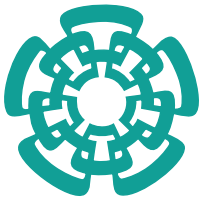


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Manipulación Tridimensional de Objetos Deformables Virtuales

Tesis que presenta

Julio Guadalupe Moctezuma Ramírez

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de
Ingeniería Eléctrica

Directores de la Tesis

Dr. Luis Gerardo de la Fraga
Dr. Vicente Parra Vega

México, D.F. **8 de diciembre de 2006**

Resumen

En esta tesis se presenta un sistema para la manipulación en tiempo real de objetos deformables virtuales sin retroalimentación de fuerzas. Los objetos se construyen a base de mallas de simplejos. En cada arista de la malla se acopla un sistema mecánico simple, compuesto por una masa un resorte y un amortiguador, sistema que le da a la malla su capacidad deformable. El modelo deformable se encuentra bajo la acción de un modelo rígido el cual se controla a través de un guante sensorizado. Se hizo uso de la biblioteca libre SOLID [38] para la detección de colisiones que tiene como fin determinar el momento y lugar en el que el objeto rígido entra en contacto con el modelo deformable. Esta tesis es una extensión de dos trabajos, "Modelos Deformables para Caracterizar Macromoléculas Biológicas" [25] y "Animación de Modelos Deformables" [27]. Ambos trabajos emplean mallas de simplejos y el sistema masa-resorte-amortiguador. De [25] se corrigió el algoritmo para la creación del toro mediante mallas de simplejos. De [27] se propone un nuevo esquema de deformación que permite obtener simulaciones realistas. En esta tesis la interacción se lleva a cabo en tiempo real; se logró la deformación de una esfera de densidad media con 1296 vértices, 1944 aristas y 650 caras.

La aplicación desarrollada se ejecuta a una frecuencia de treinta cuadros por segundo, y le fue incorporada visión estereoscópica usando unos lentes activos y una tarjeta de video NVIDIA quadro.

Abstract

A real time system for virtual deformable objects manipulation without force feedbak is presented in this thesis. The objects are built with simplex meshes. In each edge of the mesh a simple mechanical system composed by a mass, a spring and a damper is coupled. This system gives capabilities of deformation to the mesh. The deformable model is under the action of a rigid model which is controlled by an data glove. The free software library SOLID [38] was used for collisions detection, allowing us to determine the time and place where rigid object makes contact with the deformable model.

This thesis is an extension to the works "Modelos Deformables para Caracterizar Macromoléculas Biológicas" [25] and "Animación de Modelos Deformables" [27]. The torus creation algorithm of [25] was corrected in order to build the torus by simplex meshes. A different deformation model was used to get more realistic simulations than those presented in [27]. In this work the application perform the deformation in real time using a medium density object of 1296 vertices, 1944 edges and 650 faces.

The application developed executes at thirty frames per second and stereoscopic vision was added by the use of active glasses and an NVIDIA quadro video card.

Agradecimientos

Al Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional
Al Consejo Nacional de Ciencia y Tecnología
Al proyecto 45306 del Consejo Nacional de Ciencia y Tecnología
A Judith

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Antecedentes

El uso de modelos dinámicos deformables virtuales se ha extendido debido a que permiten una representación del mundo más apegada a la realidad que los modelos cinemáticos [20]. Existen diversas aplicaciones de los modelos deformables, entre ellas están la simulación de órganos humanos [15], simulación de fluidos [4], modelado de telas en personajes animados [40], entre otros.

Un modelo deformable virtual se constituye de dos partes, la representación del volumen o superficie, que especifica la geometría del objeto deformable, y el sistema de deformación que indica cómo se deformará el modelo ante la presencia de una acción externa o por inercia producida por interacciones anteriores. Para la primera parte existen diversos esquemas propuestos [20]. De uso extendido son las mallas de triángulos y las superficies parametricas. También existen otros modelos como el de mallas de simplejos, las cuales se definen como un conjunto de vértices y una función de conectividad [27].

En lo que respecta al sistema de deformación, existen distintos modelos que permiten la deformación, cada modelo aborda de manera distinta la forma en que un objeto debe deformarse. Sin embargo los modelos basados en sistemas físicos han demostrado ser altamente realistas. Así por ejemplo se tiene el modelo de partículas [15], el el cual el objeto se considera relleno de esferas rígidas. Al aplicar una fuerza, las partículas se reacomodan hasta alcanzar un estado estable dándole una nueva forma al objeto. Otro modelo de uso extendido es un sistema físico simple compuesto por una masa, un resorte y un amortiguador.

Actualmente la simulación basada en física, es decir aquella en la que el esquema de deformación se basa en un modelo físico, es de amplio uso, sin embargo el costo computacional es alto por lo que principalmente se trata de sistemas de interacción programada, es decir donde la manipulación con el modelo no se efectúa de forma interactiva, si no a través de secuencias preprogramadas. Esto es aceptable para ciertas aplicaciones donde el usuario puede esperar hasta obtener los resultados de la simulación, sin embargo en otras aplicaciones, como lo son quirófanos virtuales por ejemplo, es necesario tener un sistema que opere en tiempo real.

Si bien los sistemas de simulación demandan muchos recursos de cómputo, es posible hacer ciertas simplificaciones del modelo sin perder demasiado realismo y que permitan obtener un sistema en tiempo real.

Este trabajo se basa principalmente en dos sistemas realizados anteriormente, el primero de ellos [25] emplea mallas de simplejos y el sistema masa resorte amortiguador para representar moléculas biológicas. Utilizan un modelo básico, una esfera o un cilindro (aunque también desarrollaron el algoritmo para representar un toro) basados en mallas de simplejos y los deforman hasta obtener una forma arbitraria. El sistema opera fuera de línea, aunque se realizó una simulación visual de la transformación del objeto, el tiempo de cómputo requerido depende de la densidad de la malla (el número de vértices que tenga) y una vez que la simulación se ejecuta no hay forma de interactuar con ella. Una mejora a ese trabajo se presenta en [27]. Principalmente se introduce la posibilidad de actuar con el modelo de manera más amplia, aunque sigue siendo un sistema de interacción programada. Se presentan diversas simulaciones, como un objeto deformable chocando con una pared rígida o bien siendo comprimido por dos paredes. También se presenta una capacidad de interacción simple, que si bien tratándose de un sistema que opera fuera de línea, no es posible interactuar con él mientras se efectúa la deformación del modelo, sí permite seleccionar a qué vértices del modelo se aplica una fuerza externa, esto se logra haciendo uso de un dispositivo llamado *Phantom Omni* [27, Cap 5], el cual consiste de un señalador conectado a través de una serie de brazos mecánicos a un sistema computarizado, que su vez se conecta a una computadora, este dispositivo permite seleccionar un punto en un espacio virtual tridimensional. Haciendo contacto con el señalador del Phantom Omni sobre una cara del modelo deformable virtual, se le indicaba al programa que se deseaba seleccionar el vértice más cercano al punto con el que se hizo contacto.

Al introducir interacción de objetos rígidos con el modelo deformable es necesario implementar un sistema de detección de colisiones para determinar qué partes del objeto deformable se encuentran bajo la acción del objeto rígido. La implementación de esta etapa tiene un impacto sustancial en el costo computacional del sistema resultante puesto que la detección de colisiones puede ser un proceso complejo. Dependiendo del grado de precisión que se busque, el sistema puede resultar inviable.

Para la etapa de detección de colisiones existen diversos enfoques propuestos que varían dependiendo del tipo de aplicación que se les dé, ya que algunos emplean consideraciones especiales para simplificar los cálculos asociados. Como por ejemplo el tipo de objetos usados, la velocidad de los mismos, los parámetros del modelo deformable, entre otros.

La interacción también conlleva la necesidad de plantear modelos matemáticos para tratar la simulación en los momentos en los que se presente un contacto. Estos modelos toman en cuenta varios factores, de los cuales resulta especialmente destacable la manera en la que se maneja la interacción. En particular, el modelo matemático es más complejo si el sistema requiere una retroalimentación de fuerzas, como sería en el caso de emplear un dispositivo como el Phantom Omni (con el que es posible *sentir* el objeto). El modelo matemático se simplifica si no requiere sentir al objeto, como es el caso de usar un guante sensorizado. Normalmente estos modelos se resuelven empleando métodos numéricos simplificados, ya sea por la complejidad de las soluciones exactas o porque resulte imposible

obtener una solución exacta de las ecuaciones. Un método numérico introduce un error en la solución del sistema, este error puede ser tolerable, dependiendo del tipo de simulación del que se trate o por el contrario puede incluso llevar al sistema a la inestabilidad. Por ello puede ser necesario incluir etapas de control y corrección del error, las cuales evitarán que el sistema se desestabilice. A un sistema que implementa estas etapas para cuerpos en contacto se conoce como estabilizador de la restricción, refiriéndose como restricción al hecho de que los cuerpos no deben atravesarse por lo que su movimiento se encuentra mutuamente restringido. Existen diversos tipos de controles numéricos que pueden usarse para mantener el error del sistema dentro de rangos adecuados, sin embargo estos controles dependen de constantes (nombradas normalmente ganancias) cuyo ajuste no es sencillo.

1.2. Situación actual

El estudio de modelos deformables y simulación basada en física ha sido amplio. Existen distintas aproximaciones propuestas para resolver los diferentes problemas implicados. Uno de estos problemas es estudiado en [16], donde los autores emplean un modelo para simular objetos en contacto que permite omitir la etapa de control del error al momento de calcular el contacto entre los distintos objetos. Esto es posible gracias a que emplean un esquema llamado *coordenadas reducidas* y superficies paramétricas para la representación espacial de los objetos modelados. Empleando NURBS obtienen formas suaves y de geometría más o menos arbitraria, que para el caso de representaciones basadas en polígonos requiere una malla densa. Sin embargo, las NURBS presentan problemas de continuidad en ciertos puntos, cuestión que los autores esquivan obligando a que los objetos se alejen antes de que se produzca el contacto en los puntos con discontinuidades. El hecho de que no se necesite una etapa de estabilización de la restricción, permite un margen de error mayor en los algoritmos numéricos usados para resolver las ecuaciones diferenciales del modelo físico, sin demeritar demasiado los resultados visuales. Los autores señalan que el algoritmo que presentan permite fácilmente incorporar fricción al modelo.

Continuando con el tratamiento del error, si se emplea un modelo que requiere forzosamente estabilización de la restricción, es decir que no permite omitir el control del error, existen distintas aproximaciones (nótese que [16] es más bien la excepción que la regla). En [21] emplean un sistema de control de ganancia variable para estabilizar la restricción. El uso de ganancias fijas permite obtener resultados aceptables sólo para cierto tipo de problemas, pero no para casos generales. Por ejemplo, las ganancias que son adecuadas para objetos con movimiento lento no lo son para aquellos con velocidades grandes. Más aún, dichas ganancias son obtenidas de forma empírica y deben ajustarse finamente para obtener los mejores resultados. Los autores de este artículo señalan que su enfoque permite obtener generalidad al ser aplicable a distintos problemas, garantiza la estabilización de la restricción, el costo computacional es relativamente bajo ($O(n)$) y su paralelización es sencilla. Claro está que bajo este enfoque será necesario calcular la ganancia del control en tiempo de ejecución, aumentando con esto el costo computacional de todo el sistema.

Sin embargo, si se desea usar ganancias fijas ya sea por que el sistema lo permita, o bien por la simplicidad inherente de este enfoque, es necesario obtener los valores más adecuados. Los autores de [34] proponen el uso de métodos *predictores-correctores*, empleados en la teoría de control digital, en específico usan los métodos de Adams-Bashforth y Adams-Moulton. Mediante éstos es posible calcular los valores de las ganancias del control numérico. Si bien el método parece prometedor, los algoritmos de *predicción-corrección* usados requieren a su vez del cálculo de ciertas constantes, por lo que ahora el problema se traslada a la obtención de esas constantes, lo cual en el caso del artículo mencionado, se realiza a base de pruebas, situación que no difiere significativamente de estimar directamente las constantes del control numérico.

En [5] presentan otro enfoque para la estabilización de la restricción, el cual tiene como principal atractivo el que no requiere de constantes que se necesiten ajustar. El método es llamado *post-estabilización* y consiste en que de forma posterior a cada paso de integración para resolver las ecuaciones diferenciales asociadas, se aplica un paso de corrección. Este método permite eliminar el error conforme va apareciendo a diferencia del método que emplea un control numérico simple, en el cual si las ganancias son pequeñas, el error se irá disipando lentamente, y si son grandes, el error puede sobre-corregirse llevando a oscilaciones del objeto. Sin embargo, el método de corrección posterior agrega un requerimiento de cómputo mayor, además para errores grandes el algoritmo puede fallar según indican los autores del artículo.

En [23] presentan un algoritmo para la deformación de modelos volumétricos, este es una modificación del algoritmo *3d ChainMail* [12] en el cual se forma un volumen a través de elementos que se comportan de manera análoga a un eslabón en una cadena. Esto es que tienen un margen de movimiento dentro del cual no realizan acción alguna sobre los eslabones adyacentes, pero si se rebasa alguno de esos límites, entonces el eslabón jala y/o empuja a los eslabones adyacentes. Sin embargo este algoritmo presenta un problema, si se aplica una fuerza en cierta dirección y de cierta magnitud se logra una deformación, pero al aplicar la misma fuerza en sentido contrario el objeto no regresa a su posición original. Este comportamiento, según señalan en el artículo, produce simulaciones poco realistas y para evitarlo introdujeron modificaciones al algoritmo original. Al nuevo algoritmo le llamaron *s-chain*. Además, para obtener una interacción con retroalimentación de fuerzas, realizaron otra modificación la cual consiste en colocar resortes entre los eslabones. Si bien el enfoque que emplean les permite simular en tiempo real con retroalimentación de fuerzas para un objeto de $421,875$ elementos ($75 \times 75 \times 75$), esto solo representa una superficie de aproximadamente 30,000 elementos.

En [14] presentan un modelo que permite deformación en tiempo real con retroalimentación de fuerzas. Emplean un arreglo tridimensional de puntos igualmente espaciados que encierra al objeto a deformar, luego realizan una correspondencia entre los puntos del arreglo y los puntos del objeto. Mediante este sistema, al ejercer algún esfuerzo en el arreglo de puntos se deforma el objeto que contiene. Una ventaja notoria de este modelo es que al emplear un arreglo de solamente 64 puntos ($4 \times 4 \times 4$) obtienen una deformación realista y señalan que es posible aumentar la densidad de la malla y aún tener un sistema interactivo. Sin embargo el sistema se basa en la manipulación de los puntos del arreglo

por lo que el usuario no realiza una interacción directa con el objeto.

En el artículo [24] desarrollan un sistema de deformación con retroalimentación de fuerzas, que emplea una superficie especificada por una malla de triángulos, para representar un objeto con volumen. A cada vértice de la malla se asocia una masa y en cada arista colocan un resorte. Para dar un sentido de volumen a la superficie, emplean resortes que unen cada punto de la malla con su posición original, así después de aplicar un esfuerzo el objeto tiende a regresar a su posición original. La parte más importante del sistema presentado es una etapa de refinamiento de la malla en las áreas bajo contacto en tiempo de ejecución. Cuando el usuario realiza esfuerzo sobre una parte de la malla, los triángulos de esa área se dividen en triángulos menores para dar una sensación de gran detalle en ese lugar. Al modificar la densidad de la malla es necesario calcular valores de masa y constantes de los resortes asociados a los nuevos puntos y vértices creados, problema que resuelven de manera satisfactoria igualando la fuerza requerida para mover un vértice de la malla original (la malla de menor densidad) en una distancia dada con la requerida para un desplazamiento idéntico pero de la malla refinada. De esta manera el usuario no siente variación alguna en la rigidez del modelo. Presentan una serie de experimentos concluyendo que existe un límite más allá del cual un refinamiento mayor de la malla no representará mejora alguna dada la limitada capacidad del ser humano para discernir entre dos superficies con rugosidad ligeramente diferentes.

En el artículo [19] presentan un sistema de deformación de objetos virtuales volumétricos. Este sistema divide un objeto sólido en objetos más pequeños, que llaman celdas. Este enfoque describe tanto la superficie como el interior del modelo. Para darle al sistema un modelo de deformación realista, emplean resortes longitudinales en cada arista y además agregan resortes torsionales los cuales reaccionan a un esfuerzo de giro que sea aplicado. Una ventaja de este enfoque es que permite aumentar el nivel de detalle hasta que sea necesario, puesto que la división del objeto original puede realizarse en celdas tan pequeñas como se necesite, aunque claro que esto implica un aumento en el costo computacional del sistema. Como emplean celdas, no existen problemas al modelar objetos con huecos. Además del modelo de deformación, presentan una serie de primitivas de moldeado de los objetos. Eliminación de celdas, extrusión y unión de varios objetos son algunas de las herramientas de modelado. También se presentan algoritmos para pasar de una descripción de una superficie cerrada a un modelo por celdas. Los autores reportan haber logrado deformación en tiempo real con retroalimentación de fuerzas.

En [13] presentan una forma eficiente de representar una superficie deformable. El algoritmo aplica para sistemas en los que se desee emplear una representación con *voxels*, es decir, que utilizan un objeto relleno de cubos pequeños. A grandes rasgos los autores emplean el algoritmo *run lenght encoding* para comprimir los datos que especifican la estructura del objeto, de modo que el uso de memoria se disminuye drásticamente. Los autores reportan que este enfoque es útil en diversas tareas, como simulaciones de fluidos, metamorfosis, detección de colisiones, entre otras.

Los autores de [39] desarrollaron un sistema de trazado de formas tridimensionales que emplea NURBS. El usuario puede modificar una superficie tridimensional con trazos a mano utilizando un dispositivo inalámbrico como pincel. Este sistema está enfocado a

realizar trazos libres por lo que emplea un esquema de deformación geométrico y no uno físico. Sus aplicaciones principales son en el área de diseño.

1.3. Sistema implementado

El objetivo principal de este trabajo de tesis ha sido construir un sistema para la deformación de objetos virtuales basados en mallas de simplejos, sin retroalimentación de fuerzas, que funcione en tiempo real empleando un esquema de deformación similar al presentado en los trabajos [25, 27].

El sistema se compone de tres partes principales: el objeto deformable, el objeto que permitirá interactuar con él y una etapa de interacción que incluye un modelo matemático y un detector de colisiones.

Se emplean objetos rígidos para interactuar con el modelo deformable. Se utiliza un guante sensorizado para manipular los objetos rígidos. Dado que el guante no permite sentir el objeto virtual, se emplea un modelo sin fuerzas de retroalimentación. También se incorporó visión estereoscópica empleando unos lentes activos.

1.3.1. Metodología

Primero se definió el modelo del objeto deformable, se decidió emplear mallas de simplejos para especificar la geometría del objeto para retomar parte del trabajo presentado en [25, 27]. En [27], acoplan un sistema mecánico compuesto por una masa, un resorte y un amortiguador en cada arista de la malla. Sin embargo no obtienen una deformación realista. Por lo que se analizaron diferentes tipos de estos sistemas masa-resorte-amortiguador, buscando cual sistema es el más adecuado para incorporarlo en el modelo del objeto deformable.

En segundo lugar, se establecieron las ecuaciones que constituyen el modelo matemático correspondiente al sistema en los distintos estados en los que puede encontrarse, como lo son con o sin contacto.

El siguiente paso que se realizó fue la validación del algoritmo que genera un toro mediante mallas de simplejos, así como también se estableció el método mediante el cual se calculan las normales en una malla de simplejos.

A continuación se analizó el problema de detectar el momento y el lugar en el que el objeto deformable se haya bajo la acción de los objetos rígidos. Lo cual se realiza mediante un detector de colisiones. Se analizaron los distintos problemas y ventajas que conlleva cada enfoque para implementar aquel que fuera más adecuado.

Despues se analizó el motor de deformación presentado en [27] para determinar porque no presenta una simulación realista, planteándose qué posibles mejoras pueden hacérsele para obtener simulaciones más realistas.

El siguiente punto que se revisó fue la estabilidad del sistema, esto con el fin de determinar de que manera tener un sistema lo más estable posible.

1.4. Organización de la tesis

En el capítulo 2 se presenta un estudio de diversos modelos basados en el sistema mecánico simple compuesto por una masa un resorte y un amortiguador, de estos se seleccionarán aquellos que más convengan a los fines perseguidos y se incorporarán en la aplicación.

En el capítulo 3 se analizan las ecuaciones de modelado del sistema para los distintos estados del mismo, cuando el modelo deformable esta libre y cuando se encuentra bajo contacto.

Dado que se usarán mallas de simplejos en el capítulo 4 se presentan las diversas características principales de este método de representación de objetos, así como también se realiza el estudio de la construcción del toro mediante mallas de simplejos.

La etapa correspondiente al sistema de detección de colisiones se analiza en el capítulo 5, donde se presentan los diversos enfoques que existen para la detección de colisiones. También se presenta una biblioteca de detección de colisiones la cual será utilizada en la aplicación.

Si bien en el capítulo 2 se presenta el sistema masa-resorte-amortiguador, en el capítulo 6 se analiza ese sistema implementado en un objeto tridimensional así como la deformación debida a la acción de un objeto rígido externo. Se presenta el modelo como una revisión del motor de deformación empleado en la aplicación de [27].

El capítulo 7 está dedicado al estudio de la estabilidad del sistema así como de diferentes enfoques de estabilización basados en controles numéricos, los cuales permitirán estabilizar el sistema en caso de ser necesario.

En el capítulo 8 se detalla como se elaboró la aplicación, se describe cada bloque que la compone, también se presentan algunos experimentos realizados así como los resultados obtenidos.

Finalmente en el capítulo 9 se presentan las conclusiones así como el trabajo futuro.

Capítulo 2

Sistemas masa-resorte-amortiguador

2.1. Modelo de un resorte

Los objetos deformables con los que se trabajará en esta tesis están compuestos por mallas de simplejos. En cada arista de la malla se acopla un sistema mecánico simple, compuesto por una masa, un resorte y un amortiguador, que se identificará de aquí en adelante con las iniciales MRA. Este sistema mecánico es el que le da a la malla su característica deformable y es resuelto con las ecuaciones de movimiento de Newton.

La importancia de emplear un modelo físico radica en el realismo obtenido, pues es un modelo aplicable a sistemas físicos reales. Existen varios tipos de sistemas MRA, variando en la forma de conexión de los componentes principalmente.

En este capítulo se analizarán varios sistemas MRA con el objetivo de seleccionar el más adecuado para incorporarlo en el modelo deformable. Se presentan dos modelos sencillos y dos modelos de sistemas acoplados. Los modelos de un solo resorte difieren entre si, en la manera en que se interconectan los tres elementos. Solo uno de ellos se utilizó en la aplicación, el sistema MRA1, el cual se eligió por las características oscilatorias que presenta en su comportamiento. Se presentan también dos modelos de acoplamiento, uno en serie y otro en paralelo, que ayudarán a resolver el problema de la deformación en la malla de simplejos, ya que en esta los resortes se encuentran interconectados. Como se verá, el modelo paralelo no presenta ninguna característica destacable, por lo que se decidió emplear el modelo de acoplamiento en serie.

2.1.1. Modelo MRA 1

El modelo básico del sistema MRA es el que se muestra en la Fig. (2.1).

Este sistema se encuentra modelado por una ecuación diferencial de segundo grado para $x(t)$. El plano etiquetado como O es el origen ($x = 0$) del sistema.

El resorte es un elemento que se estira o encoge una distancia proporcional a la fuerza que actúa sobre él, como se muestra en la siguiente ecuación

$$f_k = k(x - x_0) \tag{2.1}$$

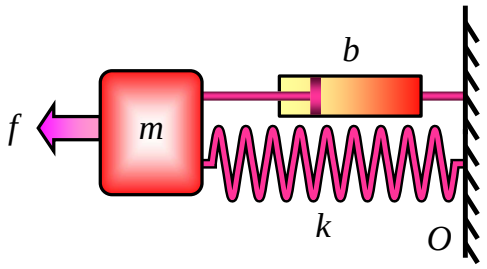


Figura 2.1: Sistema masa resorte amortiguador.

donde, x_0 es la longitud inicial del resorte y $k > 0$ es la constante del resorte.

Una característica importante del resorte es que puede almacenar energía, por ello se le llama fuerza de restitución a la fuerza asociada a un resorte, puesto que la energía que almacena lleva al resorte a tratar de recuperar su longitud inicial. Otro nombre que comúnmente recibe la fuerza del resorte es fuerza interna. Existen otro tipo de resortes cuyo comportamiento queda definido de manera distinta al señalado en la Ec. (2.1), sin embargo en este trabajo no se considerarán.

El amortiguador es un elemento que disipa energía, normalmente en forma de calor. Se considera que la fuerza asociada a este elemento es proporcional a la derivada del desplazamiento.

$$f_b = b\frac{\mathrm{d}x}{\mathrm{d}t} \tag{2.2}$$

donde, $b > 0$ es la constante del amortiguador.

Finalmente la masa tiene una fuerza asociada que por la segunda ley de Newton es proporcional a la aceleración, la cual es la segunda derivada del desplazamiento.

$$f_m = m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} \tag{2.3}$$

donde, $m > 0$ es la masa del cuerpo.

De modo que la ecuación diferencial que modela todos los efectos de los elementos es la siguiente

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + b\frac{\mathrm{d}x}{\mathrm{d}t} + k\Delta x = f \tag{2.4}$$

donde, $\Delta x = x - x_0$.

Esta es una ecuación diferencial ordinaria de segundo orden. Empleando una notación más compacta y considerando por simplicidad un resorte de longitud inicial cero, puede reescribirse como

$$m\ddot{x} + b\dot{x} + kx = f \tag{2.5}$$

Dado que todo el sistema de deformación se basa en el sistema MRA, es necesario evaluar esta ecuación de forma eficiente, aunque es posible tolerar cierto margen de error, considerando que el resultado final es una simulación gráfica, en la que un pequeño error menor a un píxel de tamaño es imperceptible.

