Virtual Drum Simulator Using Computer Vision

Shannon Rae F. Bering⁺, Sandra Mae W. Famador^{+,*}

Department of Computer Science
University of the Philippines Cebu, 6000, Philippines

*swfamador@up.edu.ph

Abstract

Computers have slowly begun to replace many of our day to day objects. Something like a laptop can easily become anything, given the right software. Numerous emulators and simulators have been developed in order to make the computer act like something completely different. This study attempts to create a virtual drum system that can be played using a webcam and a computer system alone. It utilizes the many functions of computer vision aided with the OpenCV⁽¹⁾ library. Image processing techniques such as application of a Gaussian filter and thresholding were used in order to track the user's hand and successfully emulate the drum through tracking of its various movements. The contours extracted from the thresholding techniques are tracked by finding their respective bounding rectangles. The result was a system that could emulate a drum and can be played with the hands through striking the different areas of the drum. Although this method was quite effective and did produce the desired effect, the system does not allow the user's head to be present in the video as it may be detected and mistaken for a hand. Further developments to the system should address this, and should also include a solution for supporting foot functionalities such as bass drum and hi-hat control.

Keywords: virtual drum simulator, virtual musical instruments, computer vision, OpenCV.

1. Introduction

Technology has come such a long way in such a short time. Since the invention of the first electronic camera in 1975, digital cameras and sensors are now seen as commonplace items in smart phones, computers, and even in

gaming consoles. This progress has resulted in the formation of a field in computer science known as computer vision.

This study aims to develop a virtual musical instrument that simulates the sounds produced by a drum kit by allowing the user to perform gestures associated with that of playing the instrument. The system must be able to accommodate different styles of playing the instrument. The purpose of this study is to utilize and implement computer vision techniques in order to develop a complete virtual drum system. This study is significant because it allows the user to train himself how to position his hand correctly while playing a drum. For those who want to play the drum set but are not familiar with the instrument, the system can be used to teach a user how to play the instrument. In the absence of an actual drum set, the player can still practice the right hand position and timing. With the development of this software, anyone interested to play the drum can now practice virtually. The player can play background music and practice playing the drum while the music is playing. Moreover, this study may contribute to the development of more complex virtual musical instruments.

Unlike several studies in virtual playing of musical instruments, this study does not require additional equipment or gadget. Conventional studies use additional gadgets to play the instrument. A built-in camera of a laptop or a camera attached to the web is only required to use the system. The only requirement for an efficient output is to use a high resolution camera to detect the hands of the person playing the virtual drum correctly. In the absence of a high resolution camera, the system will still run but with considerable amount of error. A high resolution camera mentioned in this study can be a high definition or HD camera.

Gesture recognition, along with facial recognition, voice recognition, eye tracking and lip movement recognition, allow users to execute a wide range of commands to the computer using only an attached camera or microphone. This reduces the amount of peripherals

⁺ Both authors contributed equally.

necessary to operate such a complex machine. Recognizing gestures enables a more natural interaction that is especially needed in gaming or when navigating a 3-D virtual environment.

Gesture recognition is also important in the development of virtual instruments. Virtual instruments process information collected by the system's different sensors and execute commands through its actuators. This is especially useful in fields such as medicine and chemistry where human physical contact can sometimes be dangerous. Virtual instruments lessen the gap of humans and machines as they allow the machines to execute commands and actions that are otherwise difficult for humans.

Virtual instruments are computer programs that interact with real world objects by means of sensors and actuators and implement functions of real or imaginary instruments. In the field of music, the concept of emulating actual musical instruments is not exactly new. Since the 18th century, electronic musical instruments have become both more sophisticated and complex, with electronic dance music quickly becoming more and more popular. With today's recent developments in virtual and augmented reality, Virtual Musical Instruments (VMIs) are taking another step forward.

A popular medium for creating VMIs is with the use of augmented reality and object detection. An example of this is the project by Rastogi and Joshi⁽²⁾, where they used a laptop camera to detect when a marker hits their designed template. The computer then produces the corresponding sound using its speakers. Although their program can detect hits to the template accurately with normal hit speeds, as hit speed is increased, the performance deteriorates.

