Modificación de brazo 7 DOF del Boston Dynamics Spot

Marcelo Contreras, *Estudiante, UTEC,* Alejandro Del Río, *Estudiante, UTEC,* Cesar Guillen, *Estudiante, UTEC,* y Andrea Chu, *Estudiante, UTEC*

***Abstract*—El presente proyecto se ha encargado de modificar el brazo robótico del Spot Robot de Boston Dynamics para aumentarle una articulación prismática, obteniendo un robot manipulador de 7 grados de libertad para amplificar el espacio operacional y alcanzable. Se definieron nuevas dimensiones y en base a ellas, se obtuvieron los parámetros Dehavit - Hartenberg. El robot modificado fue modelado mediante Inventor, generando un modelo enmallado que fue exportado como archivo formato URDF para utilizarse en ROS. Fueron definidas tres configuraciones de prueba que fueron visualizadas mediante Rviz para cinemática directa, cinemática inversa de posición y control cinemático de posición. También, fue obtenido el modelo dinámico del robot a través de la librería RBDL y fue empleado para dos esquemas de control dinámico. Tanto el control cinemático como dinámico logró seguir las referencias con una norma de error menor a 0.0001.**

***Index Terms*—Spot, Robotic Arm, 7 DOF, URDF, ROS, Gazebo, Rviz, RBDL, FK, IK, Control**

I. Introducción

EL 2016 fue un año importante para la robótica industrial y la investigación de robots cuadrúpedos por el anuncio del Spot Robot de la empresa Boston Dynamics [7] [2] [5]. Este robot muestra características en su locomoción ya vistas anteriormente por otros robots como Anymal del ETH Zurich [8] o robot Cheetah del MIT [3] pero logró compactarlas en un robot sumamente ágil de tamaño mediano y mostrando prestaciones de mayor calidad. Spot representa una oportunidad por integrar a los robots móviles en industrias como minería, petroquímica, energía, etc; gracias a su capacidad de movilidad y localizarse en tiempo real [2]. Para mejorar el desempeño de esta tarea, se le agregó un brazo de 6 DoF con un gripper para darle mayor autonomía al abrir puertas o mover obstáculos. Aplicaciones más específicas de Spot junto a su brazo son el monitoreo ante fallas de equipos industriales con recolección de data a través de cámaras integradas, mapeo de terrenos peligrosos o inaccesibles, transporte de cargas pequeñas en terrenos difíciles para la locomoción por ruedas como escaleras [4].

Este proyecto busca ampliar las capacidades del Spot Arm al agregarle un grado de libertad prismático para aumentar el espacio operacional alcanzable. Así mismo, para establecer posición y orientación en el espacio 3D solo se necesitan 6 DoF. Al incrementar su número, ahora el robot dispone de múltiples configuraciones que le permitan llegar hasta su objetivo al tener un DoF extra. Ello es de gran ayuda ante la presencia de obstáculos en entorno dinámicos donde se deben generar configuraciones alternativas en tiempo real.

Fig. 1: Robot Spot con su brazo de 6 DoF [1]

II. Componentes

La estructura del modelo modificado está conformada por una cadena cinemática de 4 articulaciones de revolución, 1 prismática y 2 adicionales de revolución, lo cual forma una estructura antropomórfica. Como gripper, el robot posee una garra mecánica de 2 dedos con un dedo fijo como soporte de agarre.

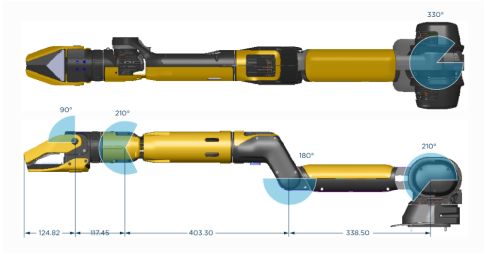


Fig. 2: Spot arm original [1]

En el apartado de sistemas de actuación, Spot Arm Mod alimenta sus sistemas eléctricos con baterías LiPo montadas en la base móvil. Los principales componentes eléctricos son servomotores DC que dentro tiene una etapa de potencia integrada y son directamente manipulados por protocolo de comunicación UART (RS-485). Un posible modelo son los Dynamixel XL-320 con control PID de bajo nivel por su compacticidad y calidad. Para aumentar su torque, se utiliza engranajes planetarios en las articulaciones revolución. Adicionalmente, para lograr el movimiento lineal de la articulación prismática, se acopla un mecanismo ball screw a uno de los servomotores.

La percepción sensorial del robot está constituida por galgas extensométricas en el efector final para sensado de fuerza y

Fig. 3: Dynamixel XL-320

una cámara instalada en la base móvil. Los servomotores de por sí ya tienen encoders instalados que no son manipulados por los usuarios en virtud del control de bajo nivel de los servos. Aunque es posible que Spot tenga IMUs y LiDAR en su base, estos no aporta información relevante para el brazo de 7 DoF.

Finalmente, se ha optado por un control descentralizado de los 7 DoF para que no solo el control, sino la percepción y localización del robot puedan ser ejecutadas en alto nivel por una single board computer en vista que ya existe un control de bajo nivel para cada actuador. La placa seleccionada ha sido la Jetson TX2 con un NVIDIA Denver 2 Dual Core, ARM Cortex Quad Core, 8 GB LPDDR4 de memoria y una GPU Nvidia Pascal capaz de hacer procesamiento de video de la cámara.

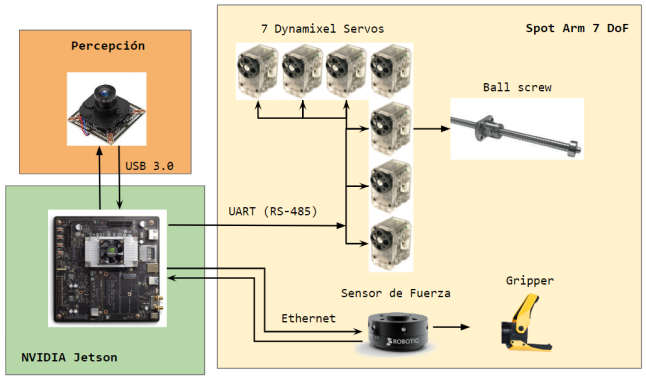


Fig. 4: Arquitectura de Spot Arm propuesta

III. Modelo del Robot

Para representar el robot en ROS y poder simularlo o visualizarlo, primero se debe definir un modelo de tipo URDF (Unified Robot Description Format). Primero, se modeló el Spot Arm usando Fusion360(Inventor), tomando en cuenta las medidas reales del robot con las modificaciones aplicadas. Al crear este modelo 3D, se definieron diferentes cuerpos por cada articulación. Asimismo, se establecieron los ejes por los que podrán rotar o desplazarse.

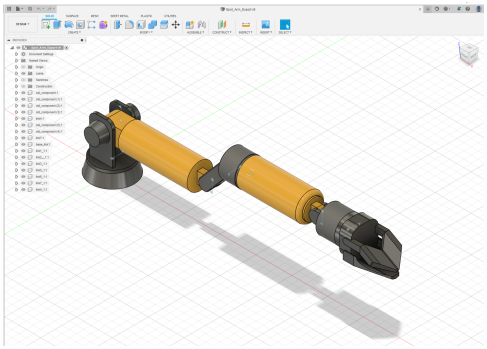


Fig. 5: Spot Arm modelado en Fusion 360

Se utilizó un repositorio público de Github [9] para exportar el modelo de Fusion 360 a un URDF. Al realizar esto se crearon los archivos de tipo '.urdf', '.xacro' y '.launch'. En el archivo Spot_Arm_Export.xacro se definen las propiedades de cada articulación, como la longitud, masa e inercia. Asimismo, al generar el URDF se creó un enmallado para cada elemento del modelo.

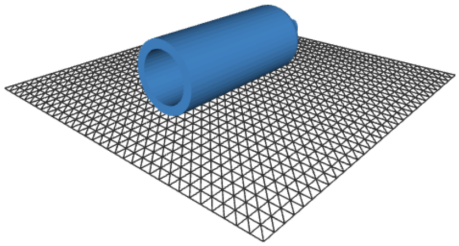


Fig. 6: Mesh en componentes del robot

Para poder visualizar el robot en Rviz se creó el archivo display.launch. Este archivo muestra el programa y permite publicar al nodo joint_state_publisher para manipular cada articulación. En caso se intente publicar los valores de las articulaciones desde un script externo, se tendrá que eliminar el nodo joint_state_publisher del archivo display.launch para evitar que se publiquen desde dos instancias diferentes.

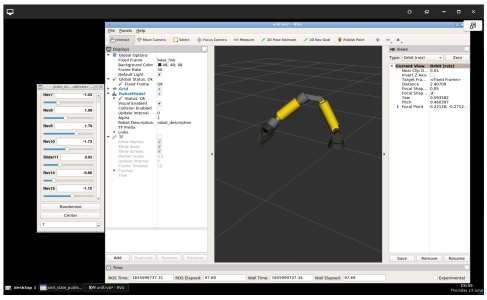


Fig. 7: Spot Arm visualizado en Rviz

De igual manera, se creó el archivo gazebo.launch, el cual permite simular el modelo en Gazebo.

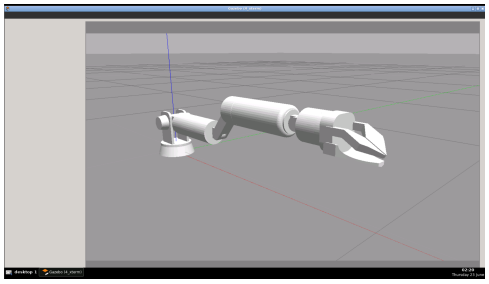


Fig. 8: Spot Arm simulado en Gazebo

IV. Cinemática directa e inversa

Para el establecimiento del modelo cinemático del robot en estudio, es necesario establecer uniformidad sobres las pruebas a realizar. En ese sentido, se ha seleccionado un total de tres configuraciones a manera de vectores articulares $\boldsymbol{q}$ para que sean asignados al modelo del robot y puedan ser evaluados desde los distintos procedimientos requeridos. Para la cinemática directa se empleará la convención de Denavit-Hartenberg estándar, mientras que para la cinemática inversa se compararán los resultados de dos métodos numéricos, el de Newton-Raphson y el descenso de Gradiente.

