Tema 2. Robótica Industrial

Resumen de instrucciones de la Toolbox Robotics Vision and Control (Peter Corke). Versión de referencia 10.3.1

Para ver más opciones utilice `help`

Transformación de coordenadas

Transformaciones básicas:

- Matrices 3x3 de rotación respect al eje indicado (argumento en radianes):

  `R=rotx(), R=roty(), R=rotz();`

- Dibujo de los ejes de la matriz de rotación:

  `trplot2(R);`

- Ángulos de Euler (ϕ, ϑ, Ψ): R=rotz(ϕ)*roty(ϑ)*rotz(Ψ)

  `R=eul2r(`ϕ, θ, Ψ`);`

  `[`ϕ, θ, Ψ`]=tr2eul(R);` función inversa. Los ángulos de euler que representan la matriz de rotación siempre son calculados con $\vartheta>0$.

- Ángulos Cardan: roll (ϑr), pitch (ϑp), yaw (ϑy). Son giros locales (uvw):

  XYZ, en robots móviles (X, eje de avance): roll -> alabeo, pitch -> cabeceo, yaw -> guiñada.

  `R=rpy2r(`$\theta r, \theta p, \theta y$`,'vehicle');`

  ZYX en robots manipuladores (Z, eje de avance).

  `R=rpy2r(`$\theta r, \theta p, \theta y$`,'arm');`

  `[`$\theta r, \theta p, \theta y$`]=tr2rpy(R,type)` -> función inversa. Obtención de los ángulos Cardan que dan lugar a la transformación R (puede ser 3x3 o 4x4). Type indicará si corresponde a manipulador o móvil.

Coordenadas homogéneas (4x4)

- Matriz de traslación (una distancia dx,dy,dz)

  `T=transl(dx,dy,dz);`

- Matriz de rotación:

  `R=trotx(), R=troty(), R=trotz();` argumento en radianes.

- Dibujo de los ejes de la matriz de transformación:

  `trplot();` argumento T o R.

Utilizando el objeto de transformación en coordenadas homogéneas: SE3

- Creación del objeto (Oh):

  Oh=SE3.eul rotación definida por ángulos de Euler

  Oh=SE3.rpy rotación definida por ángulos Cardan

  Oh=SE3.Rx rotación en el eje x

  Oh=SE3.Ry rotación en el eje y

  Oh=SE3.Rz rotación en el eje z

  Oh=SE3(dx,dy,dz) traslación pura

- Funciones:

  Oh.inv -> calcula la matriz de transformación inversa

  Oh.R -> devuelve la matriz de rotación

  Oh.t -> devuelve el vector posición (vector columna)

  Oh.T -> devuelve la matrix de coordenadas homogeneas

  Oh.toeul -> devuelve los ángulos de Euler

  Oh.torpy -> devuelve los ángulos roll, picth y yaw

  Oh.n -> devuelve vector aproximación (X)

  Oh.o -> devuelve vector orientación (Y, Tambien denominado S, deslizamiento)

  Oh.a -> devuelve vector aproximación (Z)

  Oh.plot -> dibuja el Sistema de ejes.

Manipuladores

Definición de un manipulador (Denavit-Hartenberg estándar)

- **Articulaciones (Links):**

  `L1= Prismatic('alpha',-pi/2, 'qlim',[1,3]);`

  `L2= Revolute('a',2,'d',0.5);` los valores del la matriz DH que no se indican son cero.

  `L1.A(v)` -> Obtiene la matriz de transformación para el Link L1 para el valor de la variable v (matriz ${}^{i}A_{i-1}$)

- **Construcción del robot:**

  `Rob=SerialLink([L1, L2], 'name', 'Mi_robot');`

Funciones del robot:

`Rob.teach` -> dibuja robot y se puede mover

`Rob.plot(q)` -> dibuja el robot en las posiciones dada por las variables de la articulación en q

`Rob.edit` -> permite modificar la matriz DH

`T=Rob.fkine(q)` -> calcula el objeto SE3 que representa al efector final.

Si tenemos acoplado una herramienta en el efetor final se puede poner donde está respect a su centro (el ultimo eje del robot, XnYnZn). Ejemplo:

`Rob.tool = SE3(0,0,0.2)` -> efector fina a 20 cm del eje Zn

`Q=Rob.ikine(T)` -> calcula la cinemática inversa del Rob.

Si es subactuado, se puede indicar qué valores pueden ser ignorados en la solución. Ejemplo:

`q=Rob.iKine(T, 'mask', [1 1 1 0 0 1]);` -> interesa la posición del efector final y que esté alineada con el eje Zn.

Movimiento del robot:

`[q,qd,qdd]=jtraj(q1,q2,t)` -> interpolación de trayectoria entre dos movimientos q1 y q2 (se puede indicar vector de tiempos o valores). Ejemplo:

```
q1=[1,0]; q2=[2,pi/2];  q=jtraj(q1,q2,10);
Rob.plot(q,'delay',0.5,'nobase','loop')
```

También se puede hacer la interpolación entre dos posiciones del efector final:

`[q,qd,qdd]=Rob.jtraj(T1, T2, t);`

Se puede interpolar en el espacio cartesiano:

```
Ts=ctraj(T1,T2,length(t)); qc=Rob.iKine(Ts);
```

Se puede calcular la destreza de movimiento de un manipulador (manipulabilidad, cuanto más alta mejor):

```
Rob.maniplty(qc);
```

Existen modelos de robot predefinidos, que pueden cargarse en Matlab para probar las distintas opciones de la toolbox, utilice `models` para ver cuáles son.