UNIVERSIDAD DE
COSTA RICA

Facultad de Ingeniería
Escuela de Ingeniería Mecánica

Diseño y simulación de un sistema de control de cuerpo completo para un robot humanoide con una base omnidireccional, un torso móvil y un brazo

Trabajo final de graduación sometido a la consideración de la

Universidad de Costa Rica

como parte de los requisitos
para aspirar al título y grado de

Licenciatura en Ingeniería Mecánica

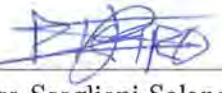
Marco Andrés Madrigal Quesada

Ciudad Universitaria Rodrigo Facio
Abril de 2021

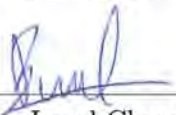
.

Hoja de tribunal

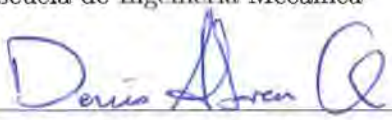
Este proyecto de graduación fue aceptado por la Comisión de Trabajos Finales de Graduación de la Escuela de Ingeniería Mecánica de la Universidad de Costa Rica, como requisito parcial para optar por el grado y título de Licenciatura en Ingeniería Mecánica.



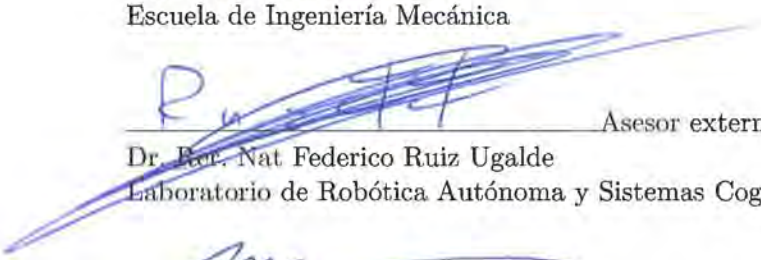
Director de la Unidad Académica
Dr. Pietro Scaglioni Solano
Director Escuela de Ingeniería mecánica



Asesor director
Ing. Israel Chaves Arbaiza
Escuela de Ingeniería Mecánica



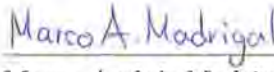
Asesor interno
Ing. Denis Abarca Quesada
Escuela de Ingeniería Mecánica



Asesor externo
Dr. Rer. Nat Federico Ruiz Ugalde
Laboratorio de Robótica Autónoma y Sistemas Cognitivos, INII

Docente curso Proyecto II
Mag. Marco Vinicio Calvo Vargas
Escuela de Ingenieria Mecánica

Por acuerdo unánime del tribunal examinador de este trabajo final de graduación, se aprueba con distinción de *sobresaliente*, al amparo de lo establecido en el Artículo 39 del *Reglamento de Trabajos Finales de Graduación.*



Ponente
Marco Andrés Madrigal Quesada

Agradecimientos

Agradezco a Dios por mostrarme tanto de sí a lo largo de este trabajo y darme la gracia de realizarlo y disfrutarlo. A mi familia, por su apoyo y consejo, no solo constante, sino también generoso. A mis amigos y hermanos por su interés y oración.
Doy las gracias también a los asesores del proyecto cuyas directrices fueron indispensables para el logro alcanzado. También al ingeniero Daniel García Vaglio, quién atendió con alegría a muchas de mis dudas y problemas.

Dedicatoria

A mis padres y hermanos. Cada uno es, con la particular y amorosa disposición de su corazón, un ejemplo de esfuerzo.

Epígrafe

¿No saben que en una carrera todos los corredores compiten, pero solo uno obtiene el premio? Corran, pues, de tal modo que lo obtengan.
Todos los deportistas se entrenan con mucha disciplina. Ellos lo hacen para obtener un premio que se echa a perder; nosotros, en cambio, por uno que dura para siempre.

1 Corintios 9:24-25

Índice general

Hoja de tribunal . . . ii
Agradecimientos . . . iii
Dedicatoria . . . iv
Epígrafe . . . v
Índice de ilustraciones . . . xi
Simbología . . . xii
Resumen . . . xv

1. Introducción **1**
1.1. Descripción general . . . 1
1.2. Objetivos . . . 2
1.2.1. Objetivo general . . . 2
1.2.2. Objetivos específicos . . . 2
1.3. Justificación . . . 2
1.4. Antecedentes . . . 3
1.5. Metodología . . . 3
1.6. Alcance y limitaciones . . . 4

2. Marco teórico **6**
2.1. Robots humanoides . . . 6
2.1.1. Manipulación móvil . . . 6
2.1.2. Robots de cuerpo entero para la manipulación . . . 7
2.1.3. El robot humanoide del ARCOS-Lab . . . 11
2.2. Fundamentos mecánicos . . . 12
2.2.1. Conceptos básicos . . . 12
2.2.2. Cinemática de cuerpo rígido . . . 13
2.2.3. Cinemática diferencial . . . 19
2.3. Control de manipuladores móviles . . . 21
2.3.1. Teoría de control de manipuladores . . . 21
2.3.2. Control cinemático . . . 24
2.3.3. Control dinámico . . . 26
2.3.4. Orocos-KDL . . . 27
2.3.5. Evasión de obstáculos . . . 28
2.4. Software del Robot Humanoide . . . 31
2.4.1. Arquitectura de software . . . 31
2.4.2. Modelos de objetos . . . 32
2.4.3. Sistema de control . . . 33

2.4.4. Simulador del robot humanoide del ARCOS-Lab . . . 35
2.5. Hardware del Robot Humanoide . . . 36
2.5.1. Cuerpo del robot humanoide . . . 36
2.5.2. Plataforma omnidireccional . . . 36
2.5.3. Torso móvil . . . 37
2.5.4. Brazos . . . 37

3. Estudio del sistema de control del ARCOS-Bot 38
3.1. Instalación del simulador . . . 38
3.2. Generalidades . . . 38
3.3. Uso del *vector fields* . . . 39
3.4. Modelos de predicción . . . 39
3.5. Cinemática inversa de velocidades . . . 41
3.5.1. Uso de la matriz jacobiana en la cinemática inversa . . . 41
3.5.2. Cómo lidiar con las singularidades . . . 42
3.6. Manejo de la redundancia . . . 42
3.6.1. *Weighted Damped Least Squares* . . . 43
3.6.2. Proyecciones en el espacio nulo . . . 43
3.7. Implementación . . . 44
3.8. Uso de la biblioteca OROCOS-KDL . . . 46
3.8.1. Cadenas cinemáticas . . . 46
3.8.2. Cinemática inversa de velocidades . . . 46
3.9. Cambios necesarios . . . 47

4. Diseño del sistema de control cinemático de cuerpo completo 49
4.1. Diseño del método de coordinación del cuerpo completo . . . 49
4.1.1. Organización del cuerpo completo . . . 49
4.1.2. Criterios que rigen la coordinación . . . 51
4.1.3. Sistemas de coordinación de cuerpo completo existentes . . . 54
4.1.4. Propuestas de sistemas de coordinación . . . 55
4.1.5. Selección del sistema de coordinación . . . 57
4.1.6. Selección de la organización del cuerpo completo . . . 58
4.2. Diseño del sistema de control de cuerpo completo . . . 58
4.2.1. Esquema general . . . 58
4.2.2. Modelos de elementos físicos . . . 59
4.2.3. Coordinación del cuerpo completo . . . 61

5. Implementación en el simulador del ARCOS-Lab 64
5.1. Aspectos generales . . . 64
5.2. Organización del código . . . 65
5.3. Determinar la mínima distancia con la mesa . . . 67
5.4. Ampliando la cadena cinemática . . . 70
5.5. Flujo de información . . . 71
5.6. Elementos para la simulación . . . 74
5.6.1. Figuras . . . 74
5.6.2. Indicadores de desviación . . . 75
5.7. Manejo de la redundancia . . . 76

5.8. Límites de las articulaciones 77
5.9. Pruebas de giro 77
5.9.1. Giro ante un mismo ángulo inicial de desviación 77
5.9.2. Giro ante ángulos de desviación grandes 80
5.10. Giro inicial 81
5.10.1. Relación con el control de cuerpo completo 81
5.10.2. Implementación dentro del ciclo de control 82
5.11. Pruebas finales 82
5.11.1. Escogencia de parámetros 82
5.11.2. Ejecución de la pruebas 86
5.12. Resultados y discusión 86
5.12.1. Postura inicial 86
5.12.2. Correlación entre parámetros 86
5.12.3. Lejanía de la mesa 90
5.12.4. Condiciones óptimas de trabajo 90

6. Conclusiones y recomendaciones **94**
6.1. Conclusiones 94
6.2. Recomendaciones 95

Bibliografía **100**

Anexos **101**
Anexo A.1. Sección de código modificada Robot_descriptions 101

Índice de figuras

2.1. Dos ejemplos de bases omnidireccionales: a la izquierda el uso de ruedas giratorias, a la derecha ruedas Mecanum 7
2.2. Robot Rollin' Justin 8
2.3. Robot TUM-Rosie acercándose a un objeto con el fin de manipularlo 9
2.4. Secuencia en la que Cody se acerca a una puerta y la abre utilizando su cuerpo entero 10
2.5. PR2 escribiendo en una pizarra trazos que requieren un movimiento preciso y suave de su efector final 10
2.6. Los tres escenarios para el robot del ARCOS-Lab 11
2.7. Un punto P representado en dos marcos de referencia distintos 14
2.8. Tipos de articulaciones 16
2.9. *joint space* y del *task space* 17
2.10. Esquemas de control de movimiento utilizando comandos de velocidad para el motor. En (a) controlando las variables cinemáticas internas y en (b) las externas 22
2.11. Esquemas de control de movimiento utilizando comandos de fuerza para el motor. 23
2.12. Esquema del control de fuerza mediante control de movimiento 24
2.13. Estrategias para la evasión de obstáculos 29
2.14. *Vector Fields* para cuerpos y para manipuladores 30
2.15. Ciclo de funcionamiento del sistema movimiento con evasión de obstáculos, VFCLIK 30
2.16. TUM-Rosie manipulando una caja mediante un modelo del objeto 32
2.17. Estructura del Sistema de Control del robot humanoide del ARCOS-Lab. . . . 33
2.18. Ciclo de control del OMS 34
2.19. Capturas de PYROVITO, la herramienta de visualización del simulador 35
2.20. El robot humanoide del ARCOS-Lab. 36

3.1. Campos vectoriales creados por el objetivo (en azul) y por un obstáculo (en rojo). Los vectores verdes, que representan la orientación actual de la mano y la dirección óptima de llegada a la meta, se utiliza para computar las velocidad angulares. 40
3.2. Modelo de predicción para las articulaciones de revolución 40
3.3. Composición de un elemento tipo *segment* de OROCOS-KDL 47

4.1. Modelo del robot completo como una sola cadena cinemática. En morado se ven las articulaciones relacionadas a la base. En rojo se visualizan los eslabones del resto del cuerpo. 51
4.2. Criterios a considerar cuando se quiere (a) rotar el robot y (b) desplazarlo . . . 54

4.3. El vector que une al robot con el objetivo se puede usar para calcular tanto la distancia, como el la proyección sobre el suelo del ángulo de desviación 56
4.4. Diagrama de bloque de la relación entre el nuevo sistema junto a VFCLIK (en el recuadro gris) y el módulo robot_descriptions 59
4.5. El modelo de la mesa es la línea recta sobre la que se encuentra su borde frontal, o bien, tangencial al punto más cercano. 60
4.6. El modelo del robot se compone de un marco de referencia para la base y otro para el final de la cadena, además de un radio de seguridad que representa el cuerpo del robot. 61
4.7. Principio de diseño del algoritmo para la coordinación del cuerpo completo del robot. A la izquierda se muestra una representación de la mesa y el robot. A la derecha, las funciones que definen los pesos de los elementos a coordinar . . 62

5.1. Flujo de la información entre los dos módulos que componen el sistema de coordinación 66
5.2. Elementos geométricos utilizados para modelar la mesa con respecto al objetivo. 68
5.3. Construcción geométrica que ejemplifica el cálculo para hallar la distancia mínima entre el cilindro de seguridad y el borde de la mesa en cuestión. El punto crítico es encuentra en la intersección de las rectas definidas por los vectores unitarios $\hat{\mathbf{n}}$ y $\hat{\mathbf{m}}$. En (a) se muestran los elementos vectoriales y en (b) una demostración gráfica para hallar la relación entre θ y α 69
5.4. Descripción cinemática del robot. Las líneas anaranjadas representan los nuevos grados de libertad, y las flechas amarillas los vectores que describen las uniones entre ellos. Entre paréntesis se describen sus dimensiones cartesianas y angulares (cuando las hay) 70
5.5. Descripción detallada de la información involucrada en el nuevo sistema de control 73
5.6. Captura de una simulación donde se pueden ver las figuras para ayudar a la comprobación visual: un cilindro gris que representa el radio de seguridad, junto a una flecha verde que indica el frente del robot, una caja amarilla para la mesa, conjuntos de flechas para los marcos de referencia de la *pose* del efector final y el objetivo; y una esfera blanca para el punto de más próximo entre la línea de la mesa y el cilindro. 75
5.7. Elementos geométricos utilizado para calcular los indicadores de desviación . . 76
5.8. Pruebas para determinar el uso de los tipos de movimiento. Se muestra un promedio adimensional del desplazamiento de cada tipo de articulación. En verde se muestra el desplazamiento en el plano XY; en azul la rotación, y en verde el brazo. Las pruebas corresponden a: (a) la pose final es equivalente a poner la palma de la mano hacia el cuerpo, con los dedos girados levemente hacia abajo; en (b) la *pose* del objetivo es muy similar a la inicial. 78
5.9. Si no se fomenta el giro del robot, se producen situaciones como la de esta imagen, donde el robot intenta pasa su brazo por encima de su cuerpo para alcanzar un objetivo que se encuentra detrás de sí. 79

5.10. Pruebas de acercamiento, con el objetivo en las coordenadas (2,-2) y el borde la mesa con una inclinación de 50°, a 40 cm del objeto. El peso de la rotación de la base se mantuvo en 1.0 a lo largo de toda la simulación. En (a) la pose final es equivalente a poner la palma de la mano hacia el cuerpo, con los dedos girados levemente hacia abajo; en (b) la *pose* del objetivo es muy similar a la inicial. 80
5.11. Las tres distintas *poses* que se tomaron como parámetro. Corresponden a los nombres de *asa*, *tomar* y *botón*, respectivamente. A la derecha puede verse cómo corresponden los ejes del marco de referencias al efector final del robot si fuese una mano antropomórfica. 83
5.12. El parámetro de distancia del objetivo al borde de la mesa toma los valores 15 cm y 45 cm 84
5.13. El parámetro de orientación de la mesa toma los valores -45°, -30°, 0°, 30° y 45°. Aquí se representan los positivos. Nótese como este parámetro no modifica la posición ni orientación del objetivo. 84
5.14. El parámetro de distancia hasta el objetivo toma los valores de 1,5 m; 2 m y 4 m. La medida se realiza desde el origen del robot y hasta el objetivo, la mesa se muestra para referencia, ya que su posición y orientación se definen mediante los parámetros anteriores. 85
5.15. Diagramas de correlación entre los parámetros de las pruebas. Un 1 indica éxito y un 0 fallo 88
5.16. Gráficas de desviación para una *pose* final tipo *asa*, con el objetivo a 15 cm del borde de la mesa y a cuatro metros de distancia. 89
5.17. Gráficas de uso de articulaciones para una *pose* final tipo *asa*, con el objetivo a 15 cm del borde de la mesa y a cuatro metros de distancia. 89
5.18. Diagramas de correlación en pruebas donde se modifica únicamente la distancia entre el objetivo y el borde de la mesa. Un 1 indica éxito y un 0 fallo 91
5.19. Diagramas de correlación en pruebas donde se modifica únicamente la el ángulo de inclinación de la mesa. Un 1 indica éxito y un 0 fallo 92
5.20. Resultado de las pruebas con los valores óptimos 93

Simbología

q Vector de la posición de las articulaciones.

x Representación del conjunto de las variables de posición y orientación del elemento final de la cadena cinemática, es decir su *pose*.

$\dot{a}$, $\ddot{a}$ Primera y segunda derivada temporal de la magnitud a.

e El error entre la *pose* actual y la deseada.

p Vector de posición cartesiana del elemento final de la cadena cinemática.

v Velocidad espacial o *twist*.

R Matriz rotacional

T Matriz de transformación homogénea

J Matriz Jacobiana

f_{ee} Vector de fuerzas en el efector final

τ Vector de momentos en las articulaciones

$A^{\dagger}$ Matriz pseudo inversa (inversa de Moore-Penrose) de A

A^{T} Matriz transpuesta de A

A^{*} Matriz transpuesta conjugada de A

α Ángulo de desviación del robot

L Distancia crítica entre el robot y la mesa

r Radio de seguridad del robot

R Distancia para activar el uso del brazo

θ Ángulo de orientación de la mesa

t Vector bidimensional que define la pose del borde de la mesa con respecto al objetivo.

b Vector bidimensional de la posición del objetivo

a Vector bidimensional de la posición del robot

w Vector con los pesos de la diagonal de la matriz de pesos

h Vector bidimensional de la posición del efector final

Glosario

ARCOS-Bot Robot humanoide que se está desarrollando en el ARCOS-Lab.

ARCOS-Lab Laboratorio de Robótica Autónoma y Sistema Cognitivos del Instituto de Investigaciones en Ingeniería y la Escuela de Ingeniería Eléctrica de la Universidad de Costa Rica.

efector final Elemento final del la cadena cinemática, en el caso de un brazo robótico puede ser o bien su mano, un dedo de la misma, o una herramienta.

holonómico Sistema móvil donde la posición y la velocidad son sistemas matemáticamente independientes.

Mecanum Un diseño popular de ruedas omnidireccionales.

Middleware Software que, en coordinación con el sistema operativo, le permite a los programas comunicarse entre sí o con el hardware con mayor facilidad.

odometría Método por el cual se determina la posición de un vehículo con respecto al desplazamiento registrado por sus ruedas o actuador equivalente.

Orocos-KDL Librería del proyecto OROCOS para modelar cadenas cinemáticas y dinámicas.

TUM-Rosie Robot humanoide desarrollado por la Universidad Técnica de Munich.

Siglas

ARCOSPYU *ARCOS-Lab Python Utilities* o Utilidades de Python del ARCOS-Lab.

DLR *Deutsches Zentrum für Luft- und Raumfahrt* o Agencia Aeroespacial Alemana.

LBR *Leichtbauroboter* o Robot Ligero, conocido también por sus siglas en inglés, LWR. Brazo robótico de KUKA.

OMS *Object Model System* o Sistema de Modelos de Objetos.

PYROVITO *Python YARP Robot Visualization Tool* o Herramienta en Python y YARP para la Visualización del Robot.

ROS Robot Operating System.

SVD Singular Value Decomposition.

VFCLIK *Vector Field Based Closed Loop Inverse Kinematics Controller* o Controlador de Cinemática Inversa de Ciclo Cerrada Basado en Campos Vectoriales.

VIK Velocity Inverse Kinematics.

YARP Yet Another Robot Platform.

Resumen

El robot humanoide del ARCOS-Lab, su sistema de control y su simulador cuentan actualmente con la capacidad de utilizar su brazo de siete grados de libertad. Este proyecto busca ampliar el control y la simulación para incluir el torso móvil y la base omnidireccional, para un total de once grados de libertad. Se busca además coordinarlos para que el robot sea capaz de utilizar todo su cuerpo para alcanzar objetos que se encuentran sobre una mesa, ya sea que requiera o no que este se desplace. Siempre con el cuidado de que el robot no colisione contra ninguna superficie.

El nuevo sistema de control de cuerpo completo unifica los movimientos de desplazamiento y rotación (base omnidireccional), con los de postura y manipulación (torso y brazo), al organizarlos en una sola cadena cinemática. Se saca provecho del método de cinemática inversa de mínimos cuadrados amortiguados y ponderados que provee OROCOS-KDL para resolver el sistema redundante, de forma que la solución resulte en movimientos seguros, que alejen al robot de singularidades externas y configuraciones indeseadas, siempre con el cuidado de evitar colisionar contra obstáculo. Esto se logra con el desarrollo de un nuevo módulo que modifica el peso con el que se considera cada una de las articulaciones de la cadena.

Su capacidad se demuestra mediante pruebas simuladas, donde se variaron cuatro distintos parámetros: el ángulo del borde frontal de la mesa, la distancia del robot hasta el objetivo, la distancia entre el objetivo y el borde de la mesa y la orientación del objetivo relativa a la orientación inicial del efector final del robot. Se evaluaron más de doscientas combinaciones con el fin de determinar los factores decisivos para la efectividad del sistema y encontrar así las condiciones óptimas de operación.

Capítulo 1

Introducción

En esta sección se describen los propósitos del proyecto y el estado actual del robot humanoide en el que este se basa. A su vez, se explican las razones que fundamentan su desarrollo, así como el alcance que tiene.

1.1. Descripción general

Con el presente trabajo se busca diseñar una alternativa con la cual el robot humanoide del ARCOS-Lab, que ya es capaz de manipular con su brazo de siete grados de libertad (7 DoF), pueda también llegar a objetos que están fuera de su alcance inmediato. Específicamente aquellos que se encuentren sobre una mesa. En otras palabras, se busca desarrollar un sistema compuesto que mediante algoritmos y modelos físicos le permita no solo mover el brazo para alcanzar un punto en el espacio, sino que si le es necesario tener que desplazarse para lograrlo, lo haga. Ahora bien, con el fin de simplificar la tarea global, se desea que un solo proceso resuelva de manera simultánea ambos movimientos. Así como el humano es capaz de caminar para alcanzar algo con su mano sin tener que tomar una decisión consciente que lo concluya. El robot actualmente cuenta con un brazo, un torso móvil (1 DoF) y una base omnidireccional (3 DoF, dos de posición y una de rotación). Se quiere incluir el desplazamiento de la base y el torso en la cadena cinemática del robot como cuatro nuevos grados de libertad. De esta forma se pueden utilizar dentro de la solución que busca cómo colocar el efector final (la mano, pinza o herramienta final de manipulación) en una posición específica del espacio. Es así como la locomoción y la configuración del brazo, que comúnmente se trabajan como tareas independientes, se pueden resolver de forma simultánea.
Ahora bien, limitarse solo a la cinemática del cuerpo entero deja por fuera algunos aspectos, pues hay muchas variables que van más allá de la posición final del efector final. Entre ellas está la trayectoria, la cual depende de los elementos presentes en el medio circundante. Por esta razón se diseña la solución de forma que se pueda considerar la información que el sistema de percepción del robot pueda darle sobre la posición y orientación de la mesa, con el fin de poder evitar colisionar con ella.

1.2. Objetivos

1.2.1. Objetivo general

Ampliar el sistema de control de un robot humanoide para que pueda usar su cuerpo completo para alcanzar un objeto sobre una mesa sin colisionar con ella.

1.2.2. Objetivos específicos

- Revisar los métodos actuales para resolver la cinemática de cuerpo completo de robots humanoides, incluyendo el simulador del robot humanoide del ARCOS-Lab.
- Estudiar las técnicas de control con las que se evaden obstáculos al aproximarse a un objeto en robótica.
- Ampliar las capacidades del simulador del robot humanoide del ARCOS-Lab al incluir la base omnidireccional y torso móvil.
- Adaptar el sistema de control de cuerpo completo en el simulador del robot humanoide para que pueda alcanzar una posición y orientación.
- Implementar un método para evitar colisiones con la mesa en el sistema de control de cuerpo completo.
- Desarrollar pruebas para el sistema de control de cuerpo completo en el simulador.

1.3. Justificación

La robótica busca darle a las máquinas la capacidad de asistirnos con tareas que originalmente eran posibles solo para el humano. En otras palabras, que puedan variar sus acciones según el estado del entorno y siguiendo un objetivo claro. Lo que la hace tan importante es que puede llegar a desempeñar acciones que son cotidianas para nosotros, y así ayudarnos con problemas más exigentes.
Este trabajo busca justamente eso, permitirle a un robot realizar tareas más complejas. Lo hace en la dirección de la robótica colaborativa, que asiste al humano. Esta rama no tiene como propósito la eficiencia última de los procesos, o el perfeccionamiento de capacidades específicas, como la robótica de manufactura, sino que busca que los robots logren ayudar de cerca al humano en ambientes cotidianos y de forma segura, lo cual resulta muy relevante.
Dentro de ese nicho, el proyecto pretende además simplificar la forma en que se realizan ciertas tareas. Debido a la capacidad de abstracción que tiene el humano, pasa desapercibida la cantidad y complejidad de las decisiones implicadas en una acción específica. Para cumplir su cometido, un robot debe ser capaz de imitar la habilidad de planear; esa es, justamente, la carga que se desea alivianar.
La propuesta para alcanzar esto es novedosa. El uso del cuerpo completo de robots humanoides en tareas específicas es un área en desarrollo, y no todos sus casos implican tanto desplazamiento como manipulación. Sumado a esto, el trabajo pretende además unir la locomoción y la configuración del cuerpo dentro de un mismo proceso, reduciendo así los pasos por planificar.
La cinemática de sistemas robóticos es un área ampliamente trabajada, por lo que resulta

accesible agregarle nuevos parámetros y aumentar la cantidad de variables. En síntesis, este trabajo hace uso de herramientas de control de robots ya existentes, cambiando la forma en que se acoplan, para así realizar una tarea conocida de una manera distinta y eficiente.

1.4. Antecedentes

El ARCOS-Lab nace como parte de la Escuela de Ingeniería Eléctrica, con el propósito trabajar en el área de los sistemas autónomos y cognitivos, bajo la premisa de desarrollarse en la frontera del conocimiento científico. Sus proyectos pretenden solucionar problemas sin resolver y generar nuevo conocimiento. Es por esto que una de sus principales ramas es la robótica al servicio del humano, la cual implica temáticas como la manipulación de objetos, la navegación, el control por impedancia, el planeamiento de tareas, percepción de cuerpos y espacios, entre otros. Este carácter innovador le lleva a formar parte del Instituto de Investigaciones en Ingeniería (INII).
Su proyecto de construir un robot humanoide asistente lleva todo esto al límite, ya que se busca que sea capaz de realizar tareas humanas básicas. Detrás del complejo tejido de control y hardware que esto implica, existen numerosas estrategias de cómo abarcar los problemas, como es usual en las áreas que están en pleno desarrollo. El ARCOS-Lab está trabajando en su propuesta particular, la cual se ha ido construyendo a lo largo de varias publicaciones, las cuales han ampliado los trabajos previos del coordinador del laboratorio.

Este trabajo comenzó con el desarrollo de modelos matemáticos de cajas pequeñas que le permitían al robot predecir su comportamiento [1] y además empujarlas sobre una mesa [2]. Esto dio lugar a desarrollar una arquitectura para sistemas de manipulación basados en modelos [3]. Además, como el fin es aplicarlo en su robot, se hizo un estudio por simulación para encontrar la colocación ideal de ambos brazos, como el fin de optimizar su capacidad de manipulación [4].
El curso que llevan estas publicaciones hace necesario avanzar en diversas direcciones, una de las cuales es darle al robot la posibilidad de utilizar todo su cuerpo, y no solo sus brazos. Otra de ellas es desarrollar un sistema que le permita coordinar el desplazamiento y uso de sus brazos, con el fin de llegar hasta los objetos que desea manipular. Eso es lo que este proyecto trabaja.

