Modelado y Simulación de Robots

Cinemática Inversa

Grado en Ingeniería de Robótica Software

Teoría de la Señal y las Comunicaciones y Sistemas Telemáticos y Computación

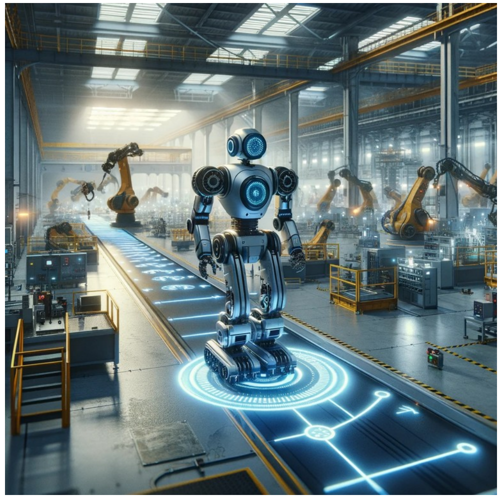
Roberto Calvo Palomino
roberto.calvo@urjc.es

Introducción

- La cinemática es la rama de la mecánica que estudia el movimiento de los cuerpos sin considerar las fuerzas que lo causan.
- Posición, velocidad y aceleración de las partes del robot.
  - <u>Cinemática Directa:</u>
    - Conocemos los ángulos y geometría
    - Calculamos la posición y orientación del elemento final.
  - <u>Cinemática Inversa:</u>
    - Conocemos la posición y orientación del elemento final.
    - Calculamos los ángulos y geometría para alcanzar la orientación y posición del elemento final.
- Permite la planificación de trayectorias y la realización de tareas específicas.

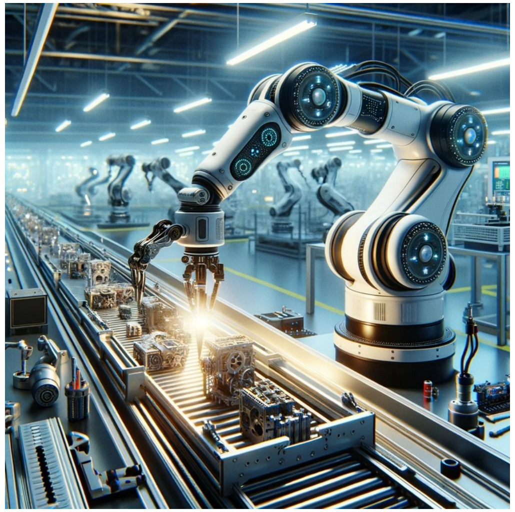
Aplicaciones

Planificación de movimientos

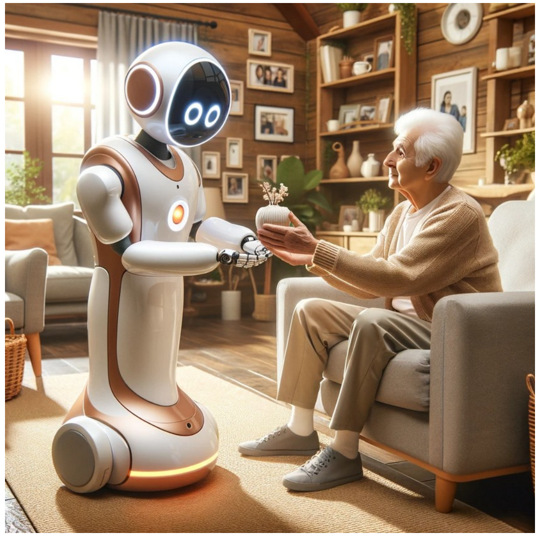


Simulación de Robots

Control de Robots

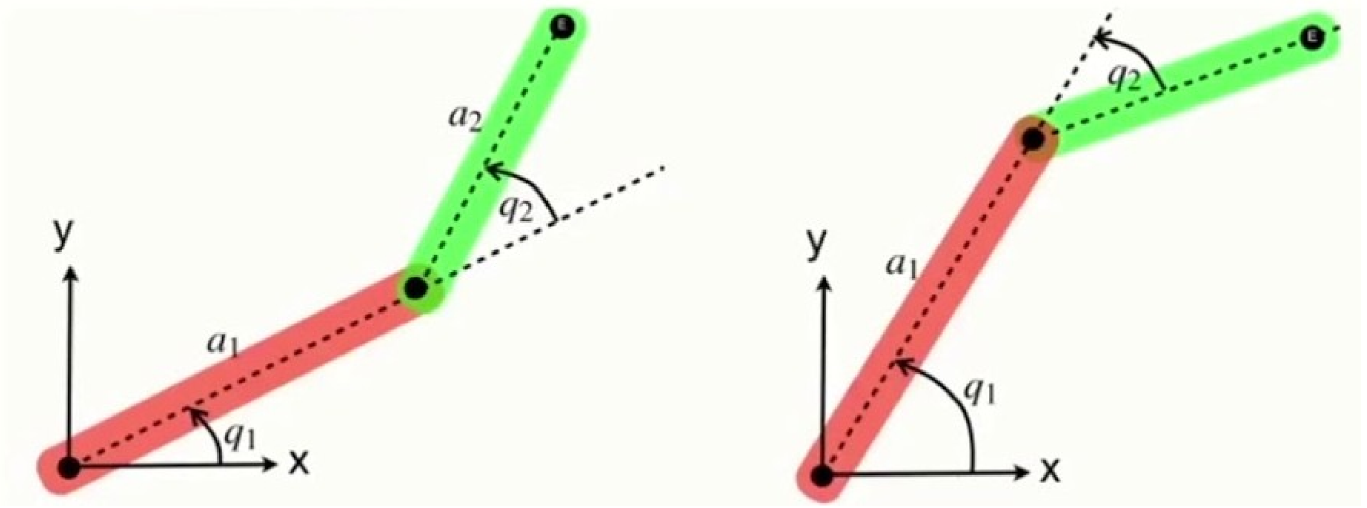


Interacción con el entorno



Introducción

- **Cinemática inversa (IK)** es la técnica que permite determinar el movimiento de una cadena de articulaciones para lograr que un actuador final se ubique en la posición deseada.
- El cálculo de la cinemática inversa se considera un problema complejo, ya que la resolución de ecuaciones no deriva en una única solución.



