AMME5710 Computer Vision & Image Processing Hand-Gesture Controlled Media Player

Siwon Kang
SID: 510440895
skan6221@uni.sydney.edu.au

Varunvarshan Sideshkumar
SID: 520534445
vsid0095@uni.sydney.edu.au

Arin Adurkar
SID: 520587980
aadu5646@uni.sydney.edu.au

ABSTRACT

This project presents a hybrid webcam pipeline combining mask fusion, geometric descriptors and lightweight classifiers for real-time hand-gesture control. The system achieves high per-class precision and recall on an in-house dataset while remaining interpretable and efficient for CPU-only deployment. Live tests show stable behaviour under moderate occlusion and face proximity; Random Forest and SVM delivered the strongest quantitative performance. Simple calibration and a compact CNN/landmark fallback are proposed to improve reliability in unconstrained settings, prioritising low data requirements and fast tuning for media-control applications.

I. INTRODUCTION

Conventional media players rely on physical input devices such as keyboards, mouse, or remote controls, which require direct contact and can be inconvenient in scenarios where hands are occupied, contaminated, or when operating in sterile environments (e.g. surgical theatres, kitchens, or public kiosks). This limitation motivates the development of a *touchless control interface*. We propose a real-time hand gesture-controlled media player that uses only a standard webcam, eliminating the need for additional hardware while enabling natural human-computer interaction.

Motivation behind choosing a *vision-based approach* offers four key advantages: **(i)** no extra hardware beyond simple webcams, **(ii)** adaptability to diverse applications (e.g. smart TVs, presentation control, accessibility aids), **(iii)** intuitive gesture semantics that align with human motor habits, and **(iv)** scalability to resource-constrained devices.

Key *challenges* include: **(i)** varying illumination and cluttered backgrounds, **(ii)** diverse hand morphologies and orientations, **(iii)** partial occlusions, **(iv)** 120 ms latency, and **(v)** limited labelled training data. Deep learning approaches, while accurate, demand GPUs and large datasets [1], rendering them impractical for lightweight deployment. Instead, we leverage classical computer vision using HSV and YCrCb skin segmentation, morphological refinement, convex hull analysis, and geometric features paired with an SVM classifier, building on low-cost, real-time precedents [2], [3].

Aim:

- To design and implement a webcam-only pipeline achieving $>$95% gesture recognition accuracy with end-to-end latency $<$120 ms on standard CPU hardware.
- To ensure robustness across lighting conditions, backgrounds, and skin tones via adaptive dual-colour-space segmentation and morphological refinement.
- To enable seamless control of play/pause, volume up/down, and track navigation using six intuitive static hand gestures, validated on an in-house dataset of $\sim$200 samples per gesture class.

This paper details our methodology, experimental validation, and future extensions toward thermal imaging and wearable integration.

II. LITERATURE REVIEW

Hand gesture recognition for touchless control has been studied across classical computer vision, deep learning, and emerging modalities. We examine foundational skin segmentation, real-time media control systems, modern alternatives, and future directions.

A. Classical Computer Vision Pipelines

Oudah et al. [2] reviewed over 150 techniques and established HSV-based skin segmentation, contour detection, convex hull analysis, and defect counting as the benchmark for low-cost, real-time systems. When paired with k-NN or SVM classifiers, these pipelines deliver reliable performance using only standard webcams, making them suitable for resource-constrained deployment. This classical framework forms the backbone of our approach due to its transparency and minimal hardware requirements.

Rautaray and Agrawal [3] implemented this pipeline to control VLC media player, using fixed HSV thresholds and finger counting via convex hull defects to trigger play/pause and volume commands. Their system achieved approximately 90% accuracy across five static gestures in controlled environments. However, the rigid HSV ranges failed under varying illumination and skin tones, limiting practical use. One way to overcome this is by introducing *live HSV tuning via OpenCV trackbars and fusing YCrCb masking to adapt dynamically to lighting and shadow variations.*

Skin colour segmentation is the cornerstone of such

systems. Shaik et al. [4] compared HSV and YCrCb colour spaces, finding that HSV processes frames in under 10 ms and resists global illumination shifts, while YCrCb better preserves hand regions in shadowed areas. Their analysis highlighted complementary strengths but no prior work fused both spaces adaptively. A possible solution to this could be addressing this through a bitwise AND of HSV and YCrCb masks followed by morphological closing operations, producing cleaner segmentation across diverse real-world conditions.

B. Deep Learning and Pre-trained Frameworks

Modern deep learning models offer higher accuracy at increased computational cost. Alnuaim et al. [1] trained ResNet and MobileNet on over 10,000 labeled samples, achieving 98% accuracy on ten gestures. While impressive, inference required GPU acceleration and extensive data collection, conflicting with our goal of webcam-only, CPU-bound operation under 120 ms latency. Classical methods remain superior for lightweight, data-scarce applications.