1.5. Metodología

1. *Estudios y decisiones previas al desarrollo del proyecto*
    - Delimitar el proyecto según los objetivos.
    - Investigar los sistemas que actualmente son utilizados para el control de cuerpo completo y la evasión de obstáculos en artículos científicos.
    - Estudiar el sistema de control con el que cuenta el robot humanoide del ARCOS-Lab.
    - Realizar una selección del sistema de evasión de obstáculos basada en las características actuales del robot y los objetivos del proyecto.

2. *Ampliación de la cinemática del sistema de control actual*
    - Ampliar el sistema de control actual para incluir los grados de libertad del torso y de la base omnidireccional.
    - Realizar pruebas para conocer el comportamiento del simulador ante este cambio.
3. *Diseño de un sistema de coordinación de cuerpo completo para el robot*
    - Realizar un estudio de las necesidades y restricciones para la tarea a llevar a cabo con el cuerpo completo.
    - Definir distintas propuestas basadas en las capacidades actuales del robot.
    - Investigar cómo se lleva a cabo la coordinación del cuerpo completo en la literatura.
    - Desarrollar el sistema y probarlo de manera aislada.
    - Evaluar el resultado y rediseñar en caso de ser necesario.
4. *Implementación del sistema en el simulador del robot*
    - Determinar las secciones de código que será necesario modificar para incluir el nuevo sistema.
    - Implementar el nuevo sistema de coordinación en el simulador.
5. *Incluir un método para evitar colisionar con la mesa*
    - Estudio de la implementación de la evasión de obstáculos en el sistema actual.
    - Realizar una propuesta para adecuarlo al cuerpo completo.
    - Llevar a cabo pruebas del sistema de manera aislada y dentro del simulador.

1.6. Alcance y limitaciones

El proyecto pretende adaptar el sistema de control cinemático del robot humanoide con que cuenta el ARCOS-Lab para que tenga la capacidad de usar no solo uno de sus brazos, sino también su torso y su base omnidireccional. Además se ampliará para que estos grados de libertad se usen simultáneamente de manera que sea capaz de desplazarse para llevar el punto final de su brazo a unas coordenadas dadas en el espacio que representen a un objeto sobre una mesa. Sumado a esto se busca que el robot tome esta última en cuenta mientras se mueve, para evitar colisionar con ella.
Para lograr esto se deben dejar por fuera algunos aspectos que son indispensables para cumplir con este objetivo, pero que no son directamente parte del problema. No se va a trabajar con el sistema de percepción para la localización del objeto o la mesa, o bien para la ubicación del robot. Se va a partir de que se conoce dicha información. Tampoco se van a desarrollar métodos a nivel de planeación de acciones para ningún momento de la tarea a desempeñar. Es decir, no hay ningún proceso de nivel superior que lleve a la elección de este sistema, o que analice sus resultados para tomar alguna decisión durante su ejecución o después.
Es necesario también especificar que las condiciones supuestas en las que se basa el desarrollo de este proyecto ignoran la posible inclinación del suelo u otros obstáculos además de la mesa. Por lo tanto sus simulaciones y pruebas también.

El trabajo no tiene como propósito que el robot sea capaz de manipular ningún objeto, se limita a acercarse a él. Tampoco se busca que el sistema de control tenga capacidad de tomar decisiones estructuradas de navegación para que pueda, por ejemplo, distinguir entre rutas a la hora de aproximarse a su objetivo. Por lo que no se trabajará en que tenga la posibilidad de recuperarse de estados fallidos. La ruta generada se limita únicamente al estado inicial del robot, y la posición y orientación del objeto por alcanzar y la mesa sobre la que se ubica.
El objetivo aquí trabajado busca demostrar que los principios y la estrategia utilizadas resultan en un sistema útil que es capaz de resolver el problema de control descrito. No se pretende llegar a un producto fuera del estado de desarrollo.

Capítulo 2

Marco teórico

En esta sección se describen los distintos conceptos necesarios para trabajar en el control de un robot humanoide para la manipulación de objetos. A su vez se presentan los antecedentes que han permitido llegar hasta el problema que se abarca en este proyecto, junto a las estrategias que actualmente se usan para solucionarlo.

2.1. Robots humanoides

La robótica nace con el propósito de crear máquinas capaces de servir al humano a resolver problemas que requieren de cierta interacción. Por tanto están pensados en imitarlo en su capacidad de percibir y dar una respuesta modificando su entorno; siguiendo una lógica preestablecida [5].
Esta intención vino como consecuencia de la idea de dar vida a sus creaciones. Es por esto que la disciplina que hoy conocemos como robótica trabaja en el estudio y desarrollo de estructuras móviles animadas, que manipulan y/o se movilizan. Estas no están restringidas a una figura en específico, sin embargo, la más conocida es la antropomórfica.
La robótica humanoide, como explica [6], se distingue de otras ramas porque tiene como objetivo que sus robots puedan trabajar en nuestro ambiente, usar nuestras herramientas y que se parezcan a nosotros. Hay varias razones que justifican esto, la más fuerte es que el medio en que vivimos y nuestros instrumentos están hechos a nuestra medida. Por otro lado, su similitud con nosotros nos facilita interactuar con ellos.
Los robots humanoides van más allá de solo imitar la forma del humano, sino que procuran adoptar (o superar) todas las características posibles, como por ejemplo sus sentidos, su comportamiento, sus vías de comunicación, etc. No obstante, para alcanzar esto, la robótica ha tenido que enfocarse en desarrollar capacidades específicas en forma independiente.

2.1.1. Manipulación móvil

Cuando el desarrollo de robots enfocados en la manipulación alcanzó cierta madurez se les proveyó de sistemas que les permitía desplazarse en el medio. Este avance abre la puerta a la investigación de varios problemas nuevos [7]:

- La coordinación de dos subsistemas móviles profundamente interdependientes.
- La interacción con un medio *no estructurado* (que no ha sido adaptado en función del robot).

- El cumplimiento de tareas compuestas que implican resolver distintos problemas de manipulación en un orden determinado.
- El balance estático y dinámico del robot se torna más complejo [5].

La ventaja de estos nuevos robots es la combinación ideal entre el alcance casi ilimitado de una base móvil y la gran capacidad de manipular de un brazo [5]. Se podría decir que esa es la configuración más simple y que también los robots humanoides entran, intrínsecamente, dentro de este grupo.
Este mismo autor explica que los dos tipos más grandes de manipuladores móviles son los bípedos y los de ruedas, siendo este último considerablemente mayor. Hay varios tipos de sistemas de ruedas, algunos de los cuales implican que hayan ciertas restricciones:

- Base móviles no holonómicas: como se ve en la sección 2.2.1, esto significa que el sistema está sometido a ciertas reglas a la hora de desplazarse, como ocurre con un carro convencional.
- Bases móviles omnidireccionales: tienen la capacidad de desplazarse hacia cualquier dirección cartesiana en el plano, en cualquier instante. Dos ejemplos de ellos son las bases con ruedas Mecanum y aquellas con al menos tres ruedas giratorias motorizadas (como las que se utilizan en las sillas de escritorio), tal como se ve en la figura 2.1.

Figura 2.1: Dos ejemplos de bases omnidireccionales: a la izquierda el uso de ruedas giratorias, a la derecha ruedas Mecanum

Fuente: elaboración propia, basado en [5]

Cabe resaltar que en el robot del ARCOS-Lab se utilizan las ruedas Mecanum, ya que en comparación con otros sistemas de ruedas, como las usadas en el Rollin' Justin resultan más económico. Además, de que permiten soportar altas cargas, a la diferencia de las ruedas giratorias.

2.1.2. Robots de cuerpo entero para la manipulación

La robótica humanoide como área de investigación es muy amplia, por tanto los proyectos y robots están enfocados en trabajar solo ciertos aspectos. En esta sección se describen algunos robots humanoides que forman parte de investigaciones similares a las del ARCOS-Lab, que como se ha mencionado busca desempeñar tareas de manipulación complejas.

Rollin' Justin

Este robot fue desarrollado por la Agencia Aeroespacial Alemana (DLR) y está pensado para realizar tareas cotidianas para el humano, por tal razón su infraestructura es antropomórfica. [8] explica sus componentes:

- Dos brazos de 7-DoF LBR generación III, ligeramente modificadas, cada una con una mano DLR Hand II, que cuentan con cuatro dedos, cada uno de 3-DoF.
- Una cabeza con numerosos sensores de percepción montada sobre un cuello de 2-DoF
- Un torso móvil con 3-DoF
- Una base móvil compuesta de tres piernas extensibles, cada una con una rueda y la capacidad de girar con respecto al torso sobre el eje vertical del robot

Se ha logrado que este robot sea capaz de atrapar dos bolas lanzadas hacia él simultáneamente y realizar tareas de varias etapas como preparar café , tal como se ve en la figura 2.2. Esta última tarea implicó el desarrollo de un sistema de control con gran precisión de movimiento y fuerza [9].

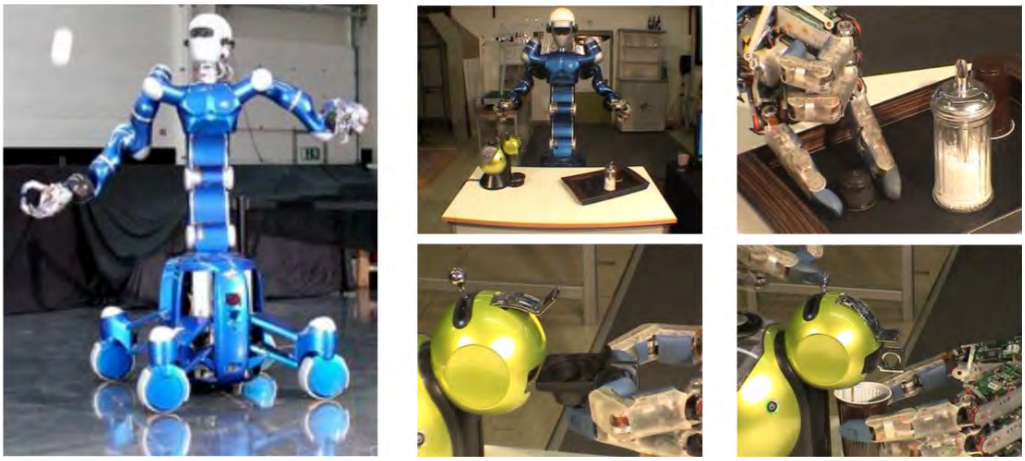


Figura 2.2: Robot Rollin' Justin: en (a) atrapando una bola que se le es lanzada. En (b) preparando café

Fuente: Elaboración propia a partir de [9]

TUM-Rosie

Es uno de los robots desarrollados por la Universidad Técnica de Munich (TUM, por sus siglas en alemán) para realizar tareas de manipulación en ambientes cotidianos. En la figura 2.3 se le puede ver cómo acerca su mano para manipular un objeto común en una cocina. En [10] se logra que cocine *pancakes*, para lo cual vierte la mezcla sobre el sartén y le da vuelta cuando es requerido utilizando una espátula. En esta demostración se hace evidente

su capacidad para buscar y tomar objetos, usarlos como herramientas para interactuar con otros elementos, separar tareas en pasos más pequeños, entre otros.
Además el robot está pensado para manipular de una forma segura para el humano, por lo que está equipado con manos y brazos que pueden ser controlados por impedancia. A continuación se listan los principales componentes [11]:

- Base Omnidireccional con cuatro ruedas Mecanum
- Dos brazos KUKA LBR de 7-DoF
- Dos manos DLR-HIT
- Una cabeza con un sensor 3D de nube de puntos, dos cámaras de alta definición y una cámara térmica; montada sobre un cuello de 2-DoF.

Figura 2.3: Robot TUM-Rosie acercándose a un objeto con el fin de manipularlo

Fuente: [12]

Cody

Fue creado por el Laboratorio de Robótica para el Cuidado de la Salud del Instituto Tecnológico de Georgia para funcionar como enfermera. Por tal razón se le usa para lograr tareas complejas que además requieren gran delicadeza. Por ejemplo, [13] lo utiliza para abrir puertas y gavetas de formas y tamaños desconocidos, utilizando la movilidad del cuerpo entero y no solo sus brazos. En la figura 2.4 se puede ver una secuencia de dicha tarea.
Este robot, similar a los anteriores, cuenta con los siguientes actuadores y sensores:

- Dos brazos MEKA A1 equipados con un sensor de fuerza de seis ejes en sus muñecas
- Una base omnidireccional Segway RMP 50 Omni
- Un torso móvil de 1-DoF que le permite subir y bajar

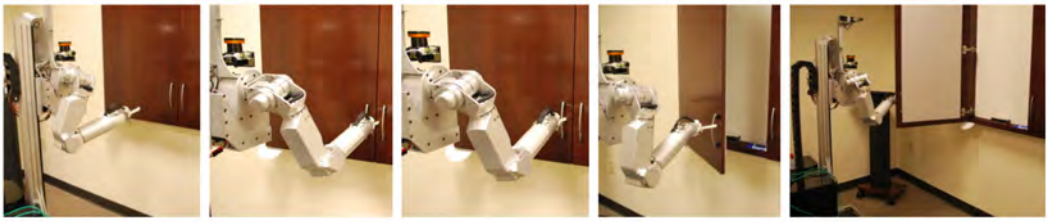


Figura 2.4: Secuencia en la que Cody se acerca a una puerta y la abre utilizando su cuerpo entero

Fuente: [13]

PR2

Este robot humanoide es comercializado por Willow Garage ampliamente utilizado para la investigación dentro del área de manipulación gracias a su gran redundancia. Por ejemplo en [10] acompaña a TUM-Rosie en la elaboración de *pancakes* y en [14] se le incorpora un sistema que le permite describir trayectorias suaves con sus brazos, como prueba se demuestra como puede escribir letras mientras sostiene la pizarra con su otra mano (figura 2.5). Algunos de sus componentes son [11]:

- Base Omnidireccional
- Dos brazos de 7-DoF
- Dos prensas con sensibilidad a la presión
- Varios sensores de percepción en distintas partes: sensor 3D de nube de puntos, una cámara de alta definición, dos escáneres láser.

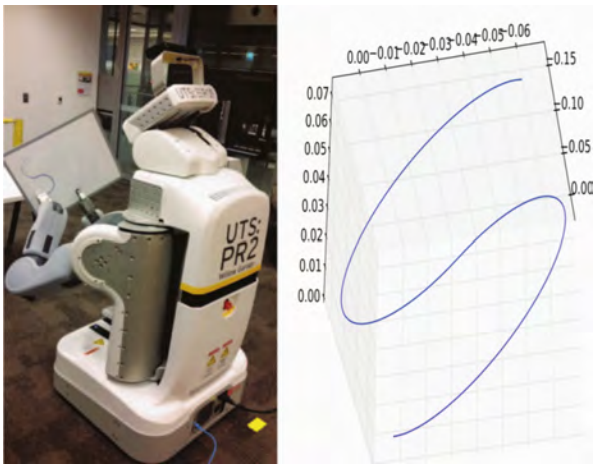


Figura 2.5: PR2 escribiendo en una pizarra trazos que requieren un movimiento preciso y suave de su efector final

Fuente: [14]

2.1.3. El robot humanoide del ARCOS-Lab

En el presente proyecto se pretende aportar al desarrollo de la capacidad de manipulación del robot humanoide del ARCOS-Lab, el cual está pensado para cumplir asistir en tres escenarios o tipos de tareas (figura 2.6):

- Colaboración en mesa de trabajo
- Tareas autónomas en una cocina
- Organización en una bodega inteligente

Esto implica que el robot debe ser capaz de desempeñarse en un ambiente cambiante, donde interactuará con numerosos objetos, y debe hacerlo sin poner en riesgo a los humanos que están cerca.

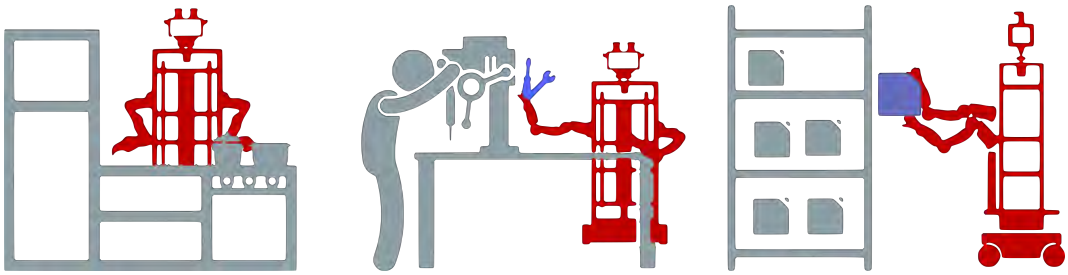


Figura 2.6: Los tres escenarios para el robot del ARCOS-Lab: en una cocina, colaborando en una mesa de trabajo, en una bodega inteligente.

Fuente: [12]

Este proyecto creció asociado con el trabajo que realizó Ruiz-Ugalde [11] con el TUM-Rosie, por lo que sus componentes y diseño se parecen mucho. Su arquitectura de software también se basa en los controladores creados para dicha investigación. Todos estos aspectos se describen con mayor detalle en las secciones 2.4 y 2.5

2.2. Fundamentos mecánicos

2.2.1. Conceptos básicos

Grados de libertad

Los grados de libertad (*Degrees of Freedom*, DoF) de un sistema indican la cantidad mínima de variables con las que se puede describir su estado. En el caso de un robot, definen su configuración. Para cuerpos rígidos finitos en tres dimensiones se pueden definir seis grados de libertad [15].
El número de grados de libertad de un robot, cuando se compone de eslabones y articulaciones, se puede describir mediante la Fórmula de Grübler:

$$m(N-1-J)+\sum_{i=1}^{J} f_i \tag{2.1}$$

Donde:

m = Grados de libertad de los cuerpos rígidos
N = Número de elementos, incluyendo la tierra
J = Número de articulaciones
f = Grados de libertad de cada articulación

Representación del espacio

A la hora de definir un modelo mecánico, es indispensable conocer las formas en que se pueden definir las variables. Puede que hayan movimientos en el espacio que no hagan uso de todos los grados de libertad disponibles, por ejemplo, debido a su configuración física. En este caso existen dos formas de describirlo [15]:

- Representación Explícita: utiliza la mínima cantidad de coordenadas necesaria, pero puede implicar singularidades.
- Representación Implícita: se utiliza un sistema de coordenadas con más variables y se reducen utilizando restricciones, de forma que describan la configuración deseada.

Restricciones

El control depende de las posibles respuestas que da un sistema, por eso es importante conocer las distintas formas en que se puede restringir el movimiento de un componente de nuestro robot. De acuerdo con [16], un sistema mecánico se puede clasificar de acuerdo con el tipo de restricciones que modelan su movimiento :

- Holonómico: aquellas que se pueden describir mediante un sistema de ecuaciones que relacionan únicamente variables de posición. Aunque en ocasiones resulte más cómodo expresarlo en términos de la velocidad (restricciones de Pfaffian), pero siempre pudiéndose integrar completamente.
- No Holonómico: las que no cumplen el criterio anterior y por tanto la posición termina siendo dependiente de la velocidad.

2.2.2. Cinemática de cuerpo rígido

Aquí se describen los conceptos fundamentales con los que popularmente se modela la cinemática de los robots en la actualidad y que son pertinentes para el control. Se trabaja utilizando una representación implícita de los sistemas, como es usual para el área.

Pose de un cuerpo

La descripción más general del espacio en que vivimos requiere de más de tres dimensiones. Por lo menos para un cuerpo finito. Nuestros cuerpos, y los objetos con los que interactuamos, (y así también para los robots) no solo pueden variar en tres ejes independientes, sino también orientarse alrededor de ellos [17]. Esto suma seis grados de libertad, tres de posición y tres de orientación, y su conjunto recibe el nombre de *pose*.

Matrices rotacionales

Son matrices que representan la orientación relativa de un marco de referencia respecto a otro (por ejemplo de un cuerpo, b, con respecto al espacio, s) [5]. Pero con la ventaja de que se pueden realizar operaciones algebraicas con ellas, gracias a la forma en que están definidas:

$$R_{sb} = [\hat{x}_b, \hat{y}_b, \hat{z}_b] = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Donde:

$\hat{x}, \hat{y}, \hat{z}$ = Vectores unitarios
R_{sb} = Matriz rotacional del marco de referencia b con respecto al s.

Nótese que las columnas, deben ser ortogonales entre sí, ya que son los vectores unitarios de un sistema de coordenadas.
El resultado de usarla en multiplicaciones matriciales se puede interpretar de dos formas (ver figura 2.7): cambiar el marco de referencia de un elemento

$$R_{sc} = R_{sb}R_{bc} \tag{2.3}$$

$$p_s = R_{sb}p_b \tag{2.4}$$

O lo que sería equivalente, "rotarlos" a la posición de ese nuevo marco de referencia descrito por la matriz.

$$R_{sb} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} = Rot(\hat{z}, \beta) \tag{2.5}$$

Donde:

β = ángulo de rotación sobre el eje z

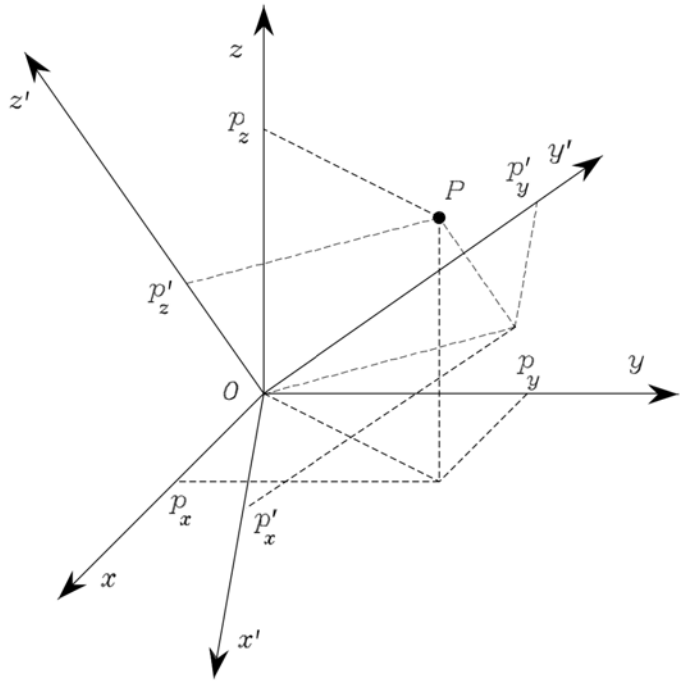


Figura 2.7: Un punto representado en dos marcos de referencia distintos

Fuente: [5]

Velocidades angulares

Una de las formas de representar esta variación es por medio de un vector unitario que define el eje de rotación y un escalar que indique la tasa y el sentido, como se muestra en la ecuación 2.6:

$$\omega = \hat{\omega}\dot{\theta} \tag{2.6}$$

Donde:

$\hat{\omega}$ = Vectores de rotación
$\hat{\omega}$ = Vector unitario de la dirección de giro
$\dot{\theta}$ = Magnitud de la velocidad angular

Con el fin de que esta definición funcione para realizar operaciones, se utiliza su matriz asimétrica $[\omega]$, como indica [15]:

$$[\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \ \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \in \mathbb{R}^3 \tag{2.7}$$

De forma que:

$$\dot{R} = R[\omega_b] \tag{2.8}$$

Matrices de transformación homogéneas

Permiten efectuar rotaciones y desplazamientos de un cuerpo rígido, utilizando la misma definición de las matrices rotacionales (los componentes de un marco fijo al cuerpo con respecto a un marco de referencia), junto a un vector que describe la posición de un punto del cuerpo, con respecto a la misma referencia [5].

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Donde:

T = Matriz de Transformación Homogénea
R = Matriz Rotacional
p = Vector de ubicación
r_{ij} = Componentes de la matriz Rotacional
p_i = Componentes del vector de ubicación

Velocidad espacial

La velocidad, tanto traslacional como rotacional, se puede expresar utilizando los vectores ω y $\dot{p}$, respectivamente. Estos dos pueden combinarse en un mismo vector $\nu \in \mathbb{R}^6$, usualmente llamado *twist*, ya sea referenciado con respecto al cuerpo o al espacio [15]:

$$\nu_b = \begin{bmatrix} \omega_b \\ v_b \end{bmatrix} \text{ or } \nu_s = \begin{bmatrix} \omega_s \\ v_s \end{bmatrix} \tag{2.10}$$

Al igual que con la velocidad angular, con el fin de realizar operaciones, resulta más útil utilizar su versión matricial

$$[\nu] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \tag{2.11}$$

Con la cual se puede calcular el cambio en la pose del cuerpo:

$$\dot{T} = T[\nu_b] \tag{2.12}$$

Palancas

Así como se ha hecho con el desplazamiento y con las velocidades, es necesario definir una expresión general que incluya los componentes traslacionales y rotacionales de las fuerzas, momento y fuerza lineal: m y f. El planteamiento es análogo, un vector $\mathcal{F} \in \mathbb{R}^6$, usualmente llamado *wrench*, referenciados al cuerpo o al espacio [15]:

$$\mathcal{F}_b = \begin{bmatrix} m_b \\ f_b \end{bmatrix} \text{ or } \mathcal{F}_s = \begin{bmatrix} m_s \\ f_s \end{bmatrix} \tag{2.13}$$

Cadenas cinemáticas

Son la composición de cuerpos rígidos consecutivos mediante restricciones de movimiento, es decir eslabones unidos por articulaciones. Estas últimas, también llamadas pares cinemáticos, se pueden clasificar en varias clases, pero para el área de robótica se pueden reducir a dos formas simples: uniones prismáticas y de revolución [16]

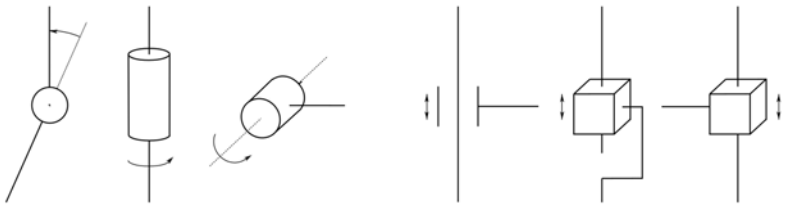


Figura 2.8: A la izquierda tres ejemplos de articulaciones de revolución, a la derecha tres prismáticas

Fuente: [5]

Cabe mencionar que pueden existir muchas clasificaciones. En lo que respecta a este proyecto, se trabaja con una cadena simple (todos los eslabones tienen máximo dos pares cinemáticos), y abierta (no hay par cinemático entre la primer y último eslabón).

Notación de Denavit-Hartenberg

Con el fin de poder realizar un modelo repetible y de fácil comprensión, en el área de robótica se han adoptado procedimientos bien definidos de cómo relacionar los pares cinemáticos y los marcos de referencia con las articulaciones y eslabones. Tal vez el más conocido es la conveción de Denavit-Hartenberg.
El algoritmo a seguir es descrito por [5] por medio de los siguientes pasos:

- Asignación de nombres: el bastidor o base es el eslabón 0 y el efector final el n. Se define de forma que el eslabón i tiene adherido en su extremo final el marco i.
- Determinación del eje z: los marcos de cada eslabón se ubican al final, de forma que el eje z_i calce con el eje de la articulación $i+1$
- Ubicación de los orígenes: se ubican el origen O_i donde se cortan el eje z_i y la normal común entre los ejes z_i y z_{i-1}.
- Demás ejes: el eje x_i se ubica sobre la normal común, en dirección hacia el eslabón $i+1$. El eje y responde a la regla de la mano derecha.

