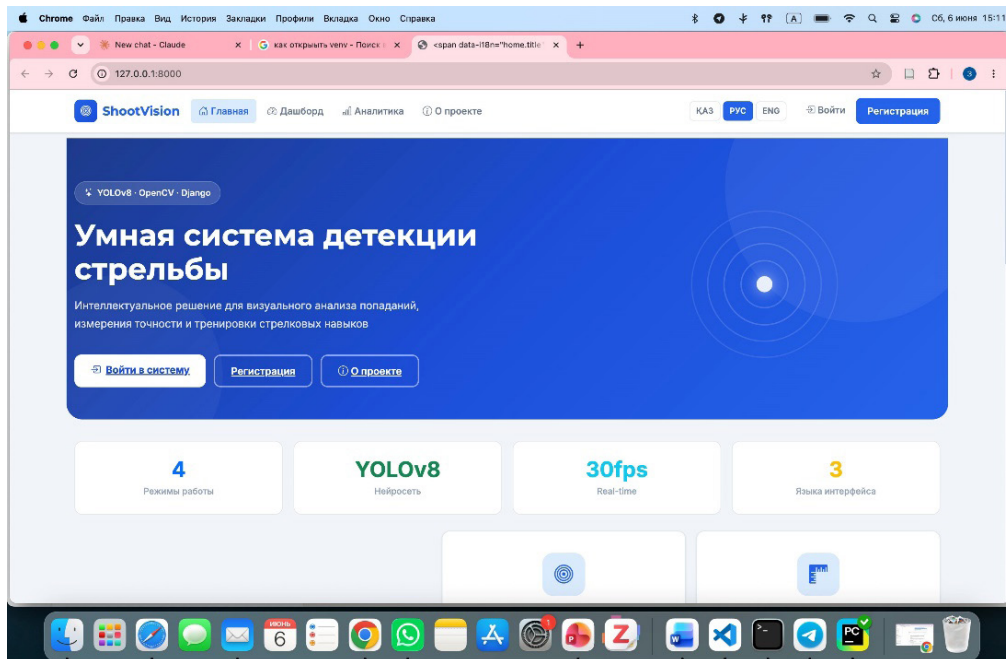


Figure 13. ShootVision landing page, summarising the system’s capabilities and technology stack

After authentication, the user reaches the dashboard shown in Figure 14, where the operating mode is selected. The dashboard presents each mode as a card with a short description and feature list, alongside a user panel that surfaces aggregate statistics (hits, misses, and accuracy) and links to

the analytics view. The basic classic mode offers an automatic hit counter, accuracy statistics, and visual detection boxes, while the advanced distance mode adds lines from the centre to each hit; minimum, maximum, and mean distance; and a history of recent hits.

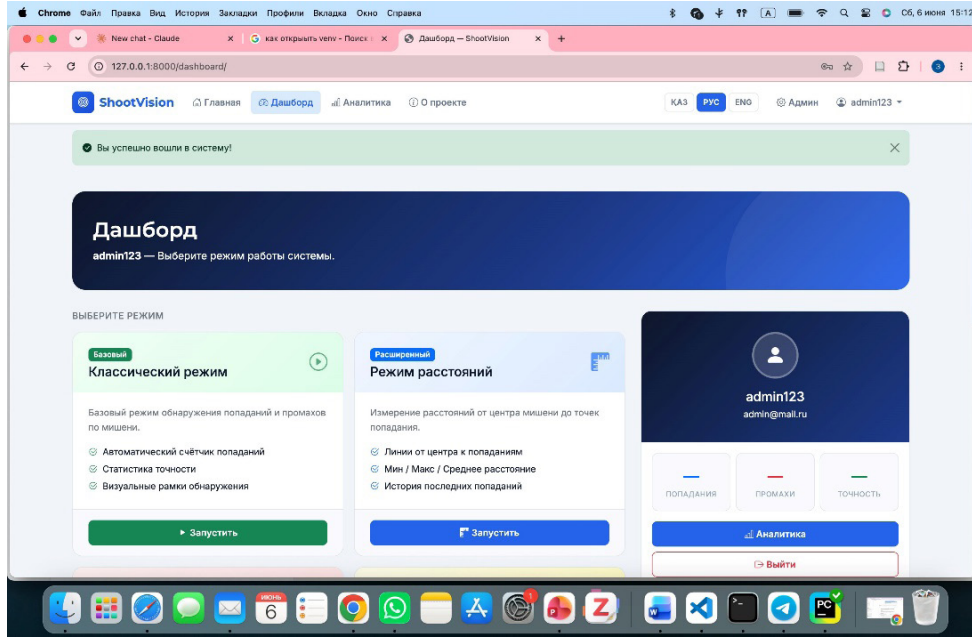


Figure 14. Authenticated user dashboard for selecting an operating mode and viewing aggregate statistics

In live operation the full pipeline sustained a throughput of approximately 30 frames per second, which is sufficient for interactive, real-time feedback to a shooter. Qualitatively, the combination of confidence thresholding, area-aware NMS, and temporal smoothing markedly reduced the flickering and duplicate detections typical of raw per-frame output, and the confirmation mechanism registered each physical shot once.

