International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 15, Issue 11 [November 2026] PP: 231-239

VISTA: Vision-Based Interaction for Smart Touchless Action with Safety-Aware Temporal Behavior Analysis

Asst. Prof. Sahana A M[1], Nimish Amruth B T[2], Roshan R[2]
Prajwal Kumar S[2], Vedant[2]

Assistant professor.Department of Computer Science and Engineering, APS College of Engineering, Bengaluru, INDIA
[2]Final-year B.Tech students.
Department of Computer Science and Engineering, APS College of Engineering, Bengaluru, INDIA
Corresponding Author: Roshan R.

ABSTRACT: *Touchless operation of a computer is valuable wherever hands are busy, gloved or must stay clean, yet most gesture interfaces either depend on depth sensors or ignore the safety information that a camera in the room already captures. This paper describes VISTA (Vision-Based Interaction for Smart Touchless Action), a modular prototype that runs entirely on a local machine and uses one ordinary webcam for two tasks: controlling the computer through recognized hand gestures in real time, and screening body movement over time so that it can be labeled NORMAL or SUSPICIOUS. The pipeline joins OpenCV preprocessing, MediaPipe hand landmarks, YOLO pose detection with multi-object tracking, a PyTorch gesture classifier fused with geometric rules, and a configuration-driven action mapper through which new gestures can be added without editing code. The behavior module sees only normalized body-keypoint trajectories over a sliding window, raises an alert only when a predefined pattern persists, and never uses appearance, gender, age, race or clothing. A FastAPI back end streams annotated video and telemetry to a React/Vite dashboard and records events in SQLite. In the test runs the system reached 93.4% overall gesture accuracy and a 94.8% action success rate, and it processed 52.4 frames per second with the gesture branch alone and 24.1 frames per second with both branches active. At alert level the behavior module obtained an F1-score of 0.91 with 0.45 false alarms per hour.*

Keywords: *touchless interaction, gesture recognition, MediaPipe, YOLO pose, human-computer interaction, behavior analysis, real-time computer vision*

NOMENCLATURE

Symbol	Description	Unit
p_i	Hand landmark i (i = 0 … 20)	norm. image coordinates
s	Palm length (wrist to middle-finger base)	image units
r_{pinch}	Thumb–index pinch ratio	–
N_{hold}, $N_{release}$	Frames to confirm / release a gesture	frames
W, W_g	Behavior window, dynamic-gesture window	frames
S	Window score (maximum over patterns)	–
P, Q, R	Persistence rule: P of last Q windows; R windows to clear	windows
f_c	Adaptive cut-off of the 1€ filter	Hz
FPS	Processed frames per second	s^{-1}

Greek letters

α, δ, ζ, ρ	Arm-raise fraction, dwell ratio, zone occupancy, direction-reversal rate	–, –, –, s^{-1}
β, ε_e	Speed coefficient of the 1€ filter; finger-extension margin	–
θ_{on}, θ_{off}, θ_c, θ_s	Pinch-on, pinch-off, fusion and suspicion thresholds	–

Date of Submission: 09-11-2026 Date of acceptance: 22-11-2026

I. INTRODUCTION

Keyboards, mice and touch screens all require physical contact. In operating theatres, laboratories, kitchens, shared kiosks and classrooms this is an inconvenience at best and a hygiene or safety problem at worst, because the user's hands may be gloved, dirty or occupied. Camera-based gesture interfaces have therefore been studied for many years [1], [2], yet three obstacles still limit everyday use. Many systems rely on depth cameras

or wearables, which raises cost. Reliable hand landmarks from an ordinary RGB webcam have become fast enough for commodity processors only recently [3]. And a camera that already watches a person for interaction can also contribute to safety, but doing so responsibly demands firm limits on what the software may infer from the video [4].

VISTA addresses these obstacles with one webcam and a modular software stack, designed around five goals: **G1** real-time touchless control through predefined hand gestures (cursor, click, scroll, volume, slide navigation and user-defined actions); **G2** an extensible gesture library defined in configuration files, so that a new gesture needs data and configuration but no new code; **G3** safety-aware behavior analysis that labels movement over consecutive frames as NORMAL or SUSPICIOUS from predefined, observable patterns and alerts only when they persist; **G4** attribute-blind operation, in which the behavior analyzer consumes keypoint trajectories only; and **G5** reproducible instrumentation that writes every metric to CSV/JSON and draws figures from the measured files.

