SIMUMEK: UN SISTEMA DE SIMULACIÓN GRÁFICA 3D PARA OPERACIONES DE MECANIZADO CNC

Aitor Moreno, Carlos Toro, Iosu Arizkuren, Alvaro Segura, Jorge Posada
Asociación VICOMTech, Paseo Mikeletegi 57, 20009 San Sebastian, España
{amoreno, ctoro, iarizkuren, asegura, jposada}@vicomtech.es

Marcelino Novo
FAGOR AUTOMATION S. Coop. Bº San Andrés, 19 - Apdo.144 - 20500 Mondragón, España
mnovo@aotek.es

Juanjo Falcón
SOME Sistemas Informáticos S.L. Avda. Navarra s/n (oficina 10). 20500 Mondragón, España
jjfalcon@somesi.com

Nieves Alcaín
Alecop, S. Coop. Loramendi, 11, Apto. 81 - 20500 Mondragón, España
nalcain@alecop.es

RESUMEN

En este trabajo se presenta un enfoque distinto en el desarrollo de un kernel de simulación para CNC cuya representación interna está basada en niveles. La característica básica de la representación basada en niveles es que permite la realización de operaciones booleanas entre modelos 3D de una forma rápida, pudiendo así ser utilizada en simulaciones donde no existe conocimiento a priori del programa NC que se está ejecutando (*online*). Para probar dicho kernel de simulación, se ha integrado dentro del control FAGOR 8070 CNC, en donde se ha conseguido simular síncronamente la salida real de la máquina. Adicionalmente, también se ha integrado dentro del software educativo WinUniSoft, logrando intercambiar las librerias comerciales originales por el kernel de simulación aquí presentado. Los resultados muestran que dichas integraciones son factibles y que obtienen unos resultados similares o incluso mejores que las alternativas tradicionales.

1. INTRODUCCIÓN

La máquina herramienta ha jugado un papel fundamental en el desarrollo tecnológico del mundo hasta el punto que no es una exageración decir que la tasa del desarrollo de máquinas herramientas gobierna directamente la tasa del desarrollo industrial.

Gracias a la utilización de la máquina herramienta se ha podido realizar de forma práctica, maquinaria de todo tipo que, aunque concebida y realizada, no podía ser comercializada por no existir medios adecuados para su construcción industrial.

Un sistema CNC (*Computerized Numeric Control, o* control numérico por ordenador), es un conjunto de un software controlador y un hardware dedicado al corte y separación de material de un bruto que puede ser metálico, polimérico o en general cualquier material solido con el que se puedan realizar herramientas o elementos de las mismas.

Cada sistema CNC posee una arquitectura que en términos generales puede ser dividida en tres partes o módulos fundamentales (i) la Máquina física, (ii) el Kernel (o motor) de procesos y (iii) el Kernel (o motor) de simulación.

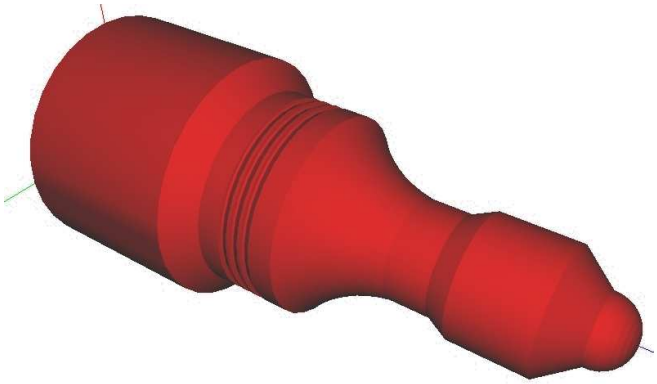

Figura 1: Un ejemplo de mecanizado utilizando una máquina CNC. El resultado corresponde a una simulación de torno.

La máquina NC es el conjunto de mecanismos físicos tales como el motor, los actuadores, etc y conforma la parte hardware de un sistema CNC. La comunicación entre este modulo y el resto del sistema se realiza a través del pre-procesador NC, que está ligado a cada tipo de máquina.

El Kernel de procesos gestiona las comunicaciones con el simulador, el usuario y la máquina, adaptando la información a cada preprocesador concreto. El kernel lee e interpreta el código G (que es el código de maquinado que representa las acciones a ser procesadas),

calculando los pulsos eléctricos que se tienen que mandar a la máquina usando un interpolador.