Esta definición recursiva permite que cuando la articulación 1 se mueve, todos los demás marcos se moverán relativos a este.

Espacios

Para robots manipuladores es útil definir términos para referirse a ciertos conjunto de variables [15]:

- *Joint Space* es aquel en el que se definen los vectores nx1 que contienen las posibles posiciones de las n articulaciones, y por tanto todas las posibles configuraciones del robot. En la figura 2.9 es representado por las variables angulares azules.
- *Operational Space*: llamado también *task space*, es el espacio de las operaciones, aquel en el que se definen los vectores mx1 que contienen las poses m-dimensionales del último eslabón de una cadena cinemática (comúnmente llamado efector final). En la figura 2.9 es representado por un vector de posición y un marco de referencia que indica la orientación, ambos en rojo.
- *Workspace*: Es el espacio de trabajo, el conjunto dentro del *operational space* que el sistema puede lograr con sus limitaciones físicas. Cuando esto implica únicamente posición se dice que es "de alcance"(*reachable workspace*), pero cuando implica tener dos o más orientaciones se le llama "de destreza"(*dexterous workspace*).

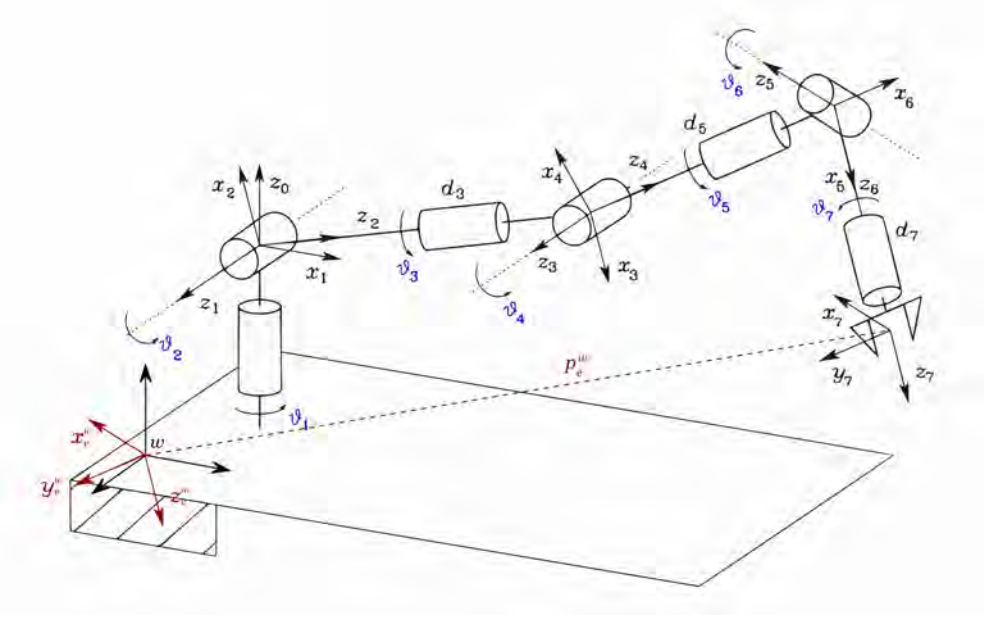


Figura 2.9: Representación del *joint space* (en azul) cuyos elementos son un vector que se compone del conjunto de los ángulos que definen la posición del brazo. Y del *task space* (en rojo) que definen la pose del efector final, en este caso en términos cartesianos, mediante el vector p_e^w que determina la posición y una matriz rotacional compuesta por x_e^w, y_e^w y z_e^w

Fuente: elaboración propia, basado en [5]

Cinemática directa

También se le llama modelo geométrico del robot [17] o problema de posicionamiento [16]. Es el procedimiento con el que se obtiene la posición del punto final del efector final utilizando

el conjunto de las posiciones de cada una de las n articulaciones del sistema , $q \in \mathbb{R}^n$, por lo que su resultado final se puede expresar en la forma de una matriz de transformación homogénea que describe al efector final, e, con respecto a la base de la cadena, b [5]:

$$T_e^b(q) = \begin{bmatrix} n_e^b(q) & s_e^b(q) & a_e^b(q) & p_e^b(q) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

Donde:

$n_e, s_e, a_e =$ vectores unitarios del marco de referencia del efector final

Típicamente se obtiene por medio de la composición de las matrices correspondientes a cada uno de los eslabones, cada una de las cuales es función de la posición según la articulación que le precede inmediatamente. Esto se puede describir de manera sistemática mediante la convención de Denavit-Hartenberg, la cual necesita únicamente de cuatro parámetros para establecer una relación entre dos marcos, donde solo una es variable, de naturaleza prismática o de revolución, respectivo al tipo de articulación. La forma más generalizada de la matriz sería la siguiente:

$$T_i^{i-1}(q_i) = \begin{bmatrix} \cos\vartheta_i & \sin\vartheta_i \cos\alpha_i & \sin\vartheta_i \sin\alpha_i & a_i \cos\vartheta_i \\ \sin\vartheta_i & \cos\vartheta_i \cos\alpha_i & -\cos\vartheta_i \sin\alpha_i & a_i \sin\vartheta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

Donde:

$\vartheta_i =$ ángulo entre los ejes x a lo largo del eje z_{i-1}
$\alpha_i =$ ángulo entre los ejes z a lo largo del eje x_i
$a_i =$ Distancia normal común entre los ejes z
$d_i =$ Distancia a lo largo del eje z_{i-1} hasta la intersección con la normal común

Cinemática inversa

Es el procedimiento por el cual se obtienen las posiciones de cada una de las articulaciones del sistema a partir de una pose deseada para el efector final. El problema es más complejo que la cinemática directa, ya que pueden existir numerosas soluciones, o incluso infinitas, en caso de que haya redundancia. Hay dos formas de abarcarlo, analíticamente, que permite discriminar entre soluciones factibles; o bien por métodos numéricos, que puede aplicarse a cualquier estructura cinemática [5].

2.2.3. Cinemática diferencial

Con el fin de poder describir la cinemática directa e inversa de manera más general, es necesario tomar en cuenta también las velocidades.

Matriz jacobiana

En este contexto podríamos decir que es la relación entre el vector $\dot{x}_e$, $m \times 1$, de velocidades del efector final en el espacio de configuraciones, y el vector $\dot{q}$, $n \times 1$, de velocidades de las articulaciones [15]:

$$\dot{x}_e = J(q)\dot{q} \tag{2.16}$$

$$J(q) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} \in \mathbb{R}^{m \times n} \tag{2.17}$$

Donde f_m es la función que define la m-ésima coordenada del efector final a partir de los valores de las articulaciones. Por tanto, la n-ésima columna es el vector que describe la velocidad de la punta si se mueve solo esa n-ésima articulación.
Con el fin de obtener un método fácil de computar, se puede determinar una forma general para el efector final, que se base únicamente en las propiedades geométricas de la cadena cinemática. Como lo demuestra [5], esto se logra determinando la columna J_i de cada articulación de forma independiente:

$$J(q) = \begin{bmatrix} J_1(q) & \ldots & J_m(q) \end{bmatrix} \tag{2.18}$$

La forma de cada una es de la siguiente forma, según sea una articulación de revolución o prismática:

$$J_i(q) = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{si la articulación es prismática} \\ \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{si la articulación es de revolución} \end{cases} \tag{2.19}$$

Donde:

z_{i-1} = vector unitario de la articulación anterior
p_e = vector de posición del efector final
p_{i-1} = vector de posición de la articulación anterior

Esta matriz también permite relacionar las fuerzas en la punta de la cadena con los momentos en cada una de las articulaciones. Se puede demostrar que la relación 2.20 es válida al asumir que no hay fuerzas externas y que el sistema está en equilibrio:

$$\tau = J^T(q) f_{ee} \tag{2.20}$$

Donde:

τ = vector de momentos en las articulaciones
f_{ee} = vector de fuerzas en el efector final

Singularidades

Corresponde a un determinado estado en el *joint space* donde el efector final no se puede mover instantáneamente en al menos una dirección [15]. Esto se puede ver desde la perspectiva de la matriz jacobiana, donde las filas corresponden a una dimensión en el campo operacional, si para un dado q una de ellas llegara a perder su independencia con otra dicha dimensión estaría supeditada. Esto sucede cuando el rango de la matriz es distinto a su valor máximo.
Esto puede ocurrir tanto en los límites de alcance del manipulador, conocidas como singularidades externas; como dentro del espacio de posibles configuraciones, debido a razones geométricas, conocidas como singularidades internas [5]. A continuación se muestran algunos casos específicos, en los que [15] indica que esto sucede:

- Dos articulaciones de revolución están colineales.
- Tres articulaciones de revolución están en un mismo plano y sus ejes son paralelas.
- Los ejes de cuatro articulaciones de revolución se intersecan en un mismo punto.
- Hay cuatro articulaciones de revolución en un mismo plano
- Seis articulaciones de revolución intersecan una línea común.

Redundancia

Es la capacidad de un sistema de tomar distintas configuraciones en el *joint space* sin modificar su pose en el *configuration space*. Es útil para evadir obstáculos o evitar singularidades. Se puede medir como la diferencia entre los grados de libertad de la cadena cinemática n y los de la tarea que se quiere realizar r.
La gran mayoría de los robots humanoides cumplen esta característica, ya que el cuerpo del humano también. Ejemplo de esto son todos los mencionados en la sección 2.1.2, donde los DoF totales pueden llegar hasta 51 en el caso de Rollin' Justin. Sacarle provecho a dicha cualidad es de suma importancia para este proyecto, ya que se va a trabajar con una cadena de 11 DoF y se desea evadir obstáculos.

2.3. Control de manipuladores móviles

2.3.1. Teoría de control de manipuladores

Controladores y predictores

Son modelos que relacionan el estado actual de un objeto (definido por medio de variables y constantes) con entidades externas que pueden modificar ese estado. La forma natural, o directa, en la que esto sucede es que un estado A, perturbado por una magnitud, cambia en un estado B. El conjunto de ecuaciones y procesos que predicen el resultado a partir del estado inicial y la perturbación se llaman predictores. El proceso inverso, en el que se determina una perturbación que podría cambiar del estado A al B se llama controlador [18].

Estrategias de control

Partiendo de las tareas que se desea que un robot pueda desempeñar, se pueden distinguir distintos modos para controlarlos. Lynch y Park [15] hacen distinción de:

- Control de Fuerza
- Control de Movimiento
- Control Híbrido de Fuerza y Movimiento: unos ejes requieren control de fuerza y otros de movimiento.
- Control por Impedancia: el actuador debe responder amortiguando la fuerza que se le aplica.

También hacen la aclaración que en ningún caso es posible controlar tanto la fuerza como el movimiento simultáneamente y en la misma dirección. Una variable será siempre definida por el actuador y otra por el ambiente.
A su vez ambas opciones se pueden llevar a cabo dándole a los motores instrucciones de torque o de velocidad. Esto también es una elección que depende de la tarea que se quiere llevar a cabo.

Control de movimiento

Este es tal vez el control más básico y que se aplica a más situaciones, siempre y cuando la variación en la fuerza no sea un riesgo. Por ejemplo, como pasa con los brazos robóticos industriales, que deben describir rutas precisas e invariantes. Por esta misma razón no es recomendable para la mayoría de las tareas que desempeñan los robots descritos en la sección 2.1.2. Por ejemplo Cody, que desempeña tareas para el cuidado de la salud [13] debe tener un control estricto de la fuerza que aplica.
Se puede llevar a cabo utilizando tanto las variables cinemáticas de las articulaciones (internas) o del efector final (externas). El primer método es el más simple y el único realmente directo, sin embargo no es recomendable para robots que requieren desempañar comportamientos variables en el tiempo o no lineales del todo. El segundo requiere de un modelo cinemático que relacione el *joint* y el *task space*, pero tiene la ventaja de que se centra en el controlar la pose del efector final, lo cual está estrechamente relacionado con las tareas que lleva a cabo el robot [17].

Las instrucciones de control deben llegar hasta los motores. La forma de estos comandos depende de su principio de funcionamiento [15]:

- Comandos de fuerza: usualmente los motores trabajan generando un torque específico según la señal recibida. En este caso las instrucciones de control de movimiento se deben traducir a un equivalente de torque en el motor de cada articulación. Esto requiere de un modelo dinámica inversa.
- Comandos de velocidad: algunos motores eléctricos, como los de pasos, controlan frecuencia y no torque. Por esta razón podemos usar directamente la señal de velocidad del *joint space*.

Las opciones anteriores dan como resultado cuatro formas distintas de controlar el movimiento del robot, se muestran de forma simplificada en las figuras 2.11 y 2.10, donde **x** representa las variables cinemáticas externas (del efector final) y **q** las internas (de las articulaciones). El subíndice **d** indica que son las posiciones deseadas y el símbolo $\sim$ hace referencia a que es un valor que ya ha pasado por el proceso de control.

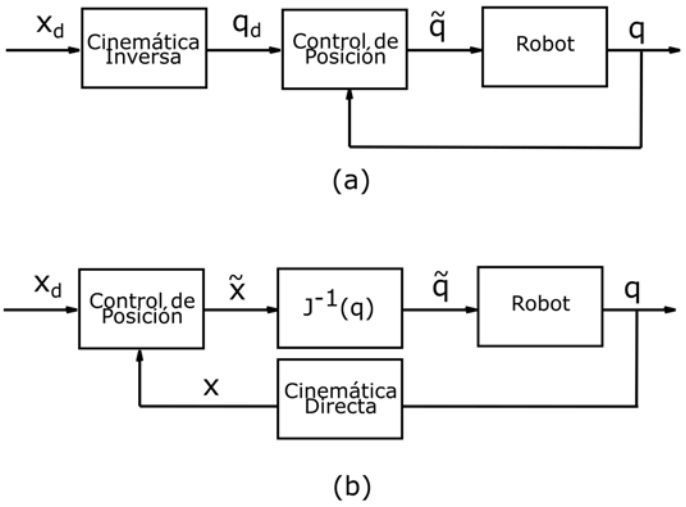


Figura 2.10: Esquemas de control de movimiento utilizando comandos de velocidad para el motor. En (a) controlando las variables cinemáticas internas y en (b) las externas

Fuente: Elaboración propia, basado en [17]

Control de fuerza

Se aplica a situaciones donde la posición del manipulador debe responder a la fuerza que aplica sobre el medio. Un ejemplo de ello sería empujar un objeto en una dirección dada, como [2].
No obstante, este escenario de control supone que se quiere controlar la fuerza en todos sus grados de libertad. Sin embargo, esto sucedería solamente en el caso en que el efector final

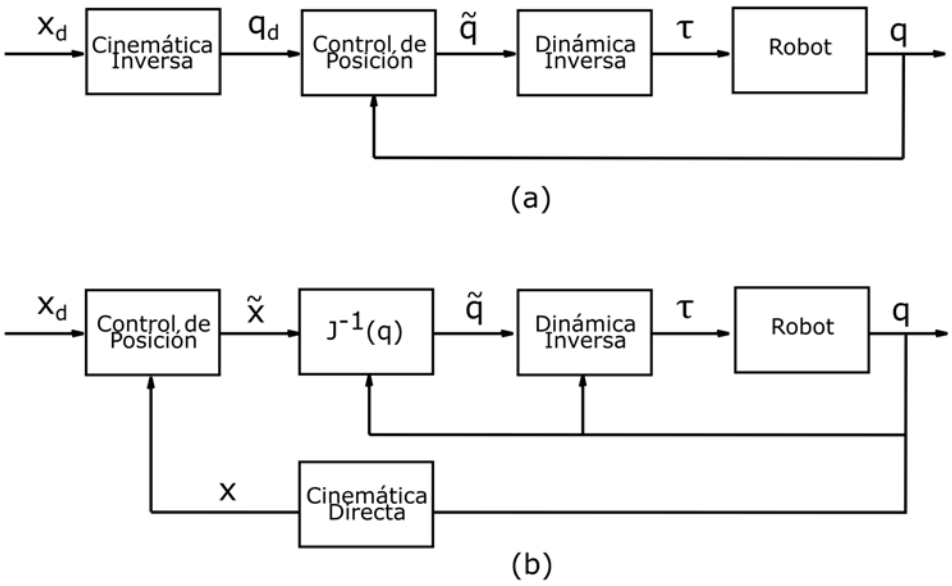


Figura 2.11: Esquemas de control de movimiento utilizando comandos de fuerza para el motor. En (a) controlando las variables cinemáticas internas y en (b) las externas

Fuente: Elaboración propia, basado en [17]

esté fijo a un resorte o algún elemento deformable. Por tanto, el desarrollo de este tipo de control está pensado para darle fundamento teórico al control híbrido, que es una aplicación con mayor cabida en la práctica [15].
Uno de los métodos para lograrlo es el descrito por [17] y [5]. Aquí se utiliza un sensor de fuerza para medir el *wrench* en el efector final. A partir de un modelo de la dinámica del manipulador, se puede determinar las variables externas de movimiento y así hacer uso del control por posición. El resultado se ve en la figura 2.12.

Control híbrido de fuerza y movimiento

Esta es la unión del control de fuerza en ciertos grados de libertad y el control de movimiento en otros. Se aplica a tareas más complejas, como por ejemplo soldar dos piezas: se requiere cierta presión normal a la superficie, pero en las otras dos dimensiones necesita desplazarse siguiendo una ruta específica [15].
El principal reto que implica este control está en la interacción del actuador con el ambiente y las restricciones que esto impone sobre la relación entre el control de fuerza y movimiento. Por ejemplo, si se quiere desplazar un cuerpo finito sobre una superficie manteniendo una fuerza de contacto constante, un cambio indeseado en la orientación del objeto puede cambiar las reglas de control de la fuerza [5].
Abrir puertas es un ejemplo claro, [13] controla la fuerza en algunos ejes de la muñeca del robot Cody, pero siempre buscando poder moverse en una trayectoria definida por la bisagra

y el radio de la puerta.

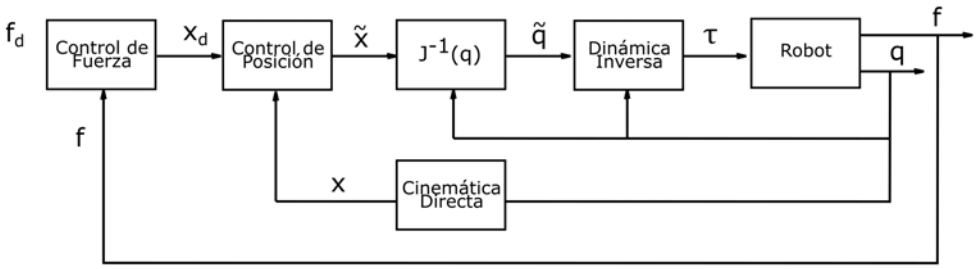


Figura 2.12: Esquema del control de fuerza mediante control de movimiento

Fuente: Elaboración propia, basado en [17]

Control por impedancia

Las estrategias de control descritas atrás (incluida la híbrida) se concentran o bien en la velocidad o en la fuerza. Esto es porque están pensadas para realizar tareas donde importa controlar una o la otra, pero no la potencia implicada [19]. Para lograrlo es necesario tener control de ambas variables, sin embargo, como ya se indicó, esto no es posible, ya que la relación depende del medio [15]. Una forma de evadir este problema es modulando la respuesta del sistema ante los estímulos, lo cual se puede hacer al cambiar los parámetro del sistema de control o al variar las propiedades mecánicas del manipulador [19].
Esto es de gran utilidad en aplicaciones como la de este proyecto, donde el movimiento del robot siempre debe estar restringido a la posibilidad de encontrar obstáculos que le impidan moverse, sobretodo cuando se sabe que podría ser un humano. Ser capaz de detenerse lo hace compatible para desempeñar tareas de cooperación, como es el caso de [11], [13], [8] y [14].
Hogan [19] explica que cuando se desea manipular el medio, este se puede emular siempre como un elemento de admitancia -que recibe fuerza y genera movimiento- y el actuador como uno de impedancia -que recibe flujo y genera fuerza-. Mediante un modelo virtual basado en las constantes de impedancia del manipulador (inercial, de amortiguación y elástica) se pueden relacionar la fuerza y el desplazamiento en el efector final [15]:

$$f_{ee} = M\ddot{x} + B\dot{x} + Kx \tag{2.21}$$

Basta con utilizar la ecuación 2.20 para poder controlar dicho modelo a partir del torque en las articulaciones.

2.3.2. Control cinemático

La cinemática es el área de la mecánica que se encarga del estudio del movimiento de los cuerpos, sin ocuparse en sus causas. Por tanto sus variables son la posición, el tiempo y todas las derivadas subsecuentes. La robótica está íntimamente relacionada con el movimiento, todas sus tareas lo requieren, y muchas de ellas podrían limitarse a él; solo a partir de cierto punto debe involucrarse la dinámica.

Para este proyecto esta es la principal rama de control que se requiere, ya que la atención está en el escenario robot necesite mover su cuerpo de un lugar a otro, sin manipular aún objetos.

Coordinación de la movilidad y la manipulación

La ventaja de tener sistemas que integran bases móviles y manipuladores es que aumentan el campo de trabajo disponible. Sin embargo, esto implica un crecimiento en la complejidad del control, ya que hay mayor dimensionalidad, incertidumbre y hay que responder a tareas más variables [7].
La forma en que se decida coordinar ambas acciones juega un papel importante en esos tres aspectos. A continuación se describen las distintas formas en que se puede abarcar el problema:

- **Sistemas no simultáneos**: esta es la opción más básica de todas, donde el robot o bien se desplaza o utiliza su manipulador, pero nunca ambos al mismo tiempo [20]. A pesar de lo limitado, esto puede ser suficiente para atender muchos de los retos que hay en esta área y resulta ser, por tanto, la práctica más común [21].
- **Modelos de control descentralizado**: la base y el brazo tienen la capacidad de moverse de manera simultánea, pero siempre como dos subsistemas distintos. Usualmente, con el fin de mejorar la manipulación, la base se mueve en respuesta a la necesidad del manipulador.
  - Khatib *et al.* [22] desarrolló una estrategia de coordinación que procura resaltar la manipulabilidad del brazo, usando el espacio nulo de los movimientos de la base. Para ello usa un modelo de dinámica inversa.
  - Inoue *et al.* [23] trabaja con un robot humanoide y diseña un sistema de coordinación para manipulación móvil basado en los criterios de estabilidad y manipulabilidad. Como su locomoción es con piernas, la dinámica del cuerpo completo se vuelve muy relevante, de forma que para mantener los criterios para el movimiento, cada paso es pensado en respuesta a los movimientos de los brazos.
  - Jain y Kemp [13] logran coordinar ambos movimientos con suficiente control para abrir puertas y gavetas de dimensiones desconocidas. La trayectoria del efector final es guiada usando el concepto de "Control de Punto de Equilibrio". La base responde al movimiento del brazo cuando este se acerca al límite de su campo de trabajo.
  - Gong y McInnes [20] buscan que ambos tipos de movimiento sean simultáneos, su estrategia para lograrlo es que el subsistema de locomoción obedezca a la posición deseada del efector final utilizando un control de alto nivel. Este además asegura que siempre se cumpla la estabilidad (que el robot no corra el riesgo de voltearse) y de manipulabilidad (cercanía a las singularidades). Estos dos criterios se utilizan como base para la coordinación de los manipuladores móviles en numerosas investigaciones, como [24] y [23].
    Esa coordinación se logra buscando soluciones posibles mediante cinemática inversa, en caso de no existir, se concluye que es necesario indicarle a la base que se mueva en la dirección del objetivo. Un controlador de estado se encarga de coordinar entre las cuatro criterios ya descritos: estabilidad, manipulabilidad, movimiento del brazo y de la base.

Los controladores de bajo nivel tanto del manipulador como de la base son independientes.

- **Modelos de control unificado**: lo que se busca aquí es que haya una relación estrecha y completamente simultánea entre la movilidad y la manipulación, sin embargo es altamente complejo porque implica el manejo de redundancia tanto en la configuración del robot como en su control [21]. El control unificado se logra al definir una única cadena cinemática redundante que incluye tanto la base como el manipulador, utilizando los métodos tradicionales de cinemática directa y diferencial. A continuación se muestran algunas estrategias referentes de como se abarca el problema:

  - Seraji [25], con el fin de limitar los r grados de libertad redundantes, propone el uso r nuevas relaciones cinemáticas a escogencia del usuario, de forma que se pueda crear una nueva ecuación de "cinemática diferenciada aumentada". Bayle [26] hace una extensión de este trabajo con un método para ampliarlo a un grupo más general de robots. Además muestra como usar las relaciones cinemáticas adicionales para evadir obstáculos.
  - Sharma and Scheurer [21] resuelve el problema de cinemática inversa al abstraer los grados de libertad redundantes como $r = n - m$ movimientos independientes. Con ellas define una relación analítica entre las posiciones cartesianas del efector final y las posiciones de las articulaciones.

  Es importante hacer resaltar el concepto de **Control de Cuerpo Completo** que en la literatura es conocido como *Whole-Body Control*. Este implica el uso de los modelos de control unificado, pero dándole prioridad a otras tareas de control, como la evasión de obstáculos o la estabilidad [27]. Los siguientes son ejemplos del uso de dicho concepto de diseño:

  - Mantian Li *et al.* [27] propone el uso de varias tareas de control que se satisfacen a través de proyecciones en el espacio nulo del efector final, juntas estas pueden superar incluso los grados de libertad redundantes disponibles. Para lograrlo se programan mecanismos de activación y desactivación, de forma que no se supere la cantidad máxima que el sistema permite.
  - Iskandar *et al.* [28] prioriza el uso de los grados de libertad de la base estableciendo límites fijos para el efector final relativos al robot. Similar a como sucede con Jain, cuando el final del manipulador llega a estos puntos se obliga a la base a movilizarse. Todo se realiza utilizando control por impedancia.

2.3.3. Control dinámico

A pesar de que es posible controlar el movimiento de un robot limitándose a su cinemática, es necesario conocer cómo se comportan las fuerzas en su estructura móvil para asegurar que el equipo es capaz de soportar las tareas. Por otro lado, un modelo dinámico lo suficientemente robusto es una estrategia viable para desarrollar un sistema de coordinación de un manipulador móvil.

2.3.4. Orocos-KDL

Orocos es un proyecto de software libre para el control de robots (Open Robot Control System). Una parte fundamental para el control de elementos mecánicos es poder modelar su comportamiento. Para ello Orocos desarrolló KDL (Kinematics and Dynamics Library), que es su librería para modelar cadenas cinemáticas y dinámicas. En esta sección se detallará esta librería específicamente debido que es la que se utiliza en el ARCOS-Lab y por tanto en este proyecto, sin embargo existen otras librerías disponibles.

