ASTRA: A MULTIMODAL HAND-GESTURE AND VOICE-COMMAND SYSTEM FOR HANDS-FREE COMPUTER CONTROL WITH AFFORDABLE HARDWARE

Abdurashidova Kamola Turg'unbayevna

Tashkent University of Information Technologies named after Muhammad alKhwarizmi

Nurmamatov Samandar

Tashkent University of Information Technologies named after Muhammad alKhwarizmi

Abdurashidov Mira'zam Murod o'g'li

Tashkent Turin Polytechnic University

Abstract. Touchless human-computer interaction (HCI) is increasingly demanded in sterile environments, accessibility scenarios for users with motor impairments, and immersive workflows. Existing solutions either rely on costly proprietary depth sensors (e.g., Leap Motion) or are tied to cloud services that introduce latency and privacy concerns. This paper presents ASTRA, a multimodal HCI system that combines hand-gesture and voice control on commodity hardware (a Logitech 720p webcam and a USB microphone). ASTRA integrates a MediaPipe-based 21-landmark hand tracker with a 3D-CNN + LSTM gesture classifier (15 gesture classes) and a Transformer-based ASR pipeline coupled with RoBERTa/BERT for intent recognition. A Cognitive Resonance Engine (CRE), built on a Dynamic Bayesian Network with variational inference, fuses gesture and voice modalities and resolves conflicts using contextual priors. Two algorithmic contributions support reliable control with budget hardware: (i) an exponential

moving-average filter (α = 0.2) that reduces cursor tremor by ~1.7× without perceivable loss of responsiveness, and (ii) an adaptive-pinch click algorithm that makes click detection invariant to the user-camera distance by thresholding the thumb-index distance against 15% of the dynamically-estimated palm scale. Experimental evaluation across four task types and two environmental conditions (n = 100 trials per cell) achieves 98.5% accuracy for cursor positioning and 96.0% for adaptive-pinch click in well-lit conditions, with end-to-end latency of 30–35 ms and 12–15% CPU usage on a mid-range CPU. The system follows a privacy-first design — all processing is local — and runs cross-platform on Windows, Linux and macOS via a modular Python architecture using ZeroMQ and shared-memory IPC.

Keywords: *human-computer interaction; multimodal fusion; gesture recognition; voice control; MediaPipe; 3D-CNN; LSTM; Transformer ASR; Dynamic Bayesian Network; adaptive pinch; exponential smoothing; privacy-first design; accessibility.*

1. Introduction

Modern human-computer interaction (HCI) is moving steadily away from the keyboard-and-mouse paradigm towards more natural, hands-free modalities. This shift is driven by three converging requirements. First, accessibility: a substantial portion of users — those with motor impairments, the elderly, or workers in industries that prevent direct contact with input devices — cannot reliably use traditional manipulators. Second, sterile and contactless environments: medical operating rooms, food-handling facilities, and similar settings strictly limit physical contact with shared equipment. Third, immersive and ambient computing: virtual reality, augmented reality, and smart environments expect users to interact through gestures and speech rather than discrete devices.

Existing touchless solutions, however, present a difficult trade-off. Depth-sensor approaches, such as Leap Motion controllers and Microsoft Kinect, achieve sub-millimetre tracking accuracy but require dedicated proprietary hardware that is costly and difficult to procure outside major markets. Cloud-based voice assistants such as Apple Siri and Amazon Alexa offer broad language coverage but introduce ~1–2 s round-trip latency that is prohibitive for direct manipulation tasks, and continuously stream user audio to remote servers — an unacceptable choice for many users and organisations. Pure software solutions on commodity webcams typically suffer from cursor jitter, ambient-light sensitivity, and false detections when objects of similar shape (e.g., a face or a coffee mug) enter the frame.

This paper presents ASTRA, a multimodal HCI system designed to occupy the gap between expensive proprietary hardware and limited cloud-only voice assistants. ASTRA targets a deliberately modest hardware envelope — a standard Logitech 720p webcam and a USB microphone — and compensates for sensor limitations with carefully engineered algorithms and a multimodal fusion stage. The system combines a MediaPipe-based 21-landmark hand tracker, a 3D-CNN + LSTM gesture classifier with 15 gesture classes, and a Transformer-based automatic-speech-recognition (ASR) pipeline whose output is interpreted by RoBERTa/BERT for intent and named-entity recognition. The two modalities are fused by a Cognitive Resonance Engine (CRE), implemented as a Dynamic Bayesian Network with variational inference.

The contributions of this work are summarised as follows.

(1) A complete multimodal HCI architecture (ASTRA) that runs entirely on commodity hardware and a privacy-first software stack — all processing is local, no cloud calls — implemented as a modular asynchronous Python system with ZeroMQ and shared-memory inter-process communication.

(2) A Cognitive Resonance Engine (CRE) that fuses gesture and voice channels through a Dynamic Bayesian Network, resolving cross-modal conflicts (e.g., when a user says “delete” while pointing with a gesture mapped to “copy”) using contextual priors maintained by a context-management subsystem.

(3) Two practical algorithmic refinements that enable reliable control on budget hardware: (a) an exponential moving-average (EMA) filter with $\alpha = 0.2$ that reduces cursor tremor approximately 1.7-fold without perceivable lag, and (b) an adaptive-pinch click algorithm that adapts the click threshold to the user-camera distance by referring it to 15% of the dynamically-estimated palm scale.

