Multiplicación de Matrices por medio de Procesadores de Señales Digitales* (DSP's)

Ing. Abraham Ceballos Zamora
Ing. Mauricio Olguín Carbajal
Ing. Guillermo Pitalua Osorio
Alumnos de la Maestría del CINTEC-IPN.
M. en C. Héctor Samuel García Salas
Profesor e Investigador del CINTEC-IPN.

Como parte de un proyecto para el desarrollo de un laboratorio de realidad virtual en el CINTEC, esta investigación trata sobre el manejo de estructuras de datos utilizando procesadores especializados, como son los procesadores de señales digitales ("Digital Signal Procesor", DSP), TI320C20 y TI320C50 de "Texas Instruments". El objetivo de este trabajo es el desarrollar las herramientas de software necesarias para la manipulación de objetos 3D, los cuales forman parte de un escenario o mundo virtual.

Por otro lado, la "renderización" de los escenarios, es decir, el cálculo de las imágenes correspondientes al mundo virtual en tiempo real, es otra de las tareas a las cuales esperamos aplicar los algoritmos que estamos desarrollando. En la "renderización" de una imagen, debemos tener en cuenta que el conjunto de bits almacenados en la memoria vídeo representan, de acuerdo a su valor, puntos en la pantalla (pixeles) con diferentes características (color e intensidad).

El cálculo del valor de cada pixel depende de su posición en la pantalla, es decir, de su posición en la memoria vídeo, ya que el arreglo de palabras en esta memoria mapea la imagen traducida del mundo virtual. Esto es, cada pixel tendrá un valor calculado representando la luminosidad, color y textura correspondiente al punto del escenario virtual.

Ahora bien, las imágenes renderizadas son desplegadas generalmente en un monitor de computadora. Sin embargo, si se quiere tener una sensación tridimensional es necesario crear imágenes estereoscópicas, que sean mostradas por medio de cascos de despliegue estereoscópico. Para esto, es necesario llevar a cabo el cálculo de las imágenes del mundo o escenario virtual, correspondientes a cada ojo con el ángulo adecuado.

La simulación del proceso de movimiento requiere de un compromiso entre la resolución de las imágenes, el número de colores, la gama de intensidades y el número de imágenes por segundo. Para que el ojo humano tenga la percepción de una continuidad, se requiere presentar secuencialmente por lo menos 24 imágenes por segundo. Sin embargo, si la resolución es muy alta el tiempo de cálculo de la imagen es muy grande, lo que incide en el tratamiento de imágenes en tiempo real.

A pesar de los avances en las investigaciones realizadas en el campo del proceso de la visión humana, este es tan complejo que hasta la fecha no se ha comprendido íntegramente. Sin embargo, hay hechos que nos permiten llegar a ciertas conjeturas:

Una unidad masiva de conmutación contiene hasta 25 millones de transistores; el cerebro por su parte contiene entre 10^{11} y 10^{12} neuronas. Sin embargo, una neurona es mucho mas lenta al conmutar (entre uno y diez milisegundos) que un transistor (aproximadamente 10 nanosegundos), de hecho es casi un millón de veces más lenta. A pesar de esto, el cerebro es capaz de procesar imágenes en tiempo real, mientras que un microprocesador común y corriente (por ejemplo un pentium) requiere de 3.8 segundos para un proceso sencillo de una imagen. Lo anterior es para muchos expertos una prueba irrefutable de que el proceso de la visión es masivamente paralelo.

En nuestra investigación, la máquina que se propone para llevar cabo operaciones lineales sobre matrices, es una PC (computadora personal), con procesador pentium como anfitrión para dos tarjetas. Cada una de estas tarjetas integran dos DSP's (TMS320C50), y se encargan a su vez de efectuar las transformaciones sobre el conjunto de primitivas de cada objeto y, a continuación, de obtener la "renderización" de las imágenes de salida, con un ritmo de generación de al menos 24 imágenes por segundo para cada ojo.

Como es necesario obtener ángulos diferentes del mismo entorno,

las tarjetas llevarán a cabo estas operaciones respectivamente para uno y otro ojo.

Desarrollo

Dada la cantidad de operaciones que se tienen que realizar, el utilizar una forma eficiente para hacerlo permitirá reducir el tiempo de operación. Como se mencionó anteriormente, cada objeto se encuentra representado en memoria por medio de sus primitivas. Estas son:

- vértices,
- centro de gravedad,
- conjunto de vértices que definen sus caras,
- normales de cada uno de sus lados y
- textura y color de las superficies.

