Self-Choreographed Musical Fountain System

Andrea Natasha Gonsalves[1], Claran Joel Martis[2], Nandini Maninarayana[3]
[1,2] *B.E. Student, Department of Electronics and Communication Engineering, SJEC, Mangaluru*
[3] *Associate Professor, Department of Electronics and Communication Engineering, SJEC, Mangaluru*

Abstract –

Self-choreographed musical fountain is a novel idea and aims to reduce design complexity of pre-choreographed musical fountains by automatically synchronizing water with random input music, depending on musical information retrieval characteristics of the song. In this paper, a novel algorithm developed for a self-choreographed, fully automatic, interactive musical fountain system is discussed. System design follows a modular approach. Python language is used to develop the software architecture for the system. A prototype is designed to test the system. System provides satisfactory results for various genres of music.

Keywords - *Chord Changes; Feedback System; Music Information Retrieval (MIR); SCPGA; Segmentation; Self Choreographed Musical Fountain*

I. INTRODUCTION

Fountain is an architectural piece which jets water into air to produce a dramatic effect. Fountains are used today to decorate city parks, hotels and offices for entertainment and attraction. Musical fountain is achieved by synchronizing light and water jets to the beats of music to produce a theatrical spectacle. Earlier, musical fountains were controlled manually by an operator, who regulated pump, valves and light by using a switchboard. Currently, most musical fountains available are pre-choreographed to sync water, lights with the music played. Once such fountains are setup, the choreographed routines are hardly changed. This repetitive nature is the reason for the poor audience attending these musical fountain shows. Pre-programmed musical fountain is a complex system, requiring trained and skilled personnel. To improve such musical fountains, a self-choreographed musical fountain system is proposed. Self-choreographed musical fountain system reduces maintenance and saves time. Research in this field has resulted in several computer aided simulation systems [1]. Self-choreographed fountains have been implemented using FFT, to turn on valves corresponding to different frequency ranges in music. This doesn't truly capture the essence of music. The proposed system allows the user to input songs through a graphical user interface (GUI). The song is segmented according to location of chord changes. Various MIR [2] parameters are extracted for every segment, using Essentia. Suitable patterns are selected from the database using proposed algorithm and corresponding signals are sent to the valves and servos. Pipeline pressure is regulated using a feedback system. The rest of this paper is organized as follows. In section II, the system design approach is discussed. Section III and IV provides an overview of implementation stages. In section V, experimental results and performance of the fountain are presented. Finally, conclusions and possible future work directions are given in section VI.

II. SYSTEM DESIGN

Proposed system depicted in Figure 1, takes input in form of audio files, through the GUI which provides necessary controls. The audio file is segmented, according to chord changes [3], [4]. For each segment, MIR Parameters such as danceability [5] and duration are computed. Suitable sequence of water jets is selected from the pre-defined pattern database based on extracted MIR parameters, using proposed algorithm. Corresponding signals are then sent to valve driver board and servo controller. Valve driver board is responsible for switching of the solenoid valves, which controls the flow of water through the nozzles. Servo drive controls movement of servo motors. Pressure sensor is used to provide feedback to the pump, to maintain required pressure in the header, by varying the speed of the pump, using TRIAC based motor controller circuit which controls the AC power given to the pump [6].

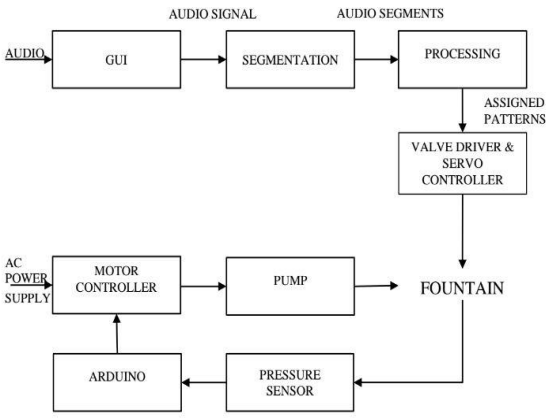

Figure.1: Block Diagram

III. SOFTWARE IMPLEMENTATION

Software architecture of the self-choreographed musical fountain system implemented is illustrated in Figure 2. Architecture uses modular approach and consists of 3 subsystems, Linux system, Raspberry Pi and Arduino. Linux system handles, GUI and audio signal processing.

