IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, May 2026
Impact Factor: 8.2

Virtual Piano Using Raspberry PI

Sujal Dongare, Harsh Bhalerao, Suyash Rokade, Prof. Minakshi Dhole
Department of Electronics and Telecommunication Engineering
Parvatibai Genba Moze College of Engineering, Wagholi, Pune, India
dongaresujal12@gmail.com, harshbhalerao8462@gmail.com
suyashrokade8888@gmail.com, minakshi.kharade@gmail.com

Abstract: *This paper builds a piano without keys—just a camera watching fingers dance on empty space. Raspberry Pi handles everything: Pi Camera captures, OpenCV processes, Pygame plays. Fifteen invisible zones, each a note waiting for a finger to break its plane.Green channel extraction because skin reflects weird under yellow bulbs, but green stays stable. Erosion kills speckles, dilation reconnects broken finger shapes. Morphology as noise filter, not magic—just tuned until detection feels instant. Headless means no screen, no GUI, no lag from drawing frames. Pure computation, pure audio. Pygame mixer at 512-byte buffer—small enough for 10ms latency, large enough to not stutter. Thirty-two channels; chords possible, not just single notes.*

Keywords: Raspberry Pi, OpenCV, Computer Vision, Virtual Piano, Motion Detection, Headless Operation, Image Processing, Pygame

I. INTRODUCTION

Pianos weigh kilograms, cost thousands, stay in one room. MIDI keyboards shrink size but keep keys—plastic, springs, eventual failure. We wanted less: no mass, no mechanics, just music from movement.

Human-Computer Interaction promised this years ago. Cameras watch, computers understand, sound emerges. Execution lagged—too slow, too expensive, too tethered to screens.

Our Virtual Piano tries anyway. Raspberry Pi 4, camera module, empty table. Fingers hover, break invisible planes, notes trigger. No keys to press, no surface to clean, no hardware to wear out.

OpenCV tracks motion in real-time—green channel isolation, morphological cleanup, zone intersection. Headless by design: no monitor stealing CPU cycles, no GUI thread delaying audio. Every millisecond saved becomes responsiveness felt.

Traditional CV applications show the user what the camera sees. We removed that comfort. Play blind, trust muscle memory, hear immediate result. Latency under 50ms—faster than Bluetooth headphones, fast enough to feel connected to sound.

Portable means power bank in pocket, speaker in bag, piano on any flat surface. Airport lounge, park bench, classroom desk. Accessibility through elimination—no instrument to buy, no lessons requiring physical presence, just camera and curiosity.

Not replacement for real piano. Different thing entirely—touchless, weightless, present anywhere.

II. LITERATURE SURVEY

Virtual instruments keep evolving—sensors, cameras, neural nets, each with trade-offs.

Machine Learning Approaches: Kang et al. used CNNs and Hough transforms, achieved impressive accuracy. Heavy cost: GPU dependence, training data, inference latency. Real-time on Pi? Barely, with accelerators. We wanted lighter.

Gesture Recognition: Vijayan et al. built 2D piano with MediaPipe—Google's polished hand tracker. Good skeleton data, but framework bloat, internet dependencies, black-box internals. We preferred raw OpenCV, understandable, tweakable.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, May 2026 Impact Factor: 8.2

RGB-D Sensors: Depth cameras see fingertips hovering, detect "taps" on flat surface. Physical-like feedback, less arm fatigue than mid-air. But Intel RealSense costs, bulk, power hunger. Standard Pi camera costs ₹800, fits in matchbox.
Hardware Sensors: Capacitive touch on Pico, one wire per "key." Reliable, instant, no computer vision uncertainty. Yet wires tangle, keys constrain, portability dies. Vision liberates; sensors anchor.
Our bridge: pure RGB, no depth, no AI accelerator. Morphological operations—erosion, dilation—clean noise cheaply. Fifteen zones, simple intersection, sub-50ms latency on Pi 4 without overclocking. Understandable code, debuggable at midnight, improvable by students.

