CAPÍTULO 3

HEURÍSTICA PARA LA GENERACIÓN DE CONFIGURACIONES

Antes de explicar la heurística diseñada en esta tesis es necesario explicar qué son los *cuaterniones* ya que éstos forman una parte fundamental en la generación de configuraciones.

3.1 CUATERNIONES

Los cuaterniones fueron descubiertos por William Rowan Hamilton en 1843 y presentados como una extensión de los números complejos. Más tarde, en 1845, Arthur Cayley publica la manera de describir rotaciones usando la multiplicación de cuaterniones. Actualmente los cuaterniones se utilizan en áreas tales como la aeronáutica, la robótica y la animación por computadora [12] [13].

3.1.1 FORMALIZACIÓN

Un cuaternión es una tupla de cuatro elementos donde los elementos atómicos son números reales y se representan con la siguiente notación:

$$q = [w, x, y, z]$$

La magnitud de q está dada por:

$$|q| = \sqrt{(w^2 + x^2 + y^2 + z^2)}$$

El cuaternión además se interpreta como un escalar y un vector 3D escribiéndose así:

$$q = [\omega, v]$$

dónde:

$$v = [x, y, z] \qquad \text{y} \qquad \omega = \text{w}$$

Otra manera de formalizar los cuaterniones es abordando el tema de los números complejos. Los números complejos son una extensión del sistema de los números reales y pueden ser escritos en la forma $a + bi$, donde a y b son números reales con $\text{i}^2 = -1$. Los cuaterniones son una extensión de los números complejos los cuales pueden ser definidos de la siguiente manera:

$$q = a + b\text{i} + c\text{j} + d\text{k}$$

dónde a, b, c y d son números reales. Para **i**, **j** y **k** se tienen las siguientes propiedades generales:

$$\text{i}^2 = -1, \qquad \text{j}^2 = -1, \qquad \text{k}^2 = -1$$

$$\begin{array}{c} \text{ij} = \text{k}, \quad \text{jk} = \text{i}, \quad \text{ki} = \text{j} \\ \text{ji} = -\text{k}, \quad \text{kj} = -\text{i}, \quad \text{ik} = -\text{j} \end{array}$$

Esto es claramente una extensión del sistema de los números complejos donde los números complejos son cuaterniones con $c = d = 0$ y los números reales son aquellos cuaterniones con $b = c = d = 0$.

3.1.2 ÁLGEBRA DE CUATERNIONES

La siguiente sección muestra las operaciones más importantes en el álgebra de cuaterniones [14].

Un cuaternión es dado por $q = w + xi + yj + zk$ donde w, x, y y z son números reales. Definimos $q_n = w_n + x_n i + y_n j + z_n k$ ($n = 0, 1$). La adición y sustracción de cuaterniones está dada por:

$$\begin{aligned} q_0 \pm q_1 &= (w_0 + x_0 i + y_0 j + z_0 k) \pm (w_1 + x_1 i + y_1 j\, z_1 k) \\ &= (w_0 \pm w_1) + (x_0 \pm x_1) i + (y_0 \pm y_1)\text{j} + (z_0 \pm z_1) k \end{aligned}$$

La multiplicación de cuaterniones esta definida como:

$$\begin{aligned} q_0 q_1 &= (w_0 + x_0 i + y_0 j + z_0 k)(w_1 + x_1 i + y_1 j + z_1 k) \\ &= (w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1) + \\ &= (w_0 x_1 + x_0 w_1 + y_0 z_1 - z_0 y_1) i + \\ &= (w_0 y_1 - x_0 z_1 + y_0 w_1 + z_0 x_1) j + \\ &= (w_0 z_1 + x_0 y_1 - y_0 x_1 + z_0 w_1) k \end{aligned}$$

La multiplicación no es conmutativa, es decir el producto de $q_0 q_1$ no es igual al producto de $q_1 q_0$.

El conjugado de un cuaternión está definida por:

$$q^* = (w + xi + yj + zk)^* = w - xi - yj - zk$$

El conjugado de un producto de cuaterniones satisface las propiedades $(p^*)^* = p$ y $(pq)^* = q^* p^*$.

La norma de un cuaternión es:

$$N(q) = N(w + xi + yj + zk) = w^2 + x^2 + y^2 + z^2$$

La norma es un valor real y la norma de un producto de cuaterniones satisface las propiedades $N(q^*) = N(q)$ y $N(pq) = N(p)\, N(q)$.

El inverso multiplicativo de un cuaternión q es denotado por q^{-1} y tiene la propiedad $qq^{-1} = q^{-1}q = 1$. Se define como:

$$q^{-1} = q^* / N(q)$$

dónde la división de un cuaternión por un escalar real es solo la división por componentes. La operación inversa satisface las propiedades $(p^{-1})^{-1} = p$ y $(pq)^{-1} = q^{-1}p^{-1}$.

Una simple pero útil función es la función *selección*:

$$W(q) = W(w + xi + yj + zk) = w$$

la cual selecciona la parte real del cuaternión. Esta función satisface la propiedad $W(q) = (q + q^*) / 2$.

El cuaternión $q = w + xi + yj + zk$ podría además ser visto como $q = w + \boldsymbol{v}$, donde $v = xi + yj + zk$. Si se identifica a v como el vector 3D (x, y, z) entonces la multiplicación de cuaterniones puede ser escrita usando el producto punto y el producto cruz, como:

$$(w_0 + v_0)\,(w_1 + v_1) = (w_0 w_1 - v_0 v_1) + w_0 v_1 + w_1 v_0 + v_0 \times v_1$$

De esta manera es claro que $q_0 q_1 = q_1 q_0$ si y solo si $v_0 \times v_1 = 0$, (*los vectores son paralelos*).

Un cuaternión q podría además ser visto como un vector 4D (w, x, y, z). El producto punto de dos cuaterniones es:

$$q_0\, q_1 = w_0 w_1 + x_0 x_1 + y_0 y_1 + z_0 z_1 = W(q_0 q_1^*).$$

Un cuaternión unitario es un cuaternión q para el cual $N(q) = 1$. El inverso de un cuaternión unitario y el producto de cuaterniones unitarios son por si mismos cuaterniones unitarios. Un cuaternión unitario puede ser representado por:

$$q = cos\ \theta + u\ sen\ \theta$$

dónde u es un vector 3D de longitud 1. Sin embargo el producto del cuaternión $uu = -1$.

3.1.3 ROTACIÓN CON CUATERNIONES

La rotación alrededor de un eje específico puede ser obtenida a partir de la representación de cuaternión como un par ordenado compuesto por una parte escalar y una parte vectorial **q = (*s*, *v*)** [15] [16].