El Kernel de procesos también recibe mensajes de la máquina NC producidos por los encoders y sensores, por ejemplo, para devolver la situación actual de la máquina. El kernel de simulación CNC es el módulo que gestiona la salida gráfica del CNC (Figura 1). Puede utilizar como información de entrada, tanto la lectura del programa G como la retroalimentación de la máquina CNC. El desarrollo del módulo es costoso y suele utilizarse sistemas comerciales de terceras empresas que son integrados en el CNC, generalmente de una forma fija y cerrada, por lo que su posterior cambio o eliminación es complicada.

La motivación de este trabajo es intentar demostrar que se pueden desarrollar sistemas de simulación alternativos y que pueden ser integrados en sistemas actuales de producción.

Este trabajo presenta el kernel de simulación que se ha integrado dentro de dos sistemas comerciales de características diferentes.

La primera sección analiza el trabajo previo relacionado con las capacidades de los sistemas comerciales existentes y en las estructuras internas sobre las que se basan las modularidad y la interoperabilidad de los diferentes subsistemas. En la segunda sección, se describen las principales características del kernel de simulación que va a ser integrado. En las secciones finales, el proceso de integración y los resultados obtenidos son comentados. Para finalizar, las conclusiones y el trabajo futuro son expuestos.

2. ESTADO DEL ARTE

La simulación virtual de mecanizado CNC ha sido investigada extensamente por diferentes autores. Algunas de las técnicas tradicionales no almacenan ninguna información geométrica durante la simulación, sino que simplemente modifican la salida grafica en

pantalla utilizando técnicas basadas en imagen, siendo la base de la mayoría de soluciones comerciales para simulación de mecanizado CNC.

Sin embargo, este tipo de técnicas basadas en imagen tienen importantes limitaciones, como pueden ser: i) la regeneración de una nueva vista desde un punto de vista diferente requiere recalcular la simulación completa y ii) la dificultad para detectar errores bajo unas tolerancias realistas. Por otro lado, la simplicidad de la representación y la velocidad son aspectos positivos a tener en cuenta.

Existen otro tipo de soluciones que almacenan en la memoria del ordenador los resultados intermedios, teniendo una representación geométrica 3D de los objetos dinámicos, lo que permite mantener la representación de forma permanente e independiente del punto de vista, mejor control de la exactitud de la geometría, detección de colisiones geométricas,...

Van Hook[13] extendió en sus trabajos la estructura *Z-Buffer* (denominada estructura *Dexel*) para la verificación gráfica. En su trabajo presentó un método de *scan* para convertir superficies a elementos definidos según la representación *Dexel*. En cada *Dexel,* almacena los valores de *Z-Buffer* de la superficie más cercano y más lejano.

Otras técnicas tradicionales dentro de la Geometría Computacional son las representaciones i) *B-Rep*, ii) *CSG (Constructive Solid Geometry)* , iii) *HSD* (Descomposición Espacial Jerárquica).

A pesar de que la representación *B-Rep* es la más utilizada en el modelado de sólidos dentro de los sistemas CAD actuales, su utilización para representar sólidos dinámicos (aquellos cuya geometría cambia en el tiempo), como son los que existen en las simulaciones de mecanizado NC, no es compatible con las necesidades de velocidad e interactividad que una simulación virtual requiere. Un problema similar ocurre con la estructura *CSG*, con ordenes computacionales en torno a $O(n^2)$, donde n es el número de primitivas [10, 11].

Para poder solucionar los problemas computacionales que la simulación de mecanizados NC conlleva, se suele recurrir a técnicas que aproximan la geometría o que subdividen espacialmente el espacio de trabajo [4, 11].

La representación en voxeles es la técnica más clásica utilizada para subdividir volúmenes (octree clásicos[5]; octree extendidos[1, 2]) y combina en una sola definición partición espacial, representación de sólidos y soporte para operaciones booleanas. El inconveniente de esta representación es la gran cantidad de memoria que requiere para obtener resoluciones aceptables.

Maeng[7] utiliza el esquema *Z-Map*, donde la pieza se aproxima por un conjunto de vectores alineados con el eje Z, lo que restringe la aplicación a mecanizados de 3 ejes.

Las representaciones basadas en niveles (o capas) son ampliamente utilizadas en entornos de telemedicina asistida por ordenador, donde se generan modelos 3D a partir de un conjunto ordenado de niveles 2D (scanners, CAT's)[3, 9]. Sin embargo, son soluciones que no tienen en cuenta el tiempo real y la interactividad.

Es este trabajo, se utiliza un kernel de simulación que utiliza como representación interna una basada en niveles, cuyas características básicas son: i) realizar operaciones booleanas de forma interactiva, ii) permitir la visualización interactiva en 3D, con independencia del punto de vista, iii) capacidades de exportar e importar la geometría, iv) aceleración gráfica por hardware utilizando OpenGL.

