In [3]:
```
import os
import tkinter as tk
from tkinter import messagebox, filedialog
from pygame import mixer

import wave
import threading
import pyaudio
import cv2
import numpy as np

# Initialize mixer for sound playback
mixer.init()
drum_clap = mixer.Sound('batterrm.wav')  # Replace with your sound file
drum_snare = mixer.Sound('button-2.ogg')  # Replace with your sound file

# Audio recording parameters
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100

# Global variables
recording = False
frames = []
recorded_sounds = []
audio_thread = None
camera = None
downloads_folder = os.path.expanduser("~/Downloads")
audio_directory = os.path.join(downloads_folder, "recorded_audios")

# Ensure the directory exists
if not os.path.exists(audio_directory):
    os.makedirs(audio_directory)

def state_machine(sumation, sound):
    if sumation > Hatt_thickness[0] * Hatt_thickness[1] * 0.8:
        if sound == 1:
            drum_clap.play()
            if recording:
                recorded_sounds.append('drum_clap')
        elif sound == 2:
            drum_snare.play()
            if recording:
                recorded_sounds.append('drum_snare')


def ROI_analysis(frame, sound):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, blueLower, blueUpper)
    sumation = np.sum(mask)
    state_machine(sumation, sound)
    return mask

# Define the HSV range for blue color (you can adjust these values based or
blueLower = np.array([100, 150, 0], dtype="uint8")  # Lower range of blue
blueUpper = np.array([140, 255, 255], dtype="uint8")  # Upper range of blue

def run_digital_drums():
    global camera, Hatt, Snare, Hatt_thickness, Snare_thickness
```

```
    camera = cv2.VideoCapture(0)
    ret, frame = camera.read()
    H, W = frame.shape[:2]

    # Define ROI positions and dimensions
    Hatt = cv2.resize(cv2.imread('Hatt.png'), (200, 100), interpolation=cv
    Snare = cv2.resize(cv2.imread('Snare.png'), (200, 100), interpolation=

    Hatt_center = [W * 2 // 8, H * 6 // 8]
    Snare_center = [W * 6 // 8, H * 6 // 8]
    Hatt_thickness = [200, 100]
    Snare_thickness = [200, 100]

    Hatt_top = [Hatt_center[0] - Hatt_thickness[0] // 2, Hatt_center[1] - H
    Hatt_btm = [Hatt_center[0] + Hatt_thickness[0] // 2, Hatt_center[1] + H

    Snare_top = [Snare_center[0] - Snare_thickness[0] // 2, Snare_center[1
    Snare_btm = [Snare_center[0] + Snare_thickness[0] // 2, Snare_center[1

    while True:
        ret, frame = camera.read()
        frame = cv2.flip(frame, 1)
        if not ret:
            break

        snare_ROI = frame[Snare_top[1]:Snare_btm[1], Snare_top[0]:Snare_btm
        ROI_analysis(snare_ROI, 1)

        hatt_ROI = frame[Hatt_top[1]:Hatt_btm[1], Hatt_top[0]:Hatt_btm[0]]
        ROI_analysis(hatt_ROI, 2)

        # Display drums and labels
        frame[Snare_top[1]:Snare_btm[1], Snare_top[0]:Snare_btm[0]] = cv2.a
            Snare, 1, frame[Snare_top[1]:Snare_btm[1], Snare_top[0]:Snare_b
        )
        frame[Hatt_top[1]:Hatt_btm[1], Hatt_top[0]:Hatt_btm[0]] = cv2.addWe
            Hatt, 1, frame[Hatt_top[1]:Hatt_btm[1], Hatt_top[0]:Hatt_btm[0
        )

        cv2.putText(frame, 'Digital Drums', (10, 30), cv2.FONT_HERSHEY_SIMP
        cv2.imshow('Output', frame)

        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

    camera.release()
    cv2.destroyAllWindows()


def audio_recording():
    global frames, recording

    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True

    while recording:
        data = stream.read(CHUNK)
        frames.append(data)

    stream.stop_stream()
    stream.close()
```

```
    p.terminate()


def start_digital_drums():
    threading.Thread(target=run_digital_drums).start()


def record_digital_drums():
    global recording, audio_thread, frames
    recording = True
    frames = []
    audio_thread = threading.Thread(target=audio_recording)
    audio_thread.start()
    start_digital_drums()


def stop_recording():
    global recording
    recording = False
    if audio_thread:
        audio_thread.join()
    messagebox.showinfo("Recording Stopped", "Recorded sounds: " + ", ".jo


def save_audio():
    if not frames:
        messagebox.showwarning("No Audio", "No audio to save.")