La parte vectorial *v* representará la rotación de un cuerpo en esa dirección con respecto a un ángulo *s* dado. Podemos definir un cuaternión de rotación con las siguientes partes escalares y vectoriales:

$$\mathbf{s = \cos(\theta / 2)} \qquad \mathbf{y} \qquad \boldsymbol{v} = \mathbf{u}\ \mathbf{sen}(\theta/2)$$

dónde *u* es un vector unitario a lo largo del eje de rotación que se seleccionó y θ es el ángulo de rotación específico con respecto de este eje [16]. Cualquier punto *P* que se desee rotar mediante un cuaternión se puede representar en la notación de cuaternión como:

$$\mathbf{P = (0, p)}$$

Las coordenadas del punto son la parte vectorial **p** = (*x, y, z*). La rotación del punto se lleva a cabo con la siguiente operación:

$$\mathbf{P'} = \mathbf{qPq^{-1}}$$
$$\mathbf{P'} = q * [0, \mathbf{p}] * q^{-1}$$

dónde $q^{-1} = (s, -v)$, es el inverso del cuaternión de rotación *q*.

Esto produce un cuaternión con la parte escalar a cero, su parte vectorial representa la rotación de **p** con respecto a la dirección ***v*** y un ángulo θ.

Las rotaciones también pueden ser realizadas definiendo una matriz de rotación a partir del cuaternión *q* [16] [17] donde *q* = (*s, a, b, c*). La matriz es la siguiente:

$$M_R(\theta) = \begin{pmatrix} 1 - 2b^2 - 2c^2 & 2ab - 2sc & 2ac + 2sb \\ 2ab + 2sc & 1 - 2a^2 - 2c^2 & 2bc - 2sa \\ 2ac - 2sb & 2bc + 2sa & 1 - 2a^2 - 2b^2 \end{pmatrix}$$

Además se tienen fórmulas directas [18] que permiten calcular la rotación de un punto una vez que se ha definido el cuaternión y son las siguientes.

Dado el cuaternión *q* y el punto **P** (x, y, z):

x = q.w * q.w * P.x + 2 * q.y * q.w * P.z - 2 * q.z * q.w * P.y + q.x * q.x * P.x + 2 * q.y * q.x * P.y + 2 * q.z * q.x * P.z – q.z * q.z * P.x – q.y * q.y * P.x

y = 2 * q.x * q.y * P.x + q.y * q.y * P.y + 2 * q.z * q.y * P.z + 2 * q.w * q.z * P.x – q.z * q.z * P.y + q.w * q.w * P.y - 2 * q.x * q.w * P.z – q.x * q.x * P.y

z = 2 * q.x * q.z * P.x + 2 * q.y * q.z * P.y + q.z * q.z * P.z - 2 * q.w * q.y * P.x – q.y * q.y * P.z + 2 * q.w * q.x * P.y – q.x * q.x * P.z + q.w * q.w * P.z

A continuación se explica detalladamente en que consiste el problema a solucionar propuesto en esta tesis.

3.2 PROBLEMA DE LOS CLAVOS

Actualmente existen un conjunto de modelos o benchmarks que pueden ser usados para comparar distintos métodos de planeación de movimientos, estos modelos se encuentran disponibles en internet para uso no comercial y destinados a la comunidad de robótica

Uno de estos benchmarks es el *problema de los clavos* conocido en la comunidad de robótica por sus términos en inglés como "*alpha puzzle*" y diseñado por Boris Yamrom de *GE Corporate Research & Development Center* en *Nueva York*.

El problema de los clavos es un problema particular que requiere para su solución el desarrollo de heurísticas particulares, mismas que hasta ahora no lo han resuelto de manera satisfactoria.

El problema consiste de dos tubos cada uno de los cuales se encuentra torcido formando la figura alfa; un tubo funciona como obstáculo y el otro se designa como el objeto móvil ó robot en el espacio de trabajo. El modelo consiste en una malla de 1008 triángulos, dados en coordenadas del mundo real, en las cuales tanto el robot como el obstáculo se encuentran en una configuración entrelazada, uno con respecto al otro tal y como se muestra en la siguiente figura.

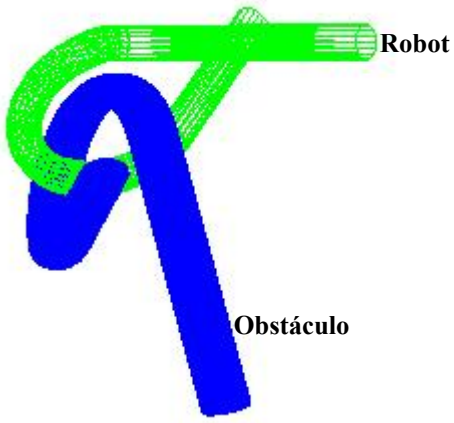

Figura 3.1 Problema de los clavos.

Actualmente existen tres modelos para el problema además del original, donde cada uno de ellos tiene diferente nivel de dificultad. Estos tres modelos se desarrollaron escalando el obstáculo a lo largo del eje z, con un factor constante más grande que 1, lo cual tuvo como efecto que se abriera el pasaje estrecho formado por las dos puntas del obstáculo [19].

Conforme vamos de una versión a otra la abertura del pasaje estrecho se acerca a su tamaño original, haciendo cada vez más complicada cada una de las versiones. Esta complejidad inicia con la *versión 1.5* y se va incrementando en cada una de las siguientes versiones, *1.2* y *1.1* respectivamente, hasta llegar a su máximo en la versión original conocida como *versión 1.0*.

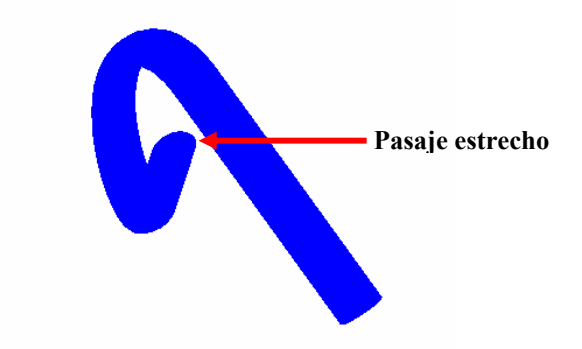

Figura 3.2 Pasaje estrecho formado por el clavo.

Las versiones *1.5*, *1.2* y *1.1* del problema de los clavos forman un pasaje estrecho bastante corto pero mucho más amplio que el formado por el modelo original (ver figura 3.2). Este pasaje, para las primeras dos versiones, es lo suficientemente amplio para que el robot pueda salir, de la configuración entrelazada en que se encuentra con el obstáculo, solamente desplazándose por él.

Sin embargo para la versión *1.1* a pesar de que el pasaje estrecho es más amplio que en la versión original, aún no tenemos una solución que nos permita asegurar como el robot se libera del obstáculo, es decir, el robot podría salir desplazándose por el pasaje estrecho o bien realizando rotaciones como en la versión original.

En el modelo original (*versión 1.0*) el robot no puede salir de la configuración entrelazada desplazándose solamente por el pasaje estrecho, porque ya no hay espacio para esto, sino debe realizar cierto tipo de rotaciones muy adecuadas que lo lleven fuera del obstáculo.

Investigadores destacados en el área de planeación de movimientos han reportado que sus métodos probabilísticos resuelven el problema de los clavos, tal es el caso de la *Dra. Nancy Amato* quien reportó haber solucionado la *versión 1.1* con el método OBPRM [19] y los *Drs. Steve Lavalle* y *J. P. Laumond* quienes reportaron haber solucionado el problema original con el método "*Rapidly Random Tree*" (*RRT*) [20] y *Visibilidad* [10, 11] respectivamente.