3. KERNEL DE SIMULACIÓN. REPRESENTACIÓN GEOMÉTRICA

El kernel de simulación esta basado en una representación basada en niveles (LBR, *Level Based Representation*). Los objetos LBR se aproximan utilizando un conjunto de niveles paralelos espaciados uniformemente cuya distancia se denomina *Delta*. Cada nivel esta compuesto por una serie de contornos no auto-intersectantes y coplanares (Figura 2).

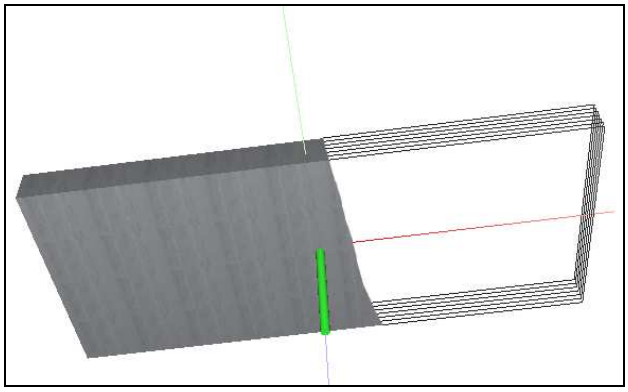

Figura 2: Un sólido LBR mostrando su visualización 3D (izquierda) y su aproximación en niveles (derecha).

El nivel de precisión obtenido viene determinado directamente por el número de niveles que se hayan utilizado, o lo que es lo mismo, por el valor de *Delta*.

3.1 Operaciones Booleanas

La característica principal de esta representación LBR es la posibilidad de realizar operaciones booleanas entre objetos LBR de una forma rápida y eficiente. Una operación booleana entre dos objetos LBR, se descompone en una serie de operaciones booleanas entre niveles 2D, en donde un nivel pertenece a un objeto y el otro nivel pertenece al otro objeto (Figura 3).

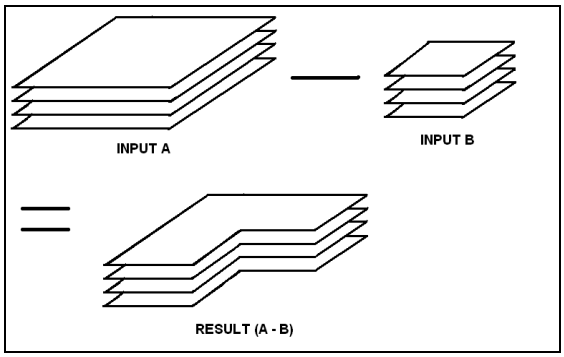

Figura 3: Un ejemplo de operación bolean entre dos objetos LBR.

La operación booleana en 2D se convierte en la función básica de bajo nivel, habiendo en la bibliografía varios autores [6, 12] que han aportado soluciones eficientes y rápidas.

Desde el punto de vista computacional, la utilización de la descomposición en niveles reduce la complejidad algorítmica de la operación booleana. Mientras hacerla de forma tradicional tiene un coste computacional de O(n4), la utilización de la descomposición en niveles reduce dicho coste a O(NL x (n x log n + k)) donde NL es el número de niveles que intervienen en la operación booleana, n es la media de puntos por nivele y k la media de contornos por nivel.

3.2 Partición Espacial. Regiones

Sin embargo, para obtener un rendimiento suficiente para su utilización en tiempo real, es necesario realizar optimizaciones al sistema a partir de algunas variables que puedan ser controladas.

Una de las variables que más afecta a la operación booleana en 2D es el número de puntos involucrados en dicha operación. Limitando dicho número se pueden obtener una mejora sustancial.

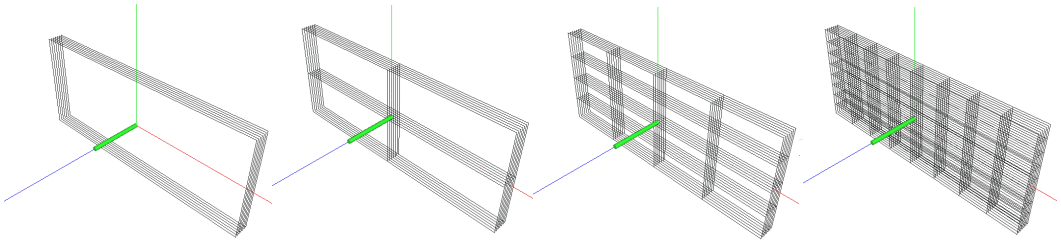

Figura 4: Diferentes particiones espaciales para un mismo modelo LBR. De izquierda a derecha: 1x1 regiones, 2x2 regiones, 4x4 regiones y 8x8 regiones.

