Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Planeación de Movimiento para Avatares.

Tesis que presenta:

Cristian Eduardo Boyain y Goytia Luna

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Diciembre de 2009.

164548-1001

Planeación de Movimiento para Avatares.

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Cristian Eduardo Boyain y Goytia Luna

Ingeniero en Computación
Universidad Autónoma de Zacatecas 2001-2006

Becario de CONACYT, expediente no. 212740

Director de Tesis

Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Diciembre de 2009.

Resumen

En esta tesis se presenta un algoritmo de planeación de movimientos que nos permite generar animaciones autónomas para avatares, de una forma rápida y sin necesidad de utilizar equipo especializado, como el de captura de movimiento.

El principio es simple, se buscan de forma aleatoria posibles acciones en el espacio de configuraciones posibles de las articulaciones de los avatares. Con estas acciones se forma una estructura en forma de árbol que se utilizará para encontrar un plan de acciones, el cual nos lleve de la configuración inicial que será la raíz del árbol, hasta una configuración final la cual está representada por una hoja de una rama del árbol. El espacio de búsqueda de estas acciones aleatorias es discretizado y sólo acciones que generen movimientos que no excedan los límites de las articulaciones del avatar son agregadas al árbol. Para poder determinar cual rama del árbol representa el mejor plan de movimientos, se utilizó la técnica de gradiente descendente sobre todos los nodos del árbol. El gradiente también es utilizado para guiar el proceso aleatorio sobre la parte del espacio de búsqueda que pueda llevar al objetivo en menos movimientos.

Lo novedoso de nuestra propuesta es que se elimina el uso de técnicas de cinemática al utilizar una base de conocimientos diseñada para la animación. Esta base de conocimientos es utilizada para consultar los rangos de movimiento de las articulaciones de los avatares El algoritmo propuesto puede utilizarse para animar todos aquellos avatares que tengan su información almacenada en la base de conocimientos.

Para probar nuestra propuesta se implantó un caso de estudio en el que se muestran los planes de movimientos calculados para que un avatar de tipo humano pueda alcanzar un objeto que puede estar en diferentes lugares.

This thesis present a motion planning algorithm useful to animate autonomously human type avatars. This proposal alleviates the need to use specialized equipment such as motion capture.

The principle of our proposal is to look randomly new possible actions in a possible configuration space of the avatar's articulations. The result of this research is used to build up a tree structure. This tree is used to find a plan taking us from an initial state (the root) to the desired final state (a leave in the tree). To build the tree the search space is discretized in such a way to include only actions do not exceed the articulation's limits. In order to choose the best branch of the tree to build the plan, a gradient descent technique is used. This technique is used also to find the plan with a minimum of steps.

Our proposal eliminates the use of cinematic techniques using a knowledge base containing information regarding avatar's articulations.

A study case was implemented to prove our proposal. The study case considers a human type avatar that must reach a final objective that can be placed in different positions.

Agradecimientos

A Dios por regalarme la vida y el tiempo para concluir con este trabajo.

A mis papás Jorge e Isidra por su apoyo, orientación y cuidados durante toda mi vida.

A mi hermano Ivan por su compañía y amistad.

A mis compañeros y amigos porque sus consejos y compañía hicieron más ameno mi trabajo.

A mi asesor, el Dr. Felix Ramos por su amistad, guía y apoyo, con los cuales fue posible realizar este trabajo de tesis.

Al CINVESTAV por ser una institución donde uno se puede preparar para afrontar mejor los retos de la vida.

Al CONACYT por otorgarme la beca que permitiera continuar con mi desarrollo académico y personal, al realizar este trabajo de tesis.

Indice general

Índice de tablas

Indice de figuras

Capítulo 1

Introducción

Cada vez más, la realidad virtual [18] está siendo utilizada por el ser humano en actividades tales como entretenimiento (como es el caso de los videojuegos), información (como en algún noticiero en el que se utilizan pantallas interactivas para la recreación de alguna jugada de fútbol), entrenamiento militar (para la realización de tácticas en el campo de batalla), en la simulación de vuelos (entrenamiento de pilotos) o también en el caso de simulación de intervenciones quirúrgicas, simulacros de desastres y en la enseñanza escolar.

Debido al gran campo de aplicación que se presenta en la realidad virtual y su beneficio para la sociedad, el presente trabajo está enfocado hacia ella.

1.1. Descripción del problema

El ser humano siempre ha utilizado su imaginación, la cual le ha permitido entre otras cosas, el generar entornos irreales como los narrados en los libros de Julio Verne. Actualmente, con el avance y desarrollo de la tecnología es posible crear estos entornos a través del uso de sistemas de realidad virtual (SRV), los cuales están compuestos por una colección integrada de hardware y software especializado para producir experiencias en entornos que nos permiten representar ambientes reales o sintéticos de forma virtual [18], lo cual quiere decir que estos entornos no tienen una existencia física ni material o tangible, sino conceptual y abstracta, pero con una representación visual. Si bien, se ha logrado la representación de estos entornos de una manera virtual a través de estos sistemas, todavía hace falta lograr un mayor grado de realismo, ya que esta representación tiene poco realismo y una persona distingue fácilmente entre los entornos creados y los entornos reales debido a las inconsistencias de los primeros con los entornos reales, también debido a la falta de retroalimentación sensorial. Al lograr una representación más real de los entornos virtuales, se nos permite tener un mejor envolvimiento tanto físico como mental. Conforme se logren resolver estos problemas el uso de la realidad

virtual se volverá más y más común en nuestras vidas.

Uno de los principales problemas a resolver en la realidad virtual es mejorar la animación de las creaturas virtuales o avatares dentro de estos entornos, para que sea lo más real posible. En este trabajo un avatar es definido como sigue:

Definición 1.1 (Avatar). Entidad virtual autónoma o semiautónoma que interactúa en un entorno virtual, la cual puede ser representada con cualquier forma o apariencia física.

En el presente trabajo se plantea el problema de animación autónoma de avatares de tipo humano mediante algoritmos de planeación. La obtención de los movimientos necesarios para animar un avatar de este tipo se pueden calcular utilizando hardware especializado o a través de algoritmos de inteligencia artificial. El proceso de calcular movimientos realistas para avatares, siempre debe de considerar lo siguiente: la cantidad de articulaciones que se necesitan mover, el espacio de configuraciones para las distintas articulaciones involucradas, los grados de libertad de dichas articulaciones, que estas representan las restricciones físicas para la generación de dichos movimientos. Además se debe de tomar en cuenta que el tiempo necesario para realizar los cálculos debe ser suficiente para tener una ejecucion en tiempo real, esto es, los mecanismos propuestos deben de ser eficientes en tiempo y además útiles para animar diferentes tipos de avatares.

1.2. Motivación

Las animaciones de avatares que actualmente se aprecian en las industrias de simulación y de juegos de video llegan a ser bastantes reales. Sin embargo, presentan la desventaja de que son predefinidas, lo cual resta realismo en el desarrollo de una escena. Por otro lado, estas animaciones son generadas mediante técnicas de diseño gráfico o utilizando captores de movimiento, lo cual resulta costoso en tiempo y dinero (costo del equipo). Desde nuestro punto de vista, el estado del arte actual en planeación dinámica permite diseñar algoritmos eficientes que nos permitan proponer algoritmos útiles en la animación autónoma de avatares.

1.3. Objetivo de la tesis

El objetivo de esta tesis es el desarrollar un algoritmo de planeación dinámica que permita la animación autónoma de avatares del tipo humano con mayor realismo y menor costo que los obtenidos mediante las técnicas utilizadas hasta hoy. El algoritmo de planeación dinámica deberá de manejar todas las articulaciones necesarias para calcular el movimiento deseado, respetando las limitaciones físicas de un humano para evitar la generación de movimientos irreales.

1.4. Estructura de la tesis

El presente trabajo de tesis está organizado de la siguiente manera: en el capítulo 2 se mostrarán algunos de los trabajos de investigación sobre planeación y areas relacionadas con la animación de avatares. En el capítulo 3 se propone un algoritmo de planeación de movimientos de avatares. En el capitulo 4 se muestra el caso de estudio en donde el algoritmo de planeación es implementado.

Capítulo 2

Estado del arte

El término planeación puede significar diferentes cosas, dependiendo del área o campo del conocimiento en que se esté hablando. Por ejemplo, en robótica se refiere al problema de determinar la traslación y rotación de un sistema mecánico para que este pueda realizar alguna tarea que se le solicite (cumpliendo las restricciones que implique esta tarea), en control al hablar de planeación de movimientos se puede referir a la construcción de señales de entrada para un sistema dinámico no lineal que nos lleve de un estado inicial a un estado objetivo [10], y finalmente en inteligencia artificial es la obtención de una secuencia apropiada de acciones, las cuales son aplicables para poder realizar una tarea. La definición de planeación para este trabajo está más relacionada a como es manejada en inteligencia artificial, por lo tanto aquí se define planeación como:

Definición 2.1 (Planeación)**.** Es el calculo de un conjunto de acciones aplicables (elementos del espacio de búsqueda), que nos permita, a partir del estado actual en que se encuentre el sistema, llegar a un estado deseado.

De la definición anterior, cuando es necesario que un avatar o un robot efectúe una tarea de forma autónoma, se puede utilizar la planeación.La planeación permitirá al avatar tomar en cuenta en todo momento su posición inicial, para calcular una trayectoria posible que lo lleve a la posición final deseada.

En la siguiente sección se mostrarán trabajos en los cuales la planeación es utilizada para lograr la animación de avatares o para el desplazamiento de robots.

2.1. Algoritmos de planeación usados en animación y robótica

Además de los algoritmos utilizados para la planeación como los son Dijkstra y A*, existen otros, los cuales ofrecen un mejor desempeño y resultado. Esto ha sido demostrado en varias de las ponencias presentadas en conferencia internacional ICAPS (International Conference on Automated Planning and Scheduling).

El algoritmo fast forward (FF) [20] fue el algoritmo campeón en la conferencia ICAPS del año 2000, el cual como su nombre lo indica, realiza una búsqueda hacia delante en el espacio de estados, utilizando una heurística que estima la distancia hacia su objetivo ignorando listas borradas, además de utilizar una estrategia de búsqueda que combina el uso de la técnica de ascenso de colinas (hill-climbing) con una búsqueda sistemática.

Otro algoritmo, cuya mejor cualidad es la eficiencia cuando se tiene un espacio de búsqueda muy grande y no se basa en ninguna heurística, es el algoritmo de rápida exploración en un árbol aleatorio (Rapidly-exploring Random Tree o RRT) [23] el cual utiliza una estructura de árbol para realizar su búsqueda donde la raíz es el estado inicial y realiza un muestreo aleatorio dentro del espacio de búsqueda para guiar el crecimiento del árbol hacia regiones inexploradas, debido a este mecanismo aleatorio, el algoritmo trabaja bien con el factor de ramificación del árbol y el espacio de configuraciones de grandes dimensiones. Sin embargo, no es capaz de replantear en caso de que algún plan llegue a ser inválido, además de que no es capaz de mejorar el plan durante la ejecución.

Estos algoritmos proporcionan la base de diferentes trabajos de investigación que se han realizado con el fin de mejorar los algoritmos sobre los que se basaron, o bien de implementarlos para desarrollar nuevas metodologías de planeación que permitan realizar alguna tarea en específico, como es el de trasladar o rotar un robot.

2.1.1. Planeación para el desplazamiento de robots

En [19] se presentan varios algoritmos en los que utilizan y mejoran el RRT para desplazar robots móviles, en donde primero se genera un RRT a partir del estado inicial utilizando el algoritmo Anytime RRT, este le asigna una variable al RRT generado, la cual representa su costo de creación, dicha variable es después usada como otro parámetro durante la fase de búsqueda, con el objetivo de poder generar soluciones de mejor calidad. En seguida, se genera otro RRT que deberá ser mejor que el anterior, así se eliminarán los nodos del árbol que no hayan contribuido con la solución y se volverá a hacer crecer el RRT hasta que el tiempo de planeación se termine o se haya alcanzado el objetivo. Si un cambio en el ambiente invalida un plan, entonces se ejecutará un segundo algoritmo llamado Dynamic RRT, el cual podará las ramas del árbol que hayan sido inválidas, así se construirá otro RRT a partir de

la parte válida del anterior. Con esto, el nuevo plan resultante es mejorado con la posterior ejecución del Anytime RRT.

Para la propuesta presentada en [1] se hace una combinación de los sistemas de colonias de hormigas (ant colony o ACO, por sus siglas en inglés) y algoritmos genéticos (GA, por sus siglas en inglés), para la elaboración de planes que se implementaron a robots. Al comenzar se libera una cantidad definida de hormigas, estas deben de ir recorriendo el ambiente y serán generadas de forma aleatoria, pero siempre controlando el tiempo de vida de cada hormiga. Debido a que ACO requiere de la determinación de varios parámetros, se combinó con la metodología de los algoritmos genéticos para determinar los valores más adecuados permitiendo poder obtener mejores planes. Si bien al mantener el número y el tiempo de las hormigas se controla el espacio de memoria requerido, la determinación adecuada de los parámetros sigue siendo un problema ya que pueden no ser los más adecuados y el uso de una propuesta evolutiva requiere de cierto tiempo antes de poder converger.

Otra metodología que solamente implementa algoritmos genéticos está dada por [5]. Aquí se presenta un algoritmo para elaborar planes que permitan a los robots atravesar diferentes tipos de terrenos. Cada individuo de la población es una ruta del estado inicial al estado final y cada cromosoma está compuesto por una posición (x, y) y un valor q que representa la calidad del terreno, así se puede determinar si el terreno se puede transitar por una posición válida. Además de las clásicas operaciones de cruza y mutación, se introduce la operación de apretamiento (tightening), la cual se encarga de hacer más corto el camino obtenido por las dos operaciones previas. Este trabajo encuentra una solución óptima, pero tiene la desventaja de que se requiere de un conocimiento previo del ambiente por donde transitarán los robots.

Los trabajos de investigación que hasta ahora se han presentado fueron aplicados al desplazamiento de robots, a continuación se muestran otros trabajos en los cuales el robot o avatar cuenta con una estructurada articulada, donde las articulaciones son utilizadas en la ejecución de alguna tarea.