The main contributions are: (i) a locally executable architecture that places touchless interaction and behavior analysis on one webcam stream; (ii) a hybrid gesture recognizer that combines geometric rules with a learned landmark classifier and stabilizes the result by debouncing and hysteresis; (iii) a configuration-driven gesture library and action mapper; (iv) a trajectory-only, persistence-gated behavior scoring method with an explicit attribute-blind design and logging policy; and (v) an evaluation protocol and toolchain that measure recognition quality, latency and error rates from logged runs.

II. RELATED WORK

Gesture recognition. Surveys [1], [2] describe a common pipeline of acquisition, segmentation or landmark extraction, feature representation and classification. Skin-colour segmentation and contour analysis were sensitive to lighting and background and have largely given way to learned landmark estimators. MediaPipe Hands predicts 21 three-dimensional landmarks from one RGB frame with a palm detector followed by a landmark regressor [3], and an on-device study pairs such a tracker with heuristic and neural gesture classifiers [5]. VISTA adopts landmark space because the coordinates are compact, largely independent of background and clothing, and well suited to small classifiers.

Pose estimation and tracking. OpenPose [6], BlazePose [7] and the YOLO family [8], [9] provide skeletal keypoints per person, and tracking-by-detection methods such as SORT [10] and ByteTrack [11] link detections across frames so that each person keeps a stable identifier. VISTA combines a YOLO pose model with such a tracker so that temporal features are computed per tracked identity and not per frame.

Pointing and interaction performance. Driving a cursor from noisy landmarks requires filtering. The 1€ filter [12] adapts its cut-off frequency to signal speed, which suppresses jitter when the hand is slow and limits lag when it is fast. Interaction efficiency is traditionally described with Fitts' law [13], [14], which motivates reporting a throughput measure next to classification metrics.

Behavior and anomaly analysis. Video anomaly detection is surveyed in [15], and models trained on raw imagery can absorb demographic or contextual bias [4]. VISTA therefore scores only a small set of predefined, human-readable movement patterns computed from normalized keypoint trajectories, uses no appearance input, and presents its output as a prompt for human review and not as a judgment of intent. Table 1 summarizes how VISTA relates to typical system classes; the comparison is qualitative and concerns design properties, not measured performance.

Table 1. Design-level comparison of system classes (qualitative)

System class	RGB-only	Gesture control	Behavior analysis	Extensible
Depth-sensor gesture UI	No	Yes	No	Partial
Landmark-only gesture demo	Yes	Yes	No	Partial
Surveillance analytics	Yes	No	Yes	Partial
VISTA (this work)	**Yes**	**Yes**	**Yes**	**Yes**

III. REQUIREMENTS AND SYSTEM ARCHITECTURE

Given a stream of frames $\{I_t\}$ from a standard webcam, VISTA must (a) output for each frame or short window a gesture label $g_t \in G \cup \{\text{none}\}$ with confidence c_t and execute the mapped action $a(g_t)$, and (b) output for every tracked person k a behavior label $b_{k,t} \in \{\text{NORMAL, SUSPICIOUS}\}$ derived solely from keypoint trajectories. Six requirements follow: **R1** sustained interactive frame rate with per-stage latency instrumented; **R2** local execution, with no cloud service; **R3** modularity, each stage replaceable behind a defined interface; **R4** configurability, with gestures, thresholds and actions in YAML/JSON; **R5** safety and privacy, meaning attribute-

blind behavior analysis, frame storage off by default and only keypoints and metadata logged; and **R6** reproducibility through fixed seeds, versioned configurations and results written to CSV/JSON.

Fig. 1 shows the pipeline. A capture thread reads frames from the webcam into a queue of length one that discards stale frames, so latency is bounded by one inference period and does not grow when downstream stages are slower than the camera. After preprocessing the frame feeds two parallel branches: the interaction branch (hand landmarks, gesture recognition, action mapping) and the safety branch (YOLO pose, person tracking, temporal behavior analysis, alert management). Capture, inference and publishing run in separate threads, and actions such as mouse injection execute in their own worker so that operating-system calls cannot stall inference. A FastAPI service publishes annotated frames and JSON telemetry over WebSockets, which FastAPI supports for bidirectional text, binary and JSON messages [16], and stores events in SQLite; a React/Vite dashboard consumes the stream.