Early in our development, we explored MediaPipe Hands [5], a pre-trained deep learning solution providing real-time landmark detection. Despite its efficiency on modest hardware, segmentation degraded in low light or with skin-like backgrounds (e.g. wooden tables), and threshold adjustment was unavailable. This prompted a complete redesign using custom classical vision, prioritising robustness and user control over black-box accuracy (refers to the high-level prediction performance of a system where the internal workings are not known or understandable, focusing solely on the accuracy of its output compared to the input).

C. Emerging Modalities for Enhanced Robustness

To eliminate illumination dependency, Vandersteegen et al. [6] demonstrated low-latency gesture recognition using low-resolution thermal imagers. By thresholding on heat contrast, their system operated reliably in complete darkness with minimal preprocessing. Although effective, thermal sensors add cost and complexity, hence, future integration could serve as a fallback for critical environments.

For dynamic gesture tracking, Wang et al. [7] surveyed MEMS-based wearable sensors (data gloves, optical, and fiber optic). These achieve sub-millimeter precision but require physical contact, contradicting touchless interaction goals. *A research gap is that there is no solution provided to retain vision as the primary modality while considering hybrid fusion in future work.*

D. Contribution and Positioning

Table I compares key systems. Prior vision-based media controllers [3] lack adaptive segmentation and real-world robustness testing. Deep learning solutions [1] demand excessive resources. No existing work integrates *dual-colour-space fusion*, *live threshold calibration*, *geometric feature stability*, and *real-world validation* in a fully integrated, webcam-only pipeline. *This research gap could be bridged with a redesigned system achieving higher accuracy and lesser latency on CPU, validated across lighting, background clutter, and skin tones.*

TABLE I
RESULTS FROM PRIOR MEDIA CONTROL SYSTEMS.

Method	Accuracy	Latency	Hardware
Rautaray [3]	∼90%	Real-time	Webcam
Alnuaim [1]	98%	High	GPU + 10k data

III. PIPELINE OVERVIEW

A. Pipeline Overview

The system follows a five-stage pipeline:

1) data collection with dual color-space masks, optional MediaPipe region-of-interest gating, contour selection, feature logging, and saving of raw, mask, crop, and overlay views
2) classical model selection across SVM, KNN, Decision Tree, and Random Forest with stratified split and cross-validation; export of scaler, best model, label map, and exact feature order
3) live controller that reproduces collection masks, applies ROI gating when available, tracks centroid trails and unwrapped rotation in radians, and maps stable labels to media actions
4) deep baseline using ResNet-18 fine-tuning with deterministic splits and ONNX export with diagnostics
5) independent verification that reloads artifacts, re-tests accuracy and calibration, checks counts and top-k monotonicity, and validates PyTorch to ONNX parity

TABLE II
SOURCE FILES AND RESPONSIBILITIES

collect_data.py	real-time collector; builds HSV and YCrCb masks, optional MediaPipe ROI gate, fuses masks, extracts features, saves raw/mask/crop/overlay, logs CSV
train_data.py	classical training and selection; standardizes numeric features, trains SVM/KNN/DT/RF with stratified k-fold, saves gesture_scaler.pkl, gesture_model.pkl, label_map.npy, feature_names.npy
media_controller.py	runtime controller; reproduces mask fusion, selects largest plausible hand blob, uses convexity-defect cues, centroid trail, radians rotation cw/ccw, stable-episode logic to trigger media keys
benchmarking_resnet_18.py	deep baseline; ImageFolder packing, deterministic train/val/test, ResNet-18 fine-tune, reports and ONNX export
resnet_18_verify.py	artifact verification; reloads weights and data, recomputes metrics, sanity checks, and PyTorch to ONNX parity

IV. METHODOLOGY

This section describes the experimental protocol, signal and image preprocessing, mask construction and fusion, geometric

and motion feature extraction, classifier training, and the live controller implementation. Explanations are written in a concise student tone and include the most important formulae used in the implementation.

A. Data Acquisition and Monitoring

- Video frames are captured at resolution 960×540 and 30 frames per second. Camera settings (exposure, gain, white-balance) are fixed where possible and recorded per session to ensure reproducibility.
- The capture interface displays a tiled monitor containing: raw RGB, HSV mask, YCrCb mask, fused mask, fused mask with contour overlay, and a motion-trail panel. This facilitates rapid threshold tuning during collection.
- Each saved sample includes the raw frame, the final fused mask crop, and an annotated JSON record of per-frame features (centroid, area, fingertip count, rotation accumulator, timestamp).

