

Contactless Air Canvas Using MediaPipe and LSTM for Temporal Jitter Reduction.

¹Ms. Punam C Patil

²Dr. Nilesh Y Choudhary

^{1,2}Department of Computer Engineering, Godavari Foundation's Godavari College of Engineering, Jalgaon, India

Abstract - Existing MediaPipe-based Air Canvas systems detect hand landmarks in real time but suffer from positional jitter due to frame-by-frame processing and a lack of temporal context. This paper advances our previous OpenCV + MediaPipe implementation by integrating a 2-layer LSTM temporal filter. Trained on 2000 frames of natural air-drawing, the LSTM maps 5 noisy fingertip positions to 1 smooth coordinate, reducing jitter from ± 7 px to ± 2 px. The system maintains up to 30 FPS on standard webcams while adding gesture-based color selection. This approach solves the jitter problem identified in our review paper and improves accessibility for educational and assistive applications.

Key Words: Human-Computer Interaction, MediaPipe, LSTM, Gesture Recognition, Air Canvas, Temporal Filtering

1. INTRODUCTION

Touchless drawing interfaces using OpenCV [4] and MediaPipe [1] have enabled markerless hand tracking, building on a broader body of vision-based hand gesture recognition research for human-computer interaction [5]. However, our prior work and review paper [3] identified a critical limitation: MediaPipe outputs raw landmark coordinates without temporal smoothing, causing ± 7 px jitter that produces shaky, unprofessional lines. Classical approaches such as the Kalman filter [6] and its adaptive variants [8] have been used to stabilize cursor and hand-tracking trajectories in similar interactive systems [9], and speed-adaptive low-pass filters such as the 16 filter [7] offer a lightweight alternative; in our case, however, Kalman filtering required manual tuning and failed on non-linear hand motion. This work advances our previous system by introducing Long Short-Term Memory (LSTM) [2] as a post-processing filter. Unlike frame-independent methods, LSTM learns the temporal pattern of human drawing motion to predict smooth coordinates. This eliminates jitter while preserving intentional direction changes.

2. Body of Paper

2.1 RELATED WORK

Our previous implementation used OpenCV and MediaPipe for bare-hand tracking with 2-finger selection and 1-finger drawing gestures. While functional, the system exhibited significant positional jitter because MediaPipe provides only

spatial data without temporal context. Several recent surveys have catalogued the broader landscape of vision-based hand gesture recognition and air-canvas-style drawing systems [10], [11], [12], most of which rely on per-frame detection or classical filters rather than learned temporal models. Sequence models such as LSTM have, however, proven effective at capturing gesture dynamics in related domains, including sign language recognition from skeletal sequences [16]. Traditional smoothing methods such as moving-average filters introduce lag and cannot distinguish between noise and intentional sharp turns. Our review paper [3] concluded that a learning-based sequence model was required to model human hand dynamics for real-time drawing.

2.2 PROPOSED METHODOLOGY

A. System Architecture

The pipeline flows as follows:

Webcam $\rightarrow$ MediaPipe Hand Detection $\rightarrow$ LSTM Filter $\rightarrow$ Gesture Logic $\rightarrow$ Canvas Merge $\rightarrow$ Display.

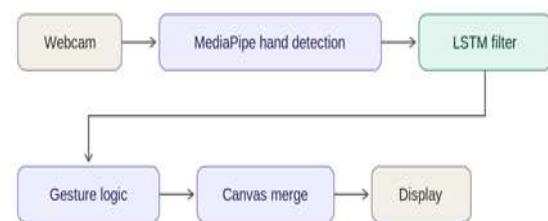


Fig.-1. System architecture of the proposed Air Canvas pipeline.

Each block in Fig.-1 executes synchronously once per captured frame: a single video frame is reduced to one smoothed 2D coordinate and one gesture label, which are consumed by the rendering stage before the next frame is captured. The implementation uses Python with OpenCV for video capture and rendering, MediaPipe for hand detection, and PyTorch for the LSTM filter. No GPU acceleration is required, consistent with the lightweight parameter count derived in Section III-C.