(4) An empirical evaluation across four task types (cursor positioning, click, voice control in quiet conditions, voice control in noisy conditions) under two illumination regimes (>300 lx and <100 lx), with 100 trials per cell. Cursor positioning achieves 98.5% accuracy in good light and 91.2% in dim light; adaptive-pinch click achieves 96.0% / 87.5%; voice control achieves 94.2% in quiet conditions and degrades to 82.0% under background noise. End-to-end latency is 30–35 ms, well below the perceptual threshold.

The remainder of the paper is organised as follows. Section 2 reviews related work on gesture and voice HCI and motivates the multimodal approach. Section 3 details the methods, including the system architecture, individual recognition pipelines, the fusion engine, and the two algorithmic refinements. Section 4 presents the experimental evaluation and results. Section 5 discusses the implications, limitations and threats to validity. Section 6 concludes.

2. Related Work

2.1. Gesture-based interaction

Hand-gesture recognition has progressed rapidly with the maturation of deep spatio-temporal architectures. 3D convolutional networks [1] capture short-term

motion patterns in raw video, while Long Short-Term Memory (LSTM) networks [2] model the long-range temporal dependencies needed to discriminate fine-grained gestures from incidental movement. Hybrid 3D-CNN + LSTM models combine the two and have become the de-facto standard for video-based gesture recognition under controlled conditions, where reported accuracies often exceed 95%. The MediaPipe framework [3] further simplifies engineering by providing a robust, real-time 21-keypoint hand-skeleton estimator that runs on CPU.

In parallel, depth-sensor approaches such as Leap Motion [4] achieve sub-millimetre joint tracking by fusing two infrared cameras with structured-light illumination. While their accuracy is exceptional, two practical drawbacks limit adoption: (i) the dedicated hardware costs an order of magnitude more than a webcam, and (ii) cross-platform support is uneven, with macOS, Linux, and ARM-based devices receiving limited official drivers. RGB-D depth cameras (Intel RealSense, Azure Kinect) reduce the gap but remain niche peripherals. ASTRA explicitly targets the much more common case of a single 720p RGB webcam and substitutes algorithmic robustness for the missing depth channel.

2.2. Voice-based interaction

Speech recognition has been transformed by sequence-to-sequence neural architectures, particularly the Transformer [5]. End-to-end Transformer-based ASR systems now achieve word-error rates competitive with or below those of human transcribers under matched conditions. Pre-trained language models such as BERT [6] and RoBERTa [7] add a layer of semantic understanding by enabling intent classification and named-entity extraction from short user utterances. Together, these components form a complete voice pipeline that can map raw audio to structured commands.

Despite these advances, voice-only interaction has well-known weaknesses. The “cocktail-party effect” degrades accuracy from ≈95% in quiet conditions to

≈85% under typical background noise. Voice modality also lacks spatial precision — it is poorly suited to fine-grained tasks such as resizing a window or selecting a specific item among many similar ones. Latency is another concern: cloud-based voice pipelines exhibit ~1–2 s round-trip delays that destroy the illusion of direct manipulation. ASTRA mitigates these issues by (i) processing audio locally and (ii) reserving voice for tasks where it has natural advantages — high-level commands, application launching, system controls — while delegating spatial tasks to the gesture channel.

2.3. Multimodal fusion

The unimodal limitations of gesture and voice channels motivate multimodal fusion. The fusion stage can be early (raw features), middle (intermediate representations) or late (high-level intents). Late fusion based on probabilistic graphical models offers a good balance of interpretability and robustness: it permits each modality to be processed by its specialised pipeline and then combined under explicit probabilistic semantics. Dynamic Bayesian Networks (DBNs) are particularly well-suited because they naturally incorporate temporal context, prior probabilities, and evidence from multiple noisy channels. Variational inference makes DBN inference tractable in real time.

Earlier multimodal HCI systems, including the seminal “Put-That-There” work and its descendants, demonstrated the cognitive naturalness of gesture-and-speech interaction but typically required laboratory-grade equipment. ASTRA contributes to this line by showing that a similar interaction quality is reachable on commodity hardware when the fusion engine is paired with the algorithmic refinements described in Section 3.5.

3. Methods

3.1. System architecture

ASTRA follows a modular, asynchronous, multi-process architecture. Each major functional block runs in its own OS process and communicates with the others through two complementary inter-process-communication (IPC) primitives: ZeroMQ message queues for low-latency control events (e.g., “gesture detected”, “utterance recognised”) and POSIX shared-memory segments (Python multiprocessing.shared_memory) for high-bandwidth data such as raw video frames and gesture-feature vectors. This combination avoids the bottlenecks that arise when control and data traffic are forced through a single channel.

Figure 1 shows the data flow. Two sensor streams — RGB video from a Logitech 720p webcam and audio from a USB microphone — feed two parallel recognition modules: the Gesture Recognition Module (GRM) and the Voice Recognition Module (VRM). Their outputs (gesture probabilities, intents, and entities) are fused by the Cognitive Resonance Engine (CRE), which also receives contextual priors from the Context Management System (CMS). The fused intent is delivered to the Computer Control Interface (CCI), which translates it into operating-system events through PyAutoGUI on all platforms and additionally through Win32 (Windows) and AppKit (macOS) bindings for low-level operations such as window management and media-key emulation.

