WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

FGA: Face Gesture Assistance

Dr. Prakash Prasad[1], Kushkumar Shanware[2], Priya Meshram[3], Arti Pannalal Thakur[4], Anirudh Sen[5], Snehal Selukar[6]

[1] Project Guide Department of Information Technology Priyadarshini College of Engineering Nagpur, Maharashtra, India
Email: prakash.prasad@pcenagpur.edu.in
[2] Co-Guide, Department of Information Technology Priyadarshini College of Engineering Nagpur, Maharashtra, India
Email: Priyameshram1485@gmail.com
[3] Department of Information Technology Priyadarshini College of Engineering Nagpur, Maharashtra, India
Email: aniruddh.sen10@gmail.com
[4]Department of Information Technology Priyadarshini College of Engineering Nagpur, Maharashtra, India
Email: Shanwarekush@gmail.com
[5]Department of Information Technology Priyadarshini College of Engineering Nagpur, Maharashtra, India
Email: aartithakur5991@gmail.com
[6] Department of Information Technology Priyadarshini College of Engineering Nagpur, Maharashtra, India
Email: Snehalganpatiselukar@gmail.com

DOI : https://doi.org/10.55248/gengpi.07.1026.34047

Abstract—

Traditional desktop interfaces rely on a physical mouse and keyboard, which can be difficult to use for people with limited hand mobility and in hands-busy situations. Face Gesture Assistant (FGA) is an offline desktop interaction prototype that combines facial-landmark pointer movement with speech commands and dictation. The camera pipeline uses OpenCV and MediaPipe Face Landmarker to estimate a facial refer-ence point, map movement to screen coordinates, and smooth pointer motion. Mouth opening is detected using the Euclidean distance between upper- and lower-lip landmarks. The audio pipeline captures 16-kHz mono audio, uses preprocessing and Silero voice activity detection (VAD), and transcribes speech locally with Faster-Whisper. A command handler normalizes text and dispatches supported desktop operations. The camera and speech pipelines run in separate Qt threads within one application process. During detected speech, mouth-click actions are suppressed to reduce unintended clicks caused by speaking-related lip movement. Runtime logs from three Windows laptops show that effective camera rate and end-to-end latency vary by run, while cursor-coordinate processing is comparatively inex-pensive. These findings are engineering observations rather than a controlled accessibility or usability study. The paper describes the implementation, summarizes the available measurements, and identifies additional testing needed to establish reliability.

***Index Terms*—Assistive technology, multimodal interaction, facial landmarks, speech recognition, offline computing, desktop automation**

1. Introduction

A mouse and keyboard are the standard means of interacting with desktop computers, but they are not suitable for every user or situation. People with limited upper-limb mobility may need alternative input methods, and hands-free interaction can also be useful when physical input devices are inconvenient. Camera-based interfaces provide a way to control a pointer through head or facial movement, while speech interfaces sup-port commands and text entry. Prior work on Camera Mouse demonstrated visual tracking as an alternative input method for people with severe disabilities [1]. Facial-feature-based mouse control has also been investigated [2]. Voice interaction has been studied as an alternative to conventional web input [7],

[8]. Tooth-click interfaces also illustrate how small facial or oral movements can serve as intentional assistive input [17]. A recent Camera Mouse case study further highlights the need to assess practical use in context rather than infer accessibility outcomes from implementation alone [18].

Face Gesture Assistant (FGA) combines these input modes in one Windows desktop application. Facial movement pro-vides continuous pointer control, and a mouth-open state op-erates the left mouse button. Speech supports discrete desktop commands and dictation. The application is designed to run locally after its required models have been installed, so routine transcription does not require a remote speech service. Local execution reduces dependence on network availability but does not by itself guarantee low latency or complete privacy under every operating-system configuration.

