HAPTIC AND SOUND RENDERING OF A VIRTUAL MILLING PROCESS

Cui-Fei Chang
Department of Applied Mathematics
State University of New York
Stony Brook, New York 11794

Amitabh Varshney
Department of Computer Science
University of Maryland
College Park, MD 20742
varshney@cs.umd.edu

Q.J. Ge
Department of Mechanical Engineering
State University of New York
Stony Brook, New York 11794-2300
qge@notes.cc.sunysb.edu

ABSTRACT

This paper deals with the development of a multisensory virtual environment with visual, haptic, and aural feedbacks for simulating the CNC milling process. The paper focuses on the haptic and sound rending of the virtual milling process. The haptic rendering provides the user with kinesthetic information and tactile information. The kinesthetic information is displayed by the cutting force of a milling machine. The tactile information is demonstrated by the haptic texturing. The sound rendering simulates the machine sound and provides the aural feedback to the user. By applying the concepts of image-based rendering, both haptic rendering and sound rendering speeds are accelerated by pre-sampling related process parameters in an perception-dependent way.

1 INTRODUCTION

A multisensory virtual environment has multiple sensory feedbacks such as visual, haptic, and aural feedbacks. This paper deals with the development of a multisensory virtual environment for modeling 5-axis CNC (Computer Numerically Controlled) milling processes. A virtual environment system has two basic elements: *rendering* and *animation*. Rendering is the process of producing a representation of a virtual world that can be displayed using whatever two-dimensional or three-dimensional output devices are available. Sometimes, we replace *rendering* with *displaying*. Animation is the process of repeatedly displaying slightly different views of the world to represent changes that occur, either through the actions taken by the user or under the control of the system. Sometimes, we may use *motion* instead of *animation*.

The cutter motion of the CNC milling process is described as a freeform curve such as a Bézier or B-spline curve in the space of dual quaternions. The algebra of quaternions and dual quaternions as well as their application in kinematics can be found in Bottema and Roth (1990) and McCarthy (1990). Various algorithms for geometric design of freeform dual quaternion curves can be found in Ge and Ravani (1991, 1994); Srinivasan and Ge (1998), Etzel and McCarthy (1996), Juttler and Wagner (1996), Ge et al. (1998), and Xia and Ge (2001).

The focus of the present paper is on haptic and sound rendering of a virtual CNC milling process. The haptic feedback is implemented with a PHANToM haptic device by modeling the force generated from a flat end mill. Both the cutter and a human finger are modeled as circular cylinders. With a PHAMToM arm, as the cutter cylinder moves along a quaternion-based pre-computed path, a user could feel the cutting force if the finger cylinder (the curser) is kept close enough to the cutter cylinder during the motion. Haptic textures simulates and conveys the surface roughness for the groove-like machined surface through texture force. When the cutting force is turned off, the system can be used to inspect the roughness of a machined surface by letting the user feel the texture force. The haptic feedback is accelerated by applying the concept of image-based rendering. The cutting force at each cutter position are pre-computed and stored in a lookup table. This reduces the complexity of haptic rendering.

Graphical textures have been used recently for the simulation of surface roughness, to enhance the interaction between a user and a virtual world. Minsky and Lederman (1996) proposed the *lateral-force gradient algorithm* on their Sandpaper system to simulate surface roughness. They first generate graphical texture based on pre-existing height maps of small features similar to the geometry of real textured materials. The generated texture has a height $h(x, y)$ at every point (x, y), and the forces are computed in x and y directions separately by a scalar K times the height difference in x and y direction of two neighboring points. In their work, they simulated the surface roughness by only using the lateral forces. Siira and Pai (1996a, 1996b) use a stochastic approach to synthesize surface roughness for sliding objects. Fritz and Barner (1996) presented two rendering methods for haptic texturing for implementation of stochastic texture models. Their goal is to synthesize perceptually distinct textures.

In this paper, instead of dealing with rough surfaces in general, we deal with surfaces with structured roughness. In particular, we assume that the CNC milling process is a semi-rough or fine machining process. Thus the surface subject to the virtual milling process is a grooved-like surface generated by rough machining. The texture force is generated geometrically by perturbing the direction and magnitude of the normal vector along the cutting path.