Cinemática Inversa

- Se espera una solución en tiempo real
- Posibles soluciones, varias políticas:
  - Menos articulaciones en movimiento
  - Más optima en términos de batería
- NO siempre es posible ofrecer una solución (limitaciones físicas de articulaciones)
- Utilizado para articulaciones en:
  - Brazo robótico
  - Bípedos
  - Cuadrúpedos

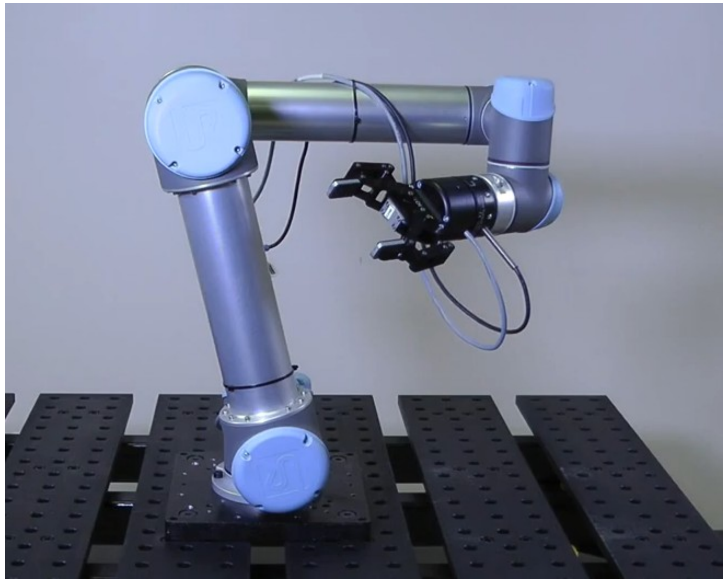
Cinemática Inversa

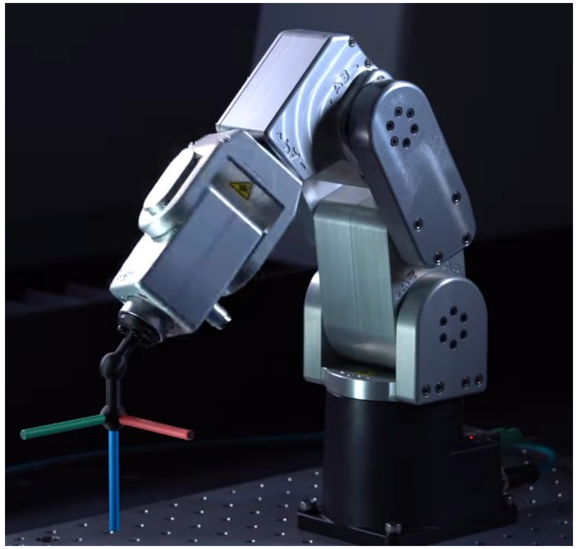
Cinemática Inversa

- El concepto de **singularidad** se utiliza para definir un mal funcionamiento de un sistema robótico
- Singularidades en robótica ocurren porque robots son controlados por matemáticas (muchas ecuaciones matemáticas tienden a infinito).
- La singularidad ocurre cuando hay múltiples soluciones en el sistema cinemático y no se escoge la óptima (asumiendo que hay una)
- Posición incorrecta y aumento drástico de la velocidad de las articulaciones son los efectos más comunes de singularidad.

Cinemática Inversa

- Ejemplos de Singularidades





Matriz Jacobiana

- La matriz Jacobiana se utiliza para representar la relación entre las variables de entrada y las variables de salida en un sistema multivariable
- Normalmente sistemas no lineales.
- Descripción del cambio de las variables de salida con respecto a pequeños cambios en las variables de entrada.

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

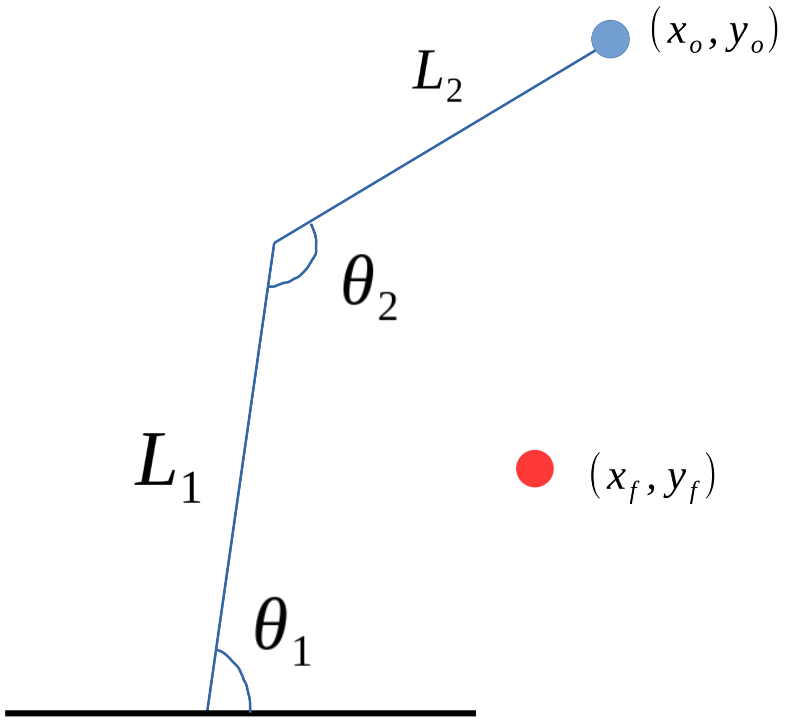
Matriz Jacobiana

- Usos:
  - Relación entre velocidades
  - Aproximaciones lineales
  - Cambio de coordenadas
- Estimación de velocidad / propiedades de fluidos en el espacio.
- Resolver problemas de cinemática inversa.
- Entender deformaciones de objetos cuando se mueven sus articulaciones.

