Low-cost virtual Kulintang using Computer Vision

Christian James E. Barimbao [1*], Martina Joanna A. Manuel [1], Claire Joy H. Ramos [1], Carl Timothy S. Tolentino [1], and Maricor N. Soriano [2]

[1] *Electrical and Electronics Engineering Institute, University of the Philippines Diliman, Philippines*
[2] *Natinal Institute of Physics, University of the Philippines Diliman, Philippines*
[*] Corresponding author: christian.barimbao@eee.upd.edu.ph

Abstract

We developed a low-cost virtual Kulintang instrument that leverages image processing techniques in Python's OpenCV to simulate the playing mechanics of the instrument using colored markers. Our goal is to introduce users to this indigenous Philippine musical instrument and further preserve its cultural significance. The system successfully detects and tracks the centroid of these markers, averaging 24 frames per second (FPS), while closely resembling the physical instrument's overall layout. In our future work, we aim to enhance the realism of the synthesized gong sound by translating the percussive mechanics into realistic sound synthesis, incorporating marker acceleration to produce varying sound intensities. Ultimately, we plan to deploy a desktop application to offer a more immersive experience for the target users. Overall, our virtual Kulintang instrument represents an innovative and accessible way to engage with this unique cultural heritage.

Keywords: kulintang, digital musical instruments, computer vision, image processing

1 Introduction

Philippine indigenous musical instruments and the manner of playing them are representative of the rich culture specific to certain tribes. The Kulintang is an indigenous instrument native to the Mindanao islands in the southern part of the country. It consists of a set of eight embossed gongs suspended horizontally on two parallel strings arranged from largest to smallest, left to right, as seen in Figure 1.

According to a personal interview with a Kulintang master at the University of the Philippines (UP) Center for Ethnomusicology, interest in Philippine indigenous musical instruments has been declining due to the high cost and limited availability of manufacturers, making procurement and learning difficult (A. Butocan, Professor, College of Music, UP Diliman, Dec. 6, 2022). This is especially true for the Kulintang, which is also challenging to transport. In this work, we aim to develop a prototype of a virtual Kulintang instrument using low-cost distinct-colored markers and a laptop webcam. While virtual instruments aim to bridge the inaccessibility of their physical counterparts, current applications mostly focus on western-based instruments like the drums [1], guitar [2–4], and violin [**?**]. Although some locally-inspired virtual instruments exist, most of them rely on touch-based feedback, which bears little resemblance to the actual playing mechanics of the instrument [5–7].

Our algorithm employs standard image processing techniques, including histogram backprojection, dilation, and thresholding – implemented using the Python OpenCV library. We aim to develop a prototype that generates sound output upon the markers' tracked position hitting the bounding boxes, simulating the play of a Kulintang. This work attempts to contribute to the appreciation and preservation of Philippine indigenous music by making a virtual prototype of Kulintang that will be easily accessible to the end users.

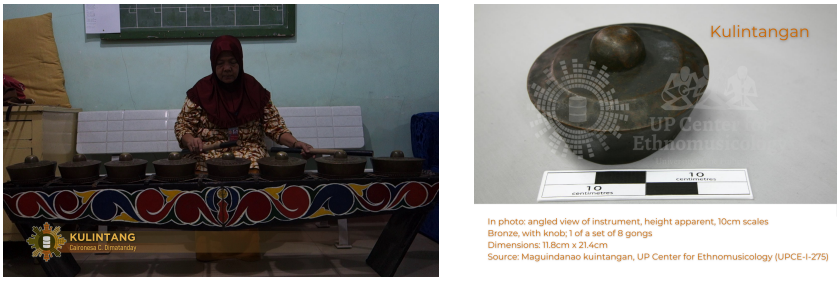
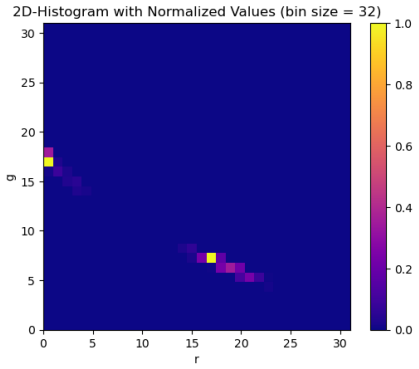


Figure 1: The Kulintang gongs are knobbed at the center and played using a pair of soft wooden sticks called the *basal*. Source of images: DOST Katunog project [8].

(a) Red marker calibration

(b) Green marker calibration



(c) 2D-Histogram of the ROI

Figure 2: The 2D histogram of the colored markers is obtained from the cut-out regions during calibration.