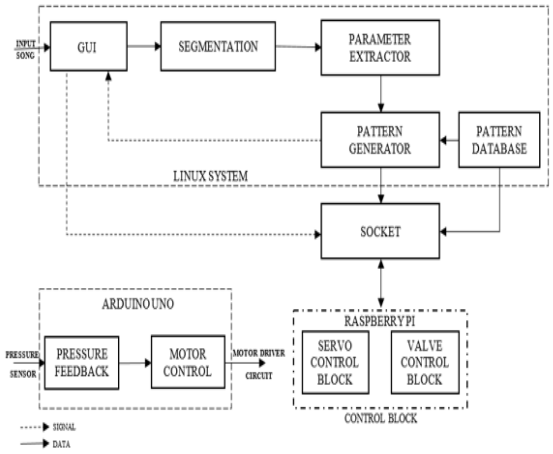

Figure.2. Software Architecture

The Linux system used is Ubuntu 14.04. Raspberry Pi, is used to control valves and servos. Sockets are used to communicate between Linux system and Raspberry Pi. Arduino is used to monitor and maintain pressure in the system. Sockets are used to communicate between the two operating systems.

A. Graphical User Interface

Front end of the system named Aqua Harmony, is designed using Python programming language as shown in Fig 3. The system is created using Python packages, Tkinter, Pygame, PIL and Shutil. The GUI is based on event-driven programming. Through the GUI, a music file can be inserted into the playlist from external source such as flash drive, CD or any other internal source on the computer. GUI provides Start, Pause, Unpause, Stop and Volume Control options for the user. Copy of the imported music file song will be saved in local drive for ease of further processing. Once playlist is created locally, user can use load button to begin background processing.

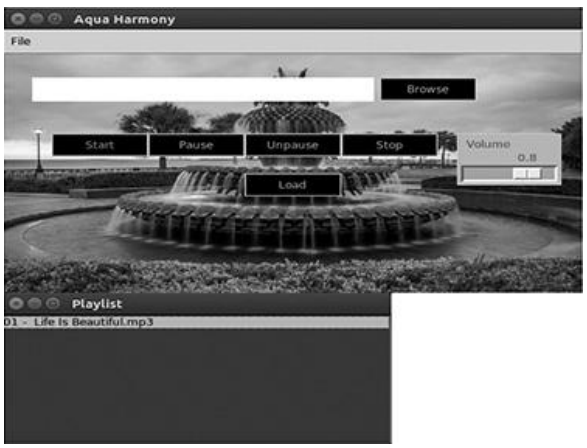

Figure.3: Graphical User Interface

B. Segmentation

Segmentation process detects location of chord changes. Chord is a combination of 3 or more music notes that is heard simultaneously. Segmentation pseudo code is as follows.

```
float[] Segmentation(music file)
{
FOR every 2048 samples
        READ chord and location of chord
IF current chord is equal to next chord
        DELETE next chord
IF duration between current chord and next
chord less than1 second
        DELETE next chord
    IF segment duration is greater than lower
    threshold
        SELECT segment
    ELSE
        Combine segments till greater than
        upper threshold
    RETURN duration of segment
}
```

C. Pattern Database

Water jet patterns are represented by a pre-assigned pattern constant, which are specified before the installation of the system. Patterns are manually arranged in ascending order, based on parameters such as, danceability and duration. Danceability (d) parameter has higher priority than duration (s). Proportionate values are assigned to each pattern, for both parameters. Sum of these values are normalized and represented as sd_{normal} as shown in equation 1. Pre-assigned pattern constant depends on system's degrees of freedom (DOF).

$$sd_{normal} = (s + d)_{normalized} \qquad (1)$$

D. SCPGA

An automatic pattern generation algorithm, SCPGA (Self choreographed pattern generation algorithm) is designed using novel concepts and techniques, to derive a mathematical relationship between patterns and music. Parameters are extracted from real time segments, to obtain a real time pattern constant, which is compared with a database of pre-assigned pattern constants. SCPGA consists of 3 main functions.

