Robótica: Modelo Cinemático del Hexapod

Universidad Tecnológica Nacional - FRBA

Federico Ortiz federicoortiz12@gmail.com Felipe Diniello felipediniello@gmail.com

17 de junio de 2013

1. Introducción

Al dar una idea de que es un robot, se piensa que es una máquina que imita a un ser humano. Ellos no deben actuar como seres humanos, pero tienen que poder realizar diferentes tareas. Lo que se busca en realidad son máquinas que interactúen sin problemas con nuestro mundo. Permitiendo reemplazar al hombre en lugares donde las condiciones físicas no permiten que este las realice, sumado a que nunca se cansan y no pueden ser distraídos de la tarea en cuestión. El concepto de robots proviene del siglo 20 de la palabra checoslovaca Robota o Robotnik que significa esclavo, siervo o trabajo forzado. Los robots actuales tienen sistemas sensoriales que procesan la información y parecen funcionar como si tuvieran cerebro. Su *cerebro* es en realidad una forma de inteligencia artificial computarizada. Hay muchos tipos de robots: robots industriales, robots agrícolas, robots móviles, robots médicos, robots teleoperadores dentro de los robots móviles.

2. Hexapod

El hexapod es un robot de diversas aplicaciones, el que se trata en el grupo está inspirado en el tipo *araña*, tiene seis patas con 3 grados de libertad cada una que permite que se mueva flexiblemente en diversos terrenos. Una virtud y la mayor de ellas de esta configuración es su estabilidad. Que a diferencia de los bípedos, son estáticamente estables, por lo que no dependen de mecanismos de equilibrio. Pero si utilizan mecanismos de realimentación y poseen un movimiento más suave. Entre las aplicaciones del hexapod en la vida real se destacan, la búsqueda y rescate, transporte, exploración de medio ambiente o distintos tipos de superficies y también como máquinas de CNC.

3. Movimientos

Para poder desplazarse el hexapod dispone de distintos modos según las dificultades que presente la superficie [1]. Se debe disponer en todo momento de un mínimo de 3 articulaciones apoyadas para dar soporte y hacer tracción en todo momento, por ésta razón es que el robot posee un equilibrio estático. El mínimo de articulaciones apoyadas es 3, pero existen otras combinaciones que conviene usar en situaciones dónde la superficie de apoyo es más accidentada, en total son tres:

3/3 Se emplean 3 miembros de apoyo y tracción, mientras que los otros 3 son reposicionados.

4/2 Se emplean 4 miembros de apoyo y tracción, mientras que los otros 2 son reposicionados.

5/1 Se emplean 5 miembros de apoyo y tracción, mientras que solo 1 es reposicionado.

3.1. Movimientos de cada miembro

El movimiento de cada miembro se puede separar en dos etapas diferentes: *tracción* y *reposicionamiento*.
Tracción: de adelante hacia atrás, con Z constante en *forma de línea*.
Reposicionamiento: de atrás hacia adelante, con Z variable *por encima de los puntos fijos*
Éste movimiento se explica en la figura 1 donde se puede ver desde un plano lateral al robot el movimiento que debe hacer el extremo de cada brazo. Donde con una flecha individual se indica el sentido de desplazamiento del robot, con línea negra se indica cómo es el movimiento de *tracción* y con línea punteada el *reposicionamiento*.

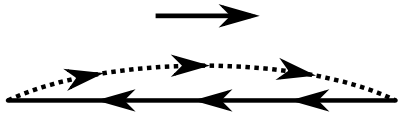

Figura 1: Vista lateral de un movimiento individual

3.2. Desplazamiento del robot completo

Para desplazarse se deberán coordinar los movimientos de los dos conjuntos de articulaciones. Todos los miembros de apoyo deben hacer exactamente el mismo movimiento de tracción hacia atrás, mientras que todos los miembros a reposicionar deberán adelantarse hasta la nueva posición para hacer el cambio *de apoyo*. Por lo que en todo momento existen dos grupos de miembros operando en *contrafase*.