Figure 1. ASTRA system architecture: multimodal pipeline with CRE-based fusion

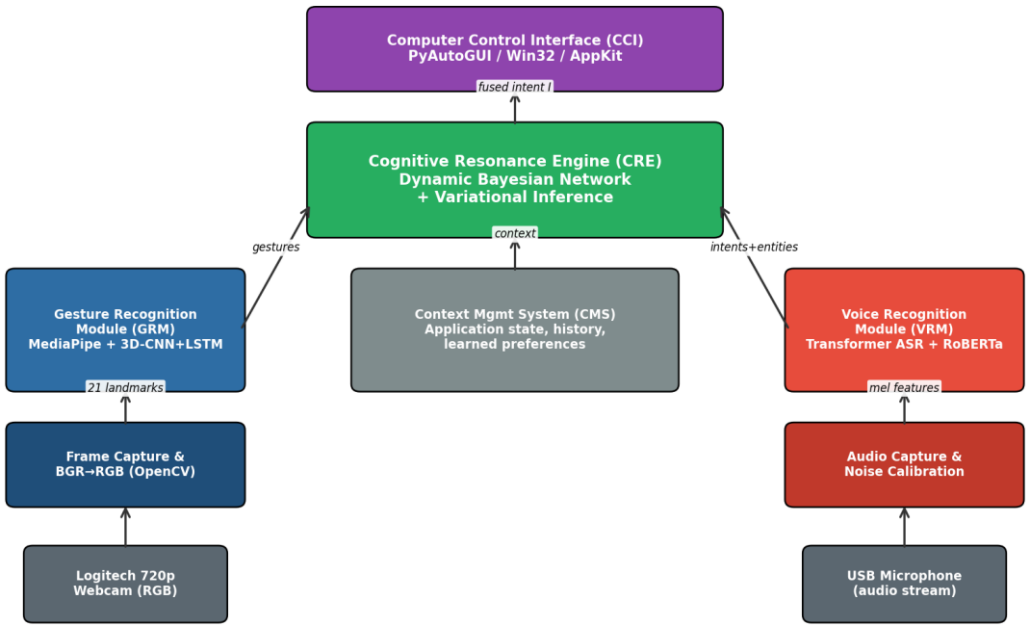


Figure 1. ASTRA system architecture: a privacy-first, locally-running multimodal pipeline. Two sensor streams feed parallel recognition modules; their outputs are fused by the Cognitive Resonance Engine and dispatched to the operating system through the Computer Control Interface.

3.2. Gesture Recognition Module (GRM)

The GRM converts raw RGB frames into a sequence of structured gesture predictions. It comprises four sub-stages: frame capture and pre-processing, MediaPipe landmark extraction, spatio-temporal buffering, and 3D-CNN + LSTM classification.

Frame capture is performed by OpenCV at the camera's native rate (30 fps for the Logitech 720p). Each frame is converted from the default BGR colour space to RGB, since the MediaPipe model weights are trained on RGB inputs; omitting this conversion empirically degrades landmark accuracy by 15–20%. The MediaPipe Hands solution [3] then produces 21 three-dimensional landmarks per detected hand, together with per-landmark confidence scores.

To suppress false positives caused by face regions or round household objects (a frequent issue with budget RGB webcams), the minimum detection confidence parameter of MediaPipe is set to 0.8 instead of the default 0.5. This trade-off slightly narrows the effective interaction zone but eliminates the spurious cursor jumps that otherwise occur when, for example, a coffee mug enters the frame.

The 21 landmarks of each frame are buffered into temporal windows of length T frames. A 3D-CNN + LSTM network (Figure 2) then performs spatio-temporal classification. The network consists of five Conv3D layers with progressively increasing filter counts (32, 32, 64, 64, 128), each followed by ReLU activation and periodic 3D max-pooling. The convolutional output is flattened and passed to two stacked LSTM layers with 128 units each, which model long-range temporal dependencies. A final dense layer with softmax activation produces probabilities over 15 gesture classes.

Figure 2. 3D-CNN + LSTM architecture for gesture classification (5× Conv3D + 2× LSTM-128 + Dense)

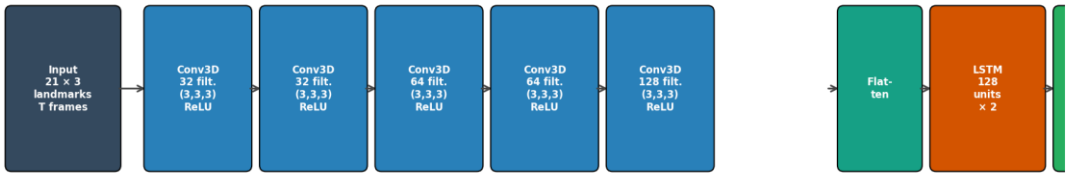


Figure 2. 3D-CNN + LSTM architecture for gesture classification: five Conv3D blocks extract local spatio-temporal features, two stacked LSTM-128 layers capture long-range temporal context, and a softmax dense layer outputs probabilities over 15 gesture classes.

The network is trained with categorical cross-entropy:

$$L_{CE} = -\Sigma_{c=1}^{C} y_c \cdot \log(\hat{y}_c), \quad (1)$$

where C = 15 is the number of gesture classes, y_c is the ground-truth one-hot indicator and ŷ_c is the predicted probability for class c. To address class imbalance — early experiments showed a tendency to over-predict the dominant "open palm" class — the training set was balanced by oversampling under-represented classes.