Elementos geométricos

A continuación se detallarán algunos de los objetos que la librería ofrece, los cuales coinciden con elementos de la sección 2.2.2:

- `Vector`: es un vector de tres elementos que describe una posición en el espacio.
- `Rotation`: es el equivalente a una matriz rotacional, por lo que se puede definir a partir de los ángulos de Euler o de un vector y un ángulo.
- `Frame`: es una matriz de transformación homogénea, por lo que se define a partir de los dos elementos anteriores.
- `Twist`: es un vector 6x1 que expresa la velocidad traslacional y rotacional.
- `Wrench`: es un vector 6x1 que expresa la fuerza y el momento que se aplica sobre un punto
- `Segment`: es un modelo de un eslabón, por lo que relaciona dos marcos de referencia: uno en la articulación inicial (`F_Reference`) y otra en al final (`F_Tip`). Se puede definir por medio de los siguientes argumentos:
  - `Joint`: articulación o par cinemático de un solo grado de libertad con el cual se relacionan dos segmentos. Los hay de dos tipos, rotacional y traslacional. Por defecto está ubicada sobre el `F_Reference`.
  - `Frame`: pose del `F_Tip` relativa a la del `F_Reference`.
  - `RigidBodyInertia`: se compone de un vector (`V_cog`) que ubica el centro de gravedad del eslabón, un marco en ese mismo punto que establece su inercia rotacional (`F_cog`).
- `Chain`: es la concatenación de los segmentos, donde se hacen coincidir los marcos de referencia (`F_Reference`) de uno con el final (`F_Tip`) de otro. Cada segmento se agrega con la función `AddSegment()`.

Comparación con otras librerías

- **IKFast**: se ofrece como parte de ROS, dando soporte mediante soluciones analíticas, no obstante, presenta algunas limitaciones: (i) sus algoritmos no están disponibles, salvo en binario, por lo que es difícil de modificar; (ii) es difícil de usar fuera de ROS y (iii) el control se limita a trayectorias punto a punto [14].

- **Track-IK**: se desarrolla posterior a Orocos-KDL, como otra alternativa numérica para el problema de la cinemática inversa, con el fin de solucionar algunas de sus limitantes: (i) tolerancias en el proceso de solución y (ii) problemas con los límites de las articulaciones [29]. Al igual que KDL es software de código abierto.

Métodos de solución

- Cinemática Directa: responde a `ChainFkSolverPos_recursive`. Utiliza un procedimiento recursivo donde soluciona la posición de cada marco según el ángulo del segmento anterior. El resultado se obtiene mediante la acción `JointToCart()` que tiene como entradas un vector del tipo `JntArray` (que contiene los ángulos de cada articulación) y una cadena
- Cinemática Inversa: responde a `ChainIkSolverVel` y `ChainIkSolverVel`, para velocidad y posición, respectivamente. Utiliza métodos numéricos, lo que le permite ser aplicado a una variedad mayor de manipuladores, aunque requiere de más cómputo para lograrlo [29].

2.3.5. Evasión de obstáculos

El propósito de esta área de investigación es el uso de los sistemas de percepción para evitar los obstáculos inmediatos que surgen a lo largo de las trayectorias que describe un robot. A diferencia de la planeación de movimiento, la cual conoce al robot y al medio de primera entrada, aquí se busca desarrollar la capacidad de enfrentar problemas desconocidos y locales [30].
Es importante aclarar que muchos trabajos en esta área están pensados solo desde la perspectiva del desplazamiento, lo que hace factible considerar al robot como una partícula que se mueve en un plano. En el caso de este proyecto se debe considerar la configuración completa del robot y no solo la trayectoria del efector final. Este contraste se hace evidente en la figura 2.14 Además de que se debe trabajar en un espacio tridimensional.
A continuación se van a mencionar algunas estrategias:

Histogramas de campos vectoriales

Está pensado para trabajar en distribuciones probabilísticas de obstáculos. Una función crea un mapa (figura 2.13) que indica la probabilidad de encontrar un obstáculo y su cercanía para cada una de las posibles direcciones que el robot puede tomar. Esto se puede graficar para encontrar picos y valles. Existen diversos criterios para escoger entre todos los valles disponibles, siempre tomando en cuenta la posición del objetivo.

Acercamiento de la ventana dinámica

Se lleva a cabo no en el *workspace*, sino en el *control space* (las velocidades del robot). Se da en dos etapas, una que define un conjunto candidato de velocidades tomando en cuenta i) límites de velocidad, ii) que se pueda lograr rápidamente según el estado actual y iii) que haya posibilidad de abortar al usar la máxima desaceleración, como se ve en la figura 2.13
El segundo paso es buscar el conjunto de velocidades que maximiza una función dada.

Penalización con problemas de optimización

Este método usado por [31] está diseñado para usarse en manipuladores. Utiliza un algoritmo para determinar la distancia entre los eslabones de su manipulador y las superficies de un modelo tridimensional del obstáculo. Los ángulos en cada articulación que llevan a que esta distancia disminuya se desestimulan. Para determinarlos se hace una optimización de la función de distancia, con las que se encuentran las configuraciones que provocan las menores distancias.

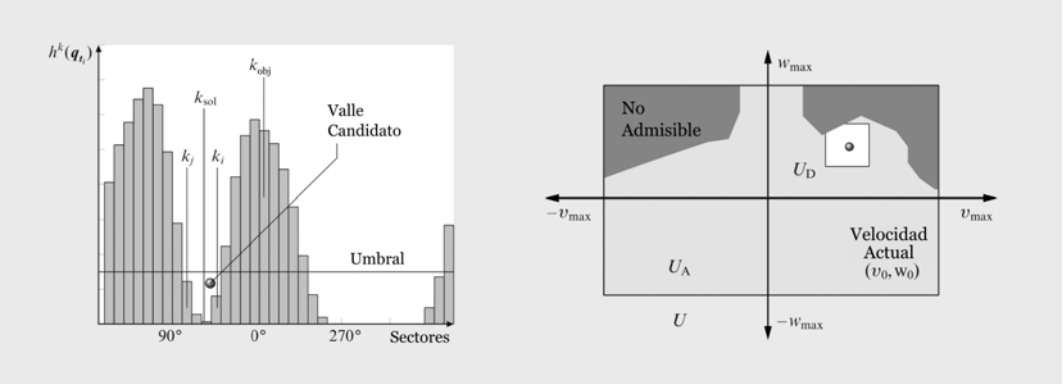


Figura 2.13: A la izquierda una representación de los Histogramas de Campos Vectoriales, donde $h^k(q_t)$ representa la probabilidad de encontrar un obstáculo y k las direcciones. A la derecha una representación de los conjuntos U de velocidades en el método de Acercamiento de la Ventada Dinámica. U_A es el conjunto de velocidades seguras, donde el robot es capaz de detenerse antes de encontrar un obstáculo, U_D son las velocidades que se pueden obtener rápidamente según la aceleración actual.

Fuente: Elaboración propia a partir de [30]

Método de campos potenciales o *vector fields*

La dirección que toma el robot depende solo de su estado actual. Este se modela como una partícula; los obstáculos y el objetivo actúan sobre él como puntos de repulsión y atracción, respectivamente. El movimiento es resultado de la fuerza virtual que suman todos los elementos en el medio. Se puede plantear el algoritmo para que considere solo la configuración del robot o también su velocidad y aceleración.
Una aplicación que se asemeja al objetivo de este proyecto es la de [32]. Aquí se define una función de repulsión para los obstáculos. Esta se activa cuando algún punto del manipulador se acerca más allá de una distancia mínima. La función de cercanía se aplica solo a ciertos puntos del manipulador, con el fin de reducir la necesidad de cómputo.
Este método es el que se utiliza en el ARCOS-Lab, ya que tiene numerosas ventajas que lo hacen atractivo para los objetivos del robot humanoide que se está desarrollando [11]:

- Por su facilidad de procesamiento responde rápidamente a obstáculos móviles, los cuales son comunes para robots que interactúan con humanos.
- Se puede adaptar más fácilmente al control por impedancia, ya que no debe recalcular rutas cuando un humano detenga o mueva el efector final.

Su principal problema son los mínimos locales, que pueden llegar a atrapar al manipulador sin haber llegado a su objetivo [32].

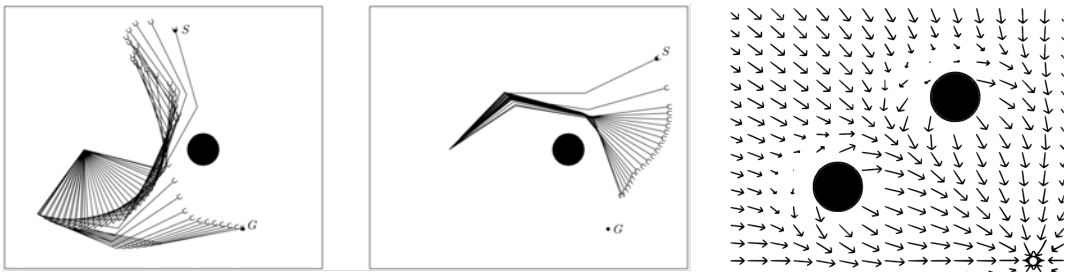


Figura 2.14: Los dos recuadros de la izquierda muestran el uso de los *Vector Fields* en el control de un manipulador. El efector final debe llegar del punto S al punto G sin que ningún punto del brazo toque el obstáculo negro. A la derecha se ve el campo vectorial de un caso con dos repulsores.

Fuente: [5]

Implementación de los *vector fields* en el ARCOS-Lab

El uso de este método en el robot del ARCOS-Lab viene desde el desarrollo de la primera versión del control del brazo humanoide del laboratorio en [11]. Aquí la evasión de obstáculos está integrada con el control de movimiento del sistema, tal como se muestra en la figura. Los objetivos y los obstáculos alimentan una función que define un campo vectorial de velocidades. A partir de las coordenadas actuales del efector final, x_a, se soluciona la función para obtener un vector de velocidad deseado $v(\dot{x})_d$. Por medio de un algoritmo de cinemática inversa de velocidades se traduce este valor al *joint space* en $\dot{\theta}_d$ y se le da como comando a los actuadores. El ciclo se cierra al traducir la nueva posición al *task space* con un módulo de cinemática directa de posición.

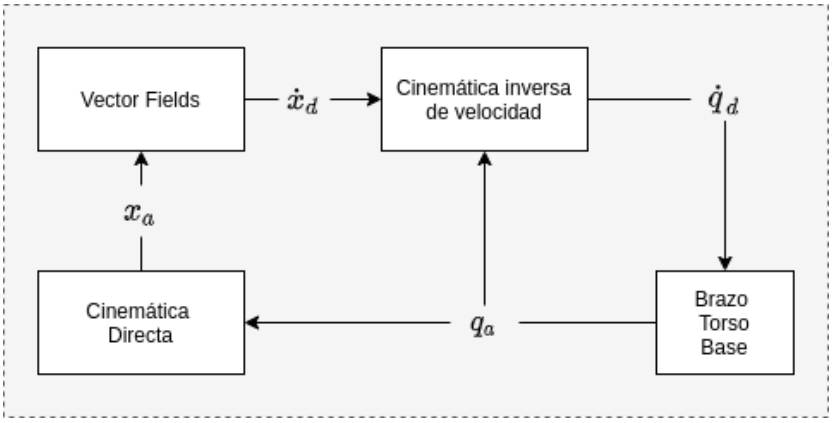


Figura 2.15: Ciclo de funcionamiento del sistema movimiento con evasión de obstáculos, VF-CLIK

Fuente: [11]

2.4. Software del robot humanoide del ARCOS-Lab

2.4.1. Arquitectura de software

Estructura

Como explica [3], generalmente los laboratorios que trabajan en manipulación siguen una estrategia similar para estructurar la cognición del robot, por lo cual emplean usualmente los mismos subsistemas; este es el caso del ARCOS-Lab también. A continuación se describen en su caso particular:

- Planeador de Acciones: es quien recibe las instrucciones de alto nivel del usuario y las interpreta; en este caso es quien las traduce en cadenas de tareas atómicas.
- Controladores de los Actuadores: se encargan de llevar a cabo los comandos recibidos para cada uno de los actuadores, como por ejemplo mover el brazo para llevar el efector final de un punto a otro.
- Sistemas de Percepción: capta e interpreta la información del medio, del objeto a manipular y del robot para determinar el estado de cada uno.
- Visualización y Recolección de Datos: son los módulos encargados de guardar información sobre las tareas desempeñadas con el fin de aprender de ellas, ya sea de forma autónoma o para análisis por parte de los desarrolladores.
- Sistema de Conocimiento: almacena y provee información referente a los objetos y a las acciones a los demás subsistemas, pero puede usarse también para datos de otro tipo.
- Sistema de Manejo Físico: es una propuesta particular del ARCOS-Lab, que lleva el nombre de *Object Model System* o Sistema de Modelos de Objetos (OMS) (**Object Model System**), y que, como se explica atrás, apoya al sistema de planeación al "organizar, seleccionar y ejecutar"los modelos de manipulación de los objetos.

Principios de diseño

La realización de tareas de forma autónoma es parte fundamental de la robótica al servicio del humano, ya que debe desempeñarse en medios poco estructurados y responder a instrucciones abstractas, donde una indicación puntual puede implicar la necesidad de realizar una cadena de tareas más pequeñas en un orden determinado. Existen diversas formas de abarcar este problema.
En el caso del robot del ARCOS-Lab se decidió estructurar el razonamiento de instrucciones basado en tres importantes decisiones de diseño [3]. La primera es la concatenación de acciones indivisibles. Su premisa es que una forma de realizar cierta tarea es encontrando la serie de etapas que la componen, las cuales a su vez podrían subdividirse. Sin embargo, existen pasos o **acciones atómicas** que ya no se pueden dividir y que resultan ser más fáciles de modelar y que al acomodarlos a conveniencia pueden crear numerosas tareas de alto nivel.
El segundo es que las acciones y los objetos, desde una perspectiva humana, están estrechamente relacionados y que por tanto puede resultar ventajoso no separarlos. De esta forma cada objeto está definido mediante una serie de posibles acciones que se le pueden aplicar. Dentro de la arquitectura de control del robot a esto se le llaman **entidades** y están pensadas

para contener información no solo de las posibles formas de interactuar con el objeto, sino también parámetros para elegir entre ellas, los modelos para hacerlo y vías para desarrollarlas. Es decir, asumen la responsabilidad de tomar las decisiones cognitivas de un objeto específico a la hora de interactuar con él para cumplir un objetivo.
Por último, y de forma complementaria, estas acciones se definen a través de **parejas de control y predicción** (CPP), es decir usando dos modelos, uno inverso y otro directo, respectivamente. Son analíticos y orientados a los objetos, no al manipulador (según fue desarrollado por [1]). Y están diseñados para trabajar de manera conjunta. Cada uno de estos CPP corresponden a una posible acción atómica dentro de una entidad.
De esta forma las tareas que el robot puede realizar se construyen a partir de la apropiada selección de estas acciones. Las instrucciones son procesadas y subdivididas por el sistema de planeamiento, quien a su vez le delega al Sistema de Modelo de Objetos (OMS) la ejecución de las acciones atómicas correctas. Es ahí donde se almacenan las entidades de cada objeto.

Figura 2.16: TUM-Rosie manipulando una caja mediante un modelo del objeto

Fuente: [2]

2.4.2. Modelos de objetos

A pesar de que en este trabajo no se contempla la manipulación, sus resultados pueden extenderse para este fin. Además, desde el punto de vista del control de este robot, el movimiento del cuerpo siempre va a estar ligado a la acción que se quiere realizar con el objeto a manipular [11].
En contraste con las estrategias que utilizan *machine learning* para darle a los robots la capacidad de realizar tareas con objetos, en el ARCOS-Lab se trabaja con modelos físicos. Estos no se desarrollan por métodos de aprendizaje, sino que son creados por el humano, dejándole al robot solamente la identificación de ciertos parámetros. Trabajar de esta manera reduce el tiempo que se requiere para lograr tareas complejas y facilita extender la aplicación a acciones ya logradas [1]. Un prueba de esto se ve en la figura 2.16.

2.4.3. Sistema de control

Es importante resaltar que este diseño está basado en el trabajo hecho por Ruiz-Ugalde en [11], por lo que comparte la mayoría de sus características. El objetivo de esa publicación fue demostrar que el uso de tareas atómicas orientadas a los objetos, acompañado con modelos que describan cómo llevar a cabo dichas acciones, ayudarían a desempeñar tareas complejas de control, en su caso empujar una caja con un solo dedo.

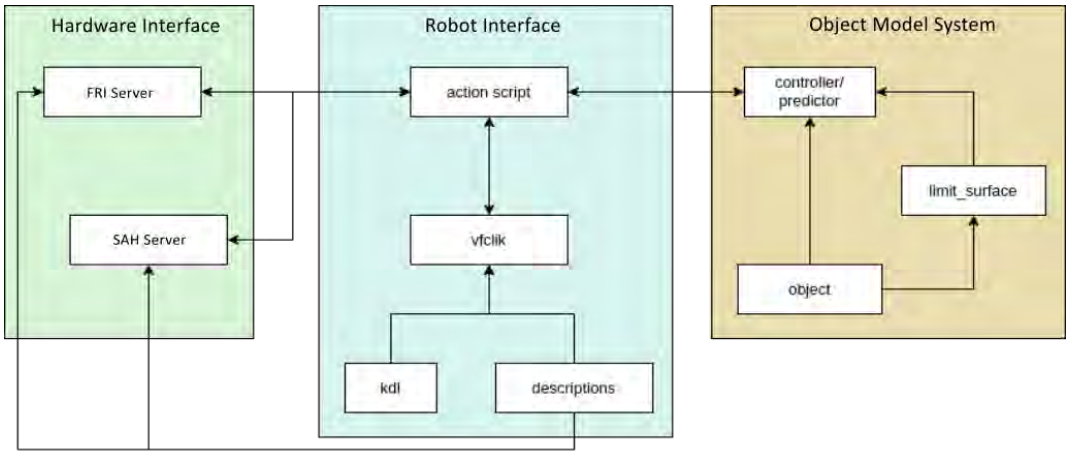


Figura 2.17: Estructura del Sistema de Control del robot humanoide del ARCOS-Lab.

Fuente: Elaboración propia, basado en [12]

La forma particular en que los elementos de control se relacionan para realizar movimientos que resuelvan tareas, se logra por medio de tres grandes módulos, tal como se muestra en la figura 2.17. Sus partes se describen a continuación:

Robot Interface

Es la parte central del control, donde se corren las secuencias de acciones. Se compone de tres principales elementos:

- **Action Script**: o bien, código de acción, es quien le indica al OMS cuáles son las acciones que se deben llevar a cabo, y quien recibe de vuelta los comandos para los actuadores.
- **VFCLIK**: recibe su nombre de las siglas en inglés para Cinemática Inversa con Campo Vectorial de Lazo Cerrado. Es utilizado por el Action Script para determinar las velocidades de las articulaciones del brazo requeridas para lograr determinada velocidad en el efector final, en función de la *pose* actual y la *pose* deseada (ver sección 3.2). Por tanto recibe instrucciones orientadas a los objetos (*en el task space*) de parte del OMS y debe traducirla a comandos en el *joint space* [11].
  La forma en que trabaja este módulo y cómo se integra con la evasión de obstáculos se explica con mayor detalle en la sección 2.3.5

- **Kinematic Descriptions**: como dice su nombre, es el código donde se define la descripción o configuración física del robot, es decir las cadenas o árboles cinemáticos que lo modela.

Estos dos últimos elementos hacen uso de la librería OROCOS-KDL (*kdl* en la figura 2.17), que como se explica en la sección 2.3.4, se encarga de realizar los cálculos de cinemática inversa.

Hardware Interface

Este módulo administra la información que se dirige y viene del hardware del robot, por tanto se compone principalmente de controladores para cada uno de los actuadores, así como interpretadores de las señales que generan de los sensores (tanto los de percepción como los que están integrados en los brazos, manos, por ejemplo). Algunos de los principales componentes (figura 2.17) son los siguientes [11]:

- **FRI Server**: *Fast Research Interface* (Interfaz Rápida de investigación), es quien controla los brazos y obtiene información sobre la posición y torque de las articulaciones.
- **SAH Server**: Se encarga de traducir los comandos para controlar la mano.

Object Model System

A través de este módulo se alivia la carga cognitiva que le corresponde al sistema de planeación, el cual solo tiene que encargarse de interpretar la instrucción maestra para separarla en tareas atómicas y comunicársela al OMS. Es aquí donde se almacena y administra -en forma de entidades- la información física que es pertinente para cada posible acción, y se escogen y ejecutan los modelos correctos.
Es de esta forma que se cumple el principio con el cual fue pensado el OMS: proyectar información importante de las instrucciones de alto nivel en las instrucciones de bajo nivel en los motores a través de modelos matemáticos de los objetos y del robot [11]. Un ejemplo de esto se ve en la figura 2.17, donde un objeto se modela (limit_surface) y controla dentro del OMS para luego mandar sus comandos al hardware.

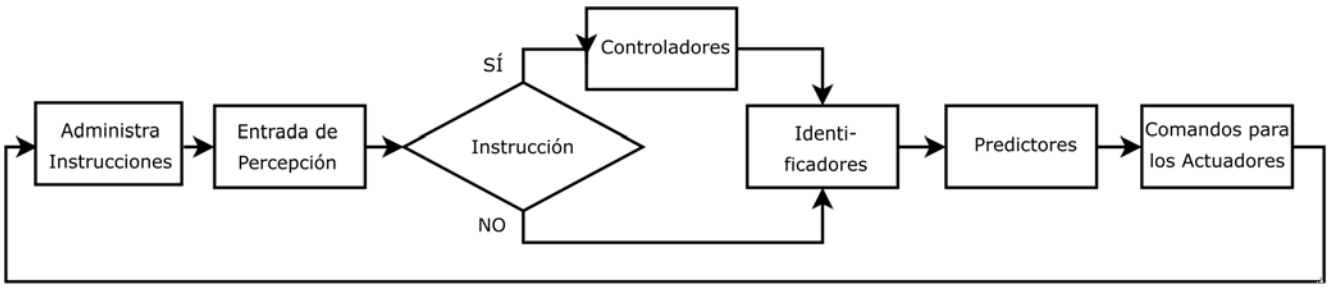


Figura 2.18: Ciclo de control del OMS

Fuente: Elaboración propia, basado en [3]

El manejo final de los actuadores queda en manos del OMS, aunque siempre a través de la comunicación con los demás sistemas. La secuencia de trabajo se ve en la figura 2.18: analiza las instrucciones del Planificador, recibe la información del Sistema de Percepción, determina el estado de la acción, ejecuta los controladores (en caso de ser necesario), identifica el predictor correspondiente, y ejecuta los comandos que serán entregados a los actuadores. Esto se logra a través de distintas interfaces que le permiten comunicarse con los demás sistemas mencionados, pero el OMS es el encargado directo de administrar los modelos de control y de predicción.

Middlewares

La comunicación entre los distintos programas que se ejecutan en el sistema de control requiere de una plataforma de transmisión de datos, lo que comúnmente se llama *middleware*. Aquí se hace uso de dos: YARP (*Yet Another Robot Platform*), para el control por medio de VFCLIK; y ROS (*Robot Operating System*), para el sistema de percepción.

2.4.4. Simulador del robot humanoide del ARCOS-Lab

Con el propósito de poder probar el comportamiento del robot sin la necesidad de poner en riesgo todo el equipo, además de reducir el tiempo de ejecución, resulta útil tener un modelo que determinar cuál será su respuesta ante las instrucciones que se le den. Para ello el controlador tiene integrado un conjunto de predictores que simulan el torque en los *joints* y la respuesta de los objetos por medio de sus modelos. Todo esto al nivel del *Hardware Interface* (figura 2.17).
El diseño actual fue desarrollado en conjunto con el sistema de control por [11]. Este además incluyó una interfaz gráfica que permite ver los movimientos en tiempo real llamada PYROVITO (*Python Robot Visualization Tool*). Capturas de su funcionamiento se pueden ver en la siguiente figura.

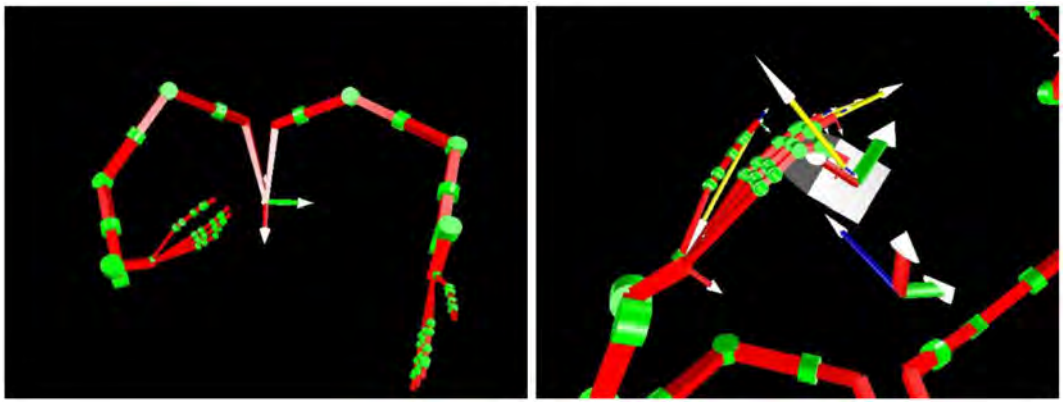


Figura 2.19: Capturas de PYROVITO, la herramienta de visualización del simulador

Fuente: [11]

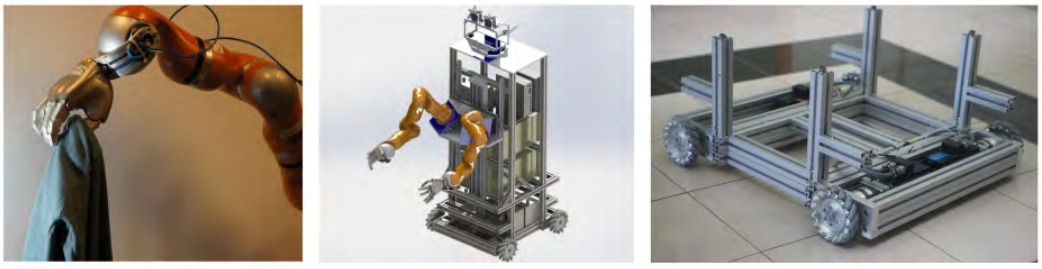
2.5. Hardware del robot humanoide del ARCOS-Lab

2.5.1. Cuerpo del robot humanoide

El robot se compone de:

- Una cabeza expresiva equipada con sensores de nubes de puntos, cámaras de alta definición y térmica. Está montada sobre un cuello con 2 DoF.
- Un cuerpo rígido que contiene las baterías y computadoras que le permitirán funcionar de manera autónoma.
- Dos manos Wessling Robotics
- Dos brazos de 7 DoF
- Un torso móvil
- Una base móvil omnidireccional

Los últimos tres elementos, que son los que tienen un papel fundamental en el desarrollo de este proyecto, se describen con mayor detalle en las secciones 2.5.2 - 2.5.4.



(a) (b) (c)

Figura 2.20: El robot humanoide del ARCOS-Lab. En (a) el brazo robótico KUKA LBR y la mano Wessling Robotics, en (b) una imagen preliminar del robot y en (c) una versión inicial de la base omnidireccional

Fuente: Elaboración propia, basado en [12]

2.5.2. Plataforma omnidireccional

Las bases móviles son muy deseables porque aumentan la capacidad de manipulación de los brazos y a la vez proveen nuevos grados de redundancia. Esto, a pesar de aumentar la complejidad del control, es muy deseable cuando se quieren llevar a cabo tareas que pueden requieren estar pendiente aspectos en forma paralela, como estabilidad, auto colisión, evasión de obstáculos y destreza para manipular.
Con el propósito de que el robot pueda desenvolverse en medios no estructurados y nuevos, resulta muy útil que su movilidad este libre de restricciones, como sí ocurre con las bases no holonómicas. Esto le permite desplazarse en cualquier dirección en todo instante, lo que resulta un alivio para los módulos que definen las rutas y el movimiento de cuerpo completo.