Tomando en consideración los límites articulares que han sido establecidos por Boston Dynamics, se han definido los siguientes vectores articulares para poner a prueba en la experimentación computarizada que permite comprobar la correcta operación del código trabajado:

1) $\boldsymbol{q}$ = {2.18, 2.86, 1.93, -2.77, 0.04, 0.8, 0.38}
2) $\boldsymbol{q}$ = {0.36, 2.89, 2.40, -1.96, 0.03, -0.04, -2.05}
3) $\boldsymbol{q}$ = {1.85, 2.93, 1.76, -1.55, 0.04, -0.35, 0.33}

A. Cinemática directa

En primer lugar, se requiere establecer la cadena cinemática del robot a manera de expresiones matemáticas que vinculen sus articulaciones y eslabones de manera coherente. Para esto, siguiendo la convención estándar de Denavit-Hartenberg, los sistemas de referencia a lo largo del brazo analizado resultan del modo que se muestra a continuación:

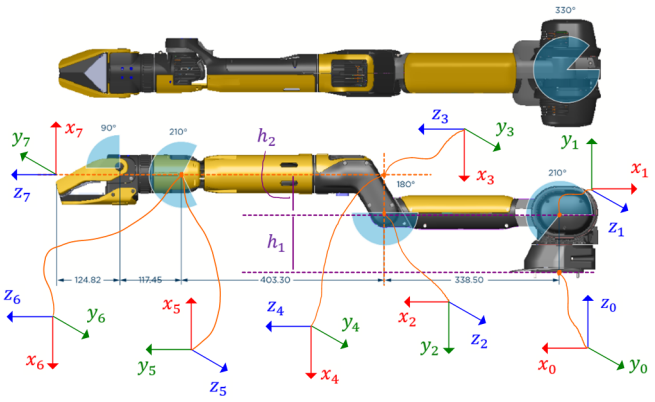


Fig. 9: Asignación de sistemas de referencia de acuerdo con la convención estándar de Denavit-Hartenberg

Los parámetros correspondientes a la asignación de ejes mostrada en la figura 9 se indican en la tabla I.

TABLE I: Parámetros DH estándar

	d_i	θ	a	α
1	0.140137	q_0+π	0	$\pi/2$
2	0	-q1+π	0.3385	0
3	0	q2+$\pi/2$	- 0.09734	$\pi/2$
4	0	q3	0	0
5	q4+0.3833	π	0	$\pi/2$
6	0	q5+π	0	$\pi/2$
7	0.275027	q6+π	-0.05025	0

Una vez definidos los parámetros, cabe resaltar que la finalidad de estos consiste en la obtención de matrices de transformación homogénea que relacionen los sistemas de referencia asignados entre sí. Al multiplicarse en secuencia, construyen la cadena cinemática presente en el robot en estudio para alcanzar el efector final comenzando en el sistema de referencia de la base. Para la convención estándar de Denavit-Hartenberg, existe una expresión general que define la construcción de la matriz de transformación homogénea que relaciona dos sistemas consecutivos a lo largo del robot.

$$ {}^{i-1}T_{i(\theta_i,d_i,\alpha_i,a_i)} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

Con la expresión general de la matriz de transformación expuesta en 1, se puede implementar el código en Python para realizar el envío del vector articular deseado al modelo del robot. Para comprobar el correcto cálculo de la cinemática directa, se realizó la visualización en Rviz. Para diferentes configuraciones, se colocaron marcadores para cada eje definido en la convención Denavit-Hartenberg (en verde) y para el eje final (en amarillo).

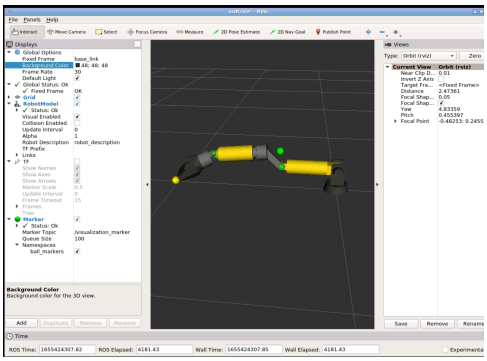


Fig. 10: Cinemática Directa en Rviz

Por medio de ambas figuras anteriores, tanto 10 como 11, se aprecia la generación de marcadores en los puntos en los que han sido asignados los sistemas de coordenadas siguiendo la distribución espacial planteada en 9. Lo siguiente consiste en entregar los valores articulares que fueron definidos al comienzo de esta sección y corroborar que el robot adopte dicha configuración sin complicaciones. Así, se puede continuar con el estudio del robot para entregar un valor deseado en el espacio operacional en lugar del de configuración.

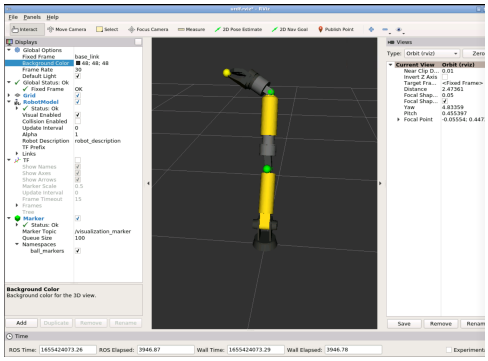
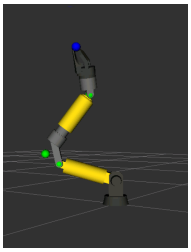
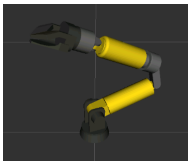


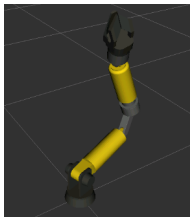
Fig. 11: Cinemática Directa en Rviz



(a) 1ra configuración



(b) 2da configuración



(c) 3ra configuración

Fig. 12: Visualización de las configuraciones articulares obtenidas por medio de cinemática directa

A partir de la correcta ejecución del código, se visualizan las tres configuraciones con las que se realizarán los ensayos posteriores respecto a cinemática inversa, control cinemático y dinámico para el robot en estudio. La operatividad del programa ha quedado verificada hasta este punto. Sin embargo, hay un paso previo antes de continuar con la sección correspondiente a cinemática inversa. Dado que debe entregarse al algoritmo un punto en el espacio para que sea alcanzado por el efector final, se extrae de la última columna de las matrices de transformación homogénea obtenidas las coordenadas x, y y z de los tres puntos deseados a ser verificados.

```
[[-0.919 -0.277  0.282  0.21 ]
 [ 0.302 -0.952  0.051 -0.124]
 [ 0.254  0.132  0.958  0.882]
 [ 0.     0.     0.     1.   ]]
```

Fig. 13: Matriz de transformación homogénas devuelta para la primera configuración de prueba

Siguiendo el procedimiento descrito para aprovechar los resultados de la presente sección, la lista de puntos deseados comprende lo siguiente:

1) $\mathbf{X}_d$ = {0.21; -0.124; 0.882}
2) $\mathbf{X}_d$ = {0.188; 0.123; 0.659}
3) $\mathbf{X}_d$ = {-0.011; -0.143; 0.886}

Con la lista de puntos deseados disponible y sabiendo que el cálculo de cinemática directa opera apropiadamente para el envío de la configuración articular al robot, se continúa con la revisión de la cinemática inversa.

B. Cinemática inversa

En lo que respecta al análisis de cinemática inversa, se han realizado pruebas con dos algoritmos numéricos para comprobar los resultados deseados. Estos dos algoritmos son el de Newton-Raphson y el descenso de Gradiente. Para la aplicación de ambos, se requiere el cálculo del correspondiente Jacobiano del robot manipulador en estudio. Esta matriz corresponde a la razón de cambio para cada una de las coordenadas x, y y z respecto al vector articular q.

$$J_{(\mathbf{q}_k)} = \frac{\partial f_{(\mathbf{q}_k)}}{\partial \mathbf{q}} \in \mathbb{R}^{n \times m} \tag{2}$$

A partir de 2, se sabe que el robot en análisis presenta como Jacobiano una matriz de dimensiones 3×7, ya que cuenta con un vector articular que va desde q_0 hasta q_6. Así, se trata de un robot de 7 grados de libertad en el espacio tridimensional. De este modo, se sabe que no se cuenta con una matriz cuadrada, lo que hace necesario el uso de su correspondiente pseudo-inversa de Moore-Penrose para los algoritmos a utilizar para la presente sección.

Existen dos formas de conseguir el cálculo de los valores que conforman esta matriz y trabajar con ellos. Debido al tipo de resultado que se busca y a que se trata de una implementación de código de tipo computacional, se opta por el cálculo numérico del Jacobiano. Esta aproximación permite evaluar la matriz sin necesidad de contar con la totalidad de las expresiones matemáticas simbólicas que se obtienen de realizar el producto de las matrices de transformación homogénea de la cadena cinemática completa.

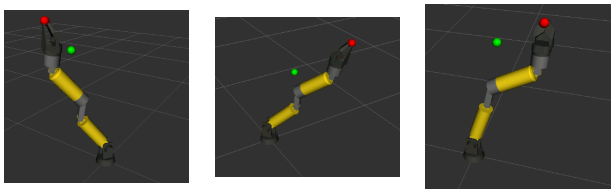
La expresión que define el Jacobiano numérico consta de la utilización de un incremento articular que es aplicado a las articulaciones conforme se progresa en la aplicación del método numérico escogido. Por tanto, se consigue una expresión semejante a la utilizada para llevar a cabo la integración de Euler. Esta ecuación está dada del siguiente modo:

$$\frac{\partial \mathbf{X}}{\partial q_1} \approx \frac{\mathbf{X}_{(\mathbf{q}+\delta q_1)} - \mathbf{X}_{(\mathbf{q})}}{\delta q_1} \tag{3}$$

En 3, cabe resaltar que $\mathbf{X}$ hace referencia a una posición en el espacio tridimensional con las tres coordenadas correspondientes. Además, para el modo en el que se ha planteado, el resultado de dicho cálculo corresponde a una de las columnas de la matriz que constituye el Jacobiano del robot. El procedimiento se repite iterativamente para completar el recorrido de todo el vector articular.

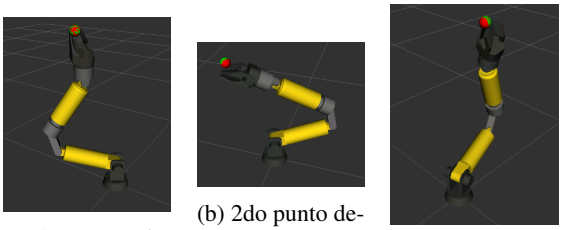
Habiendo definido este criterio para la obtención del Jacobiano del robot en estudio, se procede a describir las características principales de los métodos numéricos a ser evaluados para mostrar los resultados conseguidos para el modelo del Spot Arm. De igual manera, es importante mencionar que la implementación de este cálculo está dada en la función de Python jacobian_Spot que forma parte del programa adjunto a manera de repositorio de GitHub.