2.1.2. Solución

Solución Exacta

La Ec. (2.5) puede ser resuelta de forma exacta, sin embargo la complejidad de las soluciones es grande, y depende de la fuerza aplicada.

Para analizar la solución exacta se empleará una notación ampliamente usada en la literatura. Simplemente hay que dividir la Ec. (2.5) entre m

$$\ddot{x} + \frac{b}{m}\dot{x} + \frac{k}{m}x = \frac{f}{m}$$

haciendo $\lambda = \frac{b}{m}$, $\omega^2 = \frac{k}{m}$ y $F = \frac{f}{m}$ se obtiene la expresión siguiente

$$\ddot{x} + \lambda\dot{x} + \omega^2 x = F \tag{2.6}$$

En el caso más simple, donde la fuerza es constante, la solución exacta tiene los casos siguientes

- Sobreamortiguado. Se presenta si $(\lambda^2 - \omega^2) > 0$

$$\begin{aligned} x(t) &= \mathrm{e}^{-\lambda t}\left(c_1\mathrm{e}^{\sqrt{\lambda^2-\omega^2}t} + c_2\mathrm{e}^{-\sqrt{\lambda^2-\omega^2}t}\right) + \frac{F}{\omega^2} \\ c_2 &= \frac{\left(x(0) - \frac{F}{\omega^2}\right)\left(\sqrt{\lambda^2-\omega^2} - \lambda\right) - x'(0)}{2\sqrt{\lambda^2-\omega^2}} \\ c_1 &= x(0) - c_2 - \frac{F}{\omega^2} \end{aligned} \tag{2.7}$$

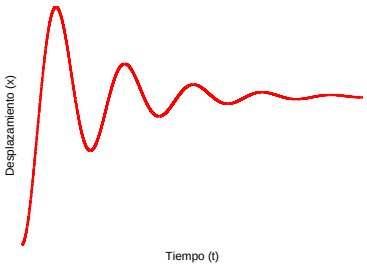
- Críticamente amortiguado. Se presenta si $(\lambda^2 - \omega^2) = 0$

$$\begin{aligned} x(t) &= \mathrm{e}^{-\lambda t}\left(c_1 + c_2 t\right) + \frac{F}{\omega^2} \\ c_1 &= x(0) - \frac{F}{\omega^2} \\ c_2 &= x'(0) + \lambda c_1 \end{aligned} \tag{2.8}$$

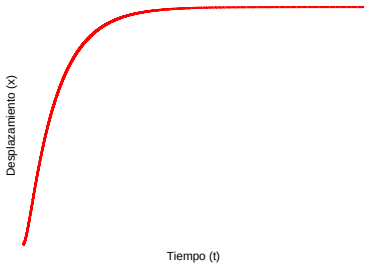
- Subamortiguado. Se presenta si $(\lambda^2 - \omega^2) < 0$

$$\begin{aligned} x(t) &= \mathrm{e}^{-\lambda t}\left(c_1\,\mathrm{sen}\left(\sqrt{\omega^2-\lambda^2}t\right) + c_2\cos\left(\sqrt{\omega^2+\lambda^2}t\right)\right) + \frac{F}{\omega^2} \\ c_2 &= x(0) - \frac{F}{\omega^2} \\ c_1 &= \frac{(x'(0) + \lambda c_2)}{\sqrt{\omega^2-\lambda^2}} \end{aligned} \tag{2.9}$$

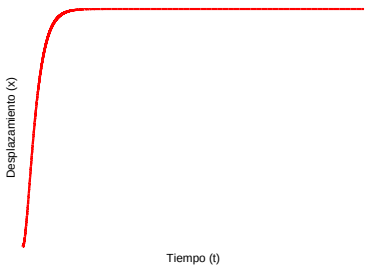
Claramente las tres soluciones tienen una complejidad considerable, por ello se analizará más adelante el uso de métodos numéricos. En la Fig. (2.2) se muestra un ejemplo de la curva solución para cada caso. En la Fig. 2.2(a) se presenta el caso subamortiguado, en la Fig. 2.2(b) el caso sobreamortiguado y finalmente en la Fig. 2.2(c) se muestra la gráfica correspondiente a un sistema MRA críticamente amortiguado. El caso sobreamortiguado llega lentamente al valor final. Para ciertas aplicaciones la ausencia de oscilaciones puede ser necesaria, sin embargo su lentitud puede ser intolerable. El caso subamortiguado alcanza el valor final en un tiempo menor que el sobreamortiguado pero presenta oscilaciones [8]



(a) Subamortiguado



(b) Sobreamortiguado



(c) Críticamente amortiguado

Figura 2.2: Los tres tipos de soluciones de la ecuación que modela al sistema MRA.

Solución mediante métodos numéricos

Existen distintos métodos numéricos que pueden emplearse para la solución de la ecuación diferencial del sistema MRA [28]. En esta tesis se evaluaron cuatro métodos numéricos de uso general y extendido, con la finalidad de implementar aquel que represente la mejor opción en cuanto a la relación precisión / costo computacional. Los métodos evaluados son los siguientes

1. Diferencias Finitas.

2. Euler.

3. Heun.

4. Runge Kutta de cuarto orden.

En la lista anterior se presentan ordenados con base en el *costo computacional*, siendo el método de Diferencias Finitas el que requiere menos operaciones y Runge Kutta el que más operaciones lleva a cabo. Como se muestra en la Tab. (2.1)

Método	No. de sumas	No. de productos
Diferencias finitas centrales	2M	2M
Euler	4M	4M
Heun	13M	10M
RK4	25M	20M

Tabla 2.1: Número de operaciones realizadas para M subintervalos [27].

Todos estos métodos son de un solo paso, es decir que para devolver un valor solo llevan a cabo una iteración.

2.1.3. Pruebas

Se realizaron distintas pruebas para los cuatro métodos propuestos, se varió el tamaño del paso de discretización, para variaciones grandes los métodos se volvían inestables, divergiendo. En la Fig. 2.3(a) se muestra la gráfica correspondiente a un sistema MRA de constantes $b = 1.2, k = 2.0, m = 1.0$ y $f = 15.0$. Se obtuvieron los resultados de aplicar los cuatro métodos para un rango de tiempo $t \in [0, 300]$ [segundos]. En el eje y se graficó el error acumulado que es la sumatoria de los errores en cada punto de discretización.

El error en un punto de discretización se define mediante la ecuación siguiente

$$e(t) = \hat{f} - f(t)$$

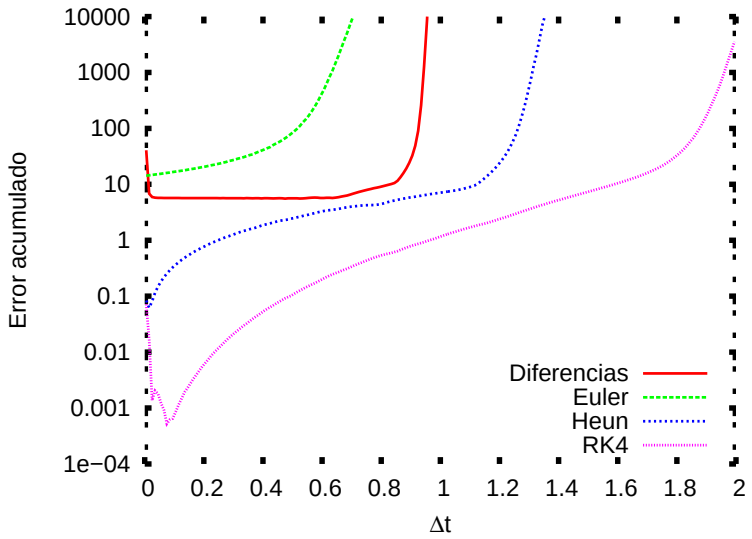
donde, $e(t)$ es el error en un punto, $f(t)$ es el valor de la solución exacta en un tiempo t y $\hat{f}$ es el valor aproximado de $f(t)$ obtenido empleando un método numérico.

El error acumulado se define por la ecuación que se muestra a continuación

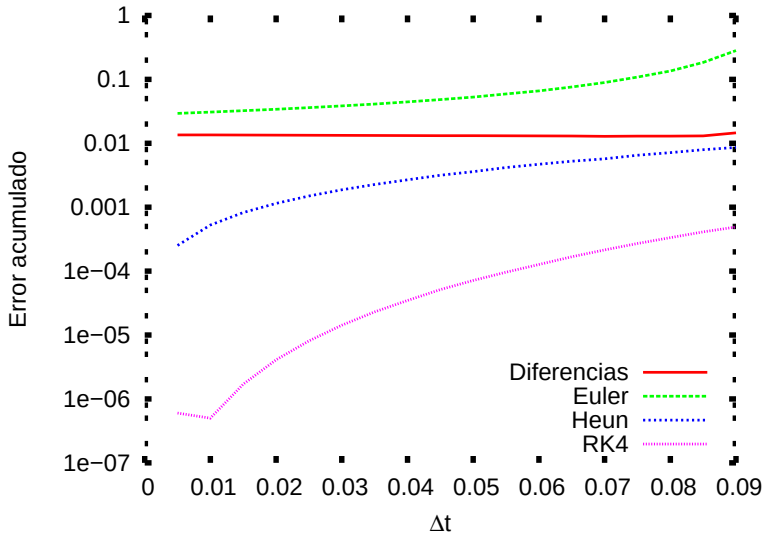
$$E = \sum_{k=0}^{k=\frac{300}{\Delta t}} e(k \cdot \Delta t)$$

donde, E es el error acumulado, Δt es el tamaño de paso de discretización o periodo de integración y $e(k \cdot \Delta t)$ es el error en $t = k \cdot \Delta t$.

La gráfica mostrada en la Fig. 2.3(b) corresponde a un sistema MRA de constantes $b = 0.01, k = 0.1, m = 0.001, f = 0$ y $x(0) = 0.02$. El intervalo de tiempo usado fue $t \in [0, 3]$ [segundos].



(a) Error acumulado para $b = 1.2$, $k = 2.0$, $m = 1.0$



(b) Error acumulado para $b = 0.01$, $k = 0.1$, $m = 0.001$

Figura 2.3: Comparación de los cuatro métodos numéricos. Error acumulado.

Como puede verse, el método de Euler es el que mayor error produce, siendo el de Runge Kutta el más exacto. En el caso de la gráfica de la Fig. 2.3(a) los cuatro métodos tienden a una línea vertical. Esa línea indica cuando los métodos divergen de la solución.

2.1.4. Modelo MRA 2

El segundo modelo que se analizó fue el que se muestra en la Fig. (2.4). El sistema

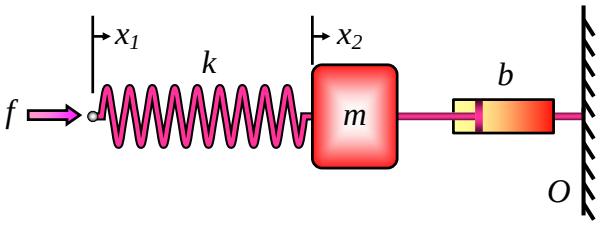


Figura 2.4: Sistema masa resorte amortiguador.

está constituido por dos nodos cuyos desplazamientos están etiquetados como x_1 y x_2 respectivamente. En cada nodo la suma de las fuerzas debe ser igual a cero [8]. En el nodo 1 se presenta la acción de dos fuerzas, f y f_k las cuales se dan en sentido contrario puesto que el amortiguador se opone a la acción de la fuerza f de modo que $f - f_k = 0$. En el nodo 2 existe la acción de tres fuerzas, f_k, f_m y f_b (la fuerza del amortiguador se considera asociada a este nodo), en este caso es la fuerza f_k la que provoca el desplazamiento, a la cual se le oponen las fuerzas f_m y f_b es decir que $f_k - f_m - f_b = 0$. De manera que se encuentra modelado por el siguiente sistema de ecuaciones

$$f = f_k = k(x_1 - x_2) \tag{2.10}$$
$$f_k = f_m + f_b = m\ddot{x}_2 + b\dot{x}_2 \tag{2.11}$$

Solución

La Ec. (2.11) es una ecuación diferencial lineal de segundo orden. Cuya solución exacta, cuando f es un valor constante, esta dada por la expresión que se muestra a continuación [41]

$$x_2 = C_1 \mathrm{e}^{-\frac{bt}{m}} + \frac{f}{b}t + C_2 \tag{2.12}$$

De modo que si se despeja x_2 de la Ec. (2.10) y se sustituye en la Ec. (2.12) se obtiene la expresión siguiente correspondiente a la solución para x_1

$$x_1 = \frac{f}{k} + C_1 \mathrm{e}^{-\frac{bt}{m}} + \frac{f}{b}t + C_2 = C_1 \mathrm{e}^{-\frac{bt}{m}} + \frac{f}{b}t + C_3 \tag{2.13}$$

Este sistema a diferencia del presentado en la Sec. (2.1.1) no tiene casos oscilatorios y su comportamiento es asintótico al término $\frac{f}{b}t$. Esto sucede porque al aplicar una fuerza

constante en cualquier dirección, la masa se desplaza sin que exista restricción alguna. Así que conforme transcurre el tiempo, la masa se desplaza alejándose del origen O.

2.2. Modelo de sistemas MRA acoplados

Como ya se mencionó, el sistema MRA se acopla en cada arista de la malla de simplejos. Así la malla se constituye de un conjunto de resortes en serie y en paralelo acoplados. Se analizarán varios casos de sistemas MRA acoplados para determinar cual es el más adecuado para el modelo deformable.

2.2.1. Sistemas MRA en serie

Se definió el modelo matemático correspondiente al sistema MRA triple en serie, el cual se muestra en la Fig. (2.5).

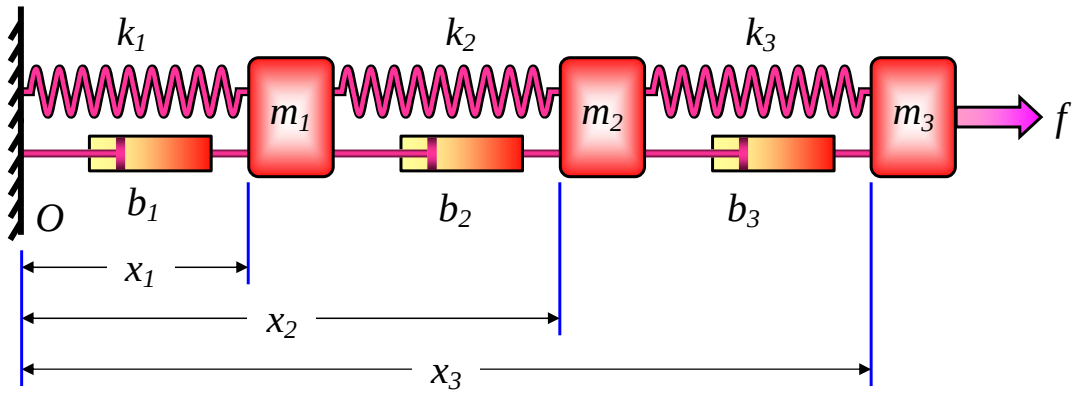


Figura 2.5: Sistema de tres resortes acoplados en serie.

La masa m_1 se encuentra bajo la acción de tres fuerzas. Provocadas por los resortes de constantes k_1 y k_2 y el amortiguador de constante b_1 (el amortiguador i se considera únicamente asociado a la masa i). La masa m_2 también se encuentra sometida a tres fuerzas, la generada por los resortes 2 y 3, así como la correspondiente al amortiguador 2. Del mismo modo la masa m_3 recibe la acción de las fuerzas debidas al resorte 3, el amortiguador 3 y la fuerza externa aplicada f.

Cada amortiguador *consume* una fuerza dada por la derivada del desplazamiento del objeto al que se encuentra asociado. En cambio los resortes generan una fuerza dada por su elongación, la cual en el caso de los resortes 1 y 2 se calcula mediante la resta de dos distancias. Esto se resume en la Tab. (2.2)

De este modo, las ecuaciones de las tres masas quedan de la forma siguiente:

Elemento	Constante	Fuerza (sin signo)
Amortiguador 1	b_1	$b_1\dot{x_1}$
Amortiguador 2	b_2	$b_2\dot{x_2}$
Amortiguador 3	b_3	$b_3\dot{x_3}$
Resorte 1	k_1	$k_1(x_1)$
Resorte 2	k_2	$k_2(x_2 - x_1)$
Resorte 3	k_3	$k_3(x_3 - x_2)$

Tabla 2.2: Fuerzas asociadas a cada elemento de la Fig. (2.5).

$$m_1\ddot{x_1} = -k_1x_1 + k_2(x_2 - x_1) - b_1\dot{x_1} \tag{2.14}$$
$$m_2\ddot{x_2} = -k_2(x_2 - x_1) + k_3(x_3 - x_2) - b_2\dot{x_2} \tag{2.15}$$
$$m_3\ddot{x_3} = -k_3(x_3 - x_2) - b_3\dot{x_3} + f \tag{2.16}$$

En la Ec. (2.14) el signo del primer término se debe a que esa fuerza se opone al movimiento, que en la Fig. (2.5) tiene como origen el plano vertical etiquetado como O. Se consideran positivos, desplazamientos hacia la derecha. Por el contrario, el segundo resorte favorece al movimiento de m_1. El amortiguador genera un término negativo pues consume fuerza. Lo mismo sucede para las demás masas. Cabe señalar que los resortes tienen una longitud inicial de cero. Si se desea considerar una longitud inicial (l_i) para cada resorte debe sustituirse cada variable de distancia, x_i por $x_i - l_i$, claro esta que esto solo afecta a los términos sin derivar puesto que como l_i es un valor constante, al derivar se elimina. Quedando el sistema como

$$m_1\ddot{x_1} = -k_1\left[x_1 - l_1\right] + k_2\left[(x_2 - l_2) - (x_1 - l1)\right] - b_1\dot{x_1} \tag{2.17}$$
$$m_2\ddot{x_2} = -k_2\left[(x_2 - l_2) - (x_1 - l_1)\right] + k_3\left[(x_3 - l_3) - (x_2 - l_2)\right] - b_2\dot{x_2} \tag{2.18}$$
$$m_3\ddot{x_3} = -k_3\left[(x_3 - l_3) - (x_2 - l_2)\right] - b_3\dot{x_3} + f \tag{2.19}$$

2.2.2. Sistema MRA en paralelo

Se definió el modelo matemático correspondiente al sistema MRA triple en paralelo el cual se presenta en la Fig. (2.6) Como es notorio, se dice que el sistema se encuentra en paralelo dado que los tres subsistemas se hallan acoplados mecánicamente en la parte final, por lo que su desplazamiento es el mismo. Bajo estas circunstancias solo existe una variable de desplazamiento, cuya solución consiste en encontrar los equivalentes de los elementos y formar un sistema equivalente.

Solución

En las figuras de los sistemas MRA mostradas hasta ahora la masa se ha colocado asociada a un rectángulo, sin embargo matemáticamente la masa se considera asociada a

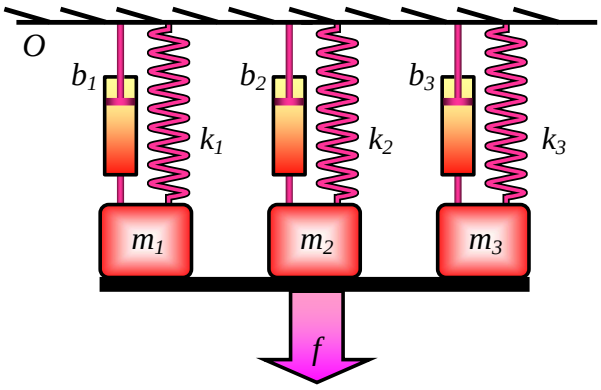


Figura 2.6: Sistema de tres resortes acoplados en paralelo.

un punto, es decir que no tiene dimensiones, de ese modo en la Fig. (2.6) el rectángulo inferior al que se encuentran ancladas las masas, y que igualmente no tiene dimensiones, únicamente sirve para indicar que las masas se encuentran acopladas y por lo tanto tienen el mismo desplazamiento de manera que es posible encontrar un elemento equivalente para las tres masas, que sería una nueva masa pero cuya magnitud será igual a la suma de las masas que se colocaron en el sistema paralelo. A su vez se sustituyen los tres resortes en paralelo por un resorte equivalente al igual que los amortiguadores. El equivalente de n resortes en paralelo está dado por un resorte cuya constante k_{eq} es igual a la suma de las n constantes de los resortes y el equivalente de n amortiguadores en paralelo es un amortiguador de constante b_{eq} igual a la suma de las n constantes de los amortiguadores conectados en paralelo [29]. En el caso de la Fig. (2.6) entonces tenemos

$$\begin{aligned} m_{eq} &= m_1 + m_2 + m_3 \\ k_{eq} &= k_1 + k_2 + k_3 \\ b_{eq} &= b_1 + b_2 + b_3 \end{aligned}$$

El sistema resultante es el que se muestra en la Fig. (2.7)

En el sistema resultante, la fuerza es la misma que la aplicada al sistema paralelo de modo que la ecuación que describe el sistema resultante esta dada por la Ec. (2.5) para los elementos equivalentes

$$m_{eq}\ddot{x} + b_{eq}\dot{x} + k_{eq}x = f \tag{2.20}$$

En el caso de tener tres sistemas iguales, es decir $k_1 = k_2 = k_3$, $b_1 = b_2 = b_3$ y $m_1 = m_2 = m_3$ el sistema tendrá un comportamiento equivalente al de uno solo de los sistemas pero con una fuerza de solo un tercio de la fuerza original aplicada al sistema paralelo.

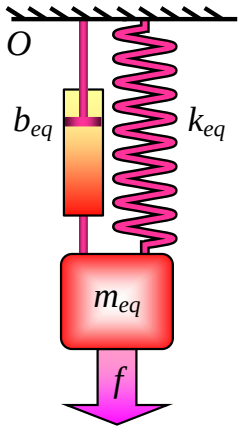


Figura 2.7: Sistema equivalente del mostrado en la Fig. (2.6).

Capítulo 3

Ecuaciones de modelado

En el sistema desarrollado la interacción del modelo deformable se lleva a cabo contra un objeto rígido que se controla por medio de un guante sensorizado. Así, el sistema se encuentra bajo la acción de una restricción unilateral. El modelo deformable no debe atravesar al objeto rígido ni debe ser atravesado por éste. Por esta razón, cada uno de los sistemas MRA puede pasar por dos estados. Uno es aquel donde se encuentra en contacto con el objeto rígido y otro donde no existe tal contacto o restricción. Cuando se da un contacto, la posición del objeto rígido es la misma que la del objeto deformable. Dado que la posición del objeto rígido es siempre conocida, la posición del objeto deformable es tambíen conocida, puesto que es la misma. De este modo, la incógnita en el modelo matemático es la fuerza que actúa en el sistema MRA que evita se atraviesen los objetos. Se decidió emplear el modelo presentado en la Sec. (2.1.1) puesto que permite modelar comportamientos elásticos. Bajo este modelo la ecuación siguiente es la que modela el sistema MRA

$$m\ddot{x} + b\dot{x} + kx = f + f_r \tag{3.1}$$

donde f_r es una fuerza que se aplica para mantener la restricción, en el caso del objeto rígido será la fuerza que este aplica evitando que el objeto deformable lo atraviese. El otro caso se presenta cuando no existe contacto. En esas circunstancias el modelo no se encuentra bajo la acción de fuerza externa alguna[1], por lo que todo su movimiento es del tipo inercial y depende de las condiciones iniciales del sistema en ese estado, es decir de los valores de posición (x) y velocidad ($\dot{x}$) justo en el momento en que se rompió el contacto anterior si es que lo hubo, en caso contrario dependerá de las condiciones iniciales en el momento de arrancar la simulación. Durante este periodo la Ec. (2.5) debe resolverse para obtener la posición del objeto a través del tiempo. Ambos casos se detallan a continuación.

[1]Es posible considerar una fuerza que se aplica sobre todo el modelo deformable, como sería el caso de considerar la acción de la gravedad por ejemplo.

3.1. Solución a las ecuaciones de modelado sin contacto

La parte más sencilla de la simulación se da cuando no existe contacto, pues la evolución del sistema se obtiene de resolver la Ec. (2.5). Ese trabajo ya fue realizado en [27] y en la Sec. (2.1.1) se detalla. Se encontró que el uso del método de diferencias finitas era el más adecuado por ser barato computacionalmente y tener un error aceptable, además de ser estable bajo los parámetros empleados en la simulación.

3.2. Solución a las ecuaciones de modelado con contacto

Para poder analizar el sistema cuando existe contacto es necesario tomar en consideración diversos aspectos del sistema mismo.

El objeto deformable simulado por el sistema que aquí se presenta se limita a ser deformado y no puede ser fracturado. De modo que el objeto sólido no debe atravesar al objeto deformable. Esta condición se cumple en el mundo real puesto que dos objetos no pueden ocupar el mismo espacio, sin embargo en una simulación es posible que eso suceda. Si se trata de un sistema continuo, en el momento en que se detecta un contacto entre el objeto rígido y el deformable se cambia el modelo matemático que se aplica de manera que no se penetren, sin embargo en un sistema discreto las posiciones de ambos objetos solo se conocen en instantes de tiempo, en un conjunto discreto de muestras del sistema, por lo que puede darse una penetración, es decir que el momento en el que se encuentran en contacto los objetos pasa desapercibido y se detecta una vez que ya se han traslapado, que se encuentran ocupando un mismo espacio. Al presentarse esta situación es necesario corregirla o bien prevenirla, calculando la posición siguiente de los objetos y corregir dichas posiciones antes de que se dé la penetración. Incluso es posible que la penetración se dé solo durante un instante de tiempo entre dos muestras del sistema, pero no en las muestras mismas. Todo esto se trata con mayor profundidad en el capítulo 5 pero es importante considerarlo en el modelo con contacto, pues en realidad es necesario tratar tres estados, cuando no existe contacto alguno, cuando existe contacto y cuando existe penetración.

