3

Modelos cinemáticos de robots

Introducción

Para tratar la cinemática de los robots manipuladores se ha optado por emplear exclusivamente MATLAB, ya que en este caso resulta más cómodo que Simulink. Eso es debido a que la mayoría de datos involucrados en la cinemática son matrices, que pueden ser tratadas más facilmente desde MATLAB que desde Simulink.

En cuanto a la cinemática de los robots móviles, se ha optado por el uso de Simulink para la simulación de los modelos cinemáticos. Eso se debe a que en las simulaciones de dichos modelos solamente se involucran operaciones con vectores que dependen del tiempo.

También se dispone de funciones para la representación gráfica esquemática de ciertos modelos de robots móviles.

En el Capítulo 4 de Ollero [3], se pueden encontrar ejemplos que ilustran el uso de las funciones y bloques de este Capítulo.

En la siguiente sección se definen una serie de vectores y matrices con objeto de facilitar la comprensión de algunas funciones de este capítulo.

La transformación Δ de traslación y rotación diferenciales.

En Paul [3] se define la matriz de transformación de traslación y rotación diferenciales como:

$$\Delta = \text{trans}\ (dx, dy, dz)\ \text{rot}\ (k, d\theta) - I \tag{3.1}$$

donde:

- Trans (dx, dy, dz) es la transformación que representa una traslación dx, dy, dz.
- Rot$(k, d\theta)$ es la transformación que representa una rotación diferencial $d\theta$ en torno a un vector k.

Sean los vectores de traslación y rotación diferenciales siguientes:

$$d = d_x\, i + d_y\, j + d_z\, k \tag{3.2}$$

$$\delta = \delta_x\, i + \delta_y\, j + \delta_z\, k \tag{3.3}$$

A veces, estos dos vectores se agrupan en uno solo que recibe el nombre de vector de movimiento diferencial:

$$D = \begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \tag{3.4}$$

En Paul [3] se llega a una expresión de Δ, en función de los elementos de este vector:

$$\Delta = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

Instrucciones MATLAB relacionadas con la cinemática de los robots manipuladores

Para resolver la cinemática de un robot solamente se necesitan conocer sus parámetros de Denavit-Hartenberg según la notación de Craig [1]. Dichos parámetros habrán de ser introducidos en una matriz según un determinado formato que se explica en el apartado denominado `dh` dentro de esta sección. Esta matriz será la que habrá que pasar como parámetro a la mayoria de las funciones de MATLAB de este capítulo, por lo que se recomienda que el lector asimile bien el contenido de la sección `dh`.

Asimismo, se adopta el convenio de emplear vectores fila para introducir las variables articulares en aquellas funciones en que sea necesario.

En la siguiente tabla se resumen las instrucciones que se emplean para efectuar ciertos cálculos relacionados con la cinemática de los robots manipuladores. Algunos de ellas no se utilizan directamente, pero aparecen en el cuerpo de funciones que sí se usan, por lo que también han sido descritas.

Tabla 3.1: Instrucciones relacionadas con la cinemática.

	Propósito
`dh`	Contener los parámetros de Denavit-Hartenberg del manipulador
`diff2tr`	Convertir un vector de movimiento diferencial en la correspondiente transformación homogénea
`fkine`	Calcular la cinemática directa de un robot manipulador
`ikine`	Calcular la cinemática inversa de un manipulador

jacob0	Calcular el Jacobiano del manipulador expresado en el sistema de referencia base
jacobn	Calcular el Jacobiano expresado en el sistema de coordenadas del efector final
linktrans	Calcular las matrices de transformación a partir de los parámetros de Denavit-Hartenberg
numcols	Calcular el número de columnas de una matriz
numrows	Calcular el número de filas de una matriz
plotbot	Representar gráficamente el manipulador
tr2diff	Convertir una transformación en un vector de movimiento diferencial y calcular la diferencia entre dos transformaciones
tr2jac	Calcular la matriz (que se podría llamar Jacobiana) que permite pasar un vector de velocidades cartesianas de un cuadro de referencia a otro distinto
velprop	Calcular el vector de velocidades en el espacio cartesiano expresado en el sistema de coordenadas del cuadro $\{n\}$

dh

Propósito

Contener los parámetros de Denavit-Hartenberg del manipulador.

Descripción

Cada vez que se quiera utilizar una función de HEMERO relacionada con la cinemática se deberán introducir en una matriz los parámetros de Denavit-Hartenberg del manipulador, de acuerdo con la notación de Craig. El modo de introducir dicha información en esa matriz es el siguiente:

- Habrá una fila por cada enlace que tenga el manipulador.
- Cada fila tendrá el siguiente formato:

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_{i-1} & a_{i-1} & \theta_i & d_i & \sigma_i \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

donde:

- $\alpha_{i-1}, a_{i-1}, \theta_i, d_i$: Son los parámetros de Denavit-Hartenberg según Craig [1].
- σ_i: Indicará el tipo de articulación (será 0 si es de rotación y un número distinto de cero si por el contrario es prismática).

Así pues para un robot con *n* enlaces, la matriz quedaría del siguiente modo:

$$\begin{bmatrix} \alpha_0 & a_0 & \theta_1 & d_1 & \sigma_1 \\ \alpha_1 & a_1 & \theta_2 & d_2 & \sigma_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_{i-1} & a_{i-1} & \theta_i & d_i & \sigma_i \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_{n-1} & a_{n-1} & \theta_n & d_n & \sigma_n \end{bmatrix}$$

que es una matriz *n*x5 donde el subíndice *i* denotaría el enlace *i*-ésimo del manipulador.

Todos los ángulos deberán ser introducidos en radianes. Las longitudes a_{i-1} y d_i podrán ser expresadas en cualquier unidad y sólo habrá que tener cuidado de recordar que las transformaciones homogéneas y los jacobianos que se calculen tendrán esas mismas unidades.

Ejemplos

Ver Ejemplos 4.1, 4.2, 4.5, 4.6 y 4.7 en Ollero [2].

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

diff2tr

Propósito

Convertir un vector de movimiento diferencial en la correspondiente transformación homogénea.

Sintaxis

```
T=diff2tr(D)
```

Descripción

Devuelve la transformación homogénea (`T`) de traslación y rotación diferenciales que corresponde al vector de movimiento diferencial (`D`) que se le pasa como parámetro.

Algoritmo

Para un vector de movimiento diferencial $D = \left[d_x\ d_y\ d_z\ \delta_x\ \delta_y\ \delta_z\right]^T$ la correspondiente transformación homogénea viene dada por:

$$\Delta = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & \delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

Ver también

`tr2diff`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Paul, R.P., *Robot manipulators: Mathematics, Programming, and Control*. Cambridge, Massachusetts: MIT Press, 1981.

fkine

Propósito

Calcular la cinemática directa de un robot manipulador.

Sintaxis

```
T=fkine(dh,q)
```

Descripción

La función fkine se encarga de calcular la cinemática directa de un manipulador para un estado dado por el vector de variables articulares `q`. Dicho manipulador quedará descrito por su matriz de parámetros de Denavit-Hartenberg (`dh`).

Es posible utilizar esta función de dos modos:

- Si `q` es un vector, entonces es interpretado como el vector de variables articulares para el que se pretende calcular el modelo directo, y `fkine` devuelve la transformación homogénea (`T`) correspondiente al último enlace del manipulador. En este caso `q` deberá ser un vector fila.

- Si `q` es una matriz, cada fila será interpretada como un vector de variables articulares, y `T` será una matriz tridimensional 4x4xm, siendo m el número de filas de `q`. `T` es lo que se denominará una trayectoria de transformaciones homogéneas.

