Alternate Interface for Electronic Drum Kits Based On Computer Vision

Suryoday Basak
Department of Computer Science and Engineering
PESIT Bangalore South Campus
Bangalore, India
suryodaybasak@gmail.com

Varun Murthy Suravarapu
Department of Computer Science and Engineering
PESIT Bangalore South Campus
Bangalore, India
varun.ms@icloud.com

***Abstract*—Commercial electronic drum kits are often expensive and cumbersome, keeping them out of the reach of many musicians. The purpose of this study was to see whether, given advancements in current camera and processor technology, computer vision techniques could be applied to create an alternate interface for an electronic drum kit. Computer vision is extensively used to aid in motion tracking, as it often reduces the need for multiple sensors. In this study, we detect regions of interest on a stationary plane, each corresponding to a particular drum sound, and extract useful features from a non-stationary object, corresponding to a drumstick. As a result, we are able to reduce the number of input units used, compared to traditional commercial electronic drum kits by attaching force sensors to the drum sticks, instead of *each* region of interest. At an intersection of the stationary and non stationary features, and with force detected above a predefined threshold, the corresponding sound is played. We conclude by listing the various limitations of these techniques suggesting adequate physical conditions under which such a system could work with acceptable latency.**

Keywords—Computer Vision, Motion Tracking, Drums, Drum Kits

I. Introduction

With improvements in technology, electronic drum kits have become far more accessible for novice and experienced musicians alike. These improvements include advancements in interfaces, sound quality, number of outputs, built-in facilities, etc. Given the tremendous strides made in camera technology and processors, we postulated whether an alternate interface for an electronic drum kit could be created using Computer Vision as a method of input to the system, which would be analogous to the motion of drumsticks.

A real drum kit may be considered to be comprising of two main parts: the stationary and the non-stationary parts, which are the different drums in the kit, and the drumsticks respectively. A system can be developed such that the stationary part is made of different defined regions in an image, and the non-stationary part is a moving object, whose motion is tracked relative to the stationary part. If these two parts intersect, then we can say that an attempt at playing a drum has been made. This can be confirmed by using piezoelectric sensors for measuring the force of the attempt: if the force is greater than a predefined threshold, the appropriate sound is played.

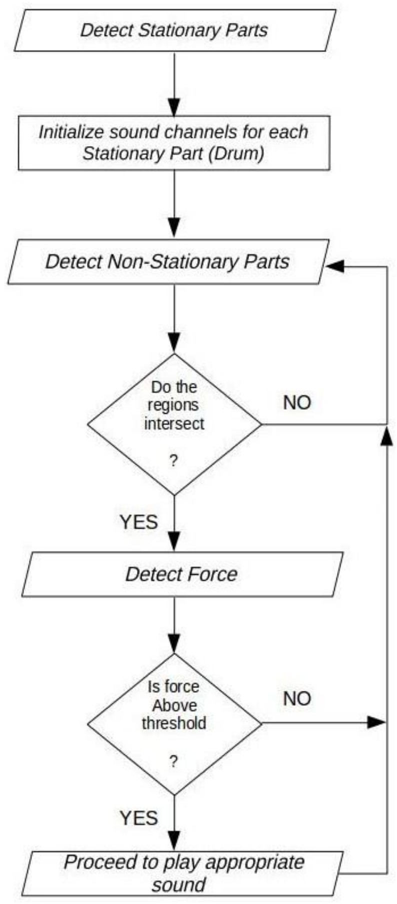

Detect Stationary Parts
Initialize sound channels for each Stationary Part (Drum)
Detect Non-Stationary Parts
Do the regions intersect ?
NO
YES
Detect Force
Is force Above threshold ?
NO
YES
Proceed to play appropriate sound

Fig.1: A brief layout of the system

II. Related Work

Various other studies explore the usage of computer vision to achieve a similar outcome using a variety of cameras and sensors of different specifications like Microsoft's Kinect sensor and Nintendo's Wii Remote. One such study involved the use of the Kinect Sensor and Wii Remote where the Kinect is used for location tracking, which in turn detects the edges of Wii Remote (Jeffrey Booth, et al., 2011). In the same study, for hit detection; the Wii Remote sensors were used to offset the latency caused due to the Kinect Sensor. Another study involves the usage of coloured regions corresponding to drums(Omar Waheed, et al.), and such a system relies on each region being covered >50% by the player, so that the corresponding sound is played. A more generalised approach is to base the system entirely on computer vision (Will Grissom, et al.) by detecting skin colour and accordingly, tracking motion.