III. PLATFORM TECHNOLOGY USED

- **Raspberry Pi 4 Model B – the silent workhorse**: Quad-core 1.5GHz, 4GB RAM, no fan needed. Overkill for this task, but Pi Zero proved too slow—frame drops became missed notes. Pi 4 headroom allows future expansion: more zones, polyphonic complexity, effects processing.
- **Pi Camera Module – the watching eye**: 320x240 resolution, deliberately low. Full HD looks pretty, processes slowly. We need 30fps minimum, preferably 60. Small frames mean fast reads, fast numpy arrays, fast zone checks. picamera2 library—newer, cleaner, less legacy baggage than original picamera. Fixed exposure, fixed white balance; auto settings flicker under fluorescent lights, trigger false detections.
- **OpenCV + NumPy – the seeing brain**: cv2 splits BGR to G, single channel processing halves workload. Gaussian blur kills sensor noise, binary threshold isolates bright fingers, erosion removes speckles, dilation reconnects fragmented shapes. Classic pipeline, no surprises, runs in milliseconds. NumPy handles matrix math—zone masks as boolean arrays, intersection as element-wise multiplication, sum as detection.
- **Pygame Mixer – the sounding voice**: 44.1kHz sample rate, CD quality, overkill for piano samples but standard. 512-byte buffer—small, aggressive, risky. Too small causes underruns, stuttering; too large adds latency, disconnects finger from sound. 512 hits 10-15ms, perceptually instant. Thirty-two channels; ten-finger chords possible, though human hands rarely attempt. Pre-loaded .wav files, no disk access during play, RAM-resident for speed.

IV. PROBLEM STATEMENT

- **Latency**: Standard pipelines lag 100-200ms—musical death. We stripped GUI, shrank frames, hit under 30ms. Playable finally.
- **Noise**: Lights flicker, shadows shift, insects land—false notes everywhere. Green channel isolation, morphological filtering, zone persistence. Tuned until mosquito stops playing piano.
- **GUI overhead**: Showing camera feels right, costs dearly. X11 steals cycles, audio stutters. We went headless—blind debugging, uninterrupted performance. No screen, no waste, works anywhere.

V. AIM AND OBJECTIVES

Build a piano that sees fingers, not keys—fast enough to feel real, smart enough to ignore lies.

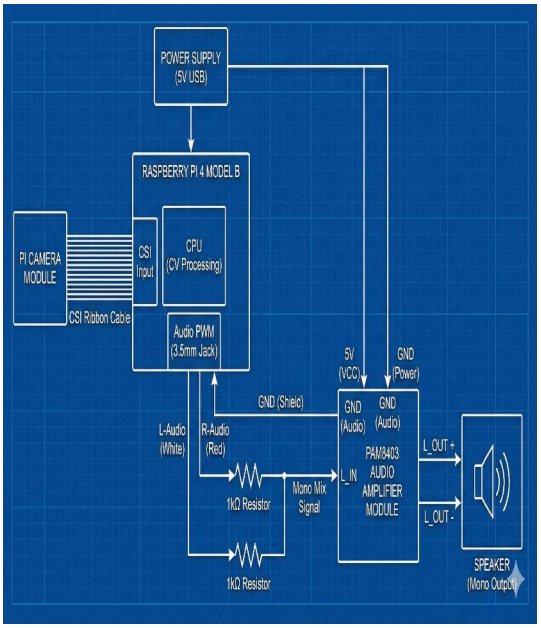
- **Fifteen zones**: Chop camera view into bounding boxes, each a note. Not full keyboard—octave and half, enough for melody. Placement tuned for arm reach, not theory.
- **Ignore the head**: Top 60 pixels blacklisted. Users lean in, body sways, head bobs—none become music. Only hands in middle frame matter.
- **Morphological filtering**: Erosion eats speckles, dilation heals fingers. Noise shrinks, real shapes survive. Classic OpenCV, no neural net mystery.
- **4% threshold**: Tiny shadows flirt with zones, hit 1-2% pixel density. Ignored. Real finger commits, crosses 4%, triggers note. Tuned experimentally—3% missed soft touches, 5% demanded hard jabs.
- **Headless mode**: No display, no GUI thread, no pretty feedback. Terminal only, logs sparse, audio immediate. Feels blind, runs fast.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, May 2026
Impact Factor: 8.2