Esta misma característica se aplica en otros robots conocidos, como el PR2 y el TUM-Rosie. Se compone de cuatro ruedas Mecanum, cada una montada en una viga con sensores para la medición de fuerzas, con el fin de darle capacidad de ser controlada por impedancia. Una versión inicial se puede ver en la figura 2.20.

2.5.3. Torso móvil

La razón de este grado de libertad es imitar la capacidad del humano de agacharse, ya que resulta indispensable para lograr tareas en ambientes donde hay objetos almacenados, como los que se describieron en la sección 2.1.3. En este caso se logra mediante un tornillo sin fin vertical al cual está unido un carro móvil en el que descansan los brazos, tal como se ve en la figura 2.20-(b). Esta es una forma fácil de dar esta movilidad, que contrasta con las tres articulaciones de revolución del Rollin' Justin [8]

2.5.4. Brazos

Sobre los hombros se ubican dos brazos KUKA LBR 4+ de 7-DoF, visibles en la figura 2.20. Este equipo de alta tecnología es muy popular en esta área de investigación por su ligereza y por contar con sensores de torque en todas sus articulaciones, lo que les permite ser manejado por control por impedancia, tal como hace [11]. Esto resulta una característica de gran importancia en lo que respecta a la seguridad del humano cuando el robot desempeña tareas colaborativas, ya que le permite reaccionar a fuerzas imprevistas de manera suave. Otro ejemplo de su uso es [8].
Otro atractivo es el grado de libertad redundante. Al manipular la pose de objetos, que se define por 6 DoF tener un grado extra le permite al manipulador contar con infinitas formas de llevar su efector final a una posición y orientación definida.
Cabe resaltar que el posicionamiento de los brazos obedece a un estudio donde se determina la posición óptima para favorecer la manipulabilidad del conjunto [4].

Capítulo 3

Estudio del sistema de control del ARCOS-Bot

3.1. Instalación del simulador

El simulador se compone de una serie de paquetes que corren en conjunto, asociadas a su vez de numerosas dependencias. Para asegurar de que el simulador se ejecute de forma estable, es necesario llevar a cabo una instalación cuidadosa siguiendo ciertos pasos. Estos se basan en las instrucciones oficiales del ARCOS-Lab [33], sin embargo, cuentan con algunos cambios.
Con el fin de poder asegurar la repetibilidad de este proyecto se describen a continuación algunos cuidados importantes a la hora de realizar la instalación:

- En caso de que no se vaya a utilizar un ambiente virtual de Python para albergar el proyecto, es necesario realizar la instalación de YARP indicando que se ejecuten los *bindings*, los cuales le permiten a Python utilizar bibliotecas de C++.
- Asegurarse de estar utilizando una versión de YARP estable.
- Verificar la instalación correcta del paquete de OROCOS-KDL.
- Tener consistencia en el uso de administradores de enlaces simbólicos (*symlink*), como *xstow* y *stow*. Puede ser necesario verificar los archivos de construcción de cada paquete.

3.2. Generalidades

Para el sistema la variable a controlar es la posición del efector final, sin embargo, como se verá, gran parte de los cálculos se basan en datos de velocidad. La implementación llevada a cabo para el ARCOS-Bot sigue el esquema de la figura 2.15. Como puede verse este es un ciclo de control cerrado, donde las entradas son la posición deseada, y la salida, el cambio a realizar. No obstante, a pesar de que el control de alto nivel se basa en la posición, la solución se realiza con cinemática inversa de velocidades (*Velocity Inverse Kinematics*, VIK), basado en la velocidad que dicta el *Vector Field* para la posición actual. El proceso es el siguiente:

1. Basado en la tarea que se está llevando a cabo, el sistema de alto nivel define cual es la posición final deseada para el efector final, p_f.

2. Se leen las posiciones actuales de todos los grados de libertad y por medio de una función de cinemática directa se calcula la posición actual del efector final, p_a.

3. Utilizando estos dos datos se computa un campo vectorial en el espacio circundante que define una velocidad hexadimensional para cada una de las coordenadas tridimensionales. Ver más detalles en la sección 3.3.

4. A partir de esto se puede definir una velocidad futura v_f según el valor calculado de p_a.

5. Por medio de una función de cinemática inversa de velocidades, se encuentra un vector $\dot{q}_f$ con las velocidades angulares de cada articulación que satisfacen la velocidad v_f. Para lograrlo se calcula la matriz jacobiana en ese instante, para lo que hace uso del resultado del paso 2.

Hasta aquí se han llevado a cabo las tareas de control, pues estos comando de velocidad son comunicadas a los controladores de los motores del brazo. Sin embargo, cuando se hace uso del simulador es necesario realizar una predicción de la posición resultante del proceso, esto se ve con más detalle en la sección 3.4. Este paso completa el ciclo, puesto que el brazo habrá adoptado una nueva posición, en la cual se debe satisfacer una nueva velocidad.

3.3. Uso del *vector fields*

Los vectores de velocidad en cada posición del campo son el resultado de la función de varios criterios. Los dos más importantes son la posición de los obstáculos y del objetivo. El atractor tiene un efecto global y constante, es decir, define vectores apuntando en su dirección en todo el espacio, todos de una misma magnitud. Ya que no importa la ubicación relativa del efector final, puesto que siempre se quiere llegar al objetivo con la misma intensidad. Los repeledores, por otro lado, tienen una acción local, ya que solo cobran importancia cuando existe un riesgo de colisionar con ellos. Los vectores de este mapa se crean proporcionales a la distancia, alejándose del punto conflictivo. La figura 3.1 muestra como se verían ambos campos, los cuales finalmente se suman.

Con el fin de evitar que la inercia del brazo lleve a una oscilación en la llegada al objetivo, o por lo contrario a que frene de forma abrupta, se modifica el campo del atractor a cierto radio del objetivo. A partir de este límite los vectores serán proporcionales a la distancia.
Las velocidades angulares se trabajan de una forma más simple. Se desea que el efector final siempre busque orientarse hacia el objetivo, por lo que los valores rotacionales de la velocidad, para cada posición, son aquellos que lleven a que el vector que define la dirección actual se oriente con el vector que une al efector final con el objetivo.

3.4. Modelos de predicción

No existe un elemento de control responsable de calcular las posiciones actuales, sino que esta tarea depende únicamente del codificador rotatorio de cada motor. O bien, de sus controladores, en el caso de la base omnidireccional, donde la posición angular del motor no está directamente relacionada con el estado del grado de libertad.
Como el motor se está controlando por velocidad, la posición resultante después de cada pequeño paso de control va a depender del estado dinámico del brazo durante ese lapso de

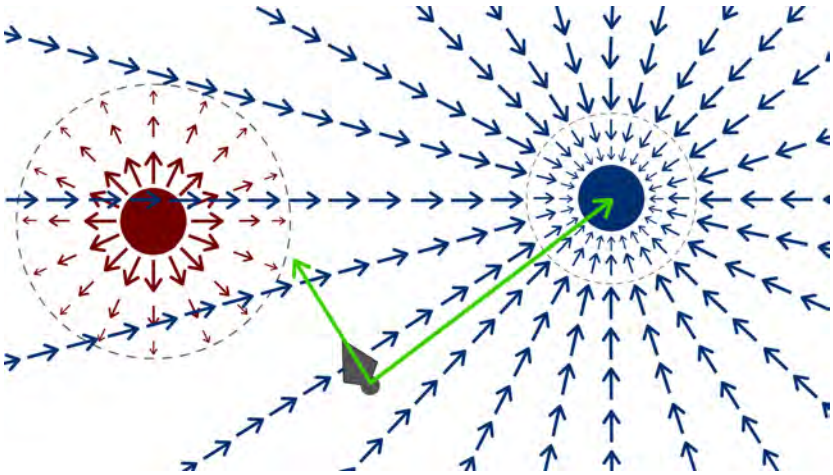


Figura 3.1: Campos vectoriales creados por el objetivo (en azul) y por un obstáculo (en rojo). Los vectores verdes, que representan la orientación actual de la mano y la dirección óptima de llegada a la meta, se utiliza para computar las velocidad angulares.

Elaboración propia, basado en [11]

ejecución. Sumado a esto, está el error inducido por elementos externos como: obstáculos, una eventual intervención humana, o cualquier otra fuerza no contemplada. En el robot real no hay necesidad de realizar un cálculo para determinarla, sino que se puede leer directamente a través de un sensor. Por otro lado, cuando se está simulando, es necesario tener un modelo físico que de alguna manera prediga la posición resultante.
Debido a esto, es indispensable tener modelos físicos que, con base en la señal de control para cada motor (velocidad instantánea deseada), determinen la posición resultante. Actualmente cada motor se modela por separado a través de una ecuación de la dinámica rotacional de un eje con una inercia, fricción, un torque de entrada y uno de salida. Por medio de esta ecuación diferencial se puede aproximar la posición de cada articulación. Una representación gráfica se muestra en la figura 3.2.

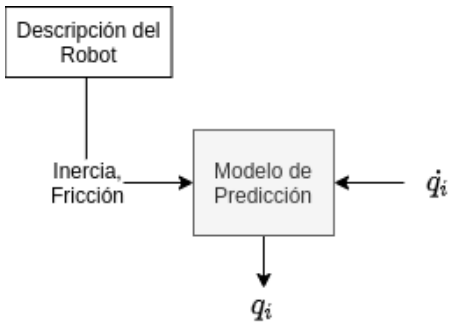


Figura 3.2: Modelo de predicción para las articulaciones de revolución

Elaboración propia, basado en [11]

3.5. Cinemática inversa de velocidades

El método implementado en el robot del ARCOS-Lab hace uso de esta tarea, en la que se calculan un grupo de velocidades para cada articulación para lograr una velocidad y velocidad angular deseadas.

3.5.1. Uso de la matriz jacobiana en la cinemática inversa

La cinemática inversa se encarga de encontrar las combinaciones de posiciones o velocidades (q o $\dot{q}$) en cada uno de las articulaciones, que corresponden a una *pose* (x) o *twist* (v_e) en el espacio cartesiano del efector final. Este problema podría representarse de la siguiente forma:

$$x = f(q) \tag{3.1}$$

Sin embargo, no siempre se puede encontrar una relación cerrada entre ambos vectores, o aún si existiese puede ser muy complejo, por lo que se recurre a métodos aproximados. Una forma de lograrlo por métodos iterativos, los cuales hacen uso de una linearización por medio de la matriz jacobiana [34]. Esto sería similar a lo que se lleva a cabo en el método Newthon-Rhapson [35], de modo que:

$$x \approx J(q)q \tag{3.2}$$

Las entradas de la matriz jacobiana pueden calcularse a partir de la posición actual de cada uno de las articulaciones [5], por lo que resulta útil plantear la ecuación 3.2 de la siguiente forma:

$$\Delta x \approx J(q)\Delta q \tag{3.3}$$

Según este enfoque se buscaría un Δq de lleva a que Δx se asemeje al error e con respecto a la pose deseada [34].

$$e = J(q)\Delta q \tag{3.4}$$

Otra posible estrategia es controlar no las posiciones, sino más bien las velocidades (cinemática inversa de velocidades) y así utilizar directamente la ecuación con la que se define la matriz jacobiana:

$$v = J(q)\dot{q} \tag{3.5}$$

Sin embargo, ambas estrategias tienen el problema de que J no siempre invertible. Además de que naturalmente va a tener problemas cerca de las singularidades [34].
Una posibilidad es utilizando la pseudoinversa de J (también conocida como inversa de Moore-Penrose) que a pesar de que sigue teniendo problemas cerca de las singularidades, se puede plantear en casos donde no existe una inversa. Se puede plantear de la siguiente forma [35]:

$$\dot{q} = J^{\dagger}v = J^{T}(JJ^{T})^{-1}v \tag{3.6}$$

Este método es análogo a resolver el problema de mínimos cuadrados para la ecuación 3.4, es decir:

$$\min_{\dot{q}}(\|v - J\dot{q}\|) \tag{3.7}$$

Esto abre la posibilidad de llevar a cabo tareas secundarias, puesto que la matriz $(I - J^{\dagger}J)$ proyecta en el espacio nulo de J [34].

3.5.2. Cómo lidiar con las singularidades

Las singularidades representan un problema para el desempeño del robot debido a que tienen implicaciones en la forma de la matriz Jacobiana. Y por tanto el espacio cartesiano se verá seriamente afectado. La primera razón es que esto implica que el sistema en 3.5 tendrá dos o más ecuaciones linealmente dependientes, y la solución dependerá del rango de J. La segunda fuente de error, puede implicar que el valor de $\dot{q}$ obtenido requiera altas velocidades para alguna articulación [36].
Para evitar este problema, en el ARCOS-bot, se utiliza el método basado en los *Damped Least Squares* (Mínimos Cuadrados Amortiguados). Aquí se busca no solo encontrar la mejor aproximación de $v = J\dot{q}$, sino que se agrega un término que ayuda a disminuir la magnitud de $\dot{q}$:

$$\dot{q} = J^T(JJ^T + \lambda^2 I)^{-1}\dot{x} \tag{3.8}$$

En otras palabras, se busca minimizar la siguiente condición [34]:

$$\min_{\dot{q}}(\|v - J\dot{q}\|^2 + \lambda^2 \|\dot{q}\|^2) \tag{3.9}$$

De esta forma el factor $\lambda \geq 0$ determina la medida en que se favorece la exactitud de la solución (término de la izquierda) o la capacidad de evitar picos de velocidad cerca de las singularidades (término de la derecha). Por tanto, se escoge λ de una forma cuidadosa, según la necesidad. Si λ es nulo, entonces la solución se reduce a invertir J.
Esta estrategia se ve beneficiada si se realiza una decomposición de valores singulares, conocido como SVD, por sus siglas en inglés, ya que este procedimiento permite identificar la cercanía a las singularidades en el espacio de trabajo del robot [36].

3.6. Manejo de la redundancia

Al querer resolver la cinemática inversa de un sistema sin redundancia, es decir, donde los grados de libertad coinciden con las dimensiones de la tarea que quiere realizar, no hay mucho espacio para considerar otros criterios, como cercanía a los límites de las articulaciones o a singularidades, puesto que hay pocas soluciones para cada estado. Sin embargo, cuando hay redundancia las opciones a elegir son infinitas, lo que permite tomar en cuenta otros aspectos cinemáticos (o tareas) además de la pose final.
El lado negativo es que el problema a resolver se vuelve más complejo ya que es más difícil encontrar soluciones analíticas debido a la complejidad del sistema mecánico. A lo largo de los años se han desarrollado múltiples estrategias para resolver la cinemática inversa en estas condiciones.
Debido a que el objetivo final de este trabajo implica controlar el cuerpo completo del robot: un brazo, el torso móvil y la base omnidireccional, con un total de 11 grados de libertad; se debe poder trabajar con cinco grados redundantes. Ahora bien, el simulador actual ya cuenta con dos distintas estrategias para solucionar y aprovechar la redundancia, puesto que el brazo tiene un grado extra. Ambas están integradas dentro del VFCLIK [4]. Se describen a continuación:

3.6.1. *Weighted Damped Least Squares*

Se puede traducir como "mínimos cuadrados amortiguados y ponderados". Este propone una forma de hallar un cambio óptimo en la posición de las articulaciones para acercarse a la posición deseada utilizando la matriz jacobiana [37]. Este método soluciona muchos problemas que el uso aislado de $J^\dagger$ tiene cerca de las singularidades
Este factor de amortiguamiento variable se puede reflejar tanto en las coordenadas cartesianas como en las coordenadas de las articulaciones, de forma que se pueda fomentar o desincentivar ciertos movimientos del efector final o ciertas articulaciones, respectivamente. Este método se utiliza, por ejemplo, para evitar que las articulaciones se acerquen a sus límites mecánicos [38], con el fin de mantenerlos en un rango seguro y no anular grados de libertad.
Como expliac [39], esto se logra al replantear la ecuación 3.5 de la siguiente forma, donde W_q y W_x son matrices diagonales con coeficientes positivos:

$$W_v v = W_v J W_q W_q^{-1} \dot{q} \tag{3.10}$$

Que se puede plantear como:

$$v_w = J_w \dot{q}_w \tag{3.11}$$

Donde se usaron las siguientes definiciones:

$v_w = W_v v$
$J_w = W_v J W_q$
$\dot{x}_w = W_q^{-1} \dot{q}$

Si resolvemos la cinemática inversa usando el método de *Weighted Damped Least Squares*, explicado en la sección 3.5.2, obtenemos un nuevo problema de minimización [39], donde buscamos no solo el menor error global, sino que se hace dándole un peso distinto a cada variable cartesiana y articulación:

$$\min_{\dot{q}}(\|W_v(v - J\dot{q})\|^2 + \lambda^2 \left\|W_q^{-1}\dot{q}\right\|^2) \tag{3.12}$$

En la sección 3.8 se explica con mayor detalle la forma en que este método se implementa en el solucionador que se utiliza en el simulador.

3.6.2. Proyecciones en el espacio nulo

Esta estrategia, propuesta por Liegois [40], permite no solo resolver la ecuación 3.5, sino que también adapta el sistema para que, de entre la infinidad de posibles soluciones, se llegue a una que permita cumplir otros criterios relacionados con el estado actual del robot. Para ello se demuestra que la siguiente relación es una solución válida:

$$\Delta q = G_1 \Delta x + (G_2 J - I)\varphi \tag{3.13}$$

Donde:

G_1, G_2 = Inversas generalizadas de J
φ = Un vector arbitrario del espacio de las articulaciones

La ventaja de esta ecuación es que el vector $(I - G_2 J)\varphi$ es una proyección de φ en el espacio nulo de J. Esto significa que $0 = J(G_2 J - I)\varphi$ [40], y por tanto se habrá encontrado

un nuevo valor de Δq que tiene el mismo Δx. Por tanto, mediante una selección adecuada de las matrices inversas y del vector φ podemos hacer que el sistema tienda a ciertas posiciones de las articulaciones que son beneficiosas, sin realmente afectar el desempeño del efector final. Se han propuesto distintas funciones para definir φ y así lograr ciertas tareas secundarias, por ejemplo, alejarse de los límites mecánicos de cada articulación [40] o evadir obstáculos.
En el caso en que se esté trabajando con velocidades, como sucede con el ARCOS-Bot, la ecuación 3.13 se puede expresar de la forma:

$$\dot{q} = G_1 v_e + (G_2 J - I)\dot{\varphi} \tag{3.14}$$

En este caso, una forma de trabajar con φ es definiendo una función $H(q)$ que se desee optimizar. Se puede entonces definir un vector de velocidades $\dot{\varphi} = \pm\beta\frac{\partial H}{\partial q}$, donde $\beta > 0$. Este minimiza o maximiza la cantidad $\|\dot{\mathbf{q}} - \dot{\varphi}\|$ según el símbolo que se use. De esta forma el segundo término de la ecuación 3.14 hace que los motores se muevan de forma que la función H decrezca [37].

3.7. Implementación

Para poder llevar a cabo las tareas anteriormente descritas, el simulador se construyó como un conjunto de subsistemas que trabajan simultáneamente. Todos ellos son paquetes de código que debe ser compilados en forma individual y ejecutados siguiendo un orden establecido. A continuación se detallan sus funciones:

- ARCOSPYU: *ARCOS-Lab Python Utilities* (Utilidades de Python del ARCOS-Lab), distintas herramientas creadas en Python que son necesarias para realizar tareas misceláneas dentro del simulador.
- PYROVITO: *Python YARP Robot Visualization Tool* (Herramienta en Python y YARP para la Visualización del Robot), es el programa encargado de crear la imagen del robot y actualizarla según las instrucciones del usuario y la cinemática inversa correspondiente.
- ROBOVIEW: Forma parte de la instalación, pero por defecto su uso no está implementado.
- AVISPY: *A Robot Visualization Tool for Python* (Una herramienta de Visualización de Robots en Python), es una dependencia de PYROVITO que genera figuras geométricas con las que se pueden posteriormente generar las imágenes que representan el robot.
- CMOC: *Compact Mathematical Object Models* (Modelos Matemáticos Compactos de Objetos), contiene los modelos del comportamiento de los objetos que se quieren manipular, además de los que se utilizan para simular el comportamiento de las articulaciones, la mano y sus torques respectivos cuando el brazo se somete a una carga. Esto se hace por medio de `joint_sim` y `torque_sim`.
- KBD-CART-CMD: *Keyboard Cartesian Commander* (Comando Cartesiano por Teclado), captura indicaciones por teclado y las comunica al sistema con el fin de que el usuario pueda manipular la *pose* del efector final del robot simulado en tiempo real.
- OROCOS-KDL: tal como se indicó en la sección 2.3.4, es la biblioteca encargada de llevar a cabo las operaciones cinemáticas del sistema. Una descripción de sus funciones se da en la sección 3.8

- *Robot_descriptions*: (Descripciones de los Robots), contiene el código en Python que construye el árbol o la cadena que representa al robot. Está descrito utilizando objetos de OROCOS-KDL.

- VFL: *Vector Fields Library* (Librería de Campos Vectoriales), es una dependencia de VFCLIK, que contiene los algoritmos con los que se crean campos vectoriales para distintas geometrías, como puntos, círculos o esferas.

- VFCLIK: *Vector Field Based Closed Loop Inverse Kinematics Controller* (Controlador de Cinemática Inversa de Ciclo Cerrada Basado en Campos Vectoriales): implementa la solución de la cinemática inversa de velocidades que proporciona el OROCOS-KDL para definir los movimientos que debe realizar cada motor para alcanzar cierto punto y orientación con la mano del robot. Su principio de funcionamiento se describe con más detalle en la sección 3.2.
  Este a su vez se compone de otros siete programas que se ejecutan como subprocesos distintos, anidados dentro de VFCLIK y heredando sus argumentos:

  - `Bridge` (puente): se encarga de determinar cuál será el o los métodos de control para el robot.
  - `Debug_jointlimits` (depurar límites de articulaciones): realiza el cálculo para encontrar qué tan cerca están las articulaciones de sus límites correspondientes
  - `Joint_p_controller` (controlador de posición de articulaciones): realiza una de las posibles formas de controlar el robot, que corresponde a indicar una posición específica para cada grado de libertad.
  - `Monitor_distance` (monitor de distancia): determina y reporta la distancia lineal y angular entre el efector final y el objetivo.
  - `Nullspace` (espacio nulo): permite incluir velocidades dentro del espacio nulo de la cinemática inversa.
  - `Object_feeder`(alimentador de objetos): administra los objetos que están actuando en la simulación.
  - `VF`: (*Vector Field Executor* o Ejecutor del Campo Vectorial): toma como parámetros los objetos que serán obstáculos u objetivos y los coloca en su respectiva posición y orientación para crear el campo vectorial.

Tras ser compilados e instalados, los distintos programas pueden ejecutarse desde la terminal cuando son requeridos. El orden y los parámetros que se les indique es de suma importancia para que se de el correcto funcionamiento. A continuación se describe los comandos básicos requeridos, los cuales deben correrse desde la ubicación donde se realizó la instalación, que en caso de seguir las indicaciones de [33], debe ser en ∼`/local/src/`:

1. `$ yarpserver --write`
   Ejecutar el servidor de YARP, el cual es requerido por los demás módulos del simulador para comunicarse entre sí.

2. ```
   $ vfclik -i lwr -i right -d robot_descriptions/arcosbot/
       kinematics/lwr/ -s
   ```
   ```

Corre un módulo de VFCLIK para una cadena cinemática. Los parámentros son: `-i` el tipo de brazo y la instancia (si es derecho o izquierdo), si es que lo requiere; y `-d` la localización de la descripción de la cadena cinemática.

3. `$ pyrovito -r lwr --arm_right -a robot_descriptions/arcosbot /kinematics/lwr/`

   Corre el módulo PYROVITO con el fin de visualizar la cadena recién creada. Los parámentros aquí utilizados son: `-r` el nombre del robot; `--arm_right` indica si el modelo tiene un brazo derecho; `-a` ubicación de la descripción del brazo; y `-d` ubicación de la descripción de la mano.

4. `$ run_right.sh /0`

   Este es el comando para ejecutar KBD-CART-CMD, por lo que permite utilizar ciertas teclas designadas para rotar y mover cartesianamente la mano.

3.8. Uso de la biblioteca OROCOS-KDL

3.8.1. Cadenas cinemáticas

En OROCOS-KDL una cadena cinemática se define por medio de segmentos, los cuales se componen a su vez de una articulación y su eslabón subsecuente, similar a cómo se realiza con la convención de Denavit-Hartemberg (ver sección 2.2.2. Tal como se muestra en la figura 3.3, esto se logra por medio de dos objetos: un marco de referencia (`Frame`) y una articulación (`Joint`), que puede ser traslacional (`TransX`, `TransY` o `TransZ`) o rotacional (`RotX`, `RotY` o `RotZ`) y que se define con respecto al marco de referencia del segmento anterior. Estos sistemas referenciales se construyen, a su vez, de un vector y una rotación, que juntos definen la transformación homogénea del eslabón. Para construir la cadena se sigue el siguiente formato:

```
cadenaCinematica = [
    Segment(Joint(Joint.TransX),
        Frame(Rotation.Identity(), Vector(0.0, 0.0, 0.0))),
]
```

3.8.2. Cinemática inversa de velocidades

El servicio de VIK se lleva a cabo utilizando la clase `KDL.ChainIkSolverVel\_wdls` del paquete Orocos KDL, la cual soporta grados redundantes. Este algoritmo se basa en el uso de una inversa generalizada ponderada y del método de cuadrados mínimos amortiguados, para lo cual hace uso de una decomposición de valores singulares [38]:

$$J^{\#} = M_q V_b \Psi(D_b) U_b^T M_x \tag{3.15}$$

M_q = Matriz de pesos de las articulaciones
M_x = Matriz de pesos cartesianos
B = $M_x J M_q$
$\Psi(A)$ = pseudo inversa por cuadrados mínimos amortiguados de A
$U_b D_b V_b^T$ = SVD de B

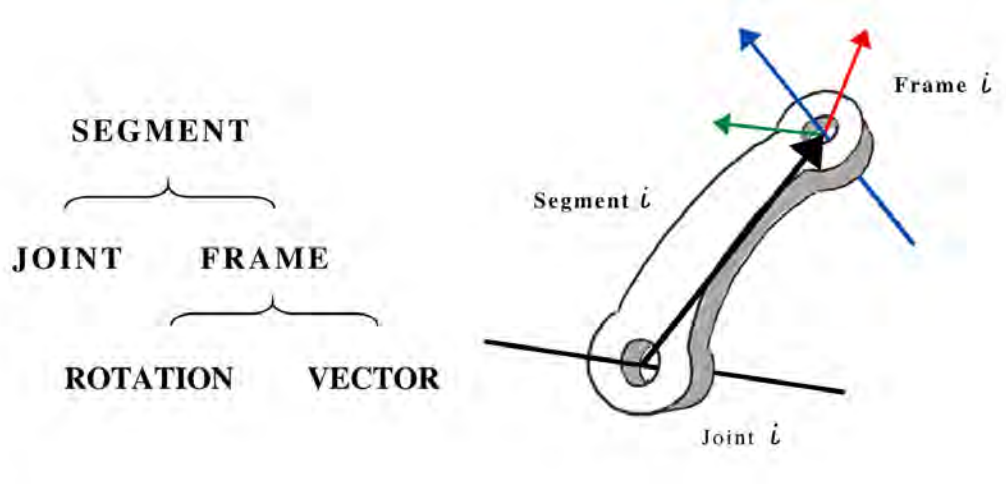


Figura 3.3: Composición de un elemento tipo *segment* de OROCOS-KDL

Para construir este objeto es necesario indicar tres parámetros: una cadena cinemática, un límite inferior para descartar un valor singular y un máximos de iteraciones para los cálculos de los valores singulares.
La función de la que se quiere hacer uso es `CartToJnt`, que devuelve un $\dot{q}$ para una v_e deseada. Para ello requiere también conocer el estado actual de la cadena, por lo que solicita como parámetro un q.
Otros métodos de los que se hace uso son:

1. `setWeightJS`: permite establecer una matriz de pesos M_q para las articulaciones. Debe ser una matriz simétrica y de valores positivos, en el caso más sencillo, una matriz diagonal. Esto va a minimizar la norma diagonalmente ponderada $\sqrt{\dot{q}^* M_q^{-2} \dot{q}}$, por tanto un valor de cero implica que ese grado de libertad no contribuirá a la sistema.