Sound simulation and generation has always had an important role in the creation of immersive multimedia applications. The auditory feedback has been recently introduced into haptic environment (Ruspini and Khatib, 1998; Grabowsky and Barner, 1999). Their work relies highly on pre-recorded sound samples. At the sound modeling level, our work seeks to generate real-time cutter sound as the cutter spins and interacts with the surface to be machined. The basic sound waves are formed by using the cutter parameters including cutter speed, cutter area, spindle speed, and the cutting force. At the implementation level, we seek to implement 3D localized sound by taking advantage of the features of 3D sound card and Microsoft DirectX technology. Both haptic and sound feedbacks can be dynamically changed throughout the simulation process.

The organization of the paper is as follows. Section 2 presents an overview of haptic rendering. Section 3 deals with the force modeling issues in virtual milling process. Section 4 presents some of the implementation details about haptic rendering. Section 5 addresses modeling issues in sound rendering. Section 6 presents how sound rendering is implemented using Microsoft DirectX technology.

2 AN OVERVIEW OF HAPTIC RENDERING

Haptic rendering allows users to feel virtual objects in virtual environments. A haptic image consists of both *kinesthetic* and *tactile* information. The kinesthetic information refers to the forces which are transmitted to us when we touch an object. The tactile information refers to feeling of touch on the skin, such as spatial and temporal variations of force distributions on the skin, within the contact region with the object. A haptic device has the following functions: (1) measure the contact position and forces from the user, (2) display the contact forces and their spatial and temporal distributions to the user. There are two basic haptic modeling methods: *point-based* and *ray-based*. Modeling algorithms based on these two methods have been incorporated by Sensable Technologies GHOST software development kit, which allows users to interact with a variety of rigid objects (Salisbury and Srinivasan, 1997).

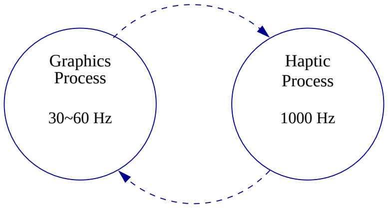


Figure 1. Haptic Rendering System

- **Point-based method** The Haptic interaction is a surface contact point (SCP), the computation of sensation of touching surfaces is based on the penetration depth of the PHANToM endpoint beneath the surface.
- **Ray-based method** The Haptic interactions with virtual objects are simulated by using ray-tracing techniques. The ray is a line segment starting from the PHANToM stylus and we take into account the ray's orientation in reflecting the forces.

A typical haptic rendering system includes two processes, one is the haptic process and the other is the graphics process. The haptic process is required to run at least 1000 Hz servo loop to generate smooth transitions and distinct sensations. The graphics process usually runs at 30 $\sim$ 60 Hz to fit human visual perception. The two processes need to seamlessly communicate to each other in order to keep the haptic and visual scene consistent.

In our research, we have built a haptic rendering system of a virtual milling process. We use textures and simple geometry to accelerate the graphics process, and use pre-computed force table for real-time lookup to reduce the workload in haptic process. Textures in haptics, like in graphics, can be used to reduce the geometric complexity of the rendered primitives. The lookup force table, inherited from the concept of image-based rendering, is used to store pre-computed forces and sent to the PHANToM device in real time. The sampling rate of pre-computed forces is perception-dependent, which is based on the sensitivity of haptic perception of user, and can be adjusted due to different users on-the-fly.

3 FORCE FEEDBACK FOR VIRTUAL MILLING PROCESS

In our implementation, we assume that a designed surface as well as a dual-quaternion representation of the cutting motions are given. We use relatively large sidestep for generating the cutter motion for rough cut. We compute the swept surface of the motion to obtain a groove-like surface. Details for this computation can be found in Xia and Ge (2001). The machined surface is displayed with textured triangles that simulate the roughness of grooves. Each textured triangle has a texture force vector stored for force feedback. The user can use the curser (in our case, the finer cylinder) of the PHANToM arm to move around the textured surface to feel the force feedback that represents the surface roughness. When the cutting force is added to the force feedback, the user can feel a combination of cutting force and the texture force when using the finger cylinder to follow the cutter motion.