Existen dos aproximaciones para resolver las ecuaciones de modelado con contacto dependiendo del sistema que se emplee para controlar el objeto rígido. Si se trata de un dispositivo háptico, es decir, uno que tiene reacción, como el Phantom Omni utilizado en [27, Cap 5], es necesario calcular la fuerza de retroalimentación que se aplicará sobre el dispositivo háptico. En caso contrario no es necesario calcular dichas fuerzas, lo cual simplifica la resolución de las ecuaciones, como se detalla a continuación.

3.2.1. Solución a las ecuaciones de modelado bajo contacto con fuerzas de retroalimentación

Un modelo propuesto para resolver el sistema cuando se encuentra en contacto o cuando existe penetración y se requiere conocer la fuerza de retroalimentación se basa en el uso de una ecuación diferencial algebraica (EDA). Esta consiste en la solución simultánea de una ecuación diferencial y una ecuación algebraica (que representa la restricción), las cuales en el presente modelo representan el siguiente sistema

$$J^T\lambda + f = m\ddot{x} + b\dot{x} + kx \tag{3.2}$$
$$\phi(x) = 0 \tag{3.3}$$

donde, $J = \frac{\partial \phi}{\partial x}$ es conocido como el Jacobiano de ϕ, $\phi(x)$ es la ecuación algebraica, λ es el multiplicador de Lagrange y representa la fuerza que mantiene el contacto sin que un objeto penetre al otro y viceversa y f es la fuerza externa a la que se encuentra sometida el objeto, como puede ser la acción de la gravedad o una acción de control exógena.

Se ha determinado que al resolver la Ec. (3.3) mediante métodos numéricos para ecuaciones diferenciales ordinarias, se obtiene un sistema inestable. La solución correcta debe ser un método que resuelva la EDA acompañado de un estabilizador de la restricción como por ejemplo el propuesto por Baumgarte [3], el cual utiliza una combinación lineal de la función y sus dos primeras derivadas. Esa combinación lineal es un sistema de control numérico para el control del error. Las ecuaciones se muestran a continuación

$$J^T\lambda + f = m\ddot{x} + b\dot{x} + kx \tag{3.4}$$
$$\ddot{\phi}(x) + K_v \cdot \dot{\phi}(x) + K_p \cdot \phi(x) = 0 \tag{3.5}$$

La Ec. (3.5) es exponencialmente estable lo que implica que $|\phi(x)| < \epsilon \ \forall \ t$ garantizando la convergencia. Sin embargo cabe destacar que las ganancias del control numérico, K_p y K_v, no son fáciles de determinar. Una mala elección puede conducir a una simulación poco realista e inestable. Típicamente estas constantes se eligen mediante pruebas, aquellas constantes que propocionen un control más estable y cuya salida sea lo más parecida a la entrada serán elegidas, sin embargo constantes que siguen la señal de entrada con poca diferencia a cierta frecuencia, no tienen un desempeño igual de bueno cuando la frecuencia cambia.

El control numérico se encarga de lidiar con la penetración en caso de que se presente. Dependiendo de las características del control, será mayor o menor la penetración permitida. Existen distintos métodos para resolver este sistema. Se ha optado por el uso de la biblioteca DASPK [1] puesto que es de código abierto y ha sido probada exitosamente. Esta biblioteca viene incluida en el programa Octave [37]. Para hacer uso de la biblioteca se emplearon dos funciones de la misma

- `daspk_options ('algebraic variables', [0,0,1]);`
- `daspk_options ('exclude algebraic variables from error test', 1)`

- `[xdae,xpdae] = daspk ('MRADAE', x0, xp0, [t(i-1),t(i)]);`

La función `daspk_options` se llama para evitar que se lleven a cabo las pruebas de error sobre las variables algebraicas, esto debido a que se obtenía un error si no se hacía de ese modo. La función `daspk` es la que propiamente realiza el cálculo numérico para resolver la EDA, recibe como parámetros el nombre de la función que evalúa el sistema que constituye la EDA, que en este caso se llama `MRADAE`; un vector con las condiciones iniciales para las variables sin derivar `x0`; un vector con las condiciones iniciales para la primer derivada `xp0`; y un vector con los tiempos en los que se desea conocer la solución.

Cabe señalar que el algoritmo empleado por DASPK es multipaso, es decir que requiere varios pasos para obtener una solución.

3.2.2. Solución a las ecuaciones de modelado bajo contacto sin fuerzas de retroalimentación

Como ya se mencionó, cuando existe contacto la incógnita en el sistema de ecuaciones es la fuerza aplicada. Cuando no es necesario conocer dicha fuerza, el sistema se encuentra resuelto, y su solución (la posición del sistema MRA) está dada por la posición del objeto rígido (recordando que la posición el objeto rígido es siempre conocida).

Sin embargo cuando existe penetración es necesario corregirla, en el caso del modelo que emplea una EDA, el control numérico es el encargado de corregir la penetración, además de proporcionar el valor de la fuerza de contacto. En este caso al no requerirse la fuerza de retroalimentación el cálculo puede simplificarse a simplemente corregir la penetración cuando se detecta. Esto es, cuando en el sistema se ha detectado que el objeto deformable y el rígido se han traslapado, se coloca el objeto deformable en una posición que no exista tal penetración. El enfoque más simple consiste en poner al objeto deformable justo sobre la superficie del objeto rígido, en contacto pero sin penetración. Este enfoque es una aproximación de la realidad, puesto que la penetración es algo imposible de suceder en el mundo real, además la penetración implica que en algún momento entre los instantes de muestreo del sistema se presentó una colisión, por lo que corregir la penetración cuando ésta ya se dio implica un retardo. Otro problema que presenta este enfoque es que no considera todos los parámetros físicos implicados en el contacto, en especial la relación entre las masas de los cuerpos en contacto. Si un objeto de cierta masa choca con otro cuya masa es significativamente menor en relación con el primero, adquirirá una velocidad grande, sin embargo, si las masas son similares la velocidad no será tan alta.

3.3. Alternación de las ecuaciones de modelado con y sin contacto

Empleando los métodos recién descritos es posible resolver el sistema en ambos casos: en contacto y sin contacto. Sin embargo hay que determinar el momento en que cada sistema aplica. Es decir cuando existe y cuando no existe contacto. Esta tarea recae

principalmente en el sistema de detección de colisiones que se presenta en el capítulo 5, el cual determina si existe o no penetración entre los objetos.

Si se trata del modelo sin fuerzas de retroalimentación, se aplica la EDO siempre que no haya penetración, cuando esta se da se aplica una etapa de corrección luego se vuelve a aplicar la EDO pero considerando condiciones iniciales, puesto que el objeto deformable empieza en una nueva posición y tiene cierta velocidad que le imprimió el objeto rígido.

En el caso de emplear el modelo con fuerzas de retroalimentación, esto es simple, se emplea la EDO siempre que no exista penetración entre los objetos, cuando esta se presenta se emplea la EDA. El sistema resultante se trata de un sistema mucho mas costoso computacionalmente que el que no requiere retroalimentación de fuerzas, esto debido principalmente a que DASPK emplea un algoritmo numérico multipaso, es decir que para cada valor el algoritmo requiere un número variable de iteraciones para proporcionar un resultado, mientras que los algoritmos para resolver la EDO solo emplean una iteración para obtener la solución aproximada.

3.4. Pruebas

3.4.1. Pruebas con fuerzas de retroalimentación

Se realizaron diferentes pruebas para caracterizar el sistema MRA, esto debido a que los valores de los elementos (m, b, k) generan diferentes comportamientos para diferentes valores. Las pruebas realizadas son de un sistema MRA en una dimensión. Se empleó una restricción de forma senoidal con respecto al tiempo, solo el semiciclo positivo con una amplitud de un centímetro, y un periodo de dos segundos (un segundo para el semiciclo positivo). Se dio un valor de un gramo a la masa, considerando que en el modelo deformable, a cada nodo se le asocia una masa m, por lo que manejando una malla de mil puntos se tendría una masa de un kilogramo, lo cual es un valor aceptable para simular objetos del mundo real. Las pruebas consistieron en graficar los comportamientos para valores de m, b y k dentro del rango $[0.001, 0.1]$. Este rango se escogió de forma arbitraria, sin embargo, para un trabajo futuro, podrían emplearse valores obtenidos de materiales reales. A continuación se muestran un par de gráficas que resultan especialmente representativas. En las gráficas de la Fig. (3.1) se emplearon los valores $m = 0.001$, $b = 0.001$ y $k = 0.001$. Las mostradas en la Fig. (3.2) usan los valores $m = 0.001$, $b = 0.01$ y $k = 0.1$. En ambas figuras se utilizaron los valores $K_v = 10$ y $K_p = 100$ los cuales se obtuvieron de igual modo mediante pruebas. En este caso se realizaron experimentos variando los valores de K_v y K_p dentro del rango $[1, 1000]$ notándose que la combinación de valores $K_v = 10$ y $K_p = 100$ proporcionaba una salida más cercana a la entrada que las otras combinaciones. Se utilizó un tamaño de paso $\Delta t = \frac{1}{30}$seg.

En cada gráfica se muestran dos curvas que indican la posición en función del tiempo, del objeto deformable y del objeto rígido o restricción respectivamente. Existe una penetración cuando la curva correspondiente al objeto deformable se encuentra por debajo de la curva del objeto rígido. Ya que este último representa una restricción al movimiento del primero, en el sentido de que no le permite tener una posición por debajo de él. Esta

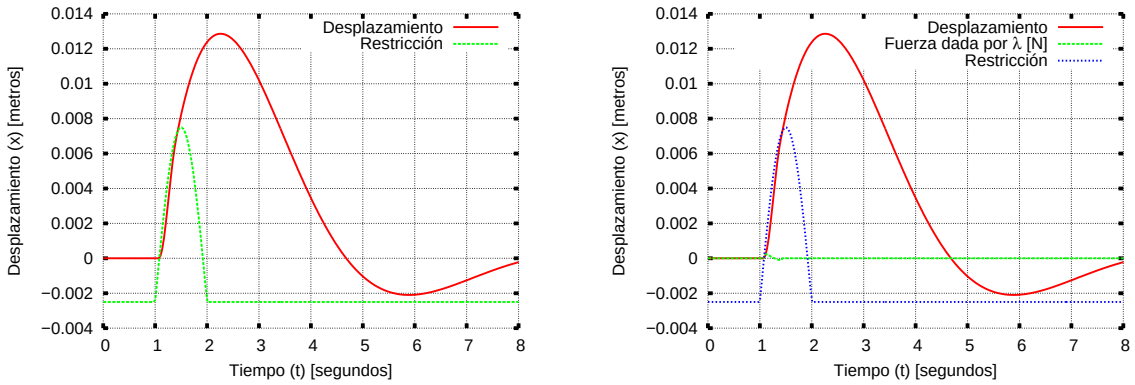


Figura 3.1: Solución para un sistema MRA unidimensional con fuerzas de retroalimentación.

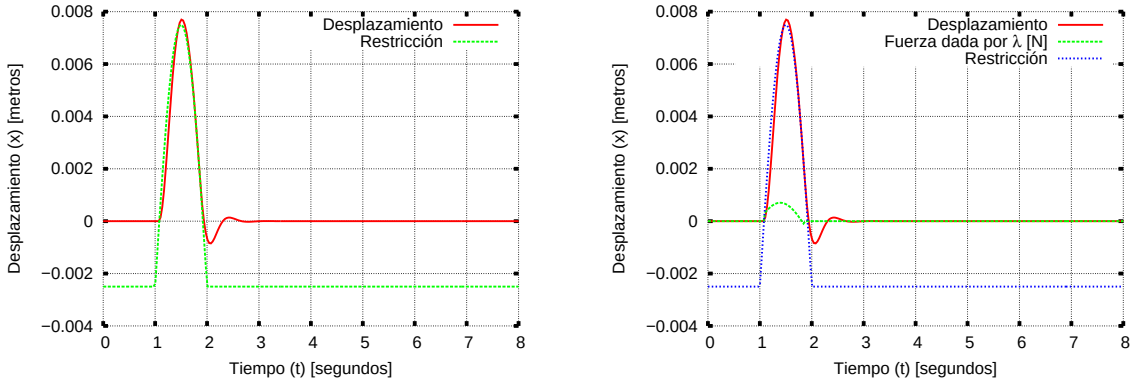


Figura 3.2: Solución para un sistema MRA unidimensional con fuerzas de retroalimentación.

restricción implica que al moverse el objeto rígido en dirección del objeto deformable lo irá empujando, evitando que se atraviesen.

En las primeras dos gráficas, que se muestran en la Fig. (3.1) se tiene un sistema más inercial que en las siguientes, ya que la acción de la restricción empuja al sistema y este tarda en recuperarse a diferencia de las otras, las mostradas en la Fig. (3.2) donde el sistema se recupera rápidamente, esto porque se aumentó el valor de amortiguamiento así como la rigidez del resorte. Nótese que los valores escogidos para m, k y b son tales que se tiene una solución subamortiguada al sistema (véase la Sec. (2.1.2) en la Pag. (11)). En estos experimentos, por simplicidad, se empleó una función del programa Octave para la solución de la EDO, aunque si bien en una implementación se realizaría empleando diferencias finitas.

3.4.2. Pruebas sin fuerzas de retroalimentación

Se realizaron las mismas pruebas que se presentaron para el modelo basado en una EDA, pero con el modelo que solo corrige la posición y que no proporciona el valor de la fuerza de retroalimentación.

Los resultados se muestran en la Fig. (3.3)

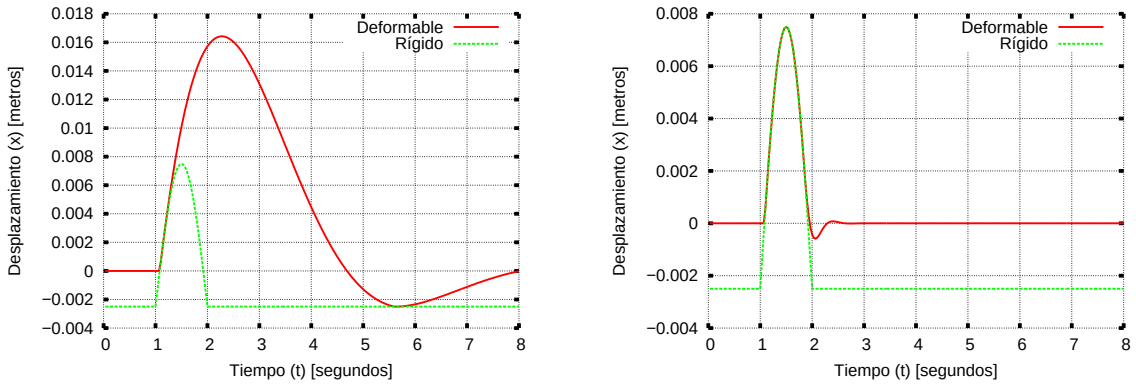


Figura 3.3: Solución para un sistema MRA unidimensional sin fuerzas de retroalimentación.

Al igual que en las gráficas mostradas en las Figs 3.1, 3.2 se emplearon los valores $m = 0.001$, $b = 0.001$, $k = 0.001$ y $m = 0.001$, $b = 0.01$, $k = 0.1$ y un paso de discretización $\Delta t = \frac{1}{30}$seg. El valor de Δt se eligió debido a que la aplicación genera 30 cuadros por segundo.

Resultan importantes dos situaciones que pueden verse de las gráficas, primero que la penetración en este caso es nula a diferencia del modelo de las Figs. 3.1, 3.2 donde se da cierto traslape. Y que este modelo es igualmente inercial, lo cual se nota puesto que la acción del objeto rígido produce un movimiento a través del tiempo del modelo deformáble que persiste aún cuando ya no existe una interacción directa. Sin embargo la forma obtenida en el modelo deformable resulta un poco diferente.

Cabe destacar también que este enfoque no requiere del cálculo de fuerzas, solamente se emplea el desplazamiento inducido en el modelo deformable por el objeto rígido.

Capítulo 4

Mallas de simplejos

4.1. Características básicas

Una malla de simplejos está definida como un conjunto de vértices y una función de conectividad [27]. En una malla de simplejos-m cada vértice se encuentra conectado a exactamente $m + 1$ vértices. En este trabajo se utilizó una malla de simplejos-2 por lo que a cada vértice se le asocian tres vértices vecinos. Para la realización de este trabajo se tomaron las construcciones de tres objetos básicos presentados en [26], la esfera, el cilindro y el toro. Estos tres modelos se muestran en la Fig. (4.1)

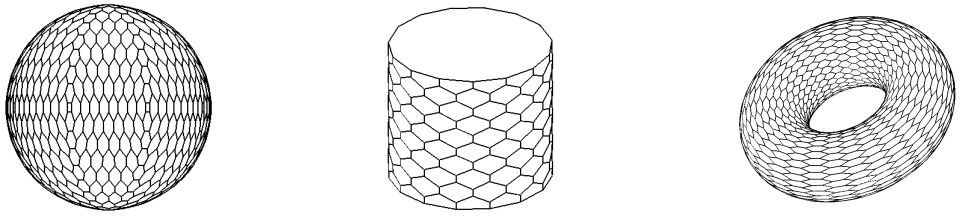


Figura 4.1: Modelos básicos representados con mallas de simplejos.

En [25] se desarrollan a profundidad los algoritmos para obtener los tres modelos básicos ya mencionados y en [27] se realizó una mejora a dichos algoritmos sin embargo ninguno de ellos permite la construcción correcta de las caras del toro.

4.2. Cálculo de las caras en una malla de simplejos

Para la creación de las caras es necesario que las aristas de todos los vértices se ordenen con respecto a la dirección de giro que se obtiene al recorrerlas (véase la Sec. (4.3)). Esto se logra definiendo un punto central de la malla y ordenando en un mismo sentido las tres aristas asociadas a cada vértice. El punto central es una referencia de la parte interior del objeto antes de que sea deformado puesto el algoritmo de ordenamiento de aristas solo se realiza una vez, al crear la malla. Sin embargo este ordenamiento falla para el toro,

puesto que para éste es necesario definir un esqueleto pues un solo punto no es suficiente para tener una referencia de la parte interna del toro.

4.2.1. Solución

Se especificó el esqueleto del toro que indica el interior del mismo, dicho esqueleto es un círculo sobre el plano $y = c$, para una constante c cualquiera. Cuyo centro tiene coordenadas (x_0, y_0, z_0). Las ecuaciones siguientes son las que determinan matemáticamente el esqueleto

$$(x - x_0)^2 + (z - z_0)^2 = r^2 \tag{4.1}$$

$$y = c \tag{4.2}$$

En la Fig. (4.2) se muestra el toro y el esqueleto, para apreciar mejor el esqueleto se muestra el toro sin una parte.

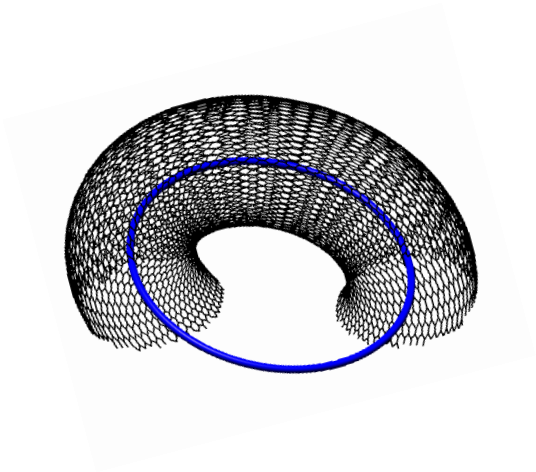


Figura 4.2: Toro y su esqueleto.

El esqueleto se emplea como una referencia del interior del toro, para llevar a cabo esto de forma práctica es necesario asociar a cada vértice de la superficie del toro un punto del esqueleto. El punto indicado para ello es el más cercano, así que se determinó qué punto de coordenadas (x, y, z) del esqueleto es el más cercano a otro punto de coordenadas (x', y', z') sobre la superficie del toro y en general para cualquier punto en el espacio tridimensional sobre el que se definió el esqueleto. Esto se llevo a cabo minimizando la función de distancia entre el esqueleto y el punto en análisis, como se muestra a continuación.

La distancia D entre ambos puntos está dada por

$$D = \left[(x - x')^2 + (y - y')^2 + (z - z')^2\right]^{\frac{1}{2}} \tag{4.3}$$

Si se despeja la Ec. (4.1) se obtiene

$$z = \pm \left[r^2 - (x - x_0)^2\right]^{\frac{1}{2}} + z_0 \tag{4.4}$$

Sustituyendo la Ec. (4.4) y la Ec. (4.2) en la Ec. (4.3)

$$D = \left\{ (x - x')^2 + (k - y')^2 + \left[\pm \left(r^2 - (x - x_0)^2 \right)^{\frac{1}{2}} + z_0 - z' \right]^2 \right\}^{\frac{1}{2}} \tag{4.5}$$

Dado que $D = f(x)$, empleando cálculo diferencial podemos encontrar los puntos críticos (máximos, mínimos y puntos de inflexión) de la función (Ec. (4.5)), derivando e igualando a cero. Para ello se tomará

$$D = (\alpha)^{\frac{1}{2}} \tag{4.6}$$

De lo anterior

$$\frac{\mathrm{d}}{\mathrm{d}x} D = \frac{1}{2\sqrt{\alpha}} \cdot \frac{\mathrm{d}\alpha}{\mathrm{d}x} = 0 \tag{4.7}$$

Lo cual implica

$$\frac{\mathrm{d}\alpha}{\mathrm{d}x} = 0 \tag{4.8}$$

$$\alpha > 0 \tag{4.9}$$

La Ec. (4.9) es una restricción que debe cumplirse para tener un resultado válido. Considerándola como satisfecha, se resuelve la Ec. (4.8)

$$\frac{\mathrm{d}\alpha}{\mathrm{d}x} = -2x' + 2x_0 \mp \frac{(z' - z_0)(-2x + 2x_0)}{\sqrt{r^2 - x^2 + 2xx_0 - x_0^2}} = 0 \tag{4.10}$$

La Ec. (4.10) implica una nueva restricción

$$r^2 - x^2 + 2xx_0 - x_0^2 > 0 \tag{4.11}$$

Considerando la Ec. (4.11) satisfecha se continúa el desarrollo. Se despeja x_c para obtener la siguiente ecuación

$$(z' - z_0)(-2x_c + 2x_0) = \mp(2x' - 2x_0)\sqrt{r^2 - x_c^2 + 2x_c x_0 - x_0^2}$$

si se elevan al cuadrado ambos términos de la igualdad anterior, se obtiene la siguiente expresión

$$\left[(z' - z_0)(-2x_c + 2x_0) \right]^2 = \left[\mp(2x' - 2x_0)\sqrt{r^2 - x_c^2 + 2x_c x_0 - x_0^2} \right]^2 \tag{4.12}$$

$$(z' - z_0)^2 (4)(x_0 - x_c)^2 = (4)(x' - x_0)^2 \left(r^2 - x_c^2 + 2x_c x_0 - x_0^2 \right)$$

expandiendo algunos términos y simplificando la ecuación anterior, se obtiene la ecuación siguiente

$$(x_c - x_0)^2 = \frac{-(z' - z_0)^2 x_0^2 + (x' - x_0)^2 (r^2 - x_0^2)}{(z' - z_0)^2 + (x' - x_0)^2} + x_0^2$$

sacando raíz cuadrada de ambos lados y despejando x_c se obtiene la siguiente expresión

$$x_c = \pm \left\{ \frac{-(z'-z_0)^2 x_0^2 + (x'-x_0)^2 (r^2 - x_0^2)}{(z'-z_0)^2 + (x'-x_0)^2} + x_0^2 \right\}^{\frac{1}{2}} + x_0$$

Al simplificar la ecuación anterior se llega a la ecuación final siguiente

$$x_c = \pm \left\{ \frac{(x'-x_0)^2 r^2}{(z'-z_0)^2 + (x'-x_0)^2} \right\}^{\frac{1}{2}} + x_0 \tag{4.13}$$

Dado que se elevó al cuadrado la expresión de la derivada igualada a cero (Ec. (4.12)), los signos de la solución encontrada no corresponden a los signos originales.

En este punto existe un camino a seguir por medio del cálculo diferencial que sería sustituir el resultado dado por la Ec. (4.13) en la segunda derivada para determinar si se trata de un mínimo, ya que hasta este momento solo se tiene la certeza de que se trata de un punto crítico. Sin embargo el significado geométrico de los resultados obtenidos hasta ahora permite una solución más simple y consistente. Para ello se reacomoda la Ec. (4.13)

$$x_c = x_0 \pm |x' - x_0| \cdot \frac{r}{\sqrt{(z'-z_0)^2 + (x'-x_0)^2}} \tag{4.14}$$

El punto bajo análisis de coordenadas (x', y', z') se proyecta sobre el plano $y = c$ que es donde se encuentra definido el esqueleto, de manera que es posible, para fines de análisis disminuir el sistema en una dimensión. Así que el significado de la Ec. (4.14) es una aplicación de triángulos semejantes, como se muestra en la Fig. 4.3(a).