  Para extraer la transformación homogénea (`Ti`) que hay "condensada" en la fila i-ésima de la matriz `T`, bastará con escribir:

  ```
  Ti=T(:,:,i)
  ```

Empleando la función `fkine` según este último modo, es posible obtener todas las transformaciones homogéneas correspondientes a una trayectoria dada en el espacio de variables articulares.

Algoritmo

Básicamente esta función se encarga de hacer:

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{N-1}_{N}T \tag{3.7} $$

donde N es el número de filas de la matriz `dh`, que se corresponde con el número de enlaces del manipulador.

Ejemplos

Ver Ejemplos 4.1 y 4.3 en Ollero [2].

Ver también

`dh`, `linktrans`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Craig, J.J., *Introduction to robotics*. Addison Wesley, segunda ed., 1989.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

ikine

Propósito

Calcular la cinemática inversa de un manipulador.

Sintaxis

```
q=ikine(dh,T,stol,ilimit)
q=ikine(dh,T,stol,ilimit,q0)
q=ikine(dh,T,stol,ilimit,q0,M)
```

Descripción

`Ikine` devuelve los valores de las variables articulares necesarios para que el efector final del manipulador tenga la posición y orientación dadas por la transformación `T`. La solución del problema cinématico inverso no es única en general, y es posible que para una misma orientación y posición deseadas, se obtengan soluciones distintas en función del vector inicial de variables articulares (`q0`) que se le pase a `ikine`.

Es posible usar la función para que devuelva las variables articulares correspondientes a una sóla posición y orientación, o bien a una trayectoria de posiciones y orientaciones. Eso dependerá del formato del parámetro `T`:

- Si `T` es una transformación homogénea, entonces `ikine` devuelve un vector fila (`q`) con las variables articulares correspondientes a la posición y orientación indicadas en la matriz `T`.
- Si `T` es una trayectoria de transformaciones homogéneas, entonces el resultado será una matriz (`q`), en la que la fila i-ésima contendrá las variables articulares correspondientes a la transformación `T(:,:,i)`. La estimación inicial para `q` en cada paso se toma de la solución obtenida en el paso anterior.

Sea cual sea el formato de `T`, la estimación inicial para el vector de variables articulares será la dada en el parámetro `q0` (puede ser una columna o una fila), y en el caso de que no se le de, se asume que es el vector nulo.

Para el caso de un manipulador con menos de 6 grados de libertad el efector final no podrá alcanzar algunas posiciones y orientaciones. Esto normalmente lleva a una no convergencia de `ikine`. Una solución consiste en especificar un vector (fila o columna) de pesos (`M`), cuyos elementos serán 0 para aquellos grados de libertad que en cartesianas estén restringidos, y 1 en otro caso. Los elementos de `M` se corresponden con las traslaciones a lo largo de los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$, y con las rotaciones en torno a los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$. En el ejemplo de esta sección, `M` es el vector [1 1 0 0 0 1], porque el manipulador no se puede desplazar a lo largo del eje $\hat{Z}$, ni rotar en torno a los ejes $\hat{X}$ e $\hat{Y}$. El

número de elementos no nulos debe ser igual al número de grados de libertad del robot.

`Ilimit` es el número máximo de iteraciones que se ejecutarán en busca de una solución (un valor usual es 1000).

`Stol` será la máxima diferencia que se admitirá entre la transformación correspondiente a las variables articulares solución y la transformación con la posición y orientación especificadas (un valor usual es 10^{-6}). Dicha diferencia se mide haciendo uso de la función `tr2diff`.

Algoritmo

Lo que se hace en cada iteración es variar el valor de las variables articulares de tal modo que se reduzca el error entre la transformación actual (correspondiente a las variables articulares en esa iteración) y la transformación `T` requerida. La mencionada variación se calcula empleando la pseudoinversa del Jacobiano y la diferencia entre transformaciones usando la función `tr2diff`.

Precauciones

En primer lugar se debe tener en cuenta que este algoritmo no siempre converge. Puede que incluso no haya convergencia para puntos y orientaciones del espacio alcanzable.

Por otro lado este método es menos eficiente que las soluciones analíticas del modelo inverso que se obtengan para un manipulador determinado.

Por último señalar que es necesario emplear las mismas unidades a la hora de escribir la cuarta columna de la transformación `T` y la matriz de parámetros de Denavit-Hartenberg (`dh`).

Ejemplos

Para ilustrar el uso de la función `ikine`, se va a aplicar al manipulador del Ejemplo 4.1 de Ollero [2], para el que se va a suponer l_1= 4m y l_2= 3m.

Ejemplo H.3.1 (archivo `ejh31.m`)

Lo primero que se debe tener presente es que el método numérico empleado para obtener el modelo inverso no siempre converge. Habrá regiones del espacio en las que algoritmo no convergerá aunque dichas regiones pertenezcan al espacio alcanzable, lo cual representa una limitación importante. En la Figura 3.1 están representados los puntos del área [0,7]x[0,7] en los que no hay convergencia para una tolerancia de 10^{-6} y un número de iteraciones igual a 1000 (el barrido del área se ha efectuado de 0.1 en 0.1). La estimación inicial del vector de variables articulares para el cálculo del modelo inverso en los puntos de dicha área será `[0 0 0]`.

Es posible elegir una trayectoria de la región donde hay convergencia para comprobar como el manipulador se mueve por ella. Una posible trayectoria sería un arco de circunferencia de radio igual a 5 m.

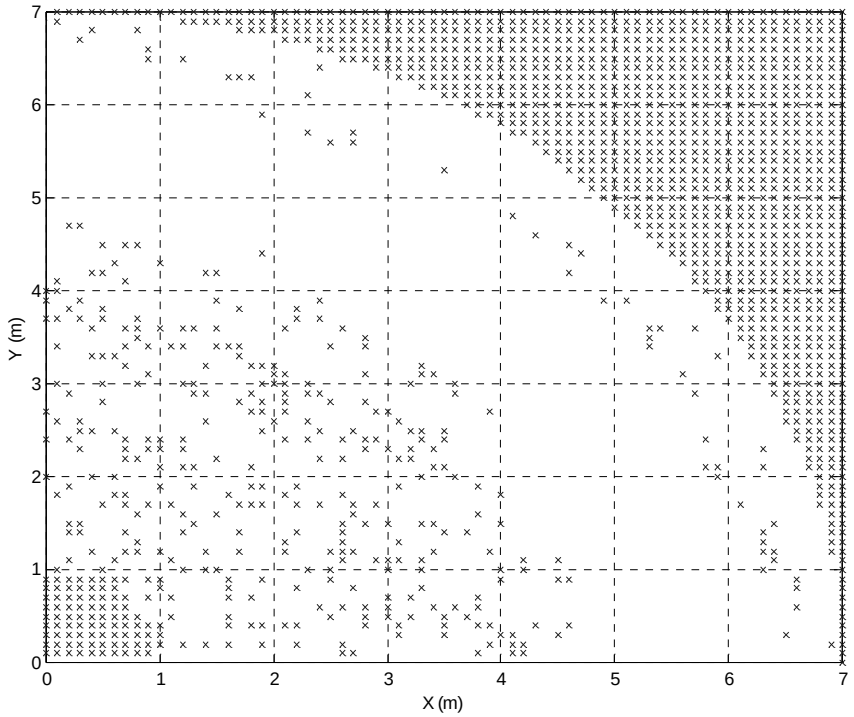


Figura 3.1: Puntos de no convergencia del algoritmo `ikine` para una tolerancia de de 10^{-6} y un número de iteraciones igual a 1000.