B. Preprocessing

- **gray_world_wb(bgr)** applies a simple but effective white-balance technique based on the gray-world assumption, which assumes that the average colour of a scene should be a neutral gray. It calculates the mean intensity of each BGR channel: m_b, m_g, m_r, then computes their average $m = (m_b + m_g + m_r)/3$ to use as a reference level. Each channel is scaled by the ratio of this global mean to its own mean:

$$B' = B \cdot \frac{m}{m_b}, \quad G' = G \cdot \frac{m}{m_g}, \quad R' = R \cdot \frac{m}{m_r} \tag{1}$$

This step reduces colour bias due to lighting and helps standardise the input across frames.

Fig. 1. Raw input frame.

- **adaptive_gamma_from_Y(bgr)** helps handle brightness variations by computing a gamma correction based on the luminance (Y channel) of the input frame. It calculates the mean luminance μ_Y, then adjusts the gamma value dynamically:

$$\gamma \approx \text{clip}(1.4 - \mu_Y, 0.7, 1.3)$$

This ensures that darker images are brightened and overly bright ones are toned down. The final gamma correction is applied using a LUT that maps original intensities to corrected values:

$$I' = \text{LUT}\big(I, (i/255)^{1/\gamma}\big) \tag{2}$$

This step improves consistency of brightness across varying lighting conditions.

C. Mask Construction and Morphology

- **build_masks(raw_bgr)** is a pipeline function that combines white-balance, gamma correction, smoothing, and thresholding. It converts the colour-balanced image into two colour spaces: HSV and YCrCb. Using OpenCV's inRange function, binary masks m_{HSV} and m_{YCrCb} are created to isolate hand-like colours.

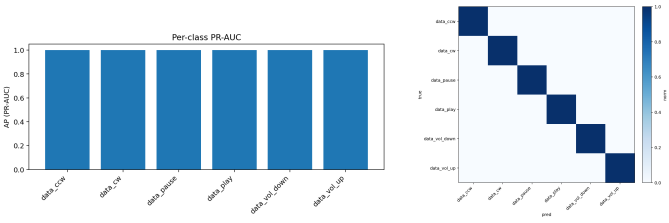

Fig. 2. Per-class PR-AUC (left) and row-normalised confusion matrix (right). These highlight detection quality and confusion trends, guiding data collection and augmentation.

- **postprocess_mask(m)** cleans up these raw masks using morphological operations, which are key to removing noise and closing small gaps in the mask. Specifically, it first applies an opening operation (erosion followed by dilation) to remove small blobs, followed by a closing operation (dilation then erosion) to seal holes:

$$m_{\text{clean}} = \text{Close}(\text{Open}(m; K_o); K_c) \tag{3}$$

The output is a smoother and more connected binary mask, ready for contour detection.
- **apply_vertical_gate(mask, $y_{\min}$, $y_{\max}$)** restricts detection to a vertical region of interest in the image. It creates a binary mask that allows only pixels between $y_{\min}H$ and $y_{\max}H$ (where H is the image height), effectively ignoring parts of the image like the top background. The final gated mask is $m \times G$.

D. Mask Scoring and Fusion Logic

- **largest_blob(m)** processes the binary mask and extracts the largest connected contour (or "blob") that fits within a given area range. This is used as the main candidate for further analysis.
- **mask_score(m)** evaluates how good or "hand-like" a mask is. It does this by combining three shape descriptors:
  - the area ratio $A/(WH)$ (relative size),
  - convexity $s = A/A_{\text{hull}}$ (how filled the shape is), and
  - circularity $c = 4\pi A/P^2$ (how close the shape is to a circle).

  These are combined into a heuristic score:

$$s(m) \approx \left(\frac{A}{WH}\right) \cdot s \cdot \text{clip}(c, 0.2, 1) \tag{4}$$

The result helps decide which mask is more likely to be valid.
- **fuse(hsv_m, ycc_m, shape)** merges the HSV and YCrCb masks using rules. If both masks agree spatially (high

IoU) and their intersection $m_\wedge$ scores well, then it is used. If not, the better of the two single masks is used. As a last resort, the union $m_\vee$ is used to maximise recall. This logic adapts to situations where one colour space fails.

E. MediaPipe Prior and ROI Gating

- **MPPrior** wraps the optional MediaPipe Hands module. If MediaPipe is not available, it disables this part of the pipeline.
- **infer(bgr)** runs the detector and outputs:
  - bounding boxes and labels for debugging or overlay, and
  - convex hull polygons of each hand used for gating.
- **gate_with_polys(m, polys, inflate_px)** restricts the binary mask to only include pixels inside the detected hand regions. The polygons are inflated by a few pixels, converted into a binary mask, and logically ANDed with m:

$$m \leftarrow m \cap R$$

If no hands are detected, this step is skipped to avoid false negatives.