3.1 Cutting Force of Flat-End Mills

There exists a substantial amount of literature on how to model the cutting force in a CNC milling process. To illustrate how the cutting force can be used for haptic rending, we select a force model for flat-end mills formulated by Abrari and Elbestawi (1997). This model is analytic and easy to implement. The general cutting force of milling process is:

$$\{F\} = [K]\{A\} \tag{1}$$

where $\{F\} = (F_x, F_y, F_z)$ is the total cutting force vector at any tool position, $\{A\} = (A_x, A_y, A_z)$ is the chip load vector at that tool position. $[K]$ is the matrix of specific pressures:

$$K = Rf \begin{bmatrix} \frac{1}{2\tan\beta}K_t & 0 & \frac{\eta r}{2\sin\beta}K_r \\ 0 & K_t & 0 \\ \frac{\eta r}{2\sin\beta}K_r & 0 & \frac{1}{2\tan\beta}K_t \end{bmatrix} \tag{2}$$

where η_r and K_t are used in the calculation of the tangential and radial components of the cutting forces including the ploughing force. The components of the load vector is given by

$$A_x = \frac{Rf}{4\tan\beta}\sum_{j=1}^{N}(\cos 2\phi_{ent} - \cos 2\phi_{ext})_j \tag{3}$$

$$A_y = Rf\sum_{j=1}^{N}(\cos\phi_{ent} - \cos\phi_{ext})_j \tag{4}$$

$$A_z = \frac{Rf}{4\tan\beta}\sum_{j=1}^{N}[2\phi_{ext} + \sin 2\phi_{ent} - (2\phi_{ent} + \sin 2\phi_{ext})]_j \tag{5}$$

where ϕ_{ent} and ϕ_{ext} are the entrance and exit angles along the cutting edge and N is the number of teeth engaged. f is feed per tooth and ϕ is tool rotational angle. β is the helix angle of flat end mill.

3.2 Texture Force

As alluded to earlier, we use texture to simulate the roughness of a surface to be machined. While haptic texturing is similar to graphics texturing, there are a few important differences. Unlike visual sense, haptic sense is a local and only needs the knowledge of the tactile features around the contact area. While in graphics only unit normal vectors are used since only directional information is needed to compute the lighting effect, in haptics, both the direction and the magnitude of a haptic force vector is used to reflect surface roughness.

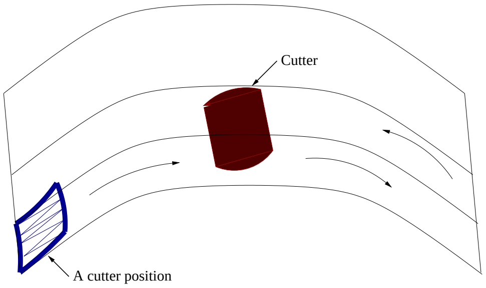


Figure 2. Tool Motion and One Cutter Position on a Surface

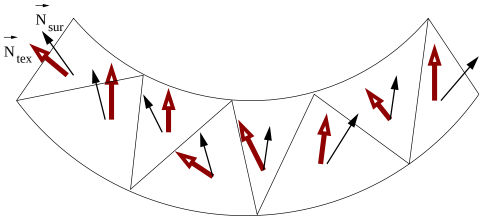


Figure 3. Haptic Texturing on A Groove

In our work, the surface to be machined or to be inspected is expressed as a triangular mesh in VRML (Virtual Reality Modeling Language) format. At each tool motion position, eight adjacent triangles are generated and displayed. A haptic force vector is then computed for each triangle by simply perturbing the direction and the magnitude of the normal vector. Figure 2 shows the tool motion on a surface as well as eight triangles are generated and displayed as portion of a small groove.

Figure 3 shows the perturbation of surface normal in both direction and magnitude. The resultant force would be $F_{result} = F_{fem} + F_{tex}$, where F_{fem} is the cutting force of a flat end mill and F_{tex} is the texture force. By Hooke's law, $F_{tex} = kN_{tex}$. The computation of F_{fem} has been discussed in previous section. A reaction force $F_{reaction} = kN_{sur}$ will replace F_{fem} for F_{result} when the user goes through the manufacturd surface after surface machining. Since the touch-mechanism of PHANToM device is point-based, we combine all eight texture normals together at each cutter position with the cutting force to represent the resultant force at the cutter position.

4 IMPLEMENTATION FOR HAPTIC RENDERING

Our haptic system is implemented on the GHOST SDK. GHOST SDK is an object-oriented 3D haptic toolkit used with SensAble Technology PHANToM haptic interfaces. It is a C++ library of objects and methods used for developing interactive, three-dimensional, touch-enabled environment. We use assumptive values for some parameters in the cutting force equations 2 3 4 5 such as the entrance and exit angles ϕ_{ent} and ϕ_{ext} for each tooth, the helix angle β and so on. Those numbers can be changed to meet the real numbers on a flat end mill. They can also be specified and read from an input file.