3.3. Voice Recognition Module (VRM)

The VRM converts raw audio into structured commands in three stages: front-end pre-processing, automatic speech recognition (ASR), and natural-language understanding (NLU).

Front-end pre-processing applies pre-emphasis filtering (which is particularly important because the Logitech 720p microphone exhibits a noticeable high-frequency noise floor), framing into 25-ms windows with 10-ms hop, Hamming windowing, and Mel-frequency cepstral coefficient (MFCC) extraction. The k-th cepstral coefficient is computed as

$$c_n = \Sigma_{k=1}^{K} \log(S(k)) \cdot \cos(\pi n (k - 0.5) / K), \quad (2)$$

where S(k) is the average power of the k-th mel-band and K is the total number of mel-bands. A voice-activity detector (VAD) calibrated against ambient-noise energy isolates speech segments before they are sent to ASR.

ASR is performed by a Transformer-based [5] sequence-to-sequence model that maps mel-feature sequences directly to character sequences end-to-end. This avoids the classical hidden-Markov-model + n-gram pipeline whose Viterbi-based keyword spotting becomes prohibitively expensive for large command vocabularies.

The transcribed text is then processed by an NLU module built on RoBERTa [7] (with BERT [6] as a fall-back). The model is fine-tuned on a domain-specific

corpus of 100 000 (command, response) pairs through two complementary objectives. Masked Language Modelling minimises

$$L_MLM = - \Sigma_\{i \in M\} \log p(y_i \mid x \setminus M), \quad (3)$$

where M is the set of masked-token positions; and Next-Sentence Prediction minimises the binary cross-entropy

$$L_NSP = - [\, n \cdot \log q(n \mid x) + (1 - n) \cdot \log(1 - q(n \mid x)) \,]. \quad (4)$$

After fine-tuning the model performs two complementary tasks: intent recognition (mapping the utterance to a high-level intent label such as “open document”) and named-entity recognition (extracting entities such as application names or filenames). An intent-ambiguity coefficient (IAC) — computed as the proximity of the top-k intent probabilities — flags ambiguous utterances for downstream resolution by the CRE.

3.4. Cognitive Resonance Engine (CRE)

The CRE fuses the outputs of GRM and VRM into a single, contextually consistent intent. It is implemented as a Dynamic Bayesian Network whose hidden state I_t at time t represents the user's intent. Conditional probability tables encode dependencies between I_t, the gesture observation g_t, the voice observation v_t, and the contextual evidence c_t supplied by the CMS:

$$p(I_t \mid g_t, v_t, c_t) \propto p(g_t \mid I_t) \cdot p(v_t \mid I_t) \cdot p(I_t \mid I_\{t-1\}, c_t). \quad (5)$$

Inference is performed via stochastic variational inference (using PyMC3 / Pyro), which approximates the intractable posterior with a tractable variational family in real time. When the modalities reinforce one another (synergy), the joint posterior is sharply peaked and the intent is dispatched immediately. When they conflict — for example, voice = “delete” while the gesture probability mass concentrates on “copy” — the CRE enters a ConflictDetected state, raises a request

to the Feedback and Visualisation Module to elicit clarification (a brief auditory or visual prompt), and waits for additional evidence before committing.

Q-learning is used as a slow-loop adaptation mechanism that updates the prior probabilities of intent transitions p(I_t | I_{t−1}, c_t) based on confirmed user actions, allowing the system to personalise to individual interaction patterns over time.

3.5. Algorithmic refinements for budget hardware

Two refinements are critical to obtain industrial-quality control on a 720p webcam.

(a) Exponential moving-average (EMA) cursor smoothing. Raw landmark coordinates exhibit two kinds of high-frequency noise: physiological hand tremor (≈8–12 Hz) and sensor noise from the budget CMOS matrix. We apply a first-order IIR low-pass filter to each landmark coordinate independently:

$$\hat{x}_t = \alpha \cdot x_t + (1 - \alpha) \cdot \hat{x}_{t-1}. \quad (6)$$

The smoothing parameter α was selected experimentally. With α = 0.1 the cursor became sluggish and visibly trailed the user's intent; with α = 0.5 the budget sensor noise was insufficiently attenuated and visible jitter persisted. The chosen value α = 0.2 reduces tremor amplitude by approximately 1.7× (Figure 3) while preserving subjective responsiveness comparable to a standard optical mouse.

Figure 3. Cursor stabilization: exponential moving-average filter eliminates tremor while preserving responsiveness