Another method of creating VMIs is with the use of in-air gestures, as with Lages, Nabiyouni, Tibau, and Bowman's Interval Player⁽³⁾. The system uses a Leap Motion controller. The VMI developed employed the use of both hands to play notes and chords. The dominant hand specifies intervals between successive notes, while the non-dominant hand plays the chords. The interface, however, has not been able to handle half step intervals, such as sharps and flats, and that it cannot handle intervals greater than five.

Kanke, Takegawa, Terada, and Tsukamoto's Airstic Drum⁽⁴⁾ modified the existing drumstick so that it outputs sound when the drummer strikes a virtual drum. Kanke et al. equipped drumsticks with an acceleration sensor, a gyro sensor, a PC, and a Musical Instrument Digital Interface (MIDI) sound generator. Data is sent through Bluetooth, and the PC sends MIDI messages to the MIDI sound generator when it recognizes the virtual drum being struck. It is

designed to be used alongside physical drums because only less frequently used parts such as the cymbal and the cow bell are being simulated by the Airstic.

Sarang, More, Gaikwad, and Varma⁽⁵⁾ designed Drum, a virtual instrument using the Microsoft Kinect⁽⁶⁾ and arduino. The researchers employed the 3D sensing framework OpenNI as the driver and API to communicate with the Kinect. They were able to obtain a user skeleton with 20 joints. When the wrist enters the region of the virtual drum sets, their program triggers static WAV sound files corresponding to the virtual drum. Sarang et al. were only able to create virtual drums for the Snare, Tom, Crash Cymbal, and Ride Cymbal. The foot pedals were simulated using the arduino and a cantilever switch. They programmed the arduino to register signals from the cantilever switch, and to play a sound whenever a trigger signal is received.

Sensors are one of the most fundamental things in developing a virtual system. One of these is described in a research article by Liu, Fan, Li, and Zhang⁽⁷⁾. Using the particle filter model, they developed a hand tracking and gesture recognition system that has a high tracking accuracy. The researchers utilized formulas that identify hand skin colour, temporal motion, and depth to identify the hand in the presence of background clutter. Initialization happens during start up when the actor is asked to place the hand above the actor body. The system described is only able to detect hand gestures when the actor is in a static position (sitting down in front of the camera), and is only able to track one hand.

Wei-chao Chen's master's thesis⁽⁸⁾ describes a system that detects the position of the hand by identifying the palm center, even in the presence of the user's forearm. It does not perform any hand or arm recognition, but merely assumes that the input contour is the contour of the hand and the forearm. The system identifies the palm center by calculating the convex hull of the areas within the color range specified and selecting the one with the largest area. The fingers, on the other hand, are identified by calculating the contours. This method occurs problems when there is an object larger than the palm whose color is within the color range indicated present within the screen. When something like a face is present on the screen, the system is more likely to detect the face rather than the hand present.

2. Flow of Work

Figure 1 shows the pipeline of the study. The system first grabs and decodes the frame read from the camera. It then

Fig. 1. The pipeline.

processes the image by converting it into the YCbCr color model. YCbCr is a family of color spaces used in video and digital photography systems. A Gaussian filter is then applied to the image. A mask is created from the converted frame, using the range of values of skin color for YCbCr, and proceeds to detect all the contours for the areas whose colors are within this range. The system identifies the hands, and creates a bounding rectangle around each. This makes it easier to track the contours as the coordinates of the bounding rectangles can be assigned to variables. If the coordinates of certain corners of the bounding rectangles come within the bounds of the drum pads, the system will then play the corresponding sound. The system loops this

Fig. 2. Screenshot of the system.

Fig. 3. Screenshot of the mask layer.

Fig. 4. Screenshot of the drum areas.

process until the user presses a key on the keyboard, indicating the end of the simulation.