This paper describes the implementation and available run-time evidence for FGA. Its contributions are: (1) an overview of the camera and speech pipelines and their coordination; (2) a description of the mouth-distance gesture and offline speech-command workflow; and (3) a summary of

measurements from three laptop sessions. The evaluation is limited to engineering logs. It does not claim formal gesture accuracy, word error rate, clinical effectiveness, or Fitts' law throughput because these outcomes were not measured using a controlled protocol.

2. Related Work

Camera-based pointing has been studied as an alternative to direct physical input. Betke *et al.* introduced Camera Mouse, which tracks a body feature and maps its movement to a pointer [1]. Tu *et al.* explored facial features for mouse control [2]. Facial-landmark localization remains sensitive to head pose, illumination, occlusion, and input-image quality [3].

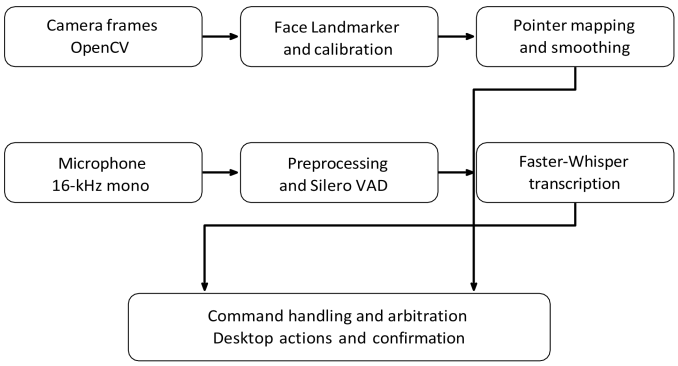


Fig. 1. FGA architecture. Camera and speech processing run in separate Qt threads and converge at command handling.

MediaPipe provides reusable components for perception pipelines, including facial landmark estimation [4], [5]. Landmark-based interfaces can avoid dedicated wearable sen-sors, although camera quality and host-computer performance constrain frame rate and processing delay. Related low-cost multimodal interfaces have likewise combined facial land-marks with voice commands, supporting the relevance of integrating complementary input channels in a desktop-access setting [19]. A smoothing filter can reduce cursor jitter, but filtering introduces a trade-off between stability and respon siveness. The 1 Euro filter was proposed as a speed-adaptive low-pass filter for interactive systems [6].

Speech input complements visual interaction because it can express discrete intentions without requiring precise pointing. Whisper was trained using large-scale weak supervision, and Faster-Whisper provides an optimized inference implementa-tion based on CTranslate2 [9], [10]. Even with local transcrip-tion, speech recognition must be distinguished from command interpretation: a transcript may be incomplete, contain an unsupported phrase, or resemble another command.

Multimodal systems must manage conflicts between input channels. Research on multimodal interfaces discusses com-bining complementary input while preserving clear interaction rules [11]. FGA uses a simple arbitration mechanism: pointer tracking continues during speech, but mouth-click actions are suppressed while speech activity is detected. This addresses one interaction conflict, though its effectiveness has not been measured in a controlled user study.

3. System Architecture

FGA runs as a single application process. The main window manages separate Qt QThread workers for camera processing and speech recognition. The workers communicate through Qt signals and slots [15]; the implementation does not use a multi-process architecture. Separating the workloads into threads keeps long-running input tasks away from the main graphical event loop, although thread scheduling and CPU contention can still affect performance.

A. Facial Interaction Pipeline

The camera worker obtains frames through OpenCV and passes RGB image data to MediaPipe Face Landmarker. A selected facial anchor provides a reference for pointer move-ment. During calibration, the application establishes a mapping between the anchor position and screen coordinates. A dead zone around the calibrated center reduces small unintended

