AIRCANVAS

Project Reference No.: *49S_BE_0444*

College : S G Balekundri Institute of Technology, Belagavi.
Branch : Department of Computer Science and Engineering
Guide(s) : Prof. Ramesh Sattigeri
Prof. Kshitij Sheth
Student(s) : Ms. Nandini Buchadi
Ms. Nandita Patil
Mr. Nayan Mathapati
Mr. Shivanand Kori

Keywords:

Gesture Recognition, Computer Vision, CNN, Virtual Drawing, MediaPipe.

Introduction:

1. Human–computer interaction (HCI) has undergone a major transformation, shifting from traditional devices like keyboards and touchscreens to more natural, gesture-based interfaces. With rapid developments in artificial intelligence, computer vision, and real-time gesture recognition, modern systems increasingly prioritize intuitive and accessible user experiences. In this direction, **Aircanvas** presents a creative, touch-free drawing system that allows users to sketch, write, or annotate in the air using only hand gestures [1]. By combining real-time hand tracking with deep learning–based gesture classification, the system translates simple hand movements into digital strokes on a virtual canvas.
2. Aircanvas eliminates the need for physical drawing tools and offers a natural way of interacting with digital content. It integrates MediaPipe for hand landmark detection, CNN models for gesture prediction, Python for processing, and Flask for web deployment. The interface, built using HTML and CSS, runs on any modern browser, making the system portable and easy to access [3].
3. The system continuously captures video frames, extracts hand landmarks such as finger tips and joints, and feeds them into a trained CNN model. The model identifies user gestures—like drawing, erasing, or changing colours—in real time. These predictions control the virtual brush, enabling users to create colourful digital strokes by simply moving their hands.
4. The project is motivated by the need for intuitive, accessible, and device-free drawing tools. Gesture-based systems overcome the limitations of styluses or touchscreens, making Aircanvas useful for digital artists, educators, individuals with motor challenges, and anyone seeking a creative outlet. It also supports educational and rehabilitative applications such as handwriting practice and motor-skill training [1].
5. Overall, Aircanvas showcases how AI and computer vision can transform simple gestures into powerful digital expressions, paving the way for future touch-free interactive platforms.

Objectives:

1. The primary objective of this project is to design and implement Aircanvas, an AI-based, hands-free virtual drawing system that allows users to draw in the air using gestures and voice commands. The system aims to accurately track hand movements in real time using MediaPipe and convert fingertip motion into smooth digital strokes on a virtual canvas.
2. Another key objective is to develop a reliable gesture recognition mechanism using a CNN model to perform actions such as drawing, erasing, changing colours, and clearing the canvas. The project also focuses on creating a touch-free drawing experience with adjustable brush features, ensuring smooth and stable output without delays.
3. In addition, the system integrates speech-to-text for voice commands and text-to-speech for audio feedback, improving accessibility and user interaction. A Flask backend is used to enable real-time communication and web-based deployment.
4. The project further aims to achieve low-latency performance, provide an inclusive user experience for different users (including those with disabilities), and maintain a modular design that supports future enhancements like multi-finger input and artwork saving.

METHODOLOGY:

1. The proposed system is designed to allow users to write in the air using their fingers, which is captured through a webcam and converted into digital text. Along with text output, the system can also provide audio feedback, making the interaction more intuitive and user-friendly.
2. There are two main approaches used in this system. The first is a TrOCR-based word recognition method, implemented in *final.py*, which uses a pre-trained model and does not require any separate dataset. The second is an optional CNN-based character recognition method, implemented in *air.py*, which recognizes individual characters using a trained dataset of handwritten letters and numbers.
3. The working of the system starts with capturing live video from the webcam. This input is processed using MediaPipe, which detects hand movements and identifies key landmarks. Based on these landmarks, the system understands gestures. For example, when the user pinches their thumb and index finger together, the system starts drawing on a virtual canvas, and when the gesture is released, drawing stops.
4. Once the drawing is complete, the system processes the input and sends it for recognition. For word-level recognition, it uses microsoft/trocr-base-handwritten, which is capable of converting handwritten words into text. In the CNN approach, each character is resized into a 28×28 grayscale image and classified accordingly.
5. Before recognition, the drawn content undergoes several preprocessing steps such as cropping the text area, removing noise, adjusting size, and converting it into a suitable format. This helps improve the accuracy of the model.
6. After recognition, the output text is displayed on the screen and stored in a session buffer. The system can also read the recognized text aloud using pyttsx3, which enhances accessibility.

7. The backend of the system is built using Flask, which manages different routes for video streaming, canvas display, text retrieval, and user controls. It also ensures smooth performance using background processing and thread-safe operations.
8. The CNN model is trained on a dataset containing 62 classes, including digits, uppercase, and lowercase letters. The training process involves preprocessing, model learning, validation, and saving the trained model for later use.
9. To evaluate the system, metrics like accuracy, error rates (CER/WER), and response time are considered. The system is tested under different lighting conditions, backgrounds, and writing speeds to check its robustness.
10. However, the system has some limitations. It may struggle with very small or faint handwriting, and gesture recognition is still basic. There can also be compatibility issues with certain software dependencies.
11. In the future, the system can be improved by using more advanced gesture recognition techniques, better preprocessing methods, and language models for correcting errors. It can also include user-specific calibration to adapt to different writing styles.