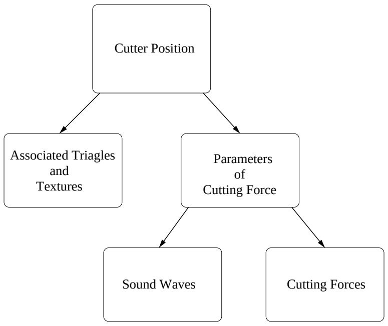


Figure 4. Cutter Position Structure

4.1 Force Feedback

GHOST SDK allows users to send forces to the PHANToM device when the PHANToM has entered the force field's bounding volume. The users can specify the actual force sent based on the data passed in via the `gstPHANToM` instance:

```
gstVector
  calculateForceFieldForce(gstPHANToM *phantom);
```

We create our own force field, which is a subclass of `gstForceField` and send the resultant force from the combination of texture force and the cutting force to PHANToM device by calling `calculateForceFieldForce()` function. `calculateForceFieldForce()` is the only functon controlling the exact force feedback which is perceived by the user in the whole haptic rendering system.

The new forcefield calss: `FEMForceField`.

```
// Flat end mill force class
class FEMForceField:public gstForceField {
public:
  FEMForceField();
  ~FEMForceField();
  gstVector
    calculateForceFieldForce(gstPHANToM *phantom);
private:
  // Pointer to an flat end mill object
  // Pointer to Texture force information
};
```

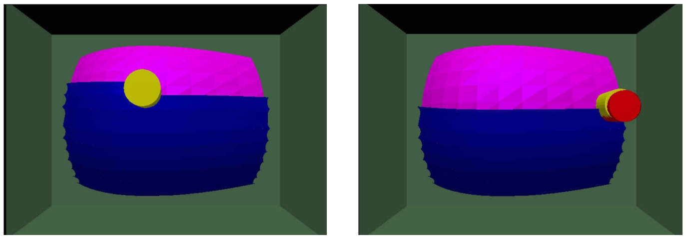
During the real tool motion, the parameters of cutting force could dynamically vary with time, this increases the complexity of computing the cutting force because the force equation requires many multiplications, divisions, and computations of sinusoidal functions. In order to reduce the complexity, we can apply the concepts of image-based rendering to the force computation. By pre- sampling those force parameters, such as the rotational position of the tool ϕ_{ent} and ϕ_{ext}, the helix angle β, and the feed rate f, in a perception- dependent manner, we can pre-compute and store the force samples in a table, and just look up the force table to get the correct force in real time.

In our system, the information of submitted cutter positions in a motion has been in discrete format and stored in an array in the order of tool motion. Each cutter position has a structure storing the parameters of the cutting force at that position and has a pointers pointing to the structure storing the information of triangles and textures for machined surface, see Figure 4. We use the parameters to compute the cutting force for that cutter position and also use them to form a sound wave, as discussed in the next section.

We compute the the cutting force F_{fem} from a flat end mill, then combine it with the texture force F_{tex}, then return the resultant force F_{result}, $F_{result} = F_{fem} + F_{tex}$, to the PHANToM device.

4.2 Parsing VRML Models

VRML is a powerful language which allows one to describe an immersive, interactive 3D world. A VRML file contains a number of objects, called nodes, which describe the world information from a static scene to a very interactive, animated scene. The VRML2.0 has a wide range of nodes which are from simple geomerty nodes to very dynamic environment nodes including the information of lighting, navigation, time-driven events and so on, see Hartman and Wernecke (1996). The VRML has been designed for being sent over networks and has become a standard file format on Internet. In order to manipulate VRML models, we have integrated a VRML2.0 parser into our haptic system. The main part which does parsing is a public-domain package (Konno, 1997). The nodes which are parsed into our system are limited to the geometry nodes for haptic-interaction environment. The `IndexedFaceSet` node is the most commonly used node in our system which is used to describe triangle-based objects.



(a) Cutter Motion (b) Two Cylinder Motion

Figure 5. Virtual Milling Machine and Tool Motions

4.3 Examples for Haptic Rendering

Figure 5(a) shows the cutter (yellow cylinder), the designed surface (pink surface) and the manufactured surface (blue surface). In Figure 5(b), if the user moves the finger cylinder (in red) closely to the cutter cylinder (in yellow), the user can feel the cutting force.

