Human Factors in Training, Education, and Learning Sciences, Vol. 231, 2026, 150–159
https://doi.org/10.54941/ahfe1008006

Evaluating Coloured Blob Tracking, CNN and Posture Detection Computer Vision Models on Latency and Accuracy in the Application of a Virtual Drum Kit

Kenneth Y. T. Lim[1] and Joshua Ze Quan Lee[2]

[1]Nanyang Technological University (NTU), Singapore 639798, Singapore
[2]Independent

ABSTRACT

Drumming is often inaccessible to beginners due to high costs, space constraints, and noise. While commercial "drumless" virtual systems exist, they rely on expensive hardware like VR goggles, infrared cameras, and motion-sensor drumsticks. These solutions remain costly and inconvenient, failing to address the core issue of accessibility. This paper proposes a low-cost computer vision (CV)-based virtual drum system that works on minimal hardware, such as a consumer laptop with a standard webcam. This study investigates the effectiveness of a range of computer vision techniques for real time drumstick and foot tracking, along with reliable event detection for a virtual drum kit. Four models were implemented using different combinations of established CV methods, including coloured blob tracking, Kalman filtering for motion prediction, dynamic region of interest (ROI) scaling, posture tracking using the MediaPipe pose estimation framework, and convolutional neural network (CNN) based drumstick detection using the YOLOv8 object detection model. All models were designed to track two drumsticks and two knees and detect the corresponding events. Each model was bound to identical hardware constraints. Performance evaluation in this study focuses on two primary metrics: accuracy, measured using precision, recall, and F1-score derived from false positives and false negatives produced during drumming sequences; and latency, measured as the temporal difference between the actual moment a drum hit occurs and when the system registers that event. These metrics were chosen because they directly impact the user experience in a real-time virtual musical instrument.

Keywords: Computer vision, CNN, Drums, Virtual drum kit, Posture detection, Machine learning, Deep learning, Object tracking, Event detection

INTRODUCTION

Traditional drum sets, acoustic and electronic, make drumming inaccessible to many due to its high cost, high space consumption and the production of high noise levels. As such, owning a drum set and practicing at home is greatly inefficient and unfeasible. This has motivated research into virtual drumming systems without requiring costly, bulky or noisy physical hardware.

There currently exist several commercial implementations of "drumless drums". Examples of these products include Paradiddle (Paradiddle,

Received February 5, 2026; Revised April 3, 2026; Accepted April 23, 2026; Available online July 20, 2026

2024), Pocket Drums (AeroBand, 2022) and Aerodrum (Aerodrums, n.d.). Paradiddle is a virtual reality (VR) drum application that requires specialised hardware such as VR goggles and controllers. Pocket Drums utilises gyro/motion sensors within custom sticks and pedals that require charging to detect the position and motions of the users sticks and feet, and costs USD$179. Aerodrums utilises wearable light reflectors for the feet, specialised drumsticks with reflective markers and a high-speed infrared sensor for tracking of the 4 reflective points of interest and costs USD$799 (Aerodrums, n.d.). Thus, the current commercial products of "drumless drums" are seen to be imperfect solutions as they all require highly specialised hardware, are not portable, inconvenient and are costly. Consequently, these products might fail to make drumming more accessible to beginner drummers.

This gap forms the motivation for this paper, to investigate computer vision methods and develop a hybrid algorithm for a computer vision virtual drum set that can run on minimal hardware, utilizing an average laptop and its webcam.

Related Work

We explore the feasibility of CV drums based on previous implementation techniques and their various performances. These approaches generally follow a common pipeline of 3 stages:

1. Object Detection and Tracking
2. Event Detection (When a drum has been struck)
3. Sound Synthesis

Although most implementations follow these 3 stages, they often differ in the way they are implemented.

Object Detection and Tracking

There are typically 4 points of interest to track: left drumstick, right drumstick, left foot/knee (hi-hat control), right foot/knee (bass drum), though past research often differs in the number of markers tracked. Some common techniques for the tracking of drumsticks and feet/knee movement include:

1. Coloured blob detection (Tolentino, Uy and Naval, 2019)
    - For each of the pre-set/calibrated colours (one per stick/foot), select the largest group of pixels of each of the colours, dilate them to be more blob-like and find the centre as the tracked keypoint. This requires the user to use markers of the correct colours (e.g. coloured post-it notes or tape) on the points to be tracked.
2. Kalman Filter (Kalman, 1960) for predicted position
    - Under the assumption that the camera is placed directly in front of the drummer, a linear second order model is used to describe the movement of the drumstick tips in the x-y plane (forward and backward displacement in the third axis is neglected) to can track its upwards and downwards acceleration and velocity. The current position of the marker can then be calculated as a prediction based on the confirmed position of the marker in the previous time step

(one frame prior), its velocity and its acceleration. The predicted position for the frame can then be used for the checking of an event occurrence.