F. Contour Selection and Fingertip Extraction

- **select_hand_like(m, shape)** evaluates all external contours and ranks them using a scoring function that combines shape features:

$$\text{score} \propto \left(\frac{A}{HW}\right)^{0.9} \cdot s^{0.6} \cdot (1-c)^{1.2} \cdot \text{extent}^{0.4} \tag{5}$$

The goal is to prioritise contours that are large, convex, and elongated—like a hand.
- **safe_convexity_defects(cnt)** tries to compute convexity defects (which help detect fingertips). If the contour is too simple or degenerate, it falls back to a polygon approximation.
- **gesture_from_cnt(cnt)** counts the number of fingertips by checking each convexity defect. A defect counts as a fingertip if it forms a sharp angle and is deep enough. The depth d is calculated as:

$$d = \frac{|(p_b - p_a) \times (p_a - p_f)|}{\|p_b - p_a\|} \tag{6}$$

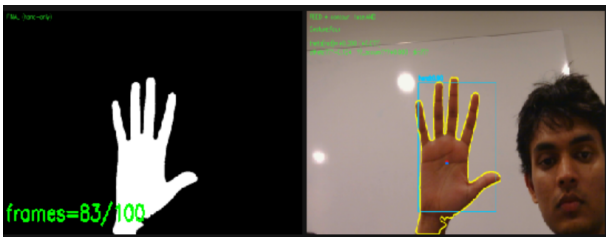

Fig. 3. Fused mask with ROI gate and final contour.

G. Centroid, Moments, and Geometric Descriptors

- The centroid is derived from image moments M_{pq} using:

$$c_x = \frac{M_{10}}{M_{00}}, \quad c_y = \frac{M_{01}}{M_{00}} \tag{7}$$

It gives the average position of the hand in the frame and is normalised by image dimensions.
- Other shape descriptors include:
  - solidity: A/A_{hull}
  - circularity: $4\pi A/P^2$
  - extent: $A/(b_w b_h)$
  - aspect ratio: b_w/b_h

These are used as input features for gesture classification.

H. Motion Trail and Rotation Estimation

- The centroid position is tracked over time. The trail canvas decays using:

$$T_t = \gamma T_{t-1} \tag{8}$$

A new segment is added when the hand moves a significant distance.
- The rotation is computed relative to a moving pivot point p_t, updated using exponential moving average:

$$p_t = (1-\alpha)p_{t-1} + \alpha q_t \tag{9}$$

where $q_t = (c_x, c_y)$ is the current centroid.
- The angle of rotation is:

$$\theta_t = \text{atan2}(v_y, v_x) \tag{10}$$

where v is the vector from pivot to centroid.
- The rotation change is unwrapped (to avoid 2π jumps) and smoothed:

$$\tilde{\Delta}\theta_t = (1-\beta)\tilde{\Delta}\theta_{t-1} + \beta\Delta\theta_t \tag{11}$$

- A cumulative rotation accumulator integrates the smoothed deltas:

$$\Theta_t = \lambda\Theta_{t-1} + \tilde{\Delta}\theta_t \tag{12}$$

If $|\Theta_t|$ exceeds a predefined threshold, a clockwise or counterclockwise gesture is triggered.

I. I/O, Visualization, and CSV Logging

- **tile2x3(...)** composes six visual panels into a 2×3 grid for debugging and live feedback. It includes raw input, binary masks, overlays, and motion trails.
- **put_kv(...)** is a helper to annotate the image with text values like rotation angle or gesture label for easy interpretation during runtime.
- Each frame saves:
  - the raw RGB image,
  - the binary mask as PNG,
  - a cropped ROI image, and
  - an overlay image.

A CSV row is also written with all key geometry and motion metrics for training or analysis.

J. Gesture Inference and Mapping

- The system supports both deterministic rule-based logic and optional supervised classification. Rule logic includes:
  - **Open palm:** detected when the number of extended fingertips $n_{\text{tips}} \geq 4$ and the normalised area $A_n \in [A_{\min}, A_{\max}]$.
  - **Closed fist:** occurs when $n_{\text{tips}} \leq 1$ and a sharp area drop is observed compared to recent frames.
  - **Thumb gestures:** thumb up or down is inferred based on the angular position of a single fingertip relative to the palm axis.
  - **Rotations:** gestures like clockwise (cw) or counterclockwise (ccw) are triggered by the sign and magnitude of the accumulated rotation angle Θ_t held across T_s frames.
- A compact feature vector is constructed for classification:

$$\mathbf{x} = [A_n, c_x^n, c_y^n, n_{\text{tips}}, \bar{r}, \omega, \text{solidity}, \text{aspect}] \tag{13}$$

  where A_n is area normalised by frame size, (c_x^n, c_y^n) is the normalised centroid, $\bar{r}$ is average fingertip distance, ω is the smoothed angular velocity, and the remaining terms describe hand geometry. This vector is used by SVM or Random Forest classifiers trained offline.
- Stability is enforced using temporal hysteresis. A label is accepted only if it appears in at least η frames within a sliding window of size L. If classifier confidence exceeds threshold κ, hysteresis is bypassed for faster transitions.