Las órdenes que se muestran a continuación, se utilizan para calcular el valor que deben tener las variables articulares para cada punto del arco de circunferencia anteriormente mencionado:

```
t1=0; t2=0; t3=0;   % Los valores que se pongan aquí para θ1, θ2
                      y θ3 son irrelevantes

l1=4; l2=3;

% Se crea la matriz dh con los parámetros de Denavit-
  Hartenberg del manipulador

dh = [ 0 0  t1 0 0;
       0 l1 t2 0 0;
       0 l2 t3 0 0];

% Se establece la tolerancia (stol) y el número máximo de
  iteraciones (ilimit)
```

```
stol=1e-6; ilimit=1000;

% Se introduce la trayectoria (arco de radio 5) y la
  orientación (0 rad) deseadas

x=0:0.05:5;
y=sqrt(25-x.^2);

phi=zeros(1,length(x));% phi es un vector fila nulo de la
                       misma dimensión que x

% Se crea la trayectoria de transformaciones

TG=[ ];
for k=1:length(x)
   T=[cos(phi(k)) -sin(phi(k)) 0 x(k); % T =
      sin(phi(k)) cos(phi(k)) 0 y(k);
      0 0 1 0;
      0 0 0 1];
   TG=[TG;T(:)'];%En cada iteración se añade una fila más a TG
end

% Se calcula el modelo inverso para cada uno de los puntos de
  la trayectoria, usando un vector inicial q0=[0 0 0] y una
  máscara M =[1 1 0 0 0 1]

q=ikine(dh,stol,ilimit,TG,[0 0 0],[1 1 0 0 0 1]);
```

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & x \\ \sin\phi & \cos\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ahora sólo resta representar gráficamente el manipulador mientras recorre la trayectoria. Para ello se utiliza la función `plotbot` que se describe en este mismo Capítulo, a la que simplemente hay que pasarle la matriz `dh` del manipulador y la matriz `q` que devuelve `ikine` con los valores de las variables articulares correspondientes a cada punto de la trayectoria. Es decir:

```
plotbot(dh,q,'dfw')
```

Con el parámetro `'d'` se especifica que la representación sea bidimensional (en el plano XY), ya que el manipulador se encuentra en dicho plano. Mediante `'f'` y `'w'` se indica que se representen los cuadros de referencia de cada una de las articulaciones y el cuadro de referencia asociado al efector final respectivamente.

En la Figura 3.2 se muestran algunas de las posiciones del manipulador a lo largo de la trayectoria.

El método numérico empleado tiene sus limitaciones como ya se ha mencionado, pero en teoría permite calcular el modelo inverso de cualquier manipulador. En los Ejemplos 4.3 y 4.4 de Ollero [2], se calcula dicho modelo inverso analíticamente. En la Tabla 3.2 se compara, para el caso del arco de circunferencia, el número de operaciones requeridas por el método numérico

con las del método analítico. Se observa que siempre que sea posible, interesa obtener el modelo inverso del manipulador de forma analítica.

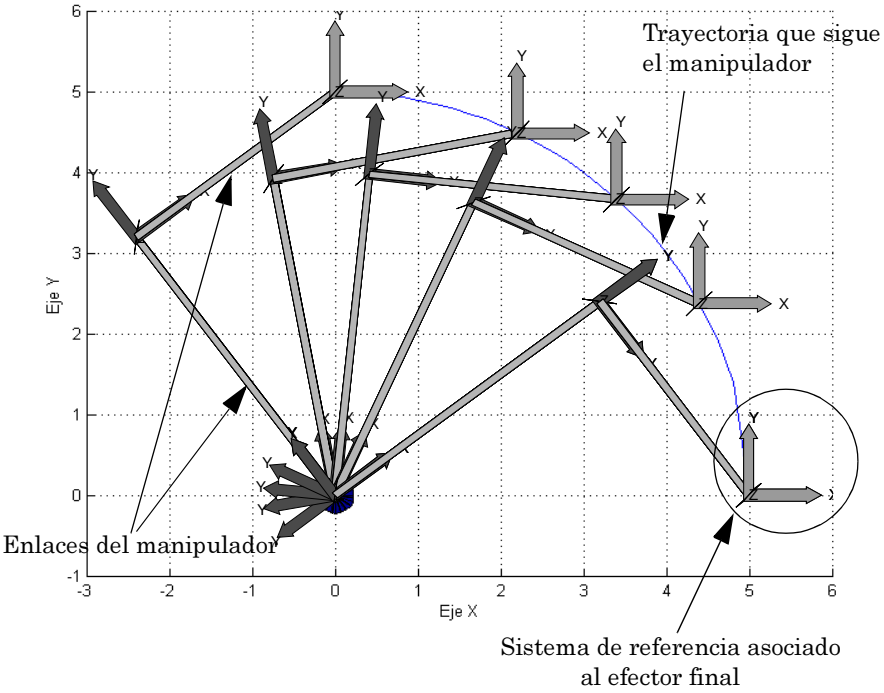


Figura 3.2: Representación gráfica del manipulador en cinco de los puntos de la trayectoria especificada.

Tabla 3.2: Número de flops de MATLAB (Pentium MMX 200 MHz).

	Método analítico	Método numérico
Manipulador Ejemplo 4.1	2428	754963

Ver también

`fkine`, `jacob0`, `tr2diff`

Referencias

Chieaverini, S., L. Sciavicco, B. Siciliano, "Control of robotic systems through singularities", en *Proc. Int. Workshop on Nonlinear and Adaptive Control: Issues in Robotics* (C. C. de Wit, ed.), Springer-Verlag, 1991.

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

jacob0

Propósito

Calcular el Jacobiano del manipulador expresado en el sistema de referencia base.

Sintaxis

```
jacob0(dh,q)
```

Descripción

Esta función devuelve el Jacobiano, correspondiente a las variables articulares q, expresado en el cuadro de referencia de la base del robot.

La matriz Jacobiana ${}^0J(q)$ permite obtener el vector de velocidades en el espacio cartesiano (${}^0\upsilon$) del efector final expresado en el cuadro {0} (base del manipulador) a partir de las derivadas de las variables articulares (q')en virtud de la expresión:

$$ {}^0\upsilon = {}^0J(q) \cdot q' \tag{3.8} $$

El vector de velocidades ${}^0\upsilon$ está formado por las velocidades lineales a lo largo de los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$, y por las velocidades angulares de giro en torno a los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$, expresadas ambas en el cuadro {0}. Es decir:

$$ {}^0\upsilon = \begin{bmatrix} {}^0\nu \\ {}^0\omega \end{bmatrix} \tag{3.9} $$

Así pues, si el manipulador tiene menos de seis grados de libertad en el Jacobiano aparecerán filas de ceros. En concreto se puede establecer la siguiente correspondencia:

Fila nula	No es posible
1	Traslación a lo largo del eje $\hat{X}$
2	Traslación a lo largo del eje $\hat{Y}$
3	Traslación a lo largo del eje $\hat{Z}$

Fila nula	No es posible
4	Rotación en torno al eje $\hat{X}$
5	Rotación en torno al eje $\hat{Y}$
6	Rotación en torno al eje $\hat{Z}$

Para un manipulador con *n* articulaciones, el Jacobiano es una matriz 6x*n*.

Algoritmo

Esta función se apoya en otra llamada `jacobn`, que calcula el Jacobiano del efector final expresado en el sistema de referencia del último enlace (*n*). `Jacob0` lo único que hace es multiplicar por las correspondientes matrices de transformación para expresar en el cuadro {0} el Jacobiano que devuelve `jacobn`.