2.1.2. Planeación de movimientos para una entidad con una estructura articulada

En [14] se utiliza un algoritmo genético para obtener planes aplicados a la animación de un brazo virtual. En este trabajo se utiliza la distancia Euclidiana entre el efector final del brazo y el punto que debe alcanzar como función de aptitud (fitness). El cromosoma de cualquier individuo de la población está compuesto por acciones que definen movimientos sobre alguna de las articulaciones del brazo. También se implementa un algoritmo de aprendizaje por refuerzo, el cual formará una base de conocimiento para que el algoritmo de aprendizaje pueda aprender movimientos que fueron previamente usados. El problema de esta metodología es

el tiempo que necesita el algoritmo genético, ya que es demasiado costoso como para implementarlo en un entorno dinámico, además de que, la implementación del algoritmo de aprendizaje no fue la mejor.Por lo cual, si el número de articulaciones incrementa, entonces el espacio de configuraciones posibles sería demasiado grande.

Otra metodología en la cual utilizan el método de colonias de hormigas (ACO), pero ahora junto con el algoritmo de mapa de rutas probabilístico (Probabilistic RoadMap o PRM, por sus siglas en inglés) es presentada en [13]. Aquí, un número de hormigas artificiales son liberadas desde el nido, donde el nido es la configuración inicial de un brazo robótico, y a continuación se realiza una búsqueda hacia delante para obtener la comida, es decir, la configuración final. Conforme se mueve, cada una va generando un camino por el cual va dejando un rastro de feromonas, luego esta información junto con la distancia a la que se encuentre este rastro es almacenada. Esta distancia es determinada en base a la distancia de cada articulación entre la configuración inicial y la configuración obtenida aleatoriamente. Un segundo grupo de hormigas es liberado pero en esta ocasión a partir del nido, de donde aleatoriamente irán generando caminos. Los nuevos caminos generados por el segundo grupo son cotejados con los almacenados previamente para comprobar si existe un camino de cada grupo de hormigas que puedan ser conectados. Si un camino entre el nido y la comida puede ser formado con algún camino de cada grupo de hormigas, esto nos dará el plan para poder mover el brazo a partir de la configuración inicial hacia a la final.

La metodología anterior no puede ser implementada en un entorno dinámico ya que el proceso utilizado por ACO no reacciona de forma inmediata, aspecto que es contemplado y para el cual se presenta una propuesta para solucionarlo en [15]. En este trabajo es desarrollado un marco de trabajo en el que se obtiene un plan para el movimiento de un brazo, implementando un algoritmo de planeación junto con un algoritmo de control, esto porque el primero es un algoritmo computacionalmente complejo, el cual lo hace ineficiente para un entorno dinámico, en cambio el algoritmo de control es rápido y determina comand de movimiento en base a la señal de retroalimentación que recibe durante la ejecución del plan. Para mejorar el desempeño del algoritmo de planeación, se propuso delimitar el espacio de configuraciones de cada una de las articulaciones, representando este espacio como una burbuja, en donde una vez que se obtenga un plan inicial que atraviesa estas burbujas, si este plan requiere ser modificado, entonces se busca un plan homotópico a éste, dentro de los límites de movimiento de cada articulación.

El delimitar el espacio sobre el cual un algoritmo de planeación trabajará es un aspecto importante para reducir su complejidad. En [2] el área de trabajo que es el espacio de configuraciones posibles, la cual puede ser muy grande es reducida, a esta área se le llama, área fundamental y representa la parte del área de trabajo que en realidad puede ser utilizada por el robot debido a su cinemática y geometría. Durante la determinación de esta área fundamental, se capturan los obstáculos fundamentales, los cuales son aquellos obstáculos que son independientes de la dinámica del ambiente. Con esto se reduce el proceso de captura

de los obstáculos en el espacio de configuraciones y el tiempo de ejecución del algoritmo de planeación. Para lo cual, se utiliza un algoritmo de planeación de movimientos basado en comportamiento para un manipulador con varios grados de libertad (degrees of freedom o DOFs, por sus siglas en inglés). Esto es, se definen comportamientos básicos para formular los movimientos de manipulación, junto con reglas para escoger estos comportamientos. La desventaja de esta propuesta es el hecho de tener que mantener en memoria el espacio de configuraciones para cada área en donde se mueva el robot que implemente esta metodología.

De manera empleada en el trabajo anterior, en [21] se presenta un algoritmo de planeación de movimientos para un brazo, tomando en cuenta la configuración topológica del brazo y obstáculos para factorizar el espacio de búsqueda, reduciendo así la complejidad del algoritmo de planeación, el cual usa programación dinámica. Primero se factoriza el espacio de configuraciones en subdominios cuyos elementos compartan la misma forma topológica, es decir, aquellos elementos que deformen el brazo suavemente durante la transición de un elemento al otro, luego utilizando un algoritmo de búsqueda, como A*, sobre los subdominios, se puede obtener un plan para mover el brazo. Este trabajo presenta la desventaja de que se tienen que generar varios caminos homotópicos entre sí, y por lo tanto no puede ser aplicable en un entorno dinámico.

Todos los trabajos hasta ahora presentados en esta sección fueron algoritmos de planeación de movimientos aplicados a brazos que podían tener varios grados de libertad, pero ahora se mostrarán propuestas que fueron implementadas en estructuras más complejas como la de un humanoide o un avatar con forma humana.

Un algoritmo de planeación de movimientos para un robot humanoide es presentado en [22], la metodología implementa un algoritmo de control para mantener estable el movimiento de todas las articulaciones y un algoritmo de planeación de movimientos que se encarga de realizar los movimientos de todas las articulaciones del cuerpo para ejecutar una tarea autónomamente. Ya que es bastante complejo planear el movimiento de todo el cuerpo, se divide esta tarea para obtener un plan para mover el tronco del cuerpo y otros para cada una de sus extremidades, al obtener estos subplanes se genera un plan completo. El funcionamiento del algoritmo de planeación está basado en una cuadricula o rejilla. Cada postura es almacenada en una única celda, por cuestiones de espacio sólo se almacenan las posturas actual y siguiente, para diferenciar entre posibles estados que están en la cuadricula, se utiliza la técnica del gradiente descendente (gradient descent). El problema que surge aquí, es que mantener la información de los posibles estados siguientes ocupa mucho espacio en memoria.

Utilizando el algoritmo RRT en [29] se presenta un método en donde de acuerdo con las condiciones actuales del entorno es el número de grados de libertad que el planeador necesitará para obtener un plan. Durante la construcción del plan se toma el nodo del RRT que representa el estado actual para ser modificado, incorporando los grados de libertad

que serán involucrados, en seguida el estado actual y un conjunto de señales de control son utilizadas para generar un conjunto de posibles estados siguientes que deben de cumplir con las restricciones cinemáticas del humanoide y del generador de rutas. Con este conjunto de posibles estados y la métrica de la distancia global entre cada uno de estos, se genera aleatoriamente un nuevo estado, además de que se habilita la elección de la acción siguiente para llegar al siguiente estado deseado. Por último antes de agregar el nuevo estado al RRT se verifica si no existen colisiones. Esta metodología describe como el algoritmo RRT genera movimientos para todo el cuerpo, aunque el uso de la cinemática incrementa la complejidad computacional conforme se requiera utilizar más grados de libertad.

Con la propuesta anterior se logra obtener la animación de un avatar cuando tiene que realizar la tarea de caminar. En [6] se utiliza un planeador de movimientos que implementa la tarea de cargar un objeto durante el proceso de caminar del avatar. Este enfoque está basado en el análisis global de la tarea de acuerdo a tres principales restricciones: evasión de obstáculos en 3D, una locomoción creíble y la manipulación del objeto. Para cumplir con las principales restricciones de la tarea, primero se ejecuta un algoritmo de planeación el cual tomando en cuenta solo los grados de libertad de los pies y la cintura, obtendrá una ruta que será trasformada en trayectoria para el avatar. Como segundo paso, se implementa un control de locomoción, el cual está basado en la técnica de captura de movimiento y con el cual se obtiene la secuencia de caminado, para después obtener la manipulación del objeto utilizando un algoritmo de cinemática inversa, donde la cadena cinemática está formada por los brazos del avatar y el objeto cargado por él. Por último, se verifica la trayectoria obtenida para corregir colisiones que pudieran existir en la trayectoria obtenida. Aunque esta propuesta obtiene una buena animación, su implementación sigue haciendo uso de secuencias pregrabadas y además se requiere de muchos cálculos por parte del algoritmo de cinemática inversa, esto es, tanto el costo computacional como el tiempo son muy altos.

Una visión más general del trabajo requerido para generar planes de movimientos en tareas que involucren la navegación y el tomar y manipular objetos, es presentada en [8]. En esta propuesta, se implementa un planeador de pasos para la navegación, donde se construlle un árbol de búsqueda con un conjunto de posiciones posibles del pie utilizando un enfoque de programación dinámica directa, cone esto se busca la secuencia de pasos que nos lleven a la posición final deseada. Además, se implementa un método para generar una trayectoria dinámicamente estable para ejecutar los pasos del plan. Dicha trayectoria es suavizada al reemplazar puntos intermedios de ciertos segmentos que representan grandes cambios de orientación por líneas rectas. Para realizar la tarea de manipulación de objetos el planeador de movimientos requiere obtener tres planes: alcanzar, tomar y regresar los objetos. Pero debido al gran espacio de configuraciones por el número de grados de libertad, se implementa una variante del algoritmo RRT llamado RRT-Connect [28] para obtener estos planes. El problema con este trabajo es no contemplar su implementación en un entorno dinámico, por lo que si la trayectoria de pasos que se calculo queda invalida, no puede calcular otra

trayectoria alterna rápidamente.

En [12] se presenta una técnica que usa algoritmos de planeación de movimientos basada en mapa de rutas probabilísticas para controlar 22 grados de libertad de un personaje humano animado. El principal objetivo en este trabajo es la síntesis automática de movimientos de agarre para ambos brazos, que estén libres de colisiones, que mantengan el equilibrio y respeten los límites de las articulaciones. El modo general de funcionamiento del algoritmo es el generar una estructura de árbol, en la que cada nodo represente una configuración, con lo cual se reduce el espacio de búsqueda, luego con esta estructura de árbol se busca un plan. Ya que normalmente el árbol de configuraciones no tiene la configuración exacta que se desea, el plan proporciona la configuración más cercana que se haya encontrado durante el proceso de construcción del árbol, a partir de la cual se puede interpolar a la configuración deseada. Esto es necesario ya que esta configuración empata con agarres prediseñados. En dicho trabajo se obtienen animaciones reales, aunque todavía se requiere de trabajo de diseño al establecer agarres prediseñados y al utilizar secuencias de movimiento para realizar la tarea de caminado.

Normalmente un robot o avatar humanoide siempre requiere desplazarse y es por eso que muchos trabajos antes mencionados cubren esta tarea, pero siempre el caminar se hacía sobre un terreno plano, en cambio en [9] se enfocan a la animación del caminar de un robot en diferentes terrenos. Para realizar la animación del proceso de caminar, se establece el concepto de contacto y postura, en donde un contacto sucede cuando una extremidad del cuerpo hace contacto con el ambiente y una postura representa a un conjunto de contactos. De esta forma, se trabaja sólo con los grados de libertad que se necesiten para establecer o eliminar un contacto. Para caminar, un robot repite el proceso de generación de un paso, donde primero se establece que contacto se rompe o se crea a partir de la configuración y postura actual, luego se genera una nueva configuración y por último, se debe encontrar un camino que conecte la configuración actual con la nueva. Para llevar a cabo este paso, se utiliza el algoritmo de mapa de rutas probabilístico, el cual modifica las trayectorias de movimientos básicos con los que ya cuenta el robot, para generar el plan que conecte las configuraciones deseadas. La forma en que son generadas las transiciones entre configuraciones y el manejo de los DOF son los aspectos más importantes de este trabajo, aunque presenta dificultades si se requiere elaborar un plan para caminar en un terreno que sea plano.

En esta sección se presentó cómo es que la planeación se ha utilizado para que una entidad con una estructura articulada como lo es un avatar pueda realizar diferentes tareas de una forma autónoma y con esto conseguir una animación más real. Para conseguir la animación de los avatares. la planeación no es la única forma que existe, por lo que en la siguiente sección se presentará otra técnica utilizada para lograr su animación.

2.2. Animación utilizando captura de movimiento

La captura de movimientos es la técnica en la cual gracias a un equipo especializado se graba la secuencia de movimientos realizada por alguna persona, para que después sean implementados en la animación de avatares.

En [11] es presentada una metodología donde se diseñan movimientos para un robot humanoide, considerando el ritmo del movimiento humano capturado con la técnica de captura de movimiento. La información capturada para el movimiento humano es adaptada para el humanoide debido a que la cinemática y dinámica son diferentes, por lo que primero se segmenta el movimiento del actor humano en movimientos básicos para después utilizar un método que satisfaga las restricciones cinemáticas y además se deben de obtener los ajustes de estabilidad. Para obtener estos ajustes, el método de punto de momento cero (zero moment point) es utilizado. Enseguida, se realiza una evaluación de similitud que considera el ritmo para poder escoger la trayectoria final con el valor de similitud más grande.

En [24] se establece un método para que un avatar pueda navegar al utilizar un control de locomoción basado en la edición de secuencias de imágenes obtenidas por la captura de movimientos. Siendo las entradas de control la velocidad lineal y angular instantánea del caminado. Este método combina tres componentes para sintetizar la locomoción: Primero, se almacena una librería de movimientos con las muestras de movimientos captados; segundo, se representan las características en un espacio de control lineal para poder elegir los movimientos capturados que más satisfagan el estado de entrada, y tercero, los ciclos de locomoción son sintetizados al mezclar los movimientos seleccionados. De estos ciclos de sintetización, se obtienen posturas sucesivas para llevar a cabo el caminado del avatar.