- Computation of real time parameters.
- Determination of real time pattern constant.
- Selecting suitable pattern from database.

1) Computation of Real Time Parameters

MIR parameters are extracted for each real time segment. Danceability was found apt to reflect the dynamism of music as shown in Figure 4. Hence danceability is considered as primary parameter and its value ranges from 0 to 3. The pre-defined Patterns in the database are suitable for variable duration. Duration of each segment is also computed and its minimum value is 2.5s.

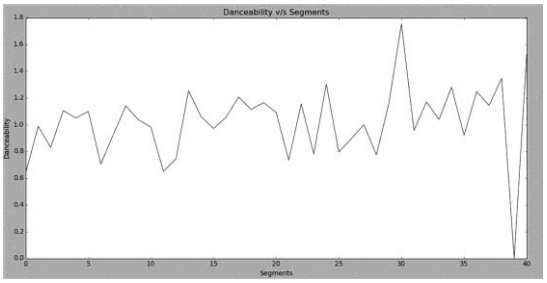

Figure.4: Variation of Danceability for 'Beautiful in White'

*2) **Real Time Pattern Constant***

Real time pattern constant is computed as follows

$$sd_{real} = (s + d) \quad (2)$$

$$sd_{normal\ ,real} = map[(sd_{real}), \min(sd_{normal}), \max(sd_{normal})] \quad (3)$$

Mapping ensures that the computed real time pattern constant is in the same range as pre-assigned pattern constant.

*3) **Selecting suitable Pattern from Database***

Obtained real time pattern constant sd_{normal_real} is compared with preassigned pattern constant sd_{normal} of the patterns in the database. The water jet sequence with its pattern constant closest to sd_{normal_real} is selected as the suitable pattern for the segment. In case of discrepancies, i.e. if sd_{normal} real matches with 2 or more patterns in the database, a suitable pattern is selected depending on the occurrence rate of the patterns. Occurrence parameter brings a certain amount of randomness. Occurrence parameter is checked in a circular manner which ensures that every equally probable pattern is given equal importance. Initial priorities are assigned for patterns which are likely to be selected together after first level of comparison. Every iteration, when a pattern is selected, occurrence parameter of corresponding pattern is incremented.

IV. HARDWARE IMPLEMENTATION

Hardware Architecture of the implemented system is shown in Figure 5. Half HP Pump draws water from source and injects it to the header. Header is a hollow pipe, used to regulate pressure and has 3-inch diameter with 7 outlets of 3/4 inch and a single outlet of 1/2 inch. 1/4-inch pressure sensor is connected to the header outlet of 1/2 inch via a reducer.

Independent feedback loop consisting of pressure sensor, Arduino and TRIAC based motor control circuit is used to maintain constant pressure in the header by controlling the speed of the motor by varying voltage. Motor Control Circuit as shown in Figure 6, consists of a Zero Crossing Detector implemented using opto-coupler 4N25, and TRIAC triggering circuit. The triac used is BT136 [6]. The outlets from the header are connected to 7 solenoid valves which are controlled by the valve driver (relay board). Depending on the water jet sequence the valves are turned on or off. The valves are connected to nozzles of internal diameter 2.5mm using rubber pipes to get a smooth jet of water. Nozzles are also attached to servo motors which have a swing of 0◦ to 180◦.

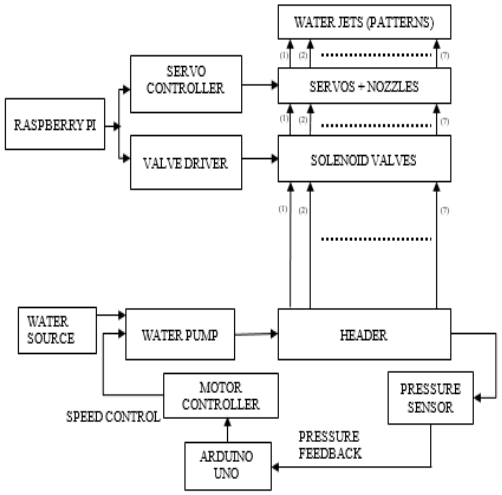

Figure.5. Hardware Architecture

Using these motors, a wide variety of patterns can be obtained. Servo motors are controlled by servo controller PCA-9685. Raspberry Pi is used to control valve driver and servo controller.