Esta información puede ser almacenada en forma de matrices. Las operaciones básicas que se realizan con los objetos 3D son: escalamiento, translación y rotación. Estas operaciones se representan por medio de transformaciones lineales; por lo tanto, la modificación del escenario virtual puede obtenerse una vez determinadas las transformaciones correspondientes, del resultado de la aplicación de éstas sobre el conjunto de matrices que representan a los objetos. Esta información, debidamente interpretada, producirá una imagen "renderizada" que se despliega en el dispositivo correspondiente (monitor, casco estereoscópico, etc.).

Ya que la aplicación de las transformaciones sobre el conjunto de objetos 3D corresponde a un producto matricial, uno de los puntos que estimamos importante es el utilizar un algoritmo basado en instrucciones especializadas de un procesador que realice multiplicaciones en un tiempo mínimo.

La mayoría de los microprocesadores necesita de varios ciclos de máquina para completar una operación de multiplicación. Sin embargo, un DSP puede realizar una multiplicación en un solo ciclo de máquina. Esto se debe a que su arquitectura interna tiene un multiplicador implementado en hardware, lo cual le permite realizar una multiplicación sin necesidad de cambiar datos entre registros, internos y/o externos.

Por otro lado, una matriz que representa un punto en un espacio tridimensional tiene la siguiente forma:

[X Y Z 1]

El cuarto componente de esta matriz se utiliza para que la traslación pueda manejarse como un operador lineal. Por ejemplo, si se desea hacer el escalamiento (modificación del volumen) de un objeto, se multiplican las primitivas respectivas (vértices) por la matriz de escalamiento de la forma:

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Utilizando el producto de matrices (cualquier par de matrices puede ser multiplicado mientras se cumpla la regla [m x n] x [n x p] donde m, n y p son los tamaños de renglones y columnas respectivamente), el procedimiento para la multiplicación se puede resumir en una sola sumatoria descrita por:

$$C_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj}$$
$$i = 1, m$$
$$j = 1, p$$

donde ***a*** y ***b*** son, respectivamente, elementos pertenecientes a dos matrices diferentes A y B, que cumplen con la regla anterior.

El DSP cuenta con instrucciones que permiten además de multiplicar como ya se mencionó arriba, el cargar y almacenar datos en un solo ciclo. Las matrices deben ordenarse de manera que el DSP pueda tomarlas fácilmente de memoria y almacenarlas nuevamente en ella.

El almacenamiento de las matrices en memoria se realizó de la siguiente forma:

- El primer renglón de la matriz A se almacena en n localidades contiguas de nuestra memoria de datos, a continuación el siguiente renglón y así sucesivamente hasta completar la matriz.
- En el caso de la matriz B, se toma la primera columna y se almacena inmediatamente después de la matriz A, en n localidades de la memoria de datos, a continuación, la siguiente, y así progresivamente hasta llegar al final. (Considérese que la matriz A corresponde a un punto y que la matriz B corresponde a la transformación).

Es importante notar que el almacenamiento de las variables que hemos utilizado, está estrechamente vinculado con el algoritmo desarrollado para realizar la multiplicación.

Uno de los valores iniciales que recibe el algoritmo es el tamaño de las matrices a utilizar, ya que es necesario saber donde finaliza cada renglón de la matriz A y donde comienza el siguiente. Por otro lado, estos valores nos permiten determinar la localización en memoria de la matriz B, así como la extensión de sus columnas. Esto es importante, ya

que los datos son tomados de manera secuencial.

Algoritmo

- **Primer paso:** se inicializan los contadores de fila de la matriz A y de columna de la matriz B (estos valores se van a traducir en la posición de comienzo de los registros, donde se encuentran almacenados los elementos de las matrices A y B), se inicializa el contador común de columnas de A y de filas de B. Se pone a ceros el registro de almacenamiento.

- **Segundo paso:** se toma el elemento correspondiente de la matriz A, apuntado por el contador de filas de la matriz A y el contador común, y se toma el elemento correspondiente de la matriz B, apuntado por el contador común y el contador de columnas de la matriz B.

- **Tercer paso:** se efectúa el producto de estos dos elementos y el resultado se suma al registro de almacenamiento.

- **Cuarto paso:** se incrementa el contador común y se repiten los pasos dos, tres y cuatro hasta que el contador común llegue al valor del tamaño correspondiente al número de columnas de la matriz A igual al número de filas de la matriz B.

- **Quinto paso:** el valor del registro de almacenamiento se envía a la localidad de memoria apuntada por el contador de filas de la matriz A y el contador de columnas de la matriz B.

- **Sexto paso:** se incrementa el contador de columna de la matriz B y se repiten los pasos segundo al sexto hasta que este contador llegue al valor del tamaño de columnas de la matriz B.

- **Séptimo paso:** se incrementa el contador de filas de la matriz A y se repiten los pasos del segundo al séptimo hasta que este contador llegue al valor del tamaño de filas de la matriz A.