2. `setWeightTS`: permite establecer una matriz de pesos M_x para los grados de libertad del espacio de la tarea a realizar. También requiere que sea una matriz simétrica de valores positivos. Tomando una matriz diagonal como ejemplo, entre más pequeño sea el valor correspondiente a cierta coordinada, el error de esta podrá ser mayor. Es decir, no implica que no se utilice dicha movilidad, sino que no se tomará en cuenta.

3. `setLambda`: permite definir el valor de λ, el cual define cuanto se desean evitar el efecto que tienen las singularidades a la hora de calcular la cinemática inversa de velocidades.

3.9. Cambios necesarios

El objetivo de este trabajo es extender el sistema aquí expuesto al incluir nuevos grados de libertad. Basado en este estudio se proponen los cambios necesarios para lograrlo:

- Una cadena cinemática actualizada que incluya todos los grados de libertad, congruentes con la geometría del robot.

- Asegurarse que el VIK pueda solucionar adecuadamente las cadenas ampliadas, a pesar del alto nivel de redundancia

- Canales de comunicación hacia el torso y la base omnidireccional que indique la velocidad que debe tomar cada articulación.
- Canales de comunicación desde los motores de los nuevos grados de libertad hacia el sistema de control para indicar su posición actual. Esta información es requerida por dos sistemas: (i) el VIK, para determinar la matriz jacobiana de la cadena en cada instante, y (ii) la Cinemática Directa de Posiciones, que se encarga de calcular la posición del efector final a partir del estado de todas las articulaciones.
- Un módulo de interpretación que transforme las señales de velocidad instantánea deseada en x y y, y la velocidad angular en z en potencia de los motores de las ruedas. Este controlador está en desarrollo como un proyecto independiente en el laboratorio.

Los elementos mencionados se comunicarán con los controladores del hardware respectivo cuando se utilice con el robot físico. Sin embargo, en este caso se desea trabajar con el simulador, por lo que basta con utilizar los modelos físicos que se tiene de las articulaciones de revolución y prismáticas.

Capítulo 4

Diseño del sistema de control cinemático de cuerpo completo

En este capítulo se describe el proceso de diseño de un sistema de control cinemático de cuerpo completo basado en VFCLIK. El propósito es darle al robot la capacidad de utilizar todos sus grados de libertad en forma coherente, en especial entre aquellos que implican desplazarse o rotar con su base, y los que modifican la postura del robot, es decir su torso móvil y brazo.
Para ello se divide este capítulo en dos secciones. Una en la que se plantean distintas opciones en las que se pueden coordinar los movimientos del cuerpo del robot, lo cual incluye: el estudio de las necesidades establecidas en los objetivos, las propuestas para su solución y su respectiva comparación con los métodos expuestos en la literatura; para finalmente escoger una. La segunda sección describe el diseño del sistema aplicado directamente al caso de estudio, por lo que se desarrollan modelos para la mesa, el robot y el objeto que pueda usarse junto al método de coordinación del cuerpo completo para lograr el comportamiento deseado.

4.1. Diseño del método de coordinación del cuerpo completo

4.1.1. Organización del cuerpo completo

La base está diseñada para lograr movimientos grandes, sin necesidad de hacerlo con delicadeza. El resto del cuerpo, al igual que con una persona, tiene una motora cada vez más fina conforme se acerca a la mano. Desde una perspectiva humana esto se puede abstraer como dos tareas distintas, una de manipulación y otra de locomoción o desplazamiento, donde la primera depende de la segunda.
Esta configuración tiene mucho sentido cuando el espacio de acción es muy amplio, pero se requiere siempre tener capacidad de realizar tareas que requieren movimientos finos. Desde una perspectiva evolutiva podría decirse que está relacionado con el hecho de que las personas, como seres terrestres, suelen mantener su centro de masa cercano al suelo, movilizándolo mayoritariamente en un plano paralelo (locomoción). Pero su campo de manipulación es todo el espacio de cuerpos rígidos circundantes, que requiere de seis grados de libertad (manipulación). Como se quiere que el robot realice lo mismo que un humano, resulta lógico que se quiera imitar esta capacidad.
Dada esta diferencia sustancial entre los movimientos de la base y el resto del cuerpo, es lógico

y práctico diferenciar en alguna medida su control, y así definir una forma en la que ambos se coordinen. Esto va ser sumamente importante a la hora de manipular objetos a lo largo de trayectorias largas, o cuando se desee alcanzar puntos distantes con el efector final. Como ya se ha explicado, este trabajo se enfoca en el segundo escenario.

Habiendo tomado la decisión de realizar esa diferencia de control, es necesario definir cómo se va a modelar el robot como un cuerpo completo, ya que existe la opción de diferenciar ambos tipos de movimiento completamente, o de ignorar dicha distinción lo máximo posible. Las ventajas correspondientes se listan a continuación:

Desacoplar el control

- Control directo sobre cada tipo de movimiento. Lo cual puede implicar mayor simplicidad.
- La coordinación es muy específica de la tarea que se quiere desempeñar y del robot que se utilice.

Unificar el control

- La carga recae sobre el método de solución de la redundancia.
- Ofrece un control de cuerpo más generalizado, tanto para la tarea a realizar como para el robot que se utilice.

Si se modela el cuerpo completo como una única cadena cinemática de cuerpos rígidos, se puede utilizar el mismo sistema de cinemática inversa para solucionar la posición y orientación del efector final, pero ahora usando todo el cuerpo. La coordinación entre los tipos de grados de libertad puede recaer en otro módulo. Como se describe en la subsección 4.1.4, una opción bastante atractiva al desacoplar el control, es manejar el robot como la coordinación de dos cadenas cinemáticas distintas, pero en serie.
En ambos casos, el desplazamiento y la rotación proporcionada por la base se podrían modelar como dos articulaciones prismáticas perpendiculares entre sí, más una rotacional (ver figura 4.1). A diferencia de los grados de libertad del brazo y del torso, estos no tendrían límites mecánicos, e incluso se podría decir que virtualmente tampoco. Este hecho no puede obviarse si se quiere que todo el robot se mueva como un conjunto.

Movimiento del torso y el brazo

Es importante discutir la la forma en que se desea abstraer la traslación vertical del torso, ya sea como una parte de la base, del brazo, o bien, como un elemento independiente. Se pueden dar varios argumentos en favor de una o de otra:

- Unirlo con la base afecta la diferenciación entre los movimientos que modifican la *pose* global del robot, de aquellos que cambian únicamente la *pose* del efector final.
- La traslación del torso sí está supeditada a límites físicos, mientras que la base no.
- Trabajarlo en conjunto con la base haría que los movimientos prismáticos queden juntos, formando un sistema cartesiano. Desde la perspectiva del solucionador de cinemática

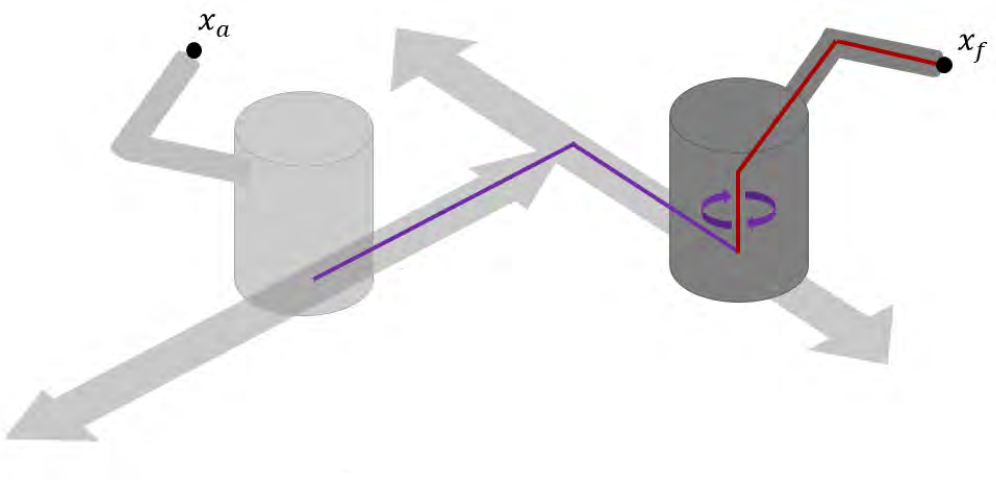


Figura 4.1: Modelo del robot completo como una sola cadena cinemática. En morado se ven las articulaciones relacionadas a la base. En rojo se visualizan los eslabones del resto del cuerpo.

Elaboración propia

directa, estos grados de libertad, junto a la rotación en z son suficientes para lograr la mayoría de las poses, por lo que ante ciertos estímulos no tendría necesidad de usar su brazo (esto se explica con más detalle en la sección 5.7).

- Dejar libre el movimiento del torso fomenta una mejor postura para el brazo y no significa un riesgo mayor para la posibilidad de colisión del cuerpo del robot con la mesa, por lo que tiene sentido considerarla como parte del manipulador.

Existe la posibilidad de que también en el brazo se fomenten ciertos grados de libertad sobre otros. No obstante, en lo que respecta a los objetivos de este trabajo, no hay criterios de suficiente peso que sea provechoso considerar. El trabajo se enfocará entonces en fomentar o desincentivar la base.

Movimiento de la base

El movimiento de la base, a su vez, se puede separar en dos naturalezas distintas, una de desplazamiento y otra angular, ambas sobre el mismo plano. Bidimensional y unidimensional, respectivamente. Se pueden trabajar en conjunto o individualmente, pero es necesario definir antes el principio detrás del control para tomar una decisión adecuada.

4.1.2. Criterios que rigen la coordinación

Con el fin de lograr un movimiento eficiente y acorde a la función de manipulación, es necesario definir criterios que determinen cuando se va a fomentar una o la otra. Las figuras 4.2 muestran las distintas consideraciones que influyen con más fuerza a la hora de decidir si es necesario o no girar o desplazarse. Estas se utilizan para elaborar las propuestas de métodos de coordinación del control de cuerpo completo. A continuación se describen con mayor detalle:

Visión y orientación

Un robot puede tener pleno conocimiento de su geometría y por medio de un modelo cinemático puede alcanzar una cierta ubicación con bastante exactitud con tan solo saber su posición relativa a él. Para una persona esto no es cierto, y por tanto resulta necesario tener una retroalimentación estereoscópica constante para saber la ubicación de los objetos, e incluso la de su propia mano.
Esto hace posible que, por ejemplo, un robot alcance objetos fuera de su rango de visión, superando así la versatilidad del humano. No obstante esto no toma en consideración los cambios inesperados, que podría inducir al fallo:

- Errores: la precisión mecánica del brazo y del torso es bastante alta, pero no se cumple lo mismo para la odometría de la base. Si se quiere conocer el cambio logrado por las ruedas es necesario que se dé un rodamiento sin deslizamiento perfecto y esto puede ser difícil de lograr. La acumulación de pequeñas desviaciones pueden terminar en un error grande al llegar a la meta, por lo que es deseable poder tener retroalimentación visual en todo momento

- Cambios en el entorno: obstáculos inesperados o móviles, la presencia de humanos, u otros errores inducidos por el medio.

La mejor forma de evitar ambos riesgos es tener la mayor información posible de las direcciones con más probabilidad de que se presenten peligros. Sin duda esta dirección es aquella hacia la cual el robot se está moviendo, ya sea girando o desplazándose. La conclusión es curiosamente trivial: se debe fomentar que el robot vea en la dirección en la que se mueve. Esto se puede traducir en dos posibles opciones, (i) como los sensores de percepción se enfocan hacia el frente, es favorable que la base rote en la dirección de cambio siempre que sea posible; (ii) para ángulos pequeños de desviación, el cuello del robot debe girar en busca de la dirección de movimiento. Sin embargo, el control del cuello no está dentro del alcance de este proyecto, por lo que es necesario seguir la primera opción.

Límites de las articulaciones

A pesar de que esta es una limitación que afecta a los grados de libertad del cuerpo del robot, es justamente la base la que puede ampliar el rango de acción del brazo al lograr *poses* corporales que sean más "cómodas". Queremos alejarnos de estos valores críticos, pues disminuyen la versatilidad del sistema al frenar la posibilidad de usar los grados de libertad. Como se quiere llegar a objetos que no están al alcance inmediato, este criterio tiene más peso para la rotación que para el desplazamiento:

- Desplazamiento: si el robot no avanza, su brazo se estiraría todo lo posible, en el caso del brazo KUKA, esto supondría la posición media de la mayoría de las articulaciones, justamente el punto más lejano del límite. El verdadero problema es que se habrá llegado a una singularidad, esto se verá más adelante.

- Rotación: en imitación al cuerpo humano, el robot está diseñado para tener la más alta capacidad de manipular justo al frente suyo, donde hay mayor información visual. Esto implica que los límites de las articulaciones se alcanzarán en la medida en que nos alejemos de esas posiciones. En otras palabras, hay una estrecha relación entre ambas, si se promueve la rotación se termina evitando indirectamente a que se llegue a los

límites. O bien, si se desestimulan las articulaciones cercanas a los límites se termina favoreciendo la rotación del robot, puesto que no tiene límites. Esta última opción es deseable, puesto que implica menos etapas de decisión, sin embargo, como se describe en la sección 5.7, no resulta efectiva en este sistema de control.

Extensión del brazo

Las singularidades se alcanzan en ciertas configuraciones del espacio de las articulaciones, donde los ejes de varios grados de libertad se cruzan o comparten planos. A pesar de que existen muchos escenarios en los que esto puede suceder, uno de gran importancia en este estudio se da cuando el brazo está completamente extendido. Existen criterios matemáticos que cuantifican la manipulabilidad como una medida inversa a la cercanía con las singularidades [5], y por tanto se relacionan con qué tanta libertad tiene el robot para moverse.
El sistema de control del robot ya cuenta con un mecanismo integrado dentro de la cinemática inversa de velocidades, para evitar las singularidades internas. No obstante, es necesario evitar llevar al robot a estados que requieran la total extensión del brazo.

Energía

Ciertas articulaciones consumen mayor energía que otras debido a que requieren ejercer torques mayores. Se puede decir que está relacionado con el orden que tienen en la cadena, puesto que los primeros mueven todo el robot, mientras que el último mueve solamente la mano. Esta consideración entra en juego cuando, por ejemplo, una persona busca realizar las cosas de la manera más fácil. A pesar de que este criterio se puede considerar con cierta facilidad, resulta conveniente ignorarlo, puesto que más bien castiga el uso de la base, contrario a lo que dictan los demás aspectos.

Alcance y distancia

La distancia entre el centro geométrico del robot y el punto que se quiere alcanzar es tal vez el parámetro del que primero se piensa cuando se pretende considerar el avance de la base. Se puede decir que es análogo a la orientación relativa del robot. Al igual que en ese caso, se puede considerar que está muy relacionado con la manipulabilidad del robot, puesto que va a determinar cuánto se debe estirar el brazo. Como ya se explicó, es posible que en la medida en que se busque evitar las singularidades se termina dando campo a que el robot avance. No obstante, este alcance también se podría considerar en forma directa, es decir, que el robot avance según la distancia que se registra entre él y la meta. Esta última opción es atractiva, sobre todo porque es una magnitud fácil de considerar.
Es importante también hacer la distinción entre alcanzar una *pose* en el espacio con el efector final, y llevar al robot a que manipule un objeto que se encuentra en cierta *pose*. La diferencia puede parecer pequeña, pero tiene una diferencia semántica importante. En el primer caso se desea nada más tomar el objeto, tocar una superficie, o alcanzar cierta postura, la importancia del punto es momentánea y después de ella probablemente el robot regresará a su estado anterior. El segundo caso implica que queremos llegar a cierta ubicación y orientación para realizar una tarea con un objeto en el lugar donde se ubica. En tal caso resulta más lógico que el robot se movilice y manipule de cerca. Se podría decir, incluso, que se tratan de tareas atómicas distintas y por tanto pueden ser controladas por distintos algoritmos. Posiblemente para ese segundo caso sea más práctico que el robot navegue hasta las cercanías y después

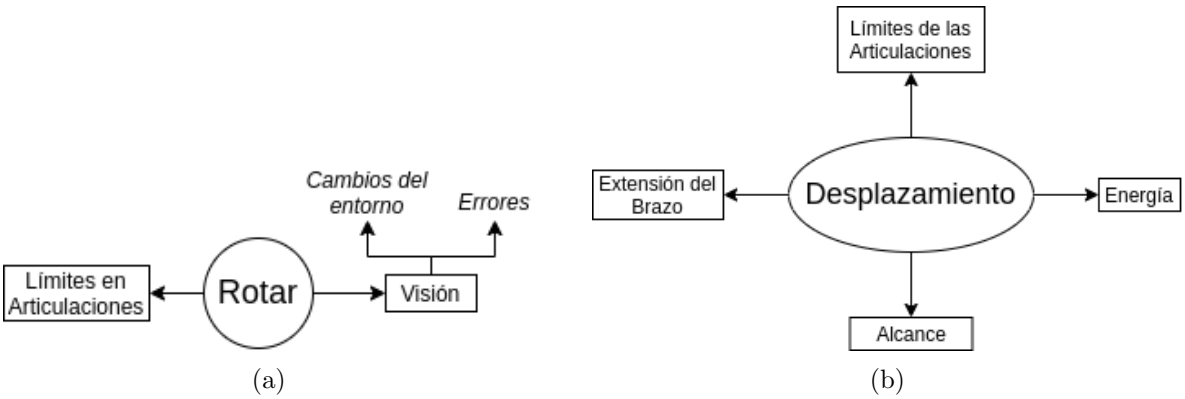


Figura 4.2: Criterios a considerar cuando se quiere (a) rotar el robot y (b) desplazarlo

Fuente: Elaboración propia

realice una acción de alcance. Para el caso particular de este trabajo, este aspecto queda sujeto a la cercanía del objeto al borde de la mesa, puesto que esto va a definir qué tanto tiene que estirarse el brazo.

4.1.3. Sistemas de coordinación de cuerpo completo existentes

La coordinación del cuerpo completo de robots humanoides ha sido un área estudiada a lo largo de más de dos décadas, con propuestas muy diversas para resolver el problema. Usualmente ha venido acompañado con la solución de problemas paralelos, de forma que se aproveche la alta redundancia del sistema [41]. La línea clásica se ha basado en dos corrientes: el uso de la matriz jacobiana, o bien en estrategias de planeamiento, que analizan el escenario y procuran seleccionar la ruta más óptima, según diversos criterios. La primera es la corriente de estudio que se ha seguido con el robot del ARCOS-Lab y que obedece a los objetivos de este proyecto. A continuación se describen algunas propuestas para coordinar el cuerpo completo que se asemejan a la organización que aquí se propone para el robot:
Iskandar [28] define límites relativos y fijos para el efector final con respecto a la base, de forma que cuando quiera cruzar ese umbral semiesférico se crean fuerzas virtuales sobre el control de la base, halándola o empujándola para que esta se desplace. La rotación se habilita con fronteras rectilíneas a cada lado. Este control resulta muy práctico a la hora de realizar tareas que requieren un gran rango de acción, pero a la hora de alcanzar objetos puede llevar a movimientos poco naturales.
Lee [41] propone el uso de las matrices de transformación de ciertos eslabones del cuerpo del robot como entrada para tareas a cumplir durante la solución de la cinemática inversa. De esta forma ciertos movimientos del cuerpo del robot se tratarán como tareas independientes a la trayectoria del efector final.
En lo que respecta al caso específico de alcanzar un objeto sobre una mesa utilizando un control de cuerpo completo, existe poco en la literatura. Una de las investigaciones más similares es la desarrollada por [42], donde se le da al robot tanto la capacidad de tomar objetos, como de desplazarse hacia la mesa con el cuidado de no tocarla. Sin embargo, ambas son tareas independientes, que dependen de la ejecución individual de un método de evasión de obstáculos. Coordinados ambos por un sistema de control de alto nivel. En [43] se estudia

el caso a mayor profundidad, aunque se lleva a cabo algo similar. Se establecen tres modos de navegación en función de la distancia del robot al objetivo y lo llevan lo más cerca posible, seguido de esto manipula. En otras palabras, no se han implementado estrategias que usen el cuerpo completo de forma unificada y por tanto, la manipulación y la movilidad dependen de un esquema de control de más alto nivel.

4.1.4. Propuestas de sistemas de coordinación

Basado en la organización del cuerpo y los criterios de movimiento expuestos atrás, se busca diseñar un sistema que coordine entre el desplazamiento y la postura del robot. Como se expuso en el capítulo 3, el sistema de control actual ya posee herramientas que le permiten resolver y sacar provecho de la redundancia. Estas se pueden aplicar ahora a la nueva cadena ampliada, justo para lograr este propósito.
Se van a detallar diferentes propuestas. Algunos principios que se busca respetar en su planteamiento son: (i) que los criterios aplicados no requieran de una decisión de alto nivel, sino que se expresen implícitamente mediante la solución de la cinemática inversa de velocidades, (ii) basarse en los módulos ya implementados en el sistema de control actual y (iii) que sea un control de cuerpo completo, y por tanto pueda usar todos los grados de libertad en la medida en que lo requiera.

Espacio nulo para minimizar criterios

Este método ya utiliza para alejarse de las singularidades en el robot del ARCOS-Lab, y tiene la ventaja de que se puede implementar al definir una función $H(q)$ que se quiera minimizar o maximizar, lo cual no es difícil de hacer con los criterios mencionados, como por ejemplo la distancia al punto o la orientación relativa del robot. Basta con expresar un sistema de cinemática directa por medio matrices de transformación para las tres primeras articulaciones (las que corresponden a la base). Este indica cual es la posición y orientación del robot torso, lo cual se puede transformar fácilmente al centro del robot:

$$T_r^0 = T_1^0(q_1)T_2^1(q_2)T_3^2(q_3)T_r^3 \tag{4.1}$$

T_r^0 = matriz de transformación del robot con respecto al marco de referencia global
T_i^j = matrices de transformación de cada una de las articulaciones
q_i = el vector de articulaciones respectivo

Como ya se conoce la *pose* del objetivo, encontrar el vector que los une es una operación muy simple. A partir de este se puede obtener la distancia y el ángulo de desviación con respecto al objetivo (figura 4.3)

Los demás criterios que se quieran aplicar se puede incluir como parte de una jerarquía de proyecciones, tal como propone [8].

Ponderación con la matriz de pesos

Otra alternativa es usar las matrices de pesos ya implementadas en el método de Mínimos cuadrados amortiguados y ponderados. Su uso es aún más directo, pues no hay necesidad de crear ninguna función, por lo que es más transparente a la hora de modificar. Como esta

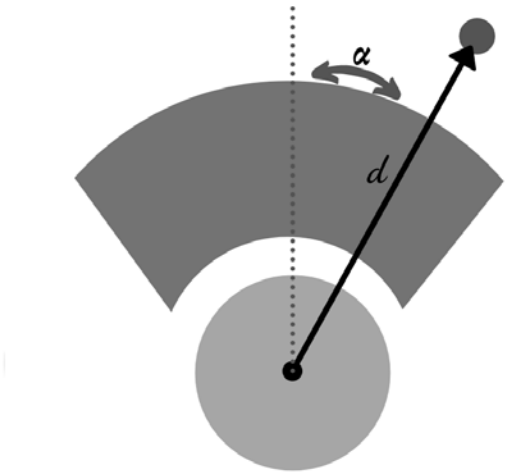


Figura 4.3: El vector que une al robot con el objetivo se puede usar para calcular tanto la distancia, como el la proyección sobre el suelo del ángulo de desviación

Elaboración propia

matriz ya es parte del sistema actual, solo sería necesario crear un algoritmo que la modifique durante cada ciclo de solución, siguiendo los criterios que se deseen.
Como los movimientos que se desean coordinar son parte de la configuración del cuerpo y no del *task space*, solo será necesario utilizar la matriz de pesos de las articulaciones. Dos formas con las que se puede hacer esto son:

- Usando los valores directos de distancia y de ángulo de desviación para modificar el índice respectivo en la matriz. Este puede ser un control proporcional o proporcional diferencial.
- Basándose en una versión ampliada del mapa de capacidades de uno de los brazos desarrollado por [4]. Este indicaría los puntos a los cuales el robot ya ha sido capaz de llegar en simulaciones anteriores. Se puede, entonces, aumentar los pesos de la base hasta que el objetivo esté dentro del mapa de alcance del robot. Esto ayudaría a no depender únicamente de un valor numérico, sino de datos ya antes probados.

Subdivisión de la cadena cinemática

Usar los valores de d y α para orientar el robot, es casi equivalente al efecto del Campo Vectorial utilizado para determinar el *twist* del efector final. Se puede usar entonces este mismo método para determinar la velocidad deseada para el robot completo. La cadena se podría entonces subdividir, y resolver dos sistemas simultáneamente, donde la cadena del torso y el brazo dependen del resultado de la cadena de la base móvil.
La estrategia de separar sistemas de gran redundancia en distintos subsistemas y controlarlos de manera separada se ha trabajado ya en el pasado. Una diferencia sustancial entre estas propuestas es qué tan desacoplados están ambos, por ejemplo [44] divide un cuerpo humanoide en cinco grupos de eslabones, formando un árbol de cinco elementos. El control de cuerpo

completo se logra mediante un algoritmo de planeación que crea una primera ruta basado en el eslabón raíz y después lo va modificando según se agregan los demás elementos. El proceso se repite para cada eslabón creando una ruta refinada para cada uno.

4.1.5. Selección del sistema de coordinación

Para escoger un sistema de coordinación es importante considerar la simplicidad de su implementación en contraste con las ventajas que ofrece. Este estudio se resume en el cuadro 4.1. Además de esto, se desea seguir los criterios especificados al inicio de esta sección, de forma que se logre un control de cuerpo completo modificando el sistema actual lo menos posible, y buscando depender lo mínimo del sistema de alto nivel.

Cuadro 4.1: Aspectos evaluados para seleccionar una estrategia de coordinación

	Proyecciones en el Espacio Nulo	Modificación de la matriz de Pesos	Matriz de Pesos y Mapa de Capacidades	Subdivisión de la Cadena
Simplicidad	Baja. Requiere de la modificación de un módulo más	Media. Solo se debe modificar los pesos durante la ejecución de VFCLIK, lo cual es un cambio muy modular	Baja. Además de modificar los pesos, se debe leer un archivo grande y definir un algoritmo para interpretarlo	Muy baja. Requiere correr dos procesos VFCLIK simultáneos, donde el segundo depende del primero
Mayor ventaja	Se conoce la prioridad relativa que tiene con respecto a otras tareas	Es fácil de implementar y ajustar y está bastante desligada de la estructura del robot	Se basa en datos previos del robot, lo cual lo hace más robusto	Se depende menos en el manejo de la redundancia para controlar los movimientos.