Para ello se introduce una partición espacial al nivel de objeto, subdividiéndolo en un conjunto de regiones, cada una de las cuales es un objeto LBR (Figura 4).

Al realizar esta subdivisión se optimiza el proceso en dos aspectos:

i) Se aprovecha la localidad temporal y espacial. Entre un instante *t* y el siguiente, no suele ocurrir que la herramienta se mueva en exceso, por lo que la región afectada por la herramienta es, sin riesgo a equivocarse, la misma que en el instante anterior. Esto tiene como consecuencia, que temporalmente hablando, es como si se estuviera trabajando con un bruto de tamaño mucho menor, lo que reduce la complejidad de las operaciones boolenas subyacentes.

ii) La propia operación booleana. Como se ha comentado previamente, el hecho de que los niveles enfrentados tengan menos puntos y menos contornos afecta positivamente al rendimiento de la operación booleana 2D, lo que redunda en una optimización global de la operación booleana 3D.

Sin embargo, la inclusión de la subdivisión espacial conlleva un coste asociado. En cada instante, hay que seleccionar qué regiones están siendo afectadas por el movimiento de la herramienta. Esta selección tiene un coste algorítmico que varía entre O(log NR), en caso de utilización de estructuras arbóreas, y O(NR), en caso de utilizar estructuras lineales, siendo NR el numero de regiones del objeto 3D.

3.3 Reconstrucción

Los algoritmos de visualización de modelos LBR son más complejos puesto que es necesario reconstruir la geometría 3D a partir de los niveles que lo conforman.

Una de las formas más sencillas de realizar esta reconstrucción es la extrusión de los niveles en la dirección perpendicular al plano que los contienen. De esta forma, se obtendría un objeto 3D que consta de los niveles y sus extrusiones, conformando un sólido 3D que se puede triangularizar y dibujar en pantalla.

Sin embargo, los gráficos obtenidos tienen un efecto ‘escalera’, lo que provoca que la calidad gráfica obtenida no sea alta. Para eliminar este efecto ‘escalera’, se propone la unión de cada par de niveles consecutivos. Esto requiere que cada nivel tenga conocimiento de cuales son sus niveles adyacentes y lo que es más importante, cómo unirse a ellos (Figura 5).

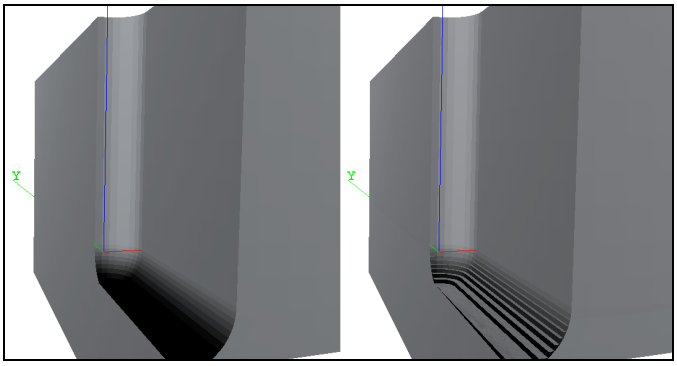

Figura 5: Visualización de un objeto LBR modificado por una herramienta ball-end. a) Uniendo niveles consecutivos y b) Extruyendo perpendicularmente cada nivel hacia el nivel siguiente.

3.4 Generación de volúmenes de barrido

Las operaciones booleanas ya descritas toman como entrada dos sólidos 3D. Uno de ellos es la representación del bruto en un instante dado. El otro es el volumen de barrido de la herramienta actual desde su posición anterior a su posición actual, lo que se define como un movimiento de la herramienta.

La algoritmo para la generación del volumen de barrido toma como entrada la propia definición de la herramienta actual (su perfil geométrico) y la definición del movimiento realizado. A la salida se obtiene un sólido representado en niveles compatible con la definición interna del bruto. Posteriormente, se realiza la resta booleana entre el bruto y el volumen de barrido, cerrándose así el ciclo.

Hay dos formas de generar la geometría del volumen de barrido de la herramienta. La primera es generar la geometría 3D del volumen de barrido (triángulos) para posteriormente intersectar dichos triángulos con una serie de planos paralelos. Esta opción es

algorítmicamente costosa, ya que al coste de la generación del volumen de barrido poligonal inicial, hay que añadir el coste de intersectar cada uno de los triángulos generados por la serie de planos paralelos, lo que generaría una serie de líneas que hay que ordenar para poder generar los contornos que conforman los niveles que aproximan el volumen de barrido.

