AirSketch: A Touchless Gesture-Based Drawing System

Manasa Kumari
Stony Brook University
manasa.kumari@stonybrook.edu

Abstract

We present AirSketch, a touchless drawing system that allows users to sketch in mid-air using hand gestures captured from a standard webcam. We built the system using MediaPipe Hands for real-time hand landmark tracking and OpenCV for rendering and image processing. AirSketch supports freehand drawing, erasing, tool selection, geometric shape drawing, and contour-based shape recognition through a set of intuitive gestures. We designed the system as a research prototype to study the feasibility and usability of gesture-based drawing using commodity hardware. In this paper, we describe the system design, implemented features, and observations from an informal user evaluation.

1 Introduction

Most digital drawing tools rely on contact-based input devices such as mice, styluses, or touchscreens. These devices assume a physical surface and a seated interaction posture. In practice, this limits drawing interaction in settings such as live presentations, standing interaction, or environments without dedicated input hardware.

Recent progress in real-time hand tracking using RGB cameras makes it possible to design touchless interaction techniques using only a webcam. In this project, we explore whether mid-air hand gestures can support expressive yet lightweight drawing interactions. Our goal is not to replace traditional drawing tools, but to investigate gesture-based drawing as a flexible, low-setup alternative for short and opportunistic use.

We designed AirSketch with a focus on intuitive gesture mappings, clear visual feedback, and mechanisms to reduce accidental activation, such as gesture gating and cooldowns.

2 Related Work

We position AirSketch in the broader evolution of human input for drawing and sketching, moving from contact-based devices toward camera-based, touchless interaction. Rather than treating related work as a list of papers, we summarize the progression of the underlying idea and the technical shifts that made systems like ours feasible.

2.1 From Contact-Based Drawing to Direct Manipulation (1960s–2000s)

Early digital drawing systems were built around *contact-based* pointing devices (light pens, mice, and later styluses and touchscreens). The dominant interaction model assumed a stable physical surface, where users could make precise strokes with low fatigue. Over time, direct manipulation became a standard paradigm: users expected immediate visual feedback, smooth strokes, undo/erase operations, and tool palettes (colors, brush sizes, shape tools). These conventions strongly influenced our UI choices in AirSketch (e.g., a paint-style menu bar and persistent system feedback).

2.2 Touchless Interaction with Specialized Sensors (mid-2000s–2010s)

As sensing hardware improved, research shifted toward *touchless* input using specialized sensors. Depth cameras and infrared tracking enabled mid-air gestures for command execution and spatial manipulation. This era made mid-air interaction practical, but typically required dedicated hardware and setup. Researchers also repeatedly observed important constraints: mid-air input can be less precise than contact input, fatigue increases over time, and unintended activations are common unless the gesture vocabulary is carefully designed and paired with clear feedback. These findings directly motivated our design choices in AirSketch: we keep the gesture set small, add gesture gating (writing pose), and use cooldowns for destructive actions such as clearing the canvas.

2.3 Commodity-RGB Hand Tracking and Practical Mid-Air Systems (late-2010s–present)

More recently, hand tracking has moved from specialized sensors to commodity RGB cameras. Real-time hand landmark detection frameworks made it feasible to build touchless systems that run on laptops with a standard webcam. This shift changes the trade-off space: accessibility and low setup improve, but robustness becomes sensitive to lighting, background clutter, occlusion, and camera distance. AirSketch lives in this modern class of systems: we rely on webcam-only tracking, and we address instability using smoothing (One Euro Filter), temporal logic (hold times / motion history), and explicit UI feedback to communicate system state.

2.4 Touchless Drawing as Continuous Input

A key distinction in touchless interfaces is whether gestures are treated as *discrete commands* (e.g., swipe to trigger an action) or as *continuous input* (e.g., controlling a cursor trajectory to produce strokes). Many gesture interfaces focus primarily on command recognition; drawing requires stable, continuous tracking and strong error tolerance because small tracking noise becomes visible as jittery strokes.

We designed AirSketch specifically around the requirements of continuous creative input. We combine (1) stable cursor motion using smoothing, (2) mode gating to reduce false positives while drawing, (3) tool selection through pinch interactions with a familiar menu bar, and (4) lightweight contour-based shape recognition to bridge freehand sketches and structured geometry. This combination places AirSketch at the intersection of modern webcam-based hand tracking and drawing-focused interaction design.