1) Método de Newton-Raphson: Debido a que se está empleando como forma de resolver el problema de cinemática inversa, el método numérico en cuestión busca la solución $f_{(\mathbf{q})}$



(a) 1er punto deseado (b) 2do punto deseado (c) 3er punto deseado

Fig. 14: Obtención de cinemática inversa con el método de Newton-Raphson con configuración inicial nula



(a) 1er punto deseado (b) 2do punto deseado (c) 3er punto deseado

Fig. 15: Obtención de cinemática inversa con el método de Newton-Raphson con configuración inicial no nula

tal que se cumpla la condición $\mathbf{X}_d - f_{(\mathbf{q})} = 0$. Esto implica que el resultado sea un vector articular con valores dados que obtenga como resultado de posición del efector final una posición deseada.

A partir de emplear una aproximación de primer orden de Taylor, la ecuación de actualización iterativa para el método es la que se muestra a continuación:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + J^{\#}_{(\mathbf{q}_k)} \cdot \left(\mathbf{X}_d - f_{(\mathbf{q}_k)}\right) \tag{4}$$

Para la ecuación anterior, $J^{\#}_{(\mathbf{q}_k)}$ representa la pseudo-inversa del Jacobiano. Este concepto requiere ser aplicado debido a que no se cuenta con una matriz cuadrada. Por tanto, esta no tiene inversa. Sin embargo, puede ser directamente calculada mediante la función disponible en numpy denominada numpy.linalg.pinv. Más adelante, se observará la posibilidad de emplear la pseudo-inversa amortiguada. Sin embargo, esta resulta útil dentro de la aplicación del control sobre el robot. Cuando se trata de evaluar posiciones específicas como en esta sección, se aprovecha la función ya mencionada.

Es importante mencionar que, para este método, el resultado no es único. Más aún, si se trata del espacio de trabajo diestro del robot, se contará con multiplicidad de soluciones dado que la referencia consta solamente de una posición que se busca alcanzar. Esto se ve reflejado al momento de obtener los primeros resultados con la aplicación del método.

Por medio de la figura 14, se observa que el método no ha alcanzado el punto deseado con el vector articular calculado. Principalmente, se muestran estas capturas para evidenciar que las articulaciones de revolución adoptan valores viables, pero que finalmente impiden que se alcance el punto final debido a que se requiere un valor negativo (retroceso) en la articulación prismática que fue agregada al robot. Por tanto, el valor no puede ser asignado al modelo, ya que es conflictivo con su estructura. Las pruebas anteriores han sido realizadas con una configuración inicial de cero para cada uno de los valores del vector articular. Se obtiene convergencia del método pues se ha impreso una configuración en el modelo visualizado en RViz. Sin embargo, no es el comportamiento deseado aún.

Para la evaluación del algoritmo, es relevante tomar en consideración que el método de Newton-Raphson obtiene distintas soluciones dependiendo del punto de partida que sea aplicado como referencia inicial para comenzar a buscar la solución deseada. En esta implementación del código, se ha optado por el criterio de parada de una norma mínima para el valor del error, el cual hace que las iteraciones se detengan y se proporcione el vector articular despejado como respuesta. Entonces, se modificó el vector articular inicial empleado para conseguir la variación de la respuesta final que se asigna al modelo del robot en estudio.

La variación en el resultado es notoria. Ahora, se cuenta con la posición deseada del efector final al conseguir la coincidencia de los marcadores verde y rojo que han sido incluidos para obtener los resultados en RViz. En los tres casos, ya no se tiene dificultades en la asignación de valores a la articulación prismática, ya que se extiende correctamente para alcanzar los valores deseados con el efector final. Claramente, el vector articular inicial está provocando la diferencia de resultados entre ambas pruebas, ya que al comparar lo observado en 14 con lo que se tiene en 15, se aprecia que el robot en estudio busca aproximarse de maneras diferentes al punto deseado para el efector final. Sobre todo, con las articulaciones iniciales de su cadena cinemática.

El nuevo vector articular asignado para conseguir el resultado esperado que se visualiza en la figura 15 ha sido obtenido a partir de las configuraciones asignadas por cinemática directa, variando los decimales de las componentes del vector q para que el robot se encuentre cercano al resultado deseado. Efectivamente, el cambio ha sido positivo dado que en la segunda prueba se logró que el efector final alcance las coordenadas espaciales de $\mathbf{X}_d$. Para concluir la evaluación del método de Newton-Raphson, es pertinente exponer la convergencia del mismo en un número de iteraciones dado para observar el decremento en el valor del error obtenido (distancia entre la posición deseada y la actual del efector final).

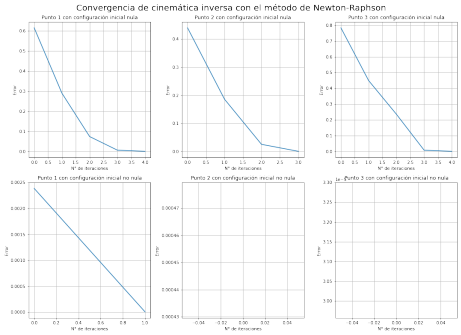


Fig. 16: Convergencia del método de Newton-Raphson utilizado para cinemática inversa

Como se observa en la figura 16, el algoritmo no requiere demasiadas iteraciones para alcanzar un resultado. Le ha representado un mayor costo computacional determinar el vector articular deseado al comenzar desde la configuración de solamente ceros. Lógicamente, esto se debe a que el efector final está más lejos de la posición en la que se busca que se encuentre. Asimismo, las articulaciones están también relativamente lejanas de los valores finales. Respecto a las gráficas de la segunda fila, donde se tiene una configuración inicial cercana a la deseada, no se alcanza a observar el comportamiento de convergencia cuadrático. Sucede tan rápidamente que basta con una sola o máximo dos iteraciones para conseguir el vector articular deseado. Este es el motivo de que las gráficas no sean tan representativas del comportamiento cuadrático que sí se aprecia de mejor forma en la primera fila de gráficas.

2) Método del descenso de Gradiente: De manera semejante a como se procedió con el método de Newton-Raphson, también se ha llevado a cabo la prueba del método del descenso de Gradiente. Su finalidad es la misma, ya que trabaja en base a un error consistente en la diferencia presente entre la posición actual y deseada del efector final. Sin embargo, no utiliza una aproximación de primer orden de Taylor, sino que utiliza el gradiente de una función escalar del error. Con esto, la búsqueda de la solución al problema de cinemática inversa consiste en continuar en la dirección opuesta al máximo incremento (gradiente) del error, para así conseguir el máximo decremento del mismo.

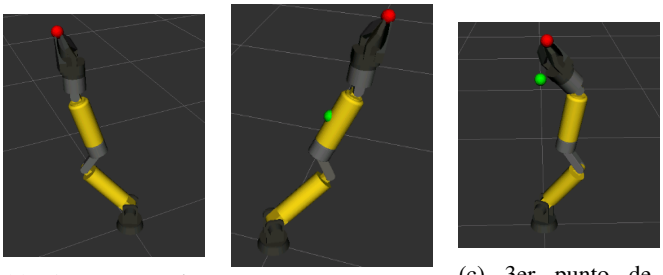
A partir de lo anterior, la expresión para su actualización iterativa del vector articular comprende la aplicación de un valor constante multiplicado a la transpuesta del Jacobiano, siendo este el tamaño de paso a ser evaluado por medio del método:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \alpha \cdot J^T_{(\mathbf{q}_k)} \cdot \left(\mathbf{X}_d - f_{(\mathbf{q}_k)}\right) \tag{5}$$

Este segundo método numérico para el problema de cinemática inversa resulta menos costoso para la computadora debido a que elimina la necesidad del cálculo de la pseudo-inversa y por lo tanto, la posibilidad de llevar el robot a singularidades por una inversa mal comportada. Además, como se está empleando un Jacobiano obtenido de manera numérica y no analítica, esto simplifica aún más las operaciones que debe realizar la computadora. Sin embargo, esto no significa que se trate de una alternativa más eficiente directamente. Su principal inconveniente es su comportamiento lineal de convergencia, lo que lo hace más lento en comparación con el método de Newton-Raphson.

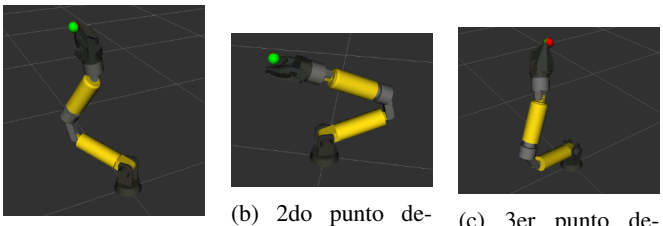
Con la finalidad de mantener la uniformidad de las pruebas realizadas para los algoritmos empleados, se presentan los resultados evaluados de la misma forma que para el método de Newton-Raphson. Primero, se aplica una configuración de solamente ceros en el vector articular. Segundo, se utiliza como referencia los vectores articulares deseados. Tercero, se exponen las gráficas de convergencia con los valores de error.

En el resultado inicial, es decir, al partir de una configuración con solamente ceros en los valores articulares, el modelo encuentra el mismo problema que para el método de Newton-Raphson. El único modo de alcanzar efectivamente el punto deseado para el efector final es generando un valor negativo para la articulación prismática. Se sabe que esto resulta inviable para el robot en estudio. Por tanto, se realiza nuevamente la prueba del mismo programa, pero con la asignación de un vector articular inicial que se encuentre más cercano al resultado final que se podría obtener de modo apropiado respecto a la estructura funcional del robot.