Otra opción es generar de una manera directa los niveles que aproximan el volumen de barrido utilizando el conocimiento que se dispone, como es la propia definición de la herramienta y los movimientos permitidos por la misma. Para cada par de herramienta y tipo de movimiento, se resuelven las ecuaciones matemáticas obteniendo la información suficiente para poder generar los niveles en los que se descompone el volumen de barrido. Sin embargo, realizar este tipo de análisis matemático para todas las combinaciones de tipos de herramienta y de movimientos es costoso en cuanto a tiempo de desarrollo y análisis se refiere.

Es por este último factor que se sugiere la implantación de una combinación de ambas opciones. De todas las herramientas posibles, se seleccionan aquellas que son más utilizadas y aquellos movimientos más típicos en programas CNC convencionales. Este subgrupo de herramientas y movimientos se implementa de forma eficiente utilizando la segunda opción planteada. El resto de combinaciones de herramienta y tipo de movimiento se ejecutarán utilizando la primera opción. De esta forma se busca un compromiso entre eficiencia conseguida y el tiempo de desarrollo de la solución.

Con este sistema híbrido, se puede extender fácilmente el conjunto de herramientas y movimientos optimizado según las aplicaciones.

3.5 Procesos de corte implementados en el kernel de simulación

Actualmente, el simulador es capaz de tratar movimientos lineales y en arco, en distintos tipos de mecanizado, como son fresado y torno y torno eje C. A continuación se detallan que operaciones de corte están presentes en el kernel de simulación:

Operaciones de fresado.

a) Mecanizado de 2.5 ejes con herramientas cilíndricas, cónicas y ball-end en los tres planos principales.

b) Mecanizado de 3 ejes completos con la herramienta cilíndrica en los tres planos principales.

Operaciones de torno

a) Torno tradicional con herramienta genérica.

b) Torno eje C con herramienta cilíndrica.

4. INTEGRACIÓN

El proceso de integración del módulo de simulación se describe a continuación y está compuesto por las siguientes acciones.

1. Elegir los sistemas comerciales objetivo
2. Especificar un API de comunicación común a los sistemas comerciales elegidos e implementarla.
3. Integrar el kernel y el API dentro de los sistemas comerciales y probar los resultados.

4.1 Sistemas Comerciales

Los sistemas comerciales elegidos han sido dos:

El primero es el **FAGOR 8070 CNC** (Fagor Automation) que puede controlar hasta 16 ejes, 3 y 4 ‘spindles’, teniendo la posibilidad de integrar software de terceras empresas. Esta capacidad es la principal razón por la que ha sido elegido como candidato para realizar la integración del kernel de simulación.

En este escenario, el kernel de simulación toma como entrada la salida directa (o feedback) del control CNC, lo que se denomina simulación online, y la característica principal que posee es que no está garantizado que la salida teórica (del programa CNC) y la simulada sean iguales. El objetivo es, por lo tanto, la simulación tiene esté lo mejor sincronizada posible con la salida de la máquina CNC, tan sólo sabiendo donde está la herramienta en cada instante de tiempo.

