1city1001vibrations : development of a interactive sound installation with robotic instrument performance

Sinan Bökesoy
composer &multimedia artist
Büyükada, Istanbul
sinan@sonic-disorder.com

Patrick Adler
Robot Engineer
Augsburg, Germany
paadler@gmx.net

ABSTRACT

"*1city1001vibrations*" is a sound installation project of Sinan Bökesoy. It does continuous analysis of live sounds with the microphones installed on top of significant places at Bosphorus - Istanbul. The transmitted sounds are accompanied by an algorithmic composition derived from this content analysis for controlling two Kuka industrial robot arms performing the percussions installed around them while creating a metaphor through an intelligent composition/performance system. This paper aims to focus on the programming strategies taken for developing a musical instrument out of an industrial robot.

Figure 1. The installation venue in Istanbul, July2010

Keywords
Sound installation, robotic music, interactive systems

1. INTRODUCTION

Robots capture great interest while creating an existence by providing physical and visual cues to the audience. (Figure 1) Our aim was creating a metaphor by translating the Bosphorus sounds to the vibrations of percussion surface of various ethnic percussions. As example applications using robots, there are mechanical systems offering one degrees of freedom by applying the hit stroke to a fixed point of percussion surface within the dynamics of a hammer action [3][6][7]. We used an industrial robot with 6 degrees of freedom offering the movement potential close to the human arm. A communication protocol with semantic description of the robot control commands has been developed with an interface between the robot software and the controlling musical application.

1.1 Listening to the "Bosphorus" sounds

The microphones, as the 'ears' of the robots do transmit the signals as internet audio stream via an AudioIP unit. This audio-stream is being interpreted by an analysis application (Figure 2) with the computer at the installation venue.

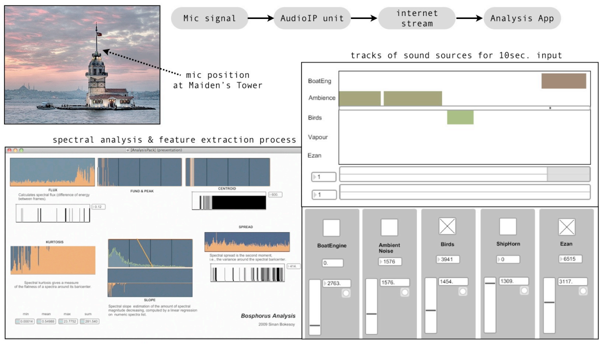

Figure 2. Transmitting the Bosporus sounds to the analysis application and tracking of the sound sources on Max/MSP

The analysis software is a Max/MSP[2] application and based on the interpretation of the spectral analysis of the audio stream with the help of the powerful Zsa.Descriptors [5]. The feature extraction process from the spectral data aims to recognize 5 specific sound sources found on everyday routine of the Bosphorus. The analysis software is reporting the onset times of the sound events we are interested in. Each tracking of the relevant sound source is used to create a percussion score according to the rules defined for the translation of Bosphorus sonic space to percussion gestures.

The input of the interactive sound system is the content analysis of the audio-stream coming directly from Bosphorus. The output of this patch is the percussion score to be performed by the robots and also some of signal processing tools to create a dispositive sound distribution at the installation venue. A 10sec. long audio analysis frame will be captured live and the analysis will be sent after each 10sec. frame to the algorithmic composition patch. Therefore the performed percussion score belongs to 10sec. in the past. Each robot has access to 2 types of Istanbul cymbals, 1 gong, one glockenspiel, 1 davul (a ethnic drum), 1 bendir (a ethnic hand drum) and 2 tibetian bowls. The generated rhythmic pattern for the relevant instrument is influenced by the length of the tracked source, and also on some statistical distribution of spectral parameters. The translation of percussion score is achieved by dedicated XML strings, which will be interpreted by the robot software. Each of the five tracks corresponds to a sound source in the analysis.

The application selects the tracks belonging to the sound sources occurring with highest density in each 10sec. There are 39 different gestures programmed on the robot software, and these can be called with appropriate parameters to modify the gesture according to the performance needs.

2. THE DEVELOPMENT PHASE

We use 2 Kuka KR16 industrial robots having 6axis motion capability, 1.5m reach capacity and up to 15kg load capacity. First, we intended to investigate the dynamics of drumming with sticks, since the percussion timbre is highly dependent on the factors including contact area, hit damping duration and pressure on the drum skin. By being able to control these parameters and simulate them with robot gestures we can achieve the timbre richness and sonic variety for each instrument.