En la Fig. 4.3(a) se emplean los siguientes símbolos:

r_{int} es el radio interior del toro

r_{ext} es el radio exterior del toro

r es el radio del esqueleto

El punto (x', z') se encuentra a una distancia r' del origen del esqueleto (x_0, z_0) y el punto (x, z) se ubica a su vez a una distancia r del centro. Ambos puntos se sitúan sobre la misma recta así como también el centro del esqueleto. Los triángulos semejantes son los siguientes

$$\begin{aligned} T_1 &= \{(x_0, z_0), (x', z'), (x_0, z')\} \\ T_2 &= \{(x_0, z_0), (x, z), (x_0, z)\} \end{aligned}$$

Los cuales se muestran en la Fig. 4.3(b). Dado que son triángulos semejantes, la relación que existe entre el radio del esqueleto r y la distancia del centro del circulo al punto en

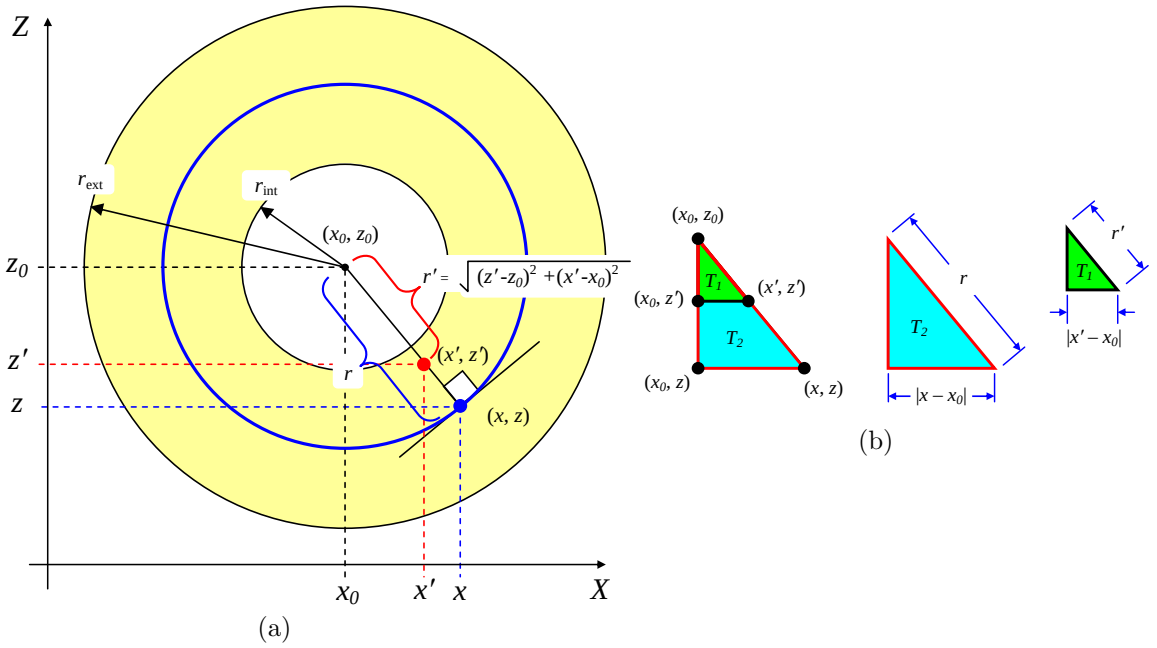


Figura 4.3: Significado geométrico de los resultados obtenidos.

análisis (r') es igual a la relación existente entre los segmentos $\overline{xx_0}$ y $\overline{x'x_0}$, que en términos algebraicos se expresa como se muestra en la siguiente ecuación:

$$\left|\frac{r}{r'}\right| = \left|\frac{r}{\sqrt{(z'-z_0)^2+(x'-x_0)^2}}\right| = \left|\frac{x-x_0}{x'-x_0}\right| \tag{4.15}$$

Si se despeja la ecuación anterior se obtiene el mismo resultado que se derivó por cálculo, con la gran ventaja de que éste enfoque permite un tratamiento intuitivo de las posibles discontinuidades, las cuales son dos:

1. Si el punto se encuentra en el origen del círculo con lo que el denominador se hace cero $\sqrt{(z'-z_0)^2+(x'-x_0)^2} = 0$. Este es un problema inherente del hecho de que lo que se busca es un mínimo y en este caso todos los puntos se hallan a la misma distancia. Si este caso se presentara habría que tratarlo como un caso especial, esto implicaría que algún punto sobre la superficie del toro pasa por el centro del esqueleto, lo cual solo sucedería si el radio interno del toro fuera cero y que además ese punto (el origen del esqueleto) fuera tomado al momento de discretizar el toro.

2. Si $x' - x_0 = 0$. Bajo estas circunstancias la Ec. (4.15) se indetermina, sin embargo la Ec. (4.14) no. Este hecho se aprovecha para usar la misma ecuación ya que si se presenta este caso el resultado será que $x = x_0$ lo cual es correcto.

La Ec. (4.14) tiene doble signo, esto es debido a que si el punto se encuentra a la derecha de x_0 (si $x' \geq x_0$) el segmento debe sumarse, en caso contrario restarse. Esto lleva a una

simplificación mayor, puesto que si $x' \geq x_0 \Rightarrow (x' - x_0) > 0$ y si $x' < x_0 \Rightarrow (x' - x_0) < 0$ es decir que el signo correcto está implícito en la resta $(x' - x_0)$ de modo que el doble signo y el valor absoluto empleado en 4.14 pueden eliminarse directamente.

El mismo enfoque que se siguió hasta aquí puede seguirse para determinar la ecuación de z correspondiente, eliminando la necesidad de evaluar el valor obtenido de x en la Ec. (4.4)

En resumen

Se tienen como datos conocidos los valores (x', y', z') que son las coordenadas del punto al que se desea encontrar el punto más cercano sobre el esqueleto del toro, (x_0, y_0, z_0) que son las coordenadas del punto central del toro y r que es el radio del esqueleto del toro, que es la distancia media entre el radio mayor y el menor del toro.
Se tiene como variables a calcular la tripleta $(x_{min}, y_{min} = y_0, z_{min})$ que son las coordenadas del punto más cercano sobre la región, $y = y_0$ es el plano sobre el que se encuentra contenido el esqueleto del toro. Las fórmulas son las siguientes

$$x_{min} = x_0 + \frac{(x' - x_0)r}{\sqrt{(z' - z_0)^2 + (x' - x_0)^2}} \tag{4.16}$$

$$z_{min} = z_0 + \frac{(z' - z_0)r}{\sqrt{(z' - z_0)^2 + (x' - x_0)^2}} \tag{4.17}$$

4.2.2. Pruebas

En la Fig. (4.4) se muestra una figura de un toro construido con mallas de simplejos. Las caras se han creado correctamente como resultado de aplicar el esqueleto, por ello puede verse la superficie del toro.

Figura 4.4: Generación de las caras.

Para cada vértice de la malla se ordenan las aristas empleando el mismo algoritmo que si existiera un único punto central, solo que en lugar de utilizar un solo punto, se emplea un punto por vértice, el punto más cercano a éste dentro del esqueleto del toro.

4.3. Cálculo de las normales para la visualización de una malla de simplejos

Se determinaron las normales correspondientes a cada vértice de la malla de simplejos. Si bien en [27] ya se realizaba el cálculo de las normales correspondientes a cada vértice de la malla, la visualización de las caras se llevaba a cabo empleando una normal para todos los vértices de la misma, con lo que la visualización era pobre, además el cálculo no consideraba el orden de los vecinos de cada vértice por lo que algunas de las normales tenían sentido inverso al requerido. El problema con la implementación de [27] es que se calculaba la proyección de P en el plano formado por sus tres vecinos y luego se calculaba el vector desde ese punto proyectado a P. Este enfoque parece correcto, sin embargo eso puede conducir a un vector con sentido opuesto al correcto, puesto que no se considera el ordenamiento de las aristas, es decir que si el ordenamiento de las aristas fuera el opuesto, lo cual se da en el caso de que el simplejo se encuentre hundido, el sentido del vector resultante sería el contrario. Para solucionar ese problema implementaron una corrección que consistía en recalcular el sentido del vector normal (la dirección se respetaba) considerando el sentido del vector que va desde el origen hacia el vértice en análisis. Con lo que el costo del algoritmo aumentaba y proporcionaba una normal incorrecta.

4.3.1. Solución

Cada vértice con sus tres vecinos puede modelarse como un tetraedro cuya base es un triángulo formado por los tres vértices vecinos, siendo el vértice bajo análisis el otro punto del tetraedro. Considerando que la base define un plano, la normal al vértice corresponde a un vector perpendicular al plano cuyo sentido queda definido por el orden de las aristas que se forman entre el vértice y los vecinos, es decir que depende de la dirección de giro obtenida al recorrer las aristas del vértice. La solución resulta relativamente sencilla puesto que al estar previamente ordenadas las aristas del vértice solo es necesario obtener dos vectores, de un vértice vecino de referencia v_r al siguiente v_{r+1} y de v_{r+1} a v_{r+2}, es decir los vectores $v_{r+1} - v_r$ y $v_{r+2} - v_{r+1}$. Con calcular el negativo del producto cruz normalizado de estos vectores se obtiene el vector normal al vértice en cuestión, véase la Fig. (4.5)

En la Fig. (4.5) se muestra el cálculo de las normales para el vértice P. En 4.5(a) se tiene el vértice con sus tres vecinos, estos se ordenan de acuerdo al número de arista al que pertenecen, de modo que formen un recorrido en sentido de las manecillas del reloj alrededor del vector que parte desde el punto central $\mathrm{P_C}$ al vértice en cuestión. En 4.5(b) se muestran los vectores calculados del vecino de referencia v_r a v_{r+1} y de este último al vértice v_{r+2}. En 4.5(c) se desplaza el vector $\overline{v_{r+1} - v_r}$ solo para fines ilustrativos, y se muestra el producto cruz negativo de los vectores, este nuevo vector corresponde al vector normal al vértice P como se muestra en la figura 4.5(d).

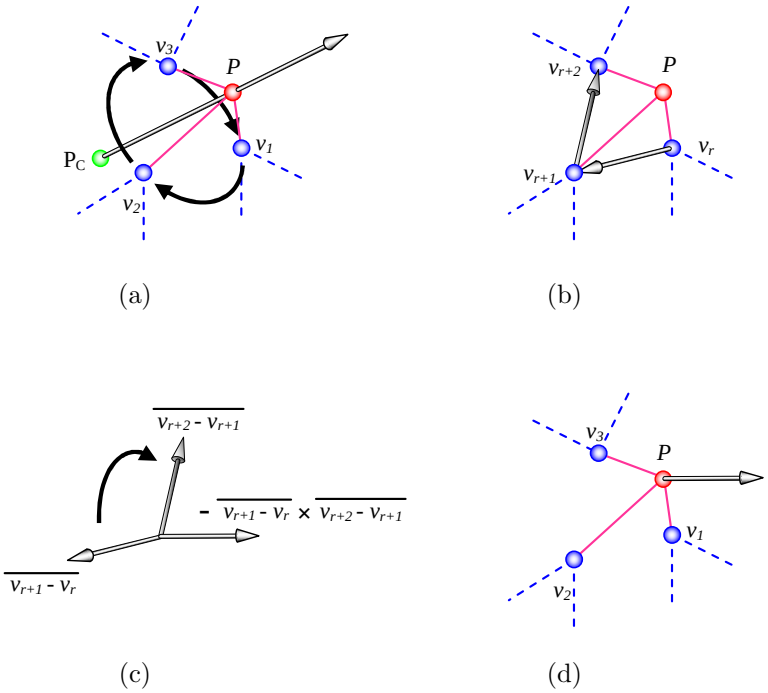


Figura 4.5: Cálculo de las normales del vértice P.

4.3.2. Pruebas

En la Fig. (4.6) se muestran los resultados obtenidos para los dos esquemas de visualización, con una normal para todos los puntos de la cara y con normales independientes para cada vértice de la malla.

Si bien es notorio el suavizado obtenido al aplicar una normal independiente para cada vértice, aún pueden notarse las aristas de la malla, esto es algo inevitable debido a que se trata de un modelo discreto, pero la visualización mejorara en la medida que se incremente la densidad de la malla.

(a) Una normal común para todos los vértices de la cara.

(b) Normales locales para cada vértice.

Figura 4.6: Comparación entre los dos enfoques de visualización.

Capítulo 5

Detección de colisiones

5.1. Panorama general

La detección de colisiones consiste en determinar si dos objetos se encuentran en contacto así como el momento y el lugar exacto donde se dio dicho contacto [10]. En algunos casos solo es necesario determinar si existe o no el contacto, en lo que respecta a este trabajo, será necesario determinar además el momento en que se dio el contacto, o bien de otro modo la profundidad de penetración del objeto, es decir qué tanto los objetos se han traslapado.

Para los fines aquí perseguidos existen dos tipos de detecciones de colisiones. Detección a través del tiempo y detección en un espacio de muestras discretas. Esta división surge del uso de modelos discretos de simulación, en los cuales solo en instantes de tiempo se conoce el estado del sistema pero no entre esas muestras. Este trabajo empleará un modelo discreto por lo que se enfocará más en el análisis de ese caso.

5.2. Detección de colisiones a través del tiempo

Si se conoce la posición de los cuerpos a través del tiempo, puede calcularse el punto de contacto y el momento en que dicho contacto se dio, de forma exacta o con una gran precisión. Pero el costo computacional regularmente resulta ser excesivamente alto. Si la geometría de los objetos y su movimiento son complejos, la detección de colisiones exacta puede resultar inviable aún para un conjunto pequeño de objetos. Por ello existen otras aproximaciones menos precisas pero más baratas computacionalmente.

Se puede emplear como base un sistema discreto que proporcione la posición de los objetos en instantes dados de tiempo y estimar la posición de los cuerpos entre esos instantes. Esto puede aún ser costoso si los objetos presentan variaciones en su velocidad durante el periodo y si rotan, lo que puede simplificarse considerando que entre cada muestra el objeto se mueve con velocidad uniforme y sin rotaciones. Incluso puede calcularse el sólido que se forma por extrusión de los objetos desde una posición, hasta la posición siguiente y calcular si existe contacto entre los sólidos formados, este enfoque

es más simple debido a que no toma en consideración los efectos de la velocidad, pero igualmente las simplificaciones pueden conducir a un sistema excesivamente impreciso, dependiendo de la aplicación en la que se le use.

5.3. Detección de colisiones en un instante de tiempo

El análisis sobre detección de colisiones que se presentará aquí se emplea para determinar si dos objetos se encuentran en contacto en un instante dado de tiempo, es decir que no importan sus posiciones anteriores o la dirección en la que se estén moviendo. Este es un enfoque típico en sistemas discretos, donde cada cierto tiempo se muestrea la posición de todos los objetos y se determina si alguno de ellos se encuentra en contacto y se realiza alguna acción en consecuencia. Sin embargo, este enfoque aproximado puede conducir a errores importantes como una colisión no detectada. Inicialmente el problema puede resolverse aumentando la frecuencia de muestreo del sistema, sin embargo es posible que aún a frecuencias altas existan colisiones no detectadas, como se muestra en la Fig. (5.1). En un sistema discreto pueden presentarse inconvenientes si la frecuencia de muestreo es pequeña en relación con la velocidad de los objetos, puesto que es posible que dos objetos se atraviesen pero exista coherencia en todas las muestras del sistema. Sin embargo, esos casos pueden desestimarse dado que para fines de visualización no existe nunca una penetración entre los objetos.

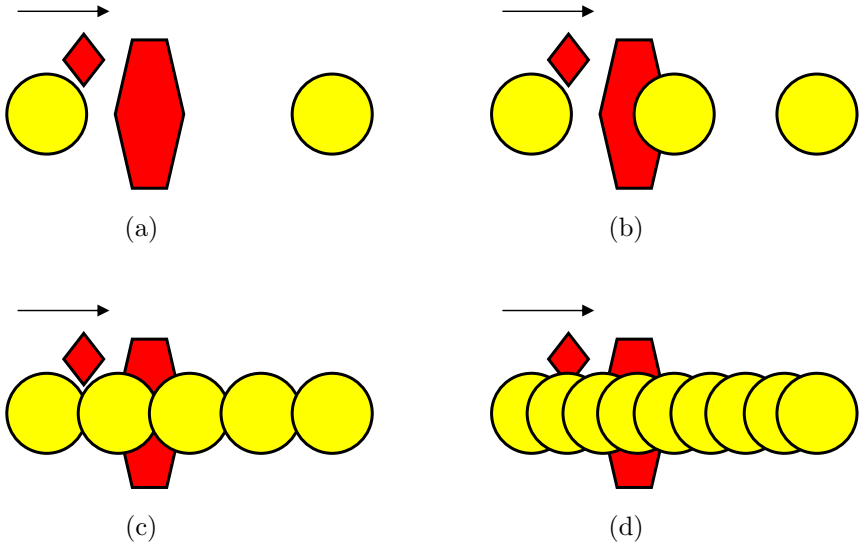
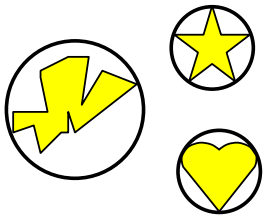


Figura 5.1: Aunque teóricamente exista una colisión, esta puede pasar desapercibida.

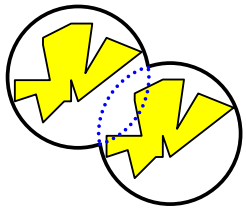
En la Fig. 5.1(a) se muestra un círculo que se mueve de izquierda a derecha, un hexágono y un pequeño rombo. Dado que el círculo se mueve a velocidad alta para la frecuencia de muestreo, la colisión del círculo con el hexágono y el rombo pasaría desapercibida, aumentando la frecuencia al doble la colisión con el hexágono sería detectada sin

embargo la del círculo contra el rombo aún pasaría desapercibida según se muestra en la Fig. 5.1(b). Aún al cuadruplicar la frecuencia de muestreo, la colisión contra el rombo no se detecta como puede verse en la Fig. 5.1(c). Al aumentar la frecuencia de muestreo ocho veces (ver Fig. 5.1(d)) ya se detectan ambas colisiones, la que se presenta contra el rombo y contra el hexágono. Es claro que la frecuencia de muestreo de un sistema completamente discreto debe ser la adecuada para evitar el mayor número de colisiones no detectadas, sin embargo bajo ciertas condiciones las frecuencias incluso altas no son suficientes para detectar todas las colisiones. Considerando que el sistema aumenta igualmente su costo computacional en la misma medida que el número de muestras por intervalo de tiempo que se tomen aumente, la cantidad de muestras debe ser el menor posible que brinde la precisión requerida. Si se requiere mucha más precisión será necesario emplear un enfoque de detección a través del tiempo.

Cuando se prueban un conjunto de objetos por colisiones entre ellos, es necesario probar cada uno contra todos los demás, tarea que tiene una complejidad computacional $O(n^2)$, por ello los sistemas de detección de colisiones buscan simplificar cada una de las pruebas que se llevan a cabo y disminuir el número de pruebas realizadas. Para este fin la detección de colisiones es dividida en dos etapas, la lejana y la cercana. La etapa lejana consiste en una serie de operaciones *sencillas* que eliminan prematuramente la posibilidad de que dos objetos estén en contacto, por ejemplo si dos objetos en 2D tienen una forma muy compleja, pueden usarse dos círculos que contengan cada uno a un objeto, de modo que si la distancia entre los dos centros de los círculos es mayor que la suma de sus radios, significa que los objetos se hallan tan lejanos que no están en contacto, de este modo la prueba por colisiones en la etapa lejana consiste simplemente en calcular la distancia entre dos puntos y compararla con un valor, lo cual es mucho más simple que emplear la geometría de los objetos originales, todo esto suponiendo que ya se han calculado los círculos, sus radios y sus centros, como se muestra en la Fig. 5.2(a)



(a) Un círculo sirve para contener figuras complejas.



(b) Aunque los círculos contenedores estén en contacto, los objetos que contienen no lo están.

Figura 5.2: Empleo de círculos contenedores.

La etapa cercana consiste en realizar los cálculos empleando la geometría original de los cuerpos, por ejemplo en el caso de la Fig. 5.2(b) los círculos están en contacto, es decir que la distancia entre sus centros es menor que la suma de sus radios, por lo que sería

necesario, ahora si probar la geometría original del objeto, lo cual es un procedimiento generalmente mucho más costoso, pero que se presenta en un número de veces pequeño[1]. Esto es necesario ya que aunque los círculos están en contacto no significa que los objetos lo están.

El ejemplo dado del círculo que contiene a los objetos es solo una posibilidad que existe para el diseño de la etapa lejana, pueden usarse otros medios como se verá en las siguientes secciones.

5.3.1. Etapa lejana

Como ya se mencionó, la etapa lejana consiste en emplear algún medio para eliminar de forma rápida y sencilla la posibilidad de la existencia de alguna colisión. Existen distintos métodos propuestos para este fin. Se analizarán dos de ellos por ser de especial importancia en la aplicación final. El primero es el uso de objetos contenedores, como el ejemplo propuesto de los círculos en la Sec. (5.1). El segundo esquema que se analizará será el de objetos contenedores jerarquizados.

Objetos contenedores

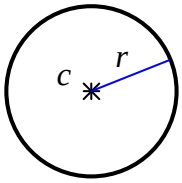
La función de los objetos contenedores es simplificar la prueba por colisiones, por ello un objeto contenedor es de geometría muy simple y contiene en su interior a uno de mayor complejidad. En algunas aplicaciones bastará con probar por colisiones los objetos contenedores para considerar que existe un contacto, en otras aplicaciones donde se requiere mayor precisión, una vez que se ha detectado una colisión de objetos contenedores se procede a probar por colisiones los objetos internos de geometría compleja. El objeto contenedor más simple es el círculo (o la esfera en 3D), pues requieren poca memoria para su almacenamiento y la prueba por colisiones entre dos círculos es rápida. Sin embargo su uso no permite una ajuste fuerte para ciertos objetos, como los ejemplos mostrados en la Fig. 5.3(b)

Otro objeto comúnmente empleado como contenedor es la *caja alineada con los ejes* (CAE), este consiste de un rectángulo (o un paralelepípedo rectangular en 3D) cuyas caras son paralelas a los ejes, como se muestra en la Fig. 5.4(a)

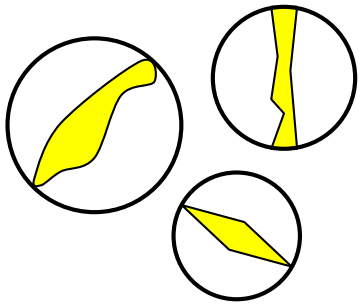
Al igual que la esfera existen casos en los que una CAE no se ajusta fuertemente al objeto que contiene como se ejemplifica en la Fig. 5.4(b).

Una variante de la CAE es la *caja orientada* (CO) que también es un rectángulo, pero de orientación arbitraria. La CO se ajusta mejor que la CAE pero su construcción resulta más costosa y si el objeto que contiene se mueve y más aún se deforma, es necesario reconstruirla en cada cambio del objeto contenido.

[1] Claro está que en caso de que esto último no fuera cierto, y los objetos estuvieran siempre o una gran parte del tiempo tan cerca que fuera necesario emplear la etapa cercana, la implementación de la etapa lejana sería inútil y solo acarrearía una disminución de la eficiencia del sistema.

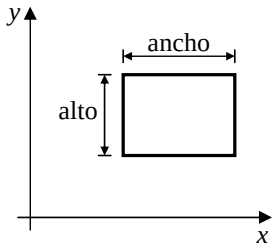


(a) El círculo se define por el centro y el radio.

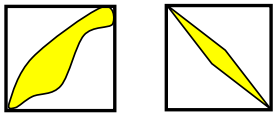


(b) Para ciertos objetos, el círculo se haya casi vacío, con lo que **no** se tiene un ajuste fuerte.

Figura 5.3: Círculos contenedores.

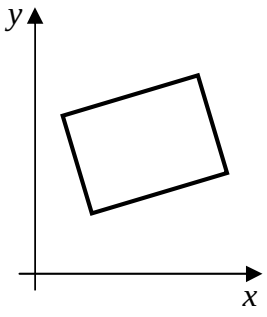


(a) La CAE tiene sus caras paralelas a los ejes.

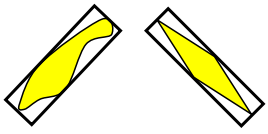


(b) CAE con ajuste débil.

Figura 5.4: Caja alineada con los ejes CAE.



(a) La CO es un rectángulo rotado arbitrariamente.



(b) La CO logra, generalmente un ajuste más fuerte que la CAE.

Figura 5.5: Caja orientada CO.

Objetos contenedores jerarquizados

El uso de objetos contenedores simplifica cada prueba por colisiones que se realiza sobre los objetos contenidos. Sin embargo la complejidad del algoritmo sigue siendo $O(n^2)$ puesto que se siguen probando cada objeto contra todos los demás. Con el fin de disminuir el número de pruebas realizadas existe una técnica llamada objetos contenedores jerarquizados, bajo la cual se emplea una estructura de árbol par almacenar los objetos contenedores. En las hojas del árbol se colocan los objetos que contienen directamente a los objetos de la escena, y en los nodos se colocan objetos contenedores que contienen a los objetos contenedores hijos, un ejemplo se muestra en la Fig. (5.6) para una jerarquía de CAES.

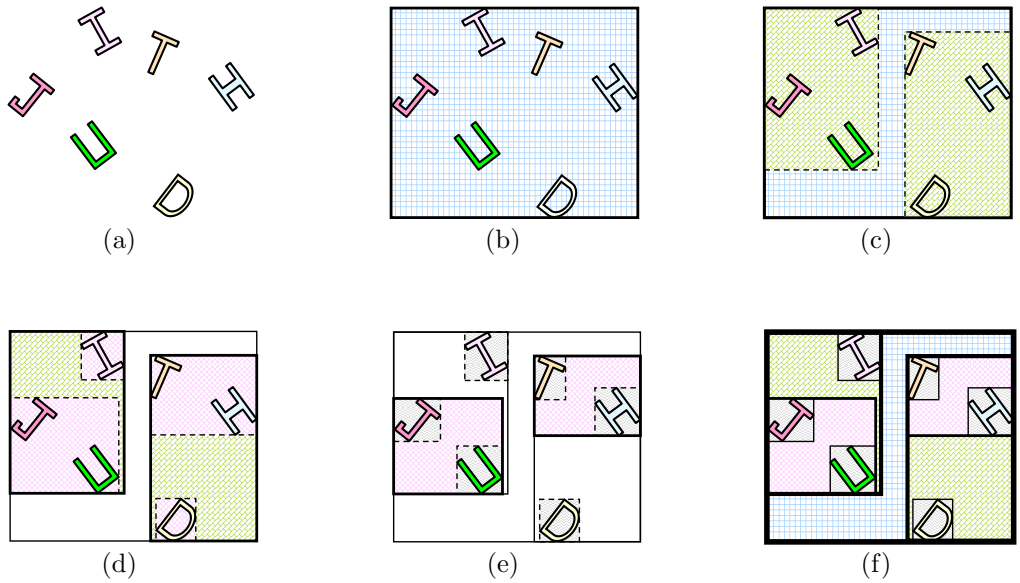


Figura 5.6: Construcción de una jerarquía de CAES. En (a) se tienen los objetos a probar por colisiones. En (b) se coloca una CAE que contiene a todos los objetos la cual se divide recursivamente de (c) a (e) donde ya se tiene una CAE por objeto. En (f) se muestran todas las CAES de la jerarquía.