Además de simplemente obtener secuencias de movimiento para después ser ejecutadas, la técnica de captura de movimiento también permite realizar otras tareas. En [26] se provee con un marco de trabajo que permite establecer límites de movimiento para las articulaciones. Para lo cual, se realizan mediciones de desempeño en sujetos humanos a través de la captura de sus movimientos, enseguida se convierte la información grabada en posturas de las articulaciones utilizando un campo de representación coherente de cuaterniones sobre el espacio de orientación de la articulación, es decir, se almacena la información del cuaternión que posea la articulación, por último, se obtiene una aproximación superficial implícita continua de los límites del espacio de orientación para el cuaternión, permitiendo obtener la posición valida más cercana para una posición de una articulación que trate de exceder los límites de movimiento.

Aunque la captura de movimientos permite captar el movimiento para la realización de diferentes acciones, se necesita determinar cómo es que estos se pudieran utilizar. En [25] se expone un método que permite que un avatar sea animado y controlado interactivamente, al poder construir políticas de control a partir de la información de movimientos capturada.

Con una discretización lo suficientemente densa del espacio de estados, la política obtenida es capaz de encontrar una secuencia óptima de acciones. De esta forma se puede seleccionar la secuencia de movimientos apropiados para obtener una reacción ante diferentes situaciones.

2.3. Discusión

Si lo que se desea es una animación autónoma y real, capaz de ser adaptada para una gran variedad de diferentes situaciones y tareas, un algoritmo de planeación es una opción adecuada, ya que libera de la necesidad de adquirir un equipo especializado como el necesario en la captura de movimiento. Dependiendo de cómo se implemente, el algoritmo de planeación podría generar secuencias de animaciones diferentes que al final lleven a la misma posición final a partir de la misma posición inicial, con esto se imitaría el hecho de que no todas las personas realizan la misma tarea de la misma forma, aumentando el nivel de realidad de la animación.

Aunque la planeación pueda ser utilizada para la animación de un avatar, se tiene que desarrollar una metodología que permita tratar y resolver los diferentes problemas que se puedan presentar, como lo son: el manejo del gran espacio de búsqueda y la capacidad de manejar un gran número de grados de libertad para que pueda ser usado en un entorno dinámico que no sea específico o particular de sólo una estructura articulada, es decir, que pueda ser utilizado para animar a cualquier tipo de avatar ya sea humano o animal o bien basado en una entidad ficticia.

Capítulo 3

Planeación dinámica de movimiento para avatares

En este capítulo se propone un algoritmo de planeación para calcular los movimientos de un avatar de manera autónoma, el cual está basado en el algoritmo de planeación Rapidly-exploring Random Tree (RRT) [23]. En el apéndice A se da una explicación del funcionamiento del RRT.

Primero se especificará en términos generales cuál es el problema que resuelve un algoritmo de planeación. después se establecen formalmente conceptos básicos sobre planeación y en seguida se proporciona una descripción del algoritmo que se propone.

3.1. Definición del problema de planeación

Ahora daremos una formulación del problema de planeación para alcanzar un objeto. El ambiente 3D es un entorno de trabajo W en el que pueden existir p obstáculos, los cuales están representados en el conjunto $O = \{o_i \mid i = 0, ..., p\}$.

Para cada avatar A que se encuentre en W. este posee j_m articulaciones que a su vez componen q_n grados de libertad (DOFs), los cuales trabajan sobre un espacio de configuraciones C_a, el cual tiene una dimensión n, que representa el número de DOFs. Una configuración en C_a, a la cual llamaremos *configuración del sistema* es de la forma $(q_1, ..., q_n)$ donde $\forall q_i : \{q_i \in C_a\}$ y cada q_i representa el valor del grado de libertad i-ésimo.

Definiremos al espacio de configuraciones C_{obst} como el conjunto de todas las *configuraciones del sistema*, donde uno o más elementos de O se intersectan con A, y $C_{libre} =$

$\{C_a/C_{obst}\}$, es decir, el conjunto de configuraciones del sistema en las que no existe ninguna colisión.

Lo que requerimos es mover el avatar a través del espacio de configuraciones C_{libre} a partir de su configuración inicial, para poder alcanzar algún objeto o desplazar al avatar.

3.2. Formalización de la Planeación

En esta sección se definen conceptos del algoritmo de planeación para la animación de un avatar. En este trabajo el objetivo primordial es que se puedan realizar movimientos semejantes a los de un ser humano.

A continuación se definen formalmente los conceptos de: plan, estado, acción, meta y espacio de configuraciones.

Un plan P es la secuencia de acciones utilizada para conectar la configuración inicial del sistema q_{sys}^{init} a la configuración final del sistema q_{sys}^{g}.

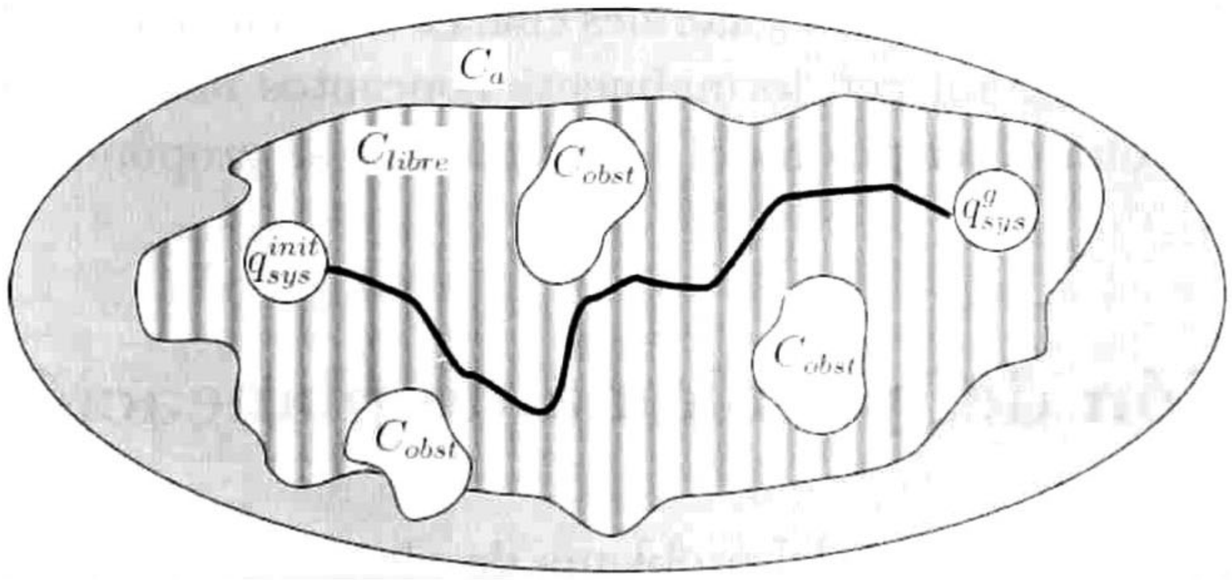


Figura 3.1: Componentes del plan de movimiento y su relación con el espacio de configuraciones C_a

Un estado $e_k = (k, q_k, \overrightarrow{v_k})$ en nuestro plan, representa la configuración actual para una k-ésima articulación, con una rotación representada por el cuaternión $q_k = (\Theta_k, \widehat{n}_k)$ (Θ_k representa el ángulo de rotación y $\widehat{n}_k$ expresa un vector unitario que indica la dirección de la rotación) y un vector de traslación $\overrightarrow{v_k}$ en $\mathbb{R}^3$ que indica la posición actual de la articulación.

Una acción $a_k = (k, q)$ representa el movimiento a realizar sobre una k-ésima articulación. donde $q = (\Theta, \widehat{u})$ es el cuaternión que representa la rotación que se aplicará sobre la articulación (Θ es el ángulo de rotación y $\widehat{u}$ representa el eje sobre el cual se rotará). Es posible

definir una acción usando el formato PDDL (Planning Domain Definition Language) de la forma siguiente:

$$a_k(k,q) \quad = \quad pre : estadoactual(e_k) \wedge estadolibre(e'_k)$$

$$pos : \neg estadoactual(e_k) \wedge estadoactual(e'_k) \wedge \neg estadolibre(e'_k)$$

El estado actual ($estadoactual(e_k)$) es un predicado para la configuración actual de una articulación y el estado libre ($estadolibre(e'_k)$) comprueba que no existan problemas con el estado alcanzable de la acción. Una meta g para un plan generado por el algoritmo consiste en llegar a una posición final $\overrightarrow{v}_{meta}$, la cual representa un punto alcanzable por el avatar dentro del espacio de configuraciones C_{libre}.

El movimiento del avatar dentro de C_{libre} involucra al conjunto de articulaciones $J = \{j_1, ..., j_m \mid j_i$ es una articulación del avatar $\}$ que serán utilizadas para la obtención del plan, $\|J\|$ no es necesariamente igual al número total de articulaciones que tiene A, porque no todas las articulaciones son necesarias para llegar a $\overrightarrow{v}_{meta}$.

Debido a que el espacio de configuraciones se vuelve cada vez más grande conforme el número de DOFs se incrementa, en nuestra propuesta C_a es discretizado para establecer los grados de movimientos $D_k = \{n \in \mathbb{N} \mid nmod_{\Delta p} = 0 \wedge (\Theta_{min} \leq n \leq \Theta_{max})\}$ que puede tener el DOF k, donde Θ_{max} indica el ángulo máximo, Θ_{min} el ángulo mínimo de rotación para el DOF, y Δp especifica la granularidad de los grados de movimiento para el DOF k. Con esta discretización, se disminuye el tamaño del espacio de configuraciones y se mantiene la distribución sobre todo el espacio de configuraciones.

Por lo tanto, el conjunto de todos los estados posibles S que el avatar puede ocupar en C_a está dado por:

$$S = \bigcup_{1=1}^{n} D_i$$

donde D_i es el conjunto de estados para cada DOF, es decir, indica el ángulo posible que puede adoptar el DOF y n indica el número de DOFs con los que se trabajará para obtener un plan.

Para dar una representación más entendible y delimitar los ejes de rotación en una acción, $\widehat{u}$ se especifica dependiendo del movimiento y orientación que genere sobre alguno de los siguientes DOFs: yaw (guiñada o attitude), pitch (cabeceo o heading), y roll (balanceo o bank), formando así un conjunto de acciones para cada DOF que posea cada articulación. Este conjunto de acciones estará definido por el grado de movimientos posibles que posee cada DOF

$$M = \{a_k(k,q) | \Theta \in D_i \wedge (\Theta_{min} \leq (\Theta_k - \Delta p) \ \wedge \ \Theta_{max} \geq (\Theta_k + \Delta p))\}$$

donde el valor de Θ_k representa el ángulo de rotación actual que tiene el DOF k sobre el cual queremos ejecutar la acción a_k y los valores de Θ_{min} y Θ_{max} fueron tomados del estudio quinesiológico realizado en [27], con lo cual aseguramos que nuestro algoritmo no ejecutará una acción que no pueda ser llevada a cabo por la estructura física del ser humano en tanto se haga una correcta asignación entre la información quinesiológica y los DOFs.

Ya que se definió el conjunto de acciones para cada DOF, el conjunto de todas las acciones posibles Ac se define como:

$$Ac = \bigcup_{i=1}^{n} M_i * 2$$

La multiplicación por dos se debe a la orientación sobre el DOF que se quiere realizar, es decir por cada DOF existen dos $\widehat{u}$, uno para rotar sobre el DOF en sentido de las manecillas del reloj y otro para el sentido inverso de las manecillas.

Como la definición de las acciones posibles en un plan dependen del avatar que las realizará, se utiliza una base de conocimientos (ontología) para obtener la información de la descripción cualitativa del avatar que implementa el algoritmo de planeación.

3.3. Ontología de los agentes CAPE

Se utiliza una implementación de la ontología descrita en [7] para almacenar la información de los avatares que pudieran implementar el algoritmo de planeación, ya que esta ontología está diseñada para la animación y permite compartir información semántica de avatares entre agentes basados en de conocimiento, como son los agentes CAPE [7], los cuales viven e interactúan en un entorno virtual dinámico 3D creado por una descripción declarativa [4] sobre la arquitectura de agentes GeDA-3D [3].
En la figura3.2 se presenta la definición sémantica del esqueleto para un avatar en la ontología.