El método probabilístico *RRT* no es descrito en esta tesis ya que no fue implementado. Sin embargo los otros dos métodos si, esto nos lleva a la siguiente pregunta *¿Por qué si estos métodos fueron implementados y sus autores afirman que resuelve el problema de los clavos, la implementación realizada no encuentra la solución*?

Los métodos *OBPRM* y *Visibilidad* implementados en esta tesis, se diseñaron para resolver eficientemente problemas de planeación de movimientos de tipo general, sin embargo existen detalles de la implementación realizada por parte de los autores que no son explicados en sus artículos y que seguro fueron muy necesarios para adecuar estos métodos para solucionar el problema de los clavos.

Debido a esto es necesario el desarrollo de heurísticas particulares que busquen solucionar éste problema separando los dos clavos de la manera más eficiente posible, tal y como es el caso de esta tesis.

3.3 HEURÍSTICA PARA LA GENERACIÓN DE CONFIGURACIONES

3.3.1 Idea general

La presente tesis se enfocó a resolver específicamente algunas de las versiones del problema de los clavos por medio del diseño e implementación de una heurística, muy particular, que generara configuraciones adecuadas para el robot permitiéndole salir del obstáculo.

Los métodos probabilísticos descritos en el capítulo 2, tal y como se ha mencionado, fueron diseñados e implementados en este trabajo, porque sus autores afirman que son métodos probabilísticamente completos que resuelven efectivamente la mayoría de los problemas existentes en el área de planeación de movimientos e incluso algunas versiones del problema de los clavos.

Sin embargo nuestra implementación no es tan poderosa como la de sus autores y el problema de los clavos exige configuraciones para el robot muy particulares y difíciles de encontrar. Entonces, dado lo anterior, se debe dar especial importancia a la etapa de generación de configuraciones de la fase de construcción de éstos métodos.

La heurística propuesta y diseñada en esta tesis genera configuraciones específicas para el problema de los clavos, cumpliendo con las características necesarias identificadas por la experiencia del grupo de robótica de la UDLAP tanto en el área de planeación de movimientos como en dicho problema. La heurística es una de las principales aportaciones de la tesis, que específicamente mejora la etapa de generación de configuraciones en la fase de construcción de un roadmap (ver figura siguiente).

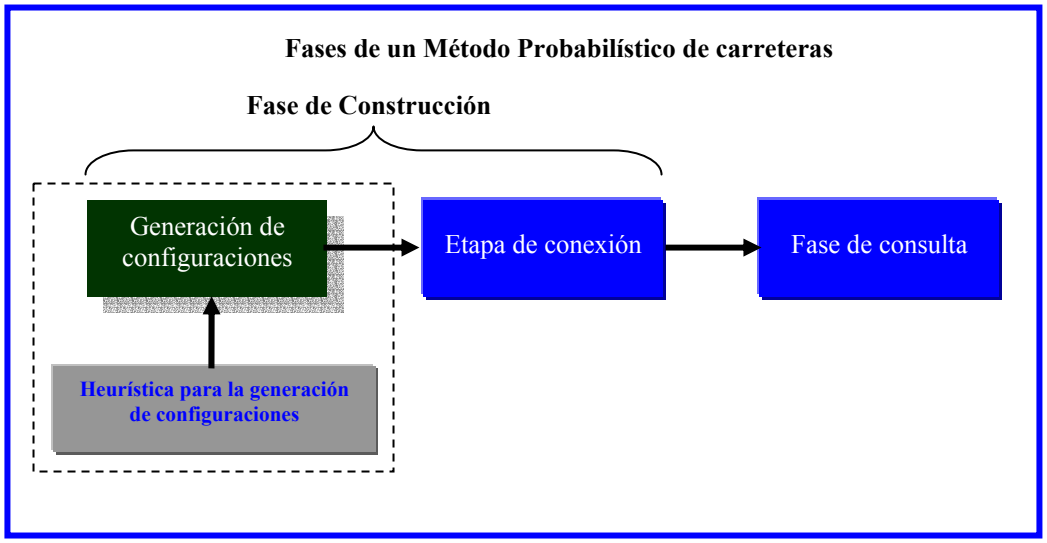

Figura 3.3 Aportación principal de la tesis.

Las configuraciones generadas por la heurística, tal y como se describió en el capítulo 1, tienen dos partes, una que indica la posición y otra que indica la orientación del robot. Varias estrategias para colocar y orientar el robot en el espacio de trabajo fueron analizadas y probadas, con el fin de generar las mejores configuraciones.

La parte fundamental de la heurística la constituyen los cuaterniones, los cuales fueron usados para rotar el robot, cierto ángulo, con respecto a ejes particulares definidos sobre la superficie del obstáculo, obteniendo así la parte de orientación que forma a la configuración.

Para la parte de posición de la configuración se definieron diferentes puntos sobre la malla de triángulos que forma al obstáculo, mismos que el usuario puede seleccionar para generar una configuración específica, ver figura 3.4.

Diferentes estrategias fueron analizadas y probadas para identificar vectores sobre la superficie del obstáculo, que definieran ejes con respecto a los cuales se rotaría el robot usando cuaterniones.

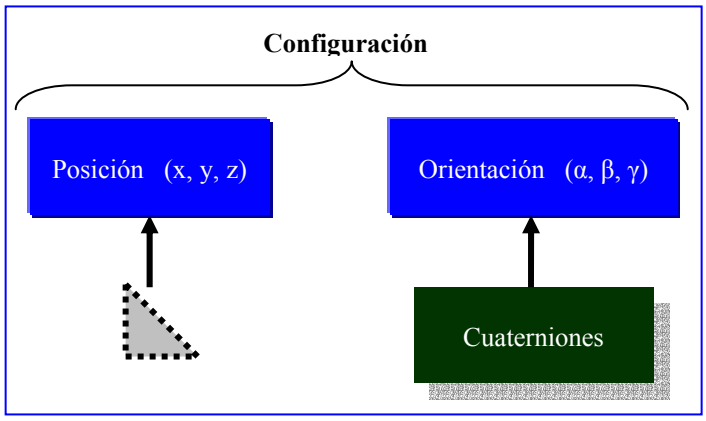

Figura 3.4 Partes de una configuración.

La heurística diseñada es probabilística, ya que el ángulo de rotación para el cuaternión se genera aleatoriamente así como también el punto, según el parámetro seleccionado, dónde el robot será colocado.

Cada una de las configuraciones generadas por la heurística es utilizada en la fase de construcción del roadmap, de cada uno de los métodos probabilísticos implementados y los resultados obtenidos por éstos son graficados y analizados con ayuda del sistema **GEMPA.**

3.3.1.1 SISTEMAS GEMPA y SPM/PC

GEMPA (*Graphics Environment for Motion Planning Algorithms* por sus siglas en inglés) es un ambiente gráfico para visualizar los resultados obtenidos durante la ejecución de algún algoritmo de planeación de movimientos, implementado durante el desarrollo de una tesis o bien durante el desarrollo de un proyecto de investigación.

Este sistema ha sido diseñado e implementado en el *Laboratorio de Robótica* del *Departamento de Ciencias Computacionales* de la *Universidad de las Américas – Puebla*, como parte del trabajo doctoral del estudiante Antonio Benítez Ruiz [21].