3.8 Analytics and Modes

Across all modes, results are accumulated and visualised through dynamic Chart.js graphs and a canvas-based map of the spatial hit distribution, giving the shooter immediate, quantitative feedback. The spatial map is particularly informative for diagnosing systematic error: rather than reporting only an aggregate score, it reveals how shots cluster relative to the target centre, so that a consistent grouping in one region, for example a tight cluster low and to the left, exposes a directional aiming bias that a single numeric total would conceal. The accom-

panying time-series view plots score and grouping across a session, allowing a shooter or instructor to track whether accuracy is improving, plateauing, or degrading with fatigue over successive shots. The target mode trains aiming against randomly generated points, while the duel mode supports a direct, side-by-side accuracy comparison between two shooters, illustrating that the same detection-and-scoring core can be reused across distinct training scenarios.

4. Discussion

The results indicate that a lightweight, transfer-learned object detector, when combined with carefully designed temporal logic, can provide an affordable and accessible alternative to dedicated electronic scoring targets. The principal engineering insight is that a raw detector is insufficient on its own: the three-level filtering scheme and the temporal-stability confirmation mechanism are what convert noisy per-frame detections into reli-

able, de-duplicated shot events, and they account for much of the system's practical robustness. The cost argument is concrete: because the pipeline runs on commodity computing and an ordinary camera and is built entirely on an open-source software stack, the marginal hardware cost of a deployment is essentially that of a camera and a modest computer, which is a small fraction of the price of the dedicated electronic targets and acoustic scoring systems that are conventionally used for the same purpose. Delivering the system as a web application with multilingual support and multiple training modes further lowers the barrier to adoption, and the accompanying hardware stand demonstrates that the pipeline integrates cleanly with low-cost, off-the-shelf mechatronic components.

Several limitations should be acknowledged. First, the detector was trained on a single Roboflow dataset, so its generalisation to different target types, lighting conditions, calibres, and camera viewpoints has not been established and may require additional data. Second, the scoring metric and the concentric scoring zones are defined in pixel units relative to an assumed target centre; without geometric calibration, scores are not directly comparable across different camera distances or target sizes, and an off-axis camera can introduce perspective distortion. Third, the shot-confirmation parameters (the minimum of two consistent detections, the ± 10 -pixel tolerance, and the one-second interval) are fixed heuristics that trade sensitivity against false positives and may need tuning for very rapid firing or for small or distant targets. Fourth, the quantitative detection metrics depend on the specific training run and dataset split and should be reported per deployment. Finally, the present evaluation emphasises the functioning deployed system rather than a controlled benchmark against ground-truth shot positions or against commercial scoring hardware.

Future work could address these issues through multi-dataset training and domain randomisation for better generalisation, geometric calibration to express scores in physical units, adaptive or learned confirmation parameters, and a controlled accuracy study comparing ShootVision against reference scoring systems.

From a deployment standpoint, the real-time budget is dominated by detector inference rather than by the lightweight filtering, confirmation, and scoring stages, so end-to-end throughput can be tuned directly through the choice of YOLOv8 model size, trading detection latency for accuracy without altering the surrounding logic. Because the

entire pipeline, from the augmentation script and training configuration to the detection thresholds and confirmation parameters, is defined explicitly in software, a given configuration can be reproduced and audited, which is a practical advantage over closed sensor-based systems whose internal scoring decisions are not exposed to the user.

5. Conclusion

This paper presented ShootVision, a real-time, web-based system for shooting detection and marksmanship accuracy analysis. The system fine-tunes a lightweight YOLOv8n detector to classify hits and misses, expands its training data sixfold with a custom augmentation pipeline that analytically preserves bounding-box geometry, and stabilises the detector's output with a three-level filtering scheme and an original temporal-stability shot-confirmation mechanism. Hits are scored by their Euclidean distance to the target centre over concentric scoring zones, and the whole pipeline is delivered through a Django web platform that streams annotated video and offers four operating modes, trilingual support, and interactive analytics at approximately 30 frames per second. A physical prototype stand built from aluminium profile, stepper-motor actuators, and a Raspberry Pi demonstrates end-to-end operation. ShootVision shows that combining accessible object detection with purpose-built temporal logic yields a low-cost, practical tool for shooting analysis, and it provides a clear foundation for future work on generalisation, calibration, and rigorous quantitative benchmarking.

Acknowledgments

The authors express their gratitude to the S. Nurmagambetov Military Institute of Land Forces for providing the research infrastructure and experimental facilities used in this study. The authors also acknowledge the technical and methodological support provided by colleagues involved in the development and testing of the mechatronic components and experimental shooting setup.