Figure 2 shows a screenshot of the system during running time. It displays the mirrored image of the user, and the drum system. The contours found in the image are outlined in dark blue, while the bounding boxes for the hands are in light green. The areas that are within the color range for skin specified are indicated by the white areas in the mask (Figure 3). In the topmost area, the coordinates of the right and left hands (with respect to the bounding boxes) are displayed.

The drum parts and their corresponding colors are as follows: white for the closed hi-hat, gray for the snare drum, light brown for the crash cymbal, pink for the first tom-tom drum, yellow orange for the second tom-tom drum, and blue violet for the ride cymbal. Figure 4 shows these areas clearly.

2. Frame Reading and Processing

Different color models such as HSV were included in the experiment. HSV is one of the two most common

```

If  $a[m,n] \geq \theta$        $a[m,n] = \text{object} = 1$ 
Else                   $a[m,n] = \text{background} = 0$ 

```

Fig. 5. Thresholding algorithm.

representations of points in an RGB color model. H means hue, while S means saturation, and V means value. In this study, it was found that YCbCr was most effective for the system. The image was then given a Gaussian filter described in equation 1 in order to reduce noise and make it easier to segment the image.

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

We used a fix threshold in this study. Figure 5 shows the thresholding algorithm where θ is the threshold and $a[m,n]$ is the current frame being processed. Values of θ are values commonly used to detect skin color. Basilio, et al.⁽⁹⁾ described the θ values to detect skin in an image. Using OpenCV, the function `cv2.inRange()` was used to implement this.

3. Contour Detection and Tracking

Green's formula⁽¹⁰⁾ as shown in equation 2 describes the moments of a contour.

$$\oint_C (L dx + M dy) = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} \right) dx dy \quad (2)$$

To implement this, we used the `cv2.FindContours()` function to obtain the contours from the thresholded image. The system then proceeds to identify which contours to track. It assumes that the two largest contours found are the hands. The pseudocode in Figure 6 shows how this is done. The code to identify the two largest contours runs with $O(n)$ complexity, with n being the number of contours found in the image. The system then creates a bounding rectangle around each contour. In geometry, a bounding rectangle is the two-

```

#Initialize variables of the largest and second largest contours to 0
#For every contour found in the image,
  #If the size of the nth element is greater than largestContour
  #Second largest contour is equal to the largest contour
  #Largest contour is equal to the nth element
#Else if nth element is greater than the second largest contour
  #Second largest contour is equal to the nth element

```

Fig. 6. Pseudocode for obtaining the two largest contours.

```

#Assign variable per instrument part
#If hand coordinate is above the instrument part,
  #Variable is true
#If hand coordinate is within instrument part bounds and variable is true
  #Play sound

```

Fig. 7. Pseudocode for playing sound.

dimensional box which contains all points of a set. The system assigns the bounding boxes to the right and left hand by comparing the x-coordinates of each contour and switching the variable assignment for the left hand coordinates and the right hand coordinates.

4. Playing Sound

The sound files need to be loaded at startup, but require no further initialization. Since each frame is looped, the system checks whether or not the hand coordinate originated from the area above the drum part. It only plays a sound when the hand 'strikes' the instrument from above, like a normal drum. Figure 7 shows the algorithm for playing sound. The code for playing sound runs with $O(1)$ complexity.

5. Copyright

This paper is an original product of the researchers and is currently unpublished. All proprietary products of researches mentioned in this study are properly cited in this paper.

6. Experiment Result

The system was developed in Ubuntu 14.04 using the Python programming language (version 2.7.6) and utilizes both the PyGame sound module and the OpenCV version 2.4.8 library. It was tested on a laptop with Intel®Core™i5-3210M 2.50GHz CPU and 4 GiB RAM. The video image was captured by the default web camera with resolution of 640x480. Due to the low quality of the camera used, large amounts of noise were encountered. Color of clothing was also a factor. Clothing with certain shades of red can be mistaken as the color of human skin. Despite this, the system performed well with the tests conducted. Tests were also performed in a Windows OS environment with a high resolution camera. Results were summarized as reflected in Table 1. Values in the table are the results of playing the virtual drum. These values may change depending on the

Table 1. Actual hand position in correctly playing the drums during conducted tests.