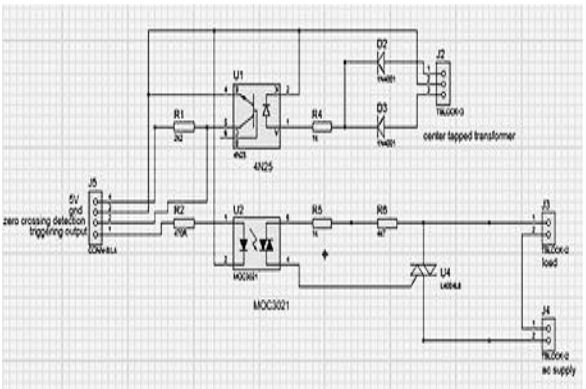

Figure.6: Motor Control

V. EXPERIMENTAL RESULTS AND ANALYSIS

Proposed self-choreographed musical fountain was designed and implemented using seven linearly positioned nozzles, valves and servos. Various tests are performed during the course of the implementation and analysis of the results obtained are discussed below. Total time required for processing depends on the number of songs in the playlist and duration of the songs. Songs are segmented depending on location of chord change. Results obtained for songs of different genre are described in Table I.

Table I : Computation Time and Number of Segment Obtained

Song	Duration (s)	Computation Time (s)	Number of Segments
Beautiful in White	232.82	3.387	41
Firework	228.31	4.05	43
Climb	236.59	4.25	43
Jail House Rock	152.11	2.95	27
Titanic (Theme)	310.93	5.32	52

Variation in height of water jets is obtained by varying speed of the motor by variation of triac firing angle. Five different levels of achieved height are shown in Fig 7.

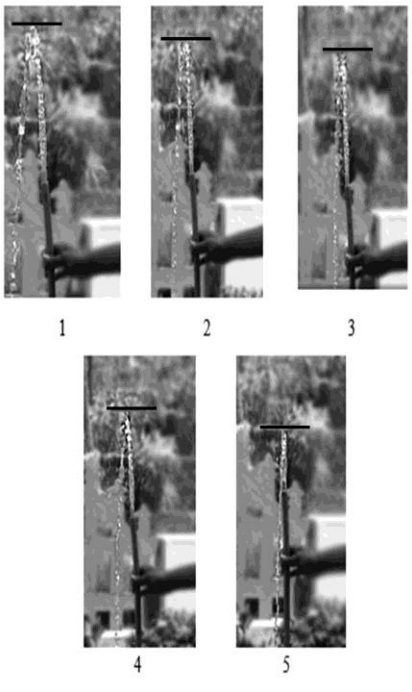

Figure.7: Variation in Height of Water Jet

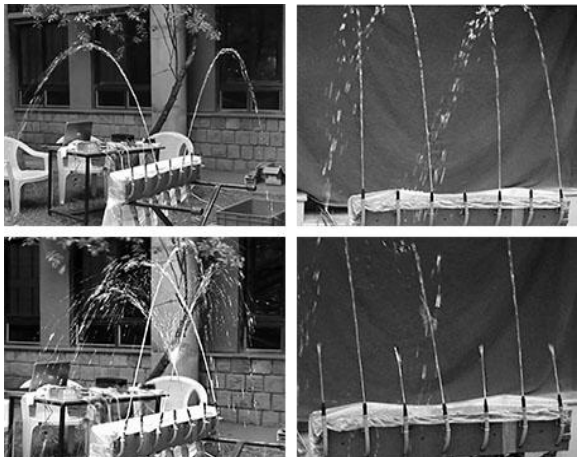

Figure.8: Patterns Obtained During Test

Database consisting of 35 pre defined water jet sequences, is designed based on hardware implementation. Pre-assigned pattern constants defined for each sequence, are tabulated in Table II. Obtained random water jet patterns, during testing is shown in Figure 8, which is efficiently in sync with music played.

Wireless remote access is employed to control the working of the fountain which provides enhanced convenience and pleasant user experience.