De entre la opciones presentadas, la más simple es la ponderación de los grados de libertad mediante la matriz de pesos; comparada con las demás opciones es la que más depende de la forma en que el método de cinemática inversa de OROCOS-KDL resuelve la redundancia. Esta es una de las razones de su simplicidad, pero también un factor que podría reflejarse en su desempeño. No obstante, sigue siendo compatible con el uso del espacio nulo, lo cuál le da opción de ser mejorada en el futuro.
Una ventaja importante sobre la propuesta de subdividir la cadena y la de usar proyecciones del espacio nulo es que en principio no hay necesidad de modificar los módulos correspondientes, los cuales ya han sido ampliamente probados y utilizados numerosas veces, sino que solo es necesario hacer uso de funciones que ya estos ya ofrecen. En esta línea, el caso más extremo sería el de la subdivisión de la cadena, el cual requeriría no solo modificar VFCLIK en varios aspectos, sino también correr dos cadenas de manera simultánea.
En lo que respecta a la capacidad de darle al robot un movimiento de cuerpo completo natural, es difícil dar un juicio previo a la etapa de implementación. Lo que sí se sabe es que todas las opciones tienen la posibilidad de lograrlo, aunque llevarlas al correcto desempeño

no parece ser igual de fácil en todas. Esto hace a la estrategia de la matriz de pesos el mejor candidato para desarrollar el sistema de coordinación.

4.1.6. Selección de la organización del cuerpo completo

Habiendo realizado esta elección, el paso siguiente es definir si se quiere modelar el robot desacoplando los grados de libertad de cada tipo de movimiento, o bien, si unificar todo el robot en una sola cadena. Considerando que la ponderación de los pesos no requiere que se haga una distinción en la forma en que se plantea la cadena, y que esto supone realizar menos cambios al sistema actual, resulta lógico escoger la opción de unificar el control del cuerpo. Sin embargo, esto va a implicar ciertas consecuencias que deben considerarse en el diseño:

- Solo se tiene control sobre qué tanto se usa uno u otro tipo de movimiento, pero no se puede modificar su dirección. Por tanto, la base siempre se moverá en línea recta hacia el objetivo.
- Debido a que toda la cinemática se modela como una misma cadena, el comportamiento de todos los grados de libertad es interdependiente, lo que puede dificultar cumplir subtareas.

4.2. Diseño del sistema de control de cuerpo completo

En la sección anterior se discuten distintos criterios que se deben considerar al llevar a cabo movimientos de cuerpo completo a la hora de alcanzar objetos. Basado en ellos se escoge utilizar los pesos de las articulaciones como método para coordinar entre el desplazamiento y la postura del robot. Ahora se debe desarrollar el sistema de control enfocado directamente en el caso de estudio, es decir, al robot humanoide del ARCOS-Lab y al objetivo de acercarse a un objeto que se encuentra sobre una mesa. Aquí se describe el proceso de diseño de dicho algoritmo.

4.2.1. Esquema general

La matriz de pesos de las articulaciones modifica el uso que el algoritmo de cinemática inversa hace de cada una de las articulaciones de un sistema redundante. Aquí se quiere aprovechar esa posibilidad para restringir la forma en que se haga uso de la base omnidireccional y el torso en relación con el brazo. Como el caso planteado corresponde a alcanzar un objeto sobre una mesa, se desea: (i) que el efector final logre una *pose* específica, (ii) que el robot se detenga de forma que no colisione con la mesa, y (iii) que la mano no toque la superficie en ningún punto de la trayectoria.
Como se está trabajando con un control unificado de cuerpo completo, el objetivo es que todos los grados de libertad del robot se controlen de la forma más similar y cercana posible,

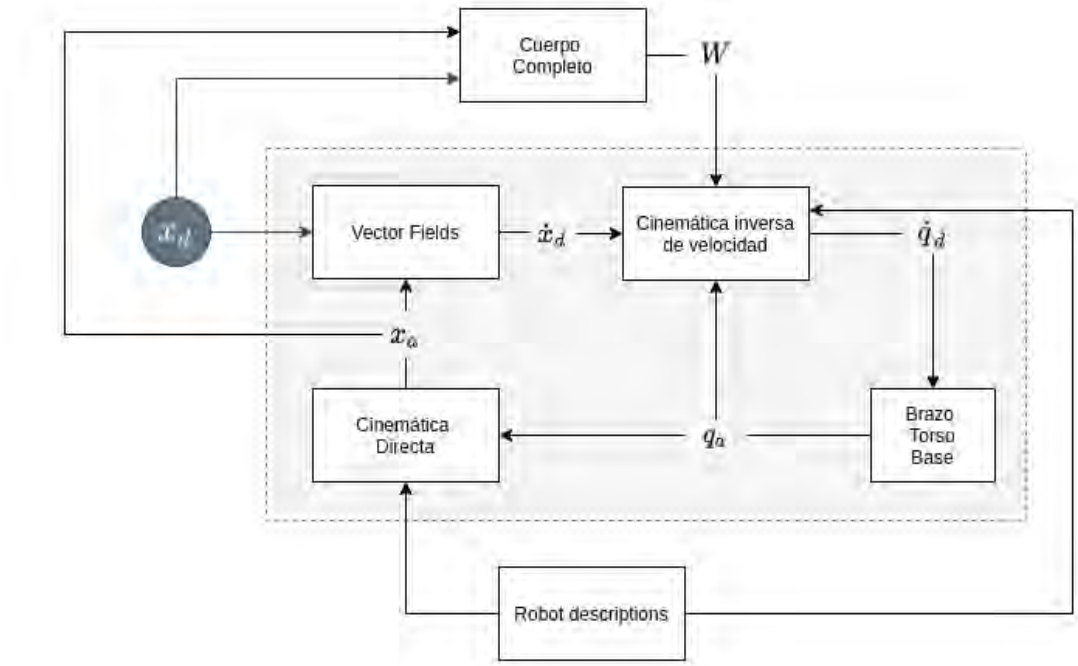


Figura 4.4: Diagrama de bloque de la relación entre el nuevo sistema junto a VFCLIK (en el recuadro gris) y el módulo robot_descriptions

Fuente: Elaboración propia

de manera que se utilicen según sean requeridos por la tarea cinemática que se está desempeñando, cumpliendo así los tres objetivos recién mencionados. Esto se puede lograr haciendo una sola cadena cinemática altamente redundante que incluya todo el robot, y haciendo que sea el algoritmo de cinemática inversa el que se encargue de comandar cada articulación. La coordinación entre los tipos de movimiento se puede lograr mediante un proceso que corra de forma paralela y cambie la matriz de pesos, condicionando así la solución dada por VFCLIK, tal como se muestra en la figura 4.4. Dentro de este nuevo módulo se debe incluir todo lo necesario para que, según el estado actual y deseado del efector final, se determine cuáles son los pesos que coordinan el cuerpo del robot para lograr la tarea estipulada.

4.2.2. Modelos de elementos físicos

El problema que se desea resolver requiere solamente de tres elementos físicos: el robot, la mesa y el objeto. Es necesario definir un modelo con el cuál se van a trabajar dentro del algoritmo de control. El más sencillo es el **objeto**, el cuál, según los alcances del trabajo, se va a ignorar, y se considerará únicamente la *pose* que un sistema de alto nivel indique necesaria para manipularlo. Se puede ver entonces como un marco de referencia estático.
La **mesa**, que en realidad puede extenderse a una superficie plana amplia, debe modelarse como un obstáculo. Entendida de esta forma, tiene muchas posibles formas de interferir; las dos más evidentes son la superficie y el borde, donde la primera representa más riesgo para el

brazo y la segunda para el cuerpo del robot. Se elige la primera como la más crítica, debido a que siempre representa un riesgo, mientras que la última depende mucho de la postura inicial del robot. Ahora bien, las mesas, aunque usualmente cuentan con cuatro aristas, pueden tener diversas formas, sin embargo si se sigue el recorrido más corto, solo la frontal será un problema. Una forma muy general de abarcar el problema es por medio de una recta que coincida con este borde y que funcione como una frontera que limite el movimiento de la base. Esto podría aplicarse a cualquier geometría, siempre y cuando se coloque de forma paralela al lado más cercano (o tangencialmente en superficies curvas), tal como se puede ver en la figura 4.5. Esta estrategia, aunque es una simplificación, no se aleja mucho de lo que en realidad es crítico para el robot, y además es coherente con la información que se obtiene del sistema de percepción, puesto que este tiene mucho más datos sobre la parte frontal de los objetos.

Figura 4.5: El modelo de la mesa es la línea recta sobre la que se encuentra su borde frontal, o bien, tangencial al punto más cercano.

Fuente: Elaboración propia

El **robot**, por otro lado, se puede dividir en sus dos principales partes en movimiento, su cuerpo, y su efector final; en coherencia con lo descrito sobre el modelo de la mesa, estos son los dos elementos que más riesgo tienen de colisionar, por lo que deben considerarse de forma individual. El último eslabón de la cadena representa la mano o herramienta final del robot, por lo que se puede simplemente hacer coincidir con la *pose* deseada para alcanzar el objeto, siguiendo el método utilizado originalmente con el ARCOS-bot. Tal como se puede ver en la figura 4.5, el cuerpo puede modelarse como un cilindro definido por un radio de seguridad, de forma que se evite que este colisione con el borde de la mesa. Según las medidas actuales del robot, el punto más lejano al centro de la base es de 62 cm, se usará entonces un valor de 65 cm.
Adicional a esto se define una distancia de proximidad o de alcance, a partir de la cual se puede considerar un objetivo como cercano. La idea detrás de esta medida es definir un límite a partir del cuál un punto se considera lo suficientemente lejano como para que el robot se mueva con su base antes de hacer cualquier intento de alcanzarlo con su brazo. Esto favorece el comportamiento natural del robot, ya que el humano también hace esta distinción, aunque de forma inconsciente. Este radio también se hace desde el centro de la base y debe cumplir dos condiciones: (i) lo suficientemente grande como para que el brazo haya pueda desarrollar sus movimientos; y (2) no tan pequeño como para que sea más cercano que la *pose* inicial del efector final. Una buena medida que cumple ambos requisitos es una distancia levemente mayor al alcance del brazo, que en este caso es cercana a los 1,3 m. Se definió como 1,5 m.

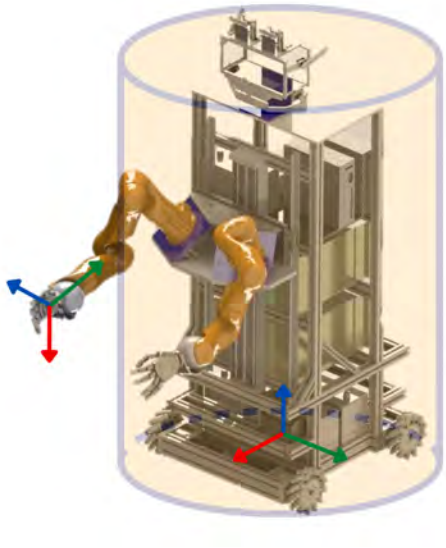


Figura 4.6: El modelo del robot se compone de un marco de referencia para la base y otro para el final de la cadena, además de un radio de seguridad que representa el cuerpo del robot.

Fuente: Elaboración propia

4.2.3. Coordinación del cuerpo completo

Aplicación de los criterios que rigen la coordinación

Con el fin de satisfacer los criterios definidos en la sección 4.1.1 es necesario ver cómo se aplican al sistema de coordinación escogido, y más específicamente, al caso de la mesa. Los siguientes son los que se consideran más importantes:

- Visión y orientación: se puede aplicar fácilmente, ya que al trabajar con la matriz de pesos se puede fomentar este movimiento directamente, gracias a que el giro del robot se concentra en una sola articulación. La forma más sencilla de lograrlo es midiendo directamente el ángulo que separa el vector que define el frente del robot y el vector que define la dirección de movimiento del robot (idealmente la línea recta entre el centro del robot y el objetivo).

- Límites de las articulaciones: el sistema de control ya cuenta con diversos métodos para desincentivar las articulaciones cerca de ese estado. Estos se encuentran a distintos niveles: mecánico, controlador de hardware y control de *software*. El último lo hace cambiando los pesos de las articulaciones cerca de un umbral ya definido. Existe la posibilidad de que este mecanismo no solo bloquee el robot para evitar dañarse, sino que fomente consecuentemente el uso de articulaciones lejos de sus límites. No obstante, las pruebas indican que aún así son comunes las situaciones donde el algoritmo de VIK se ve obligado a llevar ciertas articulaciones a su límite. Una solución ante esto es mantener la *pose* objetivo cerca del espacio de mayor manipulabilidad del brazo, ya que se sabe que ahí el brazo puede actuar con sus articulaciones lejos de los límites.

- Extensión del brazo: el propósito de este criterio es evitar acercarse a una singularidad

externa, siempre y cuando sea posible. Se puede diseñar el control de forma que sea menos probable caer en esta situación. Si se parte de que el brazo comienza en una postura poco extendida, basta con restringir el uso del manipulador y dejar que el movimiento sea únicamente en la base omnidireccional, hasta que se alcance una distancia pequeña con el objeto. La mesa, será entonces la principal limitante, puesto que interpone una restricción al movimiento de la base.

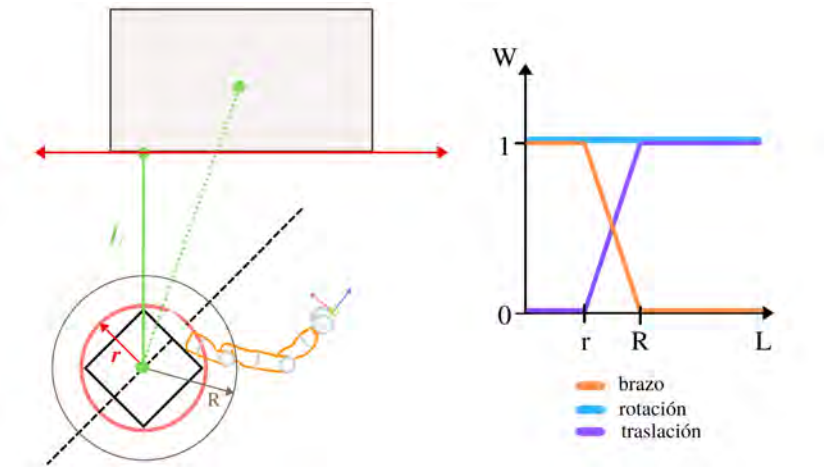


Figura 4.7: Principio de diseño del algoritmo para la coordinación del cuerpo completo del robot. A la izquierda se muestra una representación de la mesa y el robot. A la derecha, las funciones que definen los pesos de los elementos a coordinar

Fuente: Elaboración propia

Prioridad de los tipos de movimientos

Aplicar estos criterios simultáneamente permite definir la forma en que se coordina el movimiento de cuerpo completo. De todo la argumentación se resumen siguientes los requerimientos:

- Restringir los demás grados de libertad aparte del giro, hasta que el ángulo de desviación sea aceptable. Y en caso de ser necesario, fomentar el giro cuando el ángulo entre el frente del robot y la dirección hacia el objetivo sea mucho.
- Usar el brazo solo cuando la distancia entre el robot y la base sea poca.

A esto hay que sumar las restricciones que vienen directamente del problema a resolver, que se mencionan al inicio de la sección 4.2.1, se repiten aquí por comodidad: la base no debe colisionar con la mesa, favorecer la pose final del efector final y evitar que este toque la superficie. Ante este escenario se pueden organizar los movimientos del robot obedeciendo la lógica representada en la figura 3.3.

Los pesos de la base se definen en función de las distancias en el plano paralelo al suelo. Esto implica que no se toma en cuenta ni la posición del torso ni la altura del objetivo, para calcular la distancia que hay entre esta y la base del robot. La razón detrás de esta decisión

es que el riesgo que estos grados de libertad tienen de hacer que el robot sufra una colisión es diferente:

- Los movimientos de la base afectan todo el conjunto, mientras que el torso solo compromete al brazo.
- El rango de movimiento vertical es menor. Mientras que el robot puede estar a varios metros del objetivo (y la mesa sobre la que se encuentra), la diferencia de altura no será mayor al metro y medio. Además el rango de movimiento del torso es de tan solo 76,2 cm.

Esto permite crear una estrategia definida (representada en la figura 4.7). Los pesos de la traslación y el brazo dependen de la distancia L, que determina cuán cerca está el robot del punto más cercano de la mesa. Cuando la distancia es mayor a un radio R, entonces solo se quiere el robot se desplace. Al acercarse y estar entre R y el radio de seguridad r, conviven todos los grados de libertad, de manera proporcional con L. Abajo de r se inhabilita el desplazamiento y se deja el trabajo restante al brazo y al torso, puesto que si la base avanza más, tiene un alto riesgo de colisionar. En todo momento la rotación de la base se mantiene habilitada, puesto que favorece la manipulabilidad del sistema y no aumenta el riesgo de colisión.

Capítulo 5

Implementación en el simulador del ARCOS-Lab

En este capítulo se describe el proceso con el cual el sistema de control de cuerpo completo diseñado se integra dentro del simulador del robot humanoide. Por tanto, se describen los argumentos y bases con los que el algoritmo se implementa en forma de código y cómo este se relaciona con los módulos ya existentes.

5.1. Aspectos generales

El sistema, para poder cumplir con las condiciones hasta ahora estipuladas, requiere conocer o controlar al menos las siguientes variables:

- **La *pose* que corresponde al objeto que se quiere alcanzar, relativa al robot.**
- El peso de las articulaciones que describen el desplazamiento de la base.
- El peso de la articulación que describe la rotación de la base.
- La pose del último eslabón de la cadena cinemática.
- El estado de las articulaciones del robot en todo momento.
- La distancia entre la *pose* que se quiere alcanzar y el borde de la mesa sobre la que se encuentra, que es equivalente a la **posición y orientación de la mesa con respecto al objeto.**
- La menor distancia entre la base y el borde de la mesa.

Estas se podrían separar en dos grupos: las que dependen únicamente del robot (las primeras cinco) y las que dependen de la mesa (las últimas dos). De esta forma se pueden entender como características o estados de dos objetos cuyos parámetros pueden cambiar, lo cual resulta práctico de controlar desde un paradigma orientado a objetos, como se puede realizar en Python, lenguaje que se utiliza en la mayoría de los módulos del simulador del robot. Las variables resaltadas corresponden a las únicas dos entradas globales de todo el sistema de control, y por tanto, las demás son cálculos derivados, aunque tienen una importancia especial para la tarea que se está desarrollando.

Todas estas tareas únicamente se encargan de modificar los pesos, en respuesta al estado del robot y las condiciones en que se encuentra, y consecuentemente, se puede ejecutar tanto VFCLIK y PYROVITO sin ser modificados, y que así todo el control de cuerpo completo descanse en un tercer proceso, tal y como se explicó en la sección 4.2.1. Siguiendo la nomenclatura en inglés usada para el código del sistema de control, se puede llamar a este nuevo programa `wholebody_t_control`, que hace referencia al control de cuerpo completo en el contexto de la interacción con una mesa (en inglés *table*).
Nótese como todas las variables descritas aquí se encuentran en la figura 4.4, la cual describe el diseño general del algoritmo. Sin embargo, no aparecen las que corresponden a la mesa, puesto que es un elemento totalmente nuevo, del cual VFCLIK no tiene conocimiento. Toda la información referente a la superficie plana, se define y considera en el nuevo módulo.

5.2. Organización del código

La figura 5.1 muestra con más detalle el flujo de datos y la relación entre los distintos programas involucrados. Como puede verse las únicas dos variables de entrada son las mismas que se describieron antes, el objetivo y la mesa; estas deben ser dadas por el usuario, o bien por el sistema de control de alto nivel. Ambas deben ser descritas relativas a la posición actual del robot, que para el nivel de este proyecto siempre se considerará como la posición inicial. Además la figura hace claro que el control de cuerpo completo está dividido, VFCLIK se encarga de la cinemática de velocidades, pero es `wholebody_t_control` quien restringe ese resultado para cumplir la tarea de alcanzar el objetivo y evitar la mesa, todo usando solamente la matriz de pesos. Dentro del recuadro que representa a VFCLIK se puede ver el código interno que genera o recibe las variables utilizadas por el nuevo módulo. Además se incluye a PYROVITO, puesto que este se encarga de generar las gráficas correspondientes al cuerpo del robot, la mesa y el objetivo.

El nuevo módulo se organiza en dos partes, dividiendo así los procesos que tienen que ver con la mesa, de todos los demás, los cuales están relacionados con el robot, los elementos gráficos y la comunicación externa (ya sea que venga del usuario u de otro sistema. El primero es el subproceso `table` (mesa) y el segundo `wb_robot` (robot de cuerpo completo). La figura 5.1 es una representación en bloques de dicha organización. El proceso es el siguiente:

1. El módulo `wholebody_t_control` inicia los puertos de YARP y está a la espera de instrucciones.
2. Un proceso externo aporta la *pose* del objetivo y de la mesa.
3. `wb_robot` extrae las coordenadas x y y del objetivo recibido.
4. `wb_robot` solicita a VFCLIK el estado de las articulaciones del robot para determinar su posición actual.
5. `table` crea las variables requeridas para definir la mesa según los datos recibidos.
6. Se dibuja las figuras (mesa, cilindro del cuerpo, objetivo, entre otros).
7. `table` determina la distancia mínima entre el borde de la mesa y el robot, y con ello los pesos correspondientes.

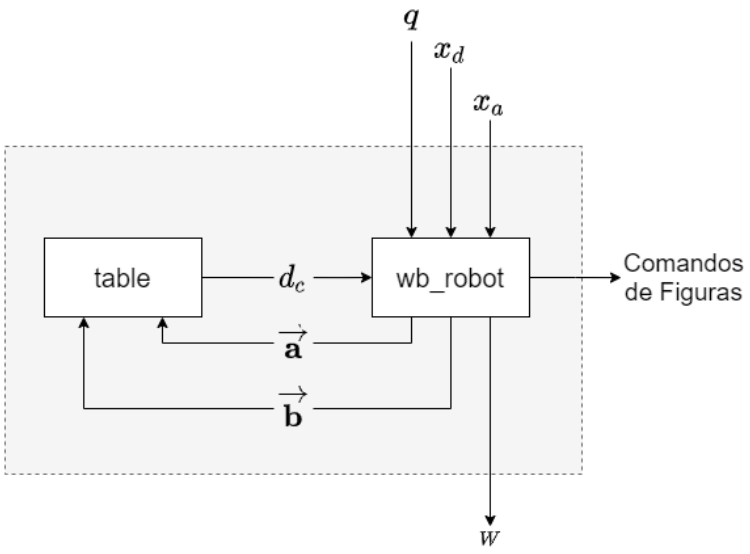


Figura 5.1: Flujo de la información entre los dos módulos que componen el sistema de coordinación

Fuente: Elaboración propia

8. `wb_robot` calcula los pesos respectivos y los envía
9. Se dibuja el punto más cercano sobre el borde de la mesa y el cilindro que representa al robot.
10. Se guardan los datos del ciclo.
11. Se verifica el cumplimiento de la tarea.

Este módulo debe activarse únicamente cuando se obtiene el objetivo y la ubicación relativa de la mesa, después de ello, el proceso debe repetirse continuamente hasta que se cumpla la tarea (ver detalles en la sección 5.11.2). El algoritmo 1 representa el código utilizado en el módulo, y por tanto explica todos los conceptos discutidos hasta el momento.

Algoritmo 1: Sistema de Control de Cuerpo Completo para alcanzar objetivos sobre una mesa

```
robot = wb_robot.inicialización()
while no hay señal de parada do
    if acaba de recibir un objetivo then
        goal = robot.localizaciónObjetivo()
        table = table.initialización(goal)
        robot.dibujarMesa()
    end
    if hay un objetivo en proceso then
        xy = robot.localización()
        t = table.distanciaCrítica(goal, xy)
        w = robot.calcularPesos()
        robot.enviarPesos(w)
        robot.dibujarIntercepto(t)
        robot.dibujarCilindro(xy)
        data = robot.guardarDatos()
        if tarea completada then
            objetivo en proceso = False
        end
        else if tarea fallida then
            objetivo en proceso = False
        end
    end
    plot(data)
    robot.reiniciarDatos()
end
robot.destrucción()
```

5.3. Determinar la mínima distancia con la mesa

La mesa se modela como una recta en el espacio que representa el borde de la superficie más cercano al robot; partiendo de que realmente fue bien identificada, no importa el resto de la geometría de la superficie, solo se necesita este dato para encontrar cuál es el punto más próximo a la circunferencia de seguridad. Este determinaría la distancia a la cual está el modelo cilíndrico del robot de colisionar, y por tanto, es de suma importancia que sea calculado con cuidado en cada ciclo del algoritmo. Este problema se puede resolver fácilmente al plantearlo bidimensionalmente como la intersección de dos rectas: la que define el borde de la mesa, y una perpendicular a la primera que además llegue al centro de la base del robot.
Para lograr esto, es necesario definir primero cómo se van a relacionar el borde y su mesa, de manera que sea intuitivo para el usuario, pero también conveniente a la hora de realizar cálculos geométricos. Una forma bastante sencilla es usar un vector $\mathbf{t}$ perpendicular a la arista, cuyo origen es la posición del objetivo ($\mathbf{b}$), y cuya magnitud define la distancia entre las dos, tal como se ve en la figura 5.2. Esto es deseable, puesto que se puede definir a partir de un ángulo que dirá que tan inclinada está la mesa, θ; y una distancia, que indica qué tanto está el objetivo dentro de la mesa. Las dimensiones y geometría de la superficie plana realmente

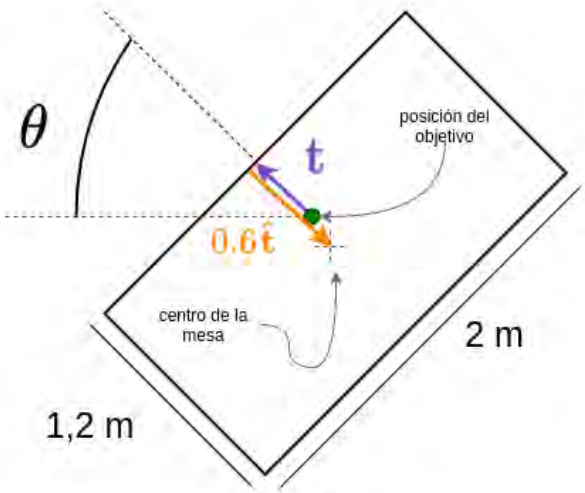


Figura 5.2: Elementos geométricos utilizados para modelar la mesa con respecto al objetivo.

Fuente: Elaboración propia

no son importantes, pues solo señalan dónde están los laterales y el fondo, los cuales no son críticos y están siendo ignorados.
Es importante señalar que **b** depende únicamente de la indicación dada por el usuario, y por tanto, se conoce a priori. No obstante, con el fin de no romper el flujo de datos en el simulador y permitir que el nuevo sistema se pueda utilizar en escenarios de control más complejos, el dato se obtendrá de manera indirecta desde el módulo de objetos de VFCLIK, por medio de su respectivo puerto de YARP.