	Pixel Position	Hi-hat	Ride Cymbal	Crash Cymbal	1 st Tom-tom	2 nd Tom-tom	Snare
Left Hand Up	x	129-138		20-21	122-137		237-245
	y	77-91		78-80	66-73		67-70
Left Hand Down	x	171-188		70-78	184-202		279-304
	y	76-87		77-78	69-79		60-63
Right Hand Up	x		23-27		121-133	242-243	
	y		453-521		33-36	496-510	
Right Hand Down	x		70-80		194-202	303-333	
	y		510-520		33-35	480-499	

camera and the resolution used while running the system. Figures 8 to 13 are results of running the system using a different camera and a different environment. It is therefore advised to use the same camera and computer during its operation.

Accuracy is inversely proportional with the speed of the hand playing the virtual drum. Also, background color affects the accuracy of the output. Table 1 describes the actual hand position in pixel location if the background is white and the person playing the virtual drum does not wear clothing within the YCbCr range of [0,77,133] to [255,127,173], or clothes with red, orange, and yellowish hues. Images showing the correct hand positioning for playing the different drum parts are shown in Figures 8-13.

Fig. 8. Correct position for playing the crash cymbal.

Fig. 9. Correct position for playing the snare.

Fig. 10. Correct position for playing the hi-hat.

Fig. 11. Correct position for playing the 1st tom-tom.

7. Conclusion and Recommendations

7.1 Conclusion

Based on the algorithms implemented, the objectives of the study were met. Computer vision can be used to play a

Fig. 12. Correct position for playing the ride cymbal.

Fig. 13. Correct position for playing the 2nd tom-tom.

drum virtually. Contour detection and tracking can identify the position of the hand of the drummer. Through the positions, the different audio frequencies of a drum set can be played by the software.

7.2 Recommendations

The current system is dependent on the fact that the head of the user is covered or is not present in the frame while using it. An algorithm that will recognize the head but will treat it as it is will be developed in the future. A better algorithm for identifying skin should also be researched upon, as clothing and objects with similar color can be mistaken by the system for skin. It is recommended that OpenCV 3.0 is used as many functions may prove useful in improving the current system. An algorithm that allows foot functionalities such as opening the hi-hat and the bass drum is recommended and will be implemented in the future.

References

- (1) Open source computer vision library (opencv). <http://sourceforge.net/projects/opencvlibrary/>.
- (2) Rastogi, A. and Joshi, A. (2014). Virtual musical

instruments.

https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Rastogi_Joshi.pdf

- (3) Lages, W., Nabyouni, M., Tibau, J., and Bowman, D. A. (2015). Interval player: Designing a virtual musical instrument using in-air gestures. In 3D User Interfaces (3DUI), 2015 IEEE.
- (4) Kanke, H., Takegawa, Y., Terada, T., and Tsukamoto, M. (2012). Airstic drum: a drumstick for integration of real and virtual drums. In Advances in Computer Entertainment, pages 57–69. Springer.
- (5) Sarang, P., More, A., Gaikwad, A., and Varma, T. (2015). Air drum using kinect and arduino. International Journal of Computer Science and Information Technologies, 6(2):1153–1155.
- (6) Xbox kinect. <http://www.xbox.com/en-US/kinect>.
- (7) Liu, W., Fan, Y., Li, Z., and Zhang, Z. (2015). Rgb-d video based human hand trajectory tracking and gesture recognition system. Mathematical Problems in Engineering, 2015.
- (8) Chen, W.-c. (2011). Real-time palm tracking and hand gesture estimation based on fore-arm contour. PhD thesis, MS Thesis, Dept. Inform. Eng., National Taiwan University of Science and Technology.
- (9) Basilio, J. A. M., Torres, G. A., Pérez, G. S., Medina, L. K. T., & Meana, H. M. P. (2011). Explicit image detection using YCbCr space color model as skin detection. Applications of Mathematics and Computer Engineering, 123-128.
- (10) Open source computer vision library (opencv) documentation. <http://docs.opencv.org/2.4/index.html>.