(a) 1er punto deseado (b) 2do punto deseado (c) 3er punto deseado

Fig. 17: Obtención de cinemática inversa con el método del descenso de Gradiente con configuración inicial nula



(a) 1er punto deseado (b) 2do punto deseado (c) 3er punto deseado

Fig. 18: Obtención de cinemática inversa con el método del descenso de Gradiente con configuración inicial no nula

Una vez más, al tomar como referencia los vectores articulares empleados para cinemática directa y acercar al robot a dicha configuración desde el principio, el método opera satisfactoriamente. Ambos marcadores coinciden en su posición en el espacio y se respetan todos los límites articulares en la solución resultante. En ese sentido, la influencia del vector articular inicial que es asignado al algoritmo tiene un impacto relevante en la obtención de los resultados deseados. Además, se aprecia que la única generación de obstáculos está dada por la inclusión de una articulación de un tipo distinto a las demás, siendo q_4 la única articulación prismática, mientras que el resto son de revolución.

A continuación, se exponen las gráficas obtenidas en torno a la convergencia del método para los seis casos evaluados, tanto con la configuración inicial compuesta solamente por ceros, como la que se encontró desde el principio más cercana a lo deseado.

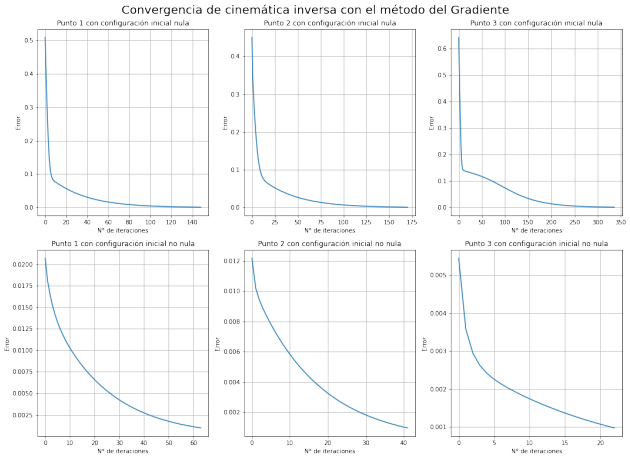


Fig. 19: Convergencia del método del descenso de Gradiente utilizado para cinemática inversa

La diferencia principal para los resultados mostrados en 19 comprende la mayor cantidad de iteraciones para conseguir el resultado deseado. Para este método, también se ha empleado el criterio de error de norma mínima para detener el algoritmo y que devuelva como respuesta un vector articular determinado. Si bien pareciera conseguirse una curva no lineal, en realidad lo que sucede es una visualización de este estilo debido a la cantidad de puntos disponibles para la gráfica. En el caso del método de Newton-Raphson, como se mencionó al comentar la figura 16, no se dispone de muchos puntos pues la convergencia es más rápida en comparación de ambos métodos empleados. En cambio, el comportamiento suavizado de la curva de error está dado de ese modo justamente por la mayor cantidad de iteraciones requeridas, lo que genera más puntos para el gráfico y provocan la formación de una curva más suave.

Habiendo realizado las correspondientes pruebas con ambos métodos, se concluye la sección correspondiente para el modelo cinemático del robot en estudio. Se ha comprobado la operatividad del algoritmo para despejar la cinemática directa, lo que verifica el correcto planteamiento de los parámetros de Denavit-Hartenberg y la realización del modelo en RViz. Luego, se ha comprobado la influencia del vector articular inicial que es proporcionado a los algoritmos de cinemática inversa, ya que la multiplicidad de soluciones puede acarrear algunas dificultades para alcanzar el resultado deseado. Sobre todo, vale la pena observar si los tipos de articulación del robot son todos de la misma naturaleza y el cómo se afecta el cálculo del resultado en cuanto se aplican combinaciones de los mismos. Una vez ha quedado establecido el modelo cinemático, se procede a aplicar control sobre el comportamiento del brazo del Boston Dynamics Spot con el que se está trabajando.

V. Control cinemático

Este tipo de control utiliza la cinemática diferencial para calcular la primera derivada temporal de los valores articulares para luego integrarlos y obtener una nueva configuración. Primero se define el vector de posición actual y el vector de posición deseada:

$$x = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \; x_d = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix}$$

El error por tanto se definirá como $e = x - x_d$, y su derivada temporal, dado que x_d es constante:

$$\dot{e} = \dot{x} \tag{6}$$

De la definición del Jacobiano analítico sabemos que $\dot{x} = J\dot{q}$, conociendo la ecuación 6 y resolviendo para $\dot{q}$, obtenemos:

$$\dot{q} = J^{\#}\dot{e} \tag{7}$$

donde $J^{\#}$ es la pseudo-inversa del Jacobiano. Finalmente, usamos una ganancia proporcional k para definir la ley de control:

$$\dot{e}^{*} = -ke \tag{8}$$

Como se puede observar, este método tiene como resultado $\dot{q}$, por lo que será necesario integrar por el método de Euler con el periodo de actualización y el q pasado para obtener la nueva configuración articular. Dicho periodo fue de 0.005 segundos debido a que la frecuencia de actualización fue de 200 Hz. Asimismo, se definió como valor aceptable para la norma del error 0.0001 m. Este valor debe ser alcanzado dentro del límite de 10000 iteraciones que se fijó para el algoritmo. Además, para cubrir también los casos de singularidad se añadió un condicional que implementa la pseudo-inversa amortiguada del Jacobiano si el rango de este es menor a 3. Esta tiene la forma:

$$J^{+} = J^{T}(JJ^{T} + \rho^{2}I)^{-1} \tag{9}$$

donde se usó para ρ un valor de 0.01. Con el fin de evitar publicar configuraciones fuera de los límites articulares, se implementó también una serie de condicionales que solo actualizan cada valor del vector q si los nuevos valores están dentro del rango posible.

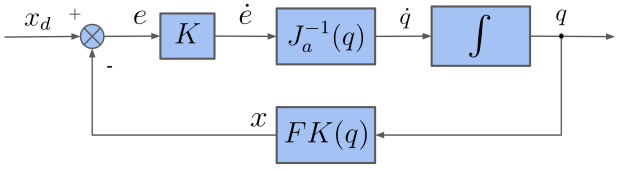


Fig. 20: Diagrama Control Cinemático

En la figura 20 se puede observar la definición del diagrama cinemático, donde $FK(q)$ denota la cinemática directa en función de q descrita en la sección anterior.

A. Implementación del control cinemático

Para probar el rendimiento del algoritmo presentado se usaron las mismas tres configuraciones de la cinemática directa. Con el método de prueba y error, se encontraron los valores de k más bajos para que el robot llegue a la posición deseada

dentro de las 10000 iteraciones. Estos valores fueron bastante altos principalmente debido a las limitaciones impuestas por los límites articulares.

TABLE II: Resultados del control cinemático

	Caso 1	Caso 2	Caso 3
k	12	5	10
Iteraciones	9065	9046	9271

En la figura 21 se observan las gráficas de los valores articulares comparados con la referencia inicial a lo largo del tiempo.

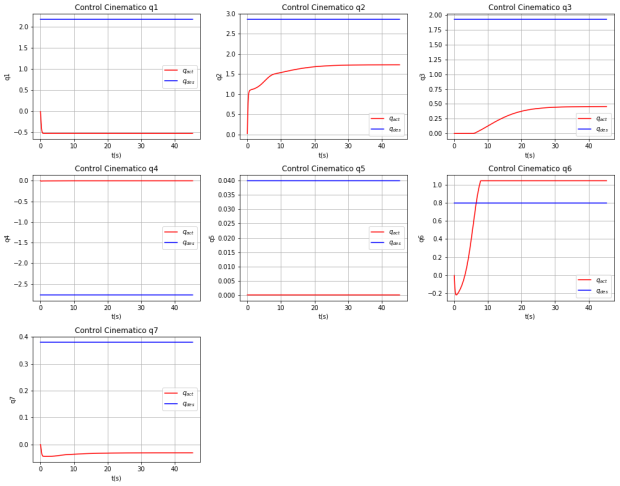


Fig. 21: Cambio de las variables articulares con el control cinemático en la configuración 1.

Cómo se puede ver, si bien todos los valores articulares convergen, estos no siguen la referencia inicial. Esto se debe a que el control cinemático se realiza sobre el espacio operacional y no el articular. Dado que solo se está realizando el control de posición y el robot es de 7 grados de libertad, existen infinitas soluciones para unas coordenadas determinadas. Si bien la configuración uno fue usada para generar la posición final, el robot puede usar otra configuración para llegar a la misma posición. Además, en el gráfico de la articulación q_6 se ve como actúa el condicional de los límites articulares, pues esta dejó de variar en un valor cercano a 1.047 rads, su límite superior.

El éxito del control por posición es mejor visualizado por las gráficas que registran el cambio en las coordenadas x, y, z del efector final.

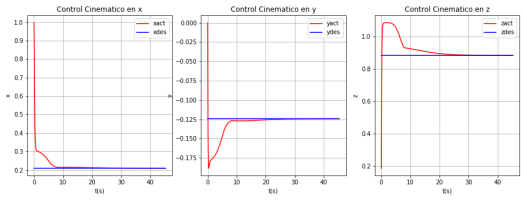


Fig. 22: Cambio de posición con el control cinemático en la configuración 1.

TABLE III: Parámetros de desempeño configuración 1

	x	y	z
%OS	0	48	29
e_{ss} (m)	0.000023	-0.000014	0.000096
T_s (s)	22	28	30

Como se puede observar, las gráficas no tienen un comportamiento estable y varían ampliamente. Esto se debe principalmente a los límites articulares impuestos. Asimismo, en el caso de la configuración 1 se puede ver un cambio muy rápido en x, y y z al inicio del movimiento. Este es consecuencia del alto valor de k usado en esta configuración.

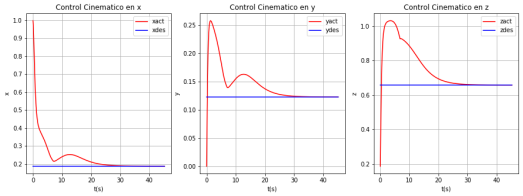


Fig. 23: Cambio de posición con el control cinemático en la configuración 2.

TABLE IV: Parámetros de desempeño configuración 2

	x	y	z
%OS	0	108	77
e_{ss} (m)	0.000054	-0.000034	0.000077
T_s (s)	30	32	30

Para la configuración 2 se usó un valor de k mucho menor, por lo que el movimiento es mucho más suave.

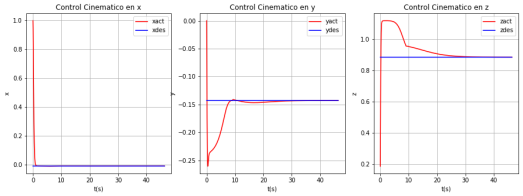


Fig. 24: Cambio de posición con el control cinemático en la configuración 3.