Table II : Pattern Database

Pre-assigned Pattern Constants	Pattern Number
0.0997	9
0.0997	3
0.1286	8
0.1994	2
0.1994	7
0.2283	5
0.2519	1
0.2703	17
0.2703	18
0.2808	4
0.2808	6
0.2992	12
0.4041	10
0.4566	11
0.4619	23
0.4619	15
0.4855	14
0.5380	16
0.5721	20
0.5853	13
0.6141	19
0.6141	22
0.6482	26
0.6482	25
0.6771	29
0.6771	30
0.7060	31
0.7060	32
0.7139	28
0.7427	27
0.7427	21
0.7532	35
0.9475	34
0.9580	24
1.0	33

VI. CONCLUSION

Fully automatic, self-choreograph musical fountain system is designed and implemented. System provides several advantages over existing pre-choreographed musical fountain. A novel algorithm is developed to choreograph the system based on musical characteristics of the song. System design can

be improvised by implementing entire system on a single computing system. Future systems can make use of more complicated fountain movements. System can be upgraded to support choreographing live music.

Table III : Computed Values and Pattern Assigned for Song Beautiful in White

Sample Number	Duration of Sample (s)	Dance-ability of Sample	Computed Value	Assigned Pattern
1	6.59	0.64	0.7132	28
2	5.25	0.99	0.5341	16
3	6.08	0.83	0.6560	25
4	5.76	1.11	0.6472	26
5	7.38	1.05	0.9289	34
6	6.22	1.09	0.7294	27
7	2.55	0.71	0.0	28
8	5.52	0.93	0.5734	20
9	7.61	1.14	0.9870	33
10	5.15	1.04	0.5264	16
11	7.85	0.98	1.0	33
12	2.92	0.65	0.0570	9
13	3.20	0.74	0.1236	8
14	6.22	1.25	0.7573	35
15	6.22	1.06	0.7223	28
16	6.04	0.97	0.6731	29
17	7.29	1.05	0.9134	34
18	5.94	1.21	0.6987	31
19	6.82	1.11	0.8404	35
20	5.06	1.16	0.5327	16
21	7.61	1.09	0.9780	24
22	3.24	0.73	0.1302	8
23	6.36	1.15	0.7645	35
24	5.52	0.78	0.5472	16
25	5.38	1.3	0.6160	22
26	7.06	0.79	0.8250	35
27	5.11	0.89	0.4928	14
28	7.15	0.99	0.8783	34
29	2.78	0.77	0.0538	35
30	5.85	1.16	0.6739	30
31	5.15	1.75	0.6552	25
32	5.85	0.96	0.6371	26
33	6.13	1.17	0.7254	28
34	5.29	1.04	0.5517	16
35	5.89	1.28	0.7037	32
36	5.99	0.92	0.6557	25
37	5.39	1.25	0.6059	19
38	5.48	1.14	0.6040	19
39	5.76	1.35	0.6906	30
40	7.19	0.0	0.7071	32
41	2.92	1.53	0.2140	5

REFERENCES

[1] D. Liu, N. Zhang, and H. Zhu, "Computer aided design system for developing musical fountain programs," Tsinghua Science and Technology, vol. 8, no. 5, pp. 612–616, Oct 2003.

[2] J. S. Downie, "Music information retrieval," Annual Review of Information Science and Technology, vol. 37, no. 1, p. 295340, 2005.

[3] H. Papadopoulos and G. Peeters, "Local key estimation from an audio signal relying on harmonic and metrical structures,"IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no. 4, pp. 1297–1312, May 2012.

[4] D. Temperley, "What's key for key? the krumhansl-schmuckler key finding algorithm reconsidered," Music Perception: An Interdisciplinary Journal, vol. 17, no. 1, pp. 65–100, 1999.

[5] P. Herrera and S. Streich, "Detrended fluctuation analysis of music signals: Danceability estimation and further semantic characterization," in Audio Engineering Society Convention 118, May 2005.

[6] Y. V. N. Kumar, P. H. Bindu, A. D. Sneha, and A. Sravan, "A novel implementation of phase control technique for speed control of induction motor using arduino," International Journal of Emerging Technology and Advanced Engineering, vol. 3, no. 4, p. 469473, Apr 2013.