Result and Conclusions:

1. The AirCanvas system produced highly promising results, proving that gesture- and voice-based drawing is both practical and effective. The integration of MediaPipe enabled accurate real-time hand tracking, consistently detecting 21 landmarks and ensuring smooth fingertip movement for drawing.
2. The CNN-based gesture recognition model performed well, correctly identifying actions such as drawing, erasing, clearing the canvas, and changing colours with good accuracy. The system responded reliably even with slight variations in gestures, showing strong generalization.
3. The drawing module worked smoothly, producing continuous strokes without lag or jitter, while erase and clear functions operated instantly. Voice interaction also showed positive results, where speech-to-text accurately recognized commands and text-to-speech provided clear feedback to the user.
4. The Flask backend ensured seamless communication between all components, with no major delays during testing. Overall, the system maintained real-time performance and a responsive user experience.
5. In conclusion, AirCanvas successfully demonstrates how computer vision, machine learning, and speech technologies can be combined to create a touch-free drawing system. It is reliable, accessible, cost-effective, and works on basic hardware like a webcam and browser. The project meets its objectives and shows strong potential for future improvements and real-world applications.

Project Outcome & Industry Relevance:

Delivers a working mid-air writing and gesture interface using only a webcam and standard PC. Real-time hand tracking, drawing, and word recognition with on-screen updates and optional audio feedback.Two recognition paths: TrOCR for word-level accuracy; optional 62-class CNN for character-level deployment. Robust preprocessing (binarization, denoise, morphology, aspect-ratio normalization) improves OCR quality. Responsive web UI streams live camera and canvas; supports mode switching and canvas clearing. Touchless interaction suits hygiene-critical contexts (healthcare kiosks, public terminals). Accessibility aid for hands-free or pen-less input in education and assistive tech. Applicable to AR/VR annotation, digital whiteboarding, and smart classrooms. Works on commodity hardware (CPU-only), avoiding special sensors or gloves. On-device processing preserves privacy; no cloud dependency for inference. Modular architecture enables swapping models, thresholds, and UX policies easily. Demonstrates low-latency HCI pipeline suitable for embedded/edge deployments.

Working Model Vs. Simulation/Study:

Working model: live webcam capture, real-time Mediapipe landmarking, drawing, and on-device OCR. No synthetic frames or mocked landmarks; end-to-end tested via browser UI. Recognition runs on actual pre-trained models (TrOCR and/or CNN), not analytical placeholders. Optional dataset is used for training the CNN variant; runtime in final.py does not depend on a local dataset. System behavior validated under typical lighting and user motion, not only in offline experiments.

Project Outcomes and Learnings:

Preprocessing quality is decisive for OCR accuracy; adaptive thresholding and padding matter. TrOCR offers strong word-level results but benefits from clean foreground isolation; CNN suits per-character UIs. Version alignment (protobuf, Mediapipe, TensorFlow) is critical on Windows; pinning resolves many runtime issues. UX tuning (pinch threshold, smoothing, idle timeout) directly impacts usability and recognition reliability. Threaded inference and MJPEG streaming keep the UI responsive under CPU-only conditions. Clear separation of modes, state, and routes simplifies maintenance and future model swaps.

References:

[1] G. Heimerl, V. Vukovic, and G. Kovacs, “Real-time hand gesture recognition using deep learning approaches,” *IEEE Access*, vol. 8, pp. 152768–152780, 2020

[2] F. Zhang, A. Bazarevsky, and Y. Vakunov, “MediaPipe Hands: On-device real-time hand tracking,”*GoogleAIBlog*,2020.

[3] Zhang, Y., “Hand Gesture Recognition Using Deep Learning Techniques,” Journal of Computer Vision and Pattern Analysis, 2022.

[4] Nguyen, T., “Convolutional Neural Networks for Image Based Gesture Classification,” International Journal of Machine Learning, 2023.
[5] Singh, K., “Real-Time Gesture Recognition Using Python Kumar, Y., “Web-Based Visualization Techniques Using HTML5 Canvas,” Web Technologies and Applications Journal, 2021.
[6] Flask Documentation Team, “Flask: A Lightweight Python Web Framework for APIs,” Flask Official Documentation, 2023.
[7] OpenCV Organization, “Real-Time Computer Vision for Human–Computer Interaction,” OpenCV Documentation, 2022.
[8] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in *Proc. 12th USENIXOSDI*,2016.

Future Scope:

- **Enhanced Gesture Recognition:**
  Future versions can include more complex gestures such as zoom, rotate, multi-finger control, and shape-specific gestures for advanced drawing features.
- **Improved Accuracy in Varying Conditions:**
  By training the model with larger and more diverse datasets, the system can perform better under low lighting, fast hand movements, and partial occlusions.
- **Advanced Deep Learning Models:**
  Implementation of LSTM, GRU, or Transformer-based models can help capture temporal motion patterns, improving recognition accuracy for continuous gestures.
- **Noise-Resistant Voice Commands:**
  Integrating noise-cancellation techniques or keyword-spotting models can improve Speech-to- Text accuracy in noisy environments.
- **More Natural TTS Output:**
  Using advanced neural TTS systems can generate more human-like voice responses, improving accessibility for visually impaired users.
- **Cloud-Based Collaboration:**
  Multiple users can draw together in real time through cloud synchronization, enabling features like remote classrooms, teamwork, and online art sessions.
- **Saving and Exporting Features:**
  The system can include options to save drawings, export as images, or store artwork in the cloud for later access.

- **Mobile and Tablet Support:**
  Developing a mobile-friendly version would make the system more accessible and expand its usability beyond desktops and laptops.
- **AR/VR Integration:**
  Implementing AirCanvas in augmented reality or virtual reality environments can enable 3D gesture-based drawing and immersive creative experiences.
- **IoT Integration:**
  The gesture recognition pipeline can be extended to control IoT devices, enabling gesture-based automation in smart homes or classrooms.
- **User Profiles & Personalization:**
  Adding user accounts, customizable gesture sets, and personalized settings can improve user experience and adaptability.
- **AI-Assisted Drawing Tools:**
  Future versions may include stroke prediction, auto-smoothing, and intelligent shape correction using advanced AI models.