La matriz resultante C (de tamaño m x p) se almacena de la misma manera que la matriz A, es decir, una fila detrás de otra. Esta matriz C es, para fines de nuestra investigación, la correspondiente a la transformación de un punto de uno de los objetos del mundo virtual o, si se trata de llevar a cabo una renderización, el valor del pixel correspondiente a un punto en la pantalla.

Ahora bien, este conjunto de operaciones que se pueden realizar de manera sencilla en un procesador normal, consume demasiado tiempo en corrimientos y movimientos entre registros, pero en el DSP se puede utilizar una instrucción para multiplicar y acumular al mismo tiempo. Dicha instrucción es la llamada MAC, que es propia del lenguaje ensamblador de los procesadores DSP TI320CX0. Esta instrucción permite tomar datos de la memoria de programa y multiplicarlos por un dato proveniente de la memoria de datos. El producto obtenido es sumado al acumulador, que puede estar en cero o contener productos anteriores.

Como puede verse, esto es precisamente lo que se requiere para una multiplicación de matrices. La multiplicación se puede repetir ejecutando direccionamientos indirectos y con una instrucción de repetición de bloque de código. Una vez obtenido el primer elemento de la matriz C resultante, deberá guardarse en memoria de la manera ya descrita.

Este proceso debe repetirse tantas veces como elementos tenga un fila. El proceso descrito es fácilmente expansible a matrices de n x n elementos, cuidando siempre cumplir con los requisitos de multiplicación de matrices ya comentados anteriormente.

Con objeto de conseguir un entendimiento mas claro de la instrucción MAC, se ilustrará su uso con un ejemplo:

```
MAC  0FF00h, *+
```

Antes de ejecutar la instrucción tenemos que los registros que intervienen se encuentran en el siguiente estado:

```
ARP  2
AR2  102h
Memoria datos 102h 13h
Memoria prog. FF00h  7h
TREG 0   35h
P 0367h
ACC  02235h
```

Después de ejecutada la instrucción, el estado de estos mismos registros será el siguiente:

```
ARP  2
AR2  103h
Memoria datos 102h 13h
Memoria prog. FF00h  7h
TREG 0   13h
P 085h
ACC  022BAh
```

Comparaciones

Un programa de multiplicación de matrices, con el algoritmo expuesto anteriormente, ejecutándose en un DSP TMS320C25 tomó 4.2 microsegundos (ms) en multiplicar una matriz A de 2 x 2 por una matriz B de 2 x 3 y almacenar la matriz C resultante de 2 x 3. En contraste, una PC con procesador Intel Pentium a 100 MegaHertz (MHz) tomó 110 ms en realizar la misma operación.

TMS320c25 @ 40 MHz	PC Intel Pentium @100 MHz
4.2 ms*	110 ms*

* Los tiempos referidos son el resultado de promediar 10 ejecuciones distintas del mismo algoritmo en cada procesador.

Metodo de Medición de los Tiempos de Ejecución en la Multiplicación de Matrices.

Para medir el tiempo que le llevó al DSP TMS320C26 ejecutar la multiplicación de las matrices solamente fue necesario revisar el registro de timer (temporizador) del propio circuito (Kit de desarrollo TMS320C26 de Texas Instrument), es posible tener acceso a el mediante el debugger que se provee con el "starter kit". Dicho registro empieza con el valor FFFFh y con cada ciclo de máquina que pasa este contador se decrementa; de este modo, si pasan 7 ciclos de máquina el contador nos muestra FFF8h. Por lo tanto, para medir el tiempo de ejecución de el programa de multiplicación de matrices se observa el contador, una vez que ha finalizado el programa, a continuacion se resta a FFFFh la cantidad que presente el contador y obtenemos un valor en hexadecimal, este valor se convierte a decimal y se multiplica por el tiempo que dura cada ciclo de reloj (para este caso es de 100 nanosegundos).

En el caso del procesador Pentium de Intel usamos el paquete de programación "MatLab", con el cual realizamos la multiplicación de las dos matrices. Este paquete nos permite el acceso a una instrucción que mide el tiempo que se toma el procesador en ejecutar una serie de operaciones (en este caso la multiplicación de dos matrices) y con la cual se obtuvo el tiempo de ejecución de la multiplicación.

Conclusiones

De acuerdo a lo anterior, el llevar a cabo un producto de matrices utilizando un procesador basado en una CPU de propósito general, por ejemplo el Pentium, toma bastante más tiempo de procesamiento que el requerido por un DSP de segunda generación (que la realiza aproximadamente 25 veces mas rápido). Los DSP de quinta generación (TMS320 C50), que son los que se piensa usar en la continuación de este proyecto, poseen mejoras en su arquitectura que le permiten mejorar aún más el desempeño del algoritmo.

Todos los datos anteriores están calculados sobre una longitud de palabra máxima de 16 bits. Es necesario recordar que un pixel en una tarjeta VGA es representado por 8 bits, por lo cual la multiplicación no se ve afectada por la longitud del dato.

