Creating Musical Expression using Kinect

Min-Joon Yoo
Yonsei University, South Korea
Eng Bld C533, Sinchondong
Seoul, South Korea
debussy@cs.yonsei.ac.kr

Jin-Wook Beak
Yonsei University, South Korea
Eng Bld C533, Sinchondong
Seoul, South Korea
alleykat@cs.yonsei.ac.kr

In-Kwon Lee
Yonsei University, South Korea
Eng Bld C533, Sinchondong
Seoul, South Korea
iklee@yonsei.ac.kr

ABSTRACT
Recently, Microsoft introduced a game interface called Kinect for the Xbox 360 video game platform. This interface enables users to control and interact with the game console without the need to touch a controller. It largely increases the users' degree of freedom to express their emotion. In this paper, we first describe the system we developed to use this interface for sound generation and controlling musical expression. The skeleton data are extracted from users' motions and the data are translated to pre-defined MIDI data. We then use the MIDI data to control several applications. To allow the translation between the data, we implemented a simple Kinect-to-MIDI data convertor, which is introduced in this paper. We describe two applications to make music with Kinect: we first generate sound with Max/MSP, and then control the adlib with our own adlib generating system by the body movements of the users.

Keywords
Kinect, gaming interface, sound generation, adlib generation

1. INTRODUCTION
Kinect (http://www.xbox.com/kinect) is a new game interface for Microsoft's Xbox 360 game console. This interface enables users to control the console with their natural motion. This freedom is achieved by analyzing the image and sound of the users that the camera and microphone of the Kinect capture. Since this 'controller-free' interface has the ability to extend the degree of freedom and expressiveness of the users, many researchers and developers have tried to apply the interface in such a way that it not only controls the game console, but also controls their own applications. Through their efforts, open source drivers for Kinect have recently been developed and released, and various applications of Kinect have now been presented.

In this paper, we introduce our system, designed to create and control music according to the user's body motion via Kinect. Through Kinect's ability to interpret a user's whole motion, users can control the sound generation system with the rich range of movement of the body.

First, we extract skeleton data from the user's body. We can obtain the position and velocity of each joint of the user's body from the skeleton data. The joint data is then converted to MIDI data, which was defined previously. To translate the data, we implemented a Kinect-to-MIDI data translator.

Since the data from the user's body is now translated to MIDI data, general programs responding to the MIDI commands can be used to create sound or visualization. We tested Max/MSP for this purpose. We created various sounds with parameters converted from body motions in Max/MSP. We then applied this interface to our adlib generation program. The movements of the entire body were interpreted to generate control parameters for adlib.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME'11, 30 May–1 June 2011, Oslo, Norway.
Copyright remains with the author(s).

2. KINECT
Kinect, officially launched in October 2010, is the first to achieve the elaborate full-body control with 3D motion capture. This interface is based on technology developed by Rare and PrimeSense for game technology and image-based 3D reconstruction. Aided by these technologies, this interface provides several useful functions to enable natural interaction.

Kinect consists of three devices: the RGB camera, the depth sensor, and the multi-array microphone (Figure 1). The camera can output video at 30 Hz frames with 8-bit VGA resolution (640 x 480 pixels). The monochrome depth sensor can then sense the depth information, also with VGA resolution with 11-bit depth, which provides 2,048 levels of sensitivity. The practical range limit of the sensor is 1.2-3.5 m (3.9-11 ft), and the sensing range is adjustable. By interpreting the data obtained from the camera and depth sensor, we can realize the 3D motion capture of the users. Kinect also has a microphone array consisting of four microphone capsules. These devices operate with each channel processing 16-bit audio at a sampling rate of 16 kHz. You can find more detailed information about the devices on the Kinect homepage and Wikipedia (http://en.wikipedia.org/wiki/Kinect).

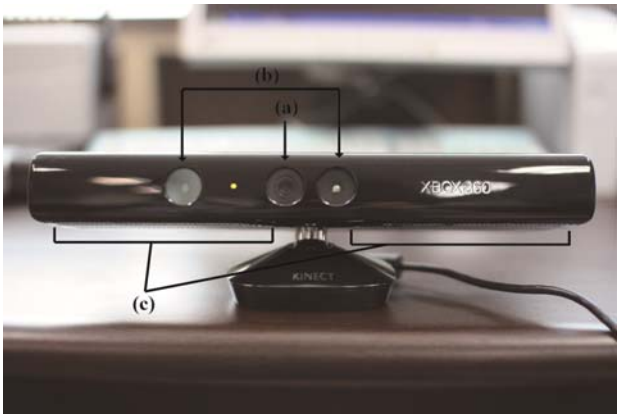

Figure 1. Kinect consists of three components: (a) RGB camera, (b) depth sensors, and (c) microphone array.