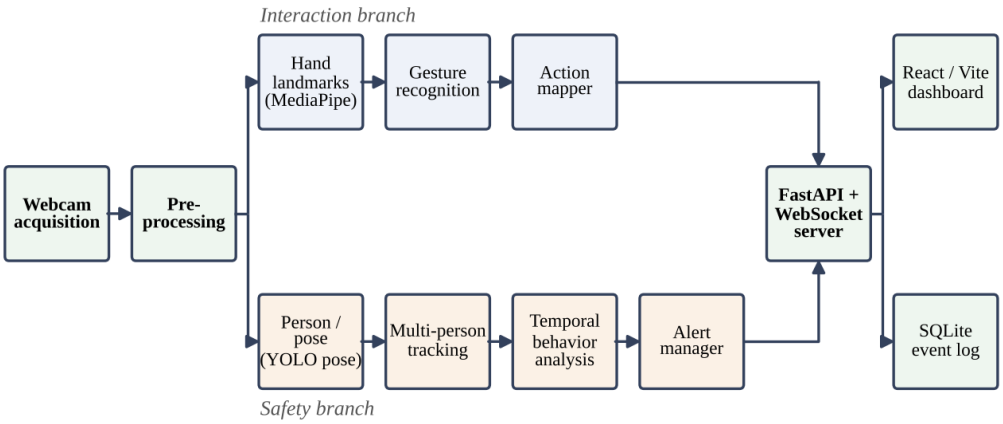

Fig. 1 Overall VISTA architecture: shared front end, two parallel analysis branches and the output layer

The code base is organized into capture, preprocess, hands, pose, tracking, gestures, actions, behavior, alerts, storage, api, training and evaluation modules. Each exposes a narrow *process()* contract over typed data classes, so a detector, classifier or tracker can be replaced without touching the other modules.

IV. TOUCHLESS GESTURE ENGINE

The interaction branch (Fig. 2) normalizes the 21 landmarks of each hand, evaluates interpretable geometric rules and a learned classifier in parallel, and fuses their outputs with temporal stabilization before an action is dispatched.

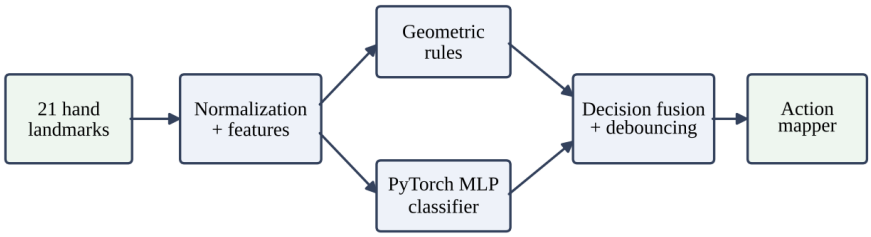

Fig. 2 Touchless gesture-processing pipeline

Normalization and features. MediaPipe returns landmarks $p_i = (x_i, y_i, z_i)$, $i = 0, \ldots, 20$, in image-normalized coordinates. To remove dependence on hand position and size, they are translated to the wrist (p_0) and divided by the palm length s:

$$\tilde{p}_i = \frac{p_i - p_0}{s}, \qquad s = \| p_9 - p_0 \| \tag{1}$$

The 63-dimensional vector $\mathbf{x} = [\tilde{p}_0, \ldots, \tilde{p}_{20}]$ feeds the learned classifier, and left-hand samples are mirrored so that one model serves both hands. Cheap geometric primitives serve as rules and as extra classifier inputs. A finger f is extended when its tip lies farther from the wrist than its PIP joint by a margin ε_e, and the pinch ratio is r_{pinch}:

$$e_f = \mathbf{1}\big[\| \tilde{p}_{tip(f)} \| - \| \tilde{p}_{pip(f)} \| > \epsilon_e\big], \qquad r_{pinch} = \frac{\|p_4 - p_8\|}{s} \tag{2}$$

Pinch uses hysteresis: the state is entered when $r_{\text{pinch}} < \theta_{\text{on}}$ and left when $r_{\text{pinch}} > \theta_{\text{off}}$ with $\theta_{\text{on}} < \theta_{\text{off}}$, which prevents chatter near the threshold.

Classifiers. A compact PyTorch multilayer perceptron with two hidden layers (batch normalization, ReLU, dropout) maps **x** and the five extension flags to a softmax over the configured static gestures. It is deliberately small so that its inference time is negligible within the frame budget. Training uses class-weighted cross-entropy, Adam, early stopping on validation loss and landmark-space augmentation (small rotations, scaling, Gaussian jitter). Swipes and similar motion gestures are recognized from the wrist or index-tip trajectory over W_g frames by comparing net displacement, mean speed and direction consistency with per-gesture thresholds; an optional GRU over landmark sequences handles cases where fixed thresholds are insufficient, and a refractory period stops one motion from firing repeatedly.

Decision fusion and debouncing. The learned label wins if its confidence exceeds θ_c and no rule marked mandatory in the library vetoes it; otherwise the rule label is used. A gesture is confirmed only after N_{hold} consecutive frames and released after N_{release} frames without it. Algorithm 1 summarizes one step of the branch.