Ejemplos

Ver Ejemplo 4.5 en Ollero [2]. En dicho ejemplo la matriz solución es:

$$ {}^0J(\theta) = \begin{bmatrix} -(l_1 s_1 + l_2 s_{12}) & -l_2 s_{12} & 0 \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad (3.10) $$

En ella se observa que hay tres filas nulas (la 3ª,4ª y 5ª), que se corresponden con la imposibilidad del movimiento a lo largo del eje $\hat{Z}$ y de las rotaciones en torno a los ejes $\hat{X}$ e $\hat{Y}$, resultado que es totalmente coherente a la vista del dibujo del manipulador (Figura 4.3 en Ollero [2]).

Ver también

`diff2tr`, `jacobn`, `tr2diff`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Paul, R.P., *Robot manipulators: Mathematics, Programming, and Control*. Cambridge, Massachusetts: MIT Press, 1981.

jacobn

Propósito

Calcular el Jacobiano expresado en el sistema de coordenadas del efector final.

Sintaxis

```
jacobn(dh,q)
```

Descripción

`Jacobn` devuelve el Jacobiano expresado en el sistema de referencia del efector final que corresponde a los valores de las variables articulares contenidos en `q`.

La matriz Jacobiana ${}^{n}J(q)$ permite obtener el vector de velocidades en el espacio cartesiano (${}^{n}\upsilon$) del efector final expresada en el cuadro {*n*} (asociado al efector final) a partir de las derivadas de las variables articulares (q') en virtud de la expresión:

$$ {}^{n}\upsilon = {}^{n}J(q) \cdot q' \tag{3.11}$$

El vector de velocidades ${}^{n}\upsilon$ está formado por las velocidades lineales a lo largo de los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$, y por las velocidades angulares de giro en torno a los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$, expresadas ambas en el cuadro {*n*}. Es decir:

$$ {}^{n}\upsilon = \begin{bmatrix} {}^{n}\nu \\ {}^{n}\omega \end{bmatrix} \tag{3.12}$$

Así pues, si el manipulador tiene menos de seis grados de libertad en el Jacobiano aparecerán filas de ceros. En concreto se puede establecer la siguiente correspondencia:

Fila nula	No es posible
1	Traslación a lo largo del eje $\hat{X}$
2	Traslación a lo largo del eje $\hat{Y}$
3	Traslación a lo largo del eje $\hat{Z}$

Fila nula	No es posible
4	Rotación en torno al eje $\hat{X}$
5	Rotación en torno al eje $\hat{Y}$
6	Rotación en torno al eje $\hat{Z}$

Para un manipulador con *n* articulaciones, el Jacobiano es una matriz 6x*n*.

Algoritmo

La función `jacobn` emplea un algoritmo basado en un desarrollo similar al que se encuentra en Paul y otros [4]. Dicho algoritmo consiste en lo siguiente:

Dada una matriz T:

$$T = {}^{i}_{i+1}T\,{}^{i+1}_{i+2}T \ldots {}^{n-1}_{n}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Para i =1 hasta n -1:

- Si la articulación i es de rotación:

$${}^{n}d_{i} = (-n_x\,p_y + n_y\,p_x)\,i + (-o_x\,p_y + o_y\,p_x)\,j + (-a_x\,p_y + a_y\,p_x)\,k \tag{3.14}$$

$${}^{n}\delta_{i} = n_z\,i + o_z\,j + a_z\,k \tag{3.15}$$

- Si la articulación i es prismática:

$${}^{n}d_{i} = n_z\,i + o_z\,j + a_z\,k \tag{3.16}$$

$${}^{n}\delta_{i} = 0\,i + 0\,j + 0\,k \tag{3.17}$$

Para $i = n$:

- Si la articulación n es de rotación:

$${}^{n}d_{n} = 0\,i + 0\,j + 0\,k \tag{3.18}$$

$${}^{n}\delta_{n} = 0\,i + 0\,j + 1\,k \tag{3.19}$$

- Si la articulación n es prismática:

$$^{n}\mathrm{d}_{n} = 0\,i + 0\,j + 1\,k \tag{3.20}$$

$$^{n}\delta_{n} = 0\,i + 0\,j + 0\,k \tag{3.21}$$

Los vectores obtenidos en la i-ésima iteración conforman la columna i-ésima del Jacobiano, es decir:

$$^{n}J(q) = \begin{bmatrix} ^{n}d_{1_x} & ^{n}d_{2_x} & \dots & ^{n}d_{n_x} \\ ^{n}d_{1_y} & ^{n}d_{2_y} & \dots & ^{n}d_{n_y} \\ ^{n}d_{1_z} & ^{n}d_{2_z} & \dots & ^{n}d_{n_z} \\ ^{n}\delta_{1_x} & ^{n}\delta_{2_x} & \dots & ^{n}\delta_{n_x} \\ ^{n}\delta_{1_y} & ^{n}\delta_{2_y} & \dots & ^{n}\delta_{n_y} \\ ^{n}\delta_{1_z} & ^{n}\delta_{2_z} & \dots & ^{n}\delta_{n_z} \end{bmatrix} \tag{3.22}$$

Ejemplos

Ver Ejemplos 4.5, 4.6 y 4.7 en Ollero [2].

Ver también

`diff2tr`, `jacob0`, `tr2diff`

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Paul, R.P., *Robot manipulators: Mathematics, Programming, and Control*. Cambridge, Massachusetts: MIT Press, 1981.

Paul, R.P., Shimano y Mayer, *Differential kinematic control equations for simple manipulators*, IEEE SMC 11(6), pp. 456-460, 1981.

linktrans

Propósito

Calcular las matrices de transformación a partir de los parámetros de Denavit-Hartenberg.

Sintaxis

```
T=linktrans(alpha,a,theta,d)
T=linktrans(dh,q)
```

Descripción

`Linktrans` calcula la matriz de transformación entre sistemas de referencia asociados a enlaces adyacentes a partir de los parámetros de Denavit-Hartenberg expresados según la notación de Craig [1].

Hay dos formas posibles de sintaxis:

- `T=linktrans(alpha,a,theta,d)`

  Calcula la matriz de transformación que corresponde a los parámetros $\alpha_{i-1}, a_{i-1}, \theta_i, d_i$.

- `T=linktrans(dh,q)`

  El primer parámetro que se le pasa es una fila de la matriz de parámetros de Denavit-Hartenberg (ver `dh`). El segundo parámetro es el valor de la variable articular para el que se quiere evaluar la matriz de transformación.

  Pero es evidente que en este caso es preciso incluir una información que con la sintaxis anterior no era necesaria. Dicha información es el tipo de articulación de que se trata, ya que de no hacerlo sería imposible determinar si el segundo parámetro que se le pasa a la función es θ_i o d_i.

  Por eso es importante recordar incluir la quinta columna (σ_i) en `dh`. En el caso de que dicha columna no aparezca se asume que la articulación es de rotación.

Algoritmo

La fórmula que se emplea para calcular la matriz de transformación es:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23} $$

y representa el sistema de referencia del enlace $\{i\}$ expresado en las coordenadas del sistema $\{i\text{-}1\}$.

Ejemplos

En Ollero [2], en el Ejemplo 4.2 se calculan las matrices de transformación empleando la sintaxis "`linktrans(dh,q)`". El listado es el siguiente:

```
syms t1 d2 t3 real    % Variables articulares θ1, d2 y θ3
l2=syms l2 real       % Parámetro l2
T1=linktrans([0 0 125 0 0],t1)   % Matriz 0T1

T2=linktrans([pi/2 0 0 426 1],d2)% Matriz 1T2. Hemos dado a σi un
                                   valor distinto de cero (1)
                                   porque la 2ª articulación es
                                   prismática.

T3=linktrans ([0 0 876 l2 0],t3) % Matriz 2T3
```

Para que queden claras las dos sintaxis posibles para la función `linktrans` se va a solucionar el mismo problema empleando ahora la otra sintaxis:

```
syms t1 d2 t3 real       % Variables articulares θ1, d2 y θ3
l2=syms l2 real          % Parámetro l2
T1=linktrans(0,0,t1,0)   % Matriz 0T1

T2=linktrans(pi/2,0,0,d2)% Matriz 1T2.

T3=linktrans (0,0,t3,l2) % Matriz 2T3
```

Ver también

`dh`, `fkine`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Craig, J.J., *Introduction to robotics*. Addison Wesley, segunda ed., 1989.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

numcols

Propósito

Calcular el número de columnas de una matriz

Sintaxis

```
n=numcols(A)
```

Descripción

La función `numcols` devuelve el número de columnas de la matriz que se le pasa como parámetro.