5 MODELING ISSUES IN SOUND RENDERING

In haptic environment, sound cues can increase the sense of solidity perceived by a user while interacting with an object and help to understand the nature of the haptic interaction, which may be baffling when only the visual cues are available. In our haptic environment, a cutting tool moves along a pre-defined path to cut the surface of an object. We simulate the machine sound in real time to provide the user the auditory feedback. The sound is generated based on the parameters of the cutter including the feed rate, spindle speed, cutter area and cutting force. The objective of our work is not to create physically realistic sounds, which heavily depends on the structure and composition of an object surface, but rather to simulate an acoustic cue from the major interaction happening in the haptic environment to provide an extra channel of perception.

5.1 Concepts of Frequency in Sound Signals

A simple continuous-time sinusoidal signal can be expressed as the following:

$$\begin{aligned} x_a(t) &= A\cos\,(\Omega t + \theta) \\ &= A\cos\,(2\pi F t + \theta) \end{aligned}$$

where $-\infty < t < \infty$. The signal is completely characterized by three parameters: A is the amplitude of the sinusoid, Ω is the frequency in radians per second (rad/s), and θ is the phase in radians. We can use the frequency F in cycles per second (Hz) to replace Ω, where $\Omega = 2\pi F$. Notice that $-\infty < \Omega < \infty$ and $-\infty < F < \infty$.

A discrete-time sinusoidal signal may be expressed as

$$\begin{aligned} x(n) &= A\cos\,(\omega n + \theta) \\ &= A\cos\,(2\pi f n + \theta) \qquad (6) \end{aligned}$$

where n is an integer variable, called the sample number, and $-\infty < n < \infty$. A is the amplitude of the sinusoidal, ω is the frequency in radians per sample, and θ is the phase in radians. We can use the frequency f in cycles per sample to replace ω, where $\omega = 2\pi f$. Unlike continuous-time sinusoids, the discrete-time sinusoids are characterized by three properties:

- A discrete-time sinusoid is periodic only if its frequency f is a rational number.
- Discrete-time sinusoids whose frequencies are separated by an integer multiple of 2π are identical, which is $\cos[(\omega_0 + 2\pi)n + \theta] = \cos(\omega_0 n + \theta)$. A sequence of any two sinusoids with frequencies in the range $-\pi \le \omega \le \pi$ or $-\frac{1}{2} \le f \le \frac{1}{2}$ are distinct. Discrete-time sinusoidal signals with frequencies $|\omega| \le \pi$ or $|f| \le \frac{1}{2}$ are unique.
- The highest rate of oscillation in a discrete-time sinusoid is attained when $\omega = \pi$ (or $\omega = -\pi$) or, equivalently, $f = \frac{1}{2}$ (or $f = -\frac{1}{2}$).

5.2 Sampling of Analog Signals

There are many ways to sample an analog signal. We are only discussing *periodic* or *uniform sampling*, which is the type of sampling used most often in practice. Consider

$$x(n) = x_a(nT), \qquad -\infty < n < \infty \qquad (7)$$

where $x(n)$ is the discrete-time signal obtained by taking samples of the analog signal $x_a(t)$ every T seconds. The $1/T = F_s$ is the *sampling rate* (samples per second) or the *sampling frequency* (Hz). Uniform sampling establishes a relationship between the time variables t and n of continuous-time and discrete-time signals, respectively. With the sampling rate $F_s = 1/T$, we have

$$t = nT = \frac{n}{F_s} \qquad (8)$$

With equation 8, we can establish a relationship between the frequency variable F (or Ω) for analog signals and the frequency variable f (or ω) for discrete- time signals. Consider an analog sinusoidal signal of the form

$$x_a(t) = A\cos\,(2\pi F t + \theta), \qquad -\infty < t < \infty \qquad (9)$$

where A is amplitude of the sinusoid, F is frequency in cycles per second (Hz). When we do sampling periodically at a rate $F_s = 1/T$ samples per second, we will have

$$\begin{aligned} x(n) = X_a(nT) &= A\cos\,(2\pi FnT + \theta) \\ &= A\cos\,(\frac{2\pi nF}{F_s} + \theta) \end{aligned} \tag{10}$$

Comparing equation (10) with (6), we have

$$f = \frac{F}{F_s}, \quad \omega = \Omega T. \tag{11}$$

Recall that the range of the frequency variable F or Ω for continuous- time sinusoids and discrete-time sinusoids are

$$-\infty < F < \infty, \quad -\infty < \Omega < \infty \tag{12}$$

and

$$-\frac{1}{2} \leq f \leq \frac{1}{2}, \quad -\pi \leq \omega \leq \pi. \tag{13}$$