Cinemática Inversa

- Algoritmos iterativos para solucionar el sistema no lineal de ecuaciones. Ofrecen una aproximación.
- Ninguna garantiza evitar singularidades.
  - Jacobian Inverse
    - Converge rápidamente pero es costosa computacionalmente.
  - Pseudoinverse
    - Inestable cerca de la singularidad.
  - Jacobian transpose
    - Velocidad muy alta cerca de la singularidad.
  - Damped least square (utilizada por pybullet).
    - Reduce la velocidad cerca de la singularidad.

Jacobiano Inverso



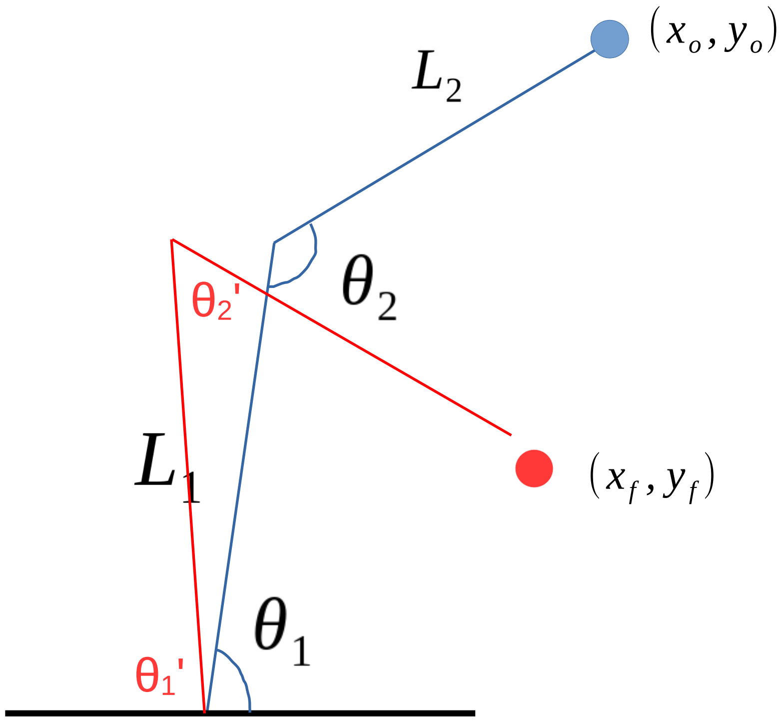
$$x_o = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \qquad \Delta x = x_f - x_o$$
$$y_o = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \qquad \Delta y = y_f - y_o$$

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

Jacobiano Inverso $$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Por tanto, obtenemos los nuevos ángulos: $$\theta_1' = \theta_1 + \Delta\theta_1$$
$$\theta_2' = \theta_2 + \Delta\theta_2$$

Jacobiano Inverso



$$x_o = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \qquad \Delta x = x_f - x_o$$
$$y_o = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \qquad \Delta y = y_f - y_o$$

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

Jacobiano Inverso $\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$

Por tanto, obtenemos los nuevos ángulos:
$$\theta_1' = \theta_1 + \Delta\theta_1$$
$$\theta_2' = \theta_2 + \Delta\theta_2$$

Cinemática Inversa

- Cuadrados Mínimos Amortiguados (Damped Least Squares, DLS)
- Parte del enfoque clásico de cuadrados mínimos para mejorar la estabilidad y la robustez de la solución en condiciones problemáticas, como las singularidades.
- Añade el factor de amortiguamiento (λ) que reduce efectos de las singularidades.

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = J^T \cdot \left(J \cdot J^T + \lambda^2 \cdot I \right)^{-1} \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

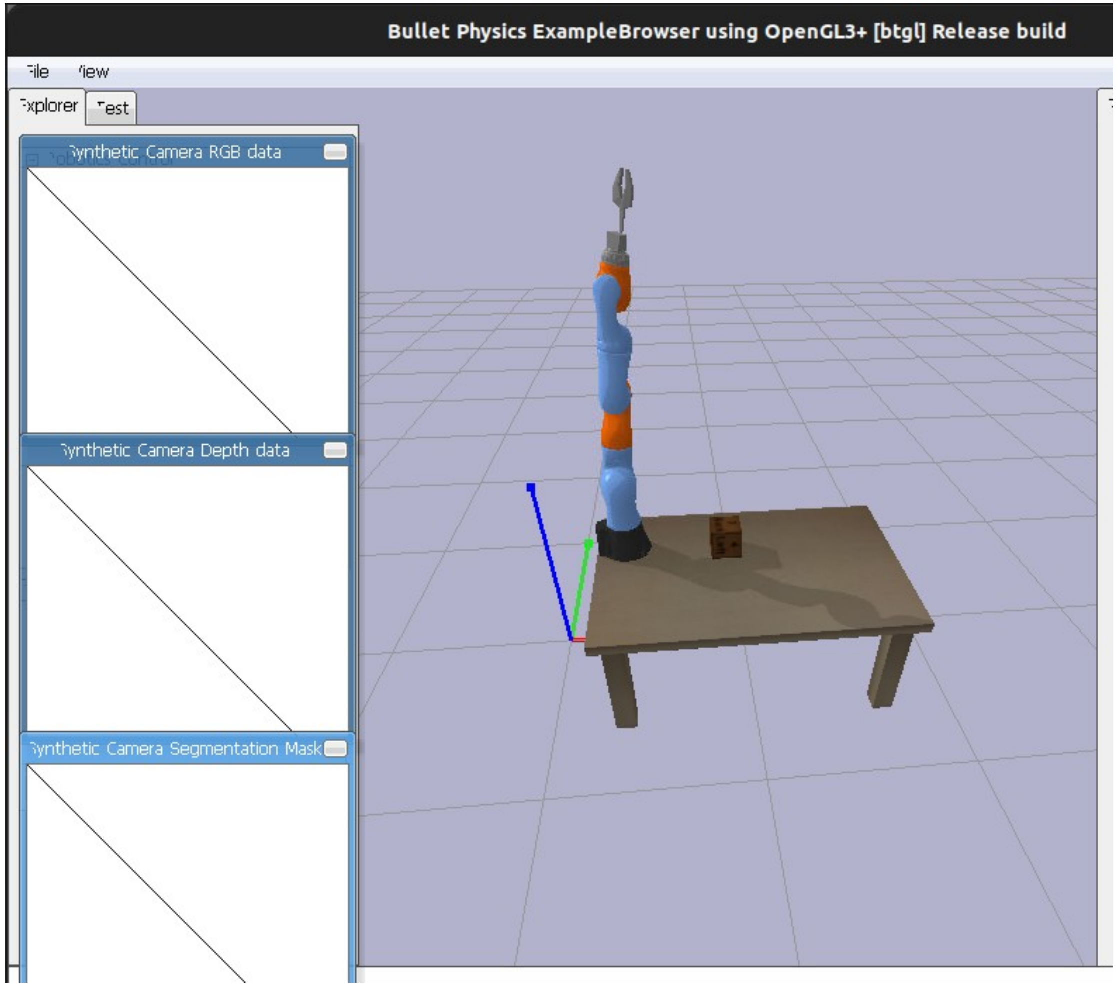
- Ventajas
  - Robustez ante las singularidades
  - Mejora de la estabilidad
  - Flexibilidad