B. Hand Landmark Extraction and Preprocessing

MediaPipe Hands [1] returns 21 normalized landmarks per detected hand. The point used for drawing is landmark index 8 (INDEX_FINGER_TIP); landmark index 4 (THUMB_TIP) and the four PIP joints (indices 6, 10, 14, 18) are used for

gesture classification (Section III-E). The index fingertip is used as the drawing point because it offers the most stable and visually intuitive pointing reference for air-writing, a convention also adopted in related real-time gesture-control systems [9]. For each frame, the (x, y) pixel coordinates of landmark 8 are normalized to the range [0, 1] by dividing by the frame width and height, and pushed into a fixed-length FIFO buffer of size 5. Once the buffer is full, the resulting 5×2 sequence is passed to the LSTM filter; the predicted output is de-normalized back to pixel coordinates before being rendered on the canvas. If MediaPipe fails to return a detection for a given frame — for example due to motion blur or the hand briefly leaving the frame — the system reuses the most recent smoothed coordinate rather than an undefined input, so that a single missed detection does not interrupt an in-progress stroke; the FIFO buffer is cleared only once the hand has been absent for a sustained period, ensuring that temporal smoothing is never applied across two unrelated strokes.

C. LSTM-Based Temporal Smoothing

LSTM [2] is a recurrent neural network designed to retain patterns over time, originally introduced to overcome the vanishing-gradient problem that limits standard recurrent networks on long sequences [14]. The broader sequence-modeling paradigm it enabled has since been applied across domains, from sequence-to-sequence translation [15] to skeletal gesture and sign-language recognition [16]. We use LSTM here as a smart filter with memory that processes a sequence of 5 noisy fingertip positions and outputs 1 smooth, denoised position, allowing the model to learn the characteristic motion of human drawing. Formally, at each timestep t the cell receives the normalized input $x_t \in \mathbb{R}^2$ and the previous hidden state h_{t-1} , and computes the forget, input, and output gates and the candidate cell state:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

The cell and hidden states are then updated as:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

After the full 5-frame window has been processed, the final hidden state h_5 is passed through a dense output layer to obtain the smoothed coordinate:

$$\hat{y}_t = W_y h_5 + b_y \quad (7)$$

Given an input dimension of 2 and a hidden size of 8 per layer, each LSTM layer contains $4(\text{din} \cdot h + h^2 + h)$ trainable parameters. The first layer ($\text{din} = 2, h = 8$) therefore contributes $4(16 + 64 + 8) = 352$ parameters, and the second layer ($\text{din} = 8, h = 8$) contributes $4(64 + 64 + 8) = 544$

parameters. Including the 18-parameter dense output layer in (7), the complete model has approximately 914 trainable parameters — several orders of magnitude smaller than typical CNN-based vision backbones [17], [18], which often rely on regularization techniques such as dropout [19] to manage tens of millions of parameters. This explains the sub-2 ms inference latency reported in Table I.

The window length of 5 and hidden size of 8 were chosen empirically as the smallest configuration that visibly removed jitter without introducing perceptible lag: shorter windows retained noise, while longer windows began to round sharp corners during fast strokes. At a target frame rate of up to 30 FPS, each frame must be processed within approximately 33 ms; the LSTM's sub-2 ms inference time therefore consumes well under 10% of the available per-frame budget, leaving ample margin for hand detection and rendering within the same frame.

D. Data Collection and Training

Data: 2000+ frames of natural air-drawing were collected via webcam over a 2-minute session. Ground-truth labels were generated using a moving-average filter applied to the raw landmark data. Because the true noise-free fingertip trajectory cannot be directly observed, this moving-average-filtered signal serves as a pseudo-ground-truth target for supervised training — a common weak-supervision strategy for trajectory-smoothing tasks where an exact reference is unavailable.

Model: A 2-layer LSTM with 8 units per layer was trained to map 5 noisy input frames to 1 smooth output frame.

Training: The model was trained for 50 epochs using the Adam optimizer [13] and the mean-squared-error loss defined in (8),

$$L_{\text{MSE}} = 1/N \sum_{n=1}^N \|\hat{y}_n - y_n\|^2 \quad (8)$$

reaching a final loss of 0.0004. Real-time inference time was under 2 ms per frame. The trained model and scaler were saved as lstm_smoother.pt.