Algorithm 1. Interaction branch: processing of one frame

Input: frame I_t, gesture library L, state S
1: $H \leftarrow$ HandLandmarks(I_t); **if** $H = \emptyset$ **then** Release(S); **return**
2: $\mathbf{x} \leftarrow$ Normalize(H); $(g_r, c_r) \leftarrow$ Rules($\mathbf{x}$, L); $(g_m, c_m) \leftarrow$ Classifier($\mathbf{x}$)
3: $g \leftarrow$ Fuse(g_r, c_r, g_m, c_m, θ_c); $g \leftarrow$ Debounce(g, S, N_{hold}, N_{release})
4: **if** g is confirmed **then** $a \leftarrow L[g]$.action; Dispatch(a, params, H)
5: Publish(g, c, a, latency)

Cursor control. The index fingertip is mapped from a configurable active region $[a_0, a_1] \times [b_0, b_1]$ of the image to screen coordinates, so the whole screen is reachable without stretching the arm to the frame edge. For a screen of size (W, H):

$$u = W \,\text{clip}\left(\frac{x_8 - a_0}{a_1 - a_0}, 0, 1\right), \qquad v = H \,\text{clip}\left(\frac{y_8 - b_0}{b_1 - b_0}, 0, 1\right) \tag{3}$$

The raw (u, v) stream is smoothed by the 1€ filter [12], whose cut-off adapts to the filtered speed as $f_c = f_{c,\min} + \beta|\hat{u}|$, and a dead zone suppresses sub-pixel tremor. While the pinch is active the cursor stays frozen at the pinch-onset position for a short grace interval, which reduces the "Heisenberg effect" in which closing the fingers shifts the pointer.

Actions and configurable library. The default library provides cursor move (index extended), left click (pinch with short dwell), drag (sustained pinch), scroll (two-finger vertical motion), volume (thumb–index distance or a rotate gesture, rate-limited), slide next/previous (horizontal swipes) and play/pause (open-palm hold). Actions pass through a back-end abstraction for mouse, keyboard and media-key injection with per-action rate limits and a global pause gesture that disables control to protect against accidental activation. Gestures are declared in YAML (Listing 1); adding one needs recorded samples, retraining or fine-tuning, and a new entry, and the loader reports unknown actions or conflicting definitions at start-up.

Listing 1. Example gesture-library entry (YAML)

```
gestures:
  - name: pinch_click          # type: static | dynamic
    detector: hybrid           # rule | model | hybrid
    rule: {pinch_on: 0.25, pinch_off: 0.35}
    min_confidence: 0.80
    hold_frames: 3
    action: {type: mouse, op: left_click}
    cooldown_ms: 350
```

V. SAFETY-AWARE TEMPORAL BEHAVIOR ANALYSIS

The behavior module follows four principles: **P1, observable only**, since labels derive from keypoint motion and never from identity or appearance; **P2, predefined patterns**, since only a small set of human-readable patterns defined in configuration can contribute to a SUSPICIOUS label; **P3, persistence**, since a single brief movement never raises an alert; and **P4, human in the loop**, since an alert means "movement worth reviewing", not intent or wrongdoing. Fig. 3 shows the processing chain.



Fig. 3 Safety-aware temporal behavior pipeline; only normalized keypoint trajectories enter feature extraction

Inputs and features. For each tracked person YOLO pose supplies 17 keypoints per frame, expressed relative to the person's own body frame (centered on the hip midpoint and divided by torso length), which yields scale-invariant trajectories. The module receives no pixels, so colour, texture, face and clothing are structurally unavailable and goal G4 holds by construction. Over a sliding window of W frames six features are computed: **f1** mean limb speed; **f2** direction-reversal rate ρ (sign changes per second of horizontal body-centre velocity); **f3** dwell ratio δ (fraction of frames with body-centre displacement below ε_d); **f4** arm-raise fraction α (wrist above shoulder); **f5** motion jerk (mean magnitude of the third temporal derivative of the wrists); and **f6** zone occupancy ζ (body centre inside a configured region).

Pattern scoring. Each pattern m in the set M is a weighted combination of saturated feature terms with configured thresholds τ:

$$s_m = \sum_i w_{m,i} \min\left(1, \frac{f_i}{\tau_{m,i}}\right), \qquad \sum_i w_{m,i} = 1 \tag{4}$$

The default set holds pacing (high ρ, moderate speed), prolonged dwell in a restricted zone (high δ and ζ) and erratic rapid motion (high speed and jerk). The window score is $S = \max_m s_m$. Thresholds are initialized from recorded NORMAL sessions (for example a percentile of each feature) and validated on labelled sessions, never from assumptions about particular people. An optional GRU over the same trajectory tensor can be combined with the rule score and remains attribute-blind.