These studies highlight the variety of challenges involved in tracking motion for playing drums; foremost of which is the inherent latency of cameras and image processing algorithms, and the requirement of a high frame rate camera to offset this latency. Detecting hits without the usage of physical sensors is yet another challenge.

III. Our Work

A. *Detection of the Stationary Part*

The stationary part can be (circular) regions on a flat, uniformly coloured surface, which can be perceived clearly, in this case (Fig.2(i)). To detect this, we use the Canny Edge Detection algorithm (after converting the original image from RGB to grayscale). Here on, the the subsequent steps in finding the drums regions involve only binary images. The edges form closed boundaries which differentiate different drum regions on the stationary part, and, as in real life, different drums can not share boundaries (Fig.2(ii)). After having detected the edges, the contours of the edges are closed inwards. These form defined regions (Fig.2(iii)). Bounding circles around each of these regions is drawn and the (X,Y) coordinates and corresponding radii (r) of the regions are stored in two arrays, such that at any index of the centers' array, we can get the corresponding radius by using the same index. Any intersection between the stationary and non-stationary parts are determined by comparing against values stored in these two arrays. Fig.2(iv) indicates the bounding circles around the drum regions in the original image.

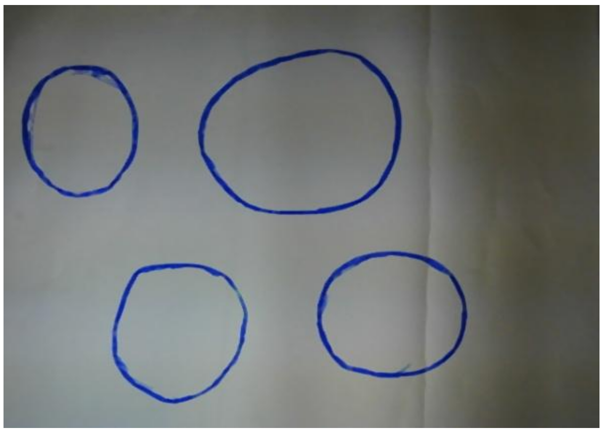


Fig.2(i): An arbitrary drum layout

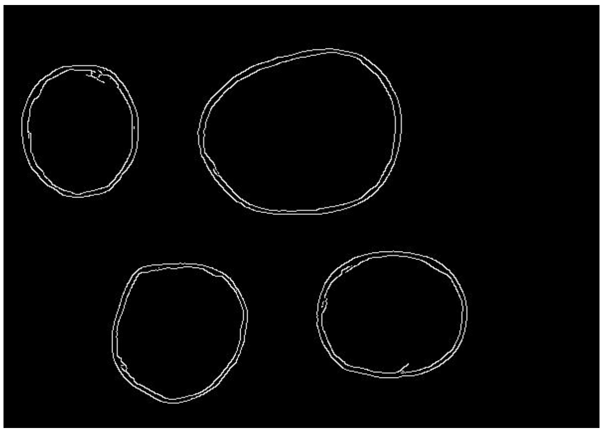


Fig.2(ii): Canny edge detection detecting edges in stationary part

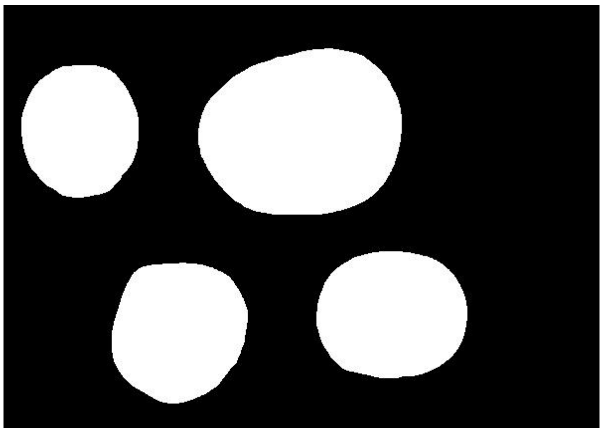


Fig.2(iii): Contours of the edges being closed inwards

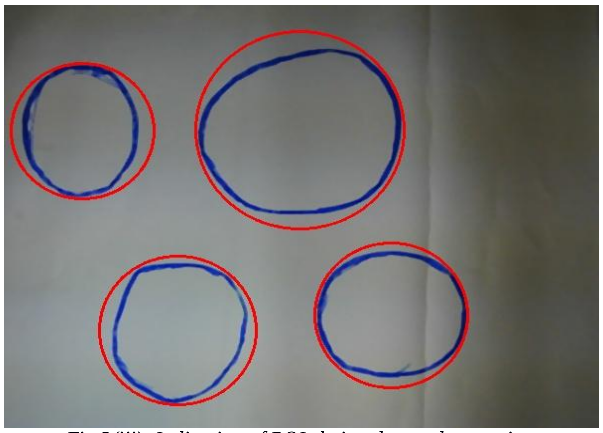


Fig.2(iii): Indication of ROIs being detected on stationary part

B. Detection of the Non-Stationary Part

The non-stationary part of the system is taken as a coloured blob, that rapidly changes coordinates. To optimize the detection of this blob, we go over the following steps:

- Background Subtraction – The first frame of input is taken as the background. All subsequent frames are subtracted from it. This gives us the foreground at e

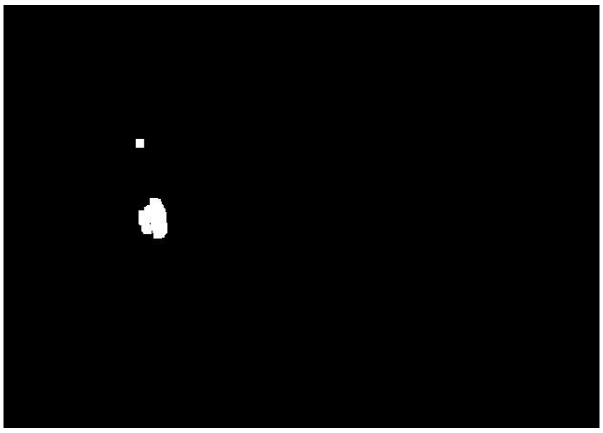
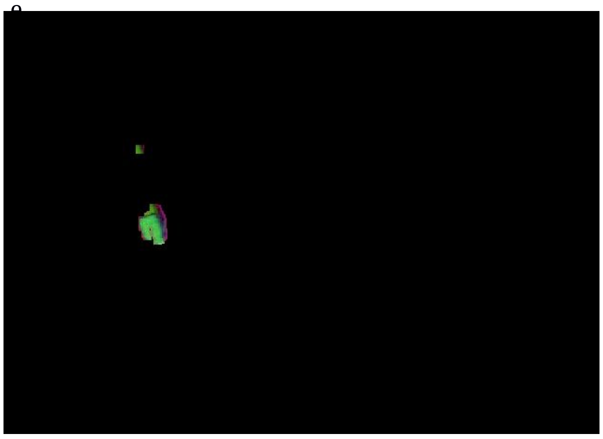
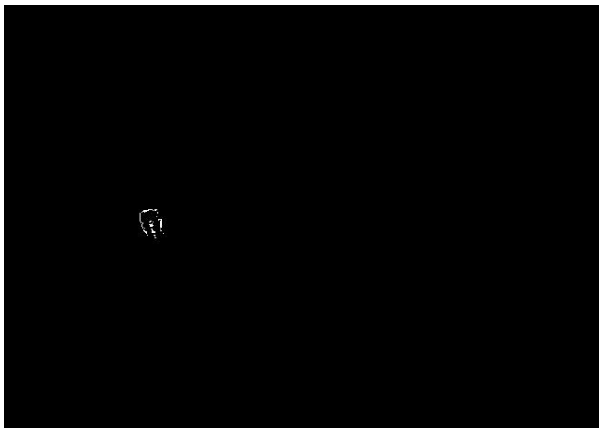


Fig.3(i): Detected foreground

- Colour Detection – The colour of the blob is pre-defined. The current frame is masked with it's respective foreground, and the corresponding colour is matched. This gives us contours of the blob. The colour d e





place in HSV space.