Comparing equations 11, 12, and 13, we have

$$-\frac{1}{2T} = -\frac{F_s}{2} \leq F \leq \frac{F_s}{2} = \frac{1}{2T} \tag{14}$$

$$-\frac{\pi}{T} = -\pi F_s \leq \Omega \leq \pi F_s = \frac{\pi}{T} \tag{15}$$

Equations (10), (11), and (14) will be directly used in our implementation.

5.3 Sound Waves of a Cutter Motion

We model the sound waves of a cutter motion as a procedural sound using the feed rate of cutter, spindle speed of cutter, the cutter area and the cutting force. In general, an analog signal for tool motion sound can be represented as a sum of sinusoids with various amplitudes, frequencies, and phases, that is,

$$x_a(t) = \sum_{i=1}^{N} A_i \sin(2\pi F_i t + \theta_i) \tag{16}$$

where N denotes the number of frequency components. In our implementation, we use the analog signals of the following form:

$$x_a(t) = A_1 \sin(2\pi F_1 t + \theta_1) + A_2 \sin(2\pi F_2 t + \theta_2) \tag{17}$$

where A_1 is the cutting force, F_1 is the spindle speed. Since the intensity is proportational to impulse (Takala and Hahn, 1992), A_2 can be formed by the feed rate × cutter area × a scalar, and F_2 is simply 0 (forming an aperiodic signal). The machine sound waveform will be dynamically changed with respect to the different spindle speed, feed rate of cutter and the cutting force as time goes.

6 DIRECTSOUND BASED IMPLEMENTATION

We use Microsoft DirectX technology to implement the sound models of the virtual milling process and take advantage of the features of 3D sound card to achieve the rolloff, arrival offset, muffling, and Doppler shift effect in our haptic environment. DirectX provides a finely tuned set of application programming interfaces (APIs) including DirectDraw, Direct3D, DirectSound, DirectPlay, DirectInput and DirectSetup. It provides windows-based applications with high-performance low-level and real-time access to available multimedia hardware on a computer system in a device-independent manner. DirectSound is the audio component of DirectX. It enables hardware and software sound mixing, capture, and 3D positional effects.

6.1 DirectSound Architecture

DirectSound has a hardware abstraction layer (HAL), and a hardware emulation layer (HEL). The HAL is a software driver provided by the sound-card vendor and processes requests from the DirectSound Object. The HAL processes a DirectSound object request from the sound hardware, and reports the cababilities of the hardware. If there is no DirectSound driver installed or the sound hardware does not support a requested operation, DirectSound will try to emulate the functionality via the HEL, see Figure 6.

6.2 DirectSound Buffers

The Windows application places a set of sounds in buffers, called secondary buffers, which are created by the application. DirectSound combines (mixes) these sounds and writes them into a primary buffer, which holds the sound that the listener actually hears. DirectSound automatically creates a primary buffer, which typically resides in memory on a sound card. The application creates the

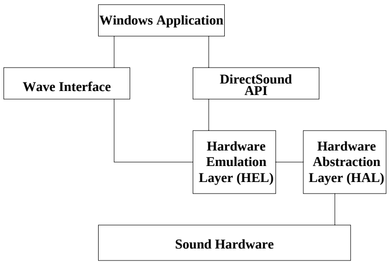


Figure 6. DirectSound Architecture

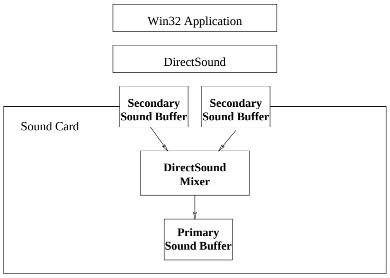


Figure 7. DirectSound Sound Buffering Process

secondary buffers either in system memory or directly on the sound card, see Figure 7.

Depending on the type of sound card, DirectSound buffers can exist in hardware as on-board RAM, wave-table memory, a direct memory access (DMA) channel, or a virtual buffer (for an input/output [I/O] port-based audio card). The sound buffers are emulated in system memory if there is no hardware memory available. Only the available processing time limits the number of buffers that DirectSound can mix, and an application can query a sound buffer to determine what percentage of main processing cycles are needed to mix the sound buffers. The DirectSound mixer can provide as little as 20 milliseconds of latency, in which are there is no perceptible delay before play begins. If DirectSound must emulate hardware features in software, the mixer can not achieve low latency and a longer delay (usually about $100 - 150$ milliseconds) occurs before the mixed sound is played.

