The Haptic SpaceMouse – an input device with force-feedback through solenoids

Andreas C. Bernstein, Robert Lenhardt
Dipl. Ing. Jan Hochstrate, Prof. Dr. Bernd Fröhlich
Bauhaus-Universität Weimar
Bauhausstrasse 11, 99423 Weimar, Germany
{andreas.bernstein,robert.lenhardt,jan.hochstrate,bernd.froehlich}@medien.uni-weimar.de

Abstract

We describe the Haptic SpaceMouse, an input device based on the SpaceMouse[1] with 2 DOF translational feedback in contrast to the rotational feedback often found in force-feedback joysticks. We use solenoids as motors in our implementation, they facilitate frictionless operation and add little inertia to the Spacemouse. The force-feedback controller is an Atmel ATmega32[6] microcontroller, which is programmed to decode the device output, thus enabling us to drive the solenoids in a closed loop. A simple host-to-device protocol is developed to specify the haptic forces. Our application-scenario demonstrates rigid-body contact of an user-controlled cursor.

1. Introduction

Conventional haptic input devices like force-feedback joysticks provide rotational feedback, which is problematic in two ways. First, the produced forces are small, because of the lever action of the stick the user manipulates, which reduce the forces of the motors. Conversely, in applications with velocity-control, the forces to bring the joystick into full stop – the center position – have to be quite high. Second, the haptic manipulation of the joystick has to move it over a relatively large distance, because of the large workspace of the stick.

Fortunately the human hand can sense forces to a very fine degree, without needing large travel, so it is not necessary to provide a big workspace with a device driving large forces. The SpaceMouse is an example of a device for the hand providing 6 DOF on a small workspace (3 mm^3). In order to provide haptic feedback for this device one is able to use linear motors such as solenoids or voice-coil motors to transmit force directly, without special transmission that causes unnecessary friction. Furthermore it is possible to change forces quickly, because no big masses that add inertia are involved.

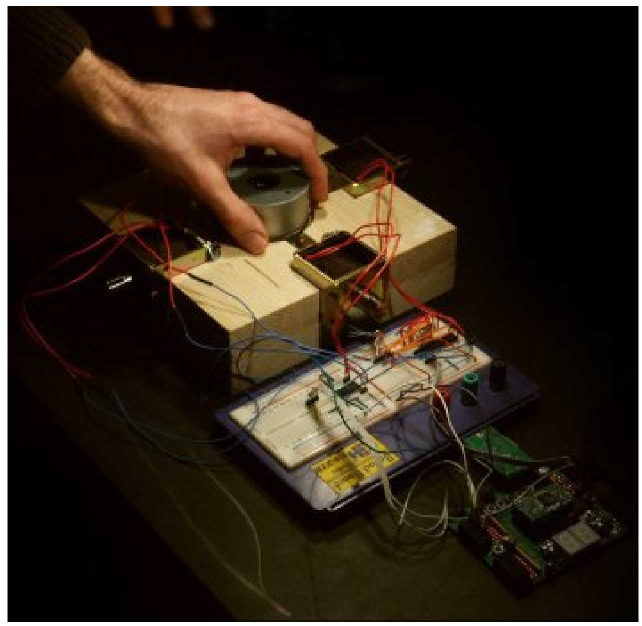

Figure 1. The Haptic SpaceMouse

We have developed the Haptic SpaceMouse device, which adds force feedback to the SpaceMouse. Our construction provides 2DOF translational feedback to the SpaceMouse. We implemented the force feedback by mounting four solenoids coaxial on two translational axes of the SpaceMouse.

To take advantage of this drive line system, a closed loop control is implemented that ensures a quick response time without large latencies. The sensor in this loop is the SpaceMouse, so we transfer its output directly to the controller and use it to drive the motors. In applications which simulate contact with walls or other rigid bodies one just needs to provide the direction of the force and its magnitude in a local area of contact and the controller is able to transform this information into the device coordinates and drive the motors accordingly. We have implemented this application scenario, where a virtual cursor is manipulated by the user and contact with rigid walls are simulated haptically.