K. Model Training and ResNet-18 Baseline

- All numeric features are standardised using:

$$x' = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}} \tag{14}$$

  where the mean and standard deviation are computed from the training split.
- A stratified k-fold cross-validation ($k = 5$) is used for hyperparameter search. Classical classifiers include SVM (RBF), KNN, Decision Tree, and Random Forest. Model selection is based on test macro-F1 and calibration error (ECE).
- For image-based classification, a ResNet-18 is fine-tuned on cropped hand images. It uses a cosine annealing learning-rate schedule and typical data augmentations. Train/val/test splits are fixed, and the best model is exported to ONNX for compatibility checks.

L. Real-Time Implementation Considerations

- To meet interactive constraints, total latency is broken down as:

$$t_{\text{frame}} = t_{\text{capture}} + t_{\text{mask}} + t_{\text{morph}} + t_{\text{contour}} + t_{\text{feature}} + t_{\text{decision}} \tag{15}$$

  with a hard budget of $t_{\text{frame}} \leq 33$ ms (30 FPS).
- Speedups include running at lower resolution (e.g. 640×360), caching morphological kernels, computing optical flow only within ROI, and using vectorised moment calculations.
- For reproducibility, the system stores all thresholds, kernel definitions, EMA parameters (α, β, λ), classifier seeds, dataset splits, and fusion logic constants in the project repository.

V. Experimental Setup and Benchmarking

A. Dataset and Split

The dataset comprises ROI crops generated by the mask-fusion collector. Frames were captured at 960×540 resolution and 30 fps in indoor settings with modest lighting variation. Crops were obtained by applying the fused mask (HSV/YCrCb), gated by MediaPipe hand ROI, and enclosing a tight square bounding box that was resized for model input.

A stratified split preserved class distribution: 80% training, 10% validation, and 10% testing. Random seeds were fixed for reproducibility. Fig. 4 shows test split support per gesture class, important when interpreting class-wise metrics.

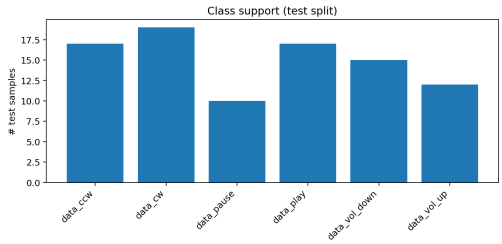

Fig. 4. Number of test samples per gesture class. Lower sample counts increase uncertainty in per-class metrics and should be noted when interpreting results.

B. Preprocessing and Augmentations

All ROI crops were squared and resized to 224×224. For experiments using ImageNet-pretrained weights, ImageNet mean/std normalisation was applied. When training from scratch, mean/std was computed on the training set. During training, on-the-fly augmentations included horizontal flip, rotation up to $\pm 15°$, brightness and contrast jitter, and $\pm 10\%$ scale/translation shifts. These augmentations help reduce overfitting and improve robustness to occlusions and lighting variation.

C. Training Protocol and Diagnostics

A ResNet-18 was fine-tuned using stochastic gradient descent (SGD) with momentum 0.9. Key hyperparameters included: initial learning rate 1e-3 with step decay, batch size 32, weight decay 1e-4, and early stopping based on validation macro-F1. The best checkpoint was exported for inference. Training and validation dynamics are shown in Fig. 5 and were used to monitor overfitting, stability, and to determine whether augmentation or regularisation needed adjustment.

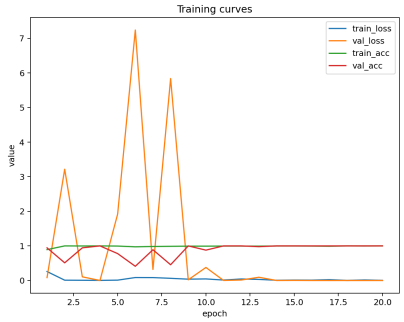

Fig. 5. Training and validation loss and accuracy across epochs. Used to detect overfitting and to select the optimal checkpoint.

D. Models and Evaluation

Two model families were evaluated:

- ResNet-18 fine-tuned on ROI crops. The final classification layer was adapted to match the number of gesture classes.
- Classical ML baselines: SVM, KNN, Decision Tree, and Random Forest, trained on handcrafted geometric and motion features from the contour pipeline.

Metrics reported include overall accuracy and macro-F1, alongside per-class PR-AUC, confusion matrices, and (where relevant) calibration metrics. Fig. 6 show key per-class diagnostics.