Para el procesamiento de una imagen (en general son transformaciones lineales), al igual que la "renderización" de un escenario virtual, se necesita (para generar 24 imágenes por segundo), una alta velocidad de procesamiento. Tomando el caso del tratamiento de una imagen calculada con la norma VGA, la cual tiene una resolución de 640 x 480 pixeles, si efectuamos una operación típica utilizando una máscara, tenemos que se debe realizar 307,200 multiplicaciones de matrices, donde cada matriz corresponde tiene un tamaño de 3x3 que corresponden al punto en cuestión y sus 8 próximos vecinos, por la matriz de transformación de un tamaño de 3x3. El procedimiento toma alrededor de 15.4 segundos utilizando el TMS320C25, para generar 24 imágenes por segundo. Sin embargo, en un entorno virtual raramente es necesario hacer modificaciones sobre toda la pantalla todo el tiempo. Por lo general se efectúan modificaciones solo sobre los sectores que sufrieron cambio.

Basados en lo anterior se cree que es posible "renderizar" las imágenes a un ritmo de 24 cuadros por segundo, para obtener el efecto de movimiento. Estas imágenes serían provenientes de un escenario virtual, y siempre y cuando se use una arquitectura en base a dos DSP's TMS320C50 que se encarguen, respectivamente, del cálculo del entorno virtual y de la renderización de las imágenes.

Bibliografía

[1] *"TMS320C5X, User's Guide"*, Texas Instruments, 1993, USA.

[2] *"TMS320C5X, DSP Design Workshop"*, Texas Instruments, 1996, USA.

[3] Alan Watt, *"3D Computer Graphics"*, Addison Wesley Publishing Company, 1994, UK.

*** Nota aclaratoria:**

Algunas personas pueden conciderar incorrecta la utilización del término "Procesador de Señales Digitales", ya que el dogma requiere que sea escrito "Procesador Digital de Señales". Sin embargo este último término resulta de una mala interpretación del idioma inglés, de Digital Signal Processor o en su defecto Digital Signal Processing.

De hecho, la enciclopedia WEBOPAEDIA describe como sigue a la abreviación DSP:

"Short for digital signal processing, which refers to manipulating analog information, such as sound or photographs that has been converted into a digital form. DSP also implies the use of a data compression technique.

When used as a noun, DSP stands for digital signal processor, a special type of coprocessor designed for performing the mathematics involved in DSP. Most DSPs are programmable, which means that they can be used for manipulating different types of information, including sound, images, and video".

Para entender lo anterior es necesario referirse al tratamiento de señales analógicas y a la forma en que los procesadores "mastican" la información. Cuando se habla de señales digitales en realidad se refiere a señales analógicas que han sido numerizadas o digitalizadas (del inglés digitalized). El tener una señal digitalizada nos permite llevar a cabo tratamientos numéricos como son el análisis discreto de Fourier y otros. Este tipo de tratamiento es confundido muy frecuentemente como un "procesamiento digital", sin embargo, este término por si mismo no tiene ningún significado.

Por otro lado, es bien conocido que los microprocesadores en general (controladores, MPU's, etc.) manejan información en forma de ceros y unos (representaciones numéricas de estados físicos de un semiconductor "pasa" (nivel alto de voltaje) y "no pasa" (nivel bajo de voltaje). Desde luego esto no los convierte en procesadores digitales ya que la operaciones que llevan a cabo son en pocas palabras sumas y multiplicaciones, corrimientos, comparaciones, búsqueda y almacenamiento en memoria y otras que en definitiva no los convierte en entidades abstractas (los dígitos ["números"], son entidades abstractas creadas por la mente humana, y que le ayudan a describir y comprender el ambiente en donde vive y en general el cosmos). Los programas que llevan a cabo complejas operaciones se basan en las operaciones básicas de los procesadores para elaborar útiles y diversos tratamientos sobre números.

Ahora bien, cabe expresar aquí la duda sobre que es entonces una señal digital. La respuesta está en la consideración de un conjunto de números agrupados de tal manera que puedan ser identificados como los representantes de un evento que ocurra y que pueda ser medible. Si a este conjunto de números lo llamo señal digital entonces se puede en algún momento aplicar un método numérico para tratar de obetener alguna información en específico.

Para un procesador, los números que representan una señal (señal digital), no significan nada ya que siguen siendo ceros y unos al final. Sin embargo, tienen sentido para el usuario que va a llevar a cabo un procesamiento de su señal digital.

Por falta de espacio y ya que el contenido de este artículo no concierne en definir lo que es el tratamiento de señales digitales, esta breve aclaración por parte del autor (Héctor Samuel García Salas), espera que sea de ayuda en el porqué de la utilización de los términos antes mencionados.