3.4. Algoritmo de planeación de movimientos

A continuación se explica el algoritmo de planeación de movimientos para avatares. Este algoritmo es el siguiente:

El algoritmo 3.1 realiza una búsqueda de configuraciones posibles que le permitan mover las articulaciones J, generando una estructura de árbol con cada posible acción que encuen-

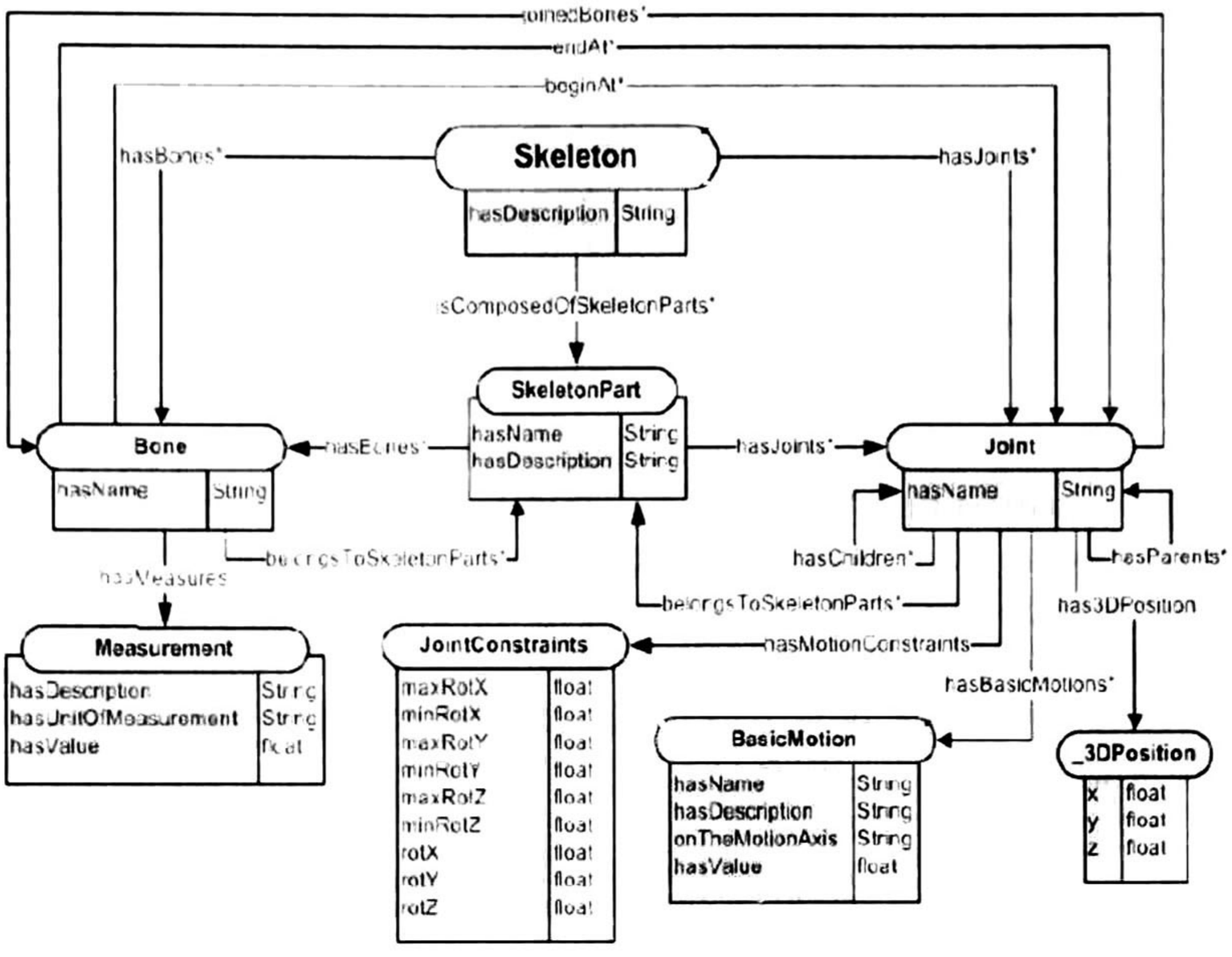


Figura 3.2: Definición del esqueleto para un avatar

Algoritmo 3.1 Planeación de movimientos para avatares

Entrada: Instancia de la ontología AvatarOntologia, el objetivo del plan, $\overrightarrow{v}_{meta}$ y las articulaciones involucradas J.

Salida: Plan de movimientos que permita llegar al objetivo partiendo de la configuración inicial del sistema

1: metaAlcanzada $\leftarrow$ **Falso**
2: rrt $\leftarrow$ motionPlanner(AvatarOntologia, $\overrightarrow{v}_{meta}$, J)
3: **Repetir**
4: **Si** gradientDescent(j_m) $\leq$ *umbral* **Entonces**
5: metaAlcanzada $\leftarrow$ **Cierto**
6: **si no**
7: rrt $\leftarrow$ crecerPlan(ultimaAcciónEjec)
8: ultimaAccionEjec $\leftarrow$ ejecutar(rrt)
9: **Fin Si**
10: **Hasta que** metaAlcanzada = **Cierto**

tra. El *rrt* es inicializado con la información de la ontología y la *configuraciones del sistema* inicial, de acuerdo con el avatar que se esté utilizando, además de que se establece la posición final a la cual debe de llegar la articulación $j_m \in J$, que en este algoritmo puede ser cualquiera de las siguientes articulaciones: la última articulación de las manos, de los pies o la articulación del abdomen, esta última para el caso que se necesite desplazar al avatar. Es importante definir que la última articulación es aquella que sus movimientos modifiquen al menor número de articulaciones y enlaces.

En seguida, se utiliza el método de gradiente descendente como heurística para comprobar si ya se ha logrado acercar suficientemente a la meta, es decir, si ya se ha cruzado un umbral definido. Si todavía no se ha conseguido la meta, se continua con la búsqueda de acciones para moverse, posteriormente se revisa el árbol de búsqueda que se forma con las acciones encontradas para encontrar cual sería la mejor secuencia de acciones hasta ahora encontrada y ejecutarse posteriormente por el avatar. Este proceso se repite hasta que se logre acercar la articulación j_m a la meta.

La forma en que se ejecutan las acciones del algoritmo dependerá del motor gráfico que se esté utilizando, pero la forma en que se hace crecer el árbol de búsqueda depende del algoritmo 3.2.

Algoritmo 3.2 crecerPlan

Entrada: La última acción del plan, que ha sido ejecutada
Salida: Crece el plan, al agregar nuevas acciones a partir de la última acción ejecutada

1: **Mientras** $i \leq x$ **Hacer**
2: $a_{rand} \leftarrow$ accionAleatoria(J)
3: **Si** esNulo(a_{rand})= **Falso** **Entonces**
4: agregarAccion(a_{rand}, ultimaAccionEjec)
5: **Fin Si**
6: $i \leftarrow i + 1$
7: **Fin Mientras**

Una vez que ya se inicializó el *rrt*, se buscaran acciones en C_a, cada acción x se obtendrá a través de un proceso aleatorio, como lo indica el algoritmo RRT original, después se comprueba si esta acción no es nula, porque puede ser el caso que el proceso aleatorio no haya podido generar un movimiento para el DOF que se escogió, ya que cualquier acción violaría las restricciones de movimiento establecidas en la ontología. Si se logra obtener una acción válida, esta se agregará a partir de la última acción del plan que fue ejecutada.

Aquí en el algoritmo 3.2 es donde se puede apreciar la principal diferencia de nuestra

implementación del algoritmo RRT, porque no requerimos tener todo el plan de acciones completo para empezar a realizar movimientos que nos lleven a nuestro objetivo, esta característica es útil en entornos dinámicos debido a que si por causas de algún obstáculo se inutiliza el plan que se estaba siguiendo, se puede construir otro alterno a partir de la última acción que se haya ejecutado.

En seguida en el algoritmo 3.3 se especificará cómo es que se obtiene cada acción x que se agregará al rrt.

Algoritmo 3.3 accionAleatoria

Entrada: El conjunto de todas las articulaciones involucradas, J
Salida: Acción generada aleatoriamente

1: **Repetir**
2: $\widehat{u} \leftarrow dofAleatorio(J)$
3: **Si** esNulo($\widehat{u}$) = **Cierto** **Entonces**
4: **Devolver** nulo
5: **Fin Si**
6: **Si** $gradientDescent(j_m) < umbral2$ **Entonces**
7: $\Theta \leftarrow ((\Delta p/2) * random[1, x_1])$
8: **si no**
9: $\Theta \leftarrow ((\Delta p/2) * random[1, x_2])$
10: **Fin Si**
11: $q \leftarrow (\Theta, \widehat{u})$
12: $a_k \leftarrow (k, q)$
13: **Hasta que** dentroRangoMov(a_k) = **Cierto**
14: **Devolver** a_k

Utilizando solamente las articulaciones J se obtendrá una acción aleatoria. Se escogerá el eje de rotación $\widehat{u}$ de alguno de los DOFs que posee $j_i \in J$, si j_i no tiene ningún DOF que esté dentro de los límites de movimiento, $dofAleatorio(J)$ regresará nulo y se terminará el algoritmo regresando una acción nula, en caso contrario, se determinará el eje de rotación para la nueva acción.

Para determinar el valor del ángulo de rotación Θ, se verifica el valor del gradiente actual para saber si ya ha pasado el $umbral2$, esto debido a que partir de cierto valor, un ángulo de rotación demasiado grande nos alejaría del objetivo en lugar de acercarnos, por eso se especifica obtener un número aleatorio, en el intervalo $[1, x_1]$ donde $x_1 < x_2$ cuando ya se pase el $umbral2$. Ya que tenemos todos los elementos para formar la acción que regresará el algoritmo, se verifica que esta acción con el valor aleatorio de Θ no exceda los límites de movimientos de la articulación sobre la cual se realizará el movimiento de rotación.

Con la acción aleatoria y el conocimiento de la última acción ejecutada, se puede obtener la acción que será agregada al rrt, este proceso es especificado en el algoritmo3.4.

Algoritmo 3.4 agregarAccion

Entrada: La acción a que se agregará y la última acción ejecutada
Salida: Acción a agregada

1: $a_{mejor} \leftarrow accionCercana(a_{rand}, ultimaAccionEjec)$
2: **Si** $gradientDescent(a_{mejor}) < umbral2$ **Entonces**
3: $\quad \Theta \leftarrow x_1$
4: **si no**
5: $\quad \Theta \leftarrow x_2$
6: **Fin Si**
7: $a_{rand} \leftarrow \Theta$
8: agregar(a_{mejor}, a_{rand})
9: explotarAccion(a_{rand})

Dada la acción a_{rand} y a partir de la última acción que ejecutó ($ultimaAccionEjec$), se buscará en la estructura de árbol la acción que se encuentre más cerca a esta acción aleatoria, porque es en esta en donde agregaremos la nueva acción que se obtendrá. Ya que sabemos donde agregaremos la nueva acción, se utilizará un criterio similar al descrito en el algoritmo 3.3 para determinar el tamaño del ángulo de rotación Θ, con la diferencia de que el valor del gradiente que se utiliza no sea el actual, si no el valor que se obtendría si se ejecutará la acción a_{mejor}. El nuevo valor de Θ reemplazará al que la acción a_{rand} tenía y así será agregada al rrt.

Por último, copiando el concepto del problema de exploración y explotación que se presenta en el área de aprendizaje por refuerzo, se explotará la nueva acción que acaba de ser agregada, este proceso se describe en el siguiente algoritmo 3.5.

Una vez más, se debe especificar el valor del ángulo de rotación para la nueva acción de la misma forma como en el algoritmo 3.4, después utilizando el gradiente se verifica que esta nueva acción nos este acercando al objetivo y esté dentro del rango de movimiento válido para poder ser agregada. El proceso de explotación de las nuevas acciones agregadas terminará hasta que el valor de la acción a explotar nos aleje de la meta. Con esto se logra obtener acciones válidas, de una forma más rápida, lo cual permite que los movimientos del avatar sean más reals y continuos.

En resumen, el funcionamiento del algoritmo de planeación se ilustra en la figura 3.3

Algoritmo 3.5 explotarAccion

Entrada: Acción a_{actual} para explotar
Salida: Acción a_{nueva}

1: **Si** $gradientDescent(a_{actual}) < umbral2$ **Entonces**
2: $\Theta \leftarrow x_1$
3: **si no**
4: $\Theta \leftarrow x_2$
5: **Fin Si**
6: $a_{nueva} = a_{actual}$
7: $a_{nueva} \leftarrow \Theta$
8: **Si** $gradientDescent(a_{nueva}) < gradientDescent(a_{actual})$ **Entonces**
9: **Si** ($dentroRangoMov(a_{nueva})$ = **Cierto**) **Entonces**
10: agregar(a_{actual}, a_{nueva})
11: explotarAccion(a_{nueva})
12: **Fin Si**
13: **Fin Si**

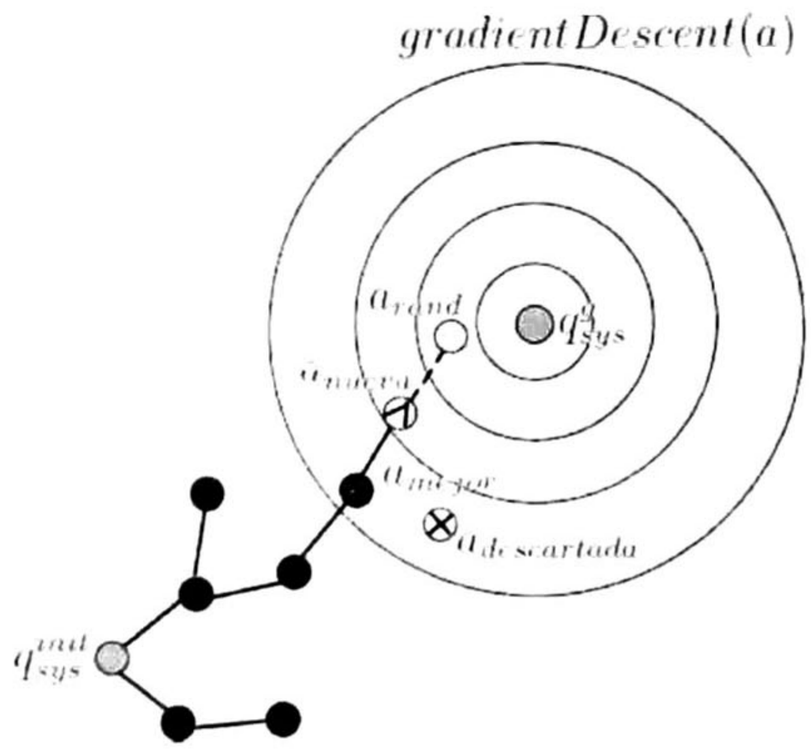


Figura 3.3: Funcionamiento del algoritmo de planeación propuesto

3.5. Conclusiones

En este capítulo se presentó un algoritmo de *planeación de movimientos para avatares* basado en el algoritmo RRT. Este algoritmo permite generar animaciones de una manera mucho más natural y eficiente, evitando así, invertir mucho tiempo en su elaboración. Con este algoritmo se logra una mejor naturalidad en la animación autónoma de los avatares, obteniendo distintas animaciones ante situaciones iguales, es decir, cuando se cuenta con el mismo estado inicial y se desea llegar al mismo estado final.

Además, se lograron hacer dos principales mejoras con respecto al RRT, las cuales son:

- No es necesario obtener el plan completo antes de ejecutar los movimientos del avatar.
- Se implementa el concepto de explotación implementado en el área de aprendizaje por refuerzo, para explorar acciones que lleven más rápido a la meta.

Estas modificaciones permiten reaccionar ante colisiones que inhabiliten cierta parte del plan que se esté ejecutando y generan un movimiento más natural al continuar moviendo la misma articulación hasta que ya nos acerque lo más que pueda al objetivo del plan.