Fig.3(ii): Foreground masking with original frame and conversion to HSV

Fig.3(iii): Detected moving part (moving ROI, green blob)

- Finding the centroid of the blob – We take the largest contour and compute it's centroid. Under broad assumptions, this can be taken as a point inside the blob, and we consider this point represent the blog in case of intersection with the stationary part

C. Determining an Intersection of the Stationary and Non-Stationary parts, and subsequently determining a hit

After having found the centroid of the largest contour of the moving part, we compute to find if it lies in any of the regions by using the formula:

$$\sqrt{[(x-C_x)^2+(y-C_y)^2]} \le r, \qquad (1)$$

where,

C_x is the X-axis coordinate of the centroid of the contour

C_y is the Y-axis coordinate of the centroid of the contour

x is the X-axis coordinate of the center of a chosen region

y is the Y-axis coordinate of the center of a chosen region

If the above condition holds good for any region of interest on the stationary part, we proceed to measure the force of a hit, taking input directly from a piezoelectric sensor.

D. Sound Playback

Every circular region of interest is given an arbitrary label, (say 1,2,3 etc.) and each label corresponds to a particular drum in the drum kit. For example, label 1 may be mapped to the cymbals, label 2 to the snare, and so on. Each drum, is in turn associated with its own characteristic sound files. The number of sound files associated with each drum is stored in a dictionary. All of these sound files differ in frequency and amplitude based on the force with which the drum is struck. Once an intersection is detected between the stationary and non stationary features, and the force returned measured is above the predefined threshold (Fig.4), a rounded float value is returned, which gives the force with which the drumstick has landed on the stationary part. Based on this returned value, the appropriate sound is played in the sound channel.

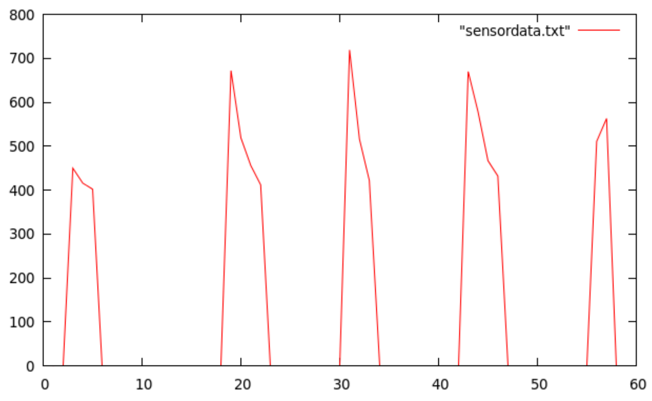


Fig.4: The spikes indicate hits (sensor feed)

Each drum must have it's own unique channel or thread associated with it. As many regions of interest (drums) may be struck simultaneously, it is necessary to ensure that each sound file is played in it's entirety and no played sound file cuts off the previous sound files playback due to insufficient number of sound channels (our system's default is 8). Also by ensuring that each drum possesses its own unique sound channel we ensure that a particular region of interest struck twice in quick succession does not generate two overlapping sound objects. The default sound buffer size (4096 bytes) has been reduced to 512 bytes in order to significantly reduce latency. The trade off is that for longer drum sound (greater than 15 to 20 seconds), playback may be cut short when the buffer is full. In a drum kit this trade off is necessary as low latency is of prime importance and under broad assumptions, most of the recorded drum sound samples are significantly shorter than 10 seconds; with most sound files becoming mostly inaudible by the 7-8 second mark.

IV. Observations

The latency from the proposed system is observed. For this, we tested the code on two machines. The latency is measured as the time between a frame is recorded (for analyzing position of moving part) to the time a sound is played. The observations have been made over approximately 250 continuous samples. Here, we state the mean in each case.