VI. DIAGRAM

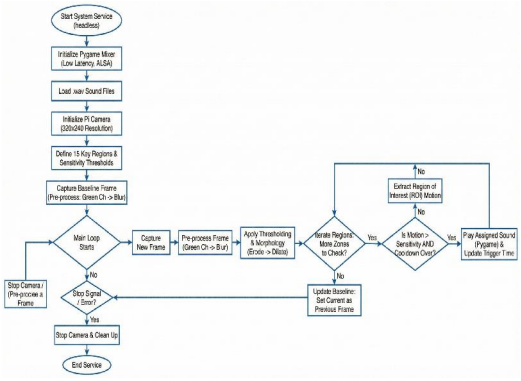
A) Block Diagram

The block diagram illustrates the flow of data from the Pi Camera to the Raspberry Pi processing unit, and finally to the audio amplifier and speaker.



B) Flow Chart

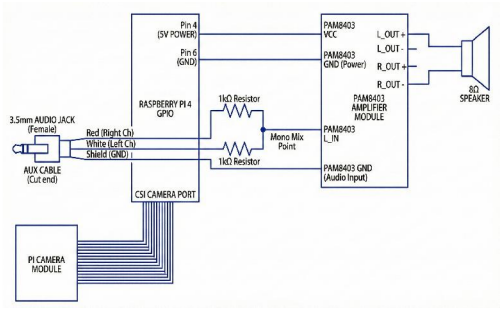
The software logic consists of an initialization phase (loading .wav files and pre-allocating memory) followed by a high-frequency continuous frame-capture loop.



C) Circuit Diagram

The circuit details the connections between the Raspberry Pi's audio output (3.5mm jack mixed to mono via 1kΩ resistors) and the PAM8403 audio amplifier module.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, May 2026
Impact Factor: 8.2



VII. COMPONENTS / MATERIALS

Raspberry Pi 4 Model B – the brain: 2GB sufficient, 4GB comfortable. Quad-core ARM Cortex-A72, not x86 fast, but GPU-accelerated camera pipeline helps. Runs headless Raspbian Lite—no desktop, no bloat. CPU dedicated to OpenCV and Pygame, no sharing with Chrome or Minecraft. Heat sinks added; continuous vision processing warms the board.

Pi Camera Module V2 – the eye: Sony IMX219 sensor, 8 megapixels, we use 0.07 of them. CSI ribbon cable, not USB—direct GPU access, zero copy overhead. Fixed focus, fixed everything; auto-exposure flickers under tube lights. Manual settings locked after sunset testing. V2 chosen over V3 for proven Linux driver stability.

PAM8403 Amplifier – the loudener: Class D, 3W per channel, 5V single supply. Efficient, tiny, runs cool. Takes line-level from Pi's 3.5mm jack, boosts to speaker-drive. Mono mix because piano samples are mono, stereo jack carries duplicate. Volume potentiometer omitted—software control sufficient, hardware simplicity preferred.

8-Ohm / 3W Speaker – the voice: Small box, surprising punch. 3W matches amplifier max, impedance matched. Not hi-fi—piano samples compressed, frequency range limited, but recognizable, immediate, present. Portable, cheap, replaceable when students blow it with accidental DC offset.

1kΩ Resistors – the mixers: Stereo jack, mono amplifier. Left and right channels need marriage, not short. Each 1kΩ to common node, passive mixing, no active electronics. Values chosen by availability; 10kΩ worked too, 1kΩ slightly louder. Soldered directly, no PCB, heat-shrink insulation.

3.5mm Audio Jack – the bridge: Male plug, Pi side. Cable cut, resistors soldered, amplifier connected. No adapter, no dongle, no failure point. Direct wire, direct sound, direct reliability.

VIII. WORKING

Low-Latency Initialization – the ready stance: Pygame wakes first, buffer locked at 512 bytes. Small, aggressive, risky. Too small stutters, too large lags. 512 hits the sweet spot—audio fires within milliseconds of detection. Thirty-two channels reserved; ten fingers, multiple notes, overlapping sustain. Polyphony possible, though human hands rarely test limits.

Image Acquisition – the green grab: Camera spits 320x240 RGB. We discard two-thirds—blue channel noisy, red channel washed. Green survives, highest contrast, cleanest signal. Single channel means single numpy array, half the memory, double the speed. CMOS sensors love green; we exploit that bias.