3 System Design

AirSketch follows a real-time pipeline architecture consisting of hand tracking, gesture recognition, and rendering. Each webcam frame is processed to extract hand landmarks, which are then interpreted into drawing and control actions that update a persistent canvas and user interface.

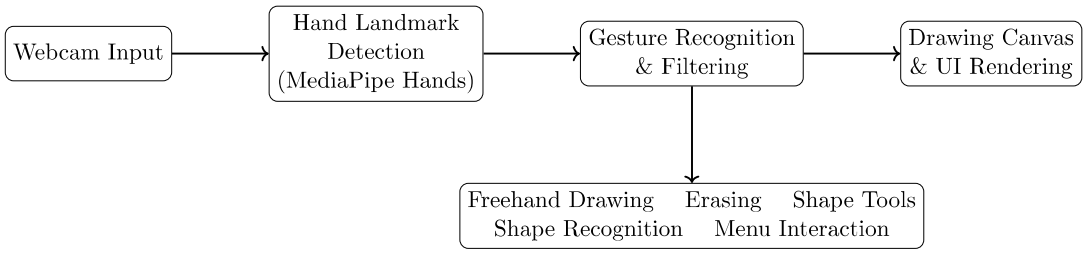

Figure 1: Overview of the AirSketch system pipeline.

3.1 Gesture Set

We designed a small set of distinct and easily differentiable gestures, summarized in Table 1.

Table 1: Gesture and keyboard controls supported in AirSketch

Input	**Action**	**Description**
Index finger up	Freehand drawing	Draws continuous strokes on the canvas using the fingertip trajectory.
Two fingers (peace sign)	Eraser mode	Erases content using a circular mask based on the current brush size.
Closed fist	Pause / resume	Temporarily pauses drawing to prevent unintended strokes; resumes on release.
Swipe left	Clear canvas	Clears the entire drawing canvas; protected by motion thresholds and cooldowns.
Pinch (thumb + index)	Menu interaction	Selects colors, brush sizes, tools, and shape modes from the top menu bar.
Thumbs up	Shape recognition	Triggers contour-based shape recognition on the current drawing; rate-limited to avoid repeated activation.
`S` key	Save canvas	Saves the current canvas to a PNG file with an automatic timestamp.
`Q` key	Quit application	Exits the application safely and releases camera resources.
`H` key	Help overlay	Toggles an on-screen gesture guide describing available controls.

Figure 2 shows the on-screen gesture reference used to communicate available controls to users.