4. Sistemas de Coordenadas

En el robot coexisten varios sistemas de coordenadas:

4.1. Propio

Sistema de referencia del *robot completo* como un *ente*. Que esta referenciado a un punto fijo de referencia en el espacio, con respecto a éste es que se desplaza el robot. El mismo podría estar referido mediante un GPS+giroscopo para determinar su verdadero valor.

4.2. Miembro

Tomando como $(0, 0, 0)$ la primer articulación se definen todos los parámetros de cada miembro. En el robot existen 6 de éstos sistemas, y para cada movimiento se debe coordinar que el movimiento sea coordinado en los 6 sistemas al mismo tiempo.

5. Cinemática Directa

Según algoritmo de Denavid-Hartenberg:

	θ	d	a	α
L_1	q_1	n_1	m_1	-90
L_2	q_2	0	m_2	0
L_3	q_3	0	m_3	0

$$ {}^{n-1}T_n = \left[\begin{array}{ccc|c} C\theta_n & -S\theta_n C\alpha_n & S\theta_n S\alpha_n & a_n C\theta_n \\ S\theta_n & C\theta_n C\alpha_n & -C\theta_n S\alpha_n & a_n S\theta_n \\ 0 & S\alpha_n & C\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] $$

5.1. Matrices de Transformaciones directas

$$ {}^0T_1 = \left[\begin{array}{ccc|c} Cq_1 & 0 & -Sq_1 & m_1 \cdot Cq_1 \\ Sq_1 & 0 & Cq_1 & m_1 \cdot Sq_1 \\ 0 & -1 & 0 & n1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (1) $$

$$ {}^1T_2 = \left[\begin{array}{ccc|c} Cq_2 & -Sq_2 & 0 & m_2 \cdot Cq_2 \\ Sq_2 & Cq_2 & 0 & m_2 \cdot Sq_2 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (2) $$

$$ {}^2T_3 = \left[\begin{array}{ccc|c} Cq_3 & -Sq_3 & 0 & m_3 \cdot Cq_3 \\ Sq_3 & Cq_3 & 0 & m_3 \cdot Sq_3 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (3) $$

Matriz total:

$$ {}^0T_3 = {}^0\,T_1 \cdot {}^1\,T_2 \cdot {}^2\,T_3 \quad (4) $$

6. Cinemática Inversa

Se emplea el método geométrico para resolver el problema de la cinemática inversa. Dado un punto (x_3, y_3, z_3) ubicado en el espacio y referido al sistema de coordenadas de base de la primer articulación(0), se debe operar de la siguiente manera:

Paso 1: q_1

La resolución de la primer articulación determina el plano en el que se va a trabajar con las siguientes articulaciones. Su solución, mucho menos compleja que las demás articulaciones, se obtiene de analizar el brazo desde una vista superior, en un plano XY, con Z salientes, como en la figura 2.