3. Dynamic Region of Interest (ROI) (Fidan et al., 2010)
   - This technique is used to localise the ROI within each frame such that the computational weight of processing each frame is reduced to just within these relevant windows. Once the keypoint for the centre of a marker is detected, a window of preset dimensions is formed around it. The predicted position of the marker in the next frame is calculated from the Kalman Filter. If the predicted position of the marker is outside of the window, the window dimensions will be scaled up according to the velocity and if the marker cannot be detected within the window, the whole image will be used for processing and the ROI is reset.
4. Posture/Body Tracking (Hong, Stephen and Kenji, 2024)
   - In addition to tracking markers, one technique is to instead use a posture detection algorithm such as MediaPipe (Google for Developers, n.d.) to track landmarks on the drummer's body. One example would be to track the angle of the drummer's knee joint, angle of elevation from foot and ankle, or velocity of knee/foot to detect whether the left foot is "stepping on" the hi-hat pedal or the right foot has kicked the bass drum pedal.
5. Trained/Fine-tuned Convolutional Neural Networks (CNN) for Detecting Drumsticks (Harel Yadid et al., 2023)
   - CNN models such as YOLOv8 or YOLOv5 are fine-tuned with images of drummers holding drumsticks, with the key markers labelled, under different lighting conditions and orientations and resolutions. The model is then used to detect on each frame in real-time the location of the key markers.

Past works have used such techniques or a combination of them to track 2-4 of these significant landmarks.

Event Detection

The significant event to check for each frame is whether a "drum" has been struck. In previous implementations, there are several criteria (or combination of them) to be checked before outputting the Boolean signal.

1. Bounding boxes/points comparison
   - The predefined zones/boxes are defined on the screen and should the detected (or in some implementations, the predicted) position of the marker crosses the boundary
2. Vertical velocity downwards
   - Since we only want the downstrokes to be detected, there must be a condition to check that the drumstick was indeed travelling downwards. This can be computed by checking the previous position to its current, or using the velocity value being tracked within the state vector for the Kalman Filter

3. Acceleration exceeds threshold
   - The point where the drumstick "contacts" the drum would cause the change in direction of the velocity and the lowest position of the stick should correspond to the peak negative acceleration, as the stick comes to a momentary stop. To accurately pick out this position, a pre-set threshold is introduced to ensure that the magnitude of the negative acceleration was large enough.
4. No hit in x time steps before (dependent on framerate)
   - Since multiple frames within several time steps where the stick is in the down position can remain in the bounding box, this criterion is introduced to ensure there is no repeated duplicate triggering and detection of sound for each downstroke.
5. Foot/Knee Movement (for posture tracking models)
   - The y-position corresponding to the height of the foot and knee can be tracked to determine whether it is in the up or down position, movement between the two states can be considered as kicking of the bass drum for the right foot or opening or closing the hi-hat for the left foot.

Sound Synthesis

Upon detection of an event, the corresponding sound depending on each box/marker will be synthesized, with the volume proportional to the velocity of the drumstick in a few frames before. This stage is relatively trivial and does not introduce much latency and as such its techniques will not be explored as in depth in this paper.

Difference in Models

Apart from differing in techniques of implementations, these previous studies also differ in metrics used for evaluating model performance, the number of significant landmarks tracked and hardware requirements. Models are evaluated on a variety of metrics such as precision and recall (penalising false positives and negatives), qualitative reviews from users of differing backgrounds, maximum number of hits per minute, latency and buffer time. The different models' robustness also suffers differently from points of errors such as too bright or low lighting or the impact of noise (other objects in the room, background colours etc.). Some models only track the drumsticks (2 landmarks), while others may additionally track the right or left knee as well. Furthermore, each model was run on differing hardware with different processing speeds, with some also requiring expensive specialised hardware such as high-speed cameras to run. As such, it is impossible to evaluate through comparisons of these findings which techniques are the most efficient and applicable for the use case of a computer vision drum set.