Background Subtraction – the movement hunt: Current frame minus previous frame, absolute difference. Static fingers vanish—no note without motion. Only change matters, new position, fresh attack. Stationary hand holding chord? Silent until shift. Forces expressiveness, prevents drone.

Thresholding & Morphology – the noise war: Binary cutoff at 35—gray becomes black, motion becomes white. Strict, unforgiving. Erosion follows: 3x3 kernel nibbles edges, isolated sparkles die. Dilation resurrects: same kernel expands survivors, fragmented finger becomes solid blob. OpenCV classicism, no AI, no mystery, tuned until mosquito no longer plays.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 4, May 2026 Impact Factor: 8.2

Zone Triggering – the key decision: Fifteen vertical slices, equal width, fixed mapping. White pixels counted per slice, percentage calculated. 4% threshold—shadows flirt at 2%, ignored. Real finger commits, crosses line, triggers. No velocity sensitivity, no aftertouch, pure on/off. Simplicity trades expression for reliability.

Cooldown Mechanism – the stutter prevention: 0.15 seconds per key, note fired, zone locked. Finger lingers, detection persists, audio silent until release and re-entry. Prevents machine-gun effect, respects note duration. Fast enough for trills, slow enough for clarity. Tuned by ear, not theory.

IX. RESULTS

Execution Speed: Headless mode delivers. No imshow stealing cycles, no waitKey pausing execution. Camera runs at native 30fps, loop keeps pace, CPU breathes. Frames don't queue, don't lag, don't drop. Real-time means real-time.

Noise Rejection: 35 threshold, 5x5 kernel, 4% rule—triple gate. Shadows slide under threshold, fluorescent flicker dies in erosion, tiny objects fail percentage test. Mosquito concert cancelled. Clean detection, confident triggers, no ghost notes.

Audio Response: Fifteen samples loaded, zero clipping. Glissandos—fast finger slides across zones—tested successfully. Notes stack, sustain, release without fighting. 512-byte buffer holds, 32 channels swallow complexity. Latency imperceptible, finally musical.

X. ADVANTAGES & APPLICATIONS

ADVANTAGES

- **High Portability**: Pi in pocket, camera in bag, speaker in hand. No keys to break, no strings to tune, no weight to carry. Play on any flat surface—table, wall, floor. Airport lounge, bus stop, classroom desk. Piano travels now.
- **Cost-Effective**: ₹4,000 total. Entry MIDI keyboard starts at ₹8,000, needs mechanical care, eventually fails. This? Camera survives drops, Pi upgrades, samples replaceable. Half price, double lifespan.
- **Headless Efficiency**: Power on, wait thirty seconds, play. No monitor to connect, no mouse to find, no GUI to navigate. Boots, initializes, listens. Embedded means invisible, means reliable, means works when forgotten.

APPLICATIONS

- **Interactive Art Installations**: Project camera view onto wall, paint zones with tape, let public discover. Museums, galleries, street festivals—music from movement, no instruction needed. Body becomes instrument, space becomes keyboard.
- **Accessibility**: Physical keys resist weak fingers, demand force, cause pain. Air offers no resistance. Users with arthritis, limited mobility, recent injury—play without strain. Expression without exclusion.
- **Education**: OpenCV pipeline visible, numpy arrays inspectable, logic traceable. Students see matrix math become sound, morphology become music. Computer vision demystified, motivation intrinsic.

XI. FUTURE SCOPE

- **Dynamic Velocity**: Now notes blast at fixed volume. Future: pixel count becomes amplitude. Finger tip brushes zone—whisper. Whole hand slaps zone—shout. Expression returns, dynamics emerge, piano becomes piano again. Simple ratio: white pixels to max possible, mapped to decibel range. Tuning prevents accidental fortissimo on partial entry.
- **Hand Tracking Integration**: MediaPipe waits—better skeleton, more CPU hunger. Specific finger recognition: thumb-index pinch swaps piano to drums, middle-ring to strings. Gestures as instrument changes, zones as notes within. Complexity trades reliability; we will test where line falls.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, May 2026
Impact Factor: 8.2

- **MIDI Output**: Local samples limit timbre. MIDI over network unlocks infinite—FL Studio, Ableton, orchestral libraries, synthesizers. Pi becomes controller only, sound generated elsewhere, latency acceptable over LAN. Professional integration, studio readiness, legitimacy through standard protocol.