6.3 3D Sound

DirectSound can apply effects to a sound as it is written from a secondary buffer into the primary buffer. Basic effects are volume, frequency control and panning (changing the relative volume between the left and right audio channels). DierctSound can aslo simulate 3D positional effects through the following techniques:

- **Rolloff** The further an object is from the listener, the quieter it sounds. This phenomenon is known as rolloff. The sound intensity decays proportionally to the square of the distance increased between sound source and the listener.
- **Arrival Offset** A sound emitted by a source to the listener's right will arrive at the right ear slightly before it arrives at the left ear. The duration of this offset is approximately a millisecond.
- **Muffling** The orientation of the ears ensures that sounds coming from behind the listener are slightly muffled compared with sounds coming from front. If a sound source is at the right, the sounds reaching the left ear will be muffled by the mass of the listener's head as well as by the orientation of the left ear.
- **Doppler Shift Effect** DirectSound can create Doppler shift effects for any buffer or listener that has a velocity. If both sound source and listener are moving, DirectSound automatically calculates the relationship between their velocities and adjusts the Doppler effect accordingly. The sound delay is proportional to the ratio of distance increased divided by the velocity of the sound source.

6.4 Implementation of Sound Rendering

DirectSound takes the Pulse Code Modulate (PCM) waveforms. The waveform has a `WAVEFORMATEX` structure which specifies the characteristics of the wave.

```
typedef struct {
    WORD  wFormatTag;
    WORD  nChannels;
    DWORD nSamplesPerSec;
    DWORD nAveBytesPerSec;
    WORD  nBlockAlign;
    WORD  wBitsPerSample;
    WORD  cbSize;
} WAVEFORMATEX;
```

In the `WAVEFORMATEX` structure, the `nChannels` describes mono or stereo sound. `nSamplesPerSec` is the sampling rate (Hz). `nAvgBytesPerSec` is the average data-transfer rate (bytes/sec). `nBlockAlign` describes the minimum atomic unit of data for `nFormatTag` format type. In our implementation, we are synthesizing machine sound on-the-fly and sending it to DirectSound buffers. The `nSamlpesPerSec` is the F_s in equation (10). In order to send the correct discrete frequency f into sound buffer, we need to divide the frequency of our sound model F by F_s, see equation (11). We create two secondary sound buffers to mix two sound waves, the first one is modeled with the spindle speed and the force of cutter, the second one is modeled with the feed rate and the cutter area. The two buffers

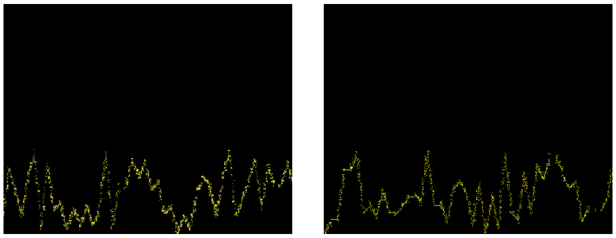


Figure 8. Two sound waves with different frequencies and amplitude

are mixed in the DirectSound primary buffers and sent to the speakers.

Figure 8 shows two sound waves sampled by our procedural sound equation with different frequency and amplitude. The sound sounds sharper when the frequecy raises, and the sound sounds heavier as the amplitude become larger. Spatial (3D) sound effects are achieved by using DirectSound 3D buffers with the setting of several parameters of sound source and listener. In our work, we put the listener at the center of the surface. With a suitable setting of distance factor and correct updating of the position of the sound source (cutter), the user can feel the Rolloff effect very clearly and easily. The addition of 3D effects slows down the rendering speed because of the compuatation of the 3D sound. In order to improve the quality and reality, some other physically-based parameters such as the material properties of the surface and geometric structures of cutter can be added into the formulation of sound. These new parameters form new sound waves and can be mixed into sound buffer for play. By applying the concepts of image-based rendering, we can uniformly sample the speed and distance from sound source and listener, pre-compute Doppler shift factors and Rolloff factors and store them in a lookup table. These numbers can be directly used to generate Doppler shift and Rolloff effect during run time. This can reduce the complexity of sound computation. Mixing the simulation sound with the pre-recorded real sound can also make the sound more real.