Funding

This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan, grant number AP23490677.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, M.S.Z. and M.N.S.; Methodology, M.S.Z. and Z.B.A.; Software, Z.B.A.; Validation, M.S.Z., Z.B.A.; Formal Analysis, M.N.S.; Investigation, Z.B.A.; Resources, Z.B.A.; Data Cu-

ration, Z.B.A.; Writing – Original Draft Preparation, Z.B.A.; Writing – Review & Editing, M.S.Z. and M.N.S.; Visualization, Z.B.A.; Supervision, M.S.Z. and M.N.S.; Project Administration, M.N.S.; Funding Acquisition, M.S.Z.

References

1. Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, 2023.
2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.
3. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, arXiv:2004.10934.
4. M. Burić, M. Pobar, and M. Ivašić-Kos, "Object detection in sports videos," in *Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, Opatija, Croatia, 2018, pp. 200–206, doi: 10.23919/MIPRO.2018.8400189.
5. K. Host and M. Ivašić-Kos, "An overview of human action recognition in sports based on computer vision," *Heliyon*, vol. 8, no. 6, 2022, Art. no. e09633, doi: 10.1016/j.heliyon.2022.e09633.
6. M. Butt, N. Glas, J. Monsuur, R. Stoop, and A. de Keijzer, "Application of YOLOv8 and Detectron2 for bullet hole detection and score calculation from shooting cards," *AI*, vol. 5, no. 1, pp. 72–90, 2024, doi: 10.3390/ai5010005.
7. Z. Lin, Z. Zhang, and X. Ding, "A novel automatic target reporting method with STC-YOLO for videos," in *Eighth International Conference on Computer Graphics and Virtuality (ICCGV 2025)*, SPIE, Apr. 2025, p. 194. doi: 10.1117/12.3060446.
8. Jilian. H. W. Yin, I. M. Tahir, A. A. Al-Talib, and O. M. Magzoub, "Detection of Bullet Holes for Target Board in Malaysia Military (ATM) Shooting Exam Application," *Proceedings of International Conference on Artificial Life and Robotics*, vol. 30, pp. 423–427, Feb. 2025, doi: 10.5954/ICAROB.2025.OS14-3.
9. M. Sedehi, P. Lombardo, and A. Farina, "A modified M/N logic for track initiation of low observable targets using amplitude information," in *Proc. Int. Radar Symp. (IRS)*, Krakow, Poland, 2006, pp. 391–394, doi: 10.1109/IRS.2006.4338080.
10. Y. Zhang, A. Liu, C. Liu, B. Ai, and X. Zhang, "A track initiation algorithm using residual threshold for shore-based radar in heavy clutter environments," *J. Mar. Sci. Eng.*, vol. 8, no. 8, 2020, Art. no. 614, doi: 10.3390/jmse8080614.
11. Y. Zhang et al., "ByteTrack: Multi-object tracking by associating every detection box," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Tel Aviv, Israel, 2022, pp. 1–21.
12. G. Jocher, A. Chaurasia, and J. Qiu. *Ultralytics YOLOv8*. (2023). Accessed: Jun. 17, 2026. [Online]. Available: <https://github.com/ultralytics/ultralytics>
13. C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Seattle, WA, USA, 2020, pp. 390–391.
14. S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.
15. T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Zurich, Switzerland, 2014, pp. 740–755.
16. B. Dwyer, J. Nelson, and J. Solawetz. *Roboflow*, Version 1.0. (2022). Accessed: Jun. 17, 2026. [Online]. Available: <https://roboflow.com/>
17. C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, 2019.
18. A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Hong Kong, 2006, pp. 850–855.
19. R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Niterói, Brazil, 2020, pp. 237–242.
20. Django Software Foundation. "Django: The web framework for perfectionists with deadlines." 2024. Accessed: Jun. 19, 2026. [Online]. Available: <https://www.djangoproject.com/>
21. G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, pp. 120–125, 2000.

Information about the authors:

Mukhtar Zhassuzak – is a PhD student and lecturer at the Department of Computer Science, al-Farabi Kazakh National University (Almaty, Kazakhstan). His research interests include artificial intelligence and robotics. ORCID iD: <https://orcid.org/0000-0001-8164-8199>. zhassuzak.mukhtar@gmail.com

Zemfira Abdirazak – is an Executive Engineer at the S. Nurmagambetov Military Institute of Land Forces (Almaty, Kazakhstan). Her research interests include artificial intelligence and robotics. ORCID iD: <https://orcid.org/0009-0006-1857-0863>. abdirazakzemfira@gmail.com

Daniyar Dauletiya – holds an MSc in Computer Engineering and is Head of the Research and Innovation Laboratory (FabLab) at Astana IT University (Astana, Kazakhstan). ORCID iD: <https://orcid.org/0000-0002-3205-7453>; Scopus Author ID: 58613538200; Researcher ID: LDC-6573-2024. d.dauletiya@astanait.edu.kz

Maxatbek Satymbekov – holds a PhD and is Dean of the Graduate School of STEM Education at Shakarim University (Semey, Kazakhstan). His research interests include artificial intelligence and intelligent control systems. ORCID iD: <https://orcid.org/0000-0002-4621-6646>. m.n.satymbekov@gmail.com

Submission received: 29 June, 2026.

Revised: 10 September, 2026.

Accepted: 10 September, 2026.