This paper aims to compare techniques (or combinations of them) discussed above. This is not a systematic review and as such, implementations may not be exactly as in the cited works. These models will be evaluated on a common base of metrics, with strengths and weaknesses analysed.

METHODOLOGY

Hardware and Model Requirements

All models were evaluated on identical hardware to ensure fair comparison. Experiments were conducted on an HP ProBook 440 G7 with an Intel® Core™ i7-10510U CPU. Video input was captured using an external webcam at a resolution of 720p and a frame rate of 60 frames per second.

The physical setup assumes a frontal camera view, positioned approximately 1.5 m from the subject, with both knees fully visible within the frame. The snare drum region of interest (ROI) is aligned near the centre of the image plane. This configuration reflects a realistic and repeatable home-based setting, while maintaining visibility of all required landmarks.

All models were implemented in Python. Coloured blob detection and image processing operations were performed using OpenCV library. Posture and knee tracking were implemented using the MediaPipe Pose estimation framework. CNN-based drumstick detection was implemented using the YOLOv8 object detection model.

Landmark Definition and Tracking

Each model is designed to track four significant landmarks corresponding to a minimal virtual drum kit setup:

1. Left drumstick tip
2. Right drumstick tip
3. Left knee (hi-hat control)
4. Right knee (bass drum control)

Depending on the model, landmarks are detected either via colour segmentation, CNN object detection, or posture estimation. All detected landmark positions apart from the left knee are passed through a Kalman filter to smooth noisy measurements and to estimate velocity and acceleration. This is not required for the left knee as its motion dynamics are not required for its event detection, the updating of the Hi-Hat state machine is solely off its coordinates.

Event Detection

All models aim to detect and record 3 possible events: Snare Hit, Bass Drum Kick, Hi-Hat open/close (change in state). Event detection is performed on each frame.

Snare Drum Hit Detection

The snare drum ROI is represented by a predefined rectangular bounding box. The conditions for a snare drum hit event are:

1. The drumstick landmark is in the snare bounding box
2. Vertical Velocity is downwards (negative)

3. Acceleration greater than threshold
4. No Snare Hit event detected for that stick marker within previous 40ms
5. Drumstick was not in the snare bounding box in the previous frame

Bass Drum Kick Detection

The criteria for a Bass Drum Kick event are:

1. Vertical velocity is downwards (negative)
2. Acceleration greater than a predefined threshold
3. No Bass Drum Kick event detected in 40ms

Hi-Hat Open/Close Detection

Hi-hat state is represented using a fixed horizontal reference line. The left knee landmark is used to determine the hi-hat state. The criteria for a Hi-Hat open/close event is:

1. There is a change in the state from the previous frame (previous was below line, current is above, or vice versa).

Selected Models

Four models were selected for evaluation, each representing a different trade-off between computational complexity, robustness, and ease of use:

Model 1: Coloured blob tracking with Kalman filtering for all four landmarks (2 sticks + 2 knees)
Model 2: Coloured blob tracking with Kalman filtering and dynamic ROI scaling for all four landmarks
Model 3: Coloured blob tracking with Kalman filtering for drumsticks, combined with posture tracking for knees
Model 4: CNN-based drumstick detection using YOLOv8 with Kalman filtering, combined with posture tracking for knees

All models follow a similar processing pipeline to ensure fair comparison. Landmarks are predicted using a Kalman filter based on previously detected position. The filter estimates position, velocity, and acceleration in the image plane, smoothing out motion trajectories and reducing measurement/process noise and jitter. This is particularly important for snare and bass drum event detection, which relies on the sign of velocity and magnitude of acceleration. Landmark detection run and corrected using the prediction if detected. Event detection is then performed using identical criteria across all models. The predictions can be used, in place of the corrected detected coordinates, if no object is detected. Detected events are logged for subsequent accuracy and latency evaluation. The left knee landmark does not utilise the Kalman filter as its event logic depends only on positional state changes rather than motion dynamics.

Model 1 segments and detects each landmark by a distinct colour. Each frame is converted from BGR to HSV, and binary masks are generated using predefined colour thresholds. Gaussian blurring and morphological dilation

are applied to reduce noise and merge fragmented regions. For each mask, the centroid of the largest connected component is used as the detected landmark position. The Kalman filter then predicts position and velocity based on its previous state. If a valid detection is available, the prediction is corrected using the measured centroid. Otherwise, the predicted state is used. While computationally efficient, colour tracking is more sensitive to lighting variation, background colour similarity, and motion blur.