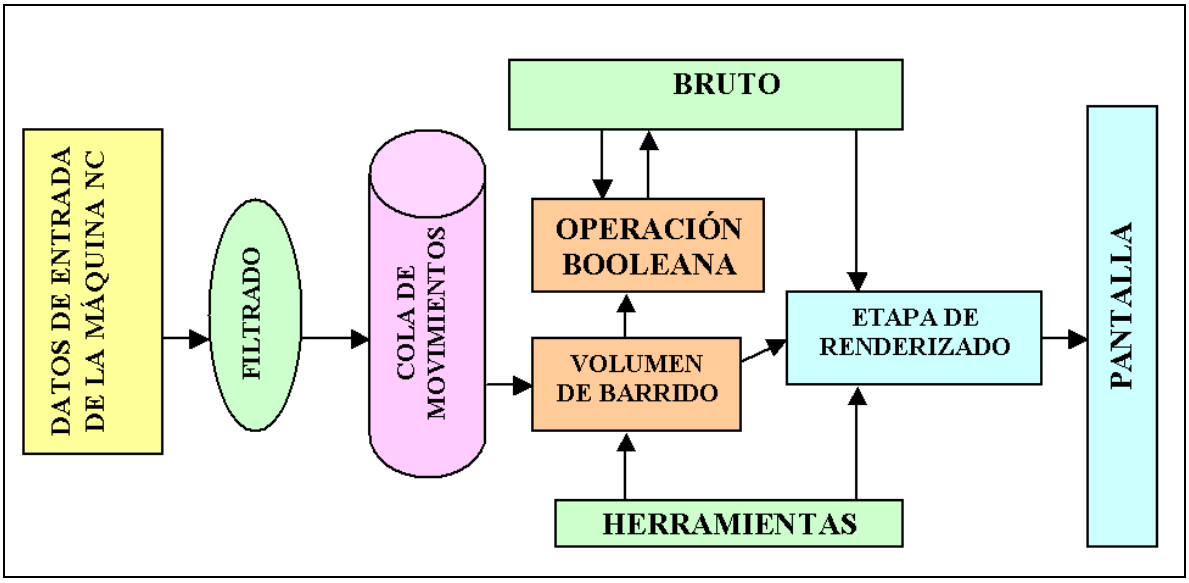

Figura 6: Arquitectura del Kernel de Simulación. Los datos de entrada de la máquina NC son filtrados y encolados (izquierda). Con cada movimiento, un volumen de barrido es generado por el Modulo de Volúmenes de Barrido, usando la descripción de la herramienta activa. Finalmente, la operación booleana es llevada a cabo entre el volumen de barrido y el bruto. La etapa final es la visualización, donde se realiza la reconstrucción del bruto y se genera la imagen gráfica final.

El control FAGOR 8070 CNC envía la posición de la herramienta y del bruto cada 2 ms. Esto es una altísima frecuencia que produce 500 movimientos pequeñísimos cada segundo. El mero hecho de leer y procesar esta entrada requiere un alto porcentaje de CPU, afectando al rendimiento del kernel de simulación.

Como primera fase, es necesario realizar un filtrado de los datos de entrada (Figura 6), para de esta forma, reducir la cantidad de puntos útiles que entran al kernel de simulación. Este filtro está compuesto por una combinación de filtros temporales y espaciales, es decir, para una posición dada, un punto se eliminará si:

- está alineado con los 5 puntos anteriores.
- ha pasado menos de 10 ms.

La primera restricción pretende conseguir movimientos lineales lo más largos posibles, ya que el kernel de simulación es más eficiente en estos casos. La segunda restricción obliga a que al menos haya un movimiento cada cierto tiempo.

El hardware disponible para el kernel de simulación suele estar limitado, y además la simulación misma es una tarea más dentro de los procesos del control y sin prioridades de ejecución. Esta característica elimina toda posibilidad de contar con hardware especializado y de alto rendimiento, que pueda ayudar a mejorar el rendimiento del kernel de simulación.

El segundo escenario es el software **WinUniSoft**, un sistema de simulación pedagógico que simula programas de CNC en entornos PC, analizando los posibles errores cometidos. Tiene una alta calidad gráfica ya que delega en software propietario y comercial las tareas de generación de los gráficos, la simulación mecánica de los programas CNC y la detección de errores producidos durante la simulación.

En contraste con el anterior escenario, en este escenario se carece de una máquina real y por lo tanto, la sincronización no es necesaria. Eliminando esta restricción, el tiempo de simulación obtenido se reduce a una aproximación a metodologías *'best-effort'*, es decir, realizar la simulación tan rápido como sea posible, por lo que, a mayor capacidades hardware (tanto en CPU como en tarjetas gráficas), se obtendrán mejores tiempos de simulación y mejor calidad gráfica.

4.2 API de comunicación

Para poder integrar el kernel de simulación en estos sistemas tan diferentes, la primera tarea es tratar de encontrar el conjunto común de funciones que ambos escenarios van a utilizar. Este conjunto de funciones se determina gradualmente, mediante una serie de

reuniones entre responsables de ambos productos. El núcleo principal del API se desarrolla especificando sistemas de referencia, unidades de medida, vocabulario, códigos de error, códigos de función, estructuras básicas, y se complementa con una descripción completa de las funciones clave del API.

Según se implementan distintas funciones básicas del API se añaden otras funciones a la definición del API. Estas nuevas funciones son más específicas, algunas incluso orientadas a un solo sistema comercial, debido a que las necesidades de ambos sistemas comerciales no son exactamente las mismas, lo que puede considerarse una ventaja debido a que la funcionalidad extra del API puede ser utilizada en el futuro, extendiendo así las capacidades de su producto.

5. RESULTADOS OBTENIDOS

Los resultados esperados en simulaciones online son que la diferencia de tiempos entre la entrada de datos de la simulación real y la visualización gráfica obtenida tras la simulación sea la mínima posible, es decir, conseguir la sincronización entre la simulación real y la virtual.

Como ya se ha expuesto anteriormente, no existe ningún tipo de preproceso lo que implica que no se puede precalcular nada, o lo que es lo mismo, que dado un movimiento, se tenga que realizar todos los pasos geométricos completos secuencialmente. Operaciones costosas como la operación booleana, la reconstrucción 3D o el dibujado en pantalla deben ser realizados con la mayor rapidez posible.