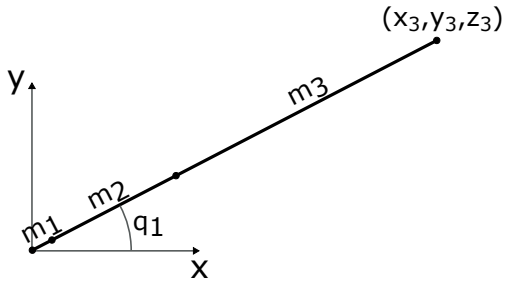

Figura 2: Vista superior

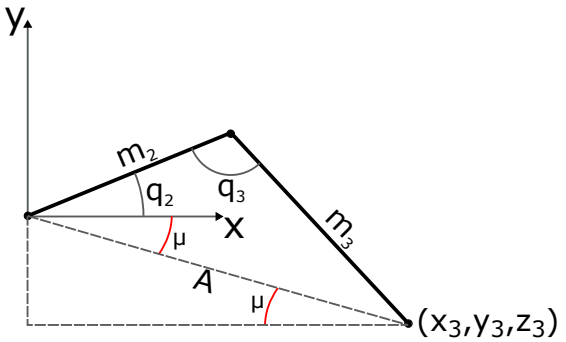

Figura 3: Vista del plano lateral

$$q_1 = \arctan\left(\frac{y_3}{x_3}\right) \tag{5}$$

Paso 2: ${}^0T_1$

Habiendo obtenido q_1 en el paso anterior ahora se pueden obtener todos los valores de la matriz ${}^0T_1$ (ver ecuación 1). Para luego poder hacer un cambio de coordenadas, y tomar como nuevo origen el comienzo de L_2. Ahora para seguir analizando el problema se toma una vista lateral del plano XY que contiene el resto del brazo, como se puede ver en la figura 3. Es necesario referir el punto original (x_3, y_3, z_3) al nuevo sistema de coordenadas de L_2. Las coordenadas del eje Z se omiten para simplificar el dibujo, ya que todos los puntos pertenecen al mismo plano.

$${}^1T_0 = \left({}^0T_1\right)^{-1} \tag{6}$$

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \\ 1 \end{bmatrix}_{L_1} = {}^1T_0 \cdot \begin{bmatrix} x_3 \\ y_3 \\ z_3 \\ 1 \end{bmatrix}_{L_0} \tag{7}$$

Paso 3: A

Las soluciones de las articulaciones restantes se obtiene mediante la aplicación del teorema de coseno, por lo que es necesario calcular el lado A restante del triángulo que forman L_2 y L_3.

$$A = \sqrt{x_3^2 + y_3^2} \tag{8}$$

Paso 4: μ

El valor de q_3 no se puede obtener directamente por el teorema del coseno, ya que ésta posición debe ser medida con respecto a la horizontal. Para ésto es necesario calcular μ, que es el ángulo que forma el lado A con respecto al piso.

$$\mu = \arcsin\left(\frac{y_3}{A}\right) \tag{9}$$

Paso 5: q_2

Aplicando el teorema del coseno, se puede obtener el valor de la suma de μ y q_2:

$$m_3^2 = m_2^2 + A^2 - 2Am_2\cos\left(q_2 + \mu\right) \tag{10}$$

Despejando nos queda que:

$$q_2 = \arccos\left(\frac{m_3^2 - m_2^2 - A^2}{-2Am_2}\right) - \mu \tag{11}$$

Paso 6: q_3

El valor de q_3 sale directamente del teorema del coseno:

$$q_3 = \arccos\left(\frac{A^2 - m_2^2 - m_3^2}{-2m_2m_3}\right) \tag{12}$$

7. Cálculo de Trayectorias

La implementación actual del programa está hecha en lenguaje C y con las librerías OpenCV. Con ésta combinación se obtienen todas las ventajas de

un lenguaje tan performante como C, y la posibilidad de realizar operaciones matriciales y/o vectoriales de un modo sencillo y en paralelo aprovechando al máximo el procesador utilizado (siempre y cuando sea compatible con OpenCV).

Para calcular todos los puntos intermedios de una trayectoria primero se debe partir de dos datos, un punto inicial $\bar{X}_i = (x_i, y_i, z_i)$ y un punto final $\bar{X}_f = (x_f, y_f, z_f)$. Se debe calcular un vector desplazamiento: $\vec{D} = \bar{X}_f - \bar{X}_i$, luego se lo divide por su módulo para calcular el versor de desplazamiento.

$$\breve{V}_D = \frac{\vec{D}}{|\vec{D}|} = \frac{\bar{X}_f - \bar{X}_i}{|\bar{X}_f - \bar{X}_i|} \tag{13}$$

Una vez obtenido éste versor todos los puntos intermedios que pertenecen a la trayectoria se obtienen como:

$$\bar{P}_n = \bar{X}_i + n \cdot \breve{V}_D \cdot k \tag{14}$$

Si se desea variar la densidad de puntos intermedios se puede modificar el módulo de $\breve{V}_D$ multiplicando todos sus términos por una constante k, como se puede ver en la ecuación 14. Para valores de $k < 1$ los puntos intermedios serán más próximos entre sí, y lógicamente para $k > 1$ los puntos estarán más distanciados.