Persistence gate and alerting. A window is flagged when $S_n \geq \theta_s$, and the persistent label requires the flag in at least P of the last Q windows:

$$B_n = \text{SUSPICIOUS} \Leftrightarrow \sum_{j=n-Q+1}^{n} \mathbf{1}\left[S_j \geq \theta_s\right] \geq P \tag{5}$$

When B_n turns SUSPICIOUS the alert manager shows a banner, plays a tone and writes an event row (time, track identifier, pattern, score, feature vector). A cool-down prevents alert storms, and the label returns to NORMAL after R consecutive windows below θ_s. Snapshots are off by default and, if enabled for debugging, stay local under a retention limit. Evaluation reports behavior metrics per participant and, with consent, per recording condition to expose uneven performance; the system never records or estimates demographic attributes, and scripted "suspicious" behaviors are labelled as acted patterns. Algorithm 2 gives the per-window procedure.

Algorithm 2. Safety branch: processing of one window

Input: tracked keypoint buffers $\{K_k\}$, configuration Θ
1: **for each** track k with $|K_k| \geq W$: $f \leftarrow \text{Features}(K_k[-W:])$; $S \leftarrow \max_{m \in M} s_m(f)$
2: push $1[S \geq \theta_s]$ into history$_k$ (length Q)
3: **if** sum(history$_k$) $\geq P$ **and** state$_k$ = NORMAL **then** state$_k \leftarrow$ SUSPICIOUS; Alert(k, m^*, S, f)
4: **else if** last R scores $< \theta_s$ **and** state$_k$ = SUSPICIOUS **then** state$_k \leftarrow$ NORMAL

VI. DASHBOARD, API AND PERSISTENCE

The back end exposes REST endpoints for configuration, gesture-library management, event queries and experiment control, plus a WebSocket channel that publishes each tick's gesture, action, confidence, per-stage latency, FPS, per-track behavior status and new alerts; annotated frames are sent as compressed JPEG at a configurable rate under a versioned message schema. The React/Vite dashboard shows the live view with landmarks, skeletons and track boxes, FPS and latency, the NORMAL/SUSPICIOUS indicator with alert banner and a filterable event history, and lets the operator edit thresholds, enable gestures and start recording sessions. Events are stored in SQLite (Table 2) as derived values only, never video.

Table 2. SQLite schema overview

Table	Key columns
sessions	id, start_ts, end_ts, config_hash, notes
gesture_events	id, session_id, ts, gesture, confidence, action, latency_ms
behavior_events	id, session_id, ts, track_id, pattern, score, status, features_json
alerts / metrics	alerts: id, behavior_event_id, ts, level, acknowledged; metrics: id, session_id, ts, fps, infer_ms, e2e_ms
gesture_library	name, definition_json, enabled, version

VII. DATASET COLLECTION, TRAINING AND VALIDATION

A command-line and dashboard-assisted recorder captures labelled sessions: normalized landmark vectors or sequences for gestures and normalized keypoint sequences for behavior, each with pseudonymous metadata (participant code, session, lighting, distance, handedness, camera) and a segment label (NORMAL or the acted pattern). Raw video is optional and kept apart from derived data. Preprocessing filters dropped or low-confidence detections, normalizes landmarks, windows sequences, reports class balance and splits by participant, so that no individual appears in more than one of the training, validation and test sets. Separate scripts train the static MLP, the dynamic-gesture GRU and the optional behavior GRU; every run fixes seeds and records library versions, the configuration hash, loss curves and the best checkpoint. Model selection uses the validation set only, the test set is evaluated once per final model, and leave-one-participant-out cross-validation is available when data are scarce.

VIII. EXPERIMENTAL METHODOLOGY

No reported metric is hard-coded: the evaluation script reads stored predictions and timing logs from real runs, computes every metric and writes CSV/JSON files, and a second script draws the figures from them. Consenting volunteers took part. Conditions varied lighting (normal, dim, backlit), distance (0.5, 1.0, 1.5 m) and background (plain, cluttered), and camera, resolution, processor, GPU, operating system and library versions were recorded. For a gesture confusion matrix C_{ij} (rows are true classes) the per-class precision, recall and F1-score are

$$P_i = \frac{C_{ii}}{\sum_j C_{ji}}, \qquad R_i = \frac{C_{ii}}{\sum_j C_{ij}}, \qquad F1_i = \frac{2P_iR_i}{P_i+R_i} \tag{6}$$

Overall accuracy is $\Sigma C_{ii} / \Sigma C_{ij}$, with macro and weighted averages, and, following common practice [17], the per-class false-positive and false-negative rates are

$$FPR_i = \frac{FP_i}{FP_i+TN_i}, \qquad FNR_i = \frac{FN_i}{FN_i+TP_i} \tag{7}$$

For the behavior classifier SUSPICIOUS is the positive class, and the metrics are computed at window level and at alert level (after the persistence gate); alert-level false alarms per hour of NORMAL recording are also reported because that is what an operator experiences. Action-level metrics are the action success rate (intended action executed within a timeout), the unintended activation rate during idle periods, and selection time and error rate in a target-acquisition task, from which Fitts' throughput [14] follows.