Model 2 builds on Model 1 by introducing an adaptive region of interest (ROI) mechanism to reduce computational load. After a landmark is first detected, a rectangular ROI of fixed initial size is centred around its location. In subsequent frames, colour segmentation is performed only within this ROI rather than the full frame. The predicted landmark position from the Kalman filter is used to update the ROI centre for the next frame's processing step. The size of the ROI is expanded proportional to the estimated velocity. If no landmark is detected within the ROI, the ROI is reset to the full image. This approach aims to reduce per-frame processing time while maintaining tracking accuracy. However, rapid movements can lead to missed detections, if the prediction or size scaling is not accurate enough to reliably encapsulate the landmark in the next frame.

Model 3 combines colour segmentation tracking from drumsticks with pose estimation. Drumsticks are detected and filtered as in Model 1. Knee landmarks are detected using the MediaPipe pose estimation framework. MediaPipe provides normalized body landmark coordinates, which are converted to pixel coordinates for consistency with the rest of the pipeline. This hybrid approach removes the need for coloured markers on the knees, improving usability and robustness to background noise. However, posture tracking introduces additional computational overhead and may be inaccurate depending on the robustness of the MediaPipe model, which can be affected by body parts being obscured in the image.

Model 4 replaces the colour segmentation drumstick detection with CNN detection model, YOLOv8. The model is fine tuned to detect drumstick tips using a custom dataset of approximately 2500 manually labelled images. Data augmentation, including rotation, scaling, and lighting variation, expands the effective training set to approximately 10000 images. At inference time, YOLOv8 detects drumstick positions in each frame. Detected positions are passed through a Kalman filter to smooth noise. Knee tracking is performed using the MediaPipe Pose framework, identical to Model 3. This approach should be more robust to lighting variation and background clutter. However, CNN inference substantially increases per-frame processing time, which can result in dropped frames and increased latency under real time constraints.

Metrics

The 4 models were given sample test videos at 60 bpm and evaluated on two metrics of accuracy and latency. Accuracy was measured by finding the total errors of false positives and negatives (missing hits and detecting non-existent hits). To measure the latency of the models, they are configured to simulate real time processing, skipping frames if the camera buffer fills due

to the processing time of each frame. Latency was measured by the timing errors between the actual hit and the output signal created. The number of frames skipped and time taken to process each frame are also recorded and analysed.

RESULTS

Table 1: Event detection accuracy across models (offline processing mode).

Model	TP	FP	FN	Total GT	Precision	Recall	F1 Score
1	48	2	2	50	0.96	0.96	0.96
2	48	2	2	50	0.96	0.96	0.96
3	39	4	11	50	0.91	0.78	0.84
4	7	3	43	50	0.70	0.14	0.23

Table 2: Event detection performance under real-time constraints (latency mode).

Model	TP	FP	FN	Total GT	Precision	Recall	F1-Score
1	22	10	28	50	0.69	0.44	0.54
2	26	9	24	50	0.74	0.52	0.61
3	15	4	35	50	0.79	0.30	0.43
4	2	1	48	50	0.67	0.04	0.08

Table 3: Runtime performance characteristics under real-time operation.

Model	Mean Processing Time/ms	Mean System Lag/ ms	Dropped Frames (Mean)
1	34.9	53.0	4.5
2	28.3	47.2	3.2
3	53.4	103.1	7.8
4	119.2	187.7	17.2

Accuracy Performance (Offline Mode)

Table 1 presents the event detection metrics of the four evaluated models when operating in offline mode, where every frame is processed without real time constraints. Models 1 and 2 achieved the highest performance, each reaching a precision, recall, and F1-score of 0.96, with only two false positives and two false negatives. Coloured blob tracking combined with Kalman filtering is highly consistent under ideal processing conditions.

Model 3 showed a reduction in performance, achieving an F1-score of 0.84. The increase in missed detections compared to Models 1 and 2 can be

attributed to the use of pose estimation knee tracking via MediaPipe, which can be less robust than colour tracking if parts of the body are obscured or the detection confidence of the model is just not high enough.

Model 4 exhibited the weakest accuracy, with an F1-score of 0.23 and a high number of false negatives. Despite the theoretical robustness of CNN-based detection to lighting variation and background clutter, the CPU-only execution of YOLOv8 resulted in inconsistent landmark detection and unstable tracking updates, significantly reducing event detection reliability. A possible explanation for the poor performance of this model is the lack of training to recognise motion blurs, as the drum stick tips are relatively small objects within the frame and travel at a high velocity. When such motion blur/streaks of the tips occur within the frame, the YOLO model has very low confidence or no ROI proposed for the drumstick tips and as such the tracked objects were missing for many frames.