TABLE V: Parámetros de desempeño configuración 3

	x	y	z
%OS	0	86	31
e_{ss} (m)	-0.000001	-0.000019	0.000098
T_s (s)	2	30	30

Se puede notar que en todos los casos el robot avanza inicialmente muy rápido pero finaliza el movimiento de manera lenta. Este comportamiento es consecuencia directa de los métodos que hemos utilizado y es la razón por la que se necesitaron valores tan altos de k. El control cinemático esta definido por el error, y siempre va a buscar reducir el

mismo. Es decir, este nunca le va a indicar al robot que pase a una posición que aumente el error respecto a la referencia. De esta manera, el robot no puede ”esquivar” una posición fuera de su espacio operacional alejándose temporalmente de la referencia. Esto hace que siempre busque la ruta que reduzca constantemente el error, aunque esta sea más lenta.

Finalmente, también es posible visualizar el control con los markers verde y rojo en el entorno de RViz. Como se puede observar, ambas esferas, la roja indica posición deseada y la verde posición actual, están superpuestas debido a que ocupan la misma posición.

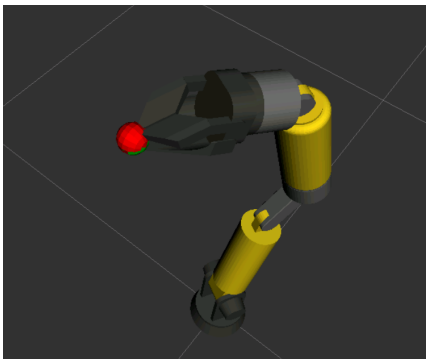


Fig. 25: Control cinemático en RViz.

B. Configuraciones singulares

Para hallar las configuraciones singulares del robot se debió hallar el Jacobiano geométrico pues la aproximación del Jacobiano analítico realizada no tendrá los resultados exactos. Asimismo, por simplicidad se tomaron únicamente las configuraciones singulares de posición y se asumió que las cuatro primeras articulaciones definen la posición del efector final. El Jacobiano geométrico tiene la siguiente forma:

$$J_g = \begin{bmatrix} z_0 \times p_{0,4} & z_1 \times p_{1,4} & z_2 \times p_{2,4} & z_3 \times p_{3,4} \end{bmatrix}$$

donde z_i es el eje de la articulación $i+1$ y $p_{i,4}$ es la distancia del origen del sistema de referencia i al origen sistema de referencia 4, ambos expresados en el sistema de referencia 0. Cómo se puede ver J_g depende de q_1, q_2, q_3 y q_4, y es una expresión trigonométrica compleja. Para que se la considere una configuración singular, esta tiene que cumplir la siguiente expresión:

$$\det(J_g J_g^T) = 0$$

De manera experimental se probó con distintas configuraciones y se dio que las siguientes cumplen con la expresión:

- $q = [0, \frac{\pi}{2}, 0, 1]$
- $q = [\frac{\pi}{2}, \frac{\pi}{2}, 0, 2]$

Estos resultados se explican en que ambos ponen al efector final en el límite del espacio operacional. Además, q_4 no afecta la posición por lo que esta no afecta el rango del Jacobiano. Si q_3 es distinto de 0, en cambio, el efector final sale del límite y el determinante da distinto de 0.

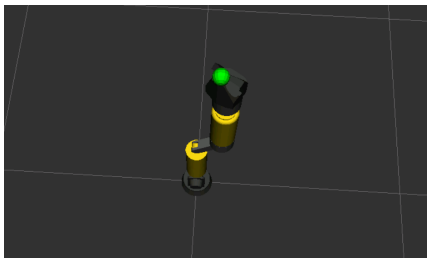


Fig. 26: Configuración singular $q = [0, \frac{\pi}{2}, 0, 1, 0, 0, 0]$

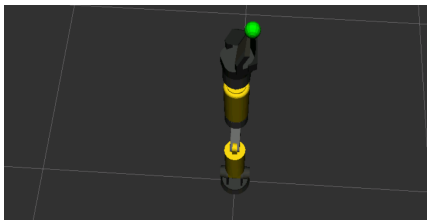


Fig. 27: Configuración singular $q = [\frac{\pi}{2}, \frac{\pi}{2}, 0, 2, 0, 0, 0]$

VI. Dinámica y control dinámico

A. Dinámica

La expresión general que puede representar la dinámica de un robot puede tomar la forma matricial siguiente según [11]:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q}) = \boldsymbol{\tau} \tag{10}$$

donde $M(\boldsymbol{q})$ es la matriz de inercia, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ la matriz de Coriolis, $g(\boldsymbol{q})$ el vector de gravedad y τ el torque resultante. Con robots de pocos DoF se puede interpretar rápidamente la dinámica inspeccionando los arreglos calculados, sobretodo la matriz de inercia para verificar si hay términos cruzados entre los DoFs que tiene un impacto notable para ciertos esquemas de control. No obstante, este análisis comienza a complicarse y pierde sentido físico para robots de más DoF. No obstante, la expresión sigue siendo de ayuda para el control dinámico, articular, operacional o de fuerzas. Por otro lado, es posible representar el modelo dinámico mediante espacio de estados, dejando de lado la forma matricial anterior.

Los dos métodos más conocidos para la obtención de la ecuación dinámica son Euler-Lagrange y Newton-Euler [11]. El primero se basa en el principio de mínima acción que obtiene de manera simbólica las expresiones dinámicas. Esta forma no es computacionalmente eficiente para robots de varios DoFs porque redunda en el planteamiento de operaciones y no es recursivo. Además, deben replantearse para cada tipo de robot diferente. El segundo es un método algorítmico recursivo hacia adelante para el cálculo de velocidades y aceleraciones, y hacia atrás para los torques y fuerzas. Es sumamente eficiente con un costo computacional de $O(n)$ porque no contiene

iteraciones anidadas. Inclusive, el método permite extraer cada matriz o vector dinámico individualmente, en vista que se necesitan para ciertas leyes de control.

Existen paquetes o librerías que tienen implementaciones del algoritmo Newton-Euler con ciertas optimizaciones, que terminan por denominarlo Recurrent Newton-Euler Algorithm (RNEA). Dentro de estas se encuentran Pinocchio y RBDL. Este trabajo hace uso de RBDL con la interfaz Cython para trabajar scripts nativos de C++ en Python [6].

Entrando en detalles, la obtención de las matrices y vectores dinámicos puede obtenerse manejando algebraicamente la expresión dinámica y observando los valores de torques resultantes que seran igual al array deseado directamente o luego de unas operaciones adicionales. Los valores de los vectores podrán ser retornados por la función InverseDynamics() que calcula el vector de torques.

A continuación, se muestra las cancelaciones de $\dot{\boldsymbol{q}}$ y $\ddot{\boldsymbol{q}}$ necesarias para obtener $q(\boldsymbol{q})$, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ y $M(\boldsymbol{q})$ respectivamente. Adicionalmente, la matriz $M(\boldsymbol{q})$ no puede ser recuperada directamente; se debe calcular los valores de cada columna al igualar $\ddot{\boldsymbol{q}}$ a una columna i de la matriz identidad. Ese arreglo es representado por $\boldsymbol{e_i}$.

$$
\begin{aligned}
M(\boldsymbol{q})\cancel{\ddot{\boldsymbol{q}}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\cancel{\dot{\boldsymbol{q}}} + g(\boldsymbol{q}) &= \boldsymbol{\tau} \\
M(\boldsymbol{q})\cancel{\ddot{\boldsymbol{q}}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} &= \boldsymbol{\tau} \; - \; g(\boldsymbol{q}) \\
M(\boldsymbol{q})\boldsymbol{e_i} \; + \; g(\boldsymbol{q}) \; &= \; \boldsymbol{m_i} \; + \; g(\boldsymbol{q}) \; = \; \boldsymbol{\tau}
\end{aligned}
$$

A partir de ahora, la ecuación 10 será reescrita como 11 para simplificar el vector de gravedad y Coriolis como una sola cantidad denominada vector de efectos no lineales. Así mismo, el torque resultante sera igualado a una ley de control en vista que se considera la inexistencia de torques externos o de fricción.

$$
M(\boldsymbol{q})\ddot{\boldsymbol{q}} + b(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} = \boldsymbol{u} \tag{11}
$$

Con esta nueva estructura, se abre la posibilidad de utilizar otros métodos para la obtención de M y b. RBDL tiene implementado nativamente el algoritmo Composite Rigid Body Algorithm (CRBA) para calcular M y NonlinearEffects() para b. Son funciones optimizadas y que dan mismos resultados que InverseDynamics. Por lo tanto, esas funciones serán empleadas dentro de este trabajo para el cálculo de los arrays en cada configuración.

A manera de ejemplo, se calcularán los arrays M,b,τ para la configuración $\boldsymbol{q}$ = {-1.24, 1.46, 1.62, -0.24, 0.03, -0.40, -3.74} con velocidades articulares $\dot{\boldsymbol{q}}$ = {0.5, 0.5, -0.5, -0.5, 0.5, -0.5, 0.5} y aceleraciones articulares nulas. Los resultados fueron:

$$
M = \begin{bmatrix}
1.295 & 0.009 & -0.001 & -0.082 & -0.021 & -0.042 & 0.01 \\
0.009 & 2.732 & -1.373 & 0.011 & -2.181 & -0.217 & 0.005 \\
-0.001 & -1.373 & 1.113 & -0.011 & 0.592 & 0.1750 & -0.004 \\
-0.082 & 0.011 & -0.011 & 0.017 & 0 & 0.002 & -0.004 \\
-0.021 & -2.181 & 0.592 & 0 & 4.7 & -0.136 & -0.004 \\
-0.042 & -0.217 & 0.1750 & 0.002 & 0.136 & 0.049 & -0.002 \\
0.01 & 0.005 & -0.004 & -0.004 & -0.004 & -0.002 & 0.003
\end{bmatrix}
$$

$$
b = \begin{bmatrix} 0.479 \\ 32.068 \\ -27.058 \\ 0.189 \\ -10.288 \\ -3.202 \\ 0.067 \end{bmatrix}, \; \tau = \begin{bmatrix} 0.479 \\ 32.068 \\ -27.058 \\ 0.189 \\ -10.288 \\ -3.202 \\ 0.067 \end{bmatrix}
$$

Algunas observaciones interesantes se puede obtener de los arrays generados para comprobar que 10 se cumple. La primera es que los elementos 1,2 y 5 de la diagonal son los mayores entre las filas y columnas respectivas, lo cual indica que los terminos de inercia cruzados no tienen tanto impacto en la dinámica del robot. No obstante, los elementos restantes tienen valores fuera de la diagonal que son mayores. En terminos generales, esta matriz de inercia tiene comportamiento cruzado y no cruzado. La segunda observación es que el vector b y τ tienen misma magnitud como consecuencia que el vector de aceleraciones articulares es nulo y cancela la matriz M.