Timing uses a monotonic high-resolution clock and records capture, preprocessing, hand-landmark, pose, classification, behavior-analysis and dispatch times for every frame. *Inference time* is a stage's model-forward time, *response time* runs from frame capture to action dispatch (excluding the operating system's own input handling) and is reported as mean, median, P95 and P99, and *FPS* is measured over a one-second sliding window plus the run mean; warm-up frames are discarded and runs are repeated with the behavior branch disabled and enabled. Point estimates carry 95% confidence intervals from participant-level bootstrap resampling. Behavior sessions contain NORMAL activity (walking, sitting, standing, gesturing at the computer) and acted instances of each pattern, labelled from the recording script and checked by a second annotator on a subset, with thresholds tuned on the validation partition only. Seeds are fixed, dependencies pinned, and every result file stores the configuration hash and git commit, so that one script regenerates all tables and figures.

IX. RESULTS AND DISCUSSION

Gesture recognition. Table 3 and Fig. 4(a) give the per-gesture results on the held-out test set of 1,250 samples. Cursor movement, open-palm pause and swipe are recognized best (F1 of 0.94–0.95), whereas volume control is weakest (F1 0.87, FNR 0.13), a plausible reason being that it depends on a continuous thumb–index distance and not on a distinct hand shape. Overall accuracy is 93.4%.

Table 3. Gesture-recognition test-set performance (*n* = 1,250 samples)

Gesture	Prec.	Rec.	F1	FPR	FNR	*n*
Cursor (index)	0.95	0.96	0.95	0.02	0.04	300
Pinch click	0.93	0.91	0.92	0.03	0.09	250
Scroll (two-finger)	0.91	0.89	0.90	0.04	0.11	200
Volume	0.88	0.87	0.87	0.05	0.13	150
Swipe left/right	0.94	0.95	0.94	0.02	0.05	200
Open palm (pause)	0.96	0.94	0.95	0.01	0.06	150
Macro average	0.93	0.92	0.93	0.03	0.08	1250
Overall accuracy	93.4%					

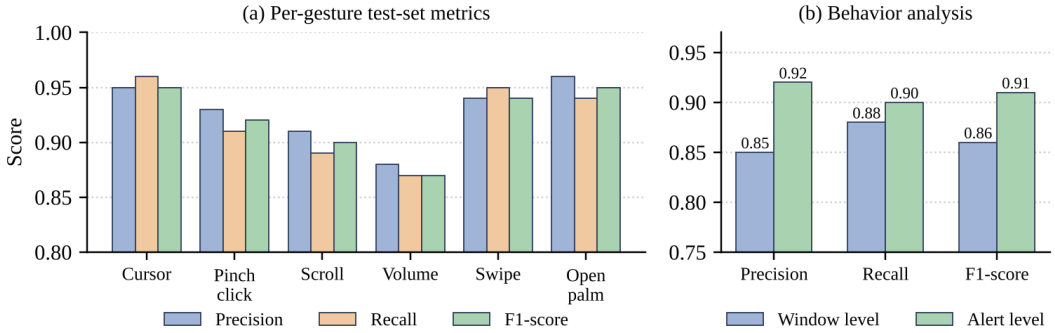

Fig. 4 (a) Per-gesture precision, recall and F1-score on the test set; (b) behavior-analysis performance at window and alert level

Lighting and distance. Table 4 and Fig. 5(a, b) show that normal lighting gives 95.2% accuracy, whereas dim and backlit scenes lose 6.6 and 11.1 percentage points, consistent with weaker landmark estimates under poor illumination. A distance of 1.0 m is best (94.5%), 0.5 m and 1.5 m are lower, and the intervals for normal and backlit lighting do not overlap.