2.1 The robot gripper part

The gripper of the robot is attached at the flange of the robot, positioned next to the 6^{th} axis motor A6. The drum sticks consist of parts named as the *butt, shaft*, *shoulder* and the *tip*. In our gripper design; the butt part, which would be grasped normally by a human hand, is positioned inside a spring for holding the stick tightly. (Figure 3) Our gripper can hold 3 different sticks; wooden, soft and plastic stick. The robot can turn the gripper with the A6 motor, so that the chosen stick can hit on the vertical axis to the percussion surface. On Figure 8, there is the list of the programmed gestures along with the selected stick type for each instrument and the impact points.

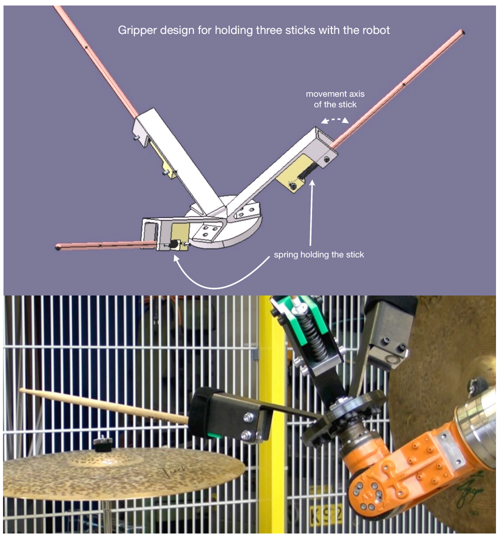

Figure 3. The gripper design (by GD Engineering, Germany) and its application

The mechanism of the "hit gesture" is inspired by a piano hammer action and it is quite simple for quick adjustments. For each stick we have chosen an appropriate spring to realize the required elastic hit motion applied by the gripper. (Figure 3)

2.2 The robot programming part

Kuka Kr16 has 6 degrees of freedom, and the movement on each axis is driven by an independent motor labeled from *A1* to *A6*. To control the motion of the robot each given value is compared to the actual angle of each axis and regulated by motion controllers for each axis. Among the two basic motion types; [4]

PTP (point to point) movement: Each axis uses the shortest way from the actual position to the destination position. PTP can be programmed directly with values for each axis (eg "*A1 30*" in degrees) or with cartesian values referring to a given base (eg "*X 100*" in mm). (Figure 4)

LIN (linear) movement: each axis is controlled for a joined movement; they start and stop its movement at the same time and the slowest axis will cause the others to slow down. The movement is a direct line from the actual position to the destination point.

The cartesian movements have to be related to a base origin, which are referred to the "robot root". The gripper hot point (the sticks end) has to be defined and are always referring to one base except from the *PTP* movements with the axis values in degrees. The rotation, the speed and acceleration for each movement has to be defined within the motion command. KUKA offers spline motion path since software version 5.5, but with the software version we had, our robot-percussion gesture ended up with linear movement path.

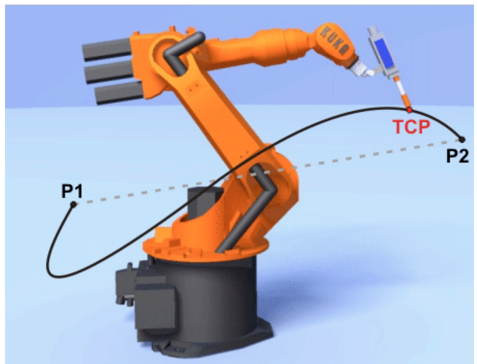

Figure 4. PTP movement: Robot axes are rotational; curved paths can be executed faster than straight paths.

2.2.1 *Communication interface*

We have chosen the *KUKA Ethernet KRL XML* platform to develop our communication interface. Both the MaxMSP and the KUKA Ethernet Interface are working as TCP clients, and it was necessary to develop a bridge application, which receives the commands from MaxMSP and sends them to the robot. The maximum speed of ToolCentralPoint transfer is 2meters/sec. After establishing the basic command transfer, we defined the parameter space for each command according to the dynamics of the "hit gesture" and the common XML structure. (Figure 5)

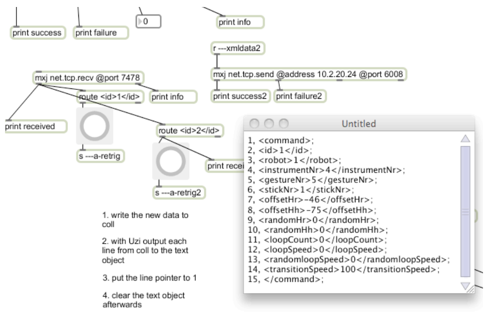

Figure 5. The Max patch building the XML messages and sending to the robot via TCP/IP protocol.