El sistema cuenta con un conjunto de métodos que permiten leer los distintos espacios de trabajo construidos para probar los métodos implementados así como las mallas de triángulos que forman a cada uno de los objetos del ambiente.

Además cuenta con un conjunto de métodos para realizar las transformaciones geométricas básicas, tales como las rotaciones generales *xyz*, necesarias al momento de graficar las rutas o configuraciones encontradas por los distintos algoritmos, así como también métodos para realizar cálculos vectoriales.

Por otra parte el sistema **SPM/PC** (*Sistema de planeación de movimientos para el problema de los clavos*) se diseñó e implemento para probar la heurística explicada anteriormente (ver apéndice I), utilizando como base algunas de las clases definidas por el sistema GEMPA, principalmente los métodos que leen el espacio de trabajo y las mallas de triángulos de archivos de texto, así como otros métodos que serán brevemente explicados mas adelante, conforme vayamos explicando el trabajo realizado en ésta tesis.

El sistema SMP/PC fue implementado bajo plataforma *Microsoft Windows XP* y desarrollado en *C++ Builder versión 6 de Borland*, siendo compatible en estas características con el sistema *GEMPA*.

Tal y como se explicó en la sección anterior la heurística diseñada genera configuraciones para el robot en el espacio de trabajo, a continuación se explican cada una de las etapas necesarias, que lleva a cabo la heurística, para construir una configuración.

3.3.2 Vectores posición

La parte de posición de la configuración significa el punto (x, y, z) en el espacio de trabajo dónde el robot será colocado. Por motivos de simplicidad nosotros frecuentemente nos referimos a un punto sobre la superficie del robot que llamamos *Punto de referencia* y con respecto al cual nosotros trasladamos al robot a una posición en específico.

El punto de referencia para un objeto, robot u obstáculo, es su centro de masa al cual le llamamos *Centroide*. Este punto (x, y, z) se calcula como el promedio aritmético de cada una de las coordenadas x, y y z de todos los vértices presentes en la malla del objeto, es decir:

B = cuerpo formado por una malla de triángulos
v = Vértice con coordenadas x, y, z.

$$B = \{v_0, v_1, v_2, \ldots, v_n\}$$

$$v_i = (x_i, y_i, z_i)$$

las coordenadas del centroide (Xc, Yc, Zc) son:

$$\begin{aligned} Xc &= \sum (x_0 + x_1 + x_2 + \ldots + x_n) / n \\ Yc &= \sum (y_0 + y_1 + y_2 + \ldots + y_n) / n \\ Zc &= \sum (z_0 + z_1 + z_2 + \ldots + z_n) / n \end{aligned}$$

El centroide de un cuerpo, que tiene forma regular o poligonal, generalmente se encuentra en su interior y en ocasiones exactamente en su centro, para cuerpos de forma irregular el

centroide regularmente está fuera de la superficie del obstáculo, tal es el caso del cuerpo que representa al clavo, como se muestra en la figura siguiente.

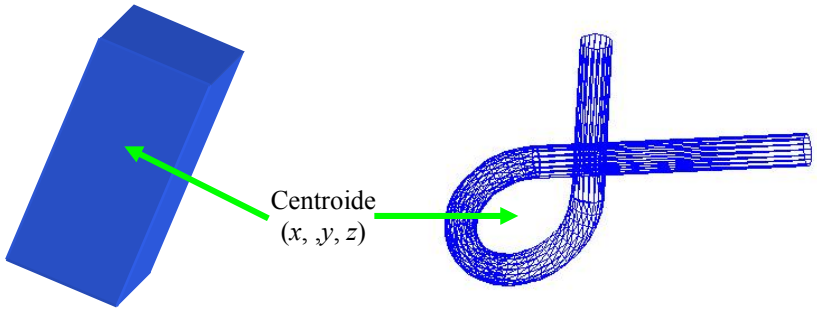

Figura 3.5 Centroide de un cuerpo.

Cuando deseamos colocar al robot en una posición en específico en el espacio de trabajo colocamos su centroide en esa posición. Durante este trabajo se definieron diferentes vectores de posición para colocar el robot sobre la superficie del obstáculo, formando así la primer parte de la configuración. Los vectores definidos (figura 3.6) son los siguientes:

a) El baricentro de cada uno de los triángulos que forman la malla del obstáculo,

b) Cada uno de los tres vértices que forman un triángulo en ésta malla,

c) Puntos sobre el plano que definen los triángulos de ésta malla.

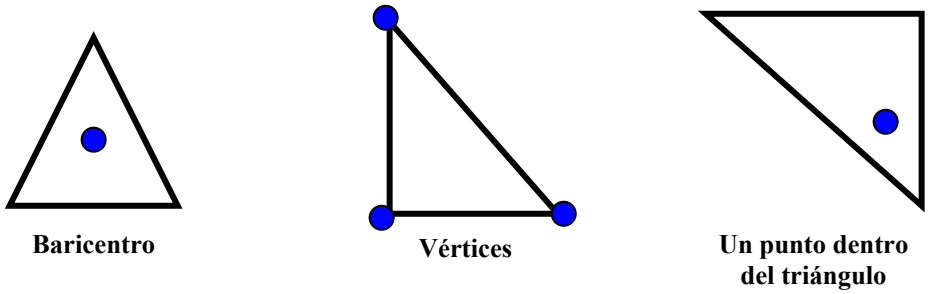

Figura 3.6 Posiciones para el robot.

Estos puntos fueron seleccionados ya que se requiere que el robot esté colocado de manera entrelazada con el obstáculo, ventaja importante que ofrece el cuerpo del clavo ya que su

centroide no se encuentra en la superficie del objeto. Para cuerpos cuyo centroide esté en el interior del objeto estos puntos no son validos tal y como explicaremos más adelante.

3.3.2.1 Baricentro de un triángulo

El *baricentro* de un triángulo es el punto dónde se intersectan sus *medianas*. La *mediana* es el segmento de línea que une los vértices de un triángulo con los puntos medios de los lados opuestos (ver figura 3.7).

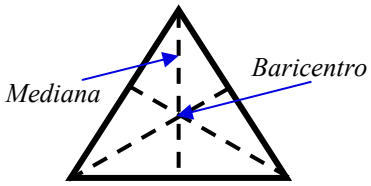

Figura 3.7 Baricentro de un triángulo.

Las coordenadas del punto medio de un segmento de línea es la media aritmética de las coordenadas de sus extremos. Las coordenadas del baricentro son la media aritmética de los tres vértices del triángulo.

t = Triángulo

v = Vértice con coordenadas x, y, z.

$t = \{v_0, v_1, v_2\}$

$$Baricentro = (Xb, Yb, Zb)$$

$$\begin{aligned} Xb &= \sum (x_0 + x_1 + x_2) / 3 \\ Yb &= \sum (y_0 + y_1 + y_2) / 3 \\ Zb &= \sum (z_0 + z_1 + z_2) / 3 \end{aligned}$$

El baricentro tiene la propiedad de que está situado en cada mediana a una distancia de 2/3 del vértice y a 1/3 del lado [22, 23].