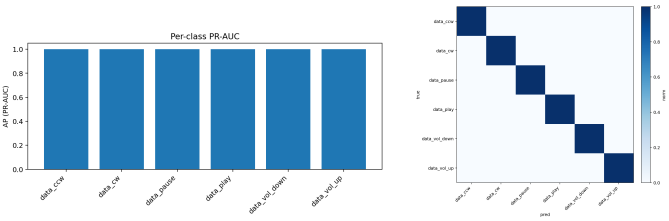

Fig. 6. Per-class PR-AUC (left) and row-normalised confusion matrix (right). These highlight detection quality and confusion trends, guiding data collection and augmentation.

Per-class PR-AUC and confusion matrix trends guide improvement priorities. Low PR-AUC or frequent confusion with specific classes suggests the need for:

- Targeted data collection: more diverse samples for underperforming classes.
- Augmentation tuning: simulate occlusions, viewpoint variation, and lighting shifts.
- Calibration or threshold tuning, especially when predictions drive downstream decisions.

VI. Results

A. Classical ML model comparison

Table III compares the classical models. Random Forest (RF) achieves perfect test accuracy and macro-F1, confirming the feature set's effectiveness.

TABLE III
Comparison of Classical Classifiers

Model	CV Acc	CV F1_macro	Test Acc	Test F1_macro
RF	0.9708	0.9708	1.0000	1.0000
KNN	0.9646	0.9642	0.9833	0.9833
SVM	0.9812	0.9812	0.9833	0.9833
DT	0.9458	0.9453	0.9583	0.9582

B. Feature-space separability

Fig. 7 shows a 2D PCA plot of the feature vectors. Clusters are well separated, which supports the classifier's high accuracy.

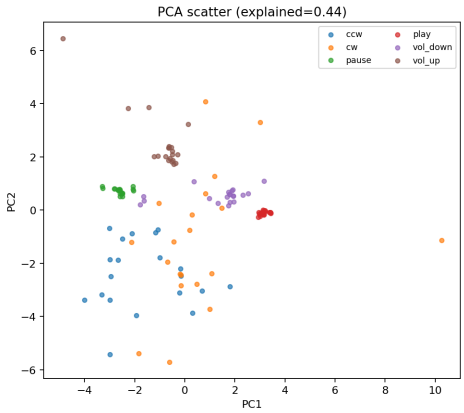

Fig. 7. 2D PCA projection of feature vectors (explained variance ≈ 0.44). Clear separation between gesture classes is observed.

C. Classification accuracy and confusion

The confusion matrix in Fig. 8 confirms high accuracy, with most predictions along the diagonal, indicating low confusion between gestures.

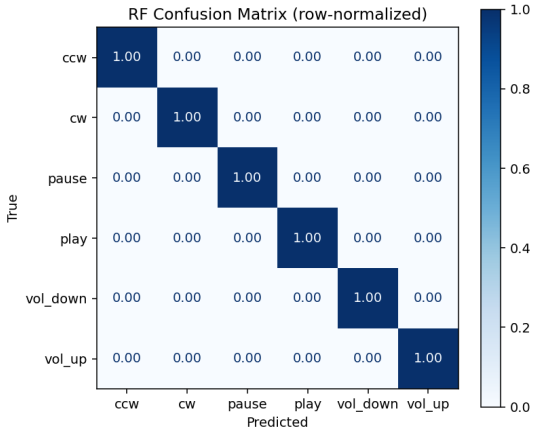

Fig. 8. Row-normalised confusion matrix (Random Forest, test split). High diagonal values indicate accurate classification.

D. Precision-Recall and ROC curves

The PR-AUC and ROC plots in Fig. 9 and Fig. 10 confirm that all classes have high recall and are linearly separable from the rest.

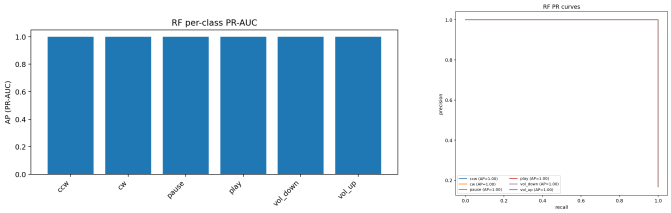

Fig. 9. Left: Per-class PR-AUC for RF. Right: PR curves inset. High scores indicate strong precision-recall balance across classes.