7 CONCLUSIONS

In this paper, we have applied virtual reality technology to the development of a multisensory virtual environment for simulating the CNC milling process. A model of the cutting force of a milling process has been used to provide the kinesthetic information for the force feedback. The geometric roughness of a surface is used to provide the tactile information for the haptic texturing. The milling parameters and the cutting force model have also been used to create the sound model for the virtual milling process. The concepts of image-based rendering are utilized to accelerate both haptic rendering and sound rendering by pre-sampling related environmental parameters in an perception-dependent way.

ACKNOWLEDGEMENT

This work has been supported by NSF grant DMI-9800690 to the State University of New York at Stony Brook.

REFERENCES

F. Abrari and M. A. Elbestawi, 1997. Closed form formulation of cutting forces for ball and flat end mills. In *International Journal of Machine Tools & Manufacture*, volume 37(1), pages 1727.

O. Bottema, and Roth, B., 1990. *Theoretical Kinematics*. Reprint by Dover Publ, New York.

K. Etzel, and McCarthy, J. M., 1996, Spatial motion interpolation in an image space of SO(4). In *Proceedings of the 1996 ASME Design Technical Conference*, Paper Number 96-DETC/MECH-1164.

J. Fritz and K. Barner, 1996. Stochastic models for haptic texture. In *Proceedings of the SPIE International Symposium on Intelligent Systems and Advanced Manufacturing*, Telemanipulator and Telepresence Technologies III.

Q.J. Ge, and Ravani, B., 1991, Computer aided geometric design of motion interpolants". *Proc. of 17th ASME Des. Auto. Conf.*, p 33-41, Miami, FL. See also *ASME J. of Mechanical Design*, 116(3):756-762.

Q.J. Ge, and Ravani, B., 1994, Geometric construction of Bézier motions, *ASME Journal of Mechanical Design*, 116(3):756-762.

Q. J. Ge, A. Varshney, J. Menon, C. Chang, 1998, Double quaternion for motion interpolation, *Proc. 1998 ASME Design Manufacturing Conference*, Atlanta, GA, Paper No. DETC98/DFM-5755.

N. A. Grabowski and K. E. Barner, 1999, Structurally-derived sounds in a haptic rendering system. In *Proceedings of the Fourth PHANToM Users Group Workshop*.

J. Hartman and J. Wernecke, 1996. *The VRML 2.0 Handbook*. Addison-Wesley Publishing Company.

B. Jüttler, and M.G. Wagner, 1996. Computer-aided design with spatial rational B-spline motions, *ASME J. of Mechanical Design*, 118(2):193-201.

S. Konno, 1997, http://www.cyber.koganei.Tokyo.jp/vrml/cv97.

J. M. McCarthy, 1990, *Introduction to Theoretical Kinematics*, MIT.

M. Minsky and S. J. Lederman, 1996. Simulated haptic textures: Roughness. In *Proceedings of the ASME Dy-*

namics Systems and Control Division, volume 58, pages 421–426.

D. Ruspini and O. Khatib, 1998, Acoustic cues for haptic rendering systems. In *Proceedings of the Third PHANToM Users Group Workshop.*

J. K. Salisbury and M. A. Srinivasan, 1997, Phantom-based haptic interaction with virtual objects. In L. J. Rosenblum and M. R. Macedonia, editors, *Projects in VR*, pages 6–10. IEEE Computer Graphics and Applications.

J. O. Siira and D. K. Pai, 1996a, Fast haptic textures. In *ACM Conference on Human Factors in Computer System*, pages 231–232.

J. O. Siira and D. K. Pai, 1996b Haptic texturing - a stochastic approach. In *Proceedings of the 1996 IEEE International Conferences on Robotics and Automation*, pages 231–232.

L. Srinivasan, and Ge, Q. J., 1998, Fine tuning of rational B-spline motions, *ASME Journal of Mechanical Design*, 120(1):46-51.

T. Takala and J. Hahn. Sound rendering. In *Proceedings SIGGRAPH 92*, volume 26(2), pages 211–220. ACM Computer Graphics, July 1992.

J. Xia, and Ge, Q. J., 2001, An exact representation of effective cutting shapes of 5-axis CNC machining using rational Bezier and B-spline tool motion, *IEEE International Conference on Robotics and Automation*, Seoul, Korea.