3.3.2.2 Punto dentro del triángulo

Para definir un punto sobre el plano que forman los vértices de un triángulo (figura 3.8) realizamos lo siguiente:

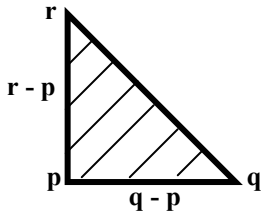

Figura 3.8 Plano definido por los vértices *p*, *q* y *r* del triángulo.

Sean *s* y *t* dos números generados aleatoriamente entre *0* y *1*. Tal que:

$$0 \leq s \leq 1 \qquad 0 \leq t \leq 1 \qquad \text{y} \qquad 0 \leq s + t \leq 1$$

definimos los vectores $v_1 = r - p$ y $v_2 = q - p$

y calculamos el punto **P** dentro del triángulo [24] como:

$$\mathbf{P} = p + (s(r - p) + t(q - p))$$

3.3.2.3 Vértices de un triángulo

Colocar el centroide del robot en uno de los vértices de un triángulo seleccionado aleatoriamente, de la malla que forma al cuerpo del obstáculo, es muy fácil dado que cada uno de estos vértices es conocido y solamente es necesario seleccionar aleatoriamente uno de ellos.

3.3.3 Ejes de rotación

Los cuaterniones, tal y como se explicó en la sección 3.1.3, permiten ejecutar rotaciones de un cuerpo, en nuestro caso el robot, alrededor de un eje cierto ángulo. Estos ejes se definieron como vectores presentes en la superficie del obstáculo y son producto del análisis vectorial realizado sobre los triángulos de la malla que forma el clavo. Los ejes definidos son los siguientes:

a) El vector normal al plano definido por cada uno de los triángulos de la malla,

b) Los vectores que pasan exactamente por cada uno de los lados de cada uno de los triángulos de la malla,

c) Un vector arbitrario configurado en sus coordenadas *x*, *y*, *z* por el usuario es dado a la heurística cuando el ambiente de trabajo del robot es leído por el sistema **SPM/PC.** Los ejes se ilustran en la siguiente figura:

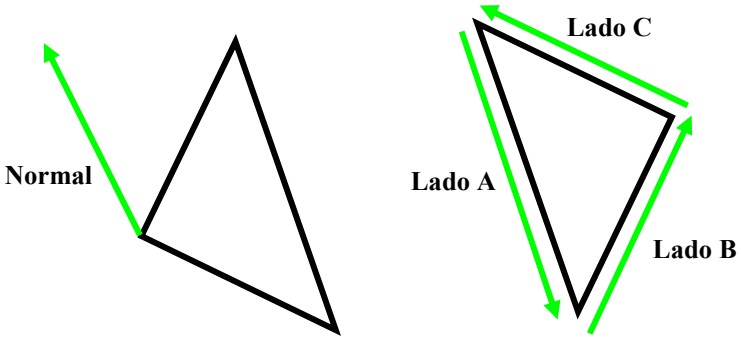

Figura 3.9 Vectores presentes en la superficie de un triángulo y empleados como ejes de rotación.

Estos ejes fueron seleccionados por ser los más representativos sobre la superficie del obstáculo y dada la experiencia que se fue adquiriendo durante el curso de esta investigación en la búsqueda de soluciones para el problema de los clavos.

3.3.3.1 Cálculo del vector normal a un triángulo

El vector normal a una superficie es un vector que apunta en dirección perpendicular a dicha superficie, si tenemos un triángulo con vértices P_1, P_2 y P_3 el procedimiento para calcular su normal (figura 3.10) consiste en escoger un vértice como origen, por ejemplo P_2, calcular los vectores $v_1 = P_1 - P_2$ y $v_2 = P_3 - P_2$ y posteriormente calcular el producto vectorial entre v_1 y v_2.

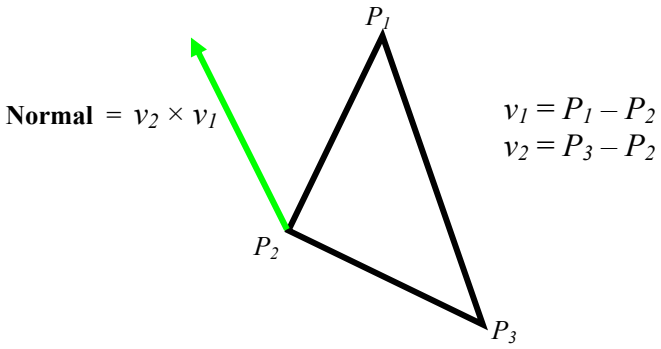

Figura 3.10 Vector normal a un triángulo

El producto vectorial no cumple con la propiedad conmutativa por lo que se debe tener cuidado al multiplicar los vectores v_1 y v_2, ya que el vector normal resultante podría apuntar en una dirección opuesta [24].

3.3.3.2 Cálculo de los vectores lados de un triángulo

Los vectores que pasan por cada uno de los lados de un triángulo pueden ser calculados a través de la diferencia de vectores entre los respectivos vértices del lado del triángulo, tal y como se muestra en la siguiente figura:

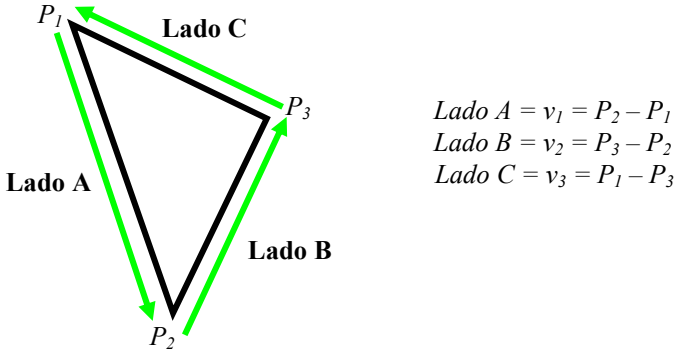

Figura 3.11 Vectores lados de un triángulo.

La figura 4.11 trata de esquematizar los vectores que pasan por los lados de un triángulo así como su dirección, en ningún momento se trata de vectores paralelos. En adelante nos referiremos a estos vectores como solamente lados del triángulo.

3.3.3.3 Cuaternión de rotación

Definidos los ejes anteriores es necesario construir el cuaternión de rotación para rotar al robot en el espacio de trabajo y obtener la parte de orientación de la configuración. El cuaternión de rotación se calcula a partir de dos partes, el eje ***v*** respecto al cual el objeto va a girar y el ángulo ***s*** que describirá cuanto rotará el objeto alrededor de ese eje.

Declaramos *quaternion* y *Eje* como estructuras de datos, dónde:

quaternion = {*x, y, z, w*}

Eje = {*x, y, z*}

El pseudocódigo para el cálculo del cuaternión de rotación es:

```
CuaterniondeRotacion(v, s)
{
   quaternion q;

   N    ← sqrt (v.x² + v.y² + v.z²);

   q.w  ← cos(s/2);
   q.x  ← ( v.x * sin(s/2) ) / N;
   q.y  ← ( v.y * sin(s/2) ) / N;
   q.z  ← ( v.z * sin(s/2) ) / N;

   return (q);
}
```

Tal y como se observa el cuaternión de rotación es una tupla de cuatro elementos que podríamos aplicar directamente para rotar cada uno de los vértices del cuerpo que forma al

robot, empleando la fórmula $q\mathbf{P}q^{-1}$, descrita en la sección 3.1.3. Sin embargo preferimos construir una matriz de rotación descrita en la misma sección por dos motivos:

1. Una vez que el robot se ha rotado necesitamos obtener esta orientación con respecto a los ángulos de Euler, es decir requerimos los parámetros α, β, γ para construir la segunda parte de la configuración.