2 Methodology

2.1 Calibration

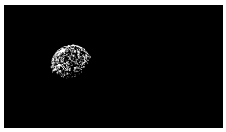
At the beginning of the program, the user is prompted to calibrate the system for initial use as shown in Figure 2. The calibration involves positioning each colored marker within the bounding circle, which serves as the region of interest (ROI), and clicking the mouse to cue the program to crop the ROI and obtain the Red, Green, and Blue (RGB) values per pixel of the markers. The maximum and minimum RGB values are also stored for thresholding later (2.2). Subsequently, the RGB values are converted to the Normalized Chromaticity Coordinates (NCC) using the following equations:

$$I = R + G + B, \quad r = R/I, \quad g = G/I \tag{1}$$

where I represents the pixel intensity, and (r, g) are the NCC coordinate values. Normalizing with intensity reduces the effect of brightness on the chromaticity values r and g. Finally, the 2D-histogram of the ROI with the NCC is computed and normalized to one. Histogram values lesser than 0.5 will be rounded to zero.

2.2 Blob Detection

In order to segment the images, histogram backprojection is employed, since a simple table lookup algorithm is proven advantageous in real-time applications [9]. For each pixel p_i of the subsequent input images, a value $S_i(r_i, g_i)$ will be assigned, where $S(r, g)$ corresponds to the histogram values calculated



(a) Red ROI

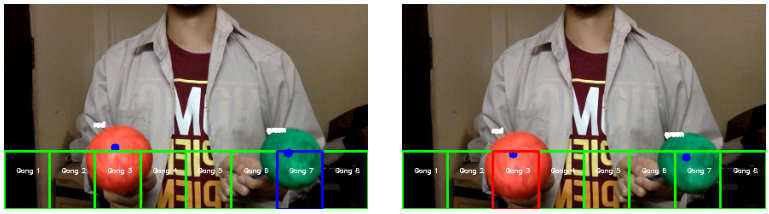
(b) Green ROI

(c) Dilated red ROI

(d) Dilated green ROI

(e) Centroid Estimation

Figure 3: Blob detection algrotihm. The input frame is segmented using histogram backprojection then dilated to fill pixel gaps in the blob before calculating for the centroid.



(a) Strike: Green Marker (b) Strike: Red Marker

Figure 4: Hit Detection.

in 2.1. Histogram backprojection is appropriate to use for real-time robust color detection since it is fast enough while being insensitive to shading and lighting variations. In this work, we used a bin size of 32 but different bin size resolutions can be set to cater different applications. Higher bin size usually means more fine-grained color segmentation while lower bin size allows more color variations.

Then, the backprojected image will be thresholded using the stored maximum and minimum RGB values in calibration to further remove background artifacts. This results to a segmented image, as demonstrated in Figures 3a and 3b.

The segmented image is then dilated to increase the blob size and improve centroid estimation, as observed in Figures 3c and 3d. The centroid is computed using the center of mass formula with $m_i = 1$:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i m_i}{\sum_{i=1}^{n} m_i}, \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i m_i}{\sum_{i=1}^{n} m_i} \tag{2}$$

where n is the total number of white pixels, x_i and y_i are the indices of the i-th pixel, and $\bar{x}$ and $\bar{y}$ are the coordinates of the centroid.

2.3 Hit Detection and Gong Playback

The event of a gong being hit is determined by the comparison of each marker's current position to its position in the previous frame, and a corresponding boolean variable called *hit-state*. The *hit-state* is initialized to *False* for both markers during startup, and will assert to *True* when the centroid goes inside a bounding box. The *hit-state* will then become *False* for the succeeding frames that the centroid remains inside the bounding box to prevent infinite triggers. The x value of the centroid during a *hit-state* = *True* will determine which gong sound will play. A strike of a gong for each marker is displayed in Figure 4

3 Results and Discussion

The system was tested using colored styro balls as markers attached to the ends of a pen. It can be noted that the algorithm was still able to detect and track the centroid of irregularly shaped objects like rectangular plastic containers, handheld fans, and bottles, but a spherical object is preferred for simplicity and invariance to the camera viewing angle. Red and green colors were used as it was readily available but it can be safely assumed that any contrasting colors will work.

It was observed during initial tests that the system can achieve an average execution time in between frames of 43 milliseconds or around 24 FPS, which is the standard for real-time video applications. This can be further improved in the future with more code optimizations (e.g. multi-threading, employing graphical processing unit (GPU) methods, etc.) The lighting conditions also seem to affect the processing speed, where images captured in darker environments having a slower processing time. This can be attributed to the greater noise and lower contrast in low-light images resulting to a greater number of digital artifacts.

The overall sample layout is presented in Figure 5. The software decision of the layout of the gongs is intentional to resemble the actual physical instrument. Because of this, there are constraints in the mobility of the user when playing the virtual instrument. Since motion tracking is now achieved in real-time, kinematical quantities such as the velocity and acceleration of the basal can then be calculated. This shall be integrated on the sound synthesis to generate more realistic sounds. It must be reiterated however, that the ultimate goal of our work is simply to open new avenues for appreciation of the Kulintang, or as a tool to introduce it to beginners, and not as a means to replace the actual instrument.