Aunque la propuesta aquí presentada es eficiente, el algoritmo podría ser mejorado si se adapta para generar planes capaces de resolver tareas en las que se involucre el movimiento de las dos manos o el movimiento de alguna extremidad inferior durante la traslación del avatar.

Capítulo 4

Animación autónoma del brazo de un avatar

Para la verificación de la funcionalidad del algoritmo propuesto en este trabajo de tésis, se utilizó el motor 3D jMonkeyEngine (jME)[16] para realizar la animación autónoma del brazo izquierdo de un avatar.

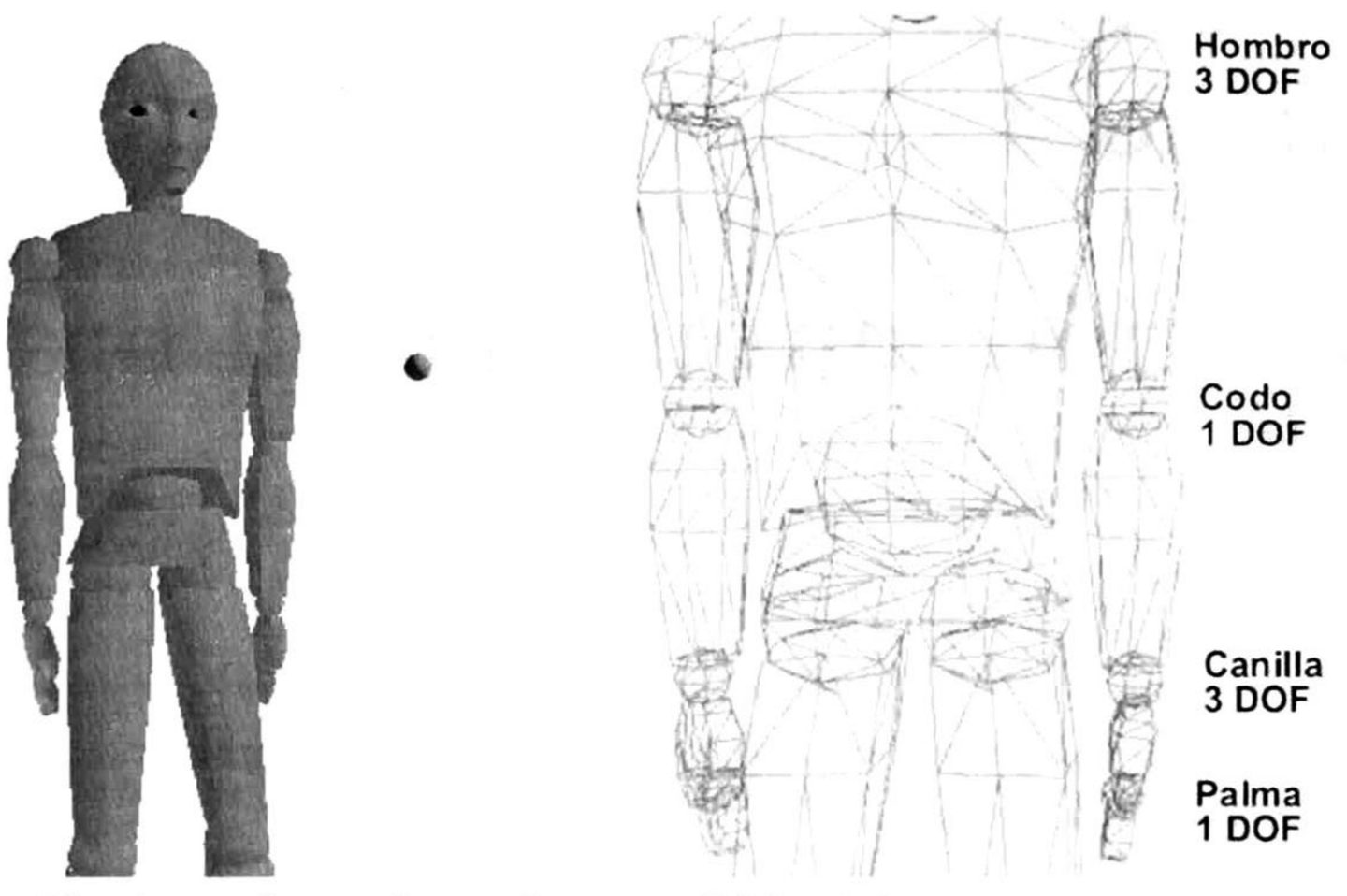
4.1. Descripción del problema

Las animaciones de avatares para desplazarse, agarrar y mover objetos en un ambiente virtual, siempre son necesarias debido a que se repiten mucho. Por lo que usando el algoritmo de planeación propuesto, la generación de este tipo de movimientos permite reducir el tiempo de ejecución de las animaciones. Para realizar cualquiera de las tareas antes mencionadas es necesario mover el avatar o alguna de sus extremidades, hasta que llegue a una posición específica.
Entonces, como caso de estudio utilizando el algoritmo propuesto, se obtendrá el plan de movimientos para que el brazo izquierdo del avatar pueda alcanzar una esfera que aparece dentro de su entorno de trabajo (espacio de configuración de movimientos válidos).

4.2. Parámetros de la planeación

Se utilizará el brazo izquierdo del avatar que se muestra en la figura4.1a, el cual cuenta con ocho grados de libertad (DOFs) como se muestra en la figura4.1b. El espacio de configuraciones solamente corresponderá al alcance que tenga este brazo, el cual se llamará C_{bizq} y ya que no se cuenta con obstáculos ($O = \emptyset$), para este caso de estudio $C_{bizq} = C_{libre}$, entonces la configuración del sistema será de la forma $\{q_1, ..., q_8\}$ usando el conjunto de articulaciones

$J = (j_1, ..., j_4)$.



(a) Avatar utilizado en el caso de estudio (b) los DOFs que se utilizaran

Figura 4.1: Avatar utilizado en los casos de estudio

La distribución de los DOFs que componen la configuración del sistema, sobre las articulaciones involucradas se especifican en la tabla 4.1:

Articulación	Grado de Libertad		
	Pitch	Roll	Yaw
Hombro	X	X	X
Codo	X		
Canilla	X	X	X
Palma	X		

Tabla 4.1: Grados de libertad que posee cada articulación del brazo izquierdo del avatar

4.2.1. Especificación del esqueleto del avatar

Debido a que normalmente una persona cuando se encuentra parada, relaja su brazo y cae, esta será la configuración inicial del sistema q_{sys}^{init} y la configuración final q_{sys}^{g}, podrá variar

mientras se logre llegar al objetivo. En la figura4.2 se puede apreciar el espacio de configuraciones y la configuración inicial.

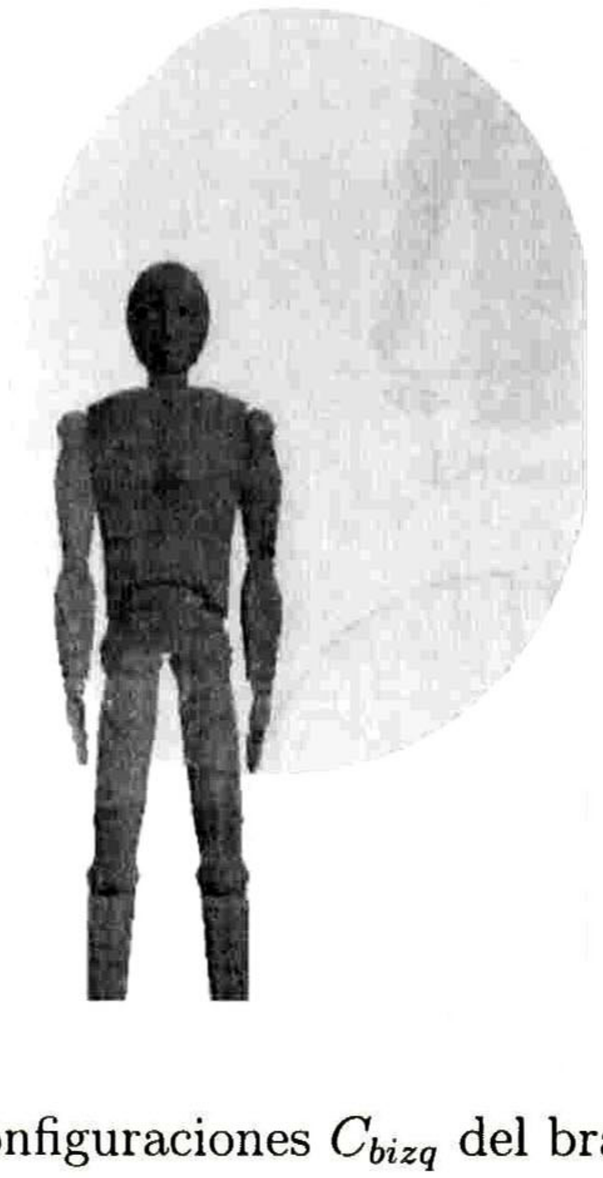


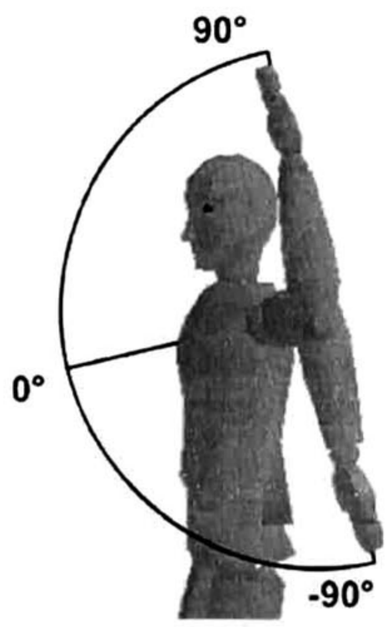
Figura 4.2: Espacio de configuraciones C_{bizq} del brazo del avatar

4.2.2. Grados de movimiento

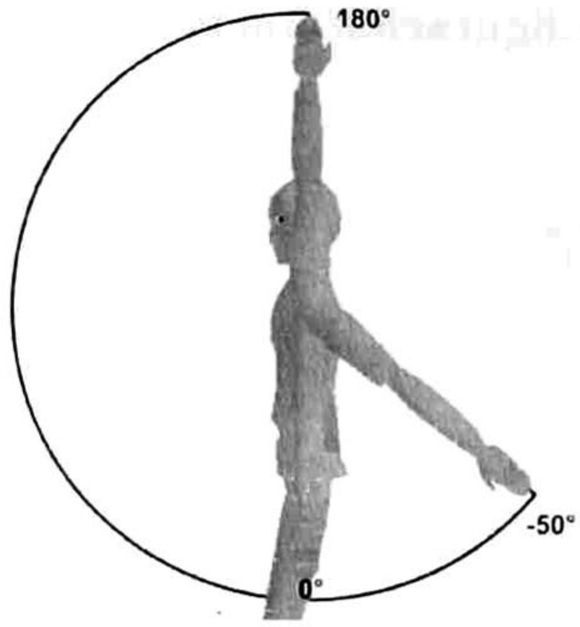
Aunque sólo se está utilizando un único brazo, se puede ver que el espacio de configuraciones es demasiado grande, entonces es por eso que se especifican los grados de movimiento(Θ) posibles para cada DOF, ya que así discretizamos el espacio de configuraciones.

Para este caso de estudio se especifica el valor de $\Delta p = 10$, con lo que sólo números que sean múltiplos de 10 pueden ser asignados como ángulos de rotación, tanto para estados como para acciones, y para obtener los valores de Θ_{min} y Θ_{max} correspondientes a cada DOF, se utiliza una implementación de la base de conocimientos de los agentes CAPE [7], desarrollada en el programa Protégé [17], donde previamente se almacenó la información del avatar que se está utilizando, incluyendo los límites de movimiento que se especifican en [27].

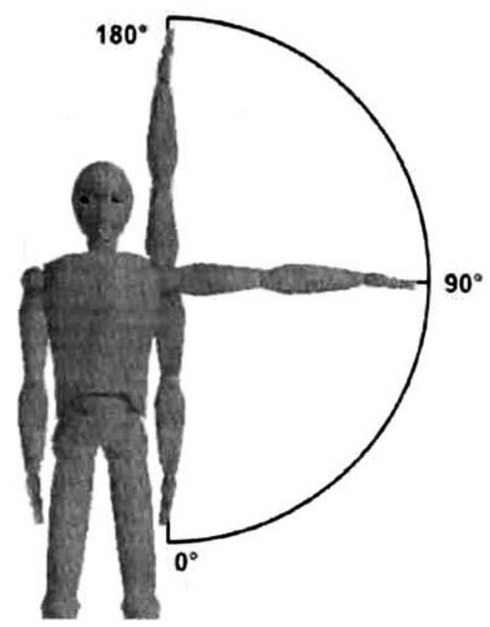
Los límites de cada uno de los grados de libertad se muestran en la figura 4.3.