Cinemática Inversa

- Algoritmos basados en aprendizaje por refuerzo pueden aprender a realizar acciones de cinemática inversa sin necesidad de conocer los fundamentos matemáticos.
- Sistemas basados en recompensa.
- Proporcionan flexibilidad, aprendizaje autónomo, adaptabilidad, generazación.
- Algortimos: DQN, PPO, DDPG, SAC, ….

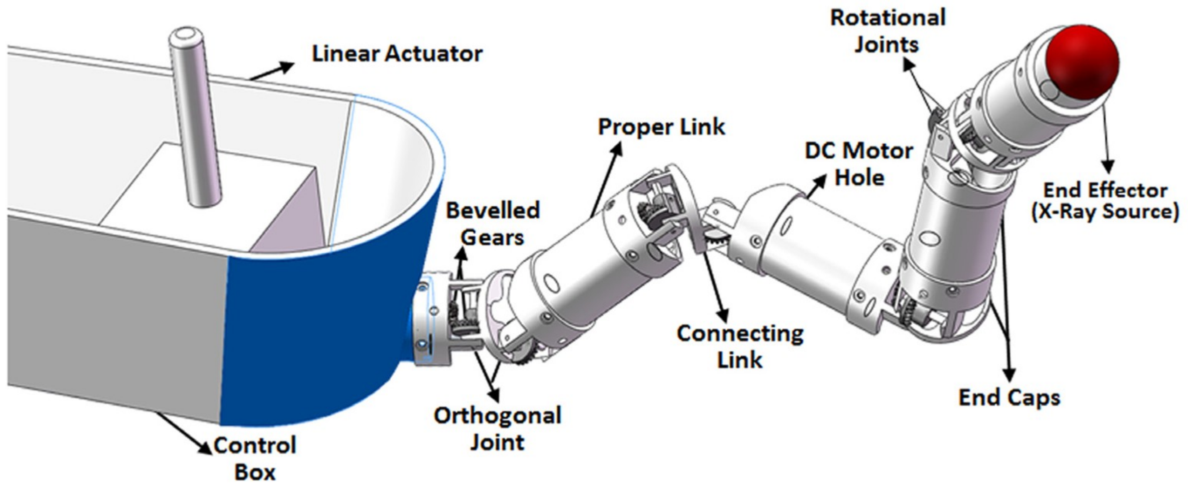
Cinemática Inversa

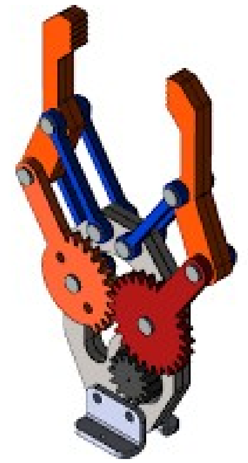
- Ejemplo DRL:



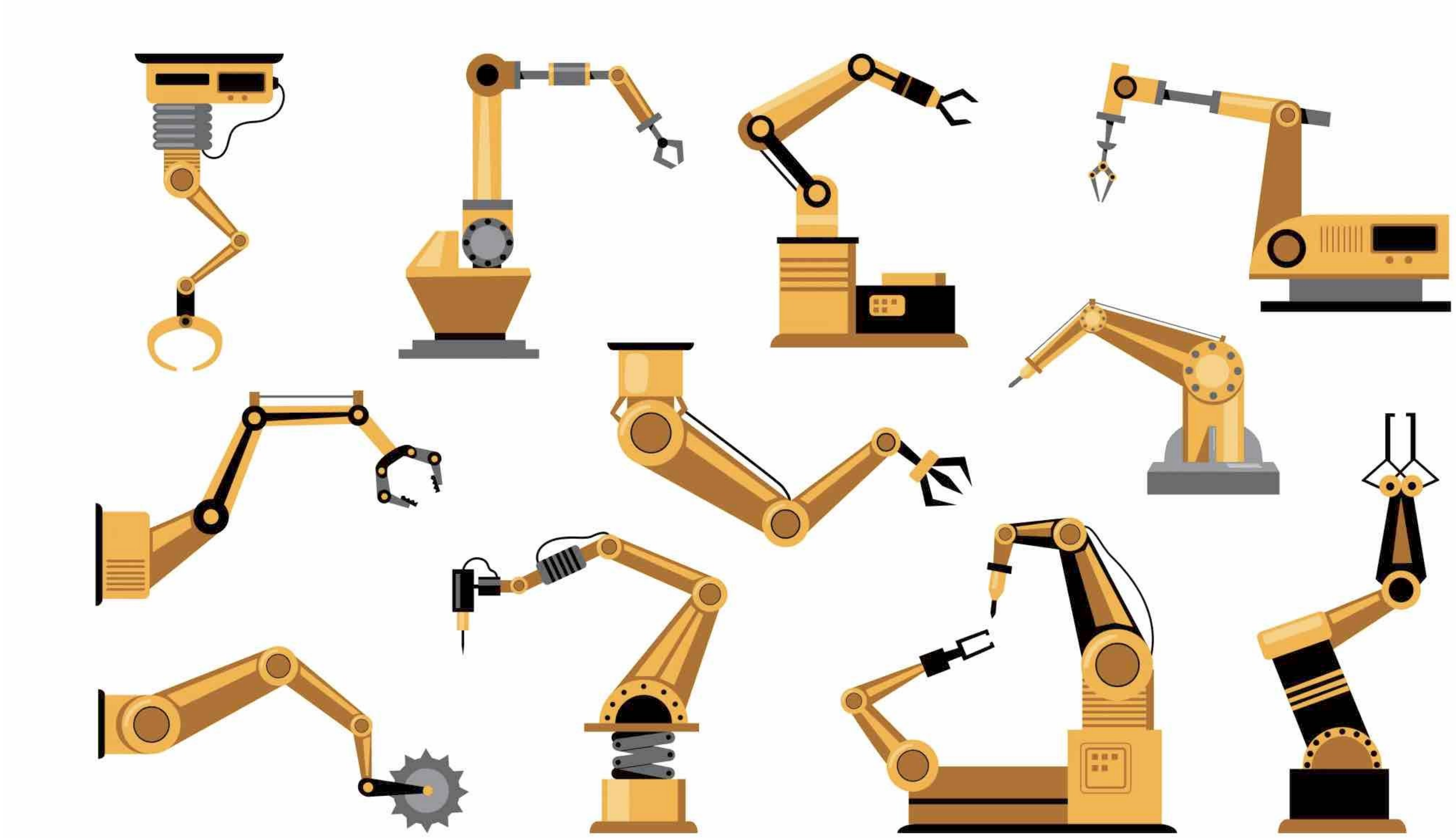
End Effecfor

- El término "end effector" se utiliza para definir la herramienta conectada al final de un brazo mecánico.
- Es el "punto de contacto" entre el robot y los objetos o el entorno de trabajo.
- Dependiendo de la aplicación final, se elegirá el diseño





YouTube Video1 Video2 Video3

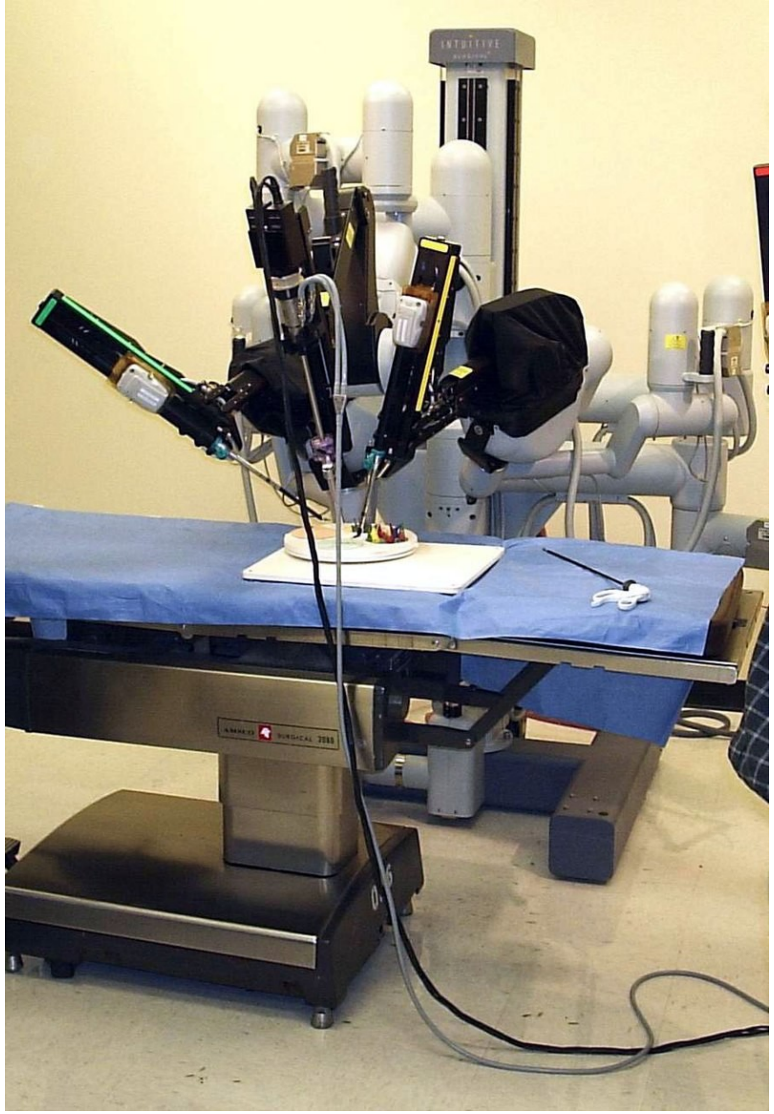


End Effecfor

- En general, un brazo mecánico está compuesto por segmentos unidos mediante articulaciones formando una cadena de articulaciones.
- En general, según avanzamos en la cadena de articulaciones los segmentos deberían ser menores y las articulaciones más restrictivas.
- Las articulaciones cercanas al “end effector” deben proveer máxima movilidad (depende de la aplicación final).