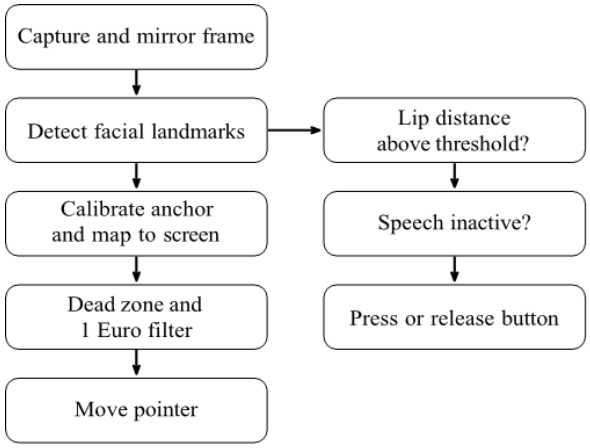


Fig. 2. Facial processing and mouth-click decision flow.

pointer changes. A 1 Euro filter smooths the pointer trajectory [6]. The camera is configured to capture at a high available rate, but the actual frame rate varies with the camera, inference cost, and system workload.

Mouth opening is detected from the normalized two-dimensional coordinates of upper-lip landmark 13 and lower-lip landmark 14. Let the coordinates be $p_{13} = (x_{13}, y_{13})$ and $p_{14} = (x_{14}, y_{14})$. The implementation computes

$$d_{\text{mouth}} = \sqrt{\quad (x_{13} - x_{14})2 + (y_{13} - y_{14})2.} \qquad (1)$$

The mouth is treated as open when $d_{\text{mouth}} > 0.035$. Opening presses the left mouse button, and returning below the threshold releases it. This is a Euclidean landmark-distance rule, not a mouth-aspect-ratio calculation. Eye-aspect-ratio gesture recognition is not implemented in the current version. The fixed threshold is simple and inexpensive, but its suitabil-ity may vary with camera placement and facial geometry.

B. Speech Interaction Pipeline

The audio worker captures mono audio at 16 kHz using SoundDevice. The audio path applies preprocessing such as high-pass filtering, echo cancellation, noise suppression, and automatic gain control when supported by the configured audio-processing module. Short audio frames are placed in a bounded queue. If the queue becomes full, the oldest frame can be discarded to avoid building up a long backlog of stale audio.

Silero VAD identifies likely speech segments [12]. The current configuration uses a 0.5 speech threshold, pre-roll of approximately 192 ms, post-roll of approximately 480 ms, and a maximum segment duration of about eight seconds. Pre-roll helps retain speech onset before the detector activates; post-roll helps retain the end of a phrase.

VAD segments are transcribed using Faster-Whisper with the base model, CPU inference, INT8 computation, English-language decoding, beam_size=3, and best_of=3 [9],

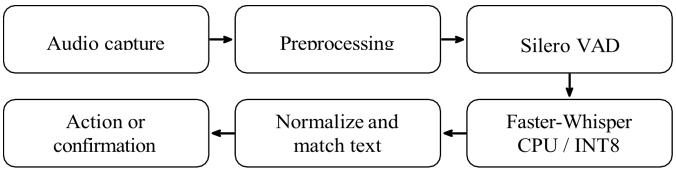


Fig. 3. Offline speech-command processing from audio capture to action dispatch.

[10]. These settings describe the implementation and do not imply that transcription is always real-time or error-free.

C. Command Handling and Arbitration

Recognized text is normalized by converting it to lowercase, removing punctuation, collapsing whitespace, and handling repeated transcription fragments. The command handler uses exact matching for single-word commands and fuzzy matching for selected multi-word commands. A cooldown helps reduce repeated activation. Dictation follows a separate path from ordinary command matching. An emergency-stop state dis-ables ordinary actions except the control used to re-enable the system.

Speech activity also affects the facial channel. While VAD reports active speech, mouth-click actions are suppressed, but pointer tracking remains available. This rule reduces the chance that lip movement during speaking will trigger a click. It is a practical arbitration mechanism rather than a learned intent model, and the current evaluation does not quantify false clicks or missed clicks.