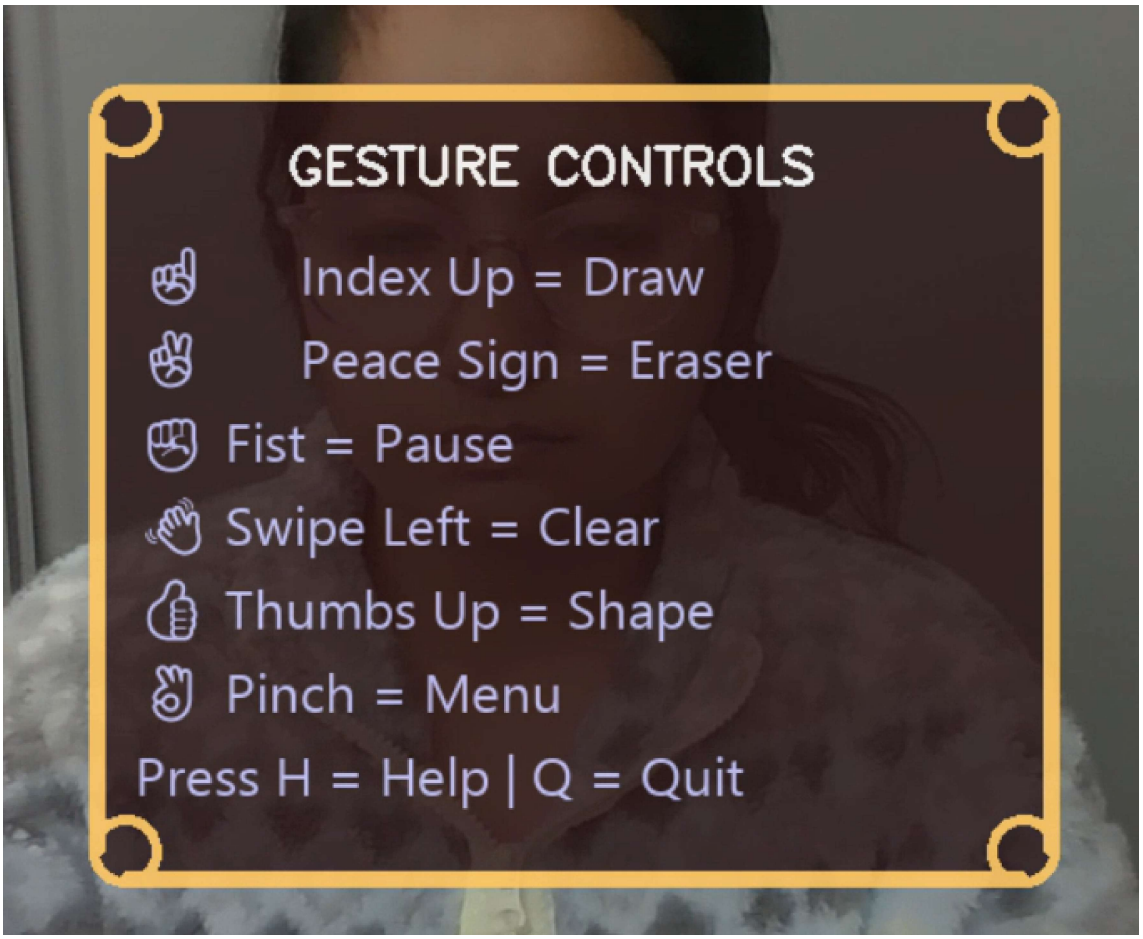

Figure 2: The interface communicates the mapping between hand gestures and system actions to support learnability and reduce user errors.

To reduce false positives, we gate drawing actions using a writing-pose detector and apply cooldowns for destructive actions such as clearing the canvas and triggering shape recognition.

3.2 Visual Feedback

We provide continuous visual feedback through a paint-style top menu bar and transient on-screen messages. These HUD messages confirm actions such as mode changes, color selection, canvas clearing, and shape recognition, helping users understand system state at all times.

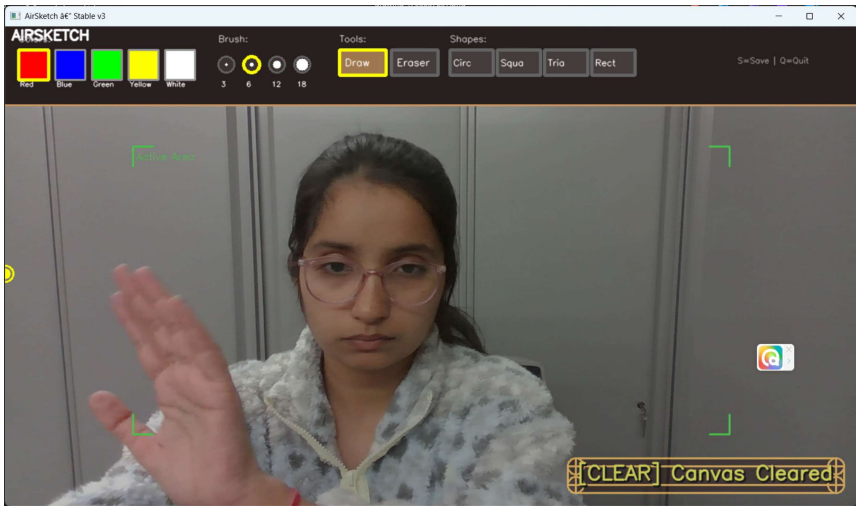

Figure 3: Depicting clear canvas feature.