Para obtener las coordenadas articulares de cada punto es necesario aplicar el algoritmo de cinemática inversa (sección **??**) para cada punto, así se obtienen las respectivas coordenadas (q_1, q_2, q_3).

$$(q_1, q_2, q_3)_n = AlgoritmoInverso\left\{\bar{P}_n\right\} \tag{15}$$

7.1. Ejemplo

Código de ejemplo utilizando OpenCV para obtener 100 puntos intermedios entre 2 puntos:

```
#include <cv.h>                                                          1
                                                                         2
CvPoint3D32f* trace_route(CvPoint3D32f                                   3
    Xi_data, CvPoint3D32f Xf_data )
{                                                                        4
  int       i;                                                           5
  CvPoint3D32f  *puntos_intermedios;                                     6
  //~ espacio para los 100 puntos:                                       7
  puntos_intermedios=malloc(100*sizeof(                                  8
      CvPoint3D32f));
  //~ espacio para los datos                                             9
  CvPoint3D32f versor_data, desp_data,                                   10
      aux_data;
  //~ Encabezados de matrices para OpenCV                                11
  CvMat    versor, desp, aux, X_i, X_f;                                  12
                                                                         13
  //~ Se inicializan los encabezados con sus                             14
       respectivos datos
  versor=cvMat(1,3,CV_32FC1,&versor_data);                               15
  desp=cvMat(1,3,CV_32FC1,&desp_data);                                   16
  aux=cvMat(1,3,CV_32FC1,&aux_data);                                     17
  X_i=cvMat(1,3,CV_32FC1,&Xi_data);                                      18
  X_f=cvMat(1,3,CV_32FC1,&Xf_data);                                      19
                                                                         20
  //~ desplazamiento = X_f - X_i                                         21
  cvSub(&X_f,&X_i,&desp,NULL);                                           22
                                                                         23
  //~ en este caso se desean obtener 100                                 24
      puntos intermedios
  //~ versor = desplazamiento ./ 100                                     25
  aux_data=cvPoint3D32f( 1/100 , 1/100                                   26
      ,1/100);
  cvMul(&desp,&aux,&versor,1);                                           27
                                                                         28
  //~ Uso la variable auxiliar como el punto                             29
       inicial
  aux_data=Xi_data;                                                      30
  for (i = 0; i < 100; i++) //~ itero 100                                31
      veces:
  {                                                                      32
    //~ guardo ese punto                                                33
    *(puntos_intermedios+i)=aux_data;                                    34
    //~ aux= aux + versor                                               35
    cvAdd(&aux,&versor,&aux,NULL);                                       36
  }                                                                      37
  return puntos_intermedios;                                             38
}                                                                        39
```

7.2. Trayectorias parabólicas

Como se explicó en la sección 3 el movimiento de una pata está separado en dos etapas, una de tracción (movimiento lineal) y otra de reposicionamiento (por arriba del plano donde se apoya el robot). El segundo movimiento re realiza con igual frecuencia que el primero ya que son las dos mitades de un *movimiento de desplazamiento*, es por eso que para simplificar el proceso de cálculos se agrega la posibilidad de calcular trayectorias parabólicas.