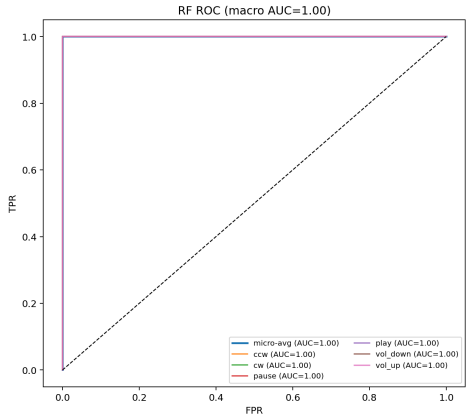

Fig. 10. ROC curves for RF (macro AUC = 1.00). All classes are clearly separable from others.

E. Calibration and reliability

As shown in Fig. 11, the RF model is slightly overconfident. Post-hoc calibration is recommended if probability thresholds will be used.

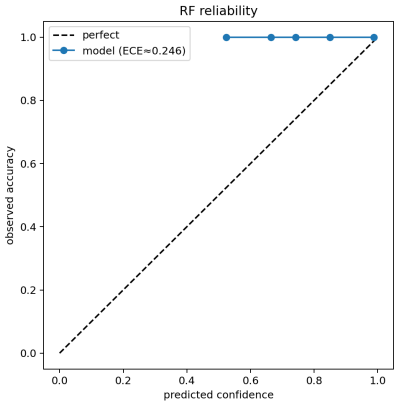

Fig. 11. Reliability diagram for RF. Expected Calibration Error (ECE) indicates mild overconfidence.

F. Generalisation and sample efficiency

Fig. 12 shows how cross-validation F1 increases and saturates around 0.97. This suggests that the current dataset is sufficient to train the model reliably.

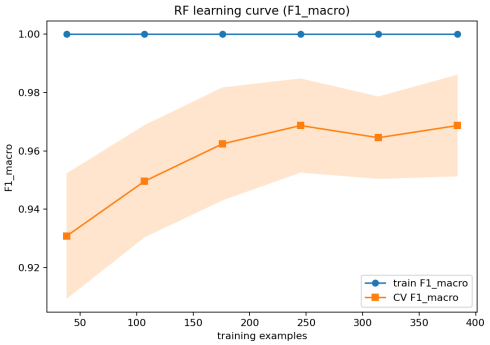

Fig. 12. Learning curve for Random Forest (macro-F1). CV score stabilises with ∼300 samples.

G. Feature importance

Model-based and permutation-based importances in Fig. 13 show that temporal rotation descriptors are most predictive for gesture discrimination.

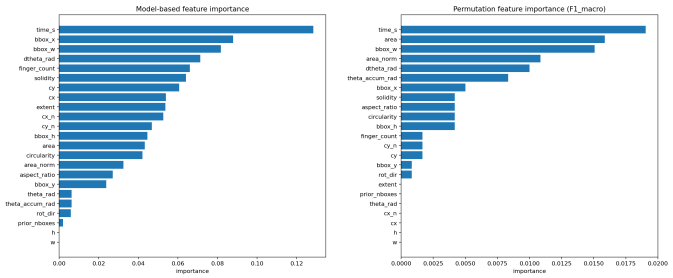

Fig. 13. Left: Feature importances from RF. Right: Permutation-based importance. Temporal and geometric features dominate.

H. Qualitative examples

Live frames in Fig. 14 illustrate successful real-time inference. Dynamic mask fusion and ROI tracking remain stable under variable poses and lighting.

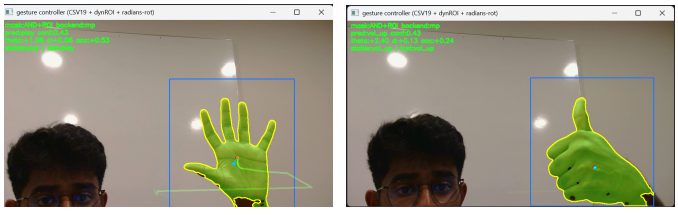

Fig. 14. Live input examples. Left: Open palm (play). Right: Thumb up (volume up). Fingertips and contours are overlaid.

VII. Discussion

The implemented hybrid pipeline mask fusion, geometric descriptors and lightweight classifiers delivers a practical balance of accuracy, explainability and runtime cost. Evaluation on the collected dataset produced consistently high per class AP and ROC values for Random Forest and SVM, and live demonstrations confirmed stable behaviour with the dynamic ROI and fused masks. The handcrafted features are data-efficient: clear class separation was achieved with a modest number of labelled samples, and common failure modes (mask loss, low hull solidity) are easy to detect and correct during deployment.

Compared to landmark-based toolkits such as MediaPipe, the hybrid approach is more transparent and easier to tune in-situ: thresholds and morphological operations can be adjusted without retraining a network [8]. Modern deep backbones such as Vision Transformers offer strong transfer performance given large, diverse datasets, but they require more compute and longer retraining cycles, which limits suitability for CPU only or embedded targets [9]. In short, the present design trades some of the raw generalisation of large CNN/transformer models for deployability, interpretability and low inference cost.