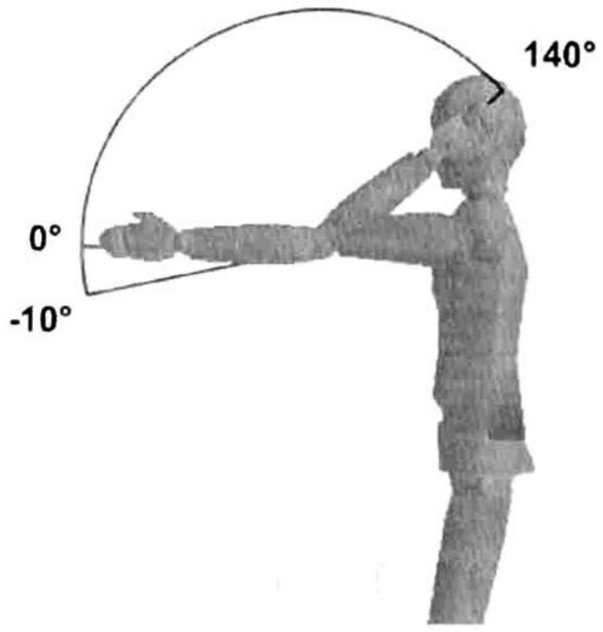
(a) hombroPich



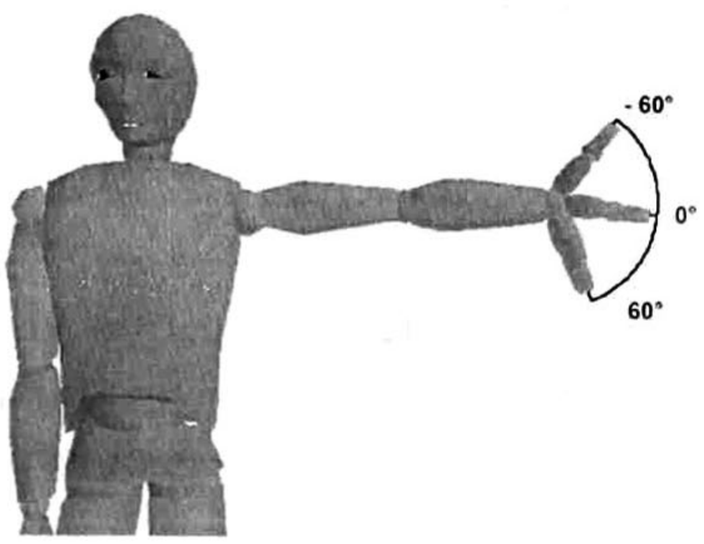
(b) hombroRoll



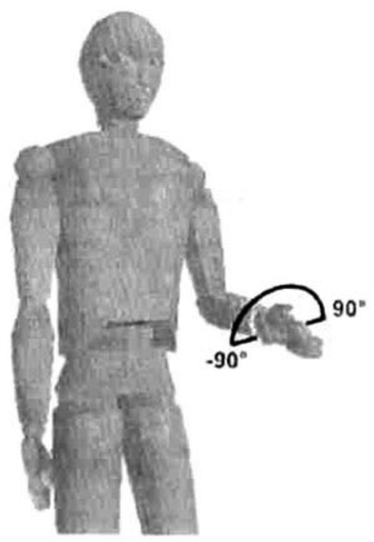
(c) hombroYaw



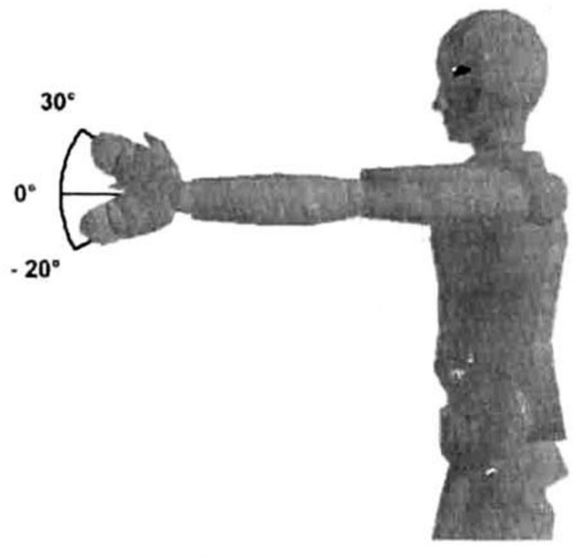
(d) codoPitch



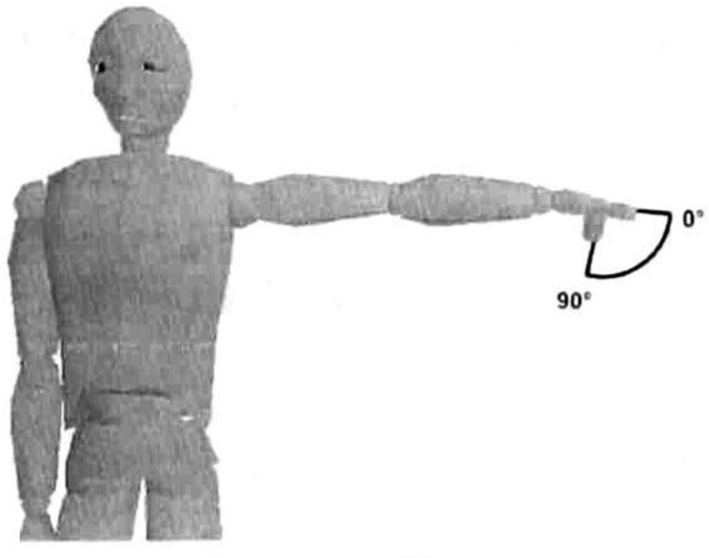
(e) canillaPitch



(f) canillaRoll



(g) canillaYaw



(h) palmaPitch

Figura 4.3: Rango de movimiento de cada DOF que posee el brazo izquierdo del avatar

mientras se logre llegar al objetivo. En la figura4.2 se puede apreciar el espacio de configuraciones y la configuración inicial.

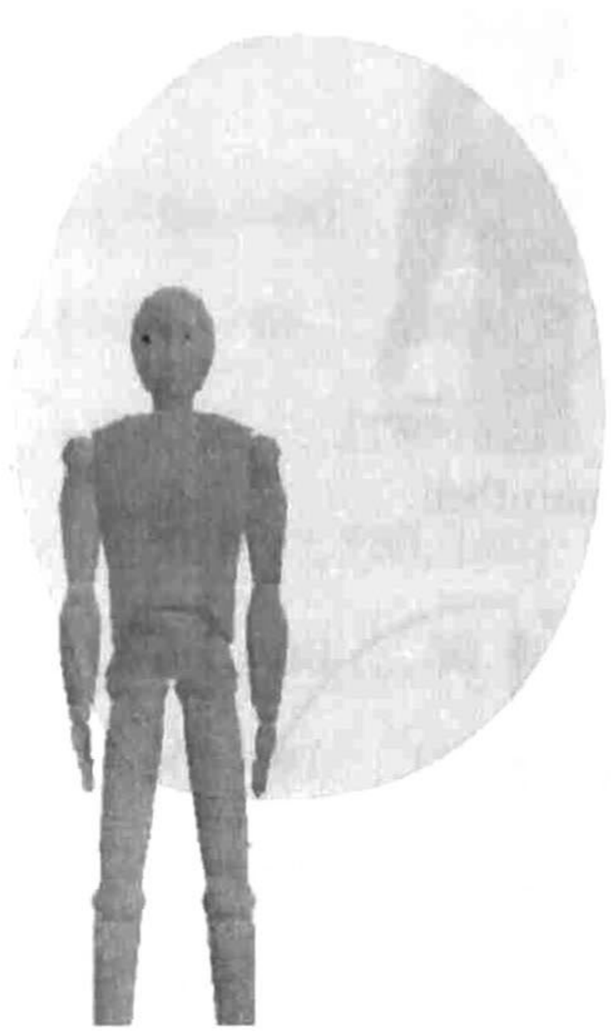


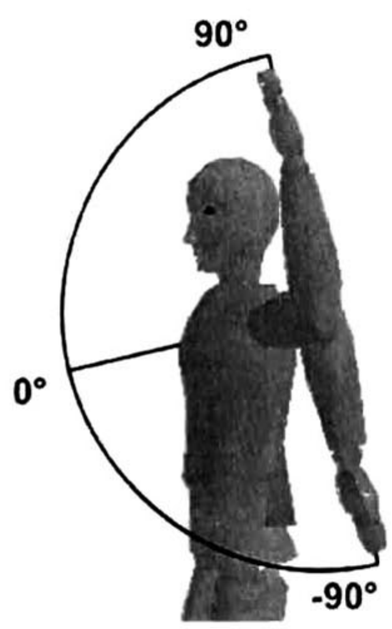
Figura 4.2: Espacio de configuraciones C_{bizq} del brazo del avatar

4.2.2. Grados de movimiento

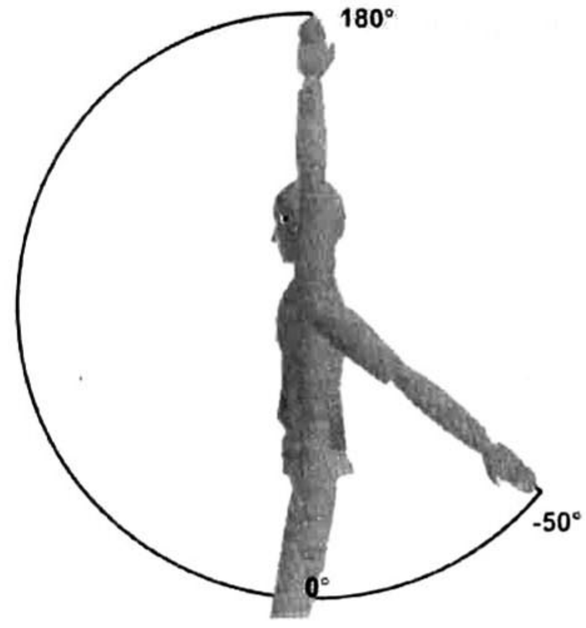
Aunque sólo se está utilizando un único brazo, se puede ver que el espacio de configuraciones es demasiado grande, entonces es por eso que se especifican los grados de movimiento(Θ) posibles para cada DOF, ya que así discretizamos el espacio de configuraciones.

Para este caso de estudio se especifica el valor de $\Delta p = 10$, con lo que sólo números que sean múltiplos de 10 pueden ser asignados como ángulos de rotación, tanto para estados como para acciones, y para obtener los valores de Θ_{min} y Θ_{max} correspondientes a cada DOF, se utiliza una implementación de la base de conocimientos de los agentes CAPE [7], desarrollada en el programa Protégé [17], donde previamente se almacenó la información del avatar que se está utilizando, incluyendo los límites de movimiento que se especifican en [27].

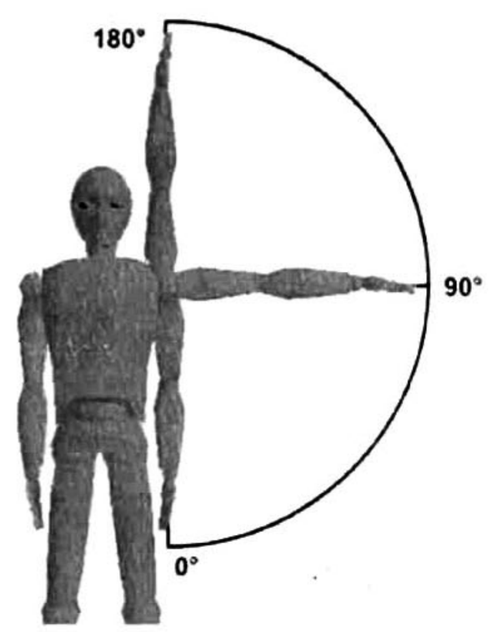
Los límites de cada uno de los grados de libertad se muestran en la figura 4.3.