Sl. No.	Clock Rate of Processor	Observed Latency (Seconds)
1	1 GHz	0.8
2	2.5 GHz	0.039

V. Limitations

- Frame Rate Limitations and Latency - Factoring a calculated latency using (which includes latency in processing the image and latency in playing the sound) and the 60 frames per Second frame rate of the camera, the maximum theoretical BPM is close to 150. The Average Beats Per Minute for Drums And Bass is usually 160-180 BPM, under broad assumptions. Our maximum theoretical Beats Per Minute is below this rough average, clearly indicating room for improvement.
- False Positives From Piezoelectric Sensors - The piezoelectric sensors used are extremely sensitive to force, hence only forces above a predefined threshold value are considered acceptable values. However, despite this measure the sensor returns occasional false positives, due to debounce effect and other issues.
- Wired System - Each Drumstick contains its own piezoelectric sensor which is in turn connected to an Arduino which interfaces the sensors with a laptop. This setup involves a considerable number of wires and cables, which restricts movement and keeps the sticks tethered; unlike a traditional drum kit which does not impose restrictions on the movement of the drum sticks.
- Lighting Conditions - Our current setup involves the use of the Playstation Eye camera, which while able to capture images at a high frame rate, is not very capable at low light imagery due to a lack of a Back Side Illumination Sensor. As a result, the setup does not function properly without sufficient light. Minimal light should be focused on the setup such that the edges of the ROIs and the background is greater than the threshold specified in the edge detection algorithm.
- Spacing of Drum ROIs - While the ROIs can be free form and need not be exactly circular, there is a minimum amount of spacing required between each ROI; such that in the process of morphological transforms, two ROIs don't get cascaded as one.

VI. Tools Used

- A Playstation Eye camera for video input (used at 60FPS; resolution: 640 by 480 pixels).
- Ceramic piezoelectric plate for measuring force of a hit.
- Arduino Uno (R3) for interfacing piezoelectric plates with a computer.
- Arduino Ide (version 1.0.6)
- OpenCV Computer Vision library (for Python 2.7; version 2.4.9)
- Pygame (version 1.9.1) module (for Python 2.7) for handling the sound channels
- Pyserial (version 2.6) module for retrieving data from arduino into the python script

- Gnuplot (version 4.6) for plotting sensor-feed graph

VII. Future Prospects

- Future setups will forego the wiring between the Arduino and the Drumstick in order to untether the drumsticks from the setup. However, care must be taken in order to ensure that further latency is not added due to the wireless mode of communication.

- Usage of force sensors with better circuitry to measure force with which drumstick is struck, instead of simple piezoelectric plates.

- It is possible to forego the use of sensors entirely and go with a purely Computer Vision based setup. However, given the success of prior studies, it does not appear to be possible to use a traditional camera(to keep costs down) and maintain acceptable levels of latency. Moreover, the sensors used are not too expensive and are far more reliable than any current computer vision techniques.

Acknowledgment

We would like to thank our mentor Dr. Anand Narasimhamurthy for guiding us in this study, Nawara Shanoon, Akshay Chawla and Sindhoor Tilak for helping us with experimental setup, Siam Guite for giving us a clearer understanding of the acoustics of a real drum kit.

References

[1] Pygame documentation: https://www.pygame.org/docs/

[2] OpenCV documentation: https://opencv-python-tutroals.readthedocs.org/en/latest/

[3] Kaan C. Fidan, Ihsan Kehribar, M. Tu˘gc¸e S¸ahin, Serhan Cos¸ar, Devrim Unay (Computer Vision and Pattern Analysis Laboratory, Sabanci University, Orhanli, Tuzla, ˙Istanbul, Turkey), Devrim Unay (Electrical and Electronics Engineering, Bahcesehir University, ˙Istanbul, Turkey), “Air Drums: A Computer Vision Based Drum Simulator.”

[4] Will Grissom, “A Computer Vision-Based MIDI Controller”

[5] Jeffrey Booth, Benjamin Ullom, Michael Sloan, Dan Gerdesmeier (University of Washington), “A KinectTM and WiimoteTM Based Digital Drum Kit”

[6] Dan S. Bloomberg, Luc Vincent, “Document Image Applications”, Chapter 46 in Livre Hermes Morphologie Mathmatique