2.2.2 *The dynamics of the "hit gesture"*

The programming of the robot was divided into two parts. First the received commands had to be read, stored to variables and then verified. The stored values were used to recognize the selected instrument and to decide which movement had to be taken in order to move to the instrument without collisions. Then, the stroke parameters had to be applied to a common stroke movement routine. According to the robot coordinate system the impact point of the stick is the end point of the arm. The robot utilizes the 3 transition points to move the arm between instrument groups: for the cymbals *Pt1*, for the drums *Pt2*, and for the bowls and the glockenspiel *Pt3*. From each transition point, it moves to the relevant *Hr* points for the instrument it will hit at the *Hh* point (Figure 6). The *Hr* point is the state of the robot, where it waits ready to apply the "hit gesture" *Hr - Hh - Hr*.

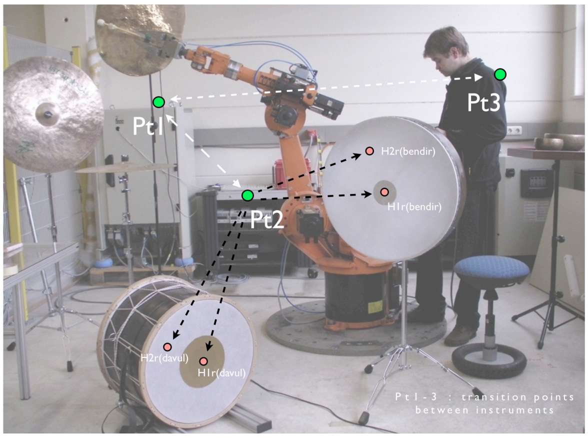

Figure 6. Robot engineer P. Adler testing the software. The robot moves itself between various taught points.

The kinetic hitting force caused by this movement is stored on the spring holding the relevant stick, and then the robot stops necessarily at the *Hh* point but the spring releases the stored energy and the stick continues and hits the surface. This is a basic spring oscillation mechanism applied on the stick axis. (Figure 7)

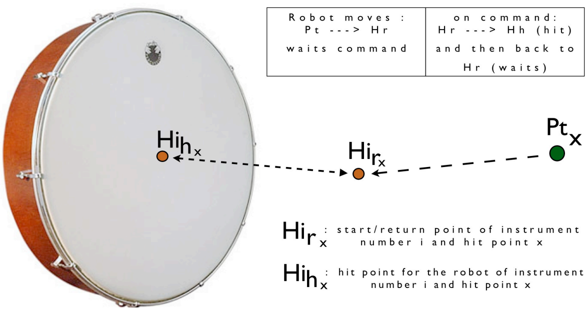

Figure 7. According a command, the robot passes through a transition point and then a *Hr* point to hit at *Hh* point

We proceeded by characterizing the percussion gestures, which will be applied by the robot arm. (Figure 8) Each instrument got its own base (coordinate system) and a common base for the movements between the instruments. Each gesture or transition movement was referred to these bases.

We have developed a special subroutine for the gestures, which uses the quicker/faster axis motors on the arm (A4 - A6, especially A5) for the basic stroke movement and added motion in the kinematics through (A1 - A3) to increase the stroke hardness. This strategy does use the advantages of a robot arm with 6 degrees of freedom against the mechanical arms with one axis movement capability, and the richness of the movement does translate into sound quality. [1]

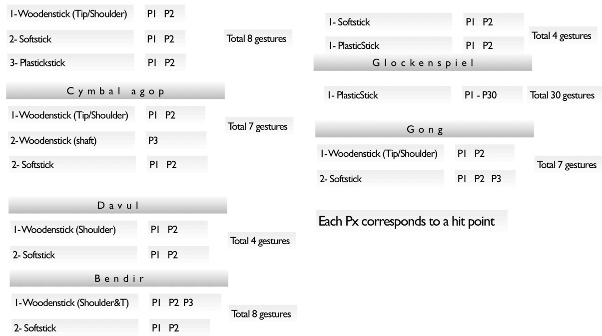

Figure 8. A catalogue of percussion gestures

Our expectations from the robot drummer were reformed with meaningful optimizations of our model according to the dynamic motion and kinematic boundaries of the robot. The deceleration of the robot was in this case (small and therefore fast robot) adequate. Thanks to the spring mechanism there was no negative effect on the speed of the stick.