4 Implementation

4.1 Software Architecture

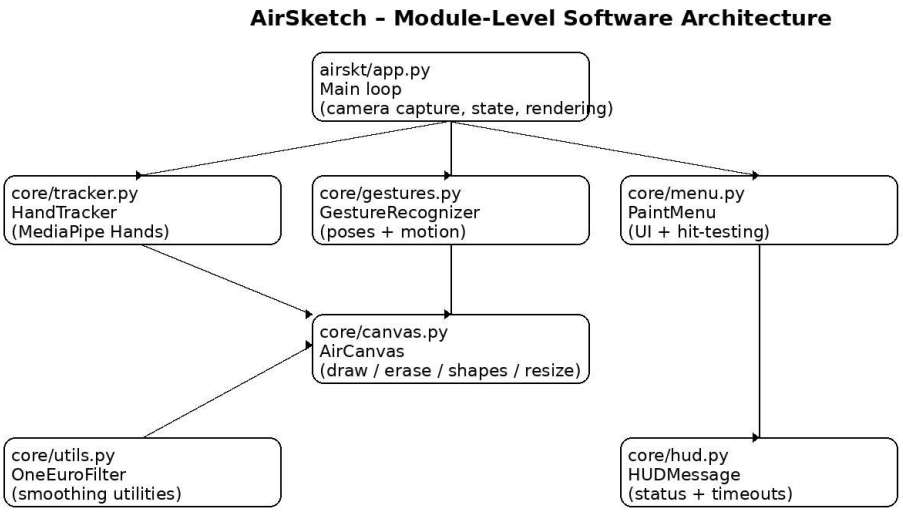

Figure 4: Module-level software architecture of AirSketch. The main application loop orchestrates hand tracking, gesture recognition, canvas operations, menu interaction, and visual feedback through modular components.

The system is organized as a real-time interaction pipeline, where each video frame is processed through hand tracking, gesture recognition, and rendering stages.

Gesture interpretation acts as a central decision point, mapping detected hand poses and motions to drawing actions, menu interactions, and shape-based operations. Rendering and

feedback are handled continuously to ensure low-latency visual response and clear communication of system state to the user.

4.2 Implemented Features

AirSketch provides a set of integrated interaction and drawing features designed to support touchless sketching while minimizing unintended activation:

- **Freehand Drawing:** Real-time mid-air drawing using the index fingertip, with cursor stabilization via a One Euro Filter to reduce jitter while preserving responsiveness.
- **Erasing:** Content removal through a two-finger gesture or explicit eraser tool mode, using a circular mask proportional to the selected brush size.
- **Menu Interaction:** Pinch-based interaction with a paint-style menu bar for selecting colors, brush sizes, tools, and geometric shapes.
- **Geometric Shape Tools:** Explicit shape drawing modes (circle, square, triangle, rectangle) that allow users to place shapes using a pinch-and-release gesture and resize them via pinch-based vertical motion.
- **Shape Recognition:** Automatic contour-based recognition of freehand sketches, classifying geometric shapes with confidence estimation after a brief pause in drawing.
- **Keyboard Fallbacks:** Optional keyboard controls for saving the canvas (`S`), quitting the application (`Q`), and toggling the gesture help overlay (`H`).

To improve robustness, all interaction modes are governed by writing-pose detection and gesture-specific cooldowns, which reduce false positives and prevent accidental activation of destructive actions.

4.3 Hand Tracking and Smoothing

We use MediaPipe Hands to extract 21 hand landmarks per frame. Fingertip positions are smoothed using a One Euro Filter to reduce jitter while maintaining responsiveness. This smoothing is critical for producing stable strokes during freehand drawing.

4.4 Canvas and Drawing Features

Drawing operations are performed on a persistent high-resolution canvas. Freehand drawing is implemented by interpolating line segments between successive fingertip positions. Erasing is handled by drawing circular masks using the current brush size.

The system supports saving the canvas to a PNG file using a keyboard shortcut (`S`) and exiting the application using `Q`.

4.5 Shape Tools and Recognition

In addition to freehand drawing, we implemented explicit shape tools that allow users to draw circles, triangles, squares, and rectangles by selecting a shape from the menu and placing it on the canvas using a pinch-and-release gesture. Shapes can be resized using pinch-based vertical motion.Figure 5 shows the interface elements used to select tools and geometric shapes during drawing.

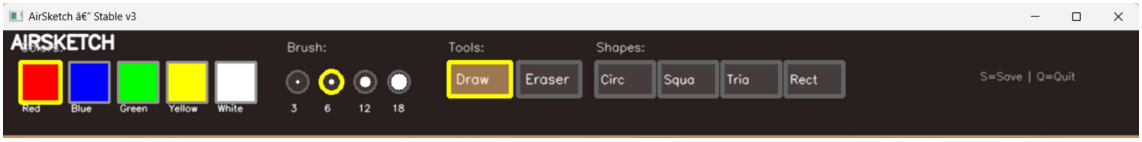

Figure 5: User interface of AirSketch showing shape detection based on brush strokes and drawings.