Latency Performance (Real-Time Mode)

Table 2 presents model performance when configured to simulate real-time operation, skipping frames if processing cannot keep up with the 60fps input stream. As expected, performance degradation is observed across all models due to dropped frames.

Model 2 achieved the best real-time detection performance, with an F1-score of 0.61, outperforming Model 1 (0.54). The improvement in recall suggests that dynamic ROI scaling effectively reduces processing time per frame, allowing more frames to be processed and increasing the likelihood of capturing fast, dynamic events such as snare hits and bass drum kicks.

Model 3 maintained relatively high precision (0.79) but suffered from low recall (0.30), indicating that while detected events are generally correct, many true events are missed under real-time constraints. Model 4 performed poorly in latency mode, achieving an F1-score of only 0.08, indicating that CNN inference combined with posture tracking might not be suitable for real-time operation at 720p and 60 fps due to its high computational cost per frame.

Runtime Characteristics

Table 3 presents the runtime performance of each model under real time operation. Model 2 demonstrates the lowest mean processing time (28.3ms) and system lag (47.2ms), corresponding with its superior latency mode detection performance. Model 1, while simpler, incurs higher processing time (34.9ms) and greater frame skipping.

Models 3 and 4 exhibit significantly increased computational cost, with mean processing times of 53.4ms and 119.2ms, respectively. This results in substantially higher system lag and frame dropping, explaining their reduced recall and overall performance in latency mode. These results proves that high processing time per frame is harmful to a real time model's performance beyond just producing delayed hits, as it also causes increased frame skipping which further harms tracking and event detection.

CONCLUSION

This study evaluated four computer vision pipelines for a virtual drum kit under identical hardware constraints, focusing on both detection accuracy and real time responsiveness. Results demonstrate that traditional computer vision techniques, specifically coloured blob tracking combined with Kalman filtering, can achieve high accuracy in offline processing scenarios using only a standard webcam and laptop hardware.

Under real time constraints, performance is governed by a both computational complexity and detection reliability. Dynamic ROI scaling is shown to improve real time performance by reducing processing time and frame skipping, yielding the highest F1-score in latency mode. In contrast, posture-based tracking and CNN-based detection introduce substantial computational overhead, leading to increased latency and missed events.

Overall, the findings suggest that lightweight, classical computer vision approaches remain highly competitive and appropriate for real time musical interaction tasks, particularly when hardware resources are limited.

REFERENCES

AeroBand. (2022). PocketDrum 2 Max. [online]. Available at: https://www.aeroband.net/products/pocketdrum2-plus.Aerodrums. (n.d.). Aerodrums | Air Drums & Virtual Electronic Drum kit. https://aerodrums.com

Aerodrums. (2025). Pick your Aerodrums | Aerodrums. [online]. Available at: https://aerodrums.com/pick-your-aerodrums

Fidan, K.C., Ihsan Kehribar, Sahin, M.T., Serhan Cosar and Devrim Unay (2010). Air drums: A computer vision based drums simulator. [online]. pp. 844–847. doi:https://doi.org/10.1109/siu.2010.5653642.

Google for Developers. (n.d.). Pose landmark detection guide | Google AI Edge. [online]. Available at: https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker.

Harel Yadid, Almog Algranti, Levin, M. and Ayal Taitler (2023). A2D: Anywhere Anytime Drumming. 2017 IEEE Region 10 Symposium (TENSYMP), pp.1–6. doi:https://doi.org/10.1109/tensymp55890.2023.10223631.

Hong, S., Stephen, K., & Kenji, T. (2024). Virtual Drum System Development using Motion Detection. IIAI Letters on Informatics and Interdisciplinary Research, *5*, 1. https://doi.org/10.52731/liir.v005.263

Kalman, R.E. (1960). A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering, 82(1), p. 35. doi:https://doi.org/10.1115/1.3662552.

Paradiddle. (n.d.). Paradiddle - Play drums without the limitations of the real world. [online]. Available at: https://paradiddleapp.com/.

Tolentino, C.T., Uy, A. and Naval, P. (2019). Air Drums: Playing Drums Using Computer Vision. [online]. pp. 1–6. doi:https://doi.org/10.1109/ismac.2019.8836175.