En Fig. 5.6(a) se tienen seis objetos dispersados en cierta área, el primer paso para crear una jerarquía de objetos contenedores consiste en colocar todos los objetos en un objeto contenedor, una CAE en este caso. En Fig. 5.6(b), Fig. 5.6(c), Fig. 5.6(d) y Fig. 5.6(e) se realiza un proceso de división de las CAES más grandes en otras pequeñas (y normalmente con menos objetos) hasta que en Fig. 5.6(f) se tiene una CAE para cada objeto, el árbol que especifica la jerarquía formada se muestra en Fig. (5.7)

En el árbol, las hojas (los nodos sin hijos) son las CAES que contienen a un solo objeto, el nodo padre se asocia a la CAE que contiene a las dos CAES hijas como se representó en la Fig. 5.6(f). De modo que la raíz del árbol referencía a la CAE que contiene tanto a todos los objetos de la escena como a todas las CAES más pequeñas.

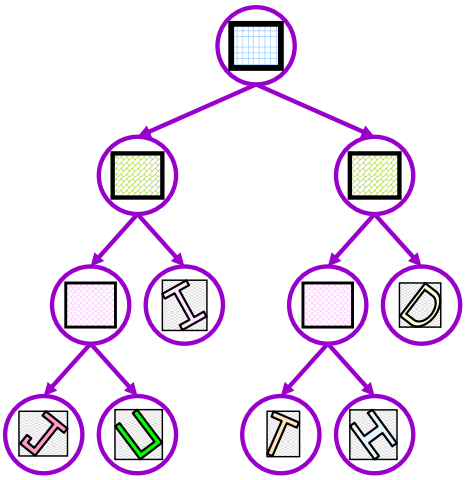


Figura 5.7: Árbol correspondiente a la Fig. (5.6). La raíz es la CAE más grande y las hojas son las caes que contienen a solo un objeto.

5.3.2. Etapa cercana

En la etapa cercana se prueban los objetos que superaron la etapa lejana. Cuando la complejidad del objeto es grande, este se descompone en objetos más simples, típicamente triángulos, para los cuales ya existen algoritmos eficientes para prueba por colisiones. De modo que esta etapa es considerablemente más costosa que la lejana y por ello se busca evitar en la mayoría de los casos.

5.4. Detección de colisiones en mallas de simplejos

Una característica de las mallas de simplejos es que se definen por medio de un conjunto de vértices y una función de conectividad entre ellos, la cual indica cuales son los vértices vecinos de cada vértice de la malla. Esta representación discreta no puede visualizarse directamente ni puede usarse para detección de colisiones puesto que solo se tendrían puntos, de modo que se hace necesario definir una superficie asociada a la malla, tanto para la visualización como para la detección de colisiones. Ambos modelos, tanto el de visualización como el empleado para detección de colisiones no tienen que ser el mismo sin embargo debe existir una coherencia entre ellos. En el trabajo [27] el modelo de visualización de una malla de simplejos emplea la biblioteca gráfica OpenGL [35] de la cual utiliza la opción `GL_POLYGON` para mostrar las caras de la malla, esta opción sirve para visualizar polígonos en un espacio tridimensional. Este enfoque no produce una superficie bien definida puesto que dicha opción solo debe ser utilizada para polígonos simples y convexos [22]. Que no es el caso de las caras generadas en una malla de simplejos, no obstante que los resultados visuales obtenidos son satisfactorios. De manera que no es posible emplear ese esquema de visualización para detección de colisiones, por lo que a continuación se presentarán algunas posibilidades para definir la superficie asociada a una

malla de simplejos.

5.4.1. Triangulación de las caras

Dado que el modelo basado en mallas de simplejos es deformable, no es posible poner restricción alguna en la geometría de las caras, es decir que estas pueden tomar cualquier forma. Por ello la triangulación de las caras resulta una opción adecuada puesto que la posibilidad de tener una representación errónea es pequeña, y puede dársele una solución en caso de que eso se presente. Un triángulo tiene tres casos en los que degenera:

- Los tres puntos son colineales. En este caso el triángulo degenera en un segmento de recta.
- Dos puntos del triángulo son iguales (tienen las mismas coordenadas). El triángulo degenera en un segmento de recta.
- Los tres puntos del triángulo son iguales (tienen las mismas coordenadas). El triángulo degenera en un punto.

En estos casos el triángulo ya no lo es más, esto implica que en la detección de colisiones se deban considerar estas posibilidades. Ya que, por ejemplo no es posible asociar un único plano al triángulo en caso de que haya degenerado a un segmento de recta, de igual manera no será posible calcular una normal al mismo.

Si los casos anteriores son considerados en el sistema de detección de colisiones el empleo de triángulos para definir la superficie de contacto no representa ningún problema. Además es el caso típico, de modo que cualquier sistema deberá implementar las validaciones necesarias para esos casos.

Ahora bien es necesario definir como se triangularán las caras. En las secciones siguientes se presentan un par de posibilidades.

Primer esquema de triangulación de las caras

Un esquema sencillo para triangular las caras consistirá en tomar un vértice como punto de referencia para todos los triángulos, luego tomar los dos siguientes para formar un triángulo. Los triángulos siguientes se forman con el vértice de referencia, el vértice anterior y el nuevo, como se muestra en la Fig. 5.8(a). Este esquema es conocido como abanico de triángulos.

Dado que la estructura empleada para almacenar las caras es una lista de vértices, es posible utilizar el primero de ellos como el vértice de referencia. Los siguientes se toman de forma consecutiva de la lista.

Esquema de triangulación de las caras empleando su centroide

El centroide de una cara de la malla de simplejos se define como el promedio de las coordenadas de todos los vértices que la forman. En un esquema muy similar al empleado

en la sección anterior se puede tomar el centroide de la cara como el vértice común a todos los triángulos, como se muestra en la Fig. 5.8(b)

El inconveniente de emplear este enfoque es que al hacer contacto con alguno de los triángulos, será necesario empujar toda la cara para poder trasladar el centroide a su nueva posición. Sin embargo es posible utilizar este mismo modelo para la visualización colocando la normal de la cara como la normal del centroide, obteniendo una visualización correcta.

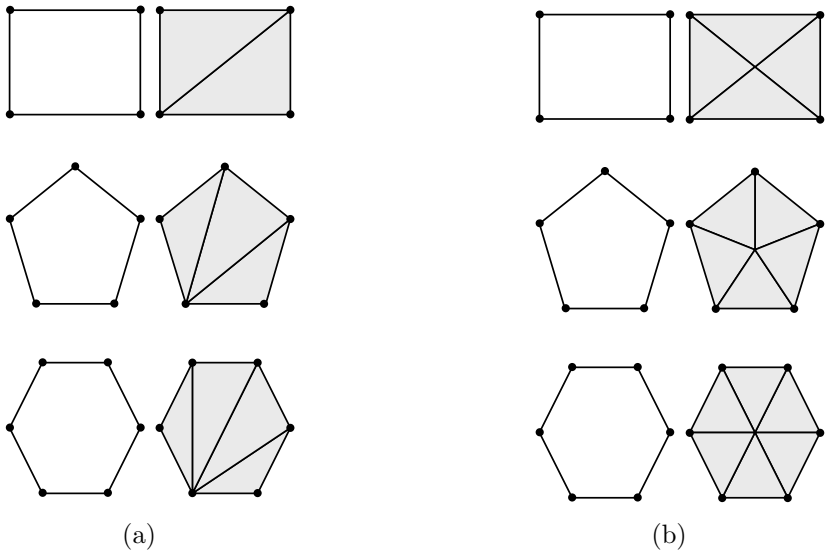


Figura 5.8: Esquemas de triangulación.

5.4.2. Superficie del objeto rígido

Considerando lo mencionado en las secciones anteriores, la superficie del objeto deformable se representará empleando triángulos. El objeto deformable interactúa con un grupo de cinco esferas controladas por el guante sensorizado. Es decir que las colisiones se detectan entre los triángulos que conforman la superficie del modelo deformable y las esferas. Las esferas son objetos de geometría muy sencilla y que permiten una prueba por colisiones rápida. Estas esferas se consideran completamente rígidas por lo que no se deforman.

5.4.3. Complejidad del algoritmo

Existe una consideración especial en lo referente a la complejidad de la detección de colisiones en una malla de simplejos. Las colisiones se detectan entre el objeto rígido y la malla de triángulos que conforma la superficie del objeto deformable, pero no entre ellos, es decir que no se detectan colisiones de un objeto rígido contra los otros objetos

rígidos, como tampoco se detectan colisiones entre un triángulo y los demás triángulos. Considerando que existen p triángulos de la superficie del objeto deformable y q objetos rígidos, un algoritmo de búsqueda exhaustiva realizaría $p \cdot q$ pruebas. Considerando que cada prueba tenga un costo de B operaciones, el costo total de una prueba por colisiones será $\mathrm{B}p \cdot q$

En el caso de emplear un algoritmo de detección de colisiones que emplee alguna jerarquización para disminuir el número de pruebas en la etapa cercana, en cada iteración sería necesario reordenar los objetos, el costo computacional de ordenar n elementos de un algoritmo de ordenamiento genérico es $O(n \log n)$ [6]. Así que para el caso presente la etapa lejana tendrá una complejidad computacional de $O\left((p+q)\log(p+q)\right)$. Además, se tienen cierto número de búsquedas sobre el conjunto de objetos previamente ordenado las cuales tienen un costo de $O(\log n)$. Sin embargo la notación O es una aproximación, que indica que el costo real del algoritmo será de $\mathrm{C}(p+q)\log(p+q)$ para alguna constante C que depende del algoritmo de ordenamiento empleado, esto claro despreciando el costo de las búsquedas o bien considerándolo dentro de la constante. Dado que $p \gg q$ (se tienen varios miles de triángulos mientras a lo más se trabajará con cinco objetos rígidos) puede hacerse la siguiente aproximación $p+q \approx p$ de modo que el costo computacional bajo esta aproximación es $\mathrm{C}p\log p$. Así, la relación existente entre las cantidades $\mathrm{B}p \cdot q$ y $\mathrm{C}p\log p$ marcará cual de los dos esquemas es el más eficiente. Si $\mathrm{B}q > \mathrm{C}\log p$ requerirá menos procesamiento calcular la detección empleando un esquema que implemente la etapa lejana. Por el contrario si $\mathrm{B}q < \mathrm{C}\log p$ será más eficiente utilizar una búsqueda exhaustiva.

5.5. Biblioteca SOLID

Una vez que se ha definido una superficie de contacto para la malla de simplejos y para el objeto rígido se requiere utilizar un sistema de detección de colisiones para dichas superficies. Se decidió utilizar la biblioteca SOLID (Software Library for Interference Detection) [38] ya que ésta se encuentra ampliamente optimizada y tiene varios años en desarrollo por lo que la versión actual es estable y eficiente.

SOLID emplea CAES y objetos contenedores jerarquizados para la etapa lejana, la etapa cercana se lleva a cabo entre los triángulos definidos para la superficie de la malla de simplejos y la superficie del objeto rígido.

SOLID requiere se especifiquen los objetos a probar por colisiones empleando su API, en la cual se cuenta con diversos tipos de objetos divididos en dos categorías, objetos simples y objetos complejos. Además se tiene la posibilidad de definir un objeto como la cubierta convexa de otros objetos y también puede especificarse un objeto como la suma de Minkowski de dos objetos, la cual expande un objeto empleando el otro, por ejemplo si se aplica la suma de Minkowski a una línea y una esfera el resultado será un objeto con la forma de una salchicha.

Para la especificación de triángulos debe usarse la función `DT_NewComplexShape` (utilizada para definir objetos complejos) indicando los vértices que los forman. Los vértices deben guardarse en un arreglo de estructuras en el programa cliente y son accedidas di-

rectamente por SOLID. Esto trajo un problema con la forma en la que se almacenaban los vértices en la aplicación de [27], la cual guarda los valores de los vértices en estructuras que forman los nodos de una lista doblemente enlazada, de modo que el área de memoria no es contigua. Para solucionar este problema se programó una función que, una vez creada la lista, reserva en memoria un arreglo del tamaño adecuado para almacenar la lista, luego vuelca todos los datos de la lista en el arreglo, y ordena los apuntadores de cada estructura para que el arreglo puede manejarse tanto como lista doblemente enlazada o como arreglo; liberando finalmente el espacio de memoria ocupado por la lista original. Una vez especificada la geometría del objeto se debe crear una instancia del mismo empleando la función `DT_CreateObject`. Para el caso de otros objetos como esferas por ejemplo, no es necesario guardar vértices por lo que su especificación es estática, ya que los valores pasados a SOLID (radio y coordenadas del centro) son constantes o al menos así lo toma SOLID. Para indicar que una esfera se ha movido debe emplearse la función `DT_SetPosition` especificando la nueva posición en la que se encuentra la esfera, también pueden aplicarse transformaciones de escalamiento y rotación. Una vez especificados los objetos solo deben emplearse las funciones para pruebas por colisiones, además de avisar a SOLID cada vez que se deforma el objeto. Las deformaciones consisten en simplemente modificar los datos de los vértices, o bien emplear alguna transformación. En este trabajo no se emplearon las transformaciones de SOLID. SOLID, además de indicar si existe o no contacto entre dos objetos, puede calcular algunos datos para emplearlos en la respuesta a la colisión. En este trabajo se utilizó el vector de penetración como dato a calcular para una colisión.

5.5.1. El vector de profundidad de penetración

La profundidad de penetración es el vector más corto necesario para trasladar uno de los dos objetos que se encuentran traslapándose para ponerlos en contacto sin penetración.

Tal como se señalo en la Sec. (3.2.2), cuando dos objetos se traslapan es necesario corregir esa situación. El uso del vector de profundidad de penetración (VPP) es una manera eficiente de corregir la penetración de dos objetos pues nos garantiza que, una vez trasladado uno de ellos con el VPP, se encontraran ambos objetos sin penetración, además el VPP es un parámetro calculado por SOLID. SOLID calcula el VPP como dos puntos, uno sobre la superficie de cada objeto. De este modo, cuando se detecta una colisión entre el objeto rígido y algún triángulo del modelo deformable, el triángulo se desplaza sumando el VPP a sus tres coordenadas.

Sin embargo el VPP solo una aproximación, que puede llevar a simulaciones poco realistas. Por ejemplo, el VPP puede servir para saber, de forma aproximada en que dirección y sobre que punto se debe deformar un modelo deformable, para lo cual fallaría el caso presentado en la Fig. (5.9), puesto que el VPP obtenido es horizontal sin embargo la acción es vertical.

En la Fig. 5.9(a) se tienen dos rectángulos, uno de ellos se mueve hacia arriba (como lo indica la flecha que contiene en su interior) mientras el otro permanece estático. En la Fig. 5.9(b) los cuadrados ya se han traslapado, en este caso la dirección en la que los

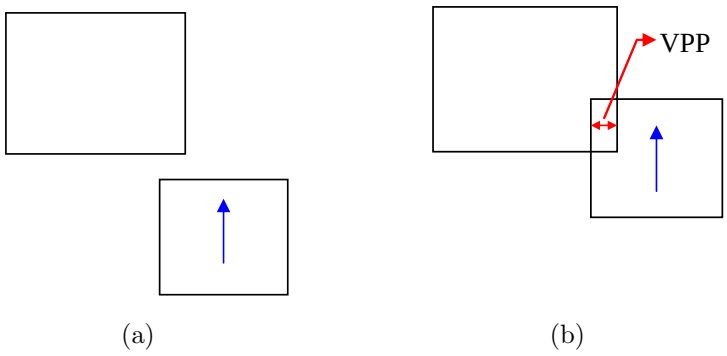


Figura 5.9: Ejemplos en los que el VPP causa una mala simulación.

cuadrados chocaron es vertical sin embargo el VPP calculado es horizontal. Este problema surge principalmente de que el uso del VPP no toma en consideración las condiciones en el momento exacto donde se presenta el contacto antes de que los objetos se traslapen, como lo son la dirección relativa de los objetos y el punto exacto donde la colisión se dio.

Dado que el VPP se empleará en una malla de simplejos cuya superficie se definió a través de una malla de triángulos, la detección de colisiones se realiza sobre cada triángulo, y se garantiza que una vez *aplicado* el VPP, el triángulo no estará en contacto con el objeto rígido, sin embargo existe una consideración especial puesto que no se puede garantizar que los triángulos que comparten vértices con el triángulo movido bajo el VPP también queden fuera del objeto rígido. Al modificar los vértices del triángulo con el que se hace contacto también se modifican todos los triángulos que comparten algún vértice. Y esto puede provocar que los otros triángulos se traslapen con el objeto rígido.

Los casos en los que el VPP es causa de una simulación pobre pueden desestimarse para así tener un sistema económico computacionalmente, sin embargo si se requiere una mayor precisión entonces el VPP puede ser una mala elección. En ese caso podría emplearse otro enfoque que busque estimar el punto y el momento de contacto. En el capítulo 6 se detalla un enfoque que implementa fricción en la resolución del contacto, la cual daría mayor realismo al sistema, independientemente de si se usa el VPP o algún otro vector.

El algoritmo 1 muestra como se realiza el proceso de detección de colisiones empleando

una estructura de CAES jerarquizadas.

```
   Entrada: Triángulos que forman la malla y los objetos rígidos.
 1 begin
 2 |  Inicialización;
 3 |    Creación de la jerarquía de CAES;
 4 |  Movimiento de los objetos;
 5 |  foreach CAE de la jerarquía (comenzando por la raíz) do
 6 |  |  if Se trata de la CAE en una hoja then
 7 |  |  |  Se prueban directamente los objetos contenidos por la CAE;
 8 |  |  |  if Se traslapan los objetos then
 9 |  |  |  |  Cálculo del VPP;
10 |  |  |  |  return COLISION;
11 |  |  |  end
12 |  |  else if Existe traslape entre las CAE then
13 |  |  |  Continuar con las dos CAES hijas;
14 |  |  end
15 |  end
16 end
```

Algoritmo 1: Algoritmo de detección de colisiones

Capítulo 6

El motor de deformación

El motor de deformación es el encargado de calcular la deformación del modelo a través del tiempo, ya sea por la acción del objeto rígido o bien por inercia de interacciones pasadas.

6.1. El sistema MRA en mallas de simplejos

El sistema físico MRA es el que le da la capacidad de deformación al modelo. Este sistema se usa sobre la malla de simplejos como puede apreciarse en la Fig. (6.1).

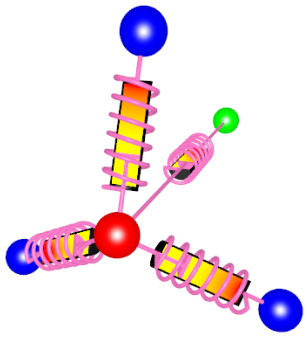


Figura 6.1: Vértice con acoplamientos a sus vecinos y al centro de la malla. El amortiguador se muestra en el interior de los resortes. Para fines ilustrativos se muestra cortado para poder ver su interior.

Visto de una manera simple, cuando algún vértice de la malla se mueve de su posición, ejerce una acción sobre sus vértices vecinos (los jala o empuja) y estos a su vez hacen lo propio con sus vecinos, el resultado es una propagación del movimiento que teóricamente llega a todos los puntos de la malla. En la práctica, dado que los amortiguadores disipan rápidamente la propagación del movimiento y debido a que la acción se divide entre los resortes, movimientos pequeños solo producen un movimiento visible (mayor o igual a un píxel) en una vecindad relativamente pequeña (solo un par de niveles de profundidad) y en niveles más profundos también se produce un movimiento pero dada su magnitud es

imperceptible. También debe considerarse que esta situación depende de los parámetros de la malla, por ejemplo si la constante del resorte es muy pequeña la propagación de la fuerza será mínima, o si se emplean constantes variables es posible que se requiera de una fuerza mínima para producir propagación, sin embargo, para fines de análisis en este trabajo puede considerarse como el caso general aquel donde se presenta propagación de la fuerza, y ya en caso necesario se tomará como un caso particular aquel donde la propagación del movimiento no se presente.

El modelo de deformación debe resolverse numéricamente, es decir se deben calcular las posiciones de todos los vértices de la malla en base a la fuerza externa aplicada y la fuerza dentro de la malla debida a la inercia del modelo (interacción con la malla en tiempo anterior). Todo esto en tiempo real.

6.1.1. Cálculo de la deformación

Todo el desarrollo presentado en el capítulo 3 en la Pag. (21) es para un sistema unidimensional sin embargo el objeto modelado es tridimensional así como los resortes que componen su sistema de deformación. De modo que fue necesario extender el sistema a tres dimensiones. En [27] se maneja un modelo de deformación tridimensional el cual se basa en la variación de la longitud de cada resorte. Después de crear la malla de simplejos se crea una lista de estructuras llamada `motor de deformación` la cual tiene tantos nodos como vértices tiene la malla de simplejos, en cada nodo de la lista se almacena la longitud inicial de los tres resortes que asocian al punto con sus vecinos, también se almacenan las coordenadas iniciales del punto entre otros valores. Para calcular la deformación de un punto (P_i) proponen calcular la fuerza interna (la fuerza asociada a cada resorte) por medio de la siguiente ecuación:

$$F_{int}(i) = -k_r \Delta l_i \left[\frac{l_i}{||l_i||}\right] - \sum_{j=1}^{3} k \Delta l_{N_j(i)} \left[\frac{l_{N_j(i)}}{||l_{N_j(i)}||}\right] \tag{6.1}$$

donde, $-k_r \Delta l_i \left[\frac{l_i}{||l_i||}\right]$ es la fuerza interna del resorte de constante k_r conectado hacia el centro de la malla, $\Delta l_i = ||l_i - L_i||$, $l_i = P_i - P_{0i}$, L_i es la longitud inicial del resorte, P_{0i} es la posición inicial del vértice P_i, $-\sum_{j=1}^{3} k \Delta l_{N_j(i)} \left[\frac{l_{N_j(i)}}{||l_{N_j(i)}||}\right]$ es la fuerza interna de los resortes de constante k conectados a los vecinos del punto i, $\Delta l_{N_j(i)} = ||l_{N_j(i)} - L_{N_j(i)}||$, $l_{N_j(i)} = P_i - P_{N_j(i)}$, $L_{N_j(i)}$ es la longitud inicial del resorte conectado del vértice i al vecino j que está dada por $P_0(i) - P_{0N_j(i)}$ y $P_{0N_j(i)}$ es la posición inicial del vecino j-ésimo del vértice i.

La Ec. (6.1) es el método presentado en el documento [27], sin embargo no coincide con la forma en que se realiza el cálculo en la aplicación desarrollada como parte del mismo trabajo. La diferencia radica en ambos términos, considerando esta discrepancia y dado que la ecuación se resuelve para coordenadas, pueden expresarse tres ecuaciones

$$
\begin{aligned}
F_{int}(i)_x &= -k_r l_{ix} - \sum_{j=1}^{3} k\left(||l_{N_j(i)}|| - ||L_{N_j(i)}||\right)\left[\frac{l_{N_j(i)x}}{||l_{N_j(i)}||}\right] \\
F_{int}(i)_y &= -k_r l_{iy} - \sum_{j=1}^{3} k\left(||l_{N_j(i)}|| - ||L_{N_j(i)}||\right)\left[\frac{l_{N_j(i)y}}{||l_{N_j(i)}||}\right] \\
F_{int}(i)_z &= -k_r l_{iz} - \sum_{j=1}^{3} k\left(||l_{N_j(i)}|| - ||L_{N_j(i)}||\right)\left[\frac{l_{N_j(i)z}}{||l_{N_j(i)}||}\right]
\end{aligned}
\tag{6.2}
$$

Estas tres ecuaciones además de representar un costo computacional importante, pues se calculan una raíz cuadrada, tres elevaciones al cuadrado y una división, por cada componente, producían una simulación poco realista. Un aspecto importante que presentan es el término $||l_{N_j(i)}|| - ||L_{N_j(i)}||$, que expresado en palabras es la resta de la longitud actual del resorte menos la longitud inicial del mismo. Sin embargo la longitud del resorte puede permanecer constante y aún así tenerse un movimiento, como se muestra en la Fig. (6.2)

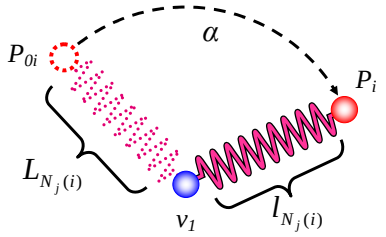


Figura 6.2: Ejemplo de un resultado anómalo empleando el enfoque de deformación de [27]. A pesar de haberse movido el vértice i, la longitud del resorte no varió $\left(||l_{N_j(i)}|| = ||L_{N_j(i)}||\right)$, por lo que el sistema funciona como si no existiera el resorte.