Ejemplos

Esta función se utiliza en el cuerpo de otras funciones.

Ver también

`numrows`

numrows

Propósito

Calcular el número de filas de una matriz

Sintaxis

```
m=numrows(A)
```

Descripción

La función `numrows` devuelve el número de filas de la matriz que se le pasa como parámetro.

Ejemplos

Esta función se utiliza en el cuerpo de otras funciones.

Ver también

`numcols`

plotbot

Propósito

Representar gráficamente el manipulador

Sintaxis

```
plotbot(dh,q)
plotbot(dh,q,opt)
```

Descripción

Esta función construye una representación gráfica del robot, a partir de los parámetros cinemáticos contenidos en `dh` y de los valores de las variables articulares (`q`) que se le pasen.

Se trata de una representación gráfica simple en la que cada enlace se modela como un paralelepípedo de color cián. El sistema de coordenadas asociado al efector final se representa en color verde, mientras que los sistemas de coordenadas asociados a cada una de las articulaciones intermedias se representan en rojo.

Si el parámetro `q` es una matriz que representa una trayectoria en el espacio de las variables articulares (es decir, que contiene en cada fila el conjunto de variables articulares del robot), entonces el resultado es una animación del robot siguiendo dicha trayectoria.

El parámetro `opt` será una cadena en la que se especificarán las opciones de representación que se deseen de entre las siguientes:

- `d` Establece que la representación se muestre en 2D (sobre el plano XY).
- `f` Dibujar los cuadros de referencia asociados a cada articulación.
- `l` No borrar el robot en cada paso si se trata de una animación.
- `r` Repetir la animación 50 veces.
- `w` Dibujar el sistema de referencia asociado al efector final.

Si no se usa el parámetro `opt`, se hace la representación convencional.

Ejemplos

En el ejemplo de la función `ikine` se hace uso de `plotbot` en la forma:

```
plotbot(dh,q,'dfw')
```

para representar una animación del manipulador siguiendo una trayectoria contenida en `q`.

Sería interesante que el lector introdujera otras opciones y comprobara los resultados. Si se quisiera por ejemplo, que además de usar la representación 2D, no se borrara el robot en cada paso, entonces se debería escribir:

```
plotbot(dh,q,'ldfw')
```

obteniéndose algo parecido a lo que se observa en la Figura 3.2

Una representación esquemática del Puma 560 en 3D se puede observar en la Figura 3.3.

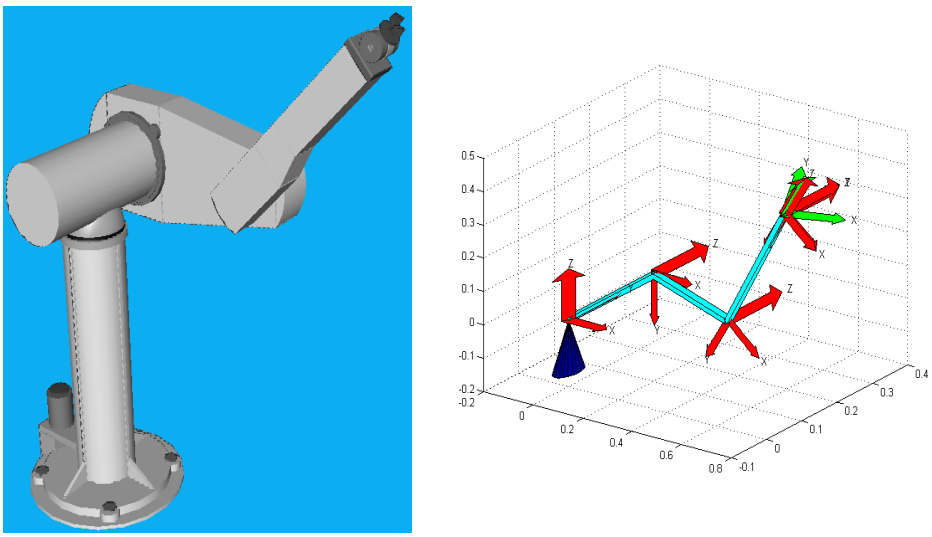


Figura 3.3: Representación simple del Puma 560 mediante el uso de `plotbot` (parte derecha de la figura).

Ver también

`dh`, `fkine`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

tr2diff

Propósito

Convertir una transformación en un vector de movimiento diferencial y calcular la diferencia entre dos transformaciones.

Sintaxis

```
D=tr2diff(T)
D=tr2diff(T1,T2)
```

Descripción

Se tendrá una descripción distinta para cada una de las sintaxis:

- `D=tr2diff(T)`

  Devuelve un vector de movimiento diferencial `D` (ver ecuación (3.4)) que contiene los seis elementos de movimiento diferencial (tres de traslación y tres de rotación) que se pueden obtener de la matriz `T`. La matriz `T` se considera que responde a la forma:

$$\begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & \delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.24}$$

  es decir, que se trata de una matriz de transformación de traslación y rotación diferenciales (ver ecuación (3.5)).

  Los elementos del vector `D` correspondientes a la traslación son extraídos directamente de `T`. Los elementos correspondientes a la rotación en torno a los ejes $\hat{X}$, $\hat{Y}$ y $\hat{Z}$ se obtienen como la media de los dos valores que aparecen en la matriz para cada uno de esos ejes.

- `D=tr2diff(T1,T2)`

  En este caso `tr2diff` devuelve un vector de seis elementos que representa el movimiento diferencial que corresponde al desplazamiento de `T1` a `T2`, es decir a la matriz `T2-T1`. La fórmula para `D` es:

$$\begin{bmatrix} p_2 - p_1 \\ \frac{1}{2}(n_1 \times n_2 + o_1 \times o_2 + a_1 \times a_2) \end{bmatrix} \tag{3.25}$$

donde n, o, a y p son las columnas de las matrices de transformación:

$$T1 = \begin{bmatrix} n_1 & o_1 & a_1 & p_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T2 = \begin{bmatrix} n_2 & o_2 & a_2 & p_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

Ejemplos

La función `tr2diff` se usa en el cuerpo de la función `ikine` por ejemplo.

Ver también

`diff2tr`, `ikine`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Paul, R.P., *Robot manipulators: Mathematics, Programming, and Control*. Cambridge, Massachusetts: MIT Press, 1981.

tr2jac

Propósito

Calcular la matriz (que se podría llamar Jacobiana) que permite pasar un vector de velocidades cartesianas de un cuadro de referencia a otro distinto.

Sintaxis

```
J=tr2jac(T)
```

Descripción

Esta función devuelve una matriz Jacobiana de dimensiones 6x6, que permite expresar en un sistema de referencia {B} un vector de velocidades cartesianas expresado en un sistema {A}. Para ello simplemente hay que pasarle como parámetro la matriz de transformación `T` que expresa el cuadro {B} en el {A} ($^{A}T_{B}$).