2. Obtener éstos tres parámetros de un matriz de rotación es un procedimiento relativamente sencillo, como veremos en la siguiente sección.

3.3.3.4 Ángulos de Euler

Comúnmente los objetos rotan en un sistema fijo de coordenadas globales, si definimos un sistema local rígido y como parte del objeto, una secuencia de tres rotaciones se ejecuta con respecto a éste sistema local de tal manera que cuando el objeto se rota éste sistema rota con él. Los sistemas coordenados locales y globales inicialmente coinciden y los ángulos son llamados *Ángulos de Euler* [25].

Los ángulos α, β, γ indican rotaciones respecto a los ejes coordenados x, y y z respectivamente, como se muestra en la figura:

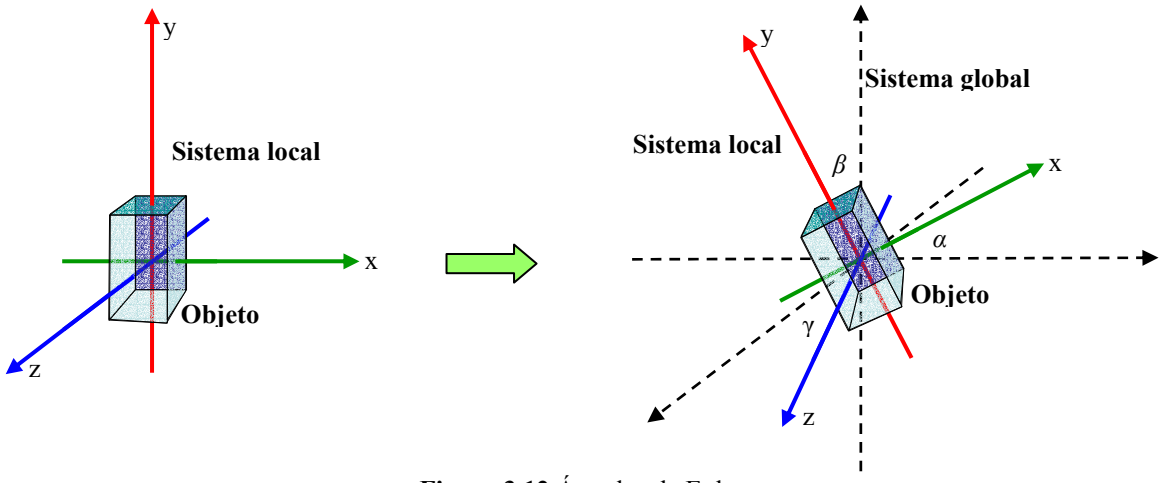

Figura 3.12 Ángulos de Euler.

Una vez que se ha construido la matriz de rotación ***m*** con el cuaternión de rotación ***q*** es posible obtener de dicha matriz los ángulos de Euler que corresponden a cada uno de los ángulos de rotación para cada eje coordenado [26]. El pseudocódigo es mostrado a continuación:

```
AngulosEuler(Matriz m, double alfa, double beta, double gamma)
{
 beta  ← atan2( -m[2,0], sqrt(m[0,0]^2 + m[1,0]^2) );

 gamma ← atan2( m[1,0] / cos(beta), m[0,0] / cos(beta) );

 alfa  ← atan2( m[2,1] / cos(beta), m[2,2] / cos(beta) );
}
```

3.4 CONFIGURACIÓN PARA EL ROBOT

Una vez que se tiene la posición (x, y, z) del espacio de trabajo, dónde colocaremos el centroide del robot y su orientación (α, β, γ) la configuración se construye como:

$$Configuración = (x, y, z, \alpha, \beta, \gamma)$$

De la forma descrita en las secciones anteriores, la heurística genera un conjunto de configuraciones que cumplen con las características necesarias para el problema de los clavos.

A continuación se esquematizan cada una de las etapas necesarias para la construcción de una configuración, llevadas a cabo por la heurística diseñada.

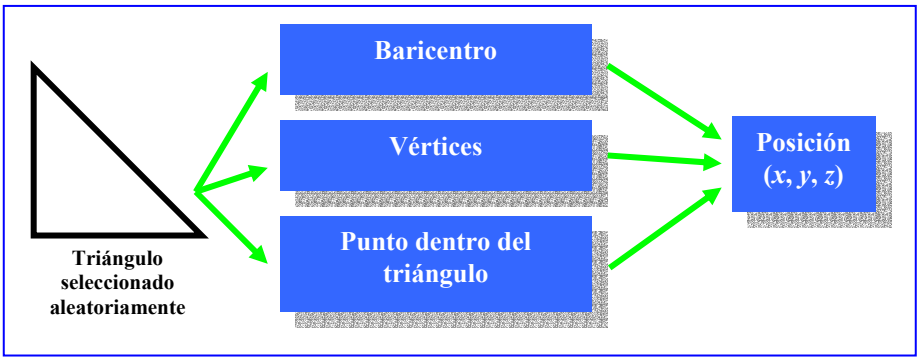

Figura 3.13 Parámetros para crear el vector posición.

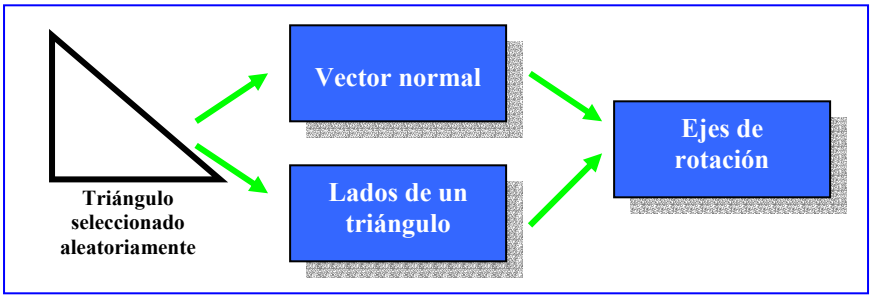

Figura 3.14 Ejes de rotación.