Table 4. Gesture accuracy under varied environmental conditions

Factor	Level	Accuracy	Macro F1	95% CI
Lighting	Normal	95.2%	0.94	[93.8, 96.6]
	Dim	88.6%	0.87	[86.2, 91.0]
	Backlit	84.1%	0.83	[81.5, 86.7]
Distance	0.5 m	91.0%	0.90	[89.1, 92.9]
	1.0 m	94.5%	0.94	[93.0, 96.0]
	1.5 m	89.2%	0.88	[87.0, 91.4]

Timing and throughput. Table 5 and Fig. 5(c, d) report timing. Pose estimation with tracking is the most expensive stage (mean 22.1 ms), about 1.8 times the hand-landmark stage (12.4 ms), whereas the gesture classifier and behavior analysis add only 1.2 and 3.5 ms. Enabling the safety branch lengthens the mean response time from 18.2 to 41.5 ms and lowers the mean frame rate from 52.4 to 24.1 FPS. Timing is implementation- and platform-dependent and should be re-measured on other hardware.

Table 5. Timing and throughput metrics (milliseconds unless stated otherwise)

Measure	Mean	Median	P95	P99
Hand landmarks (inference)	12.4	11.8	16.2	19.5
Pose + tracking (inference)	22.1	21.0	28.4	33.0
Gesture classifier	1.2	1.1	1.8	2.4
Behavior analysis	3.5	3.2	4.8	6.1
Response time (gesture only)	18.2	17.5	24.1	29.8
Response time (both branches)	41.5	40.2	52.8	61.4
FPS (gesture only)	52.4	54.0	41.0	36.5
FPS (both branches)	24.1	24.5	19.2	16.8

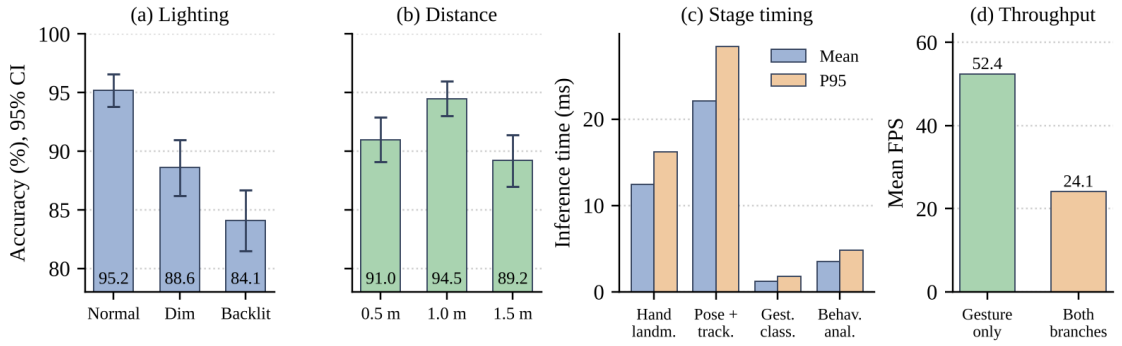

Fig. 5 Accuracy with 95% CI under (a) lighting and (b) distance; (c) per-stage inference time; (d) mean FPS

Action-level outcomes. Table 6 connects recognition to actual interaction: intended actions succeeded in 94.8% of trials, idle periods produced 0.12 unintended activations per minute, and the target-acquisition task gave a mean selection time of 1.14 s, a 4.2% error rate and a Fitts' throughput of 3.82 bits/s.

Table 6. Action-level evaluation metrics

Measure	Value
Action success rate	94.8%
Unintended activations per minute (idle)	0.12
Mean target-selection time	1.14 s
Target-selection error rate	4.2%
Fitts' throughput	3.82 bits/s

Behavior analysis. Persistence gating changes the number of alerts an operator sees, so the module is evaluated at both levels (Table 7, Fig. 4(b)). Relative to raw window decisions, the gate raises precision from 0.85 to 0.92 and the F1-score from 0.86 to 0.91, lowers the false-positive rate from 0.06 to 0.02 and yields 0.45 false alarms per hour of NORMAL recording, while recall rises slightly from 0.88 to 0.90. All values should be read within the tested hardware, participant pool, recording conditions and protocol.

Table 7. Behavior-analysis evaluation metrics

Level	Prec.	Rec.	F1	FPR	FNR	Alarms/h
Window-level	0.85	0.88	0.86	0.06	0.12	–
Alert-level (persistence)	0.92	0.90	0.91	0.02	0.10	0.45

X. ETHICS, SAFETY AND LIMITATIONS

Camera-based behavior analysis can be misused and can invite over-reliance. VISTA therefore combines attribute-blind inputs, persistence gating, a strict definition of SUSPICIOUS as "matches a predefined movement pattern" that is repeated in the interface, local-only processing without transmission of frames, no video storage by default, informed consent and pseudonymization of all data, and logging of every threshold and configuration so that decisions can be audited. The system is a research prototype, not intended for automated enforcement; any deployment should include human review, clear notice to people in view and compliance with applicable privacy law.