Dado que el vértice giró α grados, pero la longitud del resorte asociado no varió todos los valores de F_{int} se hacen cero. Esto es debido a que los sistemas MRA solo se ven afectados cuando se realiza un esfuerzo longitudinal sobre ellos no así una acción transversal. Por ello la fuerza aplicada sobre el modelo debería en principio dividirse en dos componentes, una que actúe longitudinalmente al resorte y otra que actúe perpendicularmente a éste. La fuerza longitudinal somete al resorte a un estiramiento o encogimiento como ya se analizó, pero la fuerza perpendicular representa un esfuerzo de giro. Esta fuerza no actúa sobre el resorte y debería tratarse por separado. Por simplicidad podría considerarse que toda fuerza de giro producida no se propaga, por lo que el ejemplo considerado en la Fig. (6.2) es el enfoque adecuado en ese sentido, sin embargo este mismo esquema debe emplearse en las fuerzas que se propagan a través de la malla.

En cada resorte la fuerza que aplican los resortes vecinos sobre éste es solo la componente paralela al resorte, como se muestra en la Fig. (6.3)

La fuerza etiquetada como f_4 que es aplicada al vértice 4 se descompone en dos fuerzas, una paralela (f_{4r}) y otra perpendicular (f_{4g}) al resorte entre P_4 y P_3. En la Fig. 6.3(b)

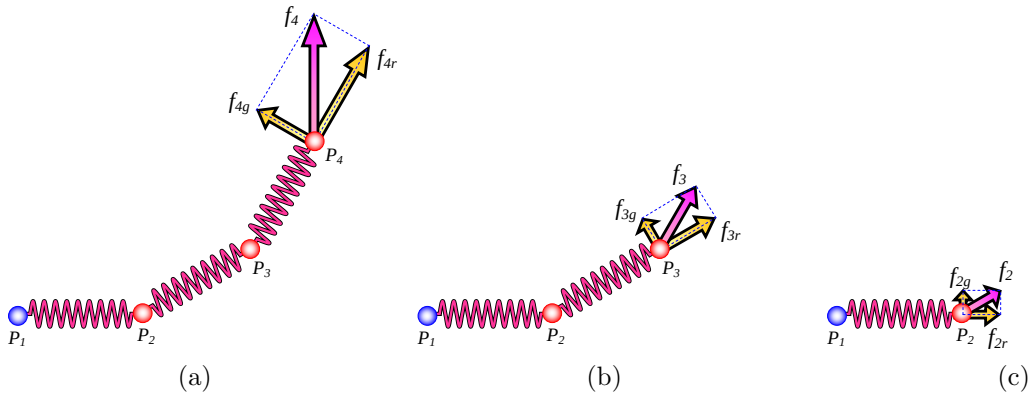


Figura 6.3: Propagación de la fuerza en tres sistemas MRA.

se muestra la que sería la fuerza propagada, esta tendrá la dirección del resorte entre P_4 y P_3, pero una magnitud menor, e igualmente debe descomponerse. Similarmente, en la Fig. 6.3(c) se muestra la fuerza propagada y sus componentes. Empleando este enfoque existen dos factores a resolver, el movimiento del resorte (en realidad se trata de un sistema MRA pero se habla solamente del resorte por simplicidad) debido a la fuerza longitudinal, cuestión ya analizada, y el movimiento debido a las fuerzas de giro (etiquetadas como f_{ig} en la figura). Situación que estaría por resolverse y que no es sencilla. Por ejemplo la fuerza f_{3g} de la Fig. 6.3(b) puede generar un giro en el resorte que va de P_2 a P_3 de modo que no varíe su longitud, sin embargo el vértice 3 no solo pertenece a un resorte si no a dos de ellos, por lo que de girar uno de los resortes moviendo este vértice se puede mantener sin variación la longitud de uno de los resortes mas no necesariamente de ambos. Si se variara la longitud de otro resorte entonces no se trataría de una fuerza de giro netamente. Todos estos efectos no están considerados en la Ec. (6.2). Una posibilidad para tratar las fuerzas de giro consistiría en el uso de resortes torsionales, sin embargo el sistema resultante sería más costoso, puesto que se añadiría un resorte extra sobre cada vértice y sería necesario calcular las componentes de la fuerza, una que actuaría sobre el resorte ya usado y la otra componente que sería perpendicular y que ejercería su acción sobre el resorte torsional. También se tendría que calcular la longitud del resorte longitudinal para así monitorear su variación, lo cual implica el cálculo de tres elevaciones al cuadrado más una raíz cuadrada. Sería también necesario establecer los modelos relativos al uso de resortes torsionales, tanto uno simple, como varios de ellos acoplados.

La complejidad inherente de este enfoque llevó a proponer un esquema más simple y por ende menos costoso computacionalmente, el cual se especifica en las ecuaciones siguientes, empleando un cambio en la notación de las Ec. (6.2) donde N se emplea para hacer referencia a un vértice vecino (N del inglés *neighbor*) se usará en su lugar v de vecino para mantener la notación usada en este documento

$$
\begin{aligned}
F_{int}(i)_x &= -k_r l_{ix} - \sum_{j=1}^{3} k \left(l_{v_j(i)x} - L_{v_j(i)x} \right) \\
F_{int}(i)_y &= -k_r l_{iy} - \sum_{j=1}^{3} k \left(l_{v_j(i)y} - L_{v_j(i)y} \right) \\
F_{int}(i)_z &= -k_r l_{iz} - \sum_{j=1}^{3} k \left(l_{v_j(i)z} - L_{v_j(i)z} \right)
\end{aligned} \tag{6.3}
$$

El término entre paréntesis representa la componente de longitud actual del resorte menos la componente de la longitud inicial del mismo. El término $L_{v_j(i)}$ se almacena justo después de crear la malla, solo que a diferencia del esquema propuesto en [27] no se almacena un único valor, si no que se guardan las tres componentes. Resultando en un sistema con menor costo computacional y que produce una simulación más realista. Empleando este nuevo esquema sobre el ejemplo mostrado en la Fig. (6.2) sí se obtiene un valor de la fuerza interna puesto que las coordenadas del vértice i han cambiado, no obstante que la longitud permanece constante, su posición no es la misma. De esta manera todo el sistema opera de forma independiente en cada coordenada, de modo que la interacción en una sola coordenada no afecta a las demás.

6.1.2. Propagación del movimiento

Cuando se aplica alguna fuerza sobre uno de los vértices de la malla, los vértices vecinos deben verse afectados, es decir, la fuerza aplicada sobre un vértice debe propagarse a través de la malla, como se mostró en la Fig. (6.1) cada vértice se acopla a sus vecinos por un sistema MRA de modo que si se mueve de su posición, los resortes jalan o empujan a los vértices vecinos y éstos a su vez a sus vecinos propagándose la acción sobre la malla. Para calcular la cantidad de movimiento que un vértice induce sobre sus vecinos se empleó el enfoque del sistema de resortes en serie de la Sec. (2.2.1). Ya que el sistema se calcula por coordenadas, el problema se reduce a resolver tres sistemas unidimensionales para los resortes asociados a los vecinos (que son los resortes que propagan el movimiento, pues el resorte al centro de la malla realiza otra función). Esto queda expresado en el segundo término de las Ec. (6.3), en esta se encuentran involucradas las longitudes de los tres resortes asociados a un vértice, de modo que si alguno de sus vecinos cambia de posición este término sufrirá a su vez un cambio que implicara la inyección de fuerza sobre el vértice. De esta forma todos los vértices de la malla se comportan como el sistema MRA central de la Sec. (2.2.1), aunque en este caso se interactúa con otros tres sistemas MRA en vez de dos, y todos los vértices son capaces de recibir una fuerza externa. Si bien estrictamente toda la malla de sistemas MRA que coexiste con la malla de simplejos debería resolverse como un sistema de ecuaciones para cada iteración del modelo, el costo computacional de tal enfoque sería excesivamente alto, pues se podría estar hablando de resolver un sistema de cientos de ecuaciones diferenciales. De modo que en cada iteración

del sistema se resuelve solo para un nivel de profundidad, es decir que el movimiento de un vértice en un instante de tiempo t afectará únicamente a sus tres vecinos en el instante $t + \Delta t$, y para el instante $t + 2\Delta t$ el movimiento se habrá propagado a los vecinos de los vecinos, así que en cada iteración la fuerza se propaga al siguiente nivel de vértices vecinos. Este enfoque conlleva un retardo puesto se necesitan de tantas iteraciones como niveles de profundidad tenga la malla, que mientras más densa sea, mayor profundidad tendrá. Sin embargo esta aproximación es tolerable (y necesaria para tener un sistema que opere en tiempo real) ya que en un sistema físico real existe el mismo tipo de retardo, esto es que cuando se tienen varios sistemas MRA en serie y se aplica una fuerza en uno del extremo el efecto tardará en propagarse, aunque matemáticamente el movimiento se propaga de inmediato, la magnitud se hace significativa solo hasta después de cierto tiempo de retardo, que será mayor mientras más sistemas MRA se interpongan entre el que recibió la fuerza y el que se está observando. Esto se muestra en las gráficas de la Fig. (6.4)

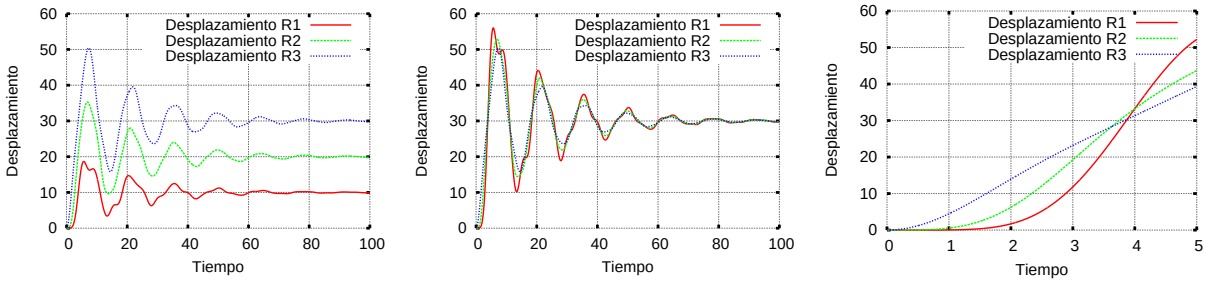


Figura 6.4: Gráficas correspondientes a un sistema MRA triple serie como el mostrado en la Fig. (2.5).

El resorte R3 es el que recibe la fuerza externa, en la Fig. 6.4(a) se tiene la gráfica para ciertos parámetros; en la Fig. 6.4(b) se escalan las tres gráficas de modo que tengan la misma amplitud, tendiendo entonces al mismo valor, treinta en este caso; en la Fig. 6.4(c) se muestra una ampliación de la gráfica Fig. 6.4(b), es claro que el resorte R2 que se encuentra junto al R3 tiene cierto retardo en su movimiento, siendo R1, que se encuentra más distanciado de R3, el que tiene el mayor retardo, es decir sigue un movimiento similar al de R3 pero desfasado en el tiempo. Los ejes muestran una graduación solo para fines ilustrativos.

6.2. Detección de colisiones y resolución del contacto

Ya en la Sec. (5.5) en la Pag. (48) se detalló la forma en la que se detectan colisiones empleando la biblioteca SOLID, también se habló del vector de profundidad de penetración para corregir la penetración, en esta sección se ahondará un poco más para introducir

el concepto de fricción entre las superficies en contacto, que permitiría, en teoría, obtener una simulación más realista.

Cuando un objeto choca con el modelo deformable aplica una fuerza sobre los triángulos con los que hizo contacto. La fuerza aplicada causa que los triángulos pasen de la posición previa al contacto a una posición nueva De forma tal que la detección de colisiones y la resolución al contacto se realiza en dos etapas, primero se realiza una prueba por colisiones entre el objeto externo y el modelo deformable, si no existe colisión entonces no se realiza acción alguna, si en efecto existe una colisión, es decir que el objeto externo ya ha penetrado al modelo deformable (el caso de que los objetos se encuentren en contacto, pero sin penetrarse, puede desestimarse puesto que visualmente no existe ninguna incoherencia), deben moverse los triángulos del objeto deformable de su posición dentro del objeto rígido a una posición en la no exista penetración. El problema es determinar esa nueva posición de los triángulos, la cual depende de las características físicas asociadas al modelo, como se verá a continuación.

6.2.1. Esquemas de resolución del contacto con y sin fricción

Si bien el presente trabajo trata de la interacción con modelos deformables, para resolver el contacto de un objeto rígido y un modelo deformable, la superficie de éste último puede considerarse como rígida durante el contacto.

Cuando dos objetos rígidos se mueven entrando en contacto, su movimiento queda determinado por las fuerzas que actúan sobre ellos. En el presente caso solo se considera la fuerza de contacto y además puede considerarse una fuerza de fricción entre las superficies que determinará la forma en la que el modelo se deforma en el área de contacto. La fuerza de fricción determina qué tan fácilmente un objeto puede deslizarse sobre otro, por ejemplo dos superficies de lisas de metal pulido se deslizarán suavemente una sobre la otra, no así las mismas superficies rugosas. La fuerza de fricción se manifiesta de forma tangencial a la superficie y se calcula como una fracción de la fuerza normal

$$F_f = \mu F_N \tag{6.4}$$

donde, F_f es la fuerza de fricción, μ es el coeficiente de fricción y F_N es la fuerza normal.

La fuerza de fricción se presenta en dos formas distintas, como fricción estática y cinética, en el caso de la fricción estática, esta se presenta cuando se aplica una fuerza para deslizar un objeto sobre otro, pero ésta no es suficiente como para lograr un desplazamiento, por ejemplo al tratar de empujar una caja pesada será necesario aplicar una fuerza de gran magnitud para moverla, mientras la caja no se mueva de su posición será la fuerza de fricción estática la responsable de que la caja permanezca en su sitio pero una vez que se haya en movimiento, la fricción presente se llama fricción cinética [30]. En este trabajo solo se considerará un tipo de fricción asociada al contacto, la cinética.

La importancia de la fuerza de fricción sobre el esquema de deformación radica en que permite darle un sentido físico a la resolución del contacto, es decir, el cómo reacciona el objeto deformable a un esfuerzo externo depende de los parámetros físicos asociados a la superficie, en este caso depende del coeficiente de fricción entre las superficies en contacto,

lo cual le da una sensación más realista al sistema puesto que en el mundo real rara vez interactuamos con un objeto sin fricción [33]. La fuerza de fricción, tambien es importante en evitar que las superficies de los cuerpos se deslicen unas sobre otras [18]

Un aspecto de especial consideración en la simulación realizada es que el objeto rígido no recibe ningún tipo de acción del modelo deformable, es decir que el objeto deformable no puede empujar al rígido, solamente puede posarse sobre él. La razón principal de seguir este enfoque es que el sistema emplea un guante sensorizado sin reacción, es decir que el guante permite conocer la posición del mismo, dicha posición se utiliza para controlar la posición del objeto sólido, pero no permite imponerle restricción alguna.

En la Fig. (6.5) se muestra un ejemplo de un contacto con fricción, se utiliza un sistema de dos dimensiones que se extenderá a tres dimensiones, para simplificar el análisis.

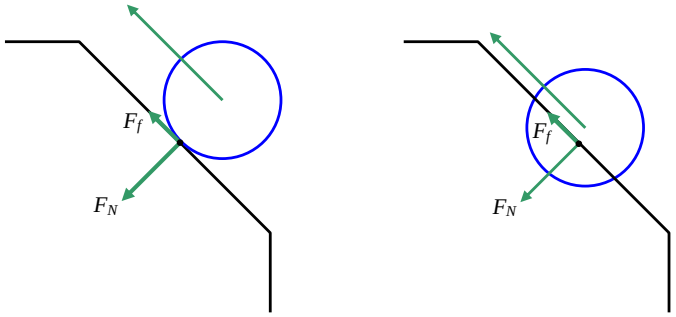


Figura 6.5: Fuerza normal y de fricción en una colisión.

Los segmentos de recta mostrados forman parte del modelo deformable, y el círculo es el objeto rígido. Como puede verse existen dos fuerzas asociadas al contacto, una que es perpendicular a la circunferencia, y otra que es tangencial a la misma. Ambas fuerzas están asociadas al círculo, puesto que es este el que aplica fuerza al modelo deformable. La fuerza normal es producida por la esfera, así como la fuerza de fricción, si bien la fuerza de fricción se opone normalmente al movimiento, en este caso su dirección es a favor del movimiento, esto es debido a que en este caso el círculo es el objeto de referencia, que no puede ser movido por el objeto deformable. Se presentan dos casos, el ideal, donde solo existe contacto que se muestra en la Fig. 6.5(a) y en la Fig. 6.5(b) se presenta el caso para un sistema discreto en el que la detección se realiza una vez que se ha dado la penetración.

Dado que el VPP es usado para la corrección de la penetración, no se calcula fuerza alguna, por lo que la fuerza normal podría considerarse como la fuerza necesaria para empujar el objeto deformable de su posición en la que existe penetración a una posición donde no existe tal. Como la fuerza de fricción es proporcional a la fuerza normal cuya magnitud no se conoce, podría simplificarse el esquema de fricción evadiendo la necesidad de calcular la fuerza normal empleando el mismo método de desplazamientos, considerando indirectamente la fuerza de fricción como un desplazamiento debido a la fuerza normal y que es proporcional al VPP, a ese vector le llamaremos VPP_f. De esta manera,

en lugar de aplicar el VPP para resolver la colisión se aplicará un vector VPP′ que será la resultante de sumar los vectores VPP y VPP_f. El vector VPP_f se calcula de la manera siguiente

$$|\text{VPP}_f| = \mu|\text{VPP}| \tag{6.5}$$
$$\text{VPP}_f \cdot VPP = 0 \tag{6.6}$$
$$\text{VPP}_f \cdot d \geq 0 \tag{6.7}$$

donde, d es el vector de dirección de movimiento del objeto rígido.

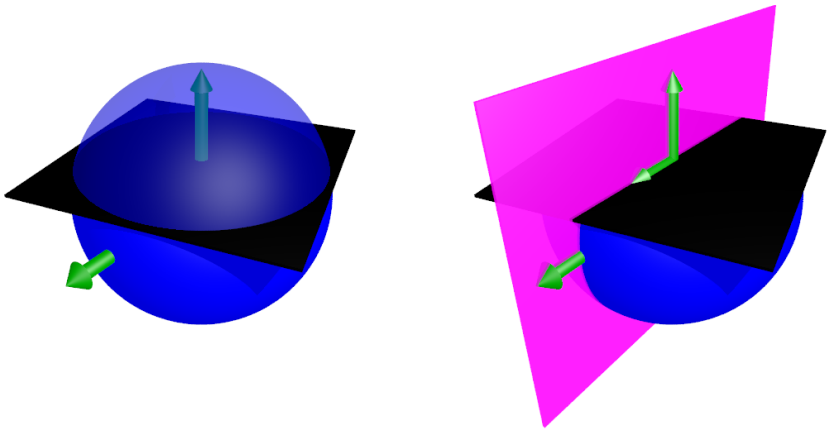
La Ec. (6.5) indica la magnitud del vector debido a la fricción (VPP_f), la Ec. (6.6) indica su dirección y la Ec. (6.7) indica su sentido. La Ec. (6.7) es necesaria debido a que existen dos vectores que son perpendiculares a VPP, ambos con la misma dirección pero sentidos opuestos.

Este análisis es válido para un sistema bidimensional, sin embargo en un sistema tridimensional el análisis es más complejo debido a que existen un número infinito de vectores que son perpendiculares a VPP, a diferencia del caso bidimensional donde solo existen dos. Para resolver el caso tridimensional, es necesario añadir una ecuación que indique la dirección de VPP_f sin ambigüedades

$$(\text{VPP} \times d) \times (\text{VPP} \times \text{VPP}_f) = 0 \tag{6.8}$$

La Ec. (6.8) implica que los tres vectores, VPP, d y VPP_f estén en el mismo plano. Cabe señalar que la fricción solo se dará si la dirección del objeto deformable no es perpendicular a la superficie de contacto, en cuyo caso la expresión $(\text{VPP} \times d)$ será nula e indicará que no existe fricción por lo que $\text{VPP}_f = 0$.

Esta situación se ejemplifica en la Fig. (6.6)



(a) (b)

Figura 6.6: Penetración entre una esfera y un cuadrilátero.

En la Fig. 6.6(a) se muestran dos objetos, una esfera y un cuadrilátero traslapándose, la parte superior de la esfera se muestra semitransparente para poder observar el VPP, que se simboliza con la flecha que sale perpendicularmente del cuadrilátero y que termina justo en la superficie de la esfera. Si el cuadrilátero se trasladara por el VPP, se colocaría justo sobre la superficie de la esfera pero sin que existiese penetración. También se muestra otra flecha que sale de la esfera y que indica la dirección de movimiento de la esfera. En la Fig. 6.6(b) se eliminó la parte superior de la esfera para poder apreciar mejor el VPP, también se colocó un plano que pasa por el VPP y por el vector de dirección de la pelota, sobre ese mismo plano, y de forma perpendicular al VPP se muestra el que sería el VPP_f. Puede notarse que el plano añadido al final es necesario para tener un VPP_f único puesto que en principio cualquier vector paralelo al cuadrilátero sería perpendicular al VPP.

6.2.2. En resumen

El sistema de deformación presentado en [27] no incorpora los efectos de las fuerzas de giro, por lo que proporciona resultados poco realistas. Una mejorá que se presenta es utilizar tres resortes unidimensionales en lugar de un resorte tridimensional. Pero sería aún más apegado a la realidad emplear resortes torsionales.

La propagación del movimiento se realiza en un nivel de profundidad en cada iteración, es decir que la acción actual de los resortes solo afectará a los resortes contiguos en la siguiente iteración, esto conlleva un retraso en la propagación, al igual que como sucede en el mundo real.

Dado que la deformación en este caso, se presenta como un efecto causado por la acción de objetos rígidos, se determinó que usando el VPP se tenía una solución simple y adecuada para la simulación.

También se estableció el concepto de fricción superficial sobre el contacto para obtener una simulación más realista.

Capítulo 7

Análisis de estabilidad

Es necesario analizar la estabilidad del sistema para determinar bajo que condiciones el sistema será estable, bajo cuales no lo será y como, cuando sea inestable, puede estabilizarse.

La ecuación diferencial que modela el sistema MRA es la siguiente

$$m\ddot{x} + b\dot{x} + kx = f \tag{7.1}$$

con $m > 0$, $b > 0$ y $k > 0$.

La forma de estabilizar el sistema consiste en el uso de un sistema de control retroalimentado, se analizarán dos de ellos, el control proporcional y el proporcional-derivativo.

7.1. Estabilidad del sistema continuo

Para realizar el análisis de estabilidad primero es necesario obtener la transformada de Laplace de la ecuación Ec. (7.1), como ya se mencionó la fuerza f en el sistema hasta ahora usado es constante, pero aplicando el sistema de control se considerarán dos casos.

7.1.1. Control proporcional

Empleando el control proporcional, la fuerza f de la Ec. (7.1) se considerará como $f = k_p y$ donde k_p es una constante y y es la variable de entrada. Si se aplica la transformada de Laplace en ambos lados de la ecuación se obtiene la siguiente igualdad

$$\begin{aligned} \mathscr{L}\{m\ddot{x} + b\dot{x} + kx\} &= \mathscr{L}\{k_p y\} & (7.2)\\ ms^2X(s) + bsX(s) + kX(s) &= k_pY(s) & (7.3) \end{aligned}$$

La función de transferencia se define como la salida del sistema entre la entrada del sistema, y es necesario conocerla para determinar la estabilidad, cabe recalcar que la entrada

es el desplazamiento y la salida será el desplazamiento provocado por la fuerza aplicada. A continuación se muestra la expresión correspondiente a la función de transferencia

$$\frac{X(s)}{Y(s)} = \frac{k_p}{ms^2 + bs + k} \tag{7.4}$$

La función de transferencia anterior se conoce como ganancia en lazo abierto y se representa por la letra G, la razón de que se llame de lazo abierto es debido a que no existe ningún tipo de retroalimentación en el sistema, de modo que cualquier variación anómala en la salida pasará desapercibida, por ello, para controlar el sistema es necesario implementar algún tipo de retroalimentación. Para continuar con el análisis de estabilidad es necesario determinar la función de transferencia del sistema en lazo cerrado, es decir con retroalimentación, como se muestra en la Fig. (7.1).

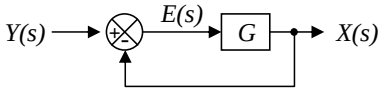


Figura 7.1: Sistema con retroalimentación.

La ganancia de un sistema de este tipo es la siguiente [17]

$$\frac{G}{1+G} \tag{7.5}$$

Si se sustituye la ganancia en lazo abierto se obtiene la siguiente ecuación

$$\frac{G}{1+G} = \frac{k_p}{ms^2 + bs + (k + k_p)} \tag{7.6}$$

La ecuación característica del sistema es aquella que hace igual a cero el denominador de la función de transferencia, haciendo $(k + k_p) = k_a$ en la ecuación anterior obtenemos la que se muestra a continuación

$$ms^2 + bs + k_a = 0 \tag{7.7}$$

Un sistema es inestable si la parte real de alguna de las raíces de la ecuación característica es positiva, marginalmente inestable si es cero y estable en otro caso (con ciertas excepciones) [17]. A las raíces de la ecuación característica se les llama polos.

De modo que todo consiste en determinar el signo de la parte real de las soluciones a la ecuación 7.7, dichas soluciones son las siguientes

$$\frac{-b \pm \sqrt{b^2 - 4mk_a}}{2m} \tag{7.8}$$

Se presentan los casos mostrados en la Tab. (7.1)

Esta tabla señala que la estabilidad del sistema depende de forma directa del valor de k_p. Para que el sistema sea estable se requiere que $k_a > 0$ que implica que $k_p > -k$.

k_a	$\mathfrak{Re}\left\{\frac{-b_a+\sqrt{{b_a}^2-4mk_a}}{2m}\right\}$	$\mathfrak{Re}\left\{\frac{-b_a-\sqrt{{b_a}^2-4mk_a}}{2m}\right\}$	Tipo de estabilidad
> 0	< 0	< 0	Estable
$= 0$	$= 0$	< 0	Marginalmente inestable
< 0	> 0	< 0	Inestable

Tabla 7.1: Estabilidad del control proporcional.