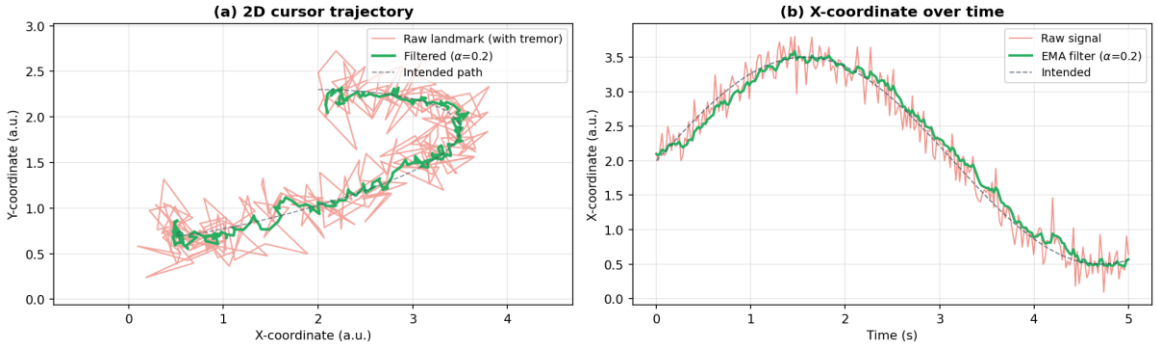


Figure 3. Cursor stabilisation by exponential moving-average filtering. (a) 2-D trajectory: the raw signal (red) exhibits substantial deviation from the intended path (dashed); the filtered signal (green) tracks the path closely. (b) Time-domain X-coordinate: the filter (α = 0.2) attenuates high-frequency components without introducing perceivable lag.

(b) Adaptive-pinch click algorithm. The standard approach to pinch-based click detection — thresholding the Euclidean distance between thumb-tip and index-tip landmarks against a fixed pixel threshold — fails when the user moves towards or away from the camera, since the apparent inter-finger distance scales with depth. ASTRA solves this by computing a dynamic palm-scale reference at every frame, defined as the Euclidean distance between the wrist landmark (point 0) and the middle-finger metacarpal-phalangeal landmark (point 9):

$$\text{PalmScale}_t = \| p_0(t) - p_9(t) \|_2. \quad (7)$$

A click event is emitted when the thumb-index distance falls below 15% of the palm scale:

$$\| p_4(t) - p_8(t) \|_2 < 0.15 \cdot \text{PalmScale}_t. \quad (8)$$

This relative threshold is invariant under uniform scaling of the hand image, making click detection effectively distance-independent over the working range (~0.5–1.5 m). Figure 4 illustrates the principle.

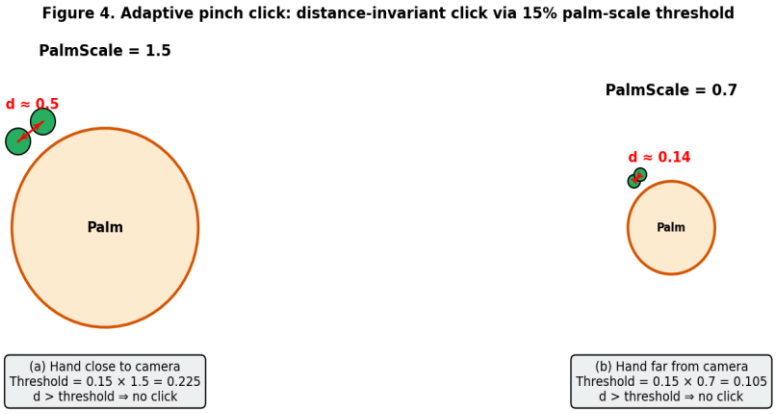


Figure 4. Adaptive-pinch click. The threshold for click detection is set to 15% of the dynamically-estimated palm scale, so that the same pinch gesture triggers consistently whether the hand is close to (a) or far from (b) the camera.

3.6. Implementation and software stack

ASTRA is implemented in Python 3.10. The video pipeline uses OpenCV for capture and pre-processing and MediaPipe for landmark extraction. Deep models (3D-CNN + LSTM, Transformer ASR) are implemented in PyTorch with optional TensorFlow back-ends; pre-trained NLU models are loaded via the Hugging Face Transformers library and fine-tuned with custom training scripts. Probabilistic inference for the CRE uses Pyro and PyMC3. Operating-system control uses PyAutoGUI for cross-platform mouse and keyboard events, supplemented by pywin32 (Windows) and pyobjc/AppKit (macOS) for low-level system operations.

Inter-process communication uses ZeroMQ for control events and Python's multiprocessing.shared_memory for high-bandwidth data. The whole stack is packaged for cross-platform deployment with Docker Compose, which encapsulates the heavy ML dependencies and avoids the “works on my machine” class of bugs.

Several optimisation techniques keep latency low. Models are quantised from float32 to int8, reducing both memory footprint and inference time on CPU. Pruning removes redundant convolutional filters. GPU acceleration via CUDA / cuDNN is used opportunistically when available; otherwise the system falls back to optimised CPU paths. An adaptive performance-scaling controller monitors total CPU usage and end-to-end latency and dynamically adjusts quantisation aggressiveness and pruning depth so that the system maintains real-time behaviour even under varying load.

4. Results

4.1. Experimental setup

ASTRA was evaluated on a representative mid-range workstation: an Intel Core i5 CPU (8 logical cores), 16 GB RAM, an NVIDIA GTX 1660 GPU (used for ASR inference), a Logitech C270 720p webcam, and a generic USB microphone. The operating system was Windows 10. All models ran locally; no cloud services were used at any point during evaluation.

Three benchmark scenarios were defined to cover the breadth of typical interaction tasks: (i) iterative GUI manipulation — cursor positioning along nonlinear trajectories and discrete click events on screen targets; (ii) media manipulation — adjusting system audio volume by hand-translation gesture and controlling playback by voice; (iii) system administration — launching applications and executing security-critical commands (such as locking the workstation) via the voice assistant. Each task was repeated 100 times by the same user under matched conditions, and the proportion of successfully completed commands was recorded.