2. Details of the Haptic SpaceMouse

2.1. Device Assembly

The SpaceMouse is a 6dof elastic input device which is manipulated using only small travel (1-2 mm) from an absolute center point. For the sake of simplicity only translations of the device are discussed. The user applies a force to the device which leads to a travel away from the center point. The extent of this travel is proportional to an uniform velocity of the manipulated object – so called rate-control – with a velocity of zero at the center point. The device coordinate system is tied to the object.

In order to render haptic effects such as a hit of a rigid object with a wall the device needs to be put in its center point i.e. the negative force needs to be applied. This can be achieved with linear motors such as solenoids, which are mounted on the axes of the SpaceMouse.

An electric solenoid can create forces on one axis in one direction. It is an iron anchor in a coil. If current flows trough the coil, the anchor is pulled into the coil. The force of a solenoid is a function of the applied current and the position of the anchor resp. the area of the gap between the anchor and the end stop. Unfortunately it is no linear function of the current, but grows quadratic as the anchor moves to its resting position. The equation for the force is dependent of the actual solenoid but can be described as an inverse square law[3]:

$$F(s) \sim \left(\frac{rNI}{s}\right)^2$$

where s is the stroke or size of the air gap between the end and the anchor, r is the radius of the anchor, N is the amount of windings of the coil and I is the current. The generated force can be quite large, especially near the end stop, but the nonlinearity turns up in this region too. If we use a long anchor and only a small area located away from the end stop we can assume a linear force-current relation with forces up to 5 N.

A voice-coil motor does not have the problem of nonlinearity – its generated force is proportional to the applied current – and can generate force in both directions, but a motor which can deliver forces of approx. 3 N was deemed too expensive. Another possibility to minimize the nonlinearity is to insert a return spring into the gap which when carefully chosen, minimizes the position dependency. Ready made solenoids, so called proportional solenoids are available, but with costs comparable to voice-coil motors.

The four solenoids (two for each axis) are coupled trough a U-shaped plate per two solenoids so that they cannot clamp the SpaceMouse. The plate is led through under the SpaceMouse and encloses its axis. They are driven from PWM signals from the Atmel which are amplified trough a ULN2064[8] Quad Darlington Switch. The PWM signals are created using hardware timers. The generation of 4 separate PWM signals from software using one timer interrupt was implemented, but the computational cost was too expensive and slowed down the other communication tasks.

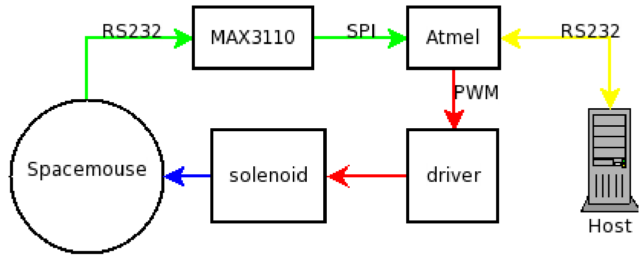

Figure 2. The closed loop data flow from SpaceMouse to solenoid

2.2. Interface between SpaceMouse and Atmel

The Atmel AVR ATmega32 is an 8bit micro-controller with Harvard architecture and 32kb program- and 2kb system memory. It runs at 16MHz, has several timers to generate PWM signals up to 4MHz. Interfaces such as UART, I^2C and SPI are available. One problem is that there is only one UART interface where we need two – one for receiving the SpaceMouse data and one for communicating with the host. Therefore we used an MAX3110[7] UART transceiver with SPI interface to connect the Atmel with the SpaceMouse. This enables us to decode the position right on the Atmel, using the SpaceMouse as sensor in a closed loop. The data is relayed to the host application on the Atmel UART. Unfortunately the SpaceMouse sends its data at a rate of 9600 bps, bounding the bandwidth of the whole loop to $\sim$100Hz. Newer SpaceMouse models using the USB protocol are able to transfer data much faster, so it is not a principal boundary, but handling USB communication would require a whole new design with another micro-controller.