End-Effector

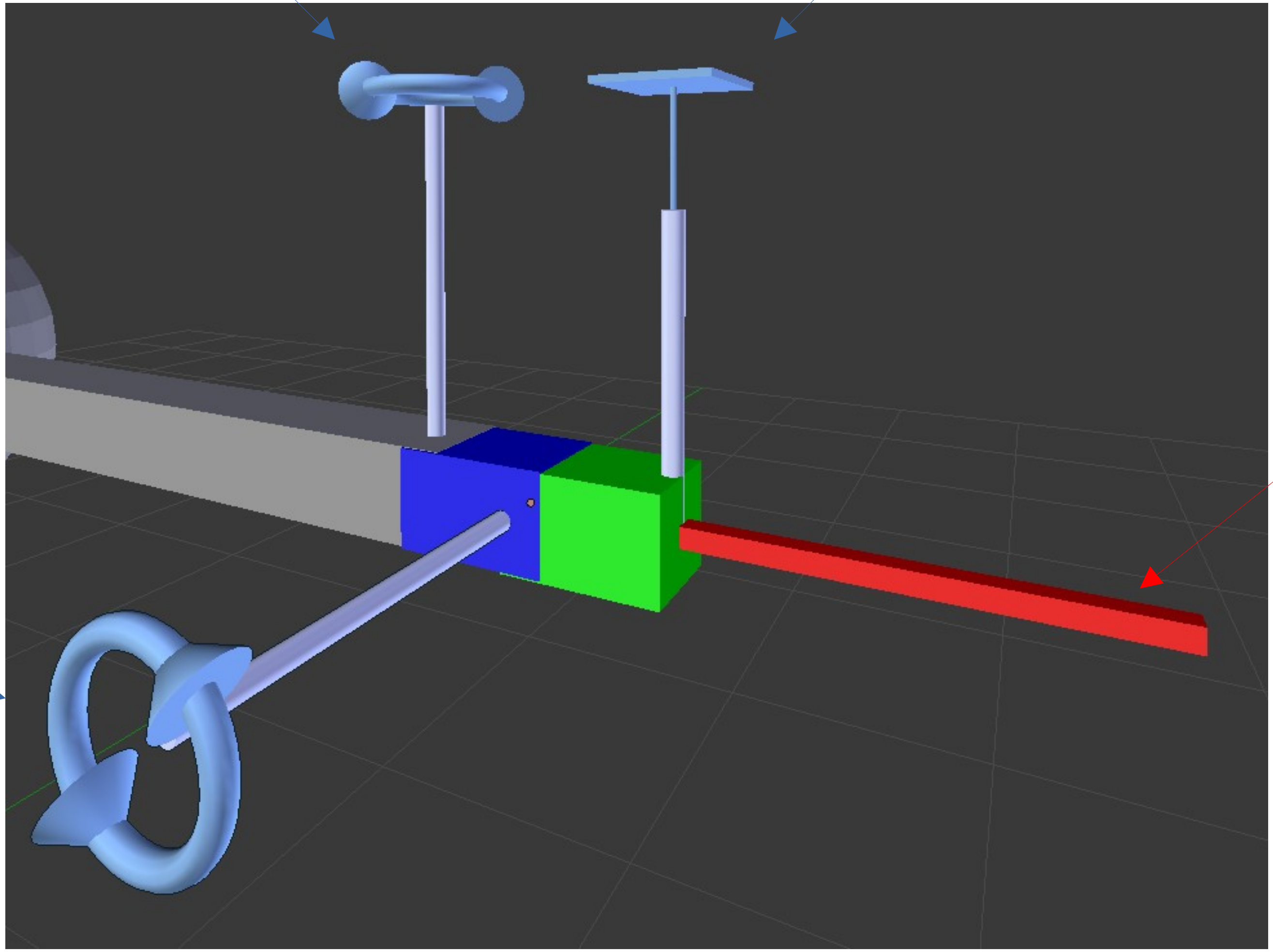
- Uso en Cirugía
- Robot Da Vinci



End Effector

Continuous Joint

Fixed Joint



End Effector

Continuous Joint

Cinemática Inversa
PyBullet

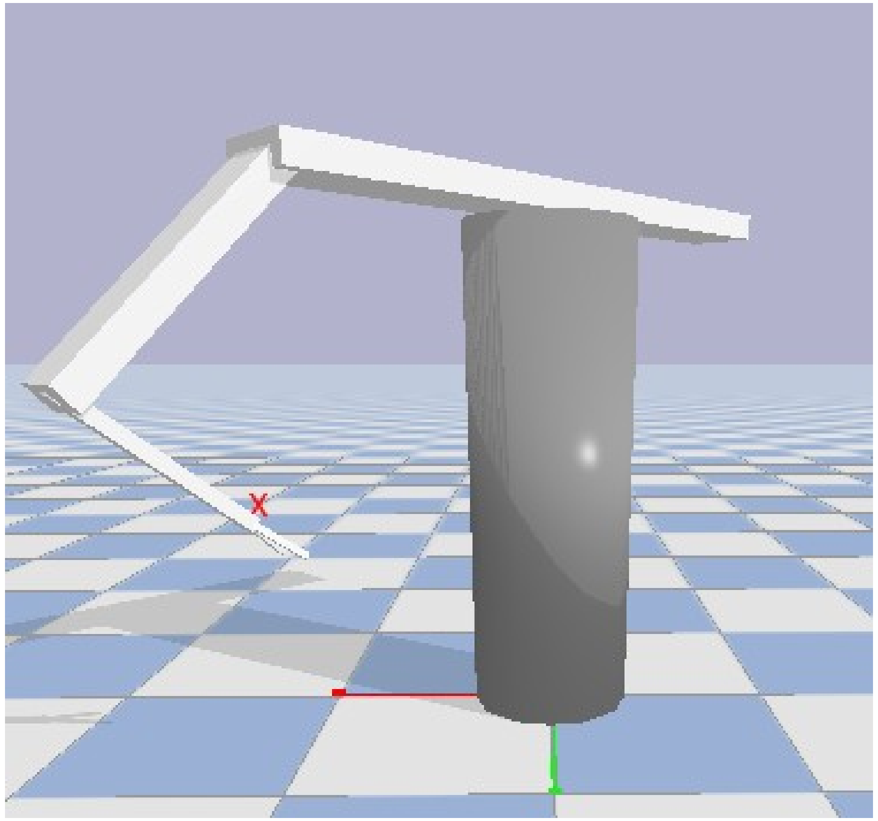
Cinemática Inversa PyBullet

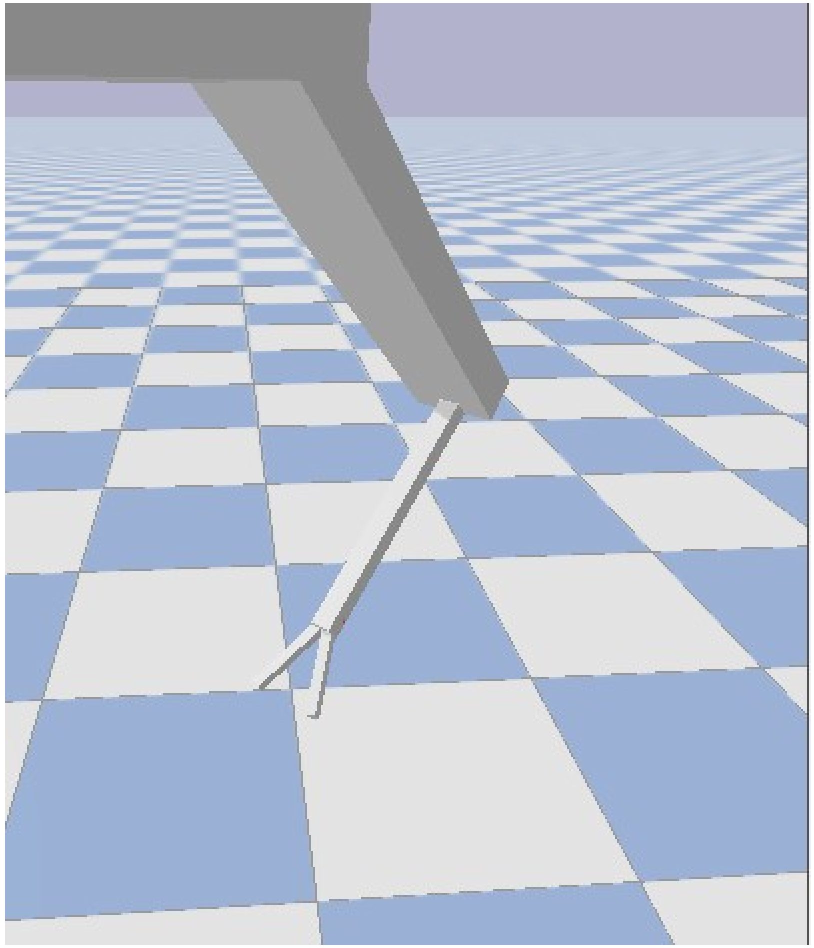
- PyBullet implementa Damped Least Squares para calcular los sistemas de cinemática inversa.
- Utilizaremos el método `calculateInverseKinematics`

```