2.2.3 *The parameter space of the "hit gesture"*

To achieve the natural dynamics of percussion performance, the robot should be able to adjust the sound quality of its gestures in several manners. The analogous method is by striking the percussion surface at different locations with different sticks and contact points. The loudness variety is achieved by hitting harder or softer, which is the amount of pressure on the surface. The angle of the stick on the impact point is also relevant. We have implemented some controllable parameters, which can adjust the position and behavior of the robot arm to modify the impact quality. The following parameters can be adjusted directly within our musical application.

- *Stick type* : 1- Woodenstick 2- Softstick 3- PlasticStick
- *Destination address* : *(Pt) (Hr)* All transition points and hit gesture start/return points available for each instrument are the possible positions of the robot. Instrument number *i* is specified like 1 for Davul 2- Bendir 3- Gong etc.. Hit positions for each instrument are sub categorized as *P1, P2, P3* etc. (Figure 8)
- *Speed* : is a percentage of the maximum speed of the motors driving the axis movement specified between *0* and *100*. The robot specs give the speed in terms of angular speed.
- *Hr position off*: offset value to move the *Hr* position towards the percussion surface or away. 0 is the original taught point.
- *Hh position off*: offset value to move the *Hr* position towards the percussion surface or away. 0 is the original taught point.

Repositioning the *Hr/Hh* points are of crucial importance in order to modify the sound quality achieved. For instance, when both the *Hr* and *Hh* points are modified with the same amount of offset value, the result is hardening or softening the impact while maintaining the speed of the robot movement.

[1] www.c-av-e.com/Kukarobot-bendir.mov

- *Gesture Type* : 0 – Basic Shot, 1 – Full Loop ; With "basic shot", the robot hits the surface and goes back to the Hr point and maintains its position until a next command. When "full loop" is selected, the robot repeats the hit sequence with the specified amount.
- *Loop amount* : The robots does repeat the hit sequence this number of times and the *Hr*, *Hh* and *speed* parameters can be adjusted inside this loop to achieve performance techniques such as accelerando, crescendo with percussion tremolos.

We have also implemented on the robot software controlled randomness for the *Hh* point in terms of adjusting its coordinate on the horizontal plane of the percussion surface, which would humanize the impact since otherwise a robot would always hit to a precise point. During the evaluation process our first hand guide is always our ears, hence it was easy to observe whether the representation of the robot performer model was sufficient to finely represent the subtility of required gesture dynamics or not. We have created a database for each gesture for the easy access of the gesture types with specific dynamics; soft hit, hard hit, slow tremolo – soft hit, fast tremolo fast hit etc.

1,1 1 1 -48 8 33 46 0 -12 4 9 0 -12 0 -12 -75 -33 -26 -7 -75 -33 -26 -7 0 0 570 640 7704 3250 3250 3250;
2,1 2 1 -56 -43 5 1 -26 -11 -13 6 -13 6 8 2 -16 -37 -16 -1 -59 -77 -53 -20 570 570 5160 9970 4860 9568 2150 3974 2150 4028 1754 3200 1686 3192 1716 3222 1810 3352;
3,1 1 2 -74 -15 -5 3 6 16 9 22 9 13 13 32 -24 -32 13 32 -24 32 13 32 620 700 6116 12000 6200 12200 2640 5050 2820 5450 2100 3950 1830 4200;
4,1 2 2 -93 42 32 59 1350 1288 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
5,2 1 1 -56 -5 -85 -12 20 2 1 9 7 0 37 -2 2 -13 9 8 -40 -38 10 -7 530 600 8100 16000 7900 15600 3000 5910 3300 6350 2200 4200 2200 4270 1940 3754 1800 3300;
6,2 2 1 -30 -25 -27 31 2 13 -3 32 -3 -19 -3 39 -3 16 13 23 19 1 -8 30 450 500 4600 8850 4500 8682 1940 3722 2230 4230 1560 2860 1700 3200 1644 3040 1620 3000;
7,2 3 1 -78 -5 -32 16 1230 1160;