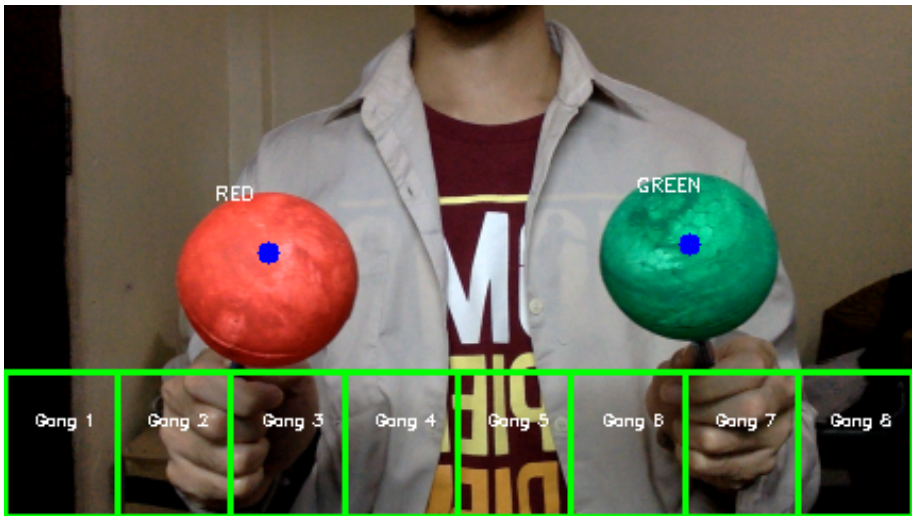


Figure 5: System Overview. Each gong is arranged horizontally with increasing pitch, analogous to the actual Kulintang.

4 Conclusions

We were able to develop a prototype of a virtual Kulintang instrument that synthesizes gong sounds using computer vision techniques. Tracking the acceleration of the markers in order to vary the intensity of the generated gong sounds, as well as packaging the application with a full graphical user interface is left for future work.

Acknowledgments

All gong sounds and images used in this work are a property of the Department of Science and Technology (DOST) *Katunog* project. Philippine Copyright 2023 by DOST-ASTI and UP [10].

References

[1] C. T. Tolentino, A. Uy, and P. Naval, Air drums: Playing drums using computer vision, in *2019 International Symposium on Multimedia and Communication Technology (ISMAC)* (IEEE, Quezon City, Philippines, 2019), 1–6, ISBN 978-1-72811-918-2, https://ieeexplore.ieee.org/document/8836175/.

[2] M. Karjalainen, T. Mäki-patola, A. Kanerva, and A. Huovilainen, Virtual air guitar, *Journal of the Audio Engineering Society* **54**, 964 (2006).

[3] J. Pakarinen, T. Puputti, and V. Välimäki, Virtual slide guitar, **32**, 42 (2008), ISSN 0148-9267, 1531-5169.

[4] J. E. Tamani, J. C. B. Cruz, J. R. Cruzada, J. Valenzuela, K. G. Chan, and J. A. Deja, Building guitar strum models for an interactive air guitar prototype, in *Proceedings of the 4th International Conference on Human-Computer Interaction and User Experience in Indonesia, CHIuXiD '18* (ACM, Yogyakarta Indonesia, 2018), 18–22, ISBN 978-1-4503-6429-4, https://dl.acm.org/doi/10.1145/3205946.3205972.

[5] R. Tanfelix, Virtual kulintang (2020), https://play.google.com/store/apps/details?id=my.virtual.kulintangapp.

[6] A. L. Baltazar, Y. J. Gozar, and P. Yap, Digital sound synthesis of gangsa implemented on an android mobile device (2020).

[7] C. R. G. Lucas and C. J. L. Soriano, Digital sound synthesis of rondalla using physical modeling implemented on PC and iPad, in *IEEE 2013 Tencon - Spring* (IEEE, Sydney, Australia, 2013), 440–444, ISBN 978-1-4673-6349-5 978-1-4673-6347-1 978-1-4673-6348-8, http://ieeexplore.ieee.org/document/6584485/.

[8] Kulintang, open-Source Database of Philippine Indigenous Instrument Sounds, https://katunog.asti.dost.gov.ph/client/instrument/view?i=2606.

[9] M. Soriano, B. Martinkauppi, S. Huovinen, and M. Laaksonen, Adaptive skin color modeling using the skin locus for selecting training pixels, *Pattern Recognition* **36**, 681 (2003), ISSN 0031-3203.

[10] Katunog project, https://katunog.asti.dost.gov.ph.