jointPoses = p.calculateInverseKinematics(robotId, robotEndEffectorIndex, pos_target)
```

  - robotId: id del robot
  - robotEndEffectorIndex: Último índice de articulación antes del EndEffector.
  - pos_target: posición en cartesianas objetivo
- Devuelve los ángulos/posición de cada JOINT para alcanzar pos_target.

Cinemática Inversa PyBullet





Cinemática Inversa PyBullet


```
jointPoses = p.calculateInverseKinematics(robotId, robotEndEffectorIndex, pos_target)
```

5

5 JOINTS

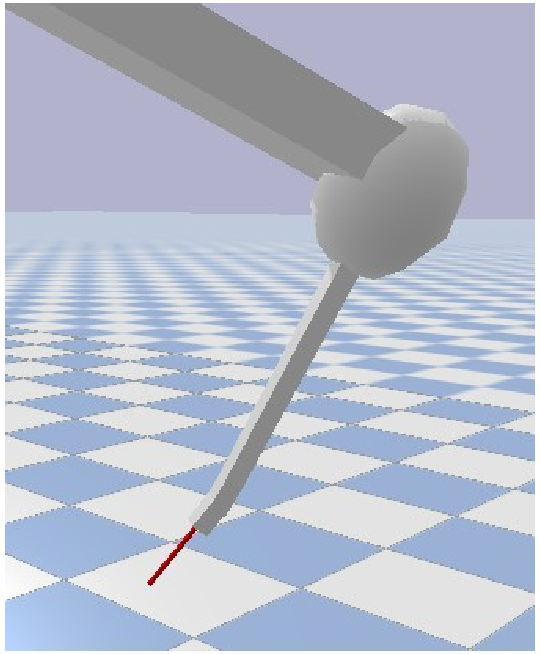
¿?