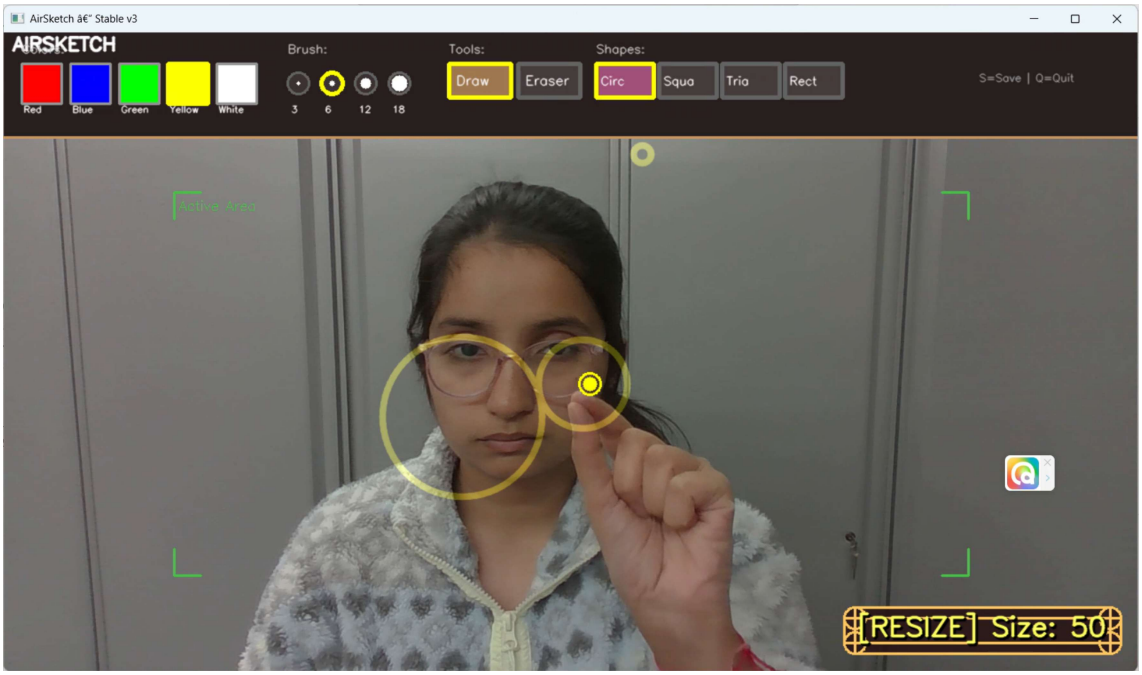

Figure 6: User interface of AirSketch showing geometric shape tool used for explicit shape drawing.

We also implemented automatic shape recognition using computer vision techniques. The recognition pipeline uses contour detection, multiple thresholding strategies, morphological cleanup, and polygon approximation to classify geometric shapes with confidence levels.Figure 7 illustrates an example of mid-air drawing and the corresponding shape recognition output.