2.3 Event scheduling

In a musical performance knowing precisely each process time is necessary. When the robot receives a command, it sends back an acknowledgement to the MaxMSP application. (Figure 9) It does also send task finished messaged when the robot arm returns back to the specific *Hr* position. We define the time of command receive as $T_{command(x-1)}$ for the hit event (x-1). Then the total task period, which lasts for the "hit gesture" is;

$$T_{return(x-1)} - T_{command(x-1)} = t_{task(x-1)}$$

We do calculate this time interval easily by starting a clock with the send of the command and stopping it with the task end message received from the robot. This is a full cycle as shown on Figure 8. We need to know, the time interval between the command receive and the surface contact of the robot precisely. We name the moment of impact as $T_{hit(x-1)}$, and the hit process interval can be calculated with;

$$T_{hit(x-1)} - T_{command(x-1)} = t_{hitproc(x-1)}$$

We use a microphone to capture precisely the act of surface impact on the percussion. For instance with the help of the maxmsp bonk~ object[8] we can detect the hit event to stop the clock. Between $T_{command(x)}$ and $T_{return(x-1)}$ the robot will be busy. The musical application has to consider this fact. Therefore we had to catalogue all possible $t_{hitproc(x)}$ and $t_{task(x)}$ values, which means $C(37,2) = \frac{39!}{37!2!} = 741$ values to be stored for both. For this purpose, another max patch has been used[2] in to automate this process. The patch measures all 741 $t_{hitproc(x)}$ and $t_{task(x)}$ values and prepares a matrix.

[2] www.c-av-e.com/gest-walk.mov

Therefore we define much time in advance we need to send the command to the robot at its last position to get a hit on a certain time $t_{(x)}$ and what would be the minimum task duration.

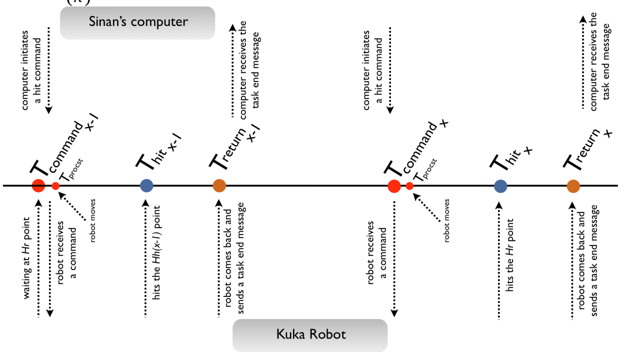

Figure 9. The chronological view to the event schedule and message exchange between 2 computers.

A high level musical representation of this structure has been required for direct score –to- robot gesture translation. We have chosen Ableton Live[1] with the Max4L addon, since it can directly integrate a MaxMSP patch, which does translate the incoming sequencer messages to XML messages interpreted by the robot software. The track midi information is translated to relevant XML strings with proper time stamping of events as explained on section 2.3 by checking the $t_{hitproc(x)}$. A midi note represents a percussion gesture. Continuous midi controller messages assigned to it are translated to real-time parameters such as gesture speed, *Hr* and *Hh* position of the stick and the loop amount.

3. ACKNOWLEDGEMENTS

My thanks to Kuka Robotics, Germany, which has supported the project by providing us 2 KR16 robots and also has assisted to the organization of robot motion programming and logistics. This project has been realized with the financial support of *Istanbul2010 European Capital of Culture Agency*.

4. REFERENCES

[1] Ableton – Ableton Live Suite, www.ableton.com
[2] Cycling74 – MaxMSP, http:www.cycling74.com
[3] Kapur A., Darling M. A Pedagogical Paradigm for Musical Robotics. In *Proceedings of the Conference on New Interfaces for Musical Expression*, (Sydney, Australia, 2010)
[4] Kuka Robot Group, *Kuka System 5.4 Software Reference Manual.* Issued on 24.04.2008.
[5] Malt, M. and Jourdan, E. Real-Time Uses of Low Level Sound Descriptors as Event Detection functions Using Max/MSP Zsa.Descriptors. In *Proceedings of the SMCM9 conference.* (Recife, Brasil, 2009)
[6] Singer, E., Feddersen, J., Redmon, C. and Bowen B. LEMUR's Musical Robots. In *Proceedings of the NIME Conference* (Hamamatsu, Japan, 2004)
[7] Weinberg, G., Driscoll S., Parry M. Musical Interactions with a Perceptual Robotic Percussionist. *Proceedings of IEEE International Workshop on Robot and Human Interactive Communication*. (Nashville, TN 2005).