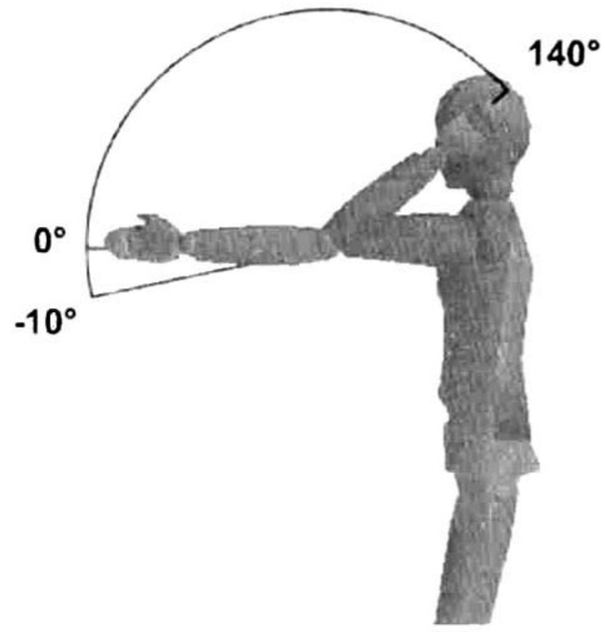
(a) hombroPich



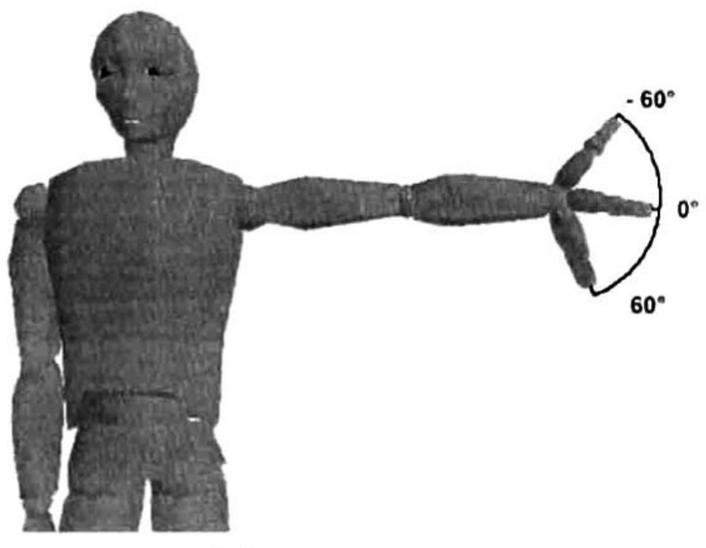
(b) hombroRoll



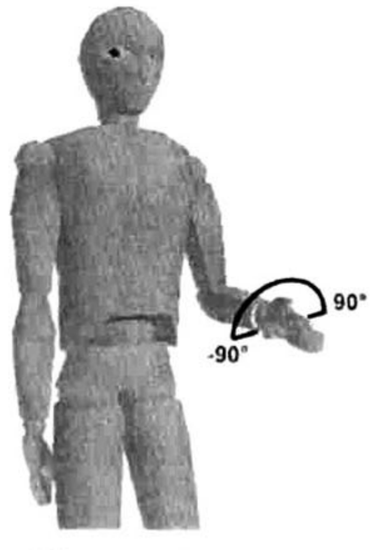
(c) hombroYaw



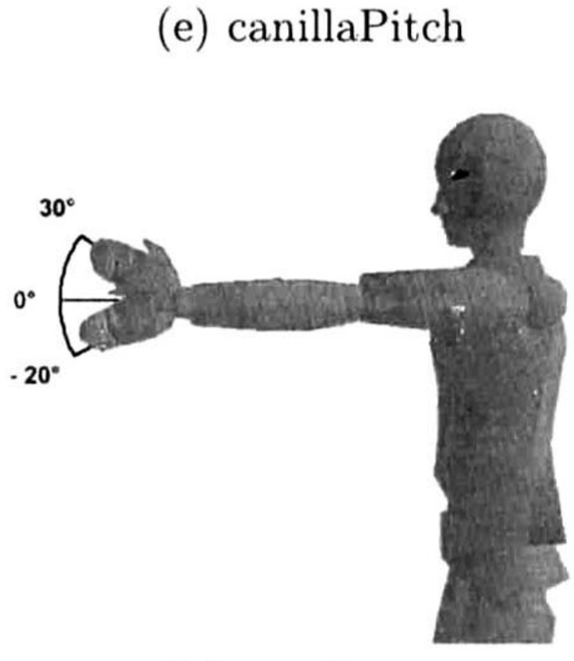
(d) codoPitch



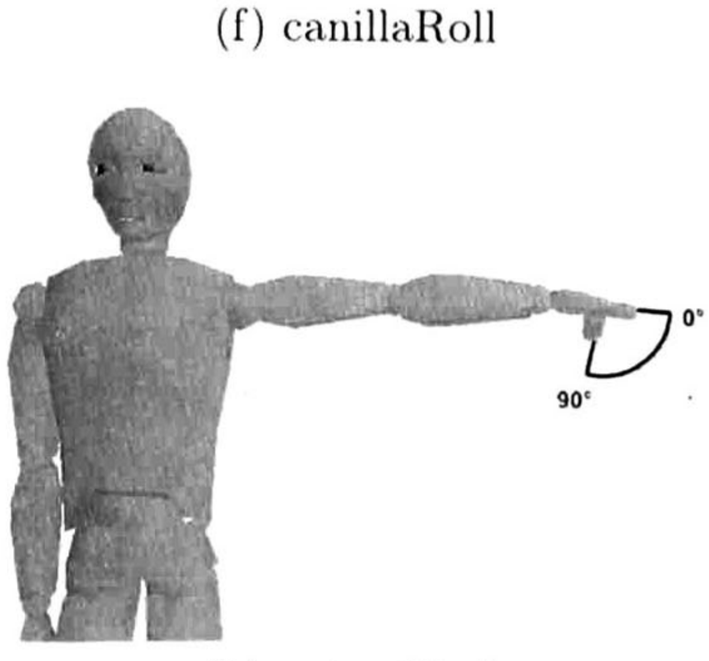
(e) canillaPitch



(f) canillaRoll



(g) canillaYaw



(h) palmaPitch

Figura 4.3: Rango de movimiento de cada DOF que posee el brazo izquierdo del avatar

4.2.3. Conjunto de estados

El conjunto de estados posibles es la unión de los conjuntos de grados de movimiento para cada DOF estos grados son:

$$D_{hombroPitch} = \{-90, -80, ..., 80, 90\}$$

$$D_{hombroRoll} = \{-50, -40, ..., 170, 180\}$$

$$D_{hombroYaw} = \{0, 10, ..., 170, 180\}$$

$$D_{codoPitch} = \{-10, 0, ..., 130, 140\}$$

$$D_{canillaPitch} = \{-60, -50, ..., 50, 60\}$$

$$D_{canillaRoll} = \{-90, -80, ..., 80, 90\}$$

$$D_{canillaYaw} = \{-20, -10, ..., 20, 30\}$$

$$D_{palmaPitch} = \{0, 10, ..., 80, 90\}$$

Esta discretización de los estados posibles sólo está delimitada sobre la estructura del brazo, ya que dependiendo de la dirección de rotación y del vector de traslación pueden existir diferentes estados en el plan, que mantienen la misma estructura interna del brazo.

4.2.4. Conjunto de acciones

Como el valor del ángulo de rotación Θ_i debe pertenecer al conjunto D_i, entonces el conjunto de acciones posibles es igual al conjunto de grados de movimiento de cada DOF por dos, esto último debido a que la acción, puede incrementar o decrementar el valor del ángulo de rotación actual del DOF.

4.2.5. Función del gradiente descendente

Es importante definir cual será la función utilizada por el método del gradiente descendente ya que de esto depende que nuestro algoritmo converja. Para este caso de estudio se utilizó la función del motor jME que nos regresa la distancia entre dos vectores tridimensionales dentro del entorno virtual. Si se desea especificar una función de distancia que no dependa del motor gráfico en el que se implemente el algoritmo, se puede utilizar la función de la distancia Euclidiana o la de Manhattan.

4.3. Parámetros del algoritmo de planeación

A continuación se especificará el valor de los parámetros utilizados en este caso de estudio para el algoritmo de planeación de movimientos, si es que no han sido especificados antes.

- Algoritmo 3.1: Como j_m es la última articulación del brazo, esta será la articulación de la palma. El valor de $umbral = 3$.

- Algoritmo 3.2: Se especifica el valor de $x = 4$ para agregar cuatro acciones nuevas cada vez que se llame a este algoritmo, se optó por utilizar este número para obtener rápidamente las acciones y no interrumpir la ejecución de los movimientos.

- Algoritmo 3.3: Los límites de los rangos de números aleatorios son $x_1 = 2$ y $x_2 = 6$, esto es para generar los posibles números siguientes {5,10} para el primer rango y {5,10,15,20,25,30} para el segundo. El valor de $umbral_2 = 2$

- Algoritmo 3.4: El valor del $umbral_2$ sigue siendo 2 y los posibles valores del ángulo de rotación son $x_1 = 1$ y $x_2 = \Delta p$.

- Algoritmo 3.5: Los valores de las variables son iguales al algoritmo anterior, es decir, $umbral_2 = 2,\ x_1 = 1$ y $x_2 = \Delta p$.

4.4. Casos de estudio

En esta sección se presentarán los resultados de algunos de los casos de estudio que se realizaron con el algoritmo de planeación propuesto.

En el primero, el objetivo del plan se encuentra en frente del brazo izquierdo, a una distancia en la cual no es necesario que el avatar extienda todo su brazo. El plan que se obtuvo se muestra en la figura 4.4.

En el siguiente caso de estudio, el objetivo se encuentra a un lado del brazo del avatar, donde el avatar para alcanzarlo tendrá que extender todo su brazo como se muestra en la figura 4.5.

Ahora en el siguiente caso de estudio la meta está por arriba del avatar, el plan que se obtuvo se muestra en la figura 4.6.

Por último, en el experimento de la figura 4.7 se coloca la meta en frente del lado derecho del avatar.

(a)

(b)

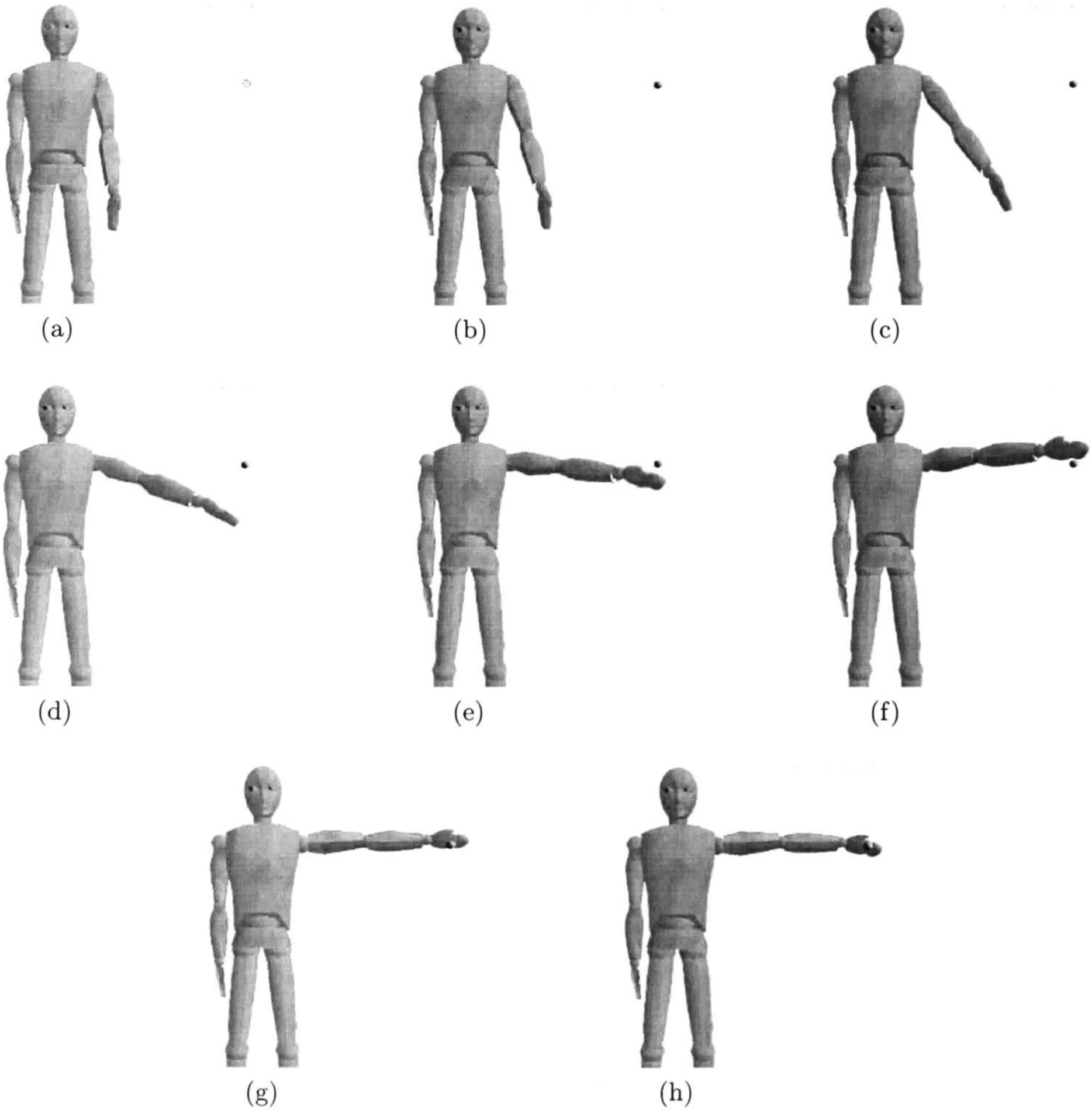
(c)

(d)

(e)

(f)

Figura 4.4: Caso de estudio donde la meta se encuentra en frente del brazo izquierdo

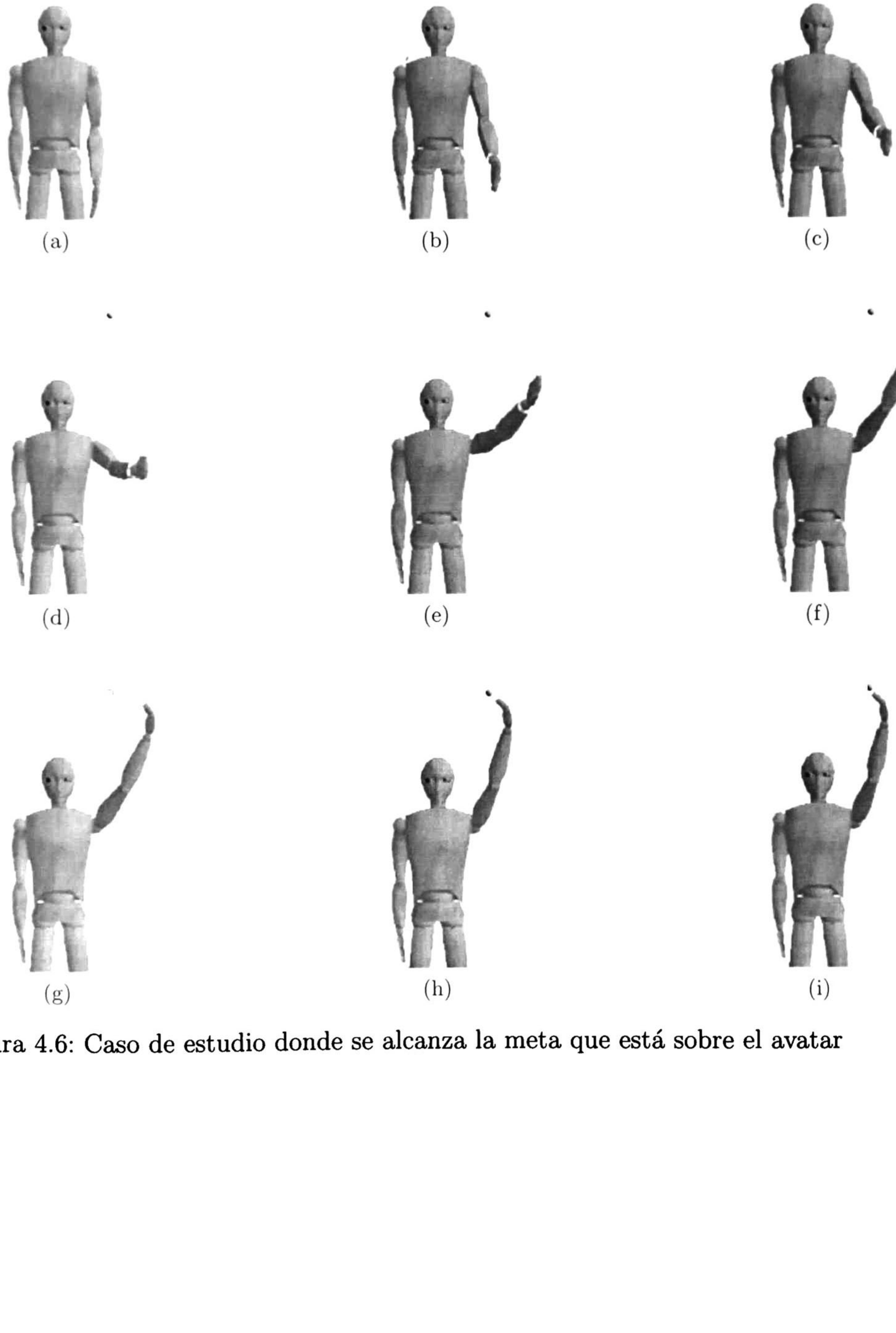


(a) (b) (c)