4. Implementation and Safety Considerations

The graphical interface is implemented with PySide6. OpenCV handles camera acquisition and image conversion, MediaPipe supplies facial landmarks, and PyAutoGUI issues pointer and mouse-button actions [13], [16]. Windows-specific desktop functionality uses Windows APIs and supporting Python libraries for window and process information. Sup-ported operations include switching, restoring, minimizing, maximizing, moving, and closing windows. Selected power operations use confirmation to reduce the risk of accidental execution.

The camera and speech workers share the same application process but execute in separate Qt threads. This design im-proves separation of responsibilities without implying that the workloads cannot contend for CPU time. Audio processing and ASR may consume substantially more resources than cursor-coordinate updates, while camera inference can also limit frame rate. The observed timing therefore depends on the full pipeline rather than on one library alone.

The command layer is a safety boundary between uncertain speech recognition and desktop actions. A transcription can be incomplete or resemble an unsupported command, and fuzzy matching may select an unintended action. Cooldowns, an emergency stop, and confirmation dialogs reduce some risks but do not constitute a formal safety certification. Before consequential use, the application should be tested with invalid phrases, near-matching commands, background conversation, and recovery after accidental activation.

5. Experimental Method

The evaluation summarizes runtime logs from three Win-dows laptops. The sessions were not conducted under a common benchmark protocol, and their duration and back-ground conditions differed. The results should therefore be treated as engineering observations, not a controlled hardware comparison.

The tested systems were: D1, an ASUS TUF laptop with an Intel Core i5-11400H, 8 GB RAM, and Windows 11; D2, an HP Victus with a 13th-generation Intel Core i5, 16 GB RAM, and Windows 11; and D3, an HP Notebook with an Intel Core i5-1235U, 16 GB RAM, and Windows 11. The exact CPU model for D2 was not available.

Camera FPS describes the effective rate recorded in a run, not the camera's advertised maximum. Face-detection time and cursor-processing time are separate stages. Cursor-processing time does not include frame acquisition, landmark inference, operating system scheduling, or display refresh. End-to-end latency is reported separately when available. VAD segment counts, ASR events, and command counters measure different stages and should not be treated as interchangeable accuracy metrics.

No reference transcript, labelled command set, formal ges-ture test, participant study, or structured Fitts' law pointing experiment was provided. The paper therefore does not report word error rate, command precision/recall, clinical outcomes, or pointing throughput. A formal pointing study would need a repeatable target-acquisition protocol and measures such as movement time and error rate.

6. Results

Table I summarizes the available measurements. D1 and D2 have aggregated statistics. For D3, only observed ranges are reported because a verified aggregate summary was not available.

On D1, mean and median camera rates were approximately

7.5 frames per second. Mean face-detection time was 9.53 ms, and the 95th-percentile value was 15.38 ms. Cursor-processing time averaged 0.14 ms. The log recorded 29 VAD segments, 29 Whisper events, and 28 command events, including one unknown command. The logged unknown-command rate was 3.6%, but this is a command-counter statistic rather than word error rate or overall speech-recognition accuracy. No audio overflow occurred in this particular normal-use run; separate functional testing showed overflow events under other conditions. Maximum logged end-to-end latency was 3267 ms. On D2, mean and median camera rates were 14.93 and

14.90 FPS. Mean face-detection time was 27.67 ms, the 95th percentile was 40.39 ms, and the maximum was 55.5 ms. Cursor processing took approximately 0.23 ms. The session recorded no VAD drops or audio overflows. The final logged command counter was 21, with seven unknown commands and a reported unknown-command rate of 33.3%. Since the

TABLE I

Runtime observations from three laptop configurations

Metric	D1: ASUS TUF	D2: HP Victus	D3: HP Notebook	Notes
Session duration	~5.8 min	~3.1 min	~2.4 min	Non-controlled
Mean camera FPS	7.51	14.93	Observed 14.4–15.6	D3 range
Median camera FPS	7.50	14.90	Not available	–