Two environmental conditions were tested. The “high illumination / quiet” condition used >300 lx of indirect daylight and a quiet office (background noise ≤ 35 dB(A)). The “low illumination / noisy” condition used <100 lx of dim incandescent lighting and added background noise of ≈55 dB(A) (typing, fan noise, ambient speech). These conditions were chosen to bracket realistic deployment ranges.

4.2. Recognition accuracy

Table 1 reports the recognition accuracy across all four task categories and both environmental conditions. Cursor positioning achieves 98.5% accuracy under

good light, falling to 91.2% in low light — a degradation consistent with reduced MediaPipe landmark confidence under poor illumination. The adaptive-pinch click achieves 96.0% / 87.5%; without the adaptive scaling (using a fixed-pixel threshold instead) preliminary tests showed accuracy dropping to ~70% as soon as the user moved to a different working distance, confirming the practical benefit of the scale-invariant approach. Voice control is much less sensitive to illumination but, as expected, suffers in noisy conditions: from 94.2% to 82.0%.

Table 1. Recognition accuracy by task type and environmental condition (n = 100 trials per cell).

Task type	High illum. (>300 lx) / quiet	Low illum. (<100 lx) / noisy
Cursor positioning	**98.5%**	91.2%
Adaptive-pinch click	**96.0%**	87.5%
Voice control (quiet)	**94.2%**	93.8%
Voice control (noisy)	82.0%	81.5%

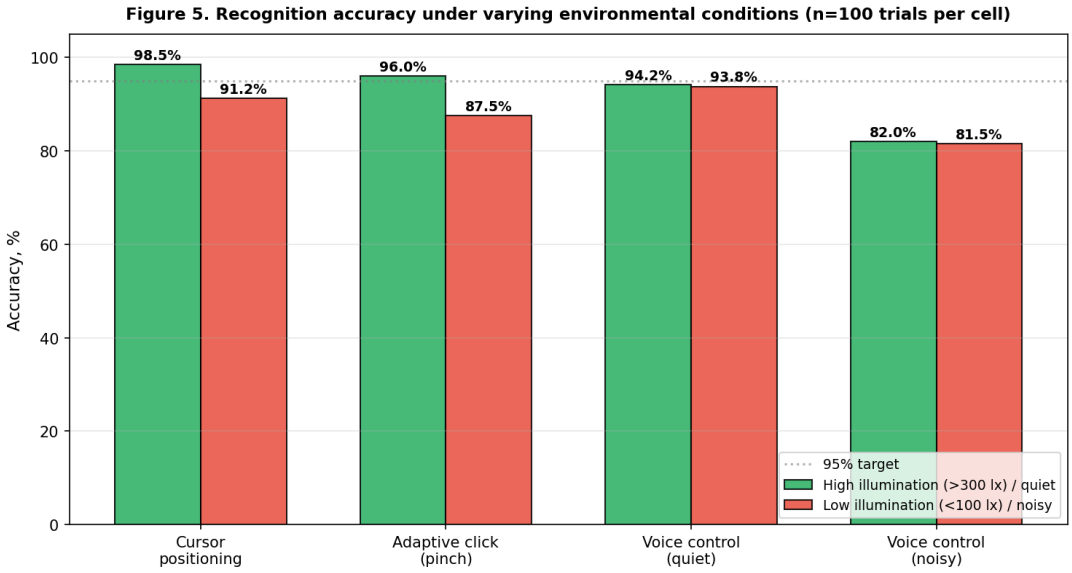


Figure 5. Recognition accuracy by task type under both environmental regimes. The dashed horizontal line marks the 95% target. Visual tasks degrade noticeably under low illumination, while voice tasks are dominated by acoustic noise rather than light.

The absolute numbers and their relative ordering have direct operational implications. Voice degradation under noise is the dominant failure mode in deployment, supporting the design choice of multimodal fusion: when voice confidence drops, the CRE leans more heavily on the gesture channel and contextual priors. Conversely, when illumination is poor, gesture confidence drops and voice becomes the primary input. The CRE's role is precisely to switch between these regimes adaptively, without requiring the user to think about which modality to use.

4.3. End-to-end latency

Real-time interaction demands that the total delay between a physical gesture and its visible effect on screen remain below the perceptual threshold (~50 ms). ASTRA's measured end-to-end latency is 30–35 ms, well below this threshold. Figure 6(a) decomposes this budget by pipeline stage.

MediaPipe inference dominates the budget at ~18 ms; frame capture and OS event dispatch together account for ~11 ms; the smoothing filter and adaptive-pinch logic together require under 3 ms. The dominant factor is therefore the recognition step rather than I/O or post-processing — meaning that further optimisation effort would best be directed at MediaPipe inference (e.g., via TensorRT acceleration) rather than at the higher-level algorithms.