7.1.2. Control proporcional-derivativo

Realizando el análisis para el control proporcional-derivativo se tiene $f = k_p y + k_d \dot{y}$, la función de transferencia en lazo abierto esta dada por la expresión siguiente

$$\frac{X(s)}{Y(s)} = \frac{k_p + k_d s}{ms^2 + bs + k} \tag{7.9}$$

La función de transferencia en lazo cerrado respectiva es la siguiente

$$\frac{X(s)}{Y(s)} = \frac{k_p + k_d s}{ms^2 + (b + k_d)s + (k + k_p)} \tag{7.10}$$

haciendo $b_a = (b + k_d)$ y $k_a = (k + k_p)$ la ecuación característica es

$$ms^2 + b_a s + k_a = 0 \tag{7.11}$$

cuyas soluciones están dadas la expresión que se muestra a continuación

$$\frac{-b_a \pm \sqrt{{b_a}^2 - 4mk_a}}{2m} \tag{7.12}$$

Nuevamente se tienen varios casos los cuales se presentan en la Tab. (7.2)

b_a	k_a	$\mathfrak{Re}\left\{\frac{-b_a+\sqrt{{b_a}^2-4mk_a}}{2m}\right\}$	$\mathfrak{Re}\left\{\frac{-b_a-\sqrt{{b_a}^2-4mk_a}}{2m}\right\}$	Tipo de estabilidad
> 0	< 0	> 0	< 0	Inestable
	$= 0$	$= 0$	< 0	Marginalmente inestable
	> 0	< 0	< 0	Estable
$= 0$	< 0	> 0	< 0	Inestable
	$= 0$	$= 0$	$= 0$	Marginalmente inestable
	> 0	$= 0$	$= 0$	Marginalmente inestable
< 0	< 0	> 0	< 0	Inestable
	$= 0$	> 0	$= 0$	Inestable
	> 0	> 0	> 0	Inestable

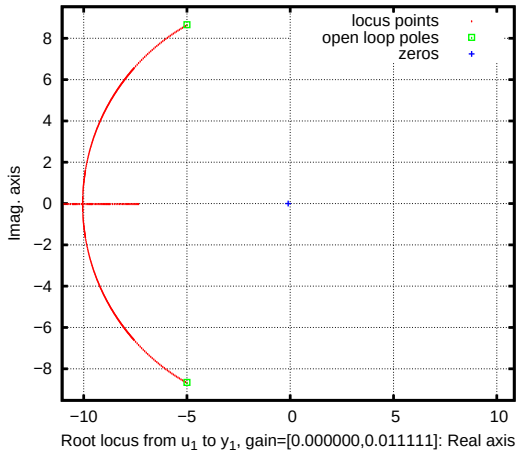
Tabla 7.2: Estabilidad del control proporcional-derivativo.

De modo que el sistema será estable solo cuando $k_a > 0$ y $b_a > 0$ lo cual implica que si $k_p > -k$ y $k_d > -b$ se tendrá un sistema estable.

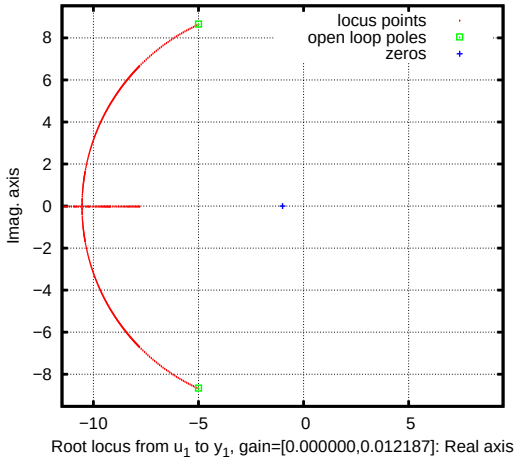
Esta situación puede verificarse de forma gráfica, haciendo un pequeño reordenamiento a la Ec. (7.10) se obtiene la siguiente relación

$$\frac{X(s)}{Y(s)} = \frac{k_p(1 + \frac{k_d}{k_p}s)}{ms^2 + (b + k_d)s + (k + k_p)} \tag{7.13}$$

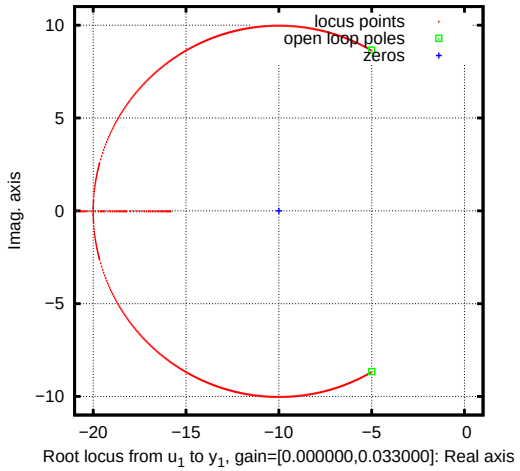
En la figura 7.2 se muestran varias gráficas de los polos de la función de transferencia retroalimentada para el control proporcional derivativo. El valor de k_p es representado como *gain* en las figuras.



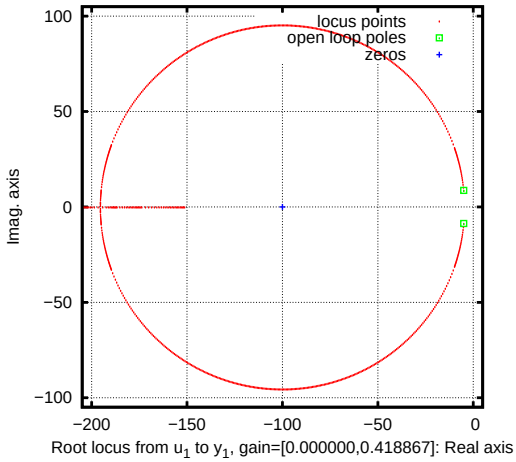
(a) $\frac{k_d}{k_p} = 0.1$



(b) $\frac{k_d}{k_p} = 1$



(c) $\frac{k_d}{k_p} = 10$



(d) $\frac{k_d}{k_p} = 100$

Figura 7.2: Gráficas de los polos de la Ec. (7.13), con $m = 0.001$, $b = 0.01$ y $k = 0.1$.

El análisis hasta aquí presentado sirve para determinar la estabilidad del sistema considerándolo continuo, sin embargo la estabilidad también está relacionada con el tamaño

de paso del sistema discreto.

7.2. Estabilidad del sistema discreto

Para llevar a cabo ese análisis primero es necesario aplicar la transformación bilineal que se define como:

$$s = \frac{2(z-1)}{T(z+1)} \tag{7.14}$$

donde, z es la variable en el dominio discreto y T es el periodo de discretización. Aplicándola a la Ec. (7.11) se deduce la siguiente igualdad

$$m\left[\frac{2(z-1)}{T(z+1)}\right]^2 + b_a\left[\frac{2(z-1)}{T(z+1)}\right] + k_a = 0 \tag{7.15}$$

manipulando algebraicamente se obtiene la expresión siguiente

$$\frac{z^2(4m + 2b_aT + k_aT^2) + z(-8m + 2k_aT^2) + (4m - 2b_aT + k_aT^2)}{T^2(z+1)^2} = 0 \tag{7.16}$$

La ecuación característica esta dada por la siguiente igualdad

$$F(z) = z^2(4m + 2b_aT + k_aT^2) + z(-8m + 2k_aT^2) + (4m - 2b_aT + k_aT^2) = 0 \tag{7.17}$$

Las condiciones necesarias y suficientes para tener un sistema discreto estable de segundo orden del tipo

$$F(z) = a_0z^2 + a_1z + a_2 = 0 \tag{7.18}$$

son las siguientes [17]:

- $F(1) > 0$
- $F(-1) > 0$
- $|a_0| > a_2$

Aplicando la primera condición se tienen las siguientes simplificaciones

$$\begin{aligned} F(1) &= (4m + 2b_aT + k_aT^2) + (-8m + 2k_aT^2) + (4m - 2b_aT + k_aT^2) > 0 \\ F(1) &= T^2(k_a + 2k_a + k_a) + T(2b_a - 2b_a) + (4m - 8m + 4m) > 0 \\ F(1) &= T^2(4k_a) > 0 \Rightarrow T^2k_a > 0 \end{aligned}$$

como $T > 0$ la condición anterior implica que $k_a > 0$. Aplicando la segunda condición se obtienen las siguientes simplificaciones

$$\begin{aligned} F(-1) &= (4m + 2b_aT + k_aT^2) - (-8m + 2k_aT^2) + (4m - 2b_aT + k_aT^2) > 0 \\ F(-1) &= T^2(k_a - 2k_a + k_a) + T(2b_a - 2b_a) + (4m + 8m + 4m) > 0 \\ F(-1) &= 16m > 0 \Rightarrow m > 0 \end{aligned}$$

esta condición se cumple puesto que $m > 0$ siempre. Finalmente la tercer condición implica la siguiente relación

$$|4m + 2b_aT + k_aT^2| > (4m - 2b_aT + k_aT^2) \tag{7.19}$$

Como $m > 0$, se deriva la siguiente relación

$$|4m + 2b_aT + k_aT^2| = 4m + |2b_aT + k_aT^2| > (4m - 2b_aT + k_aT^2) \tag{7.20}$$

$$|2b_aT + k_aT^2| > (-2b_aT + k_aT^2) \tag{7.21}$$

Como $T > 0$ se puede dividir ambos lados de la inecuación entre T para obtener la relación mostrada a continuación

$$|2b_a + k_aT| > (-2b_a + k_aT) \tag{7.22}$$

Dado que ya se había determinado que k_a debía ser mayor que cero para tener un sistema estable, considerando esa restricción, entonces puede simplificarse la expresión anterior como se muestra a continuación

$$|2b_a + k_aT| = 2|b_a| + k_aT > (-2b_a + k_aT) \tag{7.23}$$

$$2|b_a| > -2b_a \tag{7.24}$$

$$|b_a| > -b_a \tag{7.25}$$

lo cual solo se cumple si $b_a > 0$

De modo que para tener un sistema estable es necesario y suficiente que $k_a > 0$ y $b_a > 0$. Para el control proporcional esto significa que $k_p > -k$, para el control proporcional-derivativo se requiere que $k_p > 0$ y $k_d > 0$.

7.3. Implementación del sistema de control

Para incluir el sistema de control en la aplicación se utilizó el programa Octave, mediante el cual es posible obtener de forma numérica la transformada bilineal de la función de transferencia. El proceso que se siguió consistió en transformar la función de transferencia continua del control proporcional-derivativo mostrada en la Ec. (7.10) al dominio discreto haciendo uso de la transformada bilineal. Esto mediante la función `c2d` de Octave.

La cual proporciona como salida una serie de matrices que corresponden al sistema discreto en función del tiempo. Aplicando operaciones matriciales en la aplicación se obtiene el valor de salida bajo el control proporcional-derivativo.

La función `c2d` recibe como entrada un sistema continuo como el especificado en la Ec. (7.13). La salida de la función `c2d` entrega cuatro matrices del sistema discreto que es ahora:

$$\begin{aligned} x_{i+1} &= Ax_i + Bu \\ y_i &= Cx_i + Du \end{aligned}$$

donde u es la entrada del sistema y y_i la salida.

Para el sistema de la Ec. (7.13), con los valores, $m = 0.001$, $b = 0.01$, $k = 0.1$, $k_p = 0.5$, $k_d = 10.0$ y para un tiempo de muestreo de $t = 1/30$; los valores de las matrices A, B, C y D son:

$$A = \begin{pmatrix} 9.9802 \times 10^{-01} & 1.9841 \times 10^{-04} \\ -1.1905 \times 10^{-01} & -9.8810 \times 10^{-01} \end{pmatrix}$$

$$B = \begin{pmatrix} 1.8113 \times 10^{-05} \\ 1.0868 \times 10^{-03} \end{pmatrix}$$

$$C = \begin{pmatrix} -17.479 & 10.877 \end{pmatrix}$$

$$D = \begin{pmatrix} 0.99289 \end{pmatrix}$$

Un ejemplo del comportamiento del sistema discreto se muestra en la Fig. (7.3)

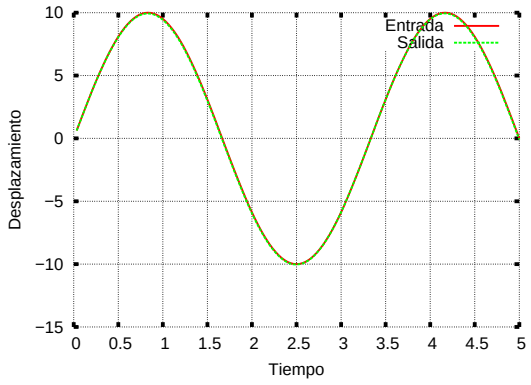


Figura 7.3: Señal de entrada y de salida bajo la aplicación de un control proporcional.

De la figura no puede notarse una diferencia significativa entre ambas curvas. La curva de entrada se encima sobre la curva de salida, lo cual significa que existe una diferencia muy pequeña entre ambas. Esta situación es precisamente lo que se buscaba.

Capítulo 8

Aplicación implementada y experimentos realizados

Las características técnicas específicas del *hardware-software* empleado son las siguientes

1. Características generales
   - *a*) Procesador AMD de 64 bits.
   - *b*) Sistema operativo Red Hat Enterprise Linux ES 3.
   - *c*) Versión del núcleo del sistema (*kernel*) 2.4.21-47.
   - *d*) Compilador gcc versión 3.2.3.
   - *e*) Se empleó Qt versión 3.1.2 para la creación de la interfaz gráfica.
   - *f*) SOLID versión 3.5.4.
2. Etapa de interacción para la deformación
   - *a*) Guantes sensorizados 5DT modelo 5N.
   - *b*) Adaptador USB-SERIAL para la conexión de los guantes a la PC.
   - *c*) Controlador libre para el manejo de los guantes a través del puerto serial [9].
3. Etapa de interacción visual
   - *a*) Lentes StereoGraphics modelo CrystalEyes3 para la visión estereoscópica.
   - *b*) Emisor StereoGraphics modelo E-2 para la sincronización con los lentes.
   - *c*) Monitor ViewSonic modelo G225F ajustado a una frecuencia de refresco de 118Hz.
   - *d*) Tarjeta gráfica NVIDIA Quadro FX1400 con capacidades para visión estereoscópica.

En las secciones siguientes se detallan las posibles opciones que existen y el porque se seleccionaron las especificaciones antes mencionadas.

8.1. Dispositivos de *hardware* empleados

8.1.1. Interacción para la deformación

Existen diversos dispositivos de hardware que pueden emplearse para permitir a un usuario deformar un modelo virtual. Estos se agrupan en dos categorías, dependiendo de si permiten retroalimentación de fuerzas o no. Entre los que sí permiten retroalimentación de fuerza destaca el *Phantom Omni* [27, Cap 5], el cual consiste de un señalador conectado a través de una serie de brazos mecánicos a un sistema computarizado, que su vez se conecta a una computadora. Si bien los modelos con retroalimentación de fuerza permiten una interacción muy realista, el modelo matemático para este tipo de dispositivos resulta complejo, como se señala en el capítulo 3 y en el caso del *Phantom Omni* la interacción únicamente puede llevarse a cabo sobre un punto.

Entre los modelos sin retroalimentación, destacan los guantes sensorizados ya que permiten una interacción con varios puntos. Típicamente cada dedo puede controlar al menos un punto en el mundo virtual. Por ejemplo el modelo 14N de la marca 5DT contiene catorce sensores, emplea dos sensores por cada dedo con la finalidad de medir el nivel de flexión de cada dedo, también cuenta con cuatro sensores que supervisan la separación entre los dedos. Otro modelo del mismo fabricante, es el 5N, el cual tiene prestaciones más modestas pues solo cuenta con cinco sensores, los cuales miden el grado de flexión de cada dedo. En ambos modelos los sensores proporcionan valores dentro de un rango dado, donde un extremo del rango significa una flexión nula mientras el otro valor extremo indica una flexión máxima, los valores intermedios hacen referencia a un grado de flexión proporcional. Los guantes también cuentan con un sensor de giros para medir la inclinación de la mano en dos marcos de referencia perpendiculares entre si. Una carencia de ambos tipos de guantes es un sensor de posición absoluta, es decir que no puede obtenerse la posición del guante, sin embargo este tipo de dispositivos se encuentran disponibles por separado y pueden añadirse al guante en caso necesario.

Dado que se contaba con un par de guantes de modelo 5N se decidió hacer uso de ellos. Aunque si bien son el modelo más simple, si permiten una interacción adecuada del usuario, además de que por su simplicidad son sencillos de incorporar en el sistema.

El sensor de giros se empleó en la aplicación para girar la escena. Los guantes se conectan a la PC a través del puerto serial. Dado que la computadora empleada no tenía ese puerto se utilizó un adaptador USB-SERIAL para la conexión de los guantes. Es de considerarse que el puerto serial proporciona una velocidad de transmisión (19200 bits por segundo para los guantes modelo 5N) no muy alta, sin embargo para la aplicación implementada es suficiente.

Para comunicarse con los guantes, se utilizó una biblioteca de código libre [9] debido a que el fabricante solo proporciona una biblioteca precompilada para enlazado dinámico. La cual solo funciona para versiones antiguas del compilador gcc. La biblioteca usada está programada en lenguaje C++ y proporciona una serie de funciones de comunicación para obtener los datos del guante.

Cabe señalar que dada la ausencia de un sensor de posición absoluta en el guante,

cualquier movimiento que preserve las inclinaciones y las flexiones de los dedos pasará desapercibido.

8.1.2. Interacción visual

Existen diferentes formas de presentar la simulación al usuario, siendo la más simple de ellas el uso de un monitor común. Otro tipo de esquemas para visualización permiten aumentar el grado de realismo al introducir el concepto de visión estereoscópica, la cual puede lograrse de múltiples maneras [32].

La visión estereoscópica se presenta en la vista humana debido a que cada ojo ve una imagen ligeramente diferente al otro. En un monitor, en principio no es posible obtener una visualización tridimensional debido a que se trata de una imagen plana, sin embargo existe una forma de presentar una imagen distinta a cada ojo empleando lentes especiales, de los cuales se tienen pasivos y activos. Los sistemas de mayor uso para lentes pasivos, presentan dos imágenes sobrepuestas, una en colores rojizos correspondiente a un ojo y otra en colores verdosos que corresponde a la imagen que se tiene desde el ángulo de visión del otro ojo. Empleando unos lentes que tienen un filtro de color verde para un ojo y un filtro rojo para el otro, el usuario verá con mayor intensidad la imagén correspondiente al ojo en cuestión (recordemos que ambas imágenes se presentan simultáneamente), de esta manera el usuario puede crear mentalmente una imagen tridimensional. Si bien el resultado es una sensación de tridimensionalidad el efecto es difícil de mantener.

El funcionamiento de los lentes activos consiste en alternar la visión de cada ojo. Estos sistemas, simbolizados por los lentes CrystalEyes de la marca StereoGraphics, muestran en la pantalla de la computadora imágenes alternadas para el ojo izquierdo y el ojo derecho, al mismo tiempo que los lentes alternan cubriendo completamente la visión de un ojo mientras el otro mantiene visión total y viceversa. Esto se logra haciendo uso de pantallas de cristal líquido que se obscurecen y aclaran completamente en un tiempo muy breve. El resultado es que el usuario ve con un ojo la imagen indicada y luego ve con el otro la otra imagen, al aumentar la frecuencia del sistema a sesenta Hertz o por arriba, el espectador obtiene una sensación de continuidad en la imagen mostrada y a su vez de tridimensionalidad. Sin embargo su uso prolongado, dado que permanece cierto efecto de vibración en la imagen, puede ser molesto, además, los monitores comunes presentan una imagen con una presistencia muy alta por lo que el usuario ve ambas imágenes simultáneamente, no obstante que una de ellas sea mucho más tenue que la otra, es visible.

Estos sitemas de visualización pueden emplearse sobre un monitor común, sin embargo existen otros enfoques que permiten al usuario explorar más ampliamente el mundo virtual creado. En particular existen dos sistemas mejorados, uno emplea lentes con dos pequeños monitores incluidos, a estos lentes se les agrega un sensor de orientación, por medio del cual es posible conocer la dirección en la que el usuario observa para así mostrarle la imagen correspondiente [2]. Este sistema ha demostrado proporcionar una experiencia más realista [31], sin embargo el *hardware* es costoso y presenta un retardo cuando el usuario gira la cabeza en otra dirección (para observar una parte de la escena distinta) puesto

que debe calcularse y presentársele la imagen correspondiente. Otro sistema de visualización mejorado consiste en el uso de una o varias pantallas alrededor del usuario, a este sistema se le conoce como CAVE [7] del inglés ***C****AVE* ***A****utomatic* ***V****irtual* ***E****nvironment*, y a diferencia del sistema de monitores empotrados en los lentes, es un sistema en tiempo real puesto que las imágenes que se despliegan en las pantallas están presentes en todo momento aunque el usuario no este observándolas, un sistema de este tipo puede ofrecer distintas capacidades dependiendo del uso, por ejemplo si se trata de una cabina para simulación de vuelo, la pantalla puede ser una semiesfera y no se requerirá una imagen detrás de usuario, pero en otro caso puede colocarse al usuario dentro de un cuarto completamente rodeado por una pantalla (con la posible excepción del suelo). El hardware también suele ser costoso y requiere que todas las áreas de la escena se calculen aunque el usuario no las observe, además requiere un espacio amplio para la colocación de todas las pantallas.

Se optó por usar los lentes CrystalEyes empleando un monitor común, ya que la biblioteca de visualización empleada (OpenGL) facilita su incorporación. En específico se utilizaron unos lentes de la marca StereoGraphics modelo CrystalEyes3. Estos lentes consisten de dos pantallas de cristal líquido las cuales se obscurecen alternadamente. Sin embargo los lentes no funcionan de manera independiente puesto que es necesario se encuentren sincronizados con la frecuencia de refresco de la pantalla, la cual debe ser mayor a 80 hertz según indica el fabricante, esto para que el usuario vea una imagen continua sin percatarse de la alternación que se presenta entre las imágenes. Para llevar a cabo la sincronización debe emplearse una tarjeta de video con capacidades de visión estereoscópica, y un emisor, el cual es un aparato que se conecta a la PC a través de la tarjeta de video y que emite una señal de sincronización infrarroja la cual es recibida por los lentes. Se utilizó la tarjeta NVIDIA Quadro FX1400 y el emisor StereoGraphics modelo E-2, además se empleó un monitor ViewSonic modelo G225F dados los requerimientos de una frecuencia de refresco más alta que la proporcionada por un monitor común.

Para la visión tridimensional es necesario calcular dos imágenes de la escena cada una correspondiente a cada ojo. De modo que cada imagen tendrá un ángulo de visión ligeramente distinto. Esta situación depende de la separación entre el observador y el objeto observado, la separación existente entre el ojo derecho y el izquierdo y el punto al que el observador este mirando. En la aplicación implementada las imágenes se calcularon empleando una pequeña rotación, sobre el plano horizontal de la escena. La diferencia radica en el sentido de la rotación el cual es opuesto en cada marco. El ajuste del ángulo se realizo aumentando el valor hasta obtener una imagen adecuada, es decir con la suficiente magnitud para apreciar el efecto tridimensional, pero no tan amplia como para que el observador vea dos imágenes diferentes en lugar de una imagen tridimensional. Este enfoque considera implícitamente que la mirada del observador se dirige al origen de la escena.

Un ejemplo de esto se muestra en la Fig. (8.1).

En la Fig. 8.1(a) se muestra la imagen correspondiente al ojo izquierdo, y en la Fig. 8.1(b) se muestra la imagen que verá el ojo derecho. Puede notarse como las imágenes presentan una pequeña variación, la del ojo izquierdo parece tomada un poco más a la

(a) Marco izquierdo. (b) Marco derecho.

Figura 8.1: Ejemplo de una imagen bajo visualización estereoscópica.

izquierda que la de la derecha. Precisamente esa variación es la que permite tener una visualización tridimensional, puesto que emula la visión estereoscópica del ser humano.

En lo referente a la programación para la visualización tridimensional una vez calculadas ambas imágenes deben visualizarse empleando la instrucción `glDrawBuffer` pasando el argumento correspondiente. Para indicar que la imagen a dibujar es la asociada al ojo izquierdo se debe emplear el argumento `(GL_BACK_LEFT)`, y el argumento `(GL_BACK_RIGHT)` correspondientemente para indicar la imagen asociada al ojo derecho.

8.2. Diseño de la aplicación

Se optó por emplear el sistema operativo Linux ya que ofrece grandes capacidades de desarrollo de este tipo de aplicaciones de manera integrada, sin tener que recurrir a costoso *software* cuya funcionalidad es en esencia la misma.

Para construir la interfaz gráfica se empleó la herramienta de desarrollo Qt [11] de la compañía trolltech, ya que esta permite crear en poco tiempo interfaces funcionales, además ofrece un completo soporte de la OpenGl y al ser de código libre, puede usársele de manera gratuita si la aplicación no se realiza con fines comerciales, y también ofrece soporte para que la misma aplicación pueda ser compilada y ejecutada bajo el sistema operativo Windows en caso de ser necesario.

La aplicación diseñada tiene la estructura mostrada en la Fig. (8.2)

A excepción de la etapa de movimiento del objeto rígido, todas las demás etapas se realizan para cada triángulo del modelo deformable. Toda la información relacionada con la malla (vértices, aristas, etc.) se almacena en estructuras, por lo que cada etapa funciona de manera autónoma sin establecer comunicación directa con las demás, si no que accede directamente a las estructuras de datos para tomar los valores que requiere, depositando en ellas sus resultados.