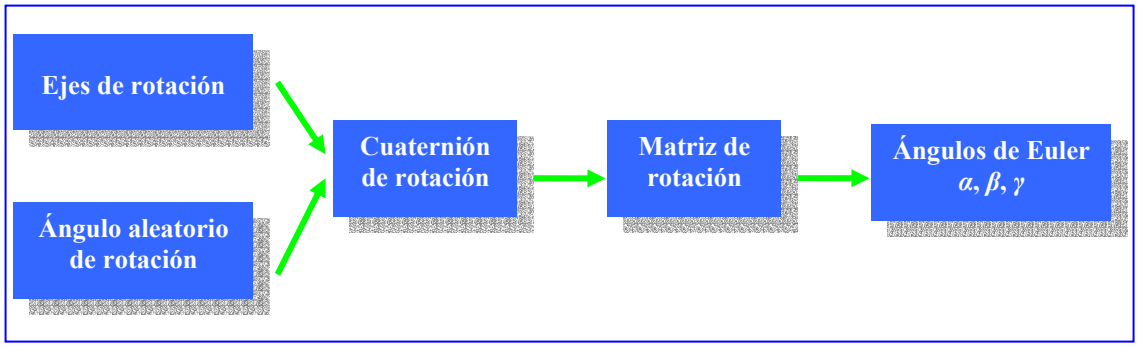

Figura 3.15 Etapas necesarias para determinar los parámetros de orientación.

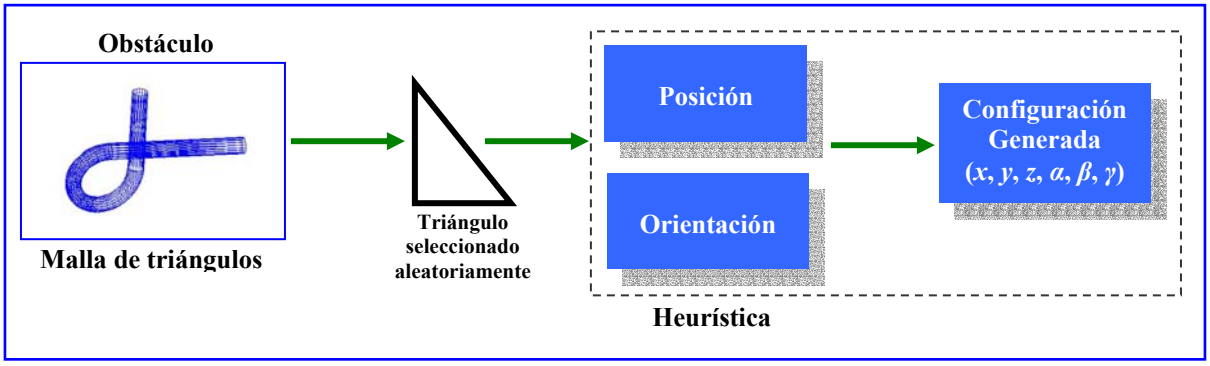

Figura 3.16 Etapas generales en la construcción de una configuración.

3.5 HEURÍSTICA PROBABILÍSTICA

La heurística diseñada es probabilística ya que varios de los parámetros necesarios para construir las configuraciones se determinan de manera aleatoria. En primer lugar el ángulo *s* requerido para construir el cuaternión de rotación se genera a través de la función generadora de números pseudoaletorios *Random*, dentro de un rango de 0 a 2π; rango que puede ser configurado por el usuario.

Además el triángulo del cual se obtiene su vector normal o algún lado de él, para funcionar como eje de rotación, también se escoge aleatoriamente de los triángulos que forman la malla del cuerpo del clavo. Escoger uno de los tres lados de un triángulo como eje de rotación también se realiza de manera aleatoria.

Del mismo triángulo aleatorio del cual se obtiene su vector normal o bien su lado también se obtiene el vector posición para el robot, ya sea el baricentro, un vértice aleatorio de los tres o bien un punto aleatorio sobre el plano que define ese triángulo.

Por otra parte, para mejorar aún más la generación de configuraciones se calcularon las áreas de todos los triángulos de la malla del obstáculo y en base a ella se determinó la probabilidad acumulada de escoger aleatoriamente cada triángulo, teniendo mayor probabilidad acumulada los de mayor área.

3.5.1 Área de un triángulo

El área de un triángulo puede ser calculada de la siguiente manera: Consideramos un triángulo de vértices *p*, *q* y *r*, calculamos los vectores $v_1 = r - p$ y $v_2 = q - p$. El vector *w* es el producto vectorial de los vectores anteriores $v_1 \times v_2$, la mitad de la magnitud de éste vector es el área del triángulo [24] (ver figura siguiente).

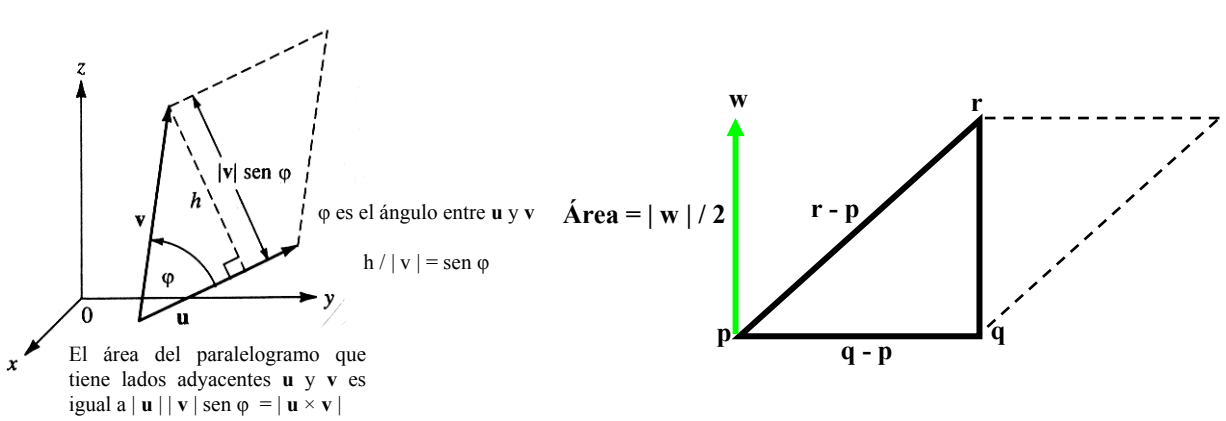

Figura 3.17 Cálculo del área de un triángulo.

El siguiente pseudocódigo muestra el método principal de la heurística, el cuál regresa como resultado una configuración dados los parámetros indicados.

Entrada: iEje e *ilado* son números enteros que indican el eje de rotación seleccionado, (*1 para el vector normal y 2 para uno de los lados*), *iPos* indica el punto escogido para posicionar el robot (*baricentro, algún vértice o punto sobre el triángulo*). *b* es un puntero al cuerpo del obstáculo.

Salida: Una configuración *c*.

```
1    NuevaConfiguracion(iEje, iPos, b, ilado)
2    {
3     Configuration c;
4     Eje          PosRobot, EjeQuat;
5     j ← b.GetNTriangle();
6     k ← Random(0, j);
7     EjeQuat   ← SelecEjeRot (iEje, k, b, ilado);
8     PosRobot ← SelecPos (iPos, k, b);
9     c          ← GeneraCfgQuat(PosRobot, EjeQuat);
10    return(c);
11   }
```

Como se observa existen varias funciones dentro del método anterior, a continuación describiremos brevemente cada una de ellas.

b.GetNTriangle, (*línea 5*) es un método propio de un objeto tipo cuerpo que regresa el número de triángulos presentes en su malla, en este caso nos referimos al cuerpo del

obstáculo apuntado por *b*, este tipo de métodos pertenecen al conjunto de clases definidas en el sistema GEMPA.