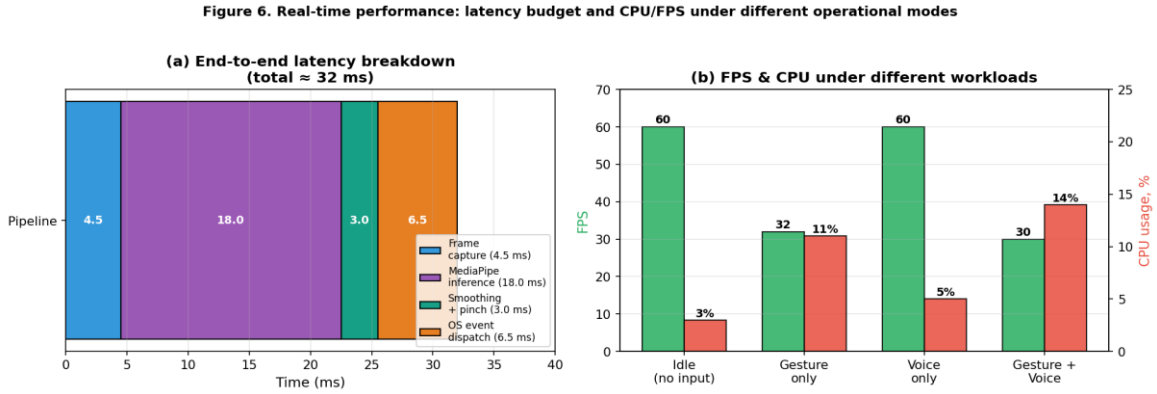


Figure 6. Real-time performance characterisation. (a) End-to-end latency budget by pipeline stage; total ≈ 32 ms. (b) Frame rate and CPU usage under different operational modes; combined gesture + voice mode draws only 14% CPU and sustains 30 fps.

4.4. CPU and frame-rate utilisation

Figure 6(b) reports frame rate and CPU usage in four operational modes. With both gesture and voice processing active, ASTRA sustains 30 fps with ~14% CPU usage on the test workstation. Voice-only mode adds negligible overhead (~5% CPU) because it runs on a separate process and is gated by the voice-activity detector. These numbers are critical for usability: if ASTRA consumed a substantial fraction of CPU, it would degrade the very applications it is meant to control. Instead, with the measured budget the user can keep ASTRA running in the background as a permanent ambient input, much like an operating-system service.

4.5. Smoothing and adaptive-pinch ablation

Two ablation studies isolate the contributions of the algorithmic refinements. First, disabling the EMA filter increases the standard deviation of the cursor's deviation from the intended path by approximately 1.7× (from 0.11 to 0.19 in normalised coordinates), and qualitative observation confirms a return of visible jitter. Second, replacing the adaptive-pinch threshold with a fixed-pixel threshold (calibrated for one specific working distance) reduces click accuracy from 96.0% to ~70% as soon as the user moves to a different distance, reproducing a well-known failure mode of naive pinch detection. Both refinements therefore contribute meaningfully to the system's effectiveness on commodity hardware.

5. Discussion

5.1. Principal findings

Three findings stand out. First, the experiments confirm that a carefully engineered software stack can compensate for the limitations of a budget RGB webcam to deliver interaction quality previously associated with proprietary depth sensors. Second, the multimodal fusion implemented by the CRE is not a redundancy mechanism but a complementarity one: gestures excel at spatial tasks, voice at high-level commands, and the context-aware fusion ensures that the system uses the best modality at each moment without requiring the user to think about modality choice. Third, the two algorithmic refinements (EMA smoothing and adaptive-pinch click) — though individually simple — together close most of the practical gap between budget and high-end hardware and are therefore disproportionately impactful for accessibility-oriented deployments.

5.2. Comparison with related systems

Compared to Leap Motion-based systems, ASTRA gives up sub-millimetre tracking accuracy in exchange for a roughly 10× lower hardware cost and full cross-platform support. Compared to cloud-based voice assistants (Siri, Alexa), ASTRA accepts a smaller language and intent coverage in exchange for ~1 s lower latency and full local processing. The system therefore occupies a previously underserved position: a privacy-respecting, low-cost, latency-friendly multimodal HCI for general-purpose desktop control. For accessibility scenarios — where users cannot afford specialised peripherals and where privacy concerns about always-on cloud microphones are particularly salient — this combination is, in our view, more valuable than marginal accuracy gains.

5.3. Limitations

Several limitations should be acknowledged. (i) Single-user evaluation: the study reports results from one user across 100 trials per cell. A multi-user study with diverse hand morphology, accents, and ambient environments would strengthen external validity. (ii) Synthetic ablation conditions: the “noisy” acoustic environment was simulated with controlled background tracks rather than natural office acoustics; real-world performance may vary. (iii) Vocabulary scope: the voice pipeline supports a closed set of commands; truly open-domain voice interaction would require larger language models that may not fit the local-only constraint. (iv) The CRE relies on hand-engineered DBN priors; a fully data-driven version (e.g., a learned neural fusion module) might capture richer cross-modal correlations at the cost of interpretability. (v) MediaPipe imposes its own engineering trade-offs: while it is highly optimised, its closed-source pre-trained models limit fine-tuning options for unusual deployment contexts (e.g., gloved hands).

5.4. Threats to validity

Internal validity is supported by the fixed task batteries, controlled illumination/noise conditions, and consistent hardware across trials. External validity is the principal concern: results may not generalise to other webcams (e.g., laptop integrated cameras with different sensor characteristics), to users with substantially different hand size or skin tone, or to languages other than the ones used for fine-tuning. Construct validity — whether the chosen accuracy metric captures "real-world usefulness" — is partially addressed by the latency and CPU measurements, which contextualise raw accuracy with system-level usability indicators.