Es decir, esta función calcula una matriz $^{B}J_{A}$ tal que:

$$^{B}\upsilon = {}^{B}J_{A} \cdot {}^{A}\upsilon \tag{3.27}$$

Ver también

`diff2tr`, `tr2diff`

Referencias

Corke, P.I., *A robotics toolbox for MATLAB*, IEEE Robotics and Automation Magazine, Vol.3, núm. 1, pp. 24-32, 1996.

Paul, R.P., *Robot manipulators: Mathematics, Programming, and Control*. Cambridge, Massachusetts: MIT Press, 1981.

velprop

Propósito

Calcular el vector de velocidades en el espacio cartesiano expresado en el sistema de coordenadas del cuadro $\{n\}$.

Sintaxis

```
z=velprop(dh,q,qd,v0,w0)
```

Descripción

Esta función calcula el vector ${}^{n}\upsilon = \begin{bmatrix} {}^{n}\nu \\ {}^{n}\omega \end{bmatrix}$, que contiene las velocidades lineales y angulares del extremo del manipulador expresadas en el cuadro $\{n\}$, donde n representa el número de enlaces del manipulador.

Para ello es necesario pasarle una serie de parámetros:

Parámetro	Significado
`q`	Vector de variables articulares
`qd`	Vector de velocidades de las variables articulares
`v0`	Velocidad lineal de la base
`w0`	Velocidad angular de la base

Algoritmo

Se emplea el método de propagación de velocidades.

Ejemplos

Ver Ejemplos 4.5, 4.6 y 4.7 en Ollero [2].

Ver también

`dh`, `fkine`

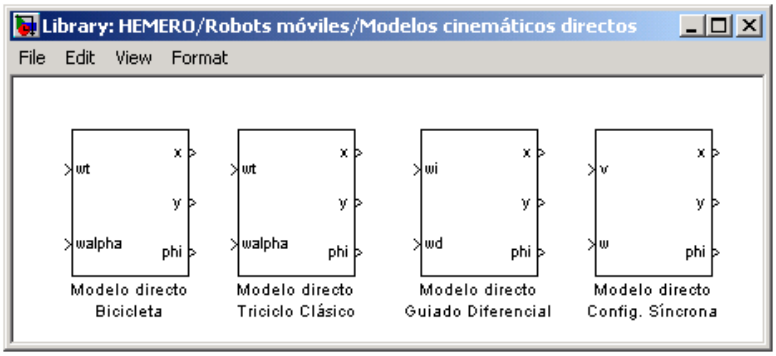
Referencias

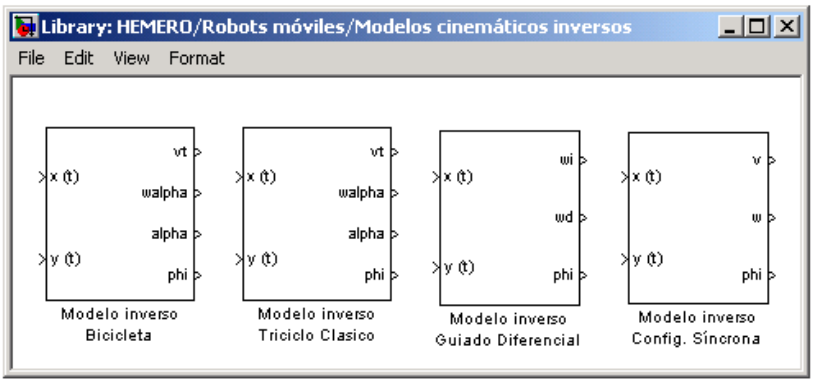
Craig, J.J., *Introduction to robotics*. Addison Wesley, segunda ed., 1989.

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelos cinemáticos de robots móviles

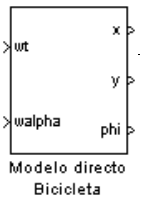
Como se comentó al comienzo de este capítulo, para la simulación de los modelos cinemáticos de ciertas configuraciones clásicas de robots móviles, se dispone de una serie de bloques Simulink que se detallan a continuación:





También hay funciones para la representación esquemática de robots móviles.

Modelo directo bicicleta



Propósito

Calcular las coordenadas y orientación de un vehículo caracterizado por el modelo de la bicicleta a partir de las señales de control correspondientes.

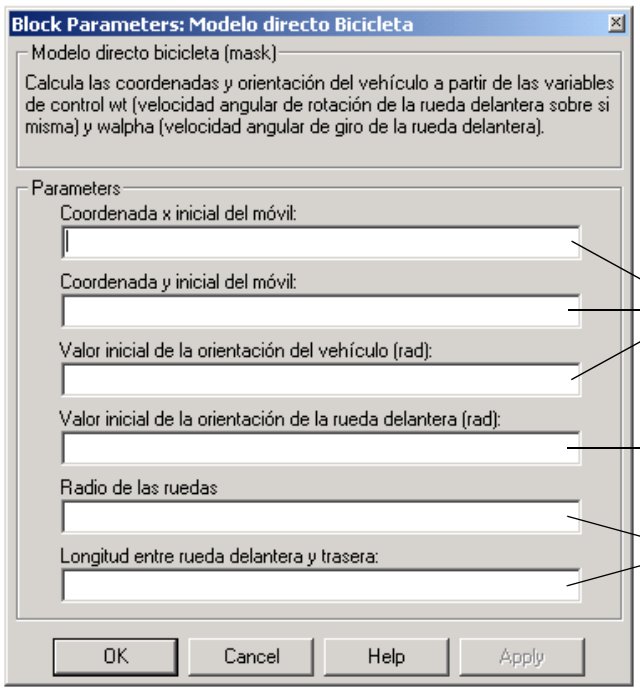
Descripción

Este bloque calcula las coordenadas (x, y) y la orientación ϕ de un vehículo caracterizado por el modelo de la bicicleta a partir de las señales de control ω_l y ω_α. Las ecuaciones del modelo son:

$$\begin{bmatrix} x' \\ y' \\ \phi' \\ \alpha' \end{bmatrix} = \begin{bmatrix} -c\sin\phi\cos\alpha & 0 \\ c\cos\phi\cos\alpha & 0 \\ (c\sin\alpha)/l & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_\alpha \end{bmatrix} \tag{3.28}$$

donde c es el radio de las ruedas y l es la distancia entre la rueda delantera y la trasera.

Parámetros



Valores de las coordenadas y orientación iniciales del vehículo

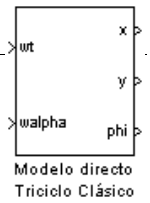
Valor inicial del ángulo α

Valores del radio (c) y de la longitud (l)

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo directo triciclo clásico



Propósito

Calcular las coordenadas y orientación de un vehículo tipo triciclo a partir de las señales de control correspondientes.

