Digital Signal Processing LAB
LAB 4 (Report)

Mohammad Parsa Dini - std id: 400101204
Mohammad Hossein Momeni - std id: 99102359

Introduction

This report presents the results and analysis of Lab 4 for the Digital Signal Processing course at Sharif University of Technology. The experiment focuses on processing a music signal sampled at 8 kHz using a Texas Instruments DSK6416 board. The objective is to detect beats in the audio signal by calculating the average energy of a buffer and its constituent chunks, as implemented in the provided `volume.c` program. The signal is anti-aliased, stored in a buffer, and divided into smaller chunks for processing. Beat detection is achieved by comparing the energy of individual chunks against the overall buffer energy, with sensitivity controlled by a constant C. This report details the code functionality, the mathematical basis for energy calculations, and the implementation of the beat detection algorithm.

Experiment Setup and Code Analysis

The experiment utilizes the `volume.c` program, which processes an audio signal on the Texas Instruments DSK6416 board. The signal is sampled at 8 kHz and stored in a buffer of size 2000 samples, divided into 20 chunks of 100 samples each, as recommended in the lab instructions. The program calculates the average energy of the entire buffer and each chunk, using these values to detect beats when a chunk's energy significantly exceeds the buffer's average energy.

Mathematical Basis

The energy calculations and beat detection are based on the following equations, as provided in the lab instructions:

1. **Average Energy of the Whole Buffer**:

$$\langle E \rangle = \frac{1}{N} \sum_{k=0}^{N} (B[k])^2 \tag{1}$$

where $B[k]$ represents the signal's sample values, and $N = 2000$ is the total buffer length.

2. **Average Energy of a Chunk**:

$$\langle e_j \rangle = \frac{1}{n} \sum_{k=i_0}^{i_0+n} (B[k])^2 \quad \text{where } i_0 = 0, n, 2n, \ldots \tag{2}$$

where $n = 100$ is the chunk length, and j indexes the chunks (20 chunks total).

The beat detection condition is implemented as:

$$\langle e_j \rangle \geq C \cdot \langle E \rangle \tag{3}$$

where $C = 1.8$ is the sensitivity constant. In the code, this is expressed as `mean_energy - frame_energy*c >= 0`, where `mean_energy` is $\langle e_j \rangle$ and `frame_energy` approximates $\langle E \rangle$ over a rolling window of 20 chunks.

Code Functionality

The `volume.c` program performs the following key operations:

- **Initialization**: The program initializes the DSK6416 board and its LEDs, setting up input and output buffers (`inp_buffer` and `out_buffer`) of size `BUFSIZE` (2000 samples). It also initializes variables for tracking chunk energies (`chunk_energy`), frame energy (`frame_energy`), and a flag for beat detection.

Listing 1: Initialization Code in volume.c

```
/* Global declarations */
int inp_buffer[BUFSIZE]; /* processing data buffers */
int out_buffer[BUFSIZE];
int index;
char v[20];
int j;
float c;                 /* sensitivity constant for beat detection */
int flag;
float chunk_energy[20];  /* energies of all chunks */
float frame_energy;
long long int tot_energy;              /* total energy of the frame */
int gain = MINGAIN;                    /* volume control variable */
unsigned int processingLoad = BASELOAD; /* processing routine load
    value */
struct PARMS str =
{
    2934,
    9432,
```

```
    213,
    9432,
    &str
};

/* In main function */
void main()
{
    DSK6416_LED_off(1);
    DSK6416_LED_off(0);
    DSK6416_LED_off(3);
    DSK6416_LED_off(2);
    flag = 0;
    index = 0;
    c = 1.8;
    for(j=0;j<20;j++){
        chunk_energy[j]=0;
    }
    DSK6416_init();
    DSK6416_LED_init();
    puts("volume example started\n");
    /* ... rest of main ... */
}
```

- **Data Input/Output**: The `dataIO` function handles the reading of input signals and writing of processed output signals, though its implementation is not detailed in the provided code.

Listing 2: Data Input/Output Code in volume.c

```
static void dataIO()
{
    /* do data I/O */
    return;
}
```

- **Energy Calculation**: The `calc_avg_energy` function computes the average energy of a buffer or chunk as the sum of squared sample values divided by the number of samples.

Listing 3: Energy Calculation Code in volume.c

```
float calc_avg_energy(int* x, int len){
    long long int energy = 0;
    int i=0;
    for(; i<len;i++){
        energy += x[i]*x[i];
    }
    return ((float)(energy)/((float)len));
}
```

- **Beat Detection**: In the `processing` function, the program calculates the average energy of the current buffer and updates a rolling window of 20 chunk energies. The frame energy is computed as the average of these chunk energies. A beat is detected if the current chunk's energy exceeds the frame energy scaled by a constant $C = 1.8$, triggering an LED indicator.

Listing 4: Beat Detection Code in volume.c

```
static int processing(int *input, int *output)
{
    int size = BUFSIZE;
    float mean_energy = calc_avg_energy(input, size);
    chunk_energy[index] = mean_energy;
    index++;
    index %= 20;
    tot_energy = 0;
    for(j=0; j<20;j++){
        tot_energy += chunk_energy[j];
    }
    frame_energy = ((float)(tot_energy))/20;
    if(mean_energy - frame_energy*c >= 0){
        flag = 1;
    }
    else flag=0;
    sprintf(v, "%f", mean_energy);
    puts(v);
    return(TRUE);
}
```

Discussion

The `volume.c` program effectively implements the beat detection algorithm by processing the audio signal in real-time on the DSK6416 board. The use of a rolling window of 20 chunk energies to compute the frame energy provides a dynamic baseline for comparison, improving robustness against gradual changes in signal amplitude. The choice of $C = 1.8$ balances sensitivity and specificity in beat detection, though it may require tuning for different music genres or signal characteristics.

The buffer size of 2000 samples, divided into 20 chunks of 100 samples, aligns with the lab's recommendation and ensures efficient processing within the DSK board's constraints. However, as noted in the lab instructions, larger buffers could improve energy estimates but risk missing rapid beats and increase processing time. The code's use of LEDs to visually indicate detected beats provides a clear output for verification during the experiment.

Future improvements could include adaptive tuning of the sensitivity constant C based on signal characteristics or implementing additional signal processing techniques, such as filtering, to enhance beat detection accuracy. Additionally, optimizing the `dataIO` function for specific input sources could improve the program's flexibility.

Impact of Coefficient C on Beat Detection

The sensitivity of the beat detection algorithm depends on the coefficient C, which sets the energy threshold for detecting beats. Lowering C (e.g., 1.8 to 1.2) reduces the threshold, increasing beat detection sensitivity but potentially **causing false positive**s due to noise. Increasing C (e.g., 1.8 to 2.5) raises the threshold, focusing on prominent beats while potentially **missing subtler ones**. Dynamic adjustment of C based on signal characteristics can optimize detection performance.

Conclusion

This lab successfully demonstrated the implementation of a beat detection algorithm using energy-based signal processing on the Texas Instruments DSK6416 board. The `volume.c` program accurately calculates buffer and chunk energies, detects beats based on a sensitivity threshold, and provides visual feedback via LEDs. The experiment highlights the trade-offs in buffer size selection and the importance of real-time processing constraints in digital signal processing applications.