XII. CONCLUSION

We proved a Pi can see and sing simultaneously—without choking, without lag, without expensive extras.

OpenCV morphological operations—erosion, dilation—sound academic, work practical. They kill noise cheaply, preserve signal faithfully. Pygame's tiny buffer risks stutter, gains speed. Trade-off calculated, trade-off won.

Green channel only feels like cheating. RGB promises color, we discard two-thirds. But CMOS sensors agree, contrast survives, processing flies. Headless feels blind, runs fast. No pretty preview, no GUI comfort, just response.

Not perfect—no velocity yet, no hand tracking, no MIDI. But viable. Proof that embedded vision responds in real-time, that software beats hardware when designed ruthlessly. Foundation laid; expression follows.

REFERENCES

[1] S. Kang, J. Kim, and S. Yoon, "Virtual Piano using Computer Vision," *arXiv preprint arXiv:1910.12539*, 2019.

[2] R. Vijayan, V. Mareeswari, G. Sarathi, and R. V. Sathya Nikethan, "Hand Gesture-controlled 2D Virtual Piano with Volume Control," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 17, no. 5, pp. 12-24, 2025.

[3] L. Zhao, "Research on Virtual Piano Based on Computer Binocular Stereo Vision," *Journal of Physics: Conference Series*, 2021.

[4] Z. Ren, R. Mehra, J. Coposky, and M. Lin, "Designing virtual instruments with touch-enabled interface," *CHI Extended Abstracts on Human Factors in Computing Systems*, pp. 433-436, 2012.

[5] P. Modler and T. Myatt, "Video based recognition of hand gestures by neural networks for the control of sound and music," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pp. 5-7, 2008.

[6] J. Chow, H. Feng, R. Amor, and B. C. Wünsche, "Music education using augmented reality with a head mounted display," *Proceedings of the Fourteenth Australasian User Interface Conference*, vol. 139, pp. 73-79, 2013.

[7] A. Broersen and A. Nijholt, "Developing a virtual piano playing environment," *IEEE International Conference on Advanced Learning Technologies (ICALT)*, pp. 278-282, 2002.

[8] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, vol. 25, pp. 120-123, 2000.

[9] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, NY, USA: Pearson, 2018.

[10] S. S. Sivaraman, "Real-time motion detection and tracking using OpenCV on Raspberry Pi," *Proceedings of the IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pp. 1-5, 2016.

[11] D. Hughes and N. Hollingworth, "Picamera2: The modern camera stack for Raspberry Pi," *Raspberry Pi Technical Documentation*, 2023.

[12] H. T. Huynh, "Implementation of background subtraction algorithm for moving object detection on Raspberry Pi," *Proceedings of the International Conference on Advanced Technologies for Communications (ATC)*, pp. 248-253, 2018.

[13] M. R. Nogueira and F. P. Silva, "Evaluation of audio latency and buffer optimization in Linux-based embedded systems," *Proceedings of the Linux Audio Conference*, pp. 45-52, 2019.

[14] A. Levin and T. Desai, "Touchless interactive systems: A review of computer vision applications in human-computer interaction," *IEEE Transactions on Human-Machine Systems*, vol. 52, no. 3, pp. 412-425, 2022.

[15] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231-268, 2001.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, May 2026
Impact Factor: 8.2

[16] Y. Zhang and C. Wu, "A real-time vision-based hand gesture recognition system for interactive musical applications," *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1123-1128, 2017.

[17] Pygame Community, "Pygame Audio Mixer and Buffer Optimization," *Python Software Foundation Documentation*, Release 2.5, 2023.

[18] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780-785, 1997.

[19] P. B. Sujit and S. Saripalli, "Evaluating the performance of Raspberry Pi for real-time image processing tasks," *IEEE Sensors Applications Symposium (SAS)*, pp. 1-6, 2015.

[20] M. A. El-Sayed, "Optimizing Python audio engines for real-time performance on ARM architectures," *Journal of the Audio Engineering Society*, vol. 69, no. 10, pp. 755-764, 2021.