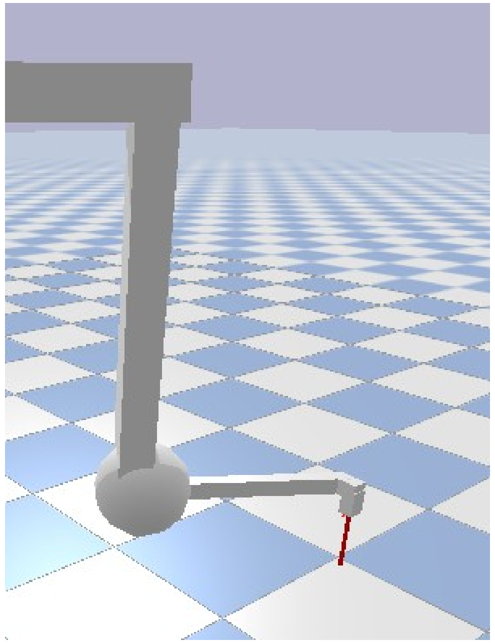
- Devuelve una lista de posición de joints para cada grado de libertad.
- Joints de tipo 'fixed' son omitidos.
- La cinemática Inversa solo es calculada sobre los primeros N Joints definidos por robotEndEffectorIndex

```
NumJoints: 5
0 - joint_root_column
1 - joint_top_platform_arm1
2 - joint_arm1_arm2
3 - joint_arm2_grip1
4 - joint_arm2_grip2
```

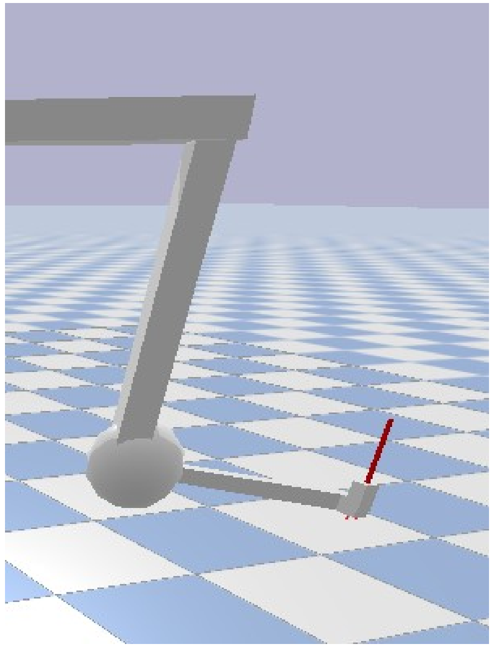
(0.8083609741375861, -0.7254140422506544, -1.7386886072323537, -3.025709040731816e-09, -3.0257217706833223e-09)

Orientación EndEffector





angles = [0, math.pi/2, 0]



angles = [0, -math.pi/2, 0]

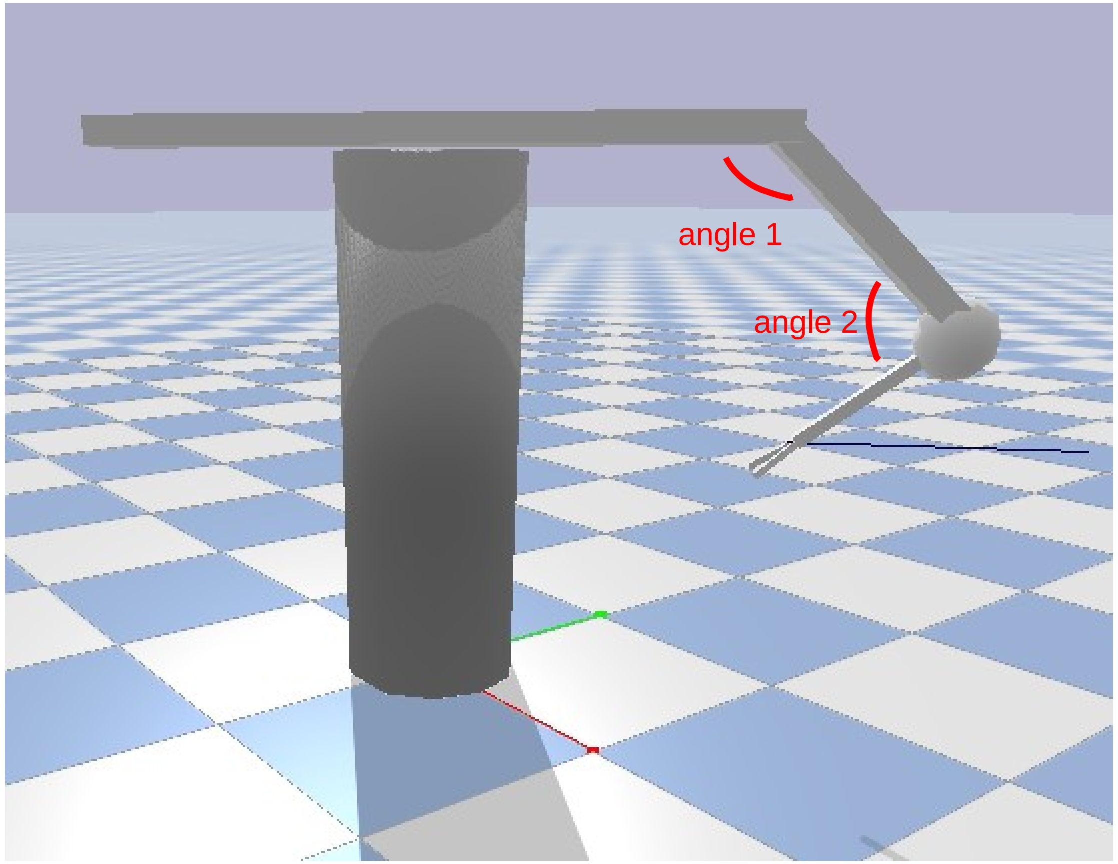
```
orn = p.getQuaternionFromEuler(angles)
jointPoses = p.calculateInverseKinematics(robotId, robotEndEffectorIndex, pos_target, orn)
```

Cinemática Inversa PyBullet

- Pybullet no tiene en cuenta las limitaciones de movimientos definidas en el URDF del modelo.
- Consulta el manual de pybullet, para ver como pasarle los siguientes límites al método calculateInverseKinematics:
  - LowerLimits
  - UpperLimits

Cinemática Inversa PyBullet

- Veamos un ejemplo [gitlab]



Referencias

- Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods
- Deeply-learnt damped least-squares (DL-DLS) method for inverse kinematics of snake-like robots

URJC

Escuela de Ingeniería
de Fuenlabrada

GSyC

RoboticsLabURJC
Programming Robot Intelligence