Con solo un parámetro adicional a la función empleada para dibujar una línea, se indica el salto de altura por encima de cualquiera de los dos puntos ($\bar{X}_i$ o $\bar{X}_f$). El movimiento en x e y se realiza de la misma manera, solo es el valor en z que se modifica. Para calcular los parámetros (a, b y c)de la parábola se disponen de 3 puntos: $\bar{X}_i$, $\bar{X}_f$ y $\bar{X_m}$, por lo que se plantean 3 ecuaciones, y se resuelve para los coeficientes del polinomio que son las incógnitas.

$$z = f(\omega) = a \cdot \omega^2 + b \cdot \omega + c \tag{16}$$

$$\begin{cases} \bar{X}_i \to (\omega_i, z_i) = (0, z_i) \\ \bar{X_m} \to (\omega_m, z_{max}) = (\frac{\omega_f}{2}, z_{max}) \\ \bar{X}_f \to (\omega_f, z_i) = (\omega_f, z_f) \\ z_{max} = MAX(z_i, z_f) + salto \end{cases} \tag{17}$$

Donde ω representa el eje del movimiento sobre el plano XY, por lo que se debe transformar a los puntos $\bar{X}_i$, $\bar{X}_f$ y $\bar{X_m}$, para que estén en coordenadas de ω y z. El máximo de altura se ubica a la mitad del recorrido.

$$\begin{cases} a \cdot \omega_i^2 + b \cdot \omega_i + c = z_i \\ a \cdot \omega_m^2 + b \cdot \omega_m + c = z_{max} \\ a \cdot \omega_f^2 + b \cdot \omega_f + c = z_f \end{cases} \tag{18}$$

8. Detección de límites

Las limitaciones para cada para robótica están definidas en las articulaciones, es decir que para una cierta q_n se definen dos valores: $q_{n,min}$ y $q_{n,max}$ dentro de los que la articulación puede operar sin problemas, por lo que al tener 3 articulaciones, se tienen 6 márgenes (3 mínimos y 3 máximos). Con éstos márgenes se puede determinar un conjunto de puntos dentro del marco de referencia para discernir que puntos son válidos y cuales no. El algoritmo para la detección de límites se divide en dos partes

1. Obtención de los límites.
2. Validación *punto a punto*.

Cuando se calcula una trayectoria en coordenadas cartesianas es útil saber cuando va a llegar al límite sin necesidad de realizar la transformación inversa, para luego comprobar que dicho punto (x, y, z) no pertenece al sub-espacio de puntos válidos. Éste proceso de cálculo de trayectorias a ciegas tiene la deficiencia de que hay que calcular las tres coordenadas articulares para ver si las coordenadas son válidas, lo cual significa poder de cómputo desperdiciado. Con el algoritmo empleado se puede *estimar* la validez del punto dentro de un margen y luego aplicarle el algoritmo inverso.

8.1. Etapas:

8.1.1. Obtención de los límites

Para la obtención de los límites primero se determina un espacio **discreto** de trabajo mayor al espacio de puntos válidos, de manera que el segundo quede contenido en el primero. Luego uno a uno se le aplica el algoritmo inverso a cada punto del espacio de trabajo, y el resultado del mismo (éxito o fracaso) se guarda en esa posición para su posterior uso.

8.1.2. Validación

Durante el trazado de una ruta se debe verificar que cada punto esté contenido al espacio válido de coordenadas. Para ello se redondean sus coordenadas cartesianas al entero más próximo, y se accede al espacio de trabajo previamente calculado usando dichas coordenadas (ahora números enteros) como índices. Accediendo a los datos guardados en una matriz tridimensional con las coordenadas ahora como índices se recupera el dato guardado que corresponde a su pertenencia o no al conjunto de coordenadas válidas.

8.2. Discusión del método

El método propuesto cuenta con algunas ventajas, por un lado siempre se emplea el algoritmo inverso con coordenadas válidas y por el otro la estimación de valides se realiza de forma muy rápida ya que los datos fueron precalculados.

La principal desventaja es el uso de memoria excesivo para guardar la validez de los puntos de antemano ya que el espacio de trabajo es proporcional al largo de las extremidades. De todas maneras ésto puede optimizarse para que el espacio discreto no represente 1:1 el espacio real sino que exista un escalamiento.

Referencias

[1] Mohammad Al-Jabari Tareq Mamkegh, Ahmad Hindash. *Hexapod Robot, Robot design, model and control.* PhD thesis, German Jordanian University, 2011.