Two straightforward improvements would increase real-world reliability: apply light calibration (e.g., temperature scaling) to correct probability overconfidence, and add a failover that invokes a compact landmark or CNN model when mask quality drops. These steps preserve the pipeline's low-latency and low-data advantages while improving robustness in unconstrained settings [10].

VIII. Conclusion

The experiments demonstrate that a carefully designed classical-vision pipeline can meet practical gesture-control requirements. Random Forest on handcrafted features produced near-ideal test metrics while remaining interpretable and efficient. The system meets the project goals of speed, accuracy and modularity, and is well suited to embedded, CPU-limited deployments.

IX. Limitations and future work

- **Calibration.** The Random Forest produces accurate labels but shows probability overconfidence in some bins. Practical action: fit a temperature parameter on the validation set by minimising negative log-likelihood (temperature scaling) or use isotonic regression on a held-out calibration fold [10]. Monitor Expected Calibration Error (ECE) and aim to reduce it to a small fraction (e.g. <0.05) before using soft thresholds for control logic.
- **Lighting and skin-tone robustness.** Fixed HSV/YCrCb thresholds are brittle under extreme illumination and diverse skin tones. Practical action: add contrast/brightness augmentation (e.g. brightness $\pm 40\%$, contrast $\pm 30\%$), apply CLAHE or simple colour-constancy preprocessing (Gray-World), and evaluate adaptive thresholding (Otsu on chroma channels) or a lightweight learned skin-segmentation model. Re-evaluate per-class AP across varied lighting to verify improvement.
- **Cross-subject generalisation.** Current splits are intra-session; generalisation to new users is unmeasured. Practical action: collect a small cross-subject dataset (10–30 additional users, 50–200 examples per gesture when feasible) and run leave-one-subject-out (LOSO) evaluation. Report the drop in macro F1 and per-class AP to prioritise further data collection.
- **Temporal modelling and stability.** Reduce flicker by smoothing predictions (EMA or majority vote over 5–10 frames, $\approx$ 0.2–0.5 s) and/or add short-window temporal features (angular velocity, optical-flow). If needed, trial a lightweight temporal conv or 1D-LSTM on CSV19 sequences.
- **Latency profiling and optimisation.** Instrument end-to-end timing (time.perf_counter or cProfile) and target $\approx$33 ms/frame. Optimise hot paths (C/C++ bindings), use ONNX Runtime with dynamic quantisation, or fall back to a quantised TinyCNN for low-power devices. .

References

[1] A. Alnuaim, M. Zakariah, W. A. Hatamleh, H. Tarazi, V. Tripathi, and E. T. Amoatey, "Human-computer interaction with hand gesture recognition using resnet and mobilenet," *National Library of Medicine*, vol. 12, no. 3, pp. –, 2022, pMC ID: PMC8976610; Accessed: 2025-11-07. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35378817/

[2] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: A review of techniques," *Journal of Imaging*, vol. 6, no. 8, p. 73, 2020, accessed: 2025-11-07. [Online]. Available: https://doi.org/10.3390/jimaging6080073

[3] S. S. Rautaray and A. Agrawal, "A vision based hand gesture interface for controlling vlc media player," *International Journal of Computer Applications*, vol. 10, no. 7, 2010, accessed: 2025-11-07. [Online]. Available: https://www.ijcaonline.org/archives/volume10/number7/1495-2012/

[4] K. B. Shaik, G. Packyanathan, V. Kalist, J. M. M. Jenitha *et al.*, "Comparative study of skin color detection and segmentation in hsv and ycbcr color space," *Procedia Computer Science*, vol. 57, pp. 41–48, 2015, accessed: 2025-11-07. [Online]. Available: https://www.researchgate.net/publication/283185121_Comparative_Study_of_Skin_Color_Detection_and_Segmentation_in_HSV_and_YCbCr_Color_Space

[5] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Mountain View, CA, USA: Google Research, 2020.

[6] M. Vandersteegen, W. Reusen, K. V. Beeck, and T. Goedemé, "Low-latency hand gesture recognition with a low resolution thermal imager," in *Proceedings of the IEEE Conference Name*. IEEE, 2020, pp. –, accessed: 2025-11-07. [Online]. Available: https://ieeexplore.ieee.org/document/9150613

[7] H. Wang, B. Ru, X. Miao, Q. Gao, M. Habib, L. Liu, and S. Qiu, "Mems devices-based hand gesture recognition via wearable computing," *Micromachines*, vol. 14, no. 5, p. 947, 2023, accessed: 2025-11-07.

[8] C. Lugaresi, J. Tang, J. Hunter *et al.*, "Mediapipe: A framework for building perception pipelines," 2019, arXiv:1906.08172.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, arXiv:2010.11929.

[10] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.