        return

    filename = filedialog.asksaveasfilename(defaultextension=".wav", filet
    if filename:
        with wave.open(filename, 'wb') as wf:
            wf.setnchannels(CHANNELS)
            wf.setsampwidth(pyaudio.PyAudio().get_sample_size(FORMAT))
            wf.setframerate(RATE)
            wf.writeframes(b''.join(frames))
        # Save the audio in the designated directory
        saved_filename = os.path.basename(filename)
        os.rename(filename, os.path.join(audio_directory, saved_filename))
        messagebox.showinfo("Audio Saved", f"Audio saved as {saved_filenam


def list_saved_files():
    # List all saved files in the directory
    saved_files = os.listdir(audio_directory)
    saved_files = [f for f in saved_files if f.endswith(".wav")]

    if not saved_files:
        messagebox.showinfo("No Files", "No recorded files found.")
        return

    # Clear the listbox and populate it with the saved files
    listbox.delete(0, tk.END)  # Clear the listbox
    for file in saved_files:
        listbox.insert(tk.END, file)


def delete_selected_files():
    selected_files = listbox.curselection()  # Get the indices of selected
    if not selected_files:
        messagebox.showinfo("No Files Selected", "Please select a file to
```

```
        return

    # Ask the user for confirmation
    confirmation = messagebox.askyesno("Delete Files", f"Are you sure you w
    if confirmation:
        for index in selected_files:
            file_name = listbox.get(index)  # Get file name from listbox
            file_path = os.path.join(audio_directory, file_name)
            try:
                os.remove(file_path)
                listbox.delete(index)  # Remove from listbox
                messagebox.showinfo("File Deleted", f"File {file_name} dele
            except Exception as e:
                messagebox.showerror("Error Deleting File", f"Failed to del


def play_audio():
    selected_file = listbox.curselection()
    if not selected_file:
        messagebox.showinfo("No File Selected", "Please select a file to pl
        return

    filename = listbox.get(selected_file[0])  # Get the selected file name
    file_path = os.path.join(audio_directory, filename)

    try:
        mixer.music.load(file_path)
        mixer.music.play()
    except Exception as e:
        messagebox.showerror("Error Playing Audio", f"An error occurred: {e


def exit_application():
    global recording, audio_thread, camera
    recording = False  # Stop the recording
    if audio_thread and audio_thread.is_alive():
        audio_thread.join()  # Wait for audio thread to finish
    if camera is not None:
        camera.release()  # Release camera resources
    cv2.destroyAllWindows()  # Close any OpenCV windows
    root.quit()  # Exit the Tkinter application immediately


# Set up Tkinter window
root = tk.Tk()
root.title("Digital Drums")
root.geometry("600x400")

# Buttons
start_button = tk.Button(root, text="Start Code", command=start_digital_dru
start_button.pack(pady=5)

record_button = tk.Button(root, text="Start Recording", command=record_digi
record_button.pack(pady=5)

stop_button = tk.Button(root, text="Stop Recording", command=stop_recording
stop_button.pack(pady=5)

save_button = tk.Button(root, text="Save Recording", command=save_audio)
save_button.pack(pady=5)
```

```
list_button = tk.Button(root, text="List Saved Files", command=list_saved_
list_button.pack(pady=5)

delete_button = tk.Button(root, text="Delete Selected Files", command=dele
delete_button.pack(pady=5)

play_button = tk.Button(root, text="Play Selected Audio", command=play_aud
play_button.pack(pady=5)

exit_button = tk.Button(root, text="Exit", command=exit_application)
exit_button.pack(pady=5)

# Listbox to display saved audio files
listbox = tk.Listbox(root, height=10, width=50)
listbox.pack(pady=10)

root.mainloop()
```

```
pygame 2.6.1 (SDL 2.28.4, Python 3.11.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
(https://www.pygame.org/contribute.html)
```