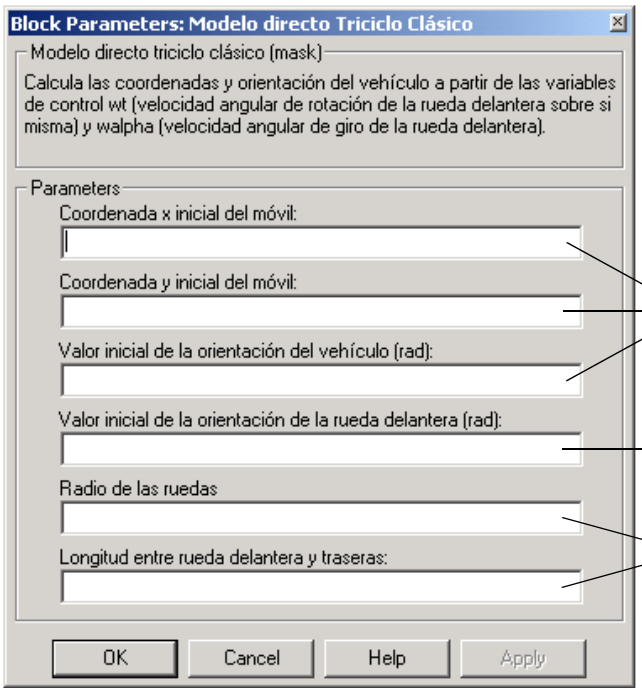
Descripción

Este bloque calcula las coordenadas (x, y) y la orientación ϕ de un vehículo tipo triciclo a partir de las señales de control ω_l y ω_α. Las ecuaciones del modelo son:

$$\begin{bmatrix} x' \\ y' \\ \phi' \\ \alpha' \end{bmatrix} = \begin{bmatrix} -c\sin\phi\cos\alpha & 0 \\ c\cos\phi\cos\alpha & 0 \\ (c\sin\alpha)/l & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_\alpha \end{bmatrix} \tag{3.29}$$

donde c es el radio de las ruedas y l es la distancia entre la rueda delantera y las traseras.

Parámetros



Valores de las coordenadas y orientación iniciales del vehículo

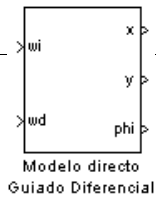
Valor inicial del ángulo α

Valores del radio (c) y de la longitud (l)

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo directo guiado diferencial



Propósito

Calcular las coordenadas y orientación de un vehículo con guiado diferencial a partir de las señales de control correspondientes.

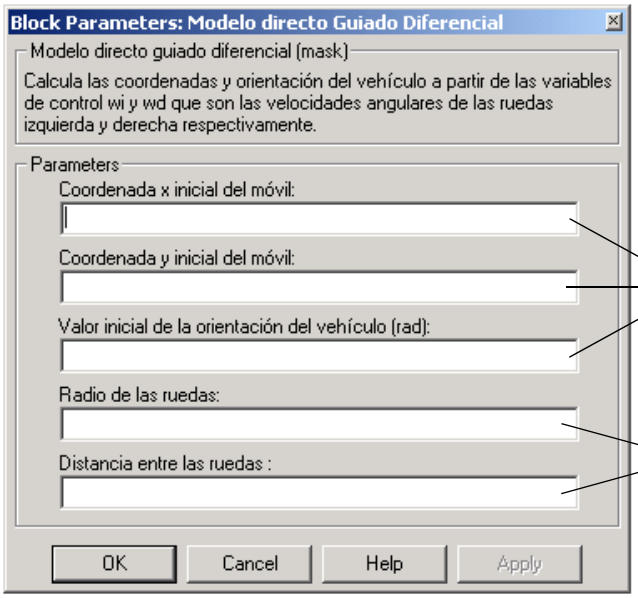
Descripción

Este bloque calcula las coordenadas (x, y) y la orientación ϕ de un vehículo con guiado diferencial a partir de las señales de control ω_i y ω_d. Las ecuaciones del modelo de este tipo de vehículos son:

$$\begin{bmatrix} x' \\ y' \\ \phi' \end{bmatrix} = \begin{bmatrix} -(c\sin\phi)/2 & -(c\sin\phi)/2 \\ (c\cos\phi)/2 & (c\cos\phi)/2 \\ -c/b & c/b \end{bmatrix} \begin{bmatrix} \omega_i \\ \omega_d \end{bmatrix} \tag{3.30}$$

donde c es el radio de las ruedas y b es la distancia entre las ruedas.

Parámetros



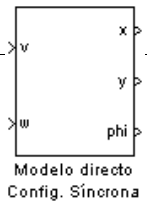
Valores de las coordenadas y orientación iniciales del vehículo

Valores del radio de las ruedas (c) y de la longitud entre ruedas (b)

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo directo config. síncrona



Propósito

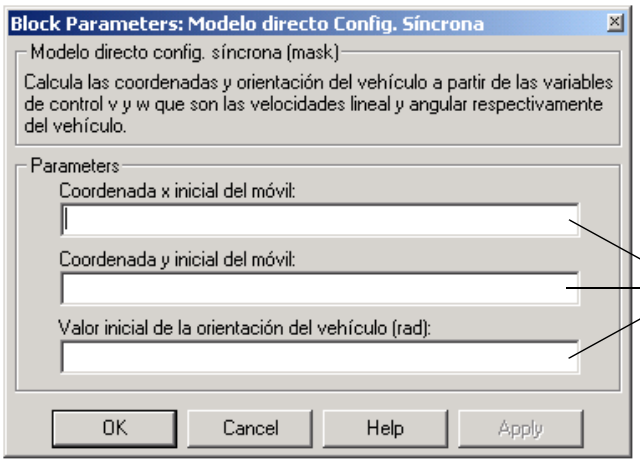
Calcular las coordenadas y orientación de un vehículo caracterizado por una configuración síncrona a partir de las señales de control correspondientes.

Descripción

Este bloque calcula las coordenadas (x, y) y la orientación ϕ de un vehículo caracterizado por una configuración síncrona a partir de las señales de control v y ω. Las ecuaciones del modelo son:

$$\begin{bmatrix} x' \\ y' \\ \phi' \end{bmatrix} = \begin{bmatrix} -\sin\phi & 0 \\ \cos\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.31}$$

Parámetros

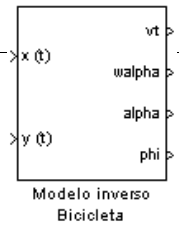


Valores de las coordenadas y orientación iniciales del vehículo

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo inverso bicicleta



Propósito

Calcular a partir de la trayectoria en el tiempo, las correspondientes variables de control para un vehículo caracterizado por el modelo de la bicicleta.

Descripción

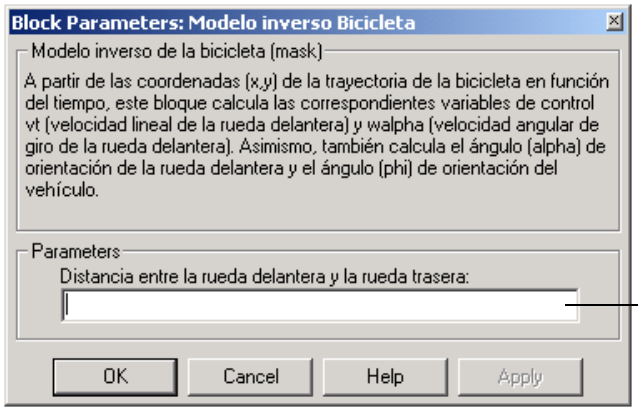
Este bloque calcula las variables de control ω_l y ω_α para un vehículo caracterizado por el modelo de la bicicleta a partir de la trayectoria en el tiempo $(x(t), y(t))$. Asimismo, también devuelve el ángulo de orientación (α) de la rueda delantera y la orientación del vehículo (ϕ). Las ecuaciones del modelo son:

$$\begin{bmatrix} v_l \\ \omega_\alpha \end{bmatrix} = \begin{bmatrix} \dfrac{-l^2 \sin\phi\cos\alpha}{(l\cos\alpha)^2 + (\sin\alpha)^2} & \dfrac{l^2\cos\phi\cos\alpha}{(l\cos\alpha)^2 + (\sin\alpha)^2} & \dfrac{l\sin\alpha}{(l\cos\alpha)^2 + (\sin\alpha)^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ \phi' \\ \alpha' \end{bmatrix} \quad (3.32)$$

$$\phi = \operatorname{atan}\left(-\frac{x'}{y'}\right)$$

donde l es la distancia entre la rueda delantera y la trasera.