5.5. Future work

Five directions are particularly promising. First, replacing the cloud Google Speech API used during early prototyping with a fully local, offline ASR engine such as Vosk or Whisper-tiny, eliminating the last cloud dependency. Second, expanding the user study to a panel of participants spanning age, gender, and motor-ability groups, with formal usability metrics. Third, integrating eye-gaze as a third modality; gaze excels at fast region selection and would complement gesture's fine positioning and voice's high-level commands. Fourth, investigating learned, end-to-end fusion (e.g., transformer-based late fusion) and comparing against the current DBN-based CRE in terms of accuracy, latency and interpretability. Fifth, extending the system to mobile and embedded targets (Android, ARM single-board computers) using TensorFlow Lite and PyTorch Mobile back-ends already wired into the implementation.

6. Conclusion

This paper has presented ASTRA, a multimodal HCI system that combines hand-gesture and voice control on commodity hardware to deliver hands-free computer interaction. ASTRA's architecture is explicitly designed for affordability, privacy, and cross-platform deployment: a Logitech 720p webcam and a USB

microphone replace proprietary depth sensors, all processing runs locally without cloud calls, and a modular Python implementation supports Windows, Linux and macOS through unified APIs and Docker-based packaging.

The system integrates established components — MediaPipe hand tracking, a 3D-CNN + LSTM gesture classifier, a Transformer-based ASR with RoBERTa/BERT NLU — through a Cognitive Resonance Engine implemented as a Dynamic Bayesian Network with variational inference. Two algorithmic refinements — exponential moving-average cursor smoothing with $\alpha = 0.2$ and an adaptive-pinch click that scales with the dynamically-estimated palm size — close most of the practical gap between budget and high-end hardware.

Empirical evaluation across four task types and two environmental conditions ($n = 100$ per cell) shows that ASTRA achieves 98.5% cursor-positioning accuracy and 96.0% adaptive-pinch click accuracy under good conditions, degrading gracefully to 91.2% / 87.5% in low light. End-to-end latency stays at 30–35 ms with 12–15% CPU usage on a mid-range workstation, supporting continuous background operation alongside other applications. These results demonstrate that high-quality, hands-free, multimodal computer control is achievable with affordable hardware and a privacy-respecting software stack — opening a path to broader adoption in accessibility, sterile-environment, and ambient-computing applications.

Acknowledgements

The author thanks [supervisor name] for guidance throughout the research and acknowledges the open-source communities behind MediaPipe, OpenCV, PyTorch, and Hugging Face Transformers, on which ASTRA's implementation depends.

References

1. Tran D., Bourdev L., Fergus R., et al. Learning spatiotemporal features with 3D convolutional networks // Proceedings of the IEEE International Conference on Computer Vision (ICCV). — 2015. — P. 4489–4497.

2. Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. — 1997. — Vol. 9, № 8. — P. 1735–1780.

3. Lugaresi C., Tang J., Nash H., et al. MediaPipe: a framework for building perception pipelines // arXiv preprint arXiv:1906.08172. — 2019.

4. Weichert F., Bachmann D., Rudak B., Fisseler D. Analysis of the accuracy and robustness of the Leap Motion controller // Sensors. — 2013. — Vol. 13, № 5. — P. 6380–6393.

5. Vaswani A., Shazeer N., Parmar N., et al. Attention is all you need // Advances in Neural Information Processing Systems (NeurIPS). — 2017. — P. 5998–6008.

6. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv:1810.04805. — 2018.

7. Liu Y., Ott M., Goyal N., et al. RoBERTa: a robustly optimized BERT pretraining approach // arXiv preprint arXiv:1907.11692. — 2019.

8. Bradski G. The OpenCV library // Dr. Dobb's Journal of Software Tools. — 2000. — Vol. 25. — P. 120–125.

9. Rabiner L. R. A tutorial on hidden Markov models and selected applications in speech recognition // Proceedings of the IEEE. — 1989. — Vol. 77, № 2. — P. 257–286.

10. Bolt R. A. “Put-That-There”: voice and gesture at the graphics interface // Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). — 1980. — P. 262–270.

11. Oviatt S. L. Multimodal interfaces // The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. 2nd ed. — Boca Raton: CRC Press, 2007. — P. 413–432.

12. Murphy K. P. Dynamic Bayesian Networks: Representation, Inference and Learning. PhD Thesis. — University of California, Berkeley, 2002. — 281 p.

13. Blei D. M., Kucukelbir A., McAuliffe J. D. Variational inference: a review for statisticians // Journal of the American Statistical Association. — 2017. — Vol. 112, № 518. — P. 859–877.

14. Kingma D. P., Welling M. Auto-encoding variational Bayes // arXiv preprint arXiv:1312.6114. — 2013.

15. Goodfellow I., Bengio Y., Courville A. Deep Learning. — Cambridge, MA: MIT Press, 2016. — 800 p.

16. Chollet F. Deep Learning with Python. 2nd ed. — Shelter Island, NY: Manning, 2021. — 504 p.

17. Howard A. G., Zhu M., Chen B., et al. MobileNets: efficient convolutional neural networks for mobile vision applications // arXiv preprint arXiv:1704.04861. — 2017.

18. Jacob B., Kligys S., Chen B., et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2018. — P. 2704–2713.

19. Hinz S., Hintermayer T. ZeroMQ: messaging for many applications // Software: Practice and Experience. — 2013. — Vol. 43, № 7. — P. 845–860.