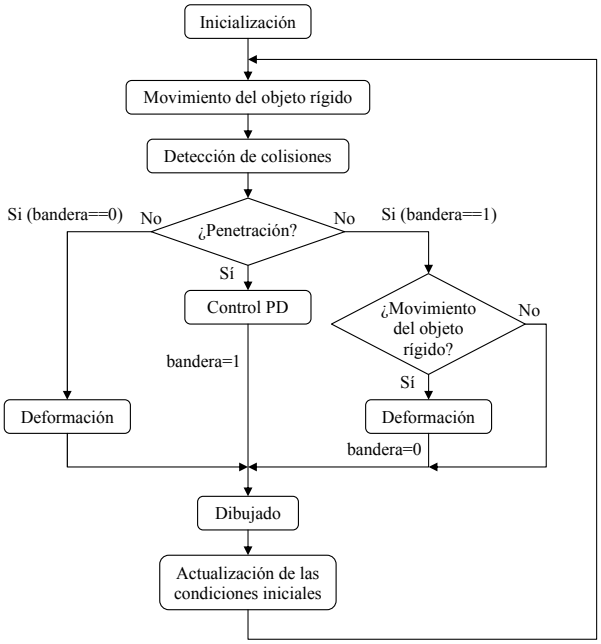


Figura 8.2: Diagrama de flujo de la aplicación.

A continuación se describen cada uno de los bloques que conforman la aplicación.

8.2.1. Inicialización

Esta fase consiste en la creación de los objetos deformables así como la inicialización de los dispositivos de *hardware*. Las etapas y el orden que se sigue es el siguiente:

1. Inicialización del hardware para la visualización tridimensional.
2. Inicialización de los guantes sensorizados.
3. Creación de los objetos deformables empleando mallas de simplejos.
4. Reordenamiento de las listas doblemente enlazadas que contienen la información de los vértices, para que puedan manejarse como arreglos.
5. Inicialización de los objetos de SOLID para la detección de colisiones.
6. Inicialización de la interfaz gráfica y los objetos para visualización.

8.2.2. Movimiento del objeto rígido

Los objetos rígidos se controlan por medio del guante sensorizado, el cual funciona de manera asíncrona. Cada vez que se requiere actualizar las posiciones de los objetos rígidos se realiza una consulta a través de las funciones del controlador del guante. El

guante entrega los valores que presentan sus sensores a la aplicación la cual los emplea para colocar en la posición correspondiente los objetos rígidos, esta posición se guarda en estructuras para ser accedidas por las demás etapas del sistema. Se emplean cinco esferas, cada esfera se asocia a cada dedo del guante. El guante proporciona valores dentro del rango $[0, 255]$ este valor se emplea para posicionar el objeto rígido correspondiente dentro de una trayectoria predefinida. Los valores extremos (0 y 255) corresponden a la posición del objeto en los límites de la trayectoria, los valores intermedios colocan el objeto en una posición proporcional al valor dentro de la trayectoria.

8.2.3. Deformación

Esta etapa consiste en resolver las ecuaciones diferenciales empleando el método de diferencias finitas y el modelo de resortes independientes para cada coordenada descrito en la Ec. (6.3) en la Pag. (57). Se realizaron pruebas comparativas para analizar de forma práctica este modelo en comparación con el propuesto en [27] resultando ser un modelo que produce simulaciones más realistas además de que se pudo implementar para operar en tiempo real sobre una malla de densidad media. Esta etapa se realiza en dos fases, la primera calcula la posición de los vértices dadas las posiciones anteriores, estos valores se almacenan en una estructura temporal. La segunda fase consiste en simplemente actualizar los valores de los vértices. La necesidad de estas dos etapas se da dado que no deben mezclarse posiciones anteriores con las que se van calculando. Sin embargo el usar dos etapas implica tener que realizar un doble *barrido* sobre los vértices lo cual tiene una complejidad $2n$ para n vértices. Las posiciones de los vértices se almacenan en estructuras.

Se ajustó el tamaño de paso de discretización en relación a la frecuencia del sistema (30 hertz). Colocándose un paso de $\Delta t = \frac{1}{30}$ segundos.

8.2.4. Detección de colisiones

Esta consiste en realizar pruebas sobre los triángulos contra los objetos rígidos. Se escogió el modelo de triangulación de las caras que no emplea el centroide, por contener menos triángulos y dado que solo emplea los vértices de la malla de simplejos evitando la necesidad de calcular y añadir a la malla el centroide.

Se implementaron dos esquemas de detección, uno de ellos consistía en realizar consultas de forma exhaustiva para cada pareja triángulo-esfera. El otro enfoque consistió en emplear la jerarquización de los objetos contenedores proporcionada por SOLID. Para su uso deben agregarse los objetos SOLID creados con `DT_CreateObject` a *escenas*. Esto se logra mediante la función `DT_AddObject`, además cada objeto agregado debe asociarse a una clase, a la cual se le asigna un tipo de respuesta a una colisión y una función que se llamará cuando se detecte alguna colisión. Una vez añadidos los objetos a la escena correspondiente se emplea una prueba por colisiones global `DT_Test`, la cual llama a las funciones asociadas a la clase a la que pertenece el objeto del que se detecta una colisión, esto se repite de manera automática para todos los objetos que se encuentran en contacto con otro. Una vez terminadas todas las llamadas a las respectivas funciones, `DT_Test` re-

gresa un valor de retorno indicando cuantas colisiones se detectaron. Dado que los puntos de los objetos que se pasan a SOLID son directamente accesibles por la biblioteca, esta funciona de manera autónoma.

Se verificó de forma práctica que para una malla de densidad media el uso de escenas introduce un retardo inaceptablemente grande, especialmente cuando los cinco objetos rígidos hacían contacto con el objeto deformable. Por ello se empleó finalmente la prueba exhaustiva entre los objetos la cual resultó más rápida.

Esta etapa incluye también el bloque de decisión donde dependiendo si existe o no penetración, el programa toma tres caminos diferentes. En caso de presentarse penetración, esta debe ser corregida, lo cual se lleva a cabo mediante el uso del control PD. Si no se presenta penetración existen dos opciones los cuales dependen de una variable llamada `bandera` esta, como puede verse del propio diagrama, esta activa (tiene el valor uno) cuando el objeto deformable se haya en contacto pero sin penetración con el objeto rígido. En cuyo caso, si no se presenta movimiento del objeto rígido no se realiza deformación. Esto permite mantener un contacto superficial entre el objeto rígido y el deformable. En cualquier otro caso se realiza la deformación normal en el objeto.

Se utilizó el vector de profundidad de penetración dado que permitió obtener una simulación realista y en tiempo real.

8.2.5. Control numérico

El control numérico empleado es el del tipo proporcional-derivativo. Para su uso debe proporcionarse una entrada, que será la posición a la que se desea mover el vértice en cuestión, y el control proporciona como salida la posición a la que debe moverse. El control solo se usa sobre los vértices que se corrigieron luego de una colisión, los demás vértices se mueven de forma libre.

8.2.6. Actualización de las condiciones iniciales

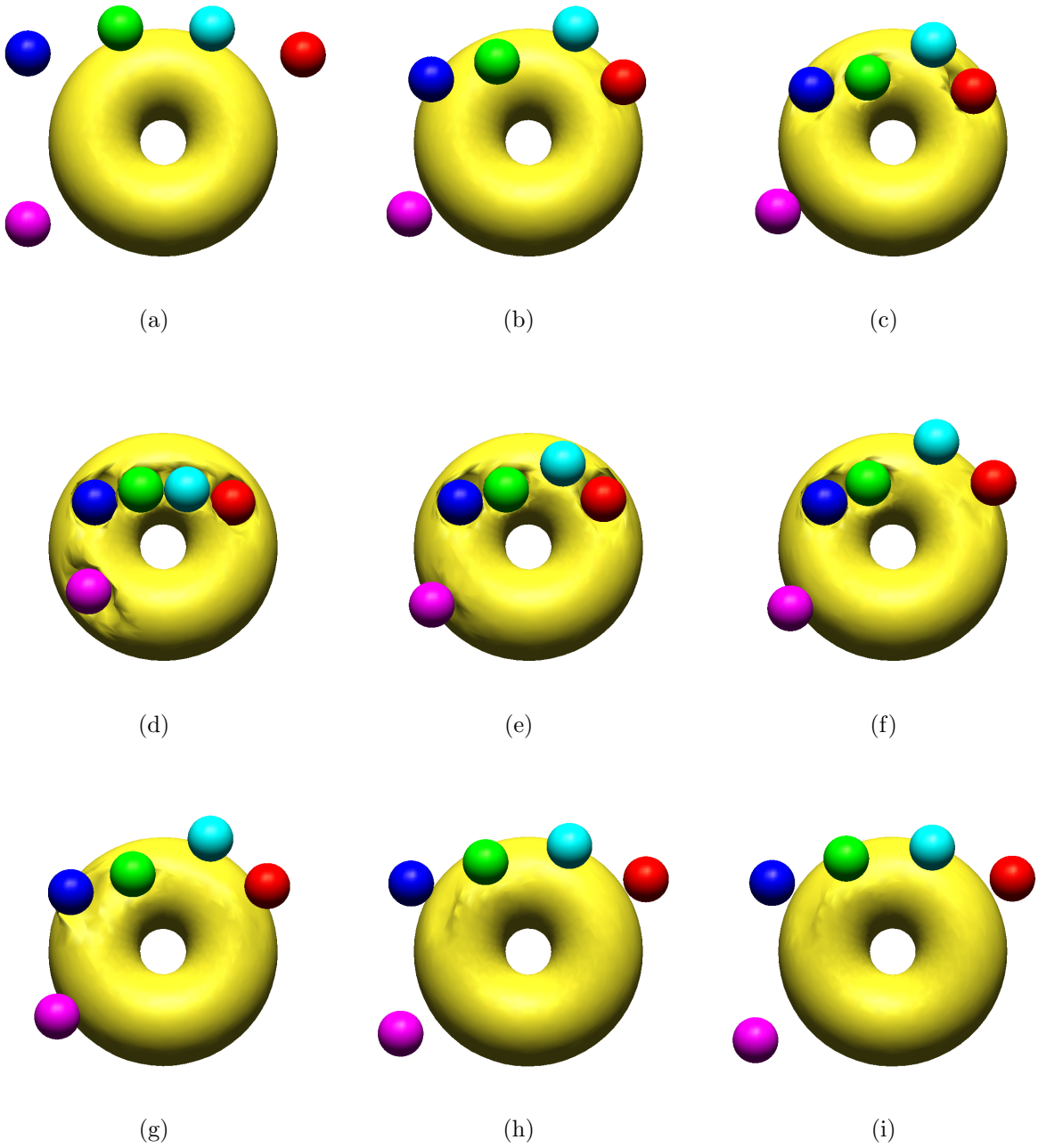
Esta etapa tiene la función de implementar un comportamiento plástico sobre el modelo deformable, cuando se aplica, los puntos reciben como condiciones iniciales su posición actual, por lo que a partir de ese momento ya no se mueven quedando estáticos en su nueva posición.

8.3. Resultados obtenidos

En la Fig. (8.3) se presentan las imágenes correspondientes al uso del sistema con el guante sensorizado. Las imágenes corresponden al resultado de cerrar la mano completamente para luego abrirla, así los objetos rígidos se acercan al modelo deformable colisionan con él, los deforman y luego se alejan.

Se empleó un toro deformable de densidad media consistente de 1860 vértices, 2790 aristas y 930 caras; y cinco esferas rígidas controladas por el guante sensorizado. Se logró una animación en tiempo real a treinta cuadros por segundo.

Para un mayor número de muestras el sistema tiende a tener un retardo muy alto que impide interactuar en tiempo real, al igual que si se aumenta la resolución de la malla de simplejos. Para menores resoluciones la simulación tiende a verse poco realista por el gran tamaño de los triángulos en relación con el tamaño de los objetos, los cuales se hacen demasiado notorios.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 8.3: Deformación de un toro deformable. Los objetos rígidos se acercan al toro deformable para luego alejarse.

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

En esta tesis se construyó un prototipo para deformar interactivamente un cuerpo a través de un guante sensorizado.

Las principales características de la aplicación son:

1. La interacción con el modelo deformable se realiza en tiempo real, a 30 marcos por segundo.

2. Se usó un sistema mecánico simple sobre cada arista del modelo deformable para dar su deformabilidad al modelo de una forma realista.

3. Se empleó un esqueleto para especificar el interior del toro, con esto fue posible obtener su modelo construido a base de mallas de simplejos. Por lo que se comprobó que el uso de un esqueleto permitiría obtener modelos de objetos más complejos mediante mallas de simplejos.

4. Se empleó el método numérico de diferencias finitas, el cual es barato computacionalmente. Fue posible obtener una deformación en tiempo real.

5. Se verificó que la biblioteca de detección de colisiones SOLID es eficiente y permite una detección de colisiones en tiempo real para objetos complejos.

6. Se verificó que el uso del vector de profundidad de penetración proporciona simulaciones en tiempo real y con un realismo aceptable.

7. También se comprobó que el uso de triángulos para describir la superficie de la malla de simplejos es adecuada para la detección de colisiones.

8. En la aplicación final se incorporó visión estereoscópica, usando unos lentes activos con un monitor que permite un refresco de 118 Hz y una tarjeta NVIDIA quadro (modelo FX1400).

9.1.1. Contribuciones

Si bien existen muchos sistemas que permiten la deformación de un objeto, las características del sistema aquí presentado son una combinación con características destacables. El uso de mallas de simplejos no esta muy extendido y este trabajo presenta una forma de interactuar en tiempo real con objetos que emplean esa representación.

Con respecto al trabajo más parecido a este, el presentado en [28], se añadieron las capacidades para deformar el sistema en tiempo real, además la incorporación de visión estereoscópica y el uso de guantes sensorizados proporcionan un sistema más completo para la manipulación de los objetos.

9.1.2. Limitaciones

El sistema presentado permite interactuar con modelos de densidad media, sin embargo dado el costo de los algoritmos no es posible emplear mallas mas densas en el sistema en tiempo real.

La detección de colisiones discreta solo permite la interacción con objetos cuya velocidad sea relativamente baja sin embargo el costo computacional es considerablemente bajo cuando se emplea este esquema.

Se emplearon cinco esferas para interactuar con el modelo deformable puesto que son objetos simples, sin embargo el uso de objetos más complejos tendrá un costo computacional mayor pudiendo impedir una respuesta en tiempo real.

9.2. Trabajo futuro

La detección de colisiones implementada es del tipo discreta, que aunque ofrece simulaciones adecuadas bajo ciertas características, en un panorama más amplio resulta inadecuada por lo que sería conveniente utilizar una detección a través del tiempo.

El uso del vector de profundidad de penetración es una aproximación que resultó ser adecuada para este trabajo, sin embargo otros enfoques pueden mejorar el realismo de la simulación.

Si bien los resultados obtenidos fueron satisfactorios, el modelo matemático empleado es simple, por lo que podría analizarse un modelo más complejo que considere aspectos como la fuerza de giro mediante el uso de resortes torsionales, que aunque en principio será más costoso computacionalmente, deberá permitir una interacción en tiempo real.

El uso de un sistema háptico permitiría sentir las características del objeto. Pero se requeriría emplear un modelo de contacto con fuerzas de retroalimentación, los cuales son más complejos que los usados, por lo que sería necesario implementar algoritmos de resolución numérica de las ecuaciones que fueran eficientes.

Para tener un sistema mucho más completo podrían añadirse capacidades de ruptura y unión de los modelos deformables, así como una posible interacción entre diversos modelos deformables.

Apéndice A

Manual de usuario de la aplicación gráfica

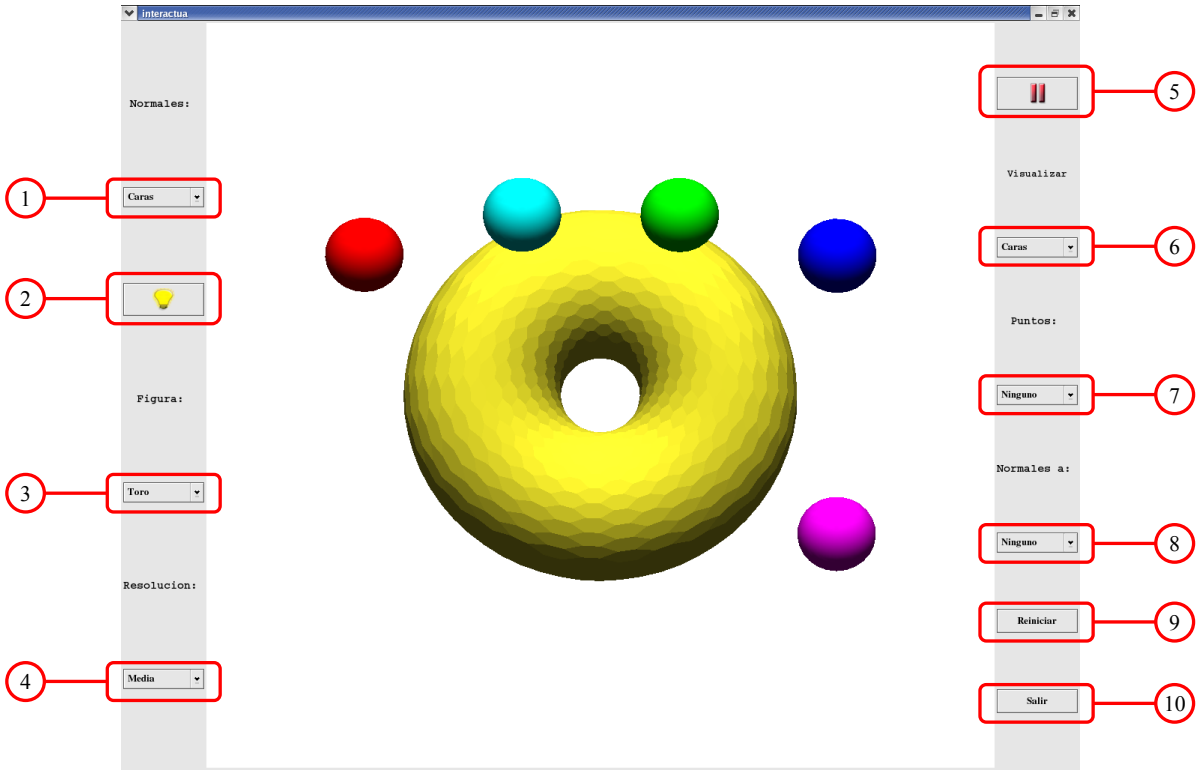


Figura A.1: Interfaz gráfica.

La Fig. (A.1) muestra el aspecto de la interfaz gráfica de la aplicación. A continuación se describirán las funciones de los diferentes botones que la componen.

1. Cortina de selección (que permite escoger uno de varios parámetros predefinidos) correspondiente al tipo de normales que se calcularán para la visualización, se tienen

tres valores predefinidos: normales a las caras, normales individuales a cada vértice, normales a los triángulos.

2. Botón que sirve para activar y desactivar la iluminación.

3. Cortina de selección para establecer el objeto visualizado, se tienen tres posibilidades: toro, cilindro y esfera.

4. Cortina de selección para establecer la densidad de la malla empleada, en la que se presentan tres valores: densidad baja, media y alta.

5. Botón para iniciar y detener la aplicación.

6. Cortina para establecer el tipo de superficie a visualizar, se presentan las opciones: caras, aristas y triángulos. Solo la opción caras visualiza propiamente una superficie, las opciones aristas y triángulos muestran una malla de líneas, aristas muestra las aristas de la malla de simplejos y triángulos muestra las aristas de la malla de triángulos.

7. Cortina que permite visualizar los vértices de la malla, los centroides a cada cara o bien ambos.

8. Cortina que permite visualizar las normales a las caras, las normales a los vértices o ambas.

9. Botón etiquetado *Reiniciar*, sirve para regresar la malla a su posición inicial, previo a cualquier deformación.

10. Botón *Salir* cuya función es terminar la aplicación.

Bibliografía

[1] U. M. Ascher and Linda Ruth Petzlod. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, 1998.

[2] R. T. Azuma. A survey of augmented reality. *Presence : Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[3] J Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computers Methods in Applied Mechanics and Engineering*, 1(2–3):1–16, 1972.

[4] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation 2005*, pages 219–228, July 2005.

[5] Michael Cline and Dinesh Pai. Post-stabilization for rigid body simulation with contact and constraints. In *ICRA*, pages 3744–3751. IEEE, 2003.

[6] Thomas F. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2001.

[7] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142, New York, NY, USA, 1993. ACM Press.

[8] John D'azzo and Constantine Houpis. *Linear Control System Analysis and Design*. Mcgraw Hill, 1963.

[9] 5DT DataGlove Driver. http://sourceforge.net/projects/driver5dt/.

[10] Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, 2005.

[11] Qt: The Cross-Platform C++ Developement Framework. http://www.trolltech.com/products/qt.

[12] Sarah F. Frisken Gibson. 3D ChainMail: A fast algorithm for deforming volumetric objects. In *Symposium on Interactive 3D Graphics*, pages 149–154, 1997.

[13] Ben Houston, Michael B. Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. Hierarchical rle level set: A compact and versatile deformable surface representation. *ACM Trans. Graph.*, 25(1):151–175, 2006.

[14] Chen Hui, Sun Hanqiu, and Jin Xiaogang. Interactive haptic deformation of dynamic soft objects. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 255–261, New York, NY, USA, 2006. ACM Press.

[15] Fabrice Jaillet, Behzad Shariat, and Denis Vandrope. Deformable object reconstruction with particle systems. *Camputers and Graphics*, 22(2–3):189–194, 1998.

[16] Paul Kry and Pai Dinesh. Continuous contact simulation for smooth surfaces. *ACM Transactions on Graphics*, 22(1):106–129, January 2003.

[17] Benjamin C Kuo. *Sistemas de Control Automático*. Prentice Hall, 1996.

[18] William R. Mark, Scott C. Randolph, Mark Finch, James M. Van Verth, and Russell M. Taylor II. Adding force feedback to graphics systems: Issues and solutions. *Computer Graphics*, 30(Annual Conference Series):447–452, 1996.

[19] Kevin T. McDonnell, Hong Qin, and Robert A. Wlodarczyk. Virtual clay: a real-time sculpting system with haptic toolkits. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 179–190, New York, NY, USA, 2001. ACM Press.

[20] J. Motagnat, H. Delignette, and N. Ayache. A review of deformable surfaces: topology, geometry and deformation. *Image and Vision Computing*, 19(14):1023–1040, December 2001.

[21] Takashi Nagata. Variable-gain constraint stabilization for general multibody systems with applications. *Journal of Vibration and Control*, 10:1335–1357, 2004.

[22] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide*. Addison Wesley, 2004.

[23] Jinah Park, Sang-Youn Kim, Seung-Woo Son, and Dong-Soo Kwon. Shape retaining chain linked model for real-time volume haptic rendering. In *Symposium on Volume Visualization and Graphics*, pages 65–72, 2002.

[24] Shahram Payandeh, John Dill, and Jian Zhang. A study of level-of-detail in haptic rendering. *ACM Trans. Appl. Percept.*, 2(1):15–34, 2005.

[25] Jorge Eduardo Ramírez Flores. Modelos deformables para caracterizar macromoléculas biológicas. Master's thesis, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, 2004.

[26] Jorge Eduardo Ramírez Flores and Luis Gerardo de la Fraga. Basic three-dimensional objects constructed with simplex meshes. In *First International Conference on Electrical and Electronic Engineering*, pages 166–171, Acapulco, México, September 2004.

[27] Claudia Magdalena Ramírez Trejo. Animación de modelos deformables. Master's thesis, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, 2005.

[28] Claudia Magdalena Ramírez Trejo and Luis Gerardo de la Fraga. Animation of deformable objects built with simplex meshes. In *Proceedings of the 2nd International Conference on Electrical and Electronics Engineering*, pages 36–39, México City, September 2005.

[29] Singiresu Rao. *Mechanical Vibrations*. Addison Wesley, 1995.

[30] Robert Resnick, David Halliday, and Kenneth Krane. *Física*. CECSA, 1998.

[31] R. Ruddle, S. Payne, and D. Jones. Navigating largescale virtual environments: What differences occur between helmet-mounted and desk-top displays, 1999.

[32] M Rydmark, T Kling-Petersen, R Pascher, and F Philip. 3d visualization and stereographic techniques for medical research and education. *Stud Health Technol Inform*, 81:434–439, 2001.

[33] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles. Haptic rendering: programming touch interaction with virtual objects. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 123–130, New York, NY, USA, 1995. ACM Press.

[34] Lin Shin-Tin and Huang Jiann-Nan. Numerical integration of multibody mechanical systems using baumgarte's constraint stabilization method. *Journal of the Chinese Institute of Engineers*, 25(2):232–252, 2002.

[35] Dave Shreiner. *OpenGL Reference Manual*. Addison Wesley, 2004.

[36] Morris Tenenbaum and Harry Pollard. *Ordinary Differential Equations*. Dover Publications, 1963.

[37] Octave: un lenguaje de alto nivel para cálculo numérico. http://www.gnu.org/software/octave/.

[38] Gino van den Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2004.

[39] Han wool Choi, Hee joon Kim, Jeong in Lee, and Young-Ho Chai. Free hand stroke based virtual sketching, deformation and sculpting of nurbs surface. In *ICAT '05: Proceedings of the 2005 international conference on Augmented tele-existence*, pages 3–9, New York, NY, USA, 2005. ACM Press.

[40] Florence Zara, François Faure, and Jean-Marc Vincent. Physical cloth simulation on a pc cluster. In *Parallel Graphics and Visualization*, 2002.

[41] Dennis G. Zill. *Ecuaciones diferenciales con aplicaciones de modelado.* Thomson Editores, 2005.