B. Control dinámico articular

Este tipo de control busca aprovechar las propiedades dinámicas del robot para generar un torque que cancele ciertas propiedades no lineales mientras impone dinámicas deseadas que permitan seguir una referencia articular. En otras palabras, se busca linealizar el sistema a partir de una ley de control elegida inteligentemente [11]. Se espera que como consecuencia, se logre un seguimiento de la referencia independiente de la posición inicial y con una alta velocidad de convergencia, superando tanto a la cinemática inversa numérica y al control cinemático. Un posible control (torque aplicado) que puede conseguir este objetivo se muestra en la ecuación 12.

$$
\boldsymbol{u} = M(\boldsymbol{q})\boldsymbol{y} + b(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{12}
$$

Al aplicar este torque, se cancela el término de los efectos no lineales. Esto impone que la ley de control requiera saber valores precisos de $\boldsymbol{q}$ y $\dot{\boldsymbol{q}}$ para estimación. Un mal modelamiento del robot llevara a un control pobre siguiendo la ley mencionada por la falta de precisión. Por otro lado, la matriz M está multiplicada por un vector $\boldsymbol{y}$ que se encarga de imponer las dinámicas deseadas para el seguimiento de referencias. Considerando los resultados prometedores del control PD para motores y otros sistemas [11] [12], el vector de dinámicas $\boldsymbol{y}$ puede incluir la aceleración articular deseada, un término proporcional del error articular y uno derivativo. En vista que el sistema es de múltiples entradas, las ganancias K_d y K_p son matrices de dimensiones $\boldsymbol{n}$ x $\boldsymbol{n}$ donde $\boldsymbol{n}$ es la cantidad de DoF. Al igualar los vectores $\ddot{\boldsymbol{q}}$ y y, se consigue establecer un error articular con dinámicas de segundo orden.

Todo este desarrollo se puede contemplar en la lista de ecuaciones a continuación.

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + b(\boldsymbol{q}, \dot{\boldsymbol{q}}) = M(\boldsymbol{q})\boldsymbol{y} + b(\boldsymbol{q}, \dot{\boldsymbol{q}})$$
$$\ddot{\boldsymbol{q}} = \boldsymbol{y}$$
$$\boldsymbol{y} = \ddot{\boldsymbol{q}}_{\boldsymbol{d}} + K_d\,(\dot{\boldsymbol{q}}_{\boldsymbol{d}} - \dot{\boldsymbol{q}}) + K_p(\boldsymbol{q}_{\boldsymbol{d}} - \boldsymbol{q})$$
$$\ddot{\boldsymbol{q}}_{\boldsymbol{d}} - \boldsymbol{q} + K_d\,(\dot{\boldsymbol{q}}_{\boldsymbol{d}} - \dot{\boldsymbol{q}}) + K_p(\boldsymbol{q}_{\boldsymbol{d}} - \boldsymbol{q}) = 0$$
$$\ddot{\boldsymbol{e}} + K_d\dot{\boldsymbol{e}} + K_p\boldsymbol{e} = 0$$

Para conseguir que el error tenga un comportamiento criticamente amortiguado, fueron elegidas las ganancias K_p y K_d a partir de frecuencias naturales de cada variable articular en matrices diagonales. Ambas formas se muestran en la ecuación 13.

$$K_p = \begin{bmatrix} w_1^2 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & w_7^2 \end{bmatrix} \quad K_d = \begin{bmatrix} 2w_1 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & 2w_7 \end{bmatrix} \tag{13}$$

Al final, reescribiendo la ecuación 12 con el vector y elegido, la ley de control dinamico articular PD es igual a:

$$\boldsymbol{u} = M(\boldsymbol{q})\,(\,\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + K_d\,(\dot{\boldsymbol{q}}_{\boldsymbol{d}} - \dot{\boldsymbol{q}}) + K_p(\boldsymbol{q}_{\boldsymbol{d}} - \boldsymbol{q}))\; + b(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{14}$$

utilizando un error articular igual a :

$$\boldsymbol{e} = \boldsymbol{q}_{\boldsymbol{d}} - \boldsymbol{q} \tag{15}$$

La implementación de esta ley de control se puede resumir en el diagrama de control en la figura 28 donde se muestran todos los bloques necesarios, entradas y salidas.

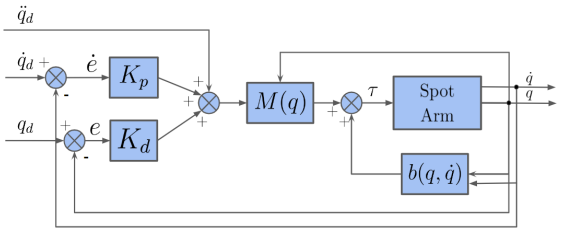


Fig. 28: Diagrama Control dinámico PD Articular

A manera de comprobación de la formulacion anterior, se hicieron testeos del control en ROS con la libreria RBDL para las 3 configuraciones mencionadas en los apartados anteriores a partir de la configuración articular nula y visualizados en Rviz. Se tuvieron como criterios de control el error en estado estacionario (e_{ss}), porcentaje de sobreimpulso (%OS) y tiempo de establecimiento. En fig. 29 se puede comprobar que Spot Arm logra llegar a la configuración deseada representada por una bola verde donde la bola roja es la configuración actual para el grupo articular 1.

Los resultados de las tres configuraciones son mostrados en las figuras 30,31,32. Las pruebas se ejecutaron hasta que la norma del error fuera menor a 0.0001 asegurando que todos

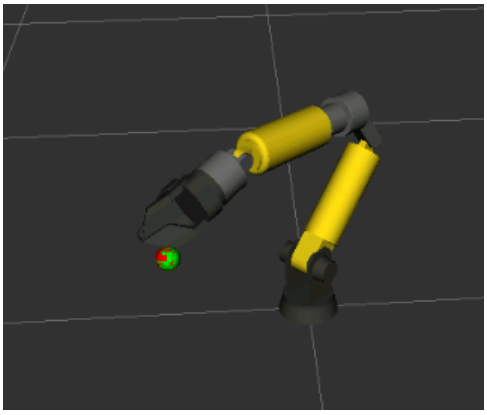


Fig. 29: Robot Spot Arm en configuración 1 para Control PD articular

los resultados mostraran seguimiento a la referencia. De esta forma, se descartaban rápidamente ganancias que no daban el resultado esperado al mantener el programa corriendo sin llegar a converger el error. Después de varias pruebas, las ganancias que se seleccionaron fueron:

$$K_p = \; diag(0.5,\; 0.5,\; 0.5,\; 0.5,\; 0.5,\; 0.5,\; 0.5)$$
$$K_d \;=\; 1.414\; K_p$$

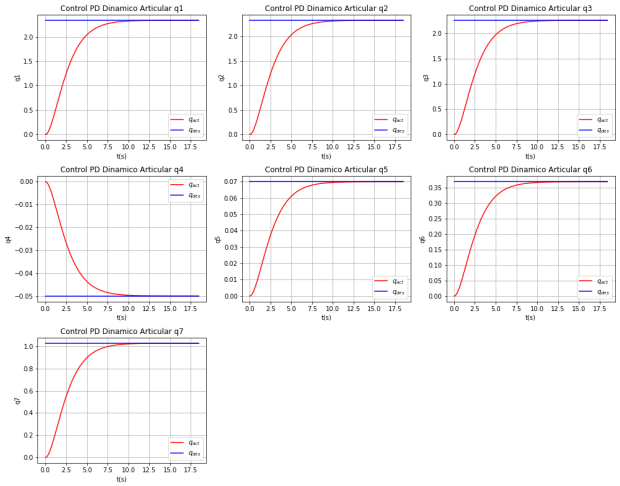


Fig. 30: Control PD Articular Config.1

Analizando los resultados, se afirma que el control PD articular ha conseguido que las variables $\boldsymbol{q}$ sigan a $\boldsymbol{q_d}$ con un comportamiento críticamente amortiguado. El rendimiento del sistema muestra un error en estado estacionario igual o cercano a 0, inexistencia de overshoot y un rápido tiempo de establecimiento, en promedio, cercano a los 10 segundos. Estos resultados son consistentes para las tres configuraciones ingresadas. Por lo tanto, se asegura el control PD articular forzará un buen seguimiento de la referencia inclusive si esta es amplificada o si su signo se invierte.

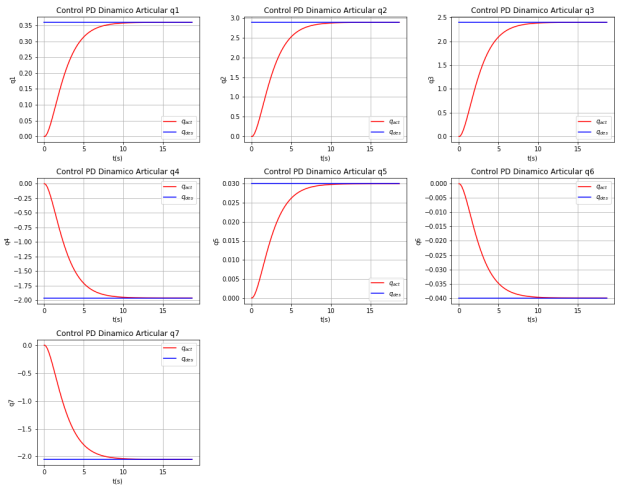


Fig. 31: Control PD Articular Config.2

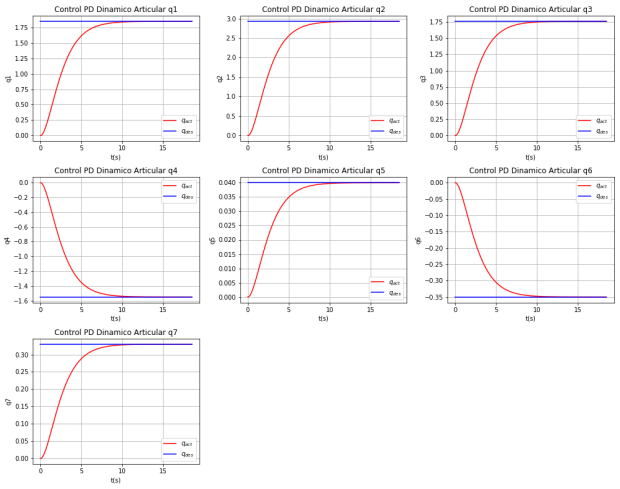


Fig. 32: Control PD Articular Config.3

C. *Control dinámico operacional*

En [10] se hace un desarrollo extensivo de como se expande la formulación dinámica articular para poder controlar el espacio operacional. Este objetivo se consigue mediante el uso del jacobiano analítico multiplicando al vector de dinámica deseada y y de su derivada multiplicando a $\dot{\boldsymbol{q}}$. Con ello se logra un mapeo completo entre los dos espacios sin la necesidad de aplicar cinemática inversa de manera explícita. En consecuencia, esta metodología de control no consume tantos recursos.