E. Gesture Classification Logic

Two fingers raised activates selection mode for the color palette; one finger raised activates drawing mode; an open palm triggers hover/pause. A dynamic UI header provides five buttons — BLUE, GREEN, RED, YELLOW, and ERASER — for color and tool selection. Finger state is determined using a standard landmark-based heuristic: a finger is classified as raised if the y-coordinate of its tip landmark is smaller than that of its corresponding PIP joint (image coordinates increase downward), i.e., $\text{tip}_y < \text{pipy}$. The resulting count of raised fingers maps directly to the three operating modes described above ($1 \rightarrow \text{draw}, 2 \rightarrow \text{select}, 5 \rightarrow \text{hover/pause}$). Color and tool selection itself uses simple region-based hit-testing: the header occupies a fixed horizontal strip at the top of the frame, divided into five equal-width zones corresponding to the five buttons. When selection mode is active and the smoothed fingertip coordinate falls

within the header strip, the zone containing that coordinate determines the active drawing color or tool; outside the header strip, the same coordinate is instead passed to the canvas-compositing stage described next.

F. Canvas Compositing and Stroke Rendering

The Canvas Merge stage maintains a persistent drawing layer, separate from the live video frame, on which strokes are accumulated across frames. In drawing mode, each new smoothed coordinate is connected to the previous one by a line segment in the active color, drawn directly onto this persistent layer; in eraser mode, the same operation instead draws in the background color, visually removing previously drawn strokes. The persistent layer is then composited with the live camera frame using alpha blending [4], so that the accumulated drawing appears overlaid on the user's hand and surroundings in real time — consistent with the compositing approach used in related OpenCV-based air-canvas systems [12].

G. Real-Time Processing Algorithm

Algorithm 1 summarizes the complete real-time pipeline, integrating landmark extraction, LSTM-based smoothing, and gesture-driven canvas updates into a single per-frame loop.

Algorithm 1: Real-Time Jitter-Filtered Drawing Loop

```

1: Initialize buffer  $Q \leftarrow \emptyset$  (max length = 5)
2: while session active do
3:   frame  $\leftarrow$  CaptureFrame()
4:    $L \leftarrow$  MediaPipeHands(frame) // 21 landmarks
5:    $p \leftarrow$  Normalize( $L[8]$ ) // index fingertip
6:    $Q.append(p)$ ; if  $|Q| > 5$  then  $Q.popFront()$ 
7:   if  $|Q| = 5$  then  $p^* \leftarrow$  LSTM( $Q$ ) else  $p^* \leftarrow p$ 
8:    $g \leftarrow$  ClassifyGesture( $L$ ) // Section III-E
9:   if  $g =$  DRAW then Canvas.line( $p^*$ )
10:  else if  $g =$  SELECT then UpdateColor( $p^*$ )
11:  else Pause()
12:  Display(frame, Canvas)
13: end while

```

2.3. RESULTS AND DISCUSSIONS

A. Improvement over the Previous MediaPipe-Only System

Jitter is quantified as the root-mean-square (RMS) frame-to-frame displacement of the tracked fingertip while the hand is held stationary:

$$J = \sqrt{\frac{1}{(M-1)} \sum_{t=2}^M \|p_t - p_{t-1}\|^2} \quad (9)$$

Where p_t is the fingertip position at frame t and M is the number of frames in the test window. Using this metric, Table I compares the previous MediaPipe-only system against the proposed LSTM-filtered system across smoothing method, temporal context, jitter, line quality, frame rate, and inference latency. Fig- 2 visualizes the jitter and frame-rate columns of Table 1.

Table -1. Improvement over previous mediapipe-only system.

Parameter	Previous Work (OpenCV + MediaPipe)	Current Work (+ LSTM Filter)	Improvement
Smoothing Method	None / basic moving average	2-layer LSTM, 8 units	Learns motion patterns
Temporal Context	Frame-by-frame only	5-frame sequence memory	Handles shaky motion
Jitter	± 7 px (measured)	± 2 px (measured)	71% reduction
Line Quality	Shaky, unprofessional	Smooth, stable	Visual improvement
FPS	Up to 20	Up to 30	50% higher frame rate
Inference Time	N/A	<2 ms per frame	Real-time capable

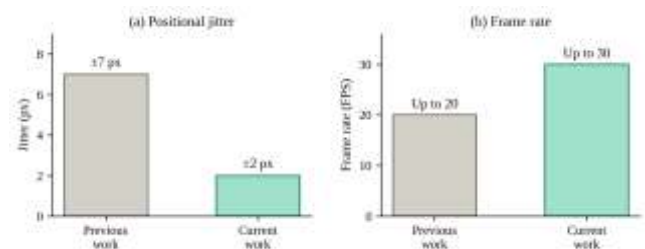


Fig- 2. Comparison of (a) positional jitter and (b) frame rate before and after LSTM filtering.