Metric	D1: ASUS TUF	D2: HP Victus	D3: HP Notebook	Notes
Mean face detection	9.53 ms	27.67 ms	Observed 7.1–44.4 ms	D3 range
P95 face detection	15.38 ms	40.39 ms	Not available	–
Cursor processing	0.14 ms mean	0.23 ms	Observed 0.07–0.38 ms	–
VAD dropped counter	0	0	0 in supplied rows	Session-specific
Audio overflows	0 in this run	0 in this run	Not verified	Other tests showed events
Command counter	28; 1 unknown	21; 7 unknown	Not available	Not ASR accuracy
aximum end-to-end latency	3267 ms	5965 ms	Not available	Long-tail delays

counter was cumulative and the test was not controlled, this value describes that log and does not establish transcription ac-curacy. Maximum end-to-end latency was 5965 ms, including a command-dispatch outlier of approximately 4243 ms.

For D3, the supplied rows show camera rates of approxi-mately 14.4–15.6 FPS, face-detection times of approximately

7.1–44.4 ms, and cursor-processing times of approximately 0.07–0.38 ms. The provided VAD dropped counter remained at zero. Since a verified aggregate summary was unavailable, only observed ranges are reported.

Across the available runs, cursor-coordinate processing was short compared with face detection and speech processing. However, a small cursor-processing duration does not imply low end-to-end latency. The multi-second delays show that ASR, command dispatch, scheduling, or other stages can dominate response time. The supplied logs do not isolate every source of delay, so stage-level timestamps are needed before assigning a cause.

The device results should be read as snapshots of appli-cation behavior rather than as a ranking of the laptops. The sessions were not matched for workload, camera settings, or background processes. A meaningful comparison would require the same application version, camera resolution, model configuration, power mode, and test script on every system.

The speech counters also have different meanings. A VAD segment indicates that an audio region passed the activity detector; it does not establish that the words were transcribed correctly. A Whisper event records transcription activity, while a command event records downstream handling. An unknown command can arise from a transcription error, a phrase outside the supported vocabulary, or the matching policy. Conse-quently, the unknown-command rates should not be treated as recognition error rates.

Maximum end-to-end latency is important for interaction design. A system may move the pointer smoothly but still feel slow when a spoken command takes several seconds to execute. Reporting median, 90th- or 95th-percentile, and maximum latency would show typical behavior and long-tail stalls. Future instrumentation should timestamp audio capture, speech endpoint detection, transcription completion, command matching, confirmation, and the final desktop action using a consistent clock.

7. Discussion and Limitations

The results show that both input pipelines can operate together on consumer Windows laptops, but they do not es-tablish a uniform interaction rate. Camera FPS varies between sessions, and observed speech-command latency has a long tail. Future profiling should record synchronized timestamps for frame capture, landmark inference, VAD segmentation, transcription, command matching, and operating-system dis-patch. Queue occupancy and CPU utilization would help identify bottlenecks.

The unknown-command counters require careful interpreta-tion. Without reference transcripts and a labelled test set, they cannot establish speech-recognition accuracy. Similarly, zero VAD drops in one session does not guarantee robustness in longer sessions or noisy environments.

The mouth-distance threshold is computationally simple, but speaking, camera angle, and facial geometry may affect the observed distance. Suppression during speech addresses one likely source of accidental clicking, but false-positive and false-negative rates remain unknown. A future gesture evaluation should include intentional mouth opening, ordinary speech, silent mouth movement, and clicking near VAD state transitions.

Cursor control has a stability–responsiveness trade-off. Smoothing can reduce jitter but may delay intentional motion. A structured evaluation should measure stationary cursor vari-ation, target acquisition time, and error under fixed camera placement and lighting. Fitts' law provides a conventional framework for studying pointing performance, but throughput should only be reported after a suitable target-acquisition experiment [21]. A user study should also consider comfort, fatigue, learnability, and suitability for people with different motor abilities.