SelecEjeRot (*línea 7*) es un método de la heurística, que selecciona un eje de rotación según los parámetros enviados por el usuario, la normal del triángulo aleatorio *k* o bien uno de sus lados especificado por el entero *ilado*. El método *SelecPos* (*línea 8*), también como parte de la heurística, selecciona la posición del robot del mismo triángulo aleatorio k.

El método *GeneraCfgQuat* (*línea 9*) construye un cuaternión de rotación con el eje enviado, la matriz de rotación y por último extrae los ángulos de Euler para la orientación, formando así la configuración final que será devuelta por la heurística. El pseudocódigo es el siguiente:

Entrada: Los ejes *vPos* y *vEje* corresponden al punto donde el robot será colocado en el espacio de trabajo y el eje para construir el cuaternión de rotación respectivamente.

Salida: La configuración final *cfg*.

```
1     GeneraCfgQuat(Eje vPos, Eje vEje)
2     {
3       Configuration   cfg;
4       quaternion      q;
5       Matriz  MQuat;
6       s  = Random(0, 6.28);
7       q  = CuaterniondeRotacion(Eje vEje, s);
8       CreaMatrizQuat(q, MQuat);
9       AngulosEuler(MQuat, a, b, g);
10      cfg ← Configuration(vPos.x, vPos.y, vPos.z, a, b, g);
11      return(cfg);
12    }
```

El método *CuaterniondeRotacion* (*línea 7*) construye un cuaternión de rotación *q* con el eje *vEje* y el ángulo aleatorio *s*. Con *q* se construye una matriz de rotación en *CreaMatrizQuat (...)*, *Mquat* es un objeto tipo matriz definido en el conjunto de clases del sistema GEMPA.

De la matriz de rotación se obtienen los ángulos de Euler (*línea 9*) y finalmente se construye la configuración *cfg* (*línea 10*) que es devuelta por el método *NuevaConfiguracion(...)*.

3.6 DETECCIÓN DE COLISIONES

Cada una de las configuraciones generadas por la heurística participa en la etapa de conexión del roadmap en cada uno de los métodos probabilísticos implementados, sin embargo antes de ser enviadas son verificadas por colisión.
La verificación por colisión consiste en determinar si el cuerpo del robot colocado en esa posición y con tal orientación no intersecta de alguna manera el cuerpo del obstáculo. Esta verificación se realiza con ayuda de rutinas de detección de colisiones ya diseñadas e implementadas por grupos dedicados a estudiar ésta área.

Tal es el caso del grupo *Gamma* (*Geometric algorithm for modeling, motion and animation* por sus siglas en inglés) de la Universidad de Carolina del Norte, Estados Unidos [27], quienes ponen a disposición de la comunidad de robótica y animación por computadora un conjunto de rutinas de detección de colisiones.

La libreria empleada en esta tesis para verificar colisiones fue **RAPID** (*Robust and accurate polygon interference detection*, por sus siglas en inglés) [28] empleada comúnmente por la comunidad de robótica para la detección de colisiones de robots rígidos (*que no tienen articulaciones*) en ambientes estáticos. La librería ejecuta dos fases antes de decidir si dos cuerpos están en colisión o no.

La primer fase crea el *Convex Hull* de cada uno de los cuerpos y verifica si éstos no se intersectan, en caso contrario RAPID reporta que no hay colisión. Si hay intersección entonces se ejecuta la segunda fase, dónde a partir de diagramas de *Voronoi* se determina exactamente el punto o los puntos (vértices y aristas de ambos cuerpos) dónde existe la colisión así como el porcentaje de penetración de un cuerpo dentro del otro. Si esta fase

determina que los diagramas de Voronoi no se intersectan RAPID reporta que no hay colisión entre ambos cuerpos.

Librerías diferentes a RAPID podrían ser utilizadas en esta tesis para mejorar la eficiencia de la detección de colisiones (ver apéndice II).

3.7 PARTICULARIDAD DE LA HEURÍSTICA

En la sección 3.3.2 hablamos de colocar el centroide del robot en alguno de los puntos definidos sobre la superficie del obstáculo (*malla de triángulos*) y destacamos el hecho de que el centroide del cuerpo del clavo no se encuentra sobre su superficie sino cerca del centro de la circunferencia que forman sus dos puntas.

Esto nos permite generar configuraciones para el robot entrelazadas con el cuerpo del obstáculo, configuraciones que son muy adecuadas al tratar de resolver el problema de los clavos, tal y como veremos en los resultados mostrados en el capitulo siguiente (ver figura 3.18).

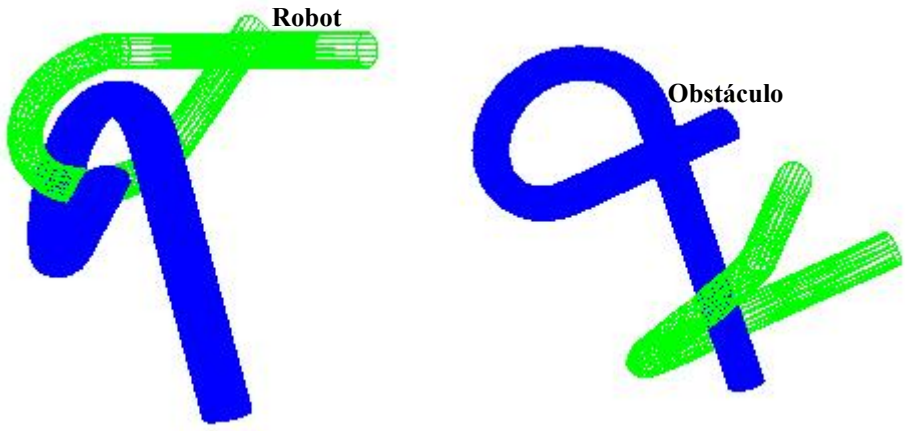

Figura 3.18 El centroide del robot se coloca en un punto seleccionado sobre la superficie del obstáculo generando configuraciones de éste tipo.

Sin embargo estos puntos no son validos para colocar un robot cuando los obstáculos del espacio de trabajo son cuerpos cuya forma es regular y poligonal, ya que colocar el centroide del robot en un vértice o en un punto sobre la superficie del obstáculo equivale inmediatamente a una colisión (ver figura 3.19).

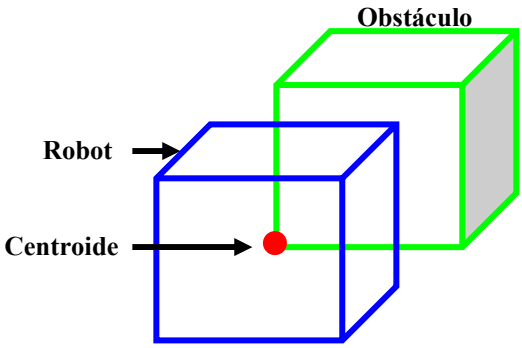

Figura 3.19 Si colocamos el centroide del robot en un vértice del obstáculo equivale a provocar una colisión entre ambos cuerpos.

Por este motivo la heurística diseñada en esta tesis esta destinada específicamente a generar configuraciones para el robot (clavo) en el problema de los clavos.