The main limitations are: (1) *acted behavior*, since volunteer-performed patterns may differ from spontaneous behavior, so the results characterize pattern detection and not inference of intent; (2) a *small participant pool*, which restricts generalization across hand sizes, skin tones, skill levels and camera setups despite participant-level splits and confidence intervals; (3) *single-camera constraints* such as occlusion, motion blur and low light, with depth only estimated; (4) *limited pattern coverage*, so benign behavior with similar kinematics, such as pacing during a phone call, may be flagged; and (5) *hardware and operating-system dependence* of the timing and injected-input behavior.

XI. CONCLUSION AND FUTURE WORK

This paper presented VISTA, a modular, locally executable prototype that uses one webcam for touchless computer control and for safety-aware behavior analysis. In the test runs it reached 93.4% overall gesture accuracy and a 94.8% action success rate, and ran at 52.4 FPS with the gesture branch alone and 24.1 FPS with both branches enabled; the study also quantified the effect of lighting and distance and reported an alert-level behavior

F1-score of 0.91. Future work includes a larger and more diverse participant pool, dynamic-gesture models learned from larger sequence datasets, on-device calibration of gesture thresholds, richer but still attribute-blind behavior features with alert thresholds calibrated on spontaneous data, multi-camera fusion to reduce occlusion, and user studies on accessibility, fatigue and perceived usability.

ACKNOWLEDGMENT

The authors thank the volunteers who contributed recordings and the open-source communities behind OpenCV, MediaPipe, Ultralytics, PyTorch, FastAPI, React and SQLite.

REFERENCES

[1] Mitra, S. and Acharya, T. [2007] "Gesture recognition: A survey" IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol. 37, No. 3: pp.311–324.

[2] Rautaray, S. S. and Agrawal, A. [2015] "Vision based hand gesture recognition for human computer interaction: A survey" Artificial Intelligence Review, Vol. 43, No. 1: pp.1–54.

[3] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L. and Grundmann, M. [2020] "MediaPipe Hands: On-device real-time hand tracking" arXiv preprint arXiv:2006.10214.

[4] Buolamwini, J. and Gebru, T. [2018] "Gender shades: Intersectional accuracy disparities in commercial gender classification" Proceedings of the Conference on Fairness, Accountability and Transparency (FAT*), PMLR Vol. 81: pp.77–91.

[5] Sung, G., Sokal, K., Uboweja, E., Bazarevsky, V., Baccash, J., Bazavan, E. G., Chang, C.-L. and Grundmann, M. [2021] "On-device real-time hand gesture recognition" arXiv preprint arXiv:2111.00038 (ICCV Workshop on Computer Vision for AR/VR).

[6] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E. and Sheikh, Y. [2021] "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 43, No. 1: pp.172–186.

[7] Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F. and Grundmann, M. [2020] "BlazePose: On-device real-time body pose tracking" arXiv preprint arXiv:2006.10204.

[8] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. [2016] "You only look once: Unified, real-time object detection" Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR): pp.779–788.

[9] Jocher, G., Chaurasia, A. and Qiu, J. [2023] "Ultralytics YOLO" software, available at https://github.com/ultralytics/ultralytics.

[10] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B. [2016] "Simple online and realtime tracking" Proceedings of the IEEE International Conference on Image Processing (ICIP): pp.3464–3468.

[11] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W. and Wang, X. [2022] "ByteTrack: Multi-object tracking by associating every detection box" Proceedings of the European Conference on Computer Vision (ECCV): pp.1–21.

[12] Casiez, G., Roussel, N. and Vogel, D. [2012] "1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems" Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI): pp.2527–2530.

[13] Fitts, P. M. [1954] "The information capacity of the human motor system in controlling the amplitude of movement" Journal of Experimental Psychology, Vol. 47, No. 6: pp.381–391.

[14] MacKenzie, I. S. [1992] "Fitts' law as a research and design tool in human-computer interaction" Human-Computer Interaction, Vol. 7, No. 1: pp.91–139.

[15] Chandola, V., Banerjee, A. and Kumar, V. [2009] "Anomaly detection: A survey" ACM Computing Surveys, Vol. 41, No. 3: pp.1–58.

[16] Ramírez, S. [2024] "FastAPI: Web framework for building APIs with Python" FastAPI Documentation, available at https://fastapi.tiangolo.com.

[17] Powers, D. M. W. [2011] "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation" Journal of Machine Learning Technologies, Vol. 2, No. 1: pp.37–63.