(d) (e) (f)

(g) (h)

Figura 4.5: Caso de estudio en el cual la meta está a un lado del avatar

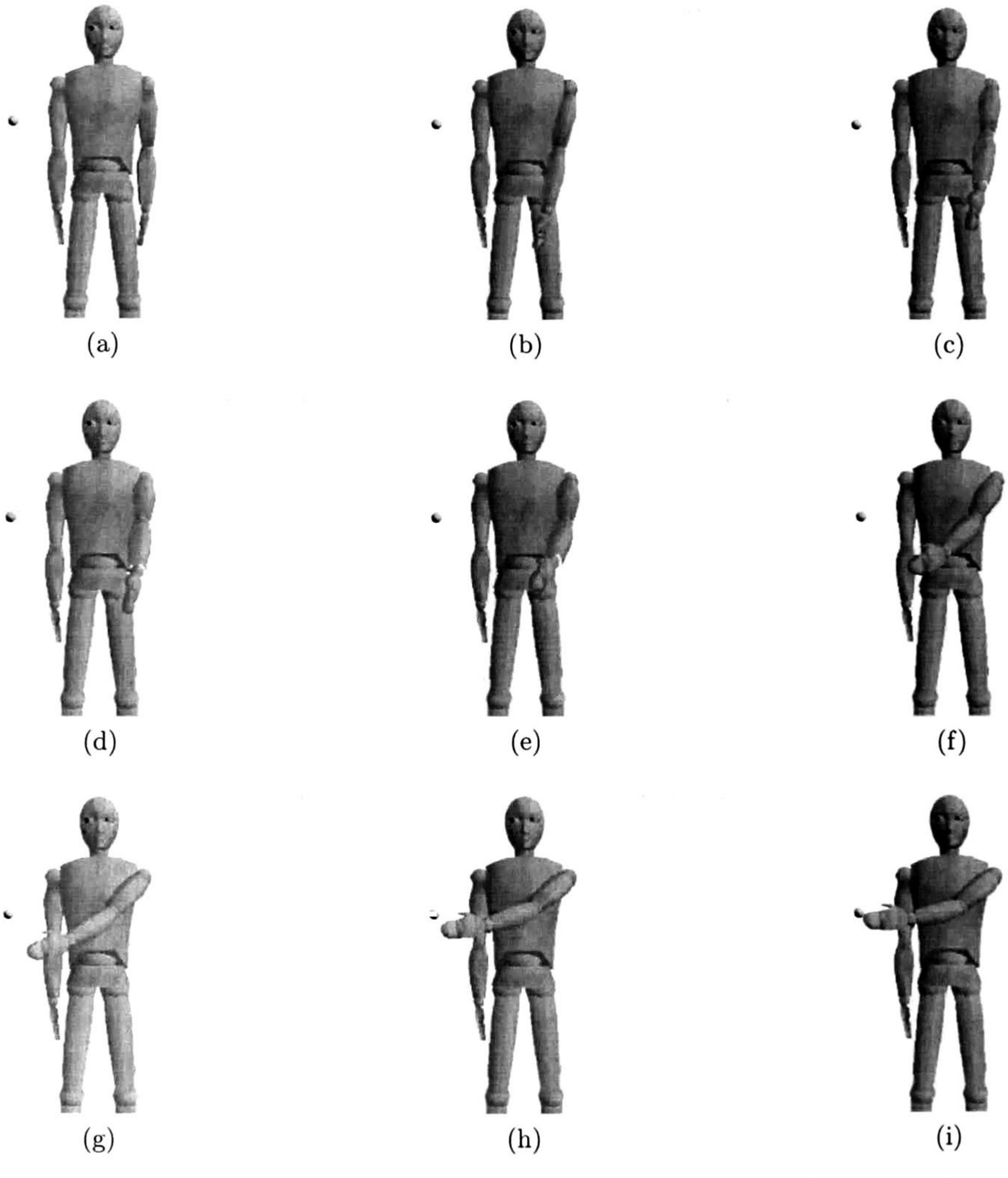


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 4.6: Caso de estudio donde se alcanza la meta que está sobre el avatar



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 4.7: Caso de estudio donde se alcanza la meta que está en frente del lado derecho el avatar

4.5. Conclusiones

Con los casos de estudio presentados y los experimentos realizados, se puede ver el buen funcionamiento del algoritmo de planeación de movimientos, ya que las animaciones que se logran obtener son muy reales y aunque se está trabajando con un gran número de grados de libertad, lo que genera un espacio de configuraciones grande, este es reducido al discretizarse el espacio de búsqueda para las acciones a ejecutar.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo de tesis se presentó un algoritmo de planeación de movimientos para avatares, el cual permite obtener la secuencia de movimientos para la generación de animaciones autónomas. Este algoritmo muestra la posibilidad de crear secuencias de acciones para la animación real de avatares.

Tomando en cuenta varios trabajos de investigación sobre la planeación de movimientos autónomos que presentan diferentes formas de atacar el problema [13],[29],[6],[12], se han encontrado dos problemas comunes, los cuales son:

1. El gran tamaño del espacio de configuraciones conforme se incrementa el número de grados de libertad utilizados.

2. La dependencia hacía los métodos de cinemática para calcular los movimientos de las diferentes articulaciones.

El algoritmo propuesto, el cual puede trabajar con cualquier número de grados de libertad que sean necesarios para cumplir con la tarea, discretiza el espacio de configuraciones posibles para estos DOFs, al delimitar los grados de rotación posibles para cada DOF, así atacamos el primer problema. Mientras que, eliminando la dependencia de los métodos de cinemática, se ataca el segundo problema. Esto se logra utilizado una base de conocimientos que posea la información sobre el esqueleto del avatar que se está utilizando, y esto aunado con la utilización de un método aleatorio para obtener los movimientos, con lo cual podemos prescindir de los métodos cinemáticos, además de que no nos interesa una secuencia específica de movimientos, mientras que estos logren llegar al objetivo y sean movimientos válidos.

Para realizar sólo movimientos válidos, se adapta la información sobre el rango de movimientos de las diferentes articulaciones humanas, obtenidas por el estudio quinesiológico que se

encuentra en [27] y asignando estos rangos de movimiento a los DOFs correspondientes. Esta información debes estar contenida en la base de conocimientos.

Por lo tanto y como se muestran en el caso de estudio presentado en el capitulo 4, se ha logrado obtener una secuencia de movimientos muy natural y real, por lo que se puede utilizar este algoritmo para reducir el tiempo que un diseñador gráfico requiere para obtener animaciones específicas sin la necesidad de algún equipo de animación especializado.

5.2. Trabajo Futuro

Como trabajo futuro se planea mejorar el algoritmo al realizar las siguientes modificaciones:

- Evitar colisiones consigo mismo.
- Manejar mejor las colisiones durante la ejecución de un plan de movimientos, al especificar reglas de comportamiento más eficientes, para que el algoritmo las pueda aplicar al momento de reaccionar ante una colisión.
- Emplear el concepto de exploración de estados, utilizado en el área de aprendizaje por refuerzo, ya que puede ser que el algoritmo no converja y se quede en un mínimo local del gradiente.

También se planea mejorar el desenvolvimiento de los avatares al utilizar un algoritmo de aprendizaje junto con este algoritmo de planeación. Se utilizará el algoritmo de aprendizaje para aprender y realizar movimientos más complejos, y el algoritmo de planeación se ejecutará en el momento que se trate de realizar alguna tarea para la cual no se tienen movimientos aprendidos y se requiera una rápida solución.

Apéndice A

Rapidly-exploring Random Tree

Este algoritmo fue desarrollado por LaValle y Kuffner [23]. El algoritmo Rapidly-exploring Random Tree (RRT) ha sido diseñado para realizar una búsqueda eficiente en un espacio muy grande construyendo una estructura en forma de árbol, además de que no requiere tener un modelo del entorno para trabajar. El RRT es construido de una manera incremental, lo cual reduce rápidamente la distancia esperada entre un punto escogido aleatoriamente y el árbol. A continuación se presenta la estructura básica del algoritmo:

Algoritmo A.1 Rapidly-exploring Random Tree

1: Construir_RRT($q_i, K, \triangle s$)
2: G.init(q_{init})
3: **Para** $k \leftarrow 1$ **Hasta** $k \leftarrow K$ **Hacer**
4: $q_{rand} \leftarrow Rand\_Conf()$;
5: $q_{near} \leftarrow Nearest\_Vertex(q_{rand}, G)$
6: $q_{new} \leftarrow New\_Conf(q_{near}, \triangle s)$
7: $G.add_vertex(q_{new})$
8: $G.add_edge(q_{near}, q_{new}$
9: **Fin Para**
10: **Devolver** (G)

El RRT inicia con una configuración q_{init} como raíz y posteriormente se construyen K vértices como se indica a continuación:

- Se elige una configuración aleatoria q_{rand}, donde, asumiendo una métrica p sobre el espacio de configuraciones C, se evalúan todos los vértices obtenidos en G y se toma el más cercano a q_{rand}.

- Se genera una nueva configuración q_{new}, al moverse de q_{near} una distancia $\triangle s$ en la

dirección de q_{rand}. De esta forma se obtiene una secuencia de configuraciones en una estructura de árbol.

Bibliografía

[1] Humberto Sossa y Roberto A. Vázquez Beatriz A. Garro. Evolving ant colony system for optimizing path planning in mobile robots. 2007.

[2] Wei LI y Yang YANG Chenyu MA. Robot motion planning with many degrees of freedom. *Revista 1*, pages 892 897, Octubre 1995.

[3] Iván Piza Fabiel Zúñiga, Félix F. Ramos. Geda-3d agent architecture. pages 201–205, Julio 2005.

[4] Jaime Zaragoza y Luis Razo Félix Ramos, Alonso Aguirre. The use of ontologies for creating virtual scenarios in geda-3d. pages 230 – 235, 2006.

[5] Michael Gerke. Genetic path planning for mobile robots. Junio 1999.

[6] Claudia Esteves y Jean-Paul Laumond Gustavo Arechavaleta. Planning fine motions for a digital factotum. Septiembre. year = 2004.

[7] Félix Ramos y Daniel Thalmann Héctor Rafael Orozco. Avatars animation using reinforcement learning in 3d distributed dynamic virtual environments. November 2007.

[8] Satoshi Kagami Masayuki Inaba y Hirochika Inoue James Kuffner, Koichi Nishiwaki. Motion planning for humanoid robots. pages 365–374, Agosto 2005.

[9] Kensuke Harada y Jean-CLaude Latombe Kris Hauser, Timothy Bretl. Using motion primitives in probabilistic sample-based planning for humanoid robots. pages 507 522, Agosto 2008.

[10] Steven M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

[11] Shusheng LV You Shi-Zhijie Wang y Ali Raza Jafri Lige Zhang, Qiang Huang. Humanoid motion design considering rhythm based oon human motion capture. pages 2491–2496, Octubre 2006.

[12] Tolga Abaci y Daniel Thalmann Marcelo Kallmann, Amaury Aubel. Planning collision-free reaching motion for intractive object manipulation and grasping. pages 313–322, Noviembre 2003.

[13] Nicholas K. Taylor y Matthew W. Mohd Murtadha Mohamad. Articulated robot motion planning using ant colony optimisation. Septiembre 2006.

[14] A. Rodriguez y Felix Ramos Moises Uc. Reinforcement learning and dynamic planning applied to virtual humans animation. Septiembre 2007.

[15] Oussama Khatib Oliver Brock. Real-time replanning in high-dimensional configuration space using sets of homotopic path. Abril 2000.

[16] Mark Powell. jmonkeyengine.com serious monkeys. serious engine.

[17] Stanford University. The protégé ontology editor and knowledge acquisition system.

[18] Alan B. Craig William R. Sherman. *Understanding Virtual Reality.* Morgan Kaufmann Publishers, 2003.

[19] Dave Ferguson y Anthony Stentz. Anytime, dynamic planning in high-dimensional search spaces. Abril 2007.

[20] Jorg Hoffmann y Bernhard Nebel. The ff planning system: Fast plan generation through heuristic serach, publisher = , journal = Journal of Artificial Intelligence Research, vol 14, pages = 253-302, month = , year = 2001.

[21] Jaesik Choi y Eyal Amir. Factor-guided motion planning for a robot arm. Noviembre 2007.

[22] Hwan-Joo Kwak y Gwi-Tae Park. Motion separating based whole body motion planning for humanoid robots using a gradient descent method. Julio 2008.

[23] S. LaValle y J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research, vol.20*, pages 368–400, 2001.

[24] Julien Pettre y Jean-Paul Laumond. A motion capture-based control-space approach for walking mannenquins. *Computer Animation and Virtual Worlds*, pages 109 126. Mayo 2006.

[25] Jehee Lee y Kang Hoon Lee. Precomputing avatar behavior from human motion data. *Graghical Models*, pages 158–174, Marzo 2006.

[26] Morten Engell Norregard y Kenny Erleben. Estimation of joint types and joint limits from motion capture data. Febrero 2009.

[27] Kathryn Luttgens y Nancy Hamilton. *Kinesiology: Scientific basis of human motion, tenth edition.* McGraw-Hill, 2002.

[28] James Kuffner y S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. pages 365 374, Abril 2000.

[29] Eiichi Yoshida. Humanoid motion planning using multi-level dof exploitation based on randomized method. 2005.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Planeación de Movimiento para Avatares.

del (la) C.

Cristian Eduardo BOYAIN Y GOYTIA LUNA

el día 17 de Diciembre de 2009.

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara

Dr. Andrés Méndez Vásquez
Investigador CINVESTAV 2A
CINVESTAV

Dr. Marco Antonio Ramos Corchado
Profesor Investigador Nivel F
Universidad Autónoma del Estado de México