La velocidad del sistema está limitada por el número de operaciones booleanas por segundo que el hardware del sistema puede realizar, a lo que hay que añadir que no todas las operaciones booleanas son igualmente costosas.

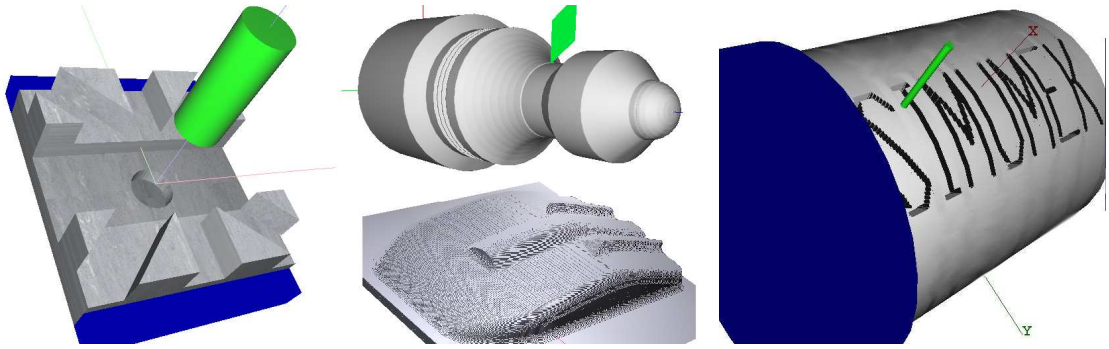

Figura 7: Algunos resultados de simulación. De Izquierda a Derecha: Fresado 2.5D, Torno 2.5D, Fresado 3D y Torno Eje C.

En consecuencia, el rendimiento del sistema está limitado tanto por el número de operaciones booleanas como su complejidad individual. Para optimizar el número de operaciones boolenas que se requieren, se pueden rebajar el número de niveles que aproximan los modelos 3D, lo que equivale a reducir el número de operaciones booleanas.

Para optimizar la complejidad algorítmica de las operaciones booleanas es necesario que el número de puntos esté controlado. Para ello, se puede hacer uso de distintas particiones espaciales (dividir el bruto en un número diferente de regiones) de forma que el número de puntos por región esté acotado dentro de lo que el hardware pueda simular. Ambas optimizaciones en conjunto permiten la simulación online de procesos de mecanizado simples y de media complejidad.

En situaciones totalmente adversas, como pueden ser modelos enormes de simulación, tanto en duración como en extensión, el sistema de simulación necesita una elección optima de los parámetros para que la simulación pueda llevarse a cabo satisfactoriamente.

En la Tabla 1 se muestran los resultados de los test realizados más significativos. Se han elegido tres modelos de complejidad diferente y se obtenido el tiempo de simulación utilizando diferentes particiones espaciales. También se muestra el tiempo de simulación real que la control FAGOR 8070 CNC obtiene por sí solo. Hay que hacer notar, que es imposible que el tiempo de simulación sea menor que el tiempo real del control CNC, puesto que el

kernel de simulación obtiene los datos una vez que el propio control CNC los ha procesado. Por lo tanto, el objetivo de las pruebas es comprobar si la diferencia de tiempos se acerca a cero.

Modelo (Complejidad)	Número Regiones	Tiempo de Simulación (s)	Factor
Test-1 (baja)	1x1	1740	10.2
	2x2	301	1.7
	4x2	221	1.3
	8x2	**174**	1.02
	Maqu Real	**170**	**1.0**
Test-2 (Media)	1x1	2020	9.6
	2x4	228	1.10
	4x8	**211**	1.01
	8x16	305	1.45
	Maqu Real	**210**	**1.0**
Test-3 (Alta)	1x1	4215	9.3
	4x2	920	2.04
	8x4	509	1.13
	8x8	**470**	1.04
	16x16	554	1.23
	Maqu Real	**450**	**1.0**

Tabla 1: Tiempos de simulación para tres modelos diferentes con diferentes complejidades y variando el número de regiones. El factor se ha calculado dividiendo el tiempo obtenido por el tiempo de mecanizado real. La complejidad de cada modelo es subjetiva, sirviendo para clasificar los modelos según sean sencillos o complicados de simular.

En la misma tabla se puede encontrar que tanto para el modelo simple como para el de complejidad media, se puede encontrar una simulación síncrona con la simulación real (véase los tiempos en negrita). Para el modelo complejo, el mejor tiempo obtenido tiene un desfase de 20s (un factor de 1.04), lo que confirma que el kernel de simulación posee problemas con modelos complejos y que hay que afinar mucho los parámetros del sistema para poder obtener unas simulaciones síncronas con la máquina real.