Parámetros

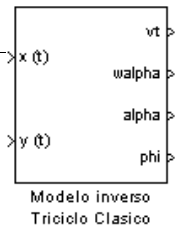


Valor de la longitud l

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo inverso triciclo clásico



Propósito

Calcular a partir de la trayectoria en el tiempo, las correspondientes variables de control para un vehículo tipo triciclo.

Descripción

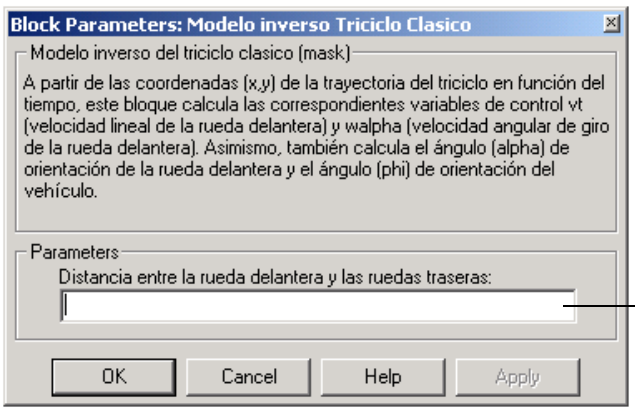
Este bloque calcula las variables de control ω_l y ω_α para un vehículo tipo triciclo a partir de la trayectoria en el tiempo $(x(t), y(t))$. Asimismo, también devuelve el ángulo de orientación (α) de la rueda delantera y la orientación del vehículo (ϕ). Las ecuaciones del modelo son:

$$\begin{bmatrix} v_l \\ \omega_\alpha \end{bmatrix} = \begin{bmatrix} \dfrac{-l^2 \sin\phi\cos\alpha}{(l\cos\alpha)^2 + (\sin\alpha)^2} & \dfrac{l^2\cos\phi\cos\alpha}{(l\cos\alpha)^2 + (\sin\alpha)^2} & \dfrac{l\sin\alpha}{(l\cos\alpha)^2 + (\sin\alpha)^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ \phi' \\ \alpha' \end{bmatrix} \quad (3.33)$$

$$\phi = \operatorname{atan}\left(-\frac{x'}{y'}\right)$$

donde l es la distancia entre la rueda delantera y las traseras.

Parámetros

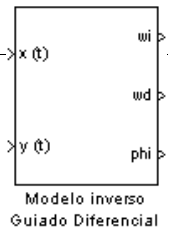


Valor de la longitud l

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo inverso guiado diferencial



Propósito

Calcular a partir de la trayectoria en el tiempo, las correspondientes variables de control para un vehículo con guiado diferencial.

Descripción

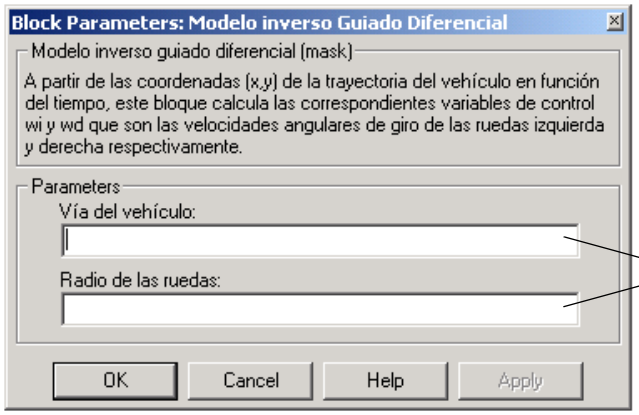
Este bloque calcula las variables de control ω_i y ω_d para un vehículo con guiado diferencial a partir de la trayectoria en el tiempo $(x(t), y(t))$. Asimismo, también devuelve la orientación del vehículo (ϕ). Las ecuaciones del modelo son:

$$\begin{bmatrix}\omega_i \\ \omega_d\end{bmatrix} = \begin{bmatrix}-\frac{\sin\phi}{c} & \frac{\cos\phi}{c} & -\frac{b}{2c} \\ -\frac{\sin\phi}{c} & \frac{\cos\phi}{c} & \frac{b}{2c}\end{bmatrix}\begin{bmatrix}x' \\ y' \\ \phi'\end{bmatrix} \qquad (3.34)$$

$$\phi = \operatorname{atan}\left(-\frac{x'}{y'}\right)$$

donde c es el radio de las ruedas y b es la distancia entre las ruedas.

Parámetros

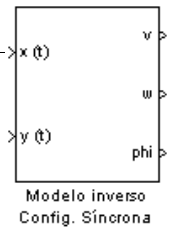


Valores de la longitud entre ruedas (*b*) y del radio de las ruedas (*c*)

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Modelo inverso config. síncrona



Propósito

Calcular a partir de la trayectoria en el tiempo, las correspondientes variables de control para un vehículo con configuración síncrona.

Descripción

Este bloque calcula las variables de control v y ω para un vehículo con configuración síncrona a partir de la trayectoria en el tiempo $(x(t), y(t))$. Asimismo, también devuelve la orientación del vehículo (ϕ). Las ecuaciones del modelo son:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ \phi' \end{bmatrix} \qquad (3.35)$$

$$\phi = \operatorname{atan}\left(-\frac{x'}{y'}\right)$$

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

trirep

Propósito

Representar gráficamente el modelo del triciclo clásico.

Sintaxis

```
trirep(x,y,phi,alpha,color,b,l,c,np,opt)
```

Descripción

Esta función se encarga de generar una representación gráfica simple del triciclo clásico a partir de las coordenadas del centro de guiado (`x`, `y`), de la orientación del vehículo (`phi`) y de la orientación de la rueda delantera del vehículo (`alpha`). Los valores de `x`, `y`, `phi` y `alpha` pueden ser vectores, en cuyo caso se representarán tantos puntos de la trayectoria como indique el parámetro `np`. En este caso, es posible especificar que aparezcan simultáneamente todas esas representaciones en pantalla dando un valor cualquiera al parámetro `opt`. Si no se introduce dicho parámetro, las representaciones aparecen y desaparecen sucesivamente, y al final solamente queda la última en pantalla.

Las dimensiones de la representación gráfica vienen determinadas por los siguientes parámetros:

- `b`: separación entre las dos ruedas traseras.
- `l`: separación entre la rueda delantera y las ruedas traseras.
- `c`: radio de las ruedas.

El color de la representación se especifica mediante el parámetro `color`.

Referencias

Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

Ejemplos

Ejemplo H.3.2 (archivo `ejh32.m`): Ver Ejemplo 4.1 en Ollero [2].

Ejemplo H.3.3 (archivo `ejh33.m`): Ver Ejemplo 4.2 en Ollero [2].

Ejemplo H.3.4 (archivo `ejh34.m`): Ver Ejemplo 4.3 en Ollero [2].

Ejemplo H.3.5 (archivo `ejh35.m`): Ver Ejemplo 4.5 en Ollero [2].

Ejemplo H.3.6 (archivo `ejh36.m`): Ver Ejemplo 4.6 en Ollero [2].

Ejemplo H.3.7 (archivo `ejh37.m`): Ver Ejemplo 4.7 en Ollero [2].

Referencias

[1] Craig, J.J., *Introduction to robotics*. Addison Wesley, segunda ed., 1989.

[2] Ollero, A., *Robótica: Manipuladores y robots móviles*, Marcombo-Boixareu editores, 2001.

[3] Paul, R.P., *Robot manipulators: Mathematics, Programming, and Control*. Cambridge, Massachusetts: MIT Press, 1981.

[4] Paul, R.P., Shimano y Mayer, *Differential kinematic control equations for simple manipulators*, IEEE SMC 11(6), pp. 456-460, 1981.