3. SYSTEM DESCRIPTION

We used modules developed in OpenNI (http://www.openni.org/) to connect Kinect with a PC. The PrimeSense's *NITE* provided useful APIs for the manipulation of naïve data. We also use FAAST (http://projects.ict.usc.edu/mxr/faast/) to easily access the joint data extracted from the body motion. The FAAST streams the user's skeleton data over a VRPN server (http://www.cs.unc.edu/Research/vrpn/).

We then implemented a Kinect-to-MIDI convertor (see Figure 2). This program listens to messages from the VRPN server embedded in FAAST, and responds if proper messages are received. We predefined a mapping between the joint data and the MIDI data. Thus, the input joint messages are translated to MIDI messages using this mapping. Finally the MIDI messages are sent to a MIDI-IN port.

Our translator is operated in a similar way to the Wii-to-MIDI translators such as GlovePIE or OSCulator. We were inspired by these programs for creating music using game controllers.

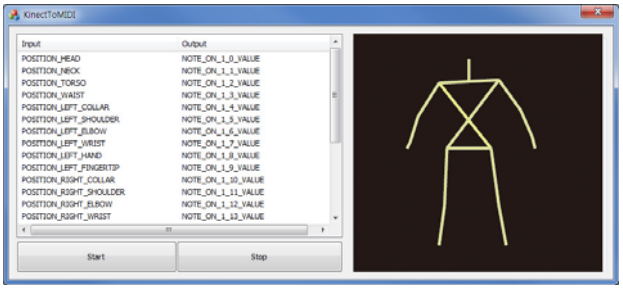

Figure 2. Kinect-to-MIDI convertor.

4. SKELETON DATA

FAAST streams a total of 24 skeleton joint data tracked in the input video. The joints are ordered corresponding to the OpenNI framework (Table 1). As you can see in this table, the skeleton data contains all the positions of the major parts of the body, and we can reconstruct the movement of the body approximately from the skeleton data.

Table 1. The skeleton joint order by OpenNI framework.

Sensor	Joint	Sensor	Joint
0	Head	12	Right Elbow
1	Neck	13	Right Wrist
2	Torso	14	Right Hand
3	Waist	15	Right Fingertip
4	Left Collar	16	Left Hip
5	Left Shoulder	17	Left Knee
6	Left Elbow	18	Left Ankle
7	Left Wrist	19	Left Foot
8	Left Hand	20	Right Hip
9	Left Fingertip	21	Right Knee
10	Right Collar	22	Right Ankle
11	Right Shoulder	23	Right Foot

5. APPLICATIONS

5.1 With Max/MSP

With our Kinect-MIDI convertor, any music application controlled MIDI data can be used to create music and sound. We chose the Max/MSP program to test the controllability of Kinect because of its efficiency in creating and modifying sound. We implemented a Max/MSP patch, which generates sounds by several parameters.

The skeleton data can be roughly divided into five segments: center (0-3), left arm (4-9), right arm (10-15), left leg (16-19), and right leg (20-23). Thus, we generated five sounds corresponding to the body segments, controlled by 4-6 parameters also corresponding to the joint data belonging to each segment. For example, one sound corresponding to the left leg had four parameters, and these parameters are controlled by the movements of the left hip (16), left knee (17), left ankle (18) and left foot (19).

We then mapped the velocity of the joints to the parameters of the sound. The velocity of a joint can be easily calculated by the difference between the position of the joint in one video frame and the position in the previous video frame. By using velocity data, users could change the sound more intuitively.

5.2 With Our Adlib Generator

We tested our system in our own adlib software. The software was originally implemented to generate an adlib sequence via the user's line drawing. The user drew lines using a mouse interface, and the adlib was generated by data obtained using the movement and position of the mouse. We extended the software by changing the input from mouse to Kinect.

The speed and pitch of the adlib was controlled by the average velocity and position of each joint, respectively. The scale of the adlib was then determined by the relative position of each end-joint (head (0), left fingertip (9). right fingertip (15), left foot (19), and right foot (23)). We predefined the positions of the end-joints generating a normal pose. Then, if the end-joints were closer to each other than the normal pose, (the user's pose shrunk), the adlib was created with a diminished scale. Also, the farther the end-joints were from each other, the tenser the scales used to generate adlib. Compared with the previous control using a mouse, this system can express more adlib styles because of the more intuitive controller. We are now conducting user surveys about the control method and resulting adlibs.

6. CONCLUSION

We first reported a system to generate and control sound with Kinect. Several open-source drivers and modules were used to extract the position data from the movement of a user's body. We then converted the data into the MIDI messages by using our Kinect-to-MIDI translator. Because the data from the body motion can be presented in MIDI messages, any music application responding to MIDI data can be used to create and control music. We first tested this method with Max/MSP software, then with our adlib generator. In our prototype tests, we could control the music more intuitively using body motions. We are exploring more applications and methods for controlling music using Kinect and also doing more theoretical research by studying literature related to analysis of movement. .