The Atmel is programmed using the GNU Compiler Collection[4] for AVR targets in C. An implementation of the C standard library[2] is available. A UART interface using interrupt routines for the Atmel hardware UART and the MAX3110 SPI interface, a closed loop control driving the solenoids through hardware generated PWM signals and the host-device protocol handling are implemented.

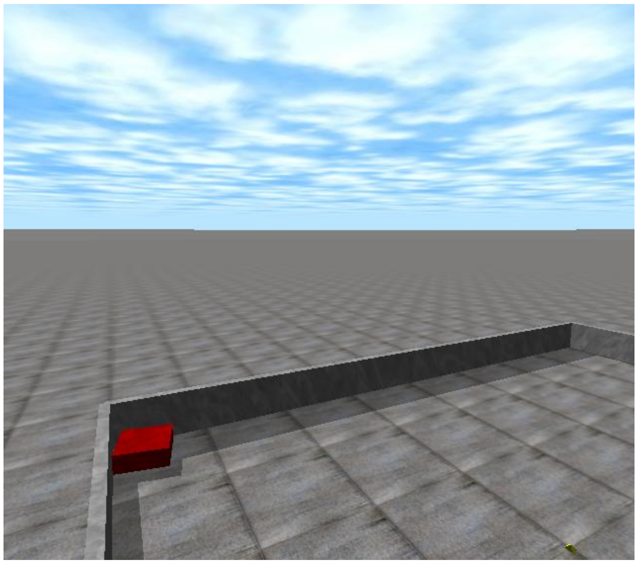

Figure 3. Demo Application

3. Application

To test our device, especially the ability to simulate contact with rigid walls, we build a simple 3D demo application. The cursor – a small red box – can freely be moved around in a fenced area in two dimensions. The area is fenced by four walls which are rigid bodies. If the box touches a wall a message is sent to the SpaceMouse over the serial port which consists of the normal at the contact joint. Thereupon the solenoids are driven to provide a counterforce and the closed loop readjusts the magnitude of the force. We use the Open Dynamics Engine (ODE)[5], an open source physics engine for simulating rigid body dynamics, the collision detection and its supplied drawstuff library for visual rendering.

The advantages of our device are that the force feedback feels natural for little forces and the solenoids showed no perceptible exponential behavior for the used travel.

When moving the box into a corner, the SpaceMouse started to jerk. If the user applied force was too strong, the solenoids were no longer able to resist und could be pushed through.

4. Conclusion and Future Work

We have developed a 2 DOF translational input device, in contrast to conventional 2 DOF rotational feedback found in joysticks. The device can provide forces up to 5 N. The solenoids used as motors are frictionless and add low inertia to the input system. We are able to decode the Sacemouse output directly on our feedback controller and use this output as feedback for a closed loop control. A simple host-device protocol based on local contact information is sufficient to render haptic effects such as rigid bodies.

References

[1] 3D Connexion. `http://www.3dconnexion.com/`.

[2] AVR Libc. `http://www.nongnu.org/avr-libc/`.

[3] G2 Consulting Course Notes. `http://www.consult-g2.com/`.

[4] GNU Compiler Collection. `http://gcc.gnu.org`.

[5] Open Dynamics Engine. `http://www.ode.org/`.

[6] Atmel Cooperation, 2325 Orchard Parkway, San Jose, CA 95131. *8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*, 2004. `http://atmel.com/dyn/resources/prod_documents/doc2503.pdf`.

[7] MAXIM Integrated Products, 120 San Gabriel Drive, Sunnydale, CA. *MAXIM SPI/MICROWIRE-Compatible UART and ±15kV ESD-Protected RS232 Transceivers with Internal Capacitors*, 1999. `http://pdfserv.maxim-ic.com/en/ds/MAX3110E-MAX3111E.pdf`.

[8] ST Microelectronics. *50V - 1.5A Quad Darlington Switches*, 2003. `http://www.st.com/stonline/books/pdf/docs/1534.pdf`.