A. Operational Boundaries and Deployment

The offline design reduces routine dependence on cloud transcription, but it does not eliminate all privacy or security risks. Privacy also depends on operating-system permissions, local logging, model distribution, and handling of saved recordings or transcripts. A deployment version should state whether audio is stored, provide an obvious microphone-active indicator, and avoid retaining raw media unless necessary and authorized by the user.

Desktop automation introduces another boundary because recognized phrases can trigger actions affecting windows, files, or power state. The command list should be explicit and limited to necessary operations. Actions with a high cost of error should require confirmation, and a visible stop control should be available if recognition or pointer behavior becomes unpredictable. Testing should examine accidental activation and recovery, not only successful command execution.

Accessibility requires more than an alternative input mech-anism. Users differ in range of motion, facial control, speech clarity, fatigue, and tolerance for repeated calibration. A threshold that works for one person may be unsuitable for another. The current prototype has not been evaluated suffi-ciently to establish suitability for a particular clinical popula-tion. Evaluation with intended users, under appropriate ethical safeguards, is needed before making claims about clinical benefit.

B. Threats to Validity

The three sessions were collected on different computers and were not run with a standardized workload. Differences in frame rate and timing may reflect the machine, camera, session conditions, or background activity. The available logs do not provide every stage timing for all devices, so the cause of each delay cannot be identified. Command counters lack reference transcripts and a predefined test set, and no independent partic-ipants or standardized accessibility measures were used. The findings should therefore be interpreted as an implementation report and a basis for further testing, not as a comparative usability result.

Future work should use a repeatable test script, record software and model versions, fix camera and audio settings, and run repeated sessions on each configuration. Reports should state the number of trials and missing observations for each metric.

C. Recommended Validation Protocol

A follow-up evaluation should separate component tests from end-to-end tasks. For the visual component, test holding the head still, moving toward targets in each screen quadrant, returning to neutral, and repeating mouth-open and mouth-closed transitions. Camera distance, frame size, lighting, and seating position should be recorded and controlled within each test block.

For speech, each supported command should be spoken repeatedly at a natural pace under defined background-noise conditions. The evaluation should retain the reference phrase, recognized transcript, command-handler result, and times-tamps for VAD start and end, transcription completion, and action dispatch. Unsupported phrases and near-matches should be included to test whether unintended commands are rejected. Dictation quality should be evaluated separately from com-mand execution.

Multimodal tests should include speaking while moving the pointer, opening the mouth without speaking, speaking without intending a click, and beginning speech while the mouth-click state is active. Measures should include unintended clicks, missed intended clicks, command completion rate, and end-to-end response time. A pointing study should define target sizes, distances, trial order, and error handling before any Fitts' law throughput is reported.

8. Conclusion

This paper described Face Gesture Assistant, an offline multimodal desktop interaction prototype combining facial-landmark pointer control with local speech commands and dictation. Its implementation uses a single Qt application pro-cess with separate worker threads, a Euclidean lip-landmark distance for mouth clicking, Silero VAD, and Faster-Whisper CPU inference. Runtime observations from three Windows laptops show that cursor-coordinate processing is inexpensive relative to other stages, while camera frame rate and end-to-end latency vary between runs. These measurements are preliminary engineering evidence, not a formal accessibility or usability validation. Next steps are to instrument pipeline latency, evaluate gesture and command errors against labelled test cases, and conduct a repeatable pointing and user study before making stronger claims about reliability or usability.

References

[1] M. Betke, J. Gips, and P. Fleming, "The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1–10, 2002, doi: 10.1109/TNSRE.2002.1021581.

[2] J. Tu, T. Huang, and H. Tao, "Face as mouse: Controlling mouse move-ment using facial features," *Computer Vision and Image Understanding*, vol. 108, nos. 1–2, pp. 35–40, 2007, doi: 10.1016/j.cviu.2006.11.007.