Su formulación es encuentra en la siguiente lista de ecuaciones:

$$\ddot{\boldsymbol{q}} = \boldsymbol{y}$$
$$J_a\ddot{\boldsymbol{q}} = J_a\boldsymbol{y}$$
$$\ddot{\boldsymbol{x}} - \dot{J}_a\dot{\boldsymbol{q}} = J_a\boldsymbol{y}$$
$$\boldsymbol{y} = J_a^{-1}\left(\ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \dot{J}_a\dot{\boldsymbol{q}} + K_d\dot{\boldsymbol{e}} + K_p\boldsymbol{e}\right)$$

donde se puede reformular el error de la ecuación 15 a 16 utilizando las variables operacionales.

$$\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x} \tag{16}$$

Finalmente, la ley de control obtenida es igual a:

$$u = M(\boldsymbol{q})J_a^{-1}\left(\ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \dot{J}_a\dot{\boldsymbol{q}} + K_d\dot{\boldsymbol{e}} + K_p\boldsymbol{e}\right) + b(\boldsymbol{q},\dot{\boldsymbol{q}}) \tag{17}$$

la cual puede repretarse graficamente con el diagrama de control de la fig. 33

De forma similar al control articular, fueron realizados tests en ROS-Rviz con las posiciones de las configuraciones deseadas 1,2,3 mediante cinemática directa para analizar los parámetros de desempeño de control generados, utilizando como referencia solo las posiciones cartesianas. Los resultados son mostrados en las figuras 34, 35, 36. Las simulaciones solo fueron ejecutadas hasta que la norma del error fuera menor a 0.0001. Así mismo, a partir de la consigna que se controlan posiciones cartesianas, las dimensiones de las ganancias deberán ser 3 x 3. Las ganancias ajustadas obtenidas fueron:

$$K_p = diag(100,\ 100,\ 100)$$
$$K_d = diag(20,\ 20,\ 20)$$

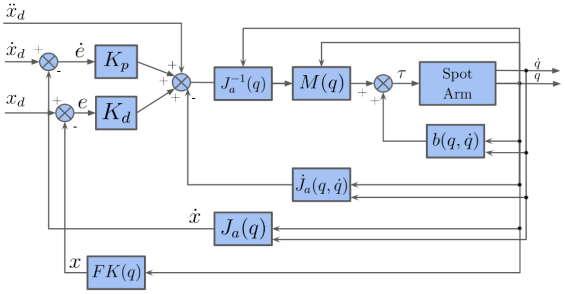


Fig. 33: Diagrama Control dinámico PD operacional

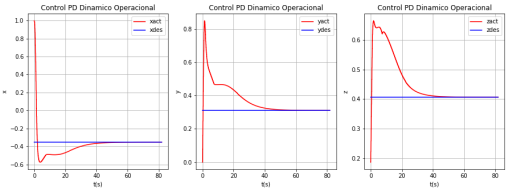


Fig. 34: Control PD Operacional Config.1

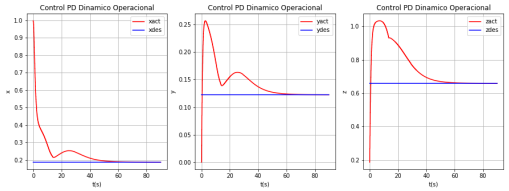


Fig. 35: Control PD Operacional Config.2

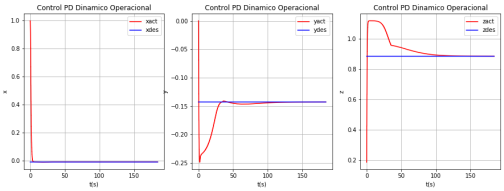


Fig. 36: Control PD Operacional Config.3

Para este control se tiene comportamientos diferentes en cada configuración con respecto al seguimiento de la referencia para x,y y z. El control de la configuración 1 34 presenta sobreimpulso en las tres coordenadas, acentuándose más en y. El seguimiento a la referencia que le sigue no es muy suave para x y y, aun así, se llega a un error de estado estacionario cercano a cero. El tiempo de establecimiento para las tres coordenadas se encuentra en promedio a los 40 segundos. El control de la configuración 2 35 no tiene sobreimpulso en la coordenada x y tiene un tiempo de establecimiento mayor (60s). Así mismo, su seguimiento a la referencia es más suave luego de su pico máximo y tambien logra seguir a la referencia en el estado estacionario. Finalmente, el control de la configuración 3 36 tiene un comportamiento perfecto (cero error y estabilización rápida) para la coordenada x. No obstante, y y z siguen presentando overshoot, curvas no muy suaves y inclusive un aumento considerable del tiempo de estabilización frente a las demás configuraciones.

A grandes rasgos, lo que se apreció en la visualización en RViz y que se refleja en 34, 35, 36, fue un end effector desplazándose rápidamente hasta una posición cercana a la deseada y luego se reacomodaba a menor velocidad buscando reducir el error operacional. Ello se manifiesta con un alto overshoot inicial y un ajuste lento no muy suave de las articulaciones hasta seguir la referencia. Aun con diferentes valores de ganancias K_p y K_d, este comportamiento se mantenía. Estas mismas características fueron visualizadas en los resultados del control cinemático. Adicionalmente, algunas articulaciones no se movían durante todo el control operacional como la articulación prismática. Es un posible que se necesite adicionar restricciones al control de posición, sea dinámico o cinemático, para considerar configuraciones más óptimas ante la redundancia y aprovechar todas las articulaciones en simultaneo durante el control.

Para una implementación real, se vería más factible utilizar el control PD articular con cinemática inversa para un manejo del espacio operacional, sobretodo por el menor coste computacional y los resultados obtenidos. De todas maneras, la tarjeta Jetson ofrecería suficiente procesamiento para un control articular o operacional.

VII. Conclusiones y recomendaciones

El presente trabajo ha conseguido rediseñar el Spot Arm a partir de la adición de una articulación prismática, actualizar los parámetros DH y generar un modelo con enmallado en Fusion 360 para exportar como URDF. Las distintas pruebas referentes a la cinemática y el control del robot lograron cumplir los siguientes puntos:

- Fue comprobado el modelo del robot y los parámetros DH para la cinemática directa con el uso de dos markers representando la posición esperada y la actual en 3 configuraciones distintas.
- Fueron probados dos algoritmos de cinemática inversa: Newton-Rhapson y descenso de gradiente. Ambos requirieron una configuración articular no nula para llegar a las coordenadas deseadas en vista que la posición inicial influencia el resultado final. El algoritmo Newton-Rhapson muestra una convergencia más rápida, como se estipula en la teoría, a coste de la posibilidad de llevar el robot a singularidades, problema que no se presenta en descenso de gradiente.
- Para el control cinemático, se generó el modelo cinemático del robot mediante el calculo numérico del Jacobiano analítico y de su pseudo inversa amortiguada para manejar casos de singularidades. Utilizando un error proporcional, se logró seguir las referencias de 3 configuraciones diferentes aunque con un overshoot considerable al inicio del movimiento.
- La librería RBDL permitió generar el modelo dinámico del robot con el algoritmo recursivo de Newton-Euler donde se calculan velocidades, aceleraciones y torques. El modelo es descrito por la matriz de inercia, vector de efectos no lineales y vector de torques; donde cada arreglo fue comprobado con una configuración de prueba.
- El modelo dinámico fue aprovechado para la formulación de dos esquemas de control PD, uno articular y otro operacional. Ambos utilizaron matrices de ganancia para conseguir un comportamiento críticamente amortiguado. Los resultados obtenidos para 3 configuraciones mostraron que el control PD articular lograba seguir las referencias con una rápida convergencia y sin sobreimpulso, mientras que el control PD operacional demoraba más en converger y tiene un sobreimpulso considerable al inicio de manera similar el control cinemático.

Luego de haber concluido el desarrolló de este proyecto, se presenta la posibilidad de generar varias sugerencias para los distintos apartados:

- Un modelo enmallado en Inventor o Fusion 360 permite analizar cada link en detalle para hacer correcciones en los parámetros DH de forma más sencilla.
- En caso la prueba de cinemática directa con markers falle, se puede generar markers en cada articulación para determinar que transformación homogénea está errónea.
- Establecer límites articulares para evitar llevar el robot a singularidades para cinemática inversa, control cinemático y control dinámico. Así mismo, establecer un criterio de parada coherente para las pruebas que utilicen algoritmos en bucle.
- Probar la cinemática inversa con varias configuraciones buscando los mejores resultados posibles.
- Para la sintonización de ganancias en el control dinámico, empezar modificando elemento por elemento y observar el efecto sobre el error articular. En ocasiones, modificar una ganancia tendrá efecto sobre dos o más articulaciones producto del acoplamiento de la matriz de inercia. Tener

cuidado con estos casos para no llevar una articulación a singularidad accidentalmente.
- Generar un control más estricto para el espacio operacional que considere minimización sobre las configuraciones redundantes y que aproveche la articulación prismática con el objetivo de reducir el tiempo de establecimiento y el overshoot inicial.