6. CONCLUSIONES

Se ha mostrado que la integración de un kernel de simulación en dos sistemas comerciales es posible. Además, dicho kernel de simulación se ha desarrollado con un

enfoque distinto al de la mayoría de sistemas comerciales, ofreciendo las mismas prestaciones que los mismos e incluso mejorandolas.

La integración se ha realizado a traves de un API de comunicación que permite ser integrado en distintos entornos de simulación.

Los resultados de diversas simulaciones muestran que el kernel de simulación obtiene tiempos de simulación sincronizados con los tiempos reales en la gran mayoría de los test realizados. En situaciones adversas, el sistema de simulación tiene que ser configurado muy rigurosamente para que la simulación tenga lugar de una forma interactiva y sincronizada.

7. TRABAJO FUTURO

Viendo los resultado obtenidos, la búsqueda de un mejor rendimiento es la tarea más relevante. Aunque el rendimiento del kernel de simulación es bueno para modelos sencillos, es muy importante obtener una parametrización correcta del sistema para obtener un rendimiento óptimo. En modelos complejos, este afinamiento de los parámetros es incluso más difícil y a la vez, más necesario para obtener un rendimiento adecuado.

Un sistema predictivo y adaptativo, utilizando técnicas de control sería una ayuda al sistema de simulación, de forma que, monitorizando las variables principales del sistema, pueda actuar adaptativamente sobre las variables de control del simulador, para obtener una simulación mejor.

Otro aspecto a mejorar es la calidad gráfica generada, lo que es algo importante para el usuario final. Este aspecto está muy influenciado por el hardware gráfico utilizado, por lo que una mejora en dicho hardware puede aportar un beneficio considerable. Sin embargo, no todas las ventajas vienen del hardware. La mejora de los algoritmos de reconstrucción (generación de geometría y sus normales), ayudarán a obtener un alto realismo en los modelos.

8. AGRADECIMIENTOS

Este trabajo está enmarcado dentro del programa INTEK del Gobierno Vasco. Los autores agradecen a los participantes del proyecto SIMUMEK[8] (Fagor Automation S. Coop, SOME Sistemas Informáticos y Alecop S.A.) por su colaboración.

9. REFERENCIAS

1. Brunet, P., Navazo, I., 1990, *Solid representation and operation using extended octrees*. In ACM Transactions on Graphics 9, 2 (1990), pp. 170-197.
2. Cano, P., 2002, *Representation of polyhedral objects using sp-octrees*. In Journal of WSCG 10, 1 (2002), pp. 95 - 101.
3. Fujimori, T., Suzuki, H., Kobayashi, Y., Kase, K., 2004, *Contouring medial surface of thin plate structure using local marching cubes*. In Shape Modeling Applications, 2004. Proceedings 2004 pp. 297 – 306.
4. Jerard R. B., Drysdale R. L., 1989, *Methods for detecting errors in sculptured surface machining*. In IEEE Computer Graphics & Applications. Jan. 1989, pp. 26 –39.
5. Karunakaran K. P., Pawan K., Mondani P. K., Gupta N, Shanmuganathan P. V., Garg S., 2000, *Octree-Based Volumetric NC Simulation System*.
6. Leonov, M. V., 1998, *Implementation of boolean operations on sets of polygons in the plane*. In BS Thesis (in Russian), Novosibirsk State University, 1998.
7. Maeng, S. R., Baek, N., Shin, S. Y., Choi, B. K., 2003, *A fast Z-map update method for linearly moving tools*. In CAD, 35(11):995-1009, 2003.
8. Moreno, A., Toro, C., Arizkuren, I., Segura, A., Posada, J., 2004, *Development of an advanced 3d graphic simulation system for milling and lathe CNC machining*, Topics 16, 4 (2004), p. 23.
9. Nielson, G. M., 2003, *On marching cubes*. In Visualization and Computer Graphics, IEEE Transactions on Volume 9, Issue 3, July-Sept. 2003 pp. 283 - 297.
10. Rappoport, A., Spitz, S., 1997, *Interactive Boolean Operations for Conceptual Design of 3D Solids*. In Computer Graphics 1997, Volume 31, pp. 269 – 278.

11. Stewart, N., Leach, G., John, S., 2003, *Improved CSG Rendering using Overlap Graph Subtraction Sequences*. In International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia - GRAPHITE 2003, pp. 47 - 53.
12. Vatti, B. R., 1992, *A Generic Solution to Polygon Clipping*. In Commun ACM 35 1992, pp. 56 - 63.
13. Tim Van Hook, 1986, *Real Time shaded NC Milling Display*. SIGGraph86, Volume 20, Number 4, pp. 15 - 20.