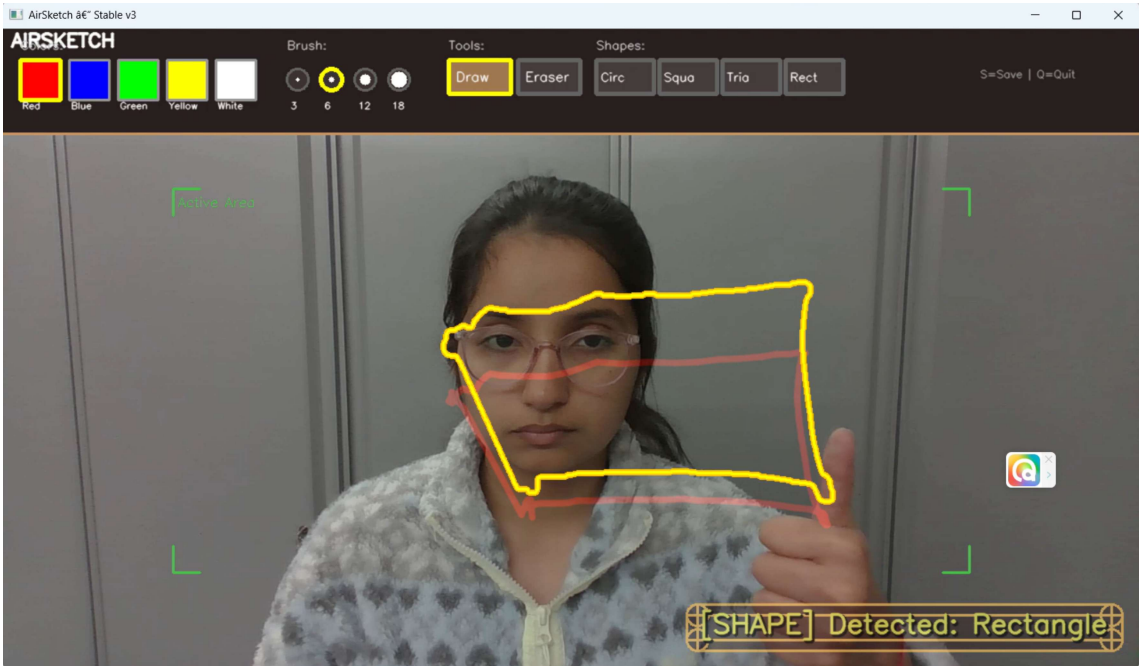

Figure 7: Example of freehand drawing and automatic shape recognition in AirSketch. A user sketches a rectangular shape in mid-air, which is detected and classified by the contour-based recognition pipeline.

5 Evaluation

We conducted an informal pilot evaluation with a small number of participants. Participants were asked to perform tasks including freehand drawing, erasing, changing colors and brush sizes, clearing the canvas, drawing shapes, and triggering shape recognition.

Participants were generally able to complete tasks with minimal instruction. Index-finger drawing and pinch-based menu interaction were perceived as intuitive. Occasional recognition errors occurred under poor lighting conditions or when the hand was far from the camera. Some participants reported mild arm fatigue during extended interaction.

Most errors were associated with motion-based gestures (e.g., swipe left), whereas static pose gestures were more reliable (Table 1).

6 Discussion

Our evaluation suggests that touchless drawing using mid-air hand gestures is feasible and learnable for short interaction sessions. Gesture gating and visual feedback were essential in preventing unintended actions. However, the absence of physical support leads to fatigue over time, and environmental factors significantly affect tracking accuracy.

These findings align with prior research on mid-air interaction and highlight the trade-offs between flexibility and precision in touchless systems.

6.1 Limitations and Future Work

AirSketch is sensitive to environmental conditions: poor lighting, background clutter, and large camera distance can reduce tracking stability and increase false triggers. Mid-air drawing can

also cause fatigue during prolonged use due to the lack of physical support. In future work, we plan to add adaptive thresholds per user, improved handedness handling, configurable gesture mappings, and more robust recognition under occlusion and varying distances. A controlled user study comparing AirSketch against mouse/stylus input would further quantify learnability, error rates, and fatigue.

7 Conclusion

We presented AirSketch, a gesture-based drawing system that enables mid-air sketching using only a webcam and computer vision techniques. By supporting freehand drawing, tool interaction, and shape recognition through intuitive gestures, AirSketch demonstrates the potential of touchless interfaces for lightweight creative tasks. As a research prototype, the system provides insights into the design and limitations of gesture-based drawing and motivates further exploration of adaptive and fatigue-aware interaction techniques.

References

1. C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Chang, M. Grundmann, and V. Bazarevsky, "MediaPipe: A Framework for Building Perception Pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

2. G. Casiez, N. Roussel, and D. Vogel, "The 1€ Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2012.

3. A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human–Computer Interaction*, 3rd ed., Pearson Education, 2004.

4. J. O. Wobbrock, M. R. Morris, and A. D. Wilson, "User-Defined Gestures for Surface Computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2009.

5. D. Wigdor and D. Wixon, *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*, Morgan Kaufmann, 2011.

6. R. K. Vatavu, L. Anthony, and J. O. Wobbrock, "Gesture Elicitation: A Survey and Taxonomy," *IEEE Transactions on Human–Machine Systems*, vol. 47, no. 3, pp. 378–396, 2017.

7. T. Oskam, A. Hornung, H. Bowyer, and L. Van Gool, "A Gesture-Based Interface for Interactive Drawing and Design," in *Proceedings of the ACM International Conference on Multimedia*, 2012.