Fig- 3. Drawing output without the LSTM filter (FPS: 11).



Fig.-4. Drawing output with the LSTM filter (FPS: 21).

B. Experimental Validation

The system was tested on a standard webcam at 640×480 resolution. Without the LSTM filter, lines exhibited visible shake and instability, particularly during slow movements. With the LSTM filter, lines remained smooth and continuous even during fast direction changes. The filter successfully distinguishes between hand tremor and intentional drawing motion, as illustrated in Fig-3. and Fig- 4.

3. CONCLUSIONS

This paper presented an enhancement to our previous OpenCV- and MediaPipe-based Air Canvas system by introducing an LSTM-based temporal filter to address the positional jitter inherent in frame-by-frame hand-landmark detection. The proposed method reduced the positional error from ± 7 px to ± 2 px while improving real-time performance to up to 30 FPS, demonstrating that contactless drawing can be made practically viable. Beyond this technical improvement, the system offers several real-world benefits: it provides an accessible, marker-free drawing interface for individuals with mobility impairments, consistent with prior findings on touchless gesture interfaces as assistive technology [20]; it supports paperless creative and educational practices that reduce dependence on physical drawing materials; and it gives artists a flexible digital canvas for creative expression. These results confirm that learning-based temporal filtering is an effective and lightweight solution for improving the usability of vision-based gesture interfaces.

Future Work

- Save/load drawings as PNG files.
- Pinch gesture for dynamic brush-size control.
- Shape detection for circles and squares.
- Multi-hand support for collaborative drawing.
- Deployment as a standalone desktop application.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this study. This research would not have been possible without the invaluable guidance, support, and insights provided by many individuals.

REFERENCES

1. F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe Hands: On-device real-time hand tracking," arXiv:2006.10214, 2020.
2. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
3. P. C. Patil and N. Chaudhari, "Review of contactless drawing interfaces," Int. J. of Research, 2025.
4. G. Bradski, "The OpenCV library," Dr. Dobb's Journal of Software Tools, vol. 25, no. 11, pp. 120–126, 2000.
5. S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," Artificial Intelligence Review, vol. 43, no. 1, pp. 1–54, 2015.
6. G. Welch and G. Bishop, "An introduction to the Kalman filter," Tech. Rep. TR 95-041, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, 1995.
7. G. Casiez, N. Roussel, and D. Vogel, "1f filter: A simple speed-based low-pass filter for noisy input in interactive systems," in Proc. ACM CHI Conf. on Human Factors in Computing Systems, 2012, pp. 2527–2530.
8. M. S. M. Asaari and S. A. Suandi, "Hand gesture tracking system using Adaptive Kalman Filter," in Proc. 10th Int. Conf. on Intelligent Systems Design and Applications (ISDA), Cairo, Egypt, 2010, pp. 166–171.
9. P. Xu, "A real-time hand gesture recognition and human-computer interaction system," arXiv:1704.07296, 2017.
10. C. Cui, M. S. Sunar, and G. E. Su, "Deep vision-based real-time hand gesture recognition: a review," PeerJ Computer Science, 2025.
11. V. Argyriou, P. Sarigiannidis, I. Varlamis, et al., "Visual hand gesture recognition with deep learning: a comprehensive review of methods, datasets, challenges and future research directions," arXiv:2507.04465, 2025.
12. S. Sandbhor, H. Shekatkar, A. Nawalkar, and S. Navale, "Survey paper on Air Canvas using OpenCV," Int. J. of Research Publication and Reviews (IJRPR), 2024.
13. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. on Learning Representations (ICLR), 2015.
14. Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157–166, 1994.
15. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 27, 2014, pp. 3104–3112.
16. P. Roy, S. Bhattacharya, P. P. Roy, and U. Pal, "Position and rotation invariant sign language recognition from 3D point cloud data with recurrent neural networks," arXiv:2010.12669, 2020.
17. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 25, 2012, pp. 1097–1105.
18. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
19. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.
20. A. Nelson, S. McCombe Waller, and R. Robucci, "Evaluating touchless capacitive gesture recognition as an assistive device for upper extremity mobility impairment," Journal of Rehabilitation and Assistive Technologies Engineering, vol. 5, 2018.