APPENDIX A
SCRIPTS UTILIZADOS

A continuación, se presentan los scripts más importantes utilizados dentro de este trabajo. Fueron omitidas algunas lineas de código para resaltar la característica principal de cada script. Cada programa puede ser observado con más detalle en el repositorio del proyecto https://github.com/MarceloContreras/FR_proyecto.

test_fkine Cinemática directa

```
# Joint names
jnames = ['Rev7', 'Rev8', 'Rev9','Rev10',
'Slider11', 'Rev14', 'Rev15']

# Joint Configuration
q = [0, 0, 0, 0, 0, 0 , 0]

# End effector with respect to the base
T = fkine_ur5(q) # Direct kinematics
print( np.round(T, 3) )
bmarker.position(T)

# Object (message) whose type is JointState
jstate = JointState()
# Set values to the message
jstate.header.stamp = rospy.Time.now()
jstate.name = jnames
# Add the head joint value (with value 0) to the joints
jstate.position = q

# Continuous execution loop
while not rospy.is_shutdown():
    # Current time (needed for ROS)
    jstate.header.stamp = rospy.Time.now()
    # Publish the message
    # pub.publish(jstate)
    bmarker.publish()
    # Wait for the next iteration
    rate.sleep()
```

test_ikine Cinemática inversa

```
# Desired position
xd = np.array([-0.011, -0.143, 0.886])
# Initial configuration
q0 = np.array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
# Inverse kinematics
q = ikine_Spot(xd, q0)

# Joint limits
if (q[0] < -2.62 or q[0] > 3.14 or
        q[1] < -0.52 or q[1] > 3.14 or
        q[2] < 0 or q[2] > 2.62 or
        q[3] < -4.36 or q[3] > 1.22 or
        q[4] < 0 or q[4] > 0.07 or
        q[5] < -1.05 or q[5] > 1.05 or
        q[6] < -4.45 or q[6] > 1.31):
    q = q0
    print("Punto deseado fuera de alcance.")

# Resulting position (end effector with respect
# to the base link)
T = fkine_Spot(q)

# Messaje JointState
jstate = JointState()
# Adding q
jstate.position = q

# Continuous execution loop
while not rospy.is_shutdown():
    # Publish the message
    pub.publish(jstate)
    rate.sleep()
```

test_diffkine Control cinemático

```
# Posicion deseada

# Caso 1
qd = np.array([2.18, 2.86, 1.93, -2.77,
0.04, 0.8, 0.38]); k = 12
# Caso 2
#qd = np.array([0.36, 2.89, 2.40, -1.96,
0.03, -0.04, -2.05]);k= 5
# Caso 3
#qd = np.array([1.85, 2.93, 1.76, -1.55,
0.04, -0.35, 0.33]); k = 10

# Se define la posicion deseada en base a
# qd usando la cinematica
# directa dependiendo del caso

T = fkine_Spot(qd)
xd = T[0:3,3]

# Initial configuration
q0  = np.array([0, 0, 0, 0, 0.000001, 0, 0])

# Resulting initial position
T = fkine_Spot(q0)
x0 = T[0:3,3]

epsilon = 0.0001
count = 0

u_lim = np.array([3.14159, 3.14159, 2.61799,
1.22173, 0.075, 1.047198, 1.30899])
l_lim = np.array([-2.61799, -0.523599, 0,
-4.3633, 0, -1.047198, -4.45059])

# Frequency (in Hz) and control period
freq = 200
dt = 1.0/freq
rate = rospy.Rate(freq)
t = 0
```

```
# Initial joint configuration
q = copy(q0)
# Main loop
while not rospy.is_shutdown():
    # Kinematic control law for position
    # Calculo del jacobiano y posicion actual
    J = jacobian_Spot(q, 0.0001)
    x = fkine_Spot(q)
    x = x[0:3, 3]

    # Calculo del error
    e = x - xd

    # Condicion para detener el algoritmo
    if(np.linalg.norm(e) < epsilon):
        print('Desired point reached')
        print(e)
        print(x)
        print('iteraciones', count)
        break

    # Ley de control
    de = -k*e

    # Comprobacion del rango del Jacobiano
    rank_J = np.linalg.matrix_rank(J)

    # Si el rango es menor a 3 se usa la pseudo-inversa
    # amortiguada
    if (rank_J < 3):
        dq = J.T.dot(np.linalg.inv(J.dot(J.T)
        +0.01*np.eye((3)))).dot(de)
        print(rank_J)
        print(J)
    else:
        dq = np.linalg.pinv(J).dot(de)

    # Se actualiza el q potencialmente
    q_pot = q + dt*dq

    # Si una articulacion sale de sus limites no se actualiza
    for i in range(7):
        if ( (q_pot[i] < u_lim[i]) and
        (q_pot[i] > l_lim[i]) ):
            q[i] = q_pot[i]

    # Se detiene el algoritmo despues de
    # 10000 iteraciones si
    # no se llego al punto deseado
    count = count + 1
    if(count > 10000):
        print('Max number of iterations reached')
        break

    t = t + dt

    # Wait for the next iteration
    rate.sleep()
```

control_dinInv.py Control dinámico operacional

```
#  =============Grupo 1============
#  Configuracion articular inicial (en radianes)
q0 = np.array([0., 0., 0., 0., 0., 0., 0.])
# Velocidad inicial
dq0 = np.array([0., 0., 0., 0., 0., 0., 0.])
# Configuracion articular deseada
qdes = np.array([2.34, 2.33, 2.26, -0.05, 0.07,
0.37, 1.03])
dqdes = np.array([0., 0., 0., 0., 0., 0., 0.])
ddqdes = np.array([0., 0., 0., 0., 0., 0., 0.])
# ================================

# Posicion resultante de la configuracion articular
# deseada
xdes = fkine_Spot(qdes)[0:3, 3]
dxdes = np.array([0, 0, 0]) # Vel deseada
ddxdes = np.array([0, 0, 0]) # Acc deseada

jstate.position = q0
pub.publish(jstate)

# Modelo RBDL
modelo = rbdl.loadModel('../urdf/Spot.urdf')
ndof = modelo.q_size     # Grados de libertad

# Se definen las ganancias del controlador
valores = 100*np.array([1.0, 1.0, 1.0])
Kp = np.diag(valores)
Kd = 2*np.sqrt(Kp)

# Limites
q_max = [3.14159, 3.14159, 2.61799, 1.22173,
0.075, 1.047198, 1.30899]
q_min = [-2.61799, -0.523599, 0.0, -4.3633,
0.0, -1.047198, -4.45059]

while not rospy.is_shutdown():

    # Leer valores del simulador
    rbdl.CompositeRigidBodyAlgorithm(modelo, q, M)
    rbdl.NonlinearEffects(modelo, q, dq, b)

    # Posicion actual
    x= fkine_Spot(q)[0:3, 3]

    # Jacobiano, inversa y derivada
    J = jacobian_Spot(q)
    invJ = np.linalg.pinv(J)
    dJ = (J - J_1)/dt
    dx = J.dot(dq)

    # Error
    de = dxdes - dx
    e = xdes - x

    # Ley de control
    u = M.dot(invJ).dot(ddxdes - dJ.dot(dq) +
                        Kd.dot(de) + Kp.dot(e)) + b

    # Actualizacion
    ddq = np.linalg.inv(M).dot(u-b)
    qprev = q + dt*dq
    dq = dq + dt*ddq

    # Comprobacion de limites articulares
    for j in range(ndof):
        if(qprev[j] > q_min[j] and qprev[j] < q_max[j]):
```

```
            q[j] = qprev[j]

    # Valor articular final
    print(np.round(q, 4))

    # Criterio de parada
    if(np.linalg.norm(xdes - x) < 1e-4):
        break

    # Publicacion del mensaje
    jstate.position = q
    pub.publish(jstate)
    t = t+dt
    # Esperar hasta la siguiente  iteracion
    rate.sleep()
```

control_dinInv2.py Control dinámico articular

```
# #=========Grupo 1==========
# Configuracion articular inicial (en radianes)
q0 = np.array([0., 0., 0., 0., 0., 0., 0.])
# Velocidad inicial
dq0 = np.array([0., 0., 0., 0., 0., 0., 0.])
# Configuracion articular deseada
qdes = np.array([2.34, 2.33, 2.26, -0.05,
0.07, 0.37, 1.03])
dqdes = np.array([0., 0., 0., 0., 0., 0., 0.])
ddqdes = np.array([0., 0., 0., 0., 0., 0., 0.])
# #==========================

# Se definen las ganancias del controlador
valores = 0.5*np.array([1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0])
Kp = np.diag(valores)
Kd = 2*np.sqrt(Kp)

# Limites
q_max = [3.14159, 3.14159, 2.61799, 1.22173,
0.075, 1.047198, 1.30899]
q_min = [-2.61799, -0.523599, 0.0, -4.3633,
0.0, -1.047198, -4.45059]

while not rospy.is_shutdown():

    # Leer valores del simulador
    rbdl.CompositeRigidBodyAlgorithm(modelo, q, M)
    rbdl.NonlinearEffects(modelo, q, dq, b)

    # Posicion actual
    x = fkine_Spot(q)[0:3, 3]

    # Error
    e = qdes - q
    de = dqdes - dq

    # Ley de control
    u = M.dot(ddqdes + Kd.dot(de) + Kp.dot(e)) + b

    # Actualizacion
    ddq = np.linalg.inv(M).dot(u-b)
    qprev = q + dt*dq
    dq = dq + dt*ddq

    # Comprobacion de limites articulares
    for j in range(ndof):
        if(qprev[j] > q_min[j] and qprev[j] < q_max[j]):
            q[j] = qprev[j]

    print(np.round(q, 4))

    # Criterio de parada
    if(np.linalg.norm(qdes - q) < 1e-4):
        break

    # Publicacion del mensaje
    jstate.position = q
    t = t+dt
    # Esperar hasta la siguiente iteracion
    rate.sleep()
```

REFERENCES

[1] Boston Dynamics lanza una nueva versión de spot, 02 2021.

[2] Debolina Biswas. All You Need To Know About Boston Dynamics' Spot, 12 2021.

[3] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018.

[4] Boston Dynamics. What's New in Spot — Boston Dynamics, 05 2022.

[5] Ben Dickson. Boston Dynamics' Spot robot is securing its position in a niche market, 09 2021.

[6] Martin L. Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, pages 1–17, 2016.

[7] Erico Guizzo. How Boston Dynamics Is Redefining Robot Agility, 08 2021.

[8] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C. Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, Amir Melzer, and Mark Hoepflinger. Anymal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016.

[9] Toshinori Kitamura. Fusion2urdf. https://github.com/syuntoku14/fusion2urdf, 2020.

[10] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

[11] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control (Advanced Textbooks in Control and Signal Processing)*. Springer, 1st ed. 2009 edition, 2008.

[12] Mark Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 1 edition, 2005.