[3] A. Johnston and P. de Chazal, "A review of facial landmark localization techniques," *EURASIP Journal on Image and Video Processing*, vol. 2018, Art. no. 86, 2018, doi: 10.1186/s13640-018-0324-4.

[4] C. Lugaresi *et al.*, "MediaPipe: A framework for building perception pipelines," arXiv:1906.08172, 2019. [Online]. Available: https://arxiv.org/abs/1906.08172.

[5] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile GPUs," arXiv:1907.06724, 2019.

[6] G. Casiez, N. Roussel, and D. Vogel, "1-Euro: A simple speed-based low-pass filter for noisy input in interactive systems," in *Proc. ACM Conference on Human Factors in Computing Systems*, 2012, pp. 2527–2530, doi: 10.1145/2207676.2208639.

[7] K. Christian, B. Kules, B. Shneiderman, and A. Youssef, "A com-parison of voice-controlled and mouse-controlled web browsing," in *Proc. ACM Conference on Assistive Technologies*, 2000, pp. 72–79, doi: 10.1145/354324.354345.

[8] A. Pradhan, K. Mehta, and L. Findlater, "Accessibility came by accident: Use of voice-controlled intelligent personal assistants by people with disabilities," in *Proc. ACM CHI*, 2018, doi: 10.1145/3173574.3174033.

[9] A. Radford *et al.*, "Robust speech recognition via large-scale weak supervision," in *Proc. 40th International Conference on Machine Learning*, PMLR, vol. 202, pp. 28492–28518, 2023. [Online]. Available: https://proceedings.mlr.press/v202/radford23a.html.

[10] SYSTRAN, "Faster-Whisper: Faster Whisper transcription with CTranslate2," [Online]. Available: https://github.com/SYSTRAN/ faster-whisper.

[11] S. Oviatt and P. Cohen, "Multimodal interfaces that process what comes naturally," *Communications of the ACM*, vol. 43, no. 3, pp. 45–53, 2000, doi: 10.1145/330534.330538.

[12] Silero Team, "Silero VAD: Pre-trained voice activity detector," [Online]. Available: https://github.com/snakers4/silero-vad.

[13] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[14] World Wide Web Consortium, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, 2023.

[15] The Qt Company, "Signals and slots," Qt Documentation. [Online]. Available: https://doc.qt.io/qt-6/signalsandslots.html.

[16] M. A. Sweigart, "PyAutoGUI documentation," [Online]. Available: https://pyautogui.readthedocs.io/.

[17] R. C. Simpson, H. S. Gauthier, and A. Prochazka, "The use of tooth-clicks as a trigger for assistive computer access," *Neurorehabilitation and Neural Repair*, 2010, doi: 10.1177/1545968309341647.

[18] M. MacLellan *et al.*, "Camera Mouse: A case study of an acces-sible computer access technology," *Assistive Technology*, 2024, doi: 10.1080/10400435.2023.2242893.

[19] "Low-cost human–machine interface for computer control with facial landmark detection and voice commands," *Sensors*, vol. 22, no. 23, Art. no. 9279, 2022, doi: 10.3390/s22239279.

[20] The Qt Company, "QThread," Qt Documentation. [Online]. Available: https://doc.qt.io/qt-6/qthread.html.

[21] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954, doi: 10.1037/h0055392.

[22] International Organization for Standardization, *ISO 9241-411:2012 Er-gonomics of human-system interaction—Part 411: Evaluation methods for the design of physical input devices*. Geneva, Switzerland: ISO, 2012.

[23] M. Weidinger, "python-sounddevice: Play and record sound with Python," documentation. [Online]. Available: https://python-sounddevice.readthedocs.io/.

[24] Google, "Face landmark detection guide," MediaPipe documenta-tion. [Online]. Available: https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker.

[25] J. P. Bigham *et al.*, "The design of human-powered access technology," in *Proc. ACM SIGACCESS Conference on Computers and Accessibility*, 2010.