Los demás elementos geométricos necesarios para este cálculo se muestran en la figura 5.3a, donde los vectores **a** y $\hat{\mathbf{n}}$ indican la posición del robot y la dirección de la distancia crítica, respectivamente. Puesto que el punto de intersección de una recta sobre una circunferencia se rige solamente por el ángulo de inclinación de esta (una demostración gráfica se ve en la figura 5.3b), el vector $\hat{\mathbf{n}} = (\cos -\theta, \sin -\theta)$.
De esta forma el problema de encontrar la intersección se puede plantear como un sistema lineal de la siguiente forma:

$$\mathbf{a} + s\mathbf{n} = \mathbf{c} + r\mathbf{m} \tag{5.1}$$

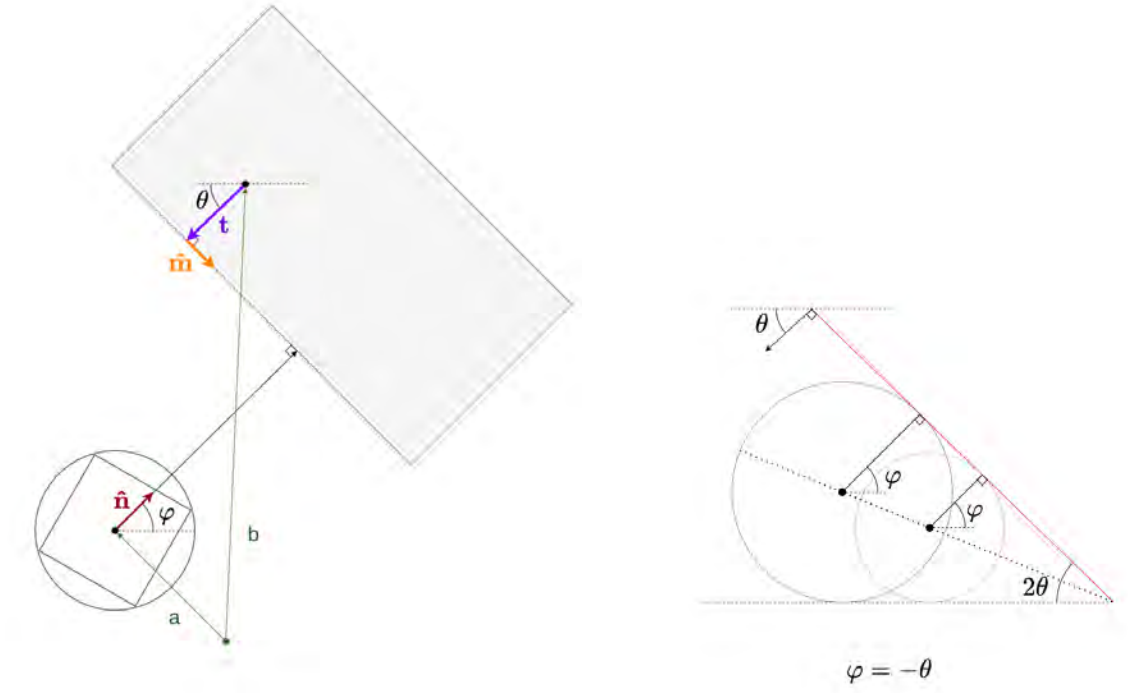
$\mathbf{n}$ = **b** - **a**
$\mathbf{c}$ = **b** + **t**
$\mathbf{m}$ = $\mathrm{Rot}(\hat{z}, 90\,\mathrm{deg})\mathbf{t}$
s, r = valores escalares

Solo es necesario hallar el valor de s o r para así encontrar el vector que indica la intersección relativo al origen: $\mathbf{w} = \mathbf{a} + s\mathbf{n} = \mathbf{c} + r\mathbf{m}$. Para este fin, el sistema se puede replantear

en su forma matricial:

$$\begin{pmatrix} | & | \\ \mathbf{n} & \mathbf{m} \\ | & | \end{pmatrix} \begin{pmatrix} s \\ r \end{pmatrix} = \mathbf{c} - \mathbf{a} \tag{5.2}$$

Donde la matriz de la izquierda tiene a los vectores $\mathbf{n}$ y $\mathbf{m}$ como columnas, y es invertible siempre y cuando ambos vectores no sean paralelos (linealmente dependientes) y, por tanto, no exista una intersección entre ellos.



(a) (b)

Figura 5.3: Construcción geométrica que ejemplifica el cálculo para hallar la distancia mínima entre el cilindro de seguridad y el borde de la mesa en cuestión. El punto crítico es encuentra en la intersección de las rectas definidas por los vectores unitarios $\hat{\mathbf{n}}$ y $\hat{\mathbf{m}}$. En (a) se muestran los elementos vectoriales y en (b) una demostración gráfica para hallar la relación entre θ y α

Fuente: Elaboración propia

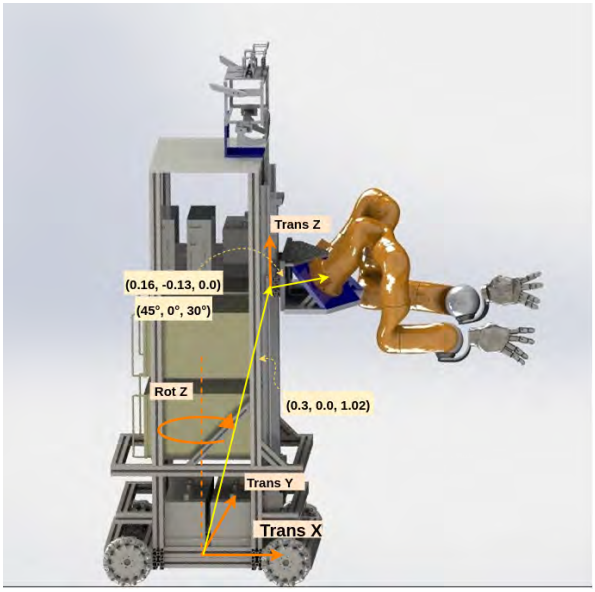


Figura 5.4: Descripción cinemática del robot. Las líneas anaranjadas representan los nuevos grados de libertad, y las flechas amarillas los vectores que describen las uniones entre ellos. Entre paréntesis se describen sus dimensiones cartesianas y angulares (cuando las hay)

Fuente: Elaboración propia

5.4. Ampliando la cadena cinemática

Tal como se mencionó en la sección 3.9 el primer paso a realizar es ampliar la descripción cinemática del robot en el simulador. Para lograr esto se deben crear nuevos archivos de configuración dentro de `robot_descriptions`. La opción más óptima es modificar la descripción actual para el ARCOS-bot.
La descripción cinemática ampliada no necesita hacer mayor diferencia entre los grados de libertad de postura y los de desplazamiento para poder solucionar la cinemática inversa. Siempre y cuando sean colocados en el orden correcto y siguiendo la geometría real del robot. En la figura 5.4 se puede ver cuáles son los parámetros que corresponden a cada uno de los segmentos nuevos del robot, en naranja se ven las articulaciones y en amarillo los marcos de referencia (cuando los hay). Como puede verse se crearon dos articulaciones de traslación y una de revolución para la base, y una de traslación en z para el torso.

Debido a que los segmentos de la base omnidireccional están embebidos en un mismo mecanismo, entonces no tienen localización física relativa al robot, ni tampoco entre ellos. Por tanto, al construir la cadena se pueden poner todos en un solo punto, que puede ser el origen mismo del robot, que en este caso es el centro de la base misma. El torso, por su parte, sí tiene una ubicación real, por lo que se debe asignar un marco de referencia de indique donde comienza. Finalmente se necesita establecer una transformación para los hombros del robot, sobre los que están montado los brazos. El código de la cadena ampliada se puede ver en los anexos.

Aparte de esto fue necesario asignarle una posición inicial y límites inferiores y superiores a cada articulación. Para los correspondientes a la base, lo más lógico es dejar las posiciones

iniciales en cero, puesto que esto significa que cada vez que se corra el proceso se tomará la ubicación inicial del robot como el origen. Sus límites tienen una consideración especial, ya que realmente pueden desplazarse indefinidamente. Empero, el sistema necesita de este valor. Tomando en cuenta que el enfoque de este trabajo es alcanzar posiciones cercanas, la limitación de la traslación en x y y puede dejarse en un valor alto, como 12 metros.
Lo mismo podría hacerse con la rotación, sin embargo, se puede argumentar que no resulta eficiente permitirle al robot dar más de media vuelta para lograr la tarea. Los límites aquí funcionarían, no para evitar un tope físico, sino para desincentivar movimientos poco razonables. Ahora bien, esto abre puerta a un problema de mínimos locales, ya que el solucionador cinemático moverá las articulaciones siguiendo el campo vectorial, el cual está construido con base únicamente en las distancias en cada punto, por lo que seguirá el camino más corto, aunque esto implique que haya un camino sin salida debido a un obstáculo o los límites de las articulaciones del brazo. Esto resulta más evidente en casos críticos donde la *pose* a la que se quiere llegar esté detrás. Por tanto es más conveniente dejar que el robot pueda girar libremente en cada dirección.
En el caso del torso móvil la posición inicial y los límites son interdependientes y deben ser cuidadosamente designados respetando la construcción del robot.

5.5. Flujo de información

Debido a que la ampliación del sistema de control se lleva a cabo prácticamente todo en un proceso a parte, es necesario crear un medio por el cual ambos programas puedan comunicarse entre sí. Una de las ventajas del diseño del simulador es que trabaja sobre un *Middleware*, lo que permite que transferir información fácilmente entre el nuevo módulo y los existentes sea más sencillo. En este caso, se realiza por medio de los puertos de YARP. A continuación se describe la función y uso que se le da a aquellos que se utilizan para este proyecto, de acuerdo con [33] y el estudio del código respectivo. Para cada uno se indica el nombre del puerto dentro de `wholebody_t_control` con el cual se conecta:

- Puerto de pesos del módulo `VF`
  - Nombre: `/lwr/right/vectorField/weight`
  - Descripción: le indica al solucionador de cinemática inversa los valores de la matriz de pesos para las articulaciones o para las variables cartesianas.
  - Uso: en cada ciclo de control se determina un peso para cada articulación con el fin de lograr la coordinación del cuerpo completo deseada.
  - Puerto local correspondiente: `/lwr/right/wholebody/weight_out`
- Puerto de salida de objetos del módulo `object_feeder`:
  - Nombre: `/lwr/right/ofeeder/objectOut`
  - Descripción: publica la pose correspondiente al objetivo que se está buscando actualmente
  - Uso: debido a que este sistema es tan solo una adición al simulador actual, debe recibir de parte de él y no por otro medio.
  - Puerto local correspondiente: `/lwr/right/wholebody/goal_in`

- Puerto de estado de las articulaciones del módulo `joint_sim`
  - Nombre: `/lwr/right/joint_sim/qout`
  - Descripción: indica cuál es la posición de cada una de las articulaciones del robot en todo momento.
  - Uso: esta información es necesaria para poder conocer la posición y orientación del robot, con lo cual es necesario para calcular posteriormente el peso de las articulaciones.
  - Puerto local correspondiente: `/lwr/right/wholebody/q_in`
- Puerto de salida de pose el módulo `VF`
  - Nombre: `/lwr/right/vectorField/pose`
  - Descripción: indica la *pose* actual del final de la cadena
  - Uso: es requerido para determinar cuándo se logra cumplir con la tarea solicitada. Además, su información se usa calcular ciertos parámetros considerados en las pruebas.
  - Puerto local correspondiente: `/lwr/right/wholebody/pose_in`
- Puerto de pesos del módulo `bridge`:
  - Nombre: `/lwr/right/bridge/weights`
  - Descripción: le indica al simulador cuál es el o los controladores que debe utilizar para mover el robot. Las opciones disponibles son: VFCLIK, Nullspace, jointcontroller, mechanism, xtra1, xtra2. En lo que respecta a este proyecto, solo las dos primeras son pertinentes, sus nombres son auto explicativos.
  - Uso: permite cambiar al modo de control de posiciones, con el cual se lleva a cabo el giro inicial del robot. Además se usa reiniciar el robot a su estado inicial en medio de pruebas.
- Puerto de entrada del módulo `joint_p_control`
  - Nombre: `/lwr/jpctrl/ref`
  - Descripción: recibe listas que corresponden a las posiciones deseadas para cada una de las articulaciones de la cadena cinemática.
  - Uso: permite indicar el ángulo para el giro inicial del robot.
  - Puerto local correspondiente: `/lwr/right/wholebody/joints_out`

A parte de estos, se hace uso de otros puertos que no son necesarios para el algoritmo de control de cuerpo completo, pero son requeridos para otras funciones asociadas a la realización de pruebas, como: registro de datos de diagnóstico, ayudas visuales y automatización.

- Puerto de entrada de objetos de PYROVITO
  - Nombre: `/lwr/roboviewer/objects:i`
  - Descripción: aquí se envían las instrucciones para crear objetos en el visualizados y cambiar sus propiedades, tales como color, *pose* y tamaño.

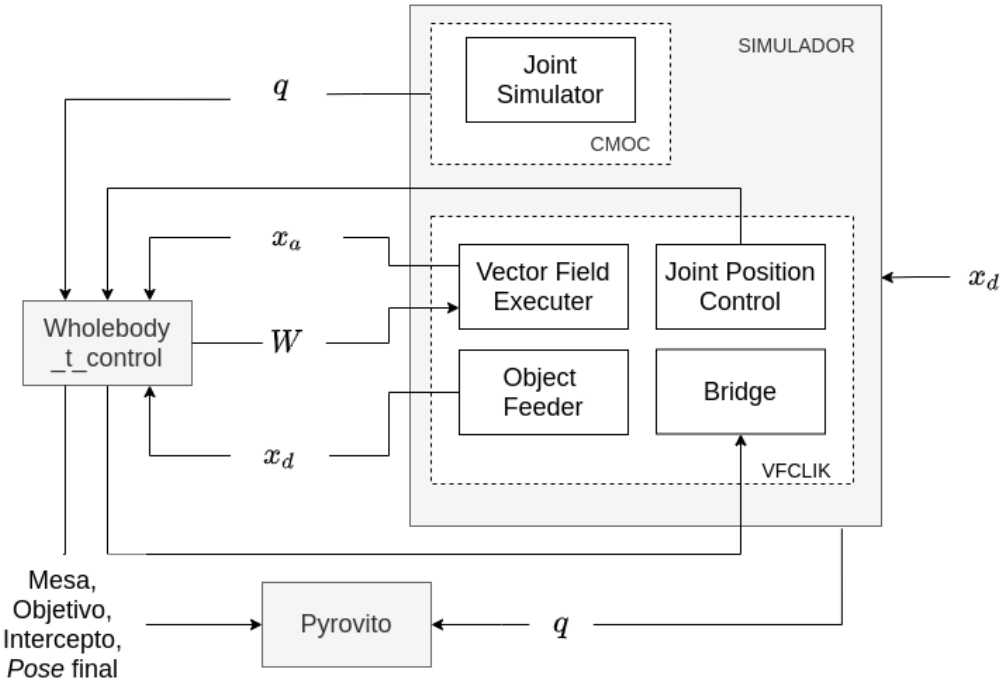


Figura 5.5: Descripción detallada de la información involucrada en el nuevo sistema de control

Fuente: Elaboración propia

  - Uso: se crean ayudas gráficas en la simulación para facilitar la verificación y caracterización visual de los resultados. Se pueden ver más detalles en la sección 5.6.1.
  - Puerto local correspondiente: `/lwr/right/wholebody/object_out`

- Puerto de objetos del módulo `object_feeder`

  - Nombre: `/lwr/right/ofeeder/object`
  - Descripción: Por medio de este puerto se le indica a VFCLIK la posición de los objetivos y los obstáculos. Basta con indicar sus respectivas *poses* y su tipo.
  - Uso: tampoco es utilizado por el nuevo módulo de control, sin embargo, es aquí donde eventualmente se envía la indicación del objetivo que se quiere alcanzar, por tanto se usa en las pruebas.

  La figura 5.5 describe de forma detallada el flujo de los datos involucrados en la ampliación del sistema de control. La mayoría de esta información se transmite por los puertos que se describen arriba, desde y hacia distintos submódulos de VFCLIK o a PYROVITO.

- Puerto del límite de las articulaciones del módulo `debug`
  - Nombre: `/lwr/debug/qdist`
  - Descripción: publica en forma de porcentaje, la cercanía que tienen todas las articulaciones de sus límites respectivos.
  - Uso: un promedio de esta lista de datos se utiliza como parámetro para clasificar el resultado de las pruebas.
  - Puerto local correspondiente: `/lwr/right/wholebody/qdist_in`

5.6. Elementos para la simulación

5.6.1. Figuras

El sistema de control no tiene necesidad de una representación visual, sin embargo, debido a que el objetivo es comprobar su funcionamiento con simulaciones resulta práctico que hayan elementos gráficos que ayuden a entender lo que está sucediendo con el robot y con el algoritmo de coordinación. El módulo correspondiente para esta tarea es PYROVITO, quien por si solo, se encarga de abrir un portal con una serie de figuras móviles que describen la cadena cinemática creada en `robot_description`, de acuerdo con las posiciones iniciales ahí indicadas. Además abre el puerto `/lwr/roboviewer/objects:i` al cual se pueden enviar instrucciones para crear distintas figuras, como cajas, cilindros, marcos de referencia, flechas y esferas.
Los mensajes se deben enviar dentro de una lista con la siguiente estructura:

```
[[1, "type", "nombre"],
[1, "scale", x, y, z,
[1, "color", r, g, b],
[1, "timeout" t],
[1, "pose" + pose]]
```

Donde el número al inicio de cada lista corresponde al identificador de la figura. De esta forma el sistema lleva un registro de los elementos creados y permite modificar sus propiedades posteriormente. Con `nombre` se indica si es una esfera, caja, etc. La variable `t` define la cantidad de tiempo que se quiere que la figura permanezca después de que se envía el comando. El color se envía en formato RGB, con valores individuales entre 0 y 1. Finalmente, `pose` debe ser una lista que corresponda a los 16 elementos de una matriz de transformación homogénea.
Las siguientes son las figuras adicionales que se crearon por medio de este puerto, que además pueden verse en la figura 5.6, la cual muestra una toma real de una simulación:

- Un cilindro que represente el radio de seguridad de la base, de forma que permita comprobar que en ningún momento el robot colisione contra la mesa. Se hizo de 70 cm de alto, para así no ocultar la articulación prismática que corresponde al torso.
- Dos marcos de referencia. Uno para la *pose* de la mano y otro para el objetivo sobre la mesa. Estos cumplen varios propósitos: (i) hacer más fácil la comprensión de cómo se orienta la mano con respecto al brazo, (ii) dar a entender la forma en que se quiere que el efector final interactúe con el objetivo y (iii) comprobación visual de que se logró la tarea solicitada

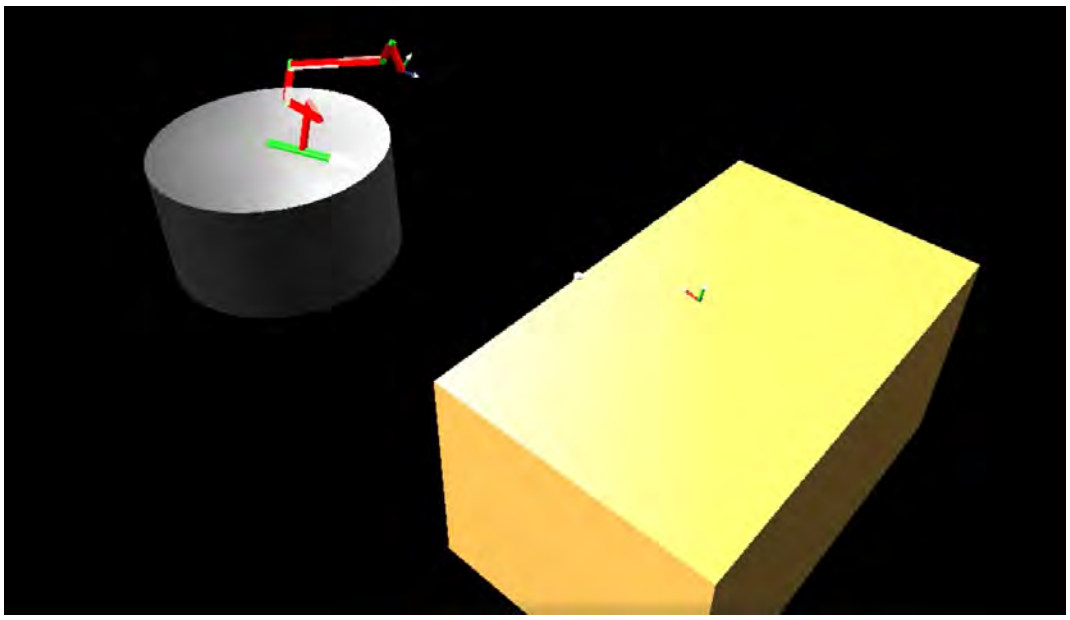


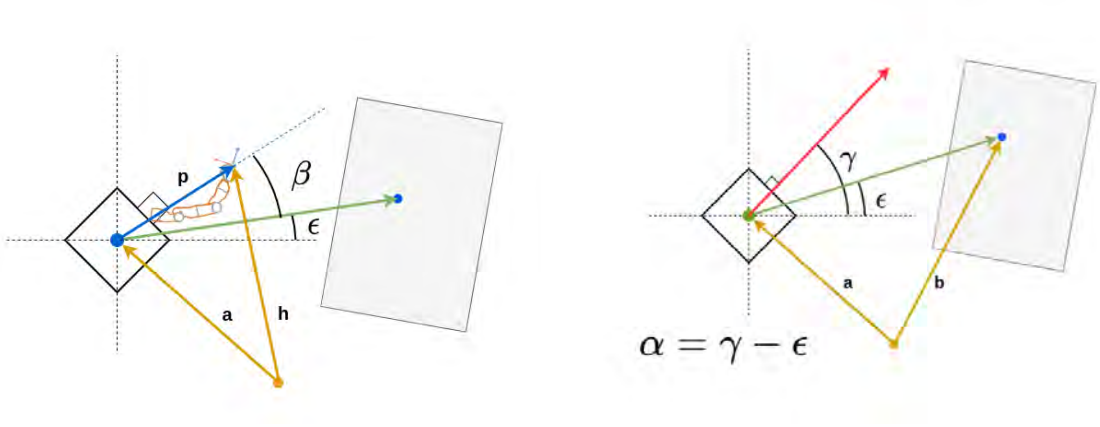
Figura 5.6: Captura de una simulación donde se pueden ver las figuras para ayudar a la comprobación visual: un cilindro gris que representa el radio de seguridad, junto a una flecha verde que indica el frente del robot, una caja amarilla para la mesa, conjuntos de flechas para los marcos de referencia de la *pose* del efector final y el objetivo; y una esfera blanca para el punto de más próximo entre la línea de la mesa y el cilindro.

Fuente: Elaboración Propia

- Una flecha centrada sobre la base del robot y orientada hacia el frente del mismo, ya que al solo mostrarse uno de los brazos no es tan fácil interpretar hacia dónde está viendo el robot durante la simulación.
- Una esfera móvil que indique el punto sobre el borde de la mesa que está más cercano a colisionar con el cilindro que representa la base. Este es sumamente importante, pues en realidad es una representación gráfica de la distancia crítica con la cuál se calculan los pesos del sistema de coordinación, y es fundamental saber que es correcto.
- Una caja que represente la mesa, de forma que su posición y orientación coincidan con el escenario solicitado al algoritmo.

5.6.2. Indicadores de desviación

Para crear algunas de estas figuras y para mejorar la comprensión de lo que está sucediendo en la simulación, es necesario definir ciertos parámetros. Los dos más importantes son los de desviación, los cuales indican qué tan grande es el ángulo entre la dirección del robot o del brazo, con respecto al vector que une el centro de la base con el objetivo final, puede verse una representación gráfica en la figura 5.7 Haciéndose referencia a esta misma figura, para calcular la desviación del robot, se puede encontrar fácilmente el ángulo ϵ a partir de $\mathbf{o} = \mathbf{b} - \mathbf{a}$ y γ como la posición de la articulación de rotación del robot. Con el fin de simplificar su interpretación, el indicador de desviación se define como la diferencia entre los dos, de forma que sea positivo



(a) Desviación del brazo

(b) Desviación del robot

Figura 5.7: Elementos geométricos utilizado para calcular los indicadores de desviación

Fuente: Elaboración propia

cuando el objetivo está a la derecha y negativo cuando está a la izquierda. El cálculo del ángulo del brazo es análogo, la única diferencia es que el primer vector de la resta, **h**, se encuentra utilizando la *pose* actual del efector final, proveída por el simulador.

5.7. Manejo de la redundancia

Con el fin de poder definir una función que controle los pesos de las articulaciones, se realizaron pruebas para determinar la forma en que VFCLIK manejaría la redundancia del sistema y cómo esto resultaría en el movimiento de cada parte del robot.
El primer resultado importante es que los grados de libertad de la base, por si solos, son capaces de lograr cualquier movimiento del efector final sobre el plano XY. Ante esta posibilidad el algoritmo no recurre a utilizar el brazo. Lo mismo pasa si se quiere realizar un desplazamiento vertical: solo se hace uso del torso. Este es un comportamiento esperable si se considera que ninguno de esos grados de libertad es dependiente de otro, es decir, no importa cuanto se muevan siempre van a conservar la misma orientación entre sí. Esto no ocurre en el brazo. Por tanto, las articulaciones nuevas requieren ser ponderadas con respecto al resto del robot.
Es importante resaltar que estos resultados se obtuvieron utilizando KBD-CART-CMD, el cual solicita al VIK velocidades de prueba, en x, y, z o rotacionalmente sobre estos mismos ejes, todo según el marco de referencia global. Como las direcciones de las articulaciones prismáticas se definieron con los mismos ejes, siempre que se solicite una de dichas velocidades, basta con usar únicamente las respectivos articulaciones y se puede dejar en reposo el resto de la cadena.
Es por esto que es necesario utilizar VFCLIK con atractores para conocer la forma en que realmente se soluciona la redundancia. Con el fin de poder determinar el uso que se le da a cada tipo, en la figura 5.8 se muestra un registro del desplazamiento relativo promedio para las articulaciones de cada uno de los tres tipos de movimiento discutidos. Se realizó el estudio para distancias de 1, 3 y 5 m, todos al frente del robot. La rotaciones se midieron en radianes y el desplazamiento se reporta adimensional con respecto a la distancia inicial por recorrer.

Es importante notar que debido a que en las pruebas la rotación del objetivo es muy similar a la inicial, la mayor trabajo solicitado es de desplazamiento. Es por esto que el algoritmo hace poco uso del brazo y más bien procura explotar la rotación para mover el efector final en la velocidad que el campo vectorial le exige, lo que explica por qué aumenta el uso de esta articulación virtual conforme se aleja el objetivo.
El resultado indica que el algoritmo recurre al uso de todos los grados de libertad, pero que reserva el brazo para realizar velocidades complejas, las cuales se dan cuando el robot se acerque a poses con orientaciones complejas, o bien cuando los demás grados de libertado no estén disponibles, como se ve en la sección 5.9.

5.8. Límites de las articulaciones

Las primeras pruebas con el cuerpo completo determinan que el método que ya posee el sistema de control para evitar el límite de las articulaciones, es efectivo. Aún así es posible llevar ciertas juntas a este estado cuando se simulan casos críticos donde no se fomenta adecuadamente la rotación del cuerpo y el brazo se ve obligado a tomar posturas no deseadas. Esto se puede ver en la figura 5.9 donde el objetivo se encuentra detrás del robot, pero aún no se encuentra implementado el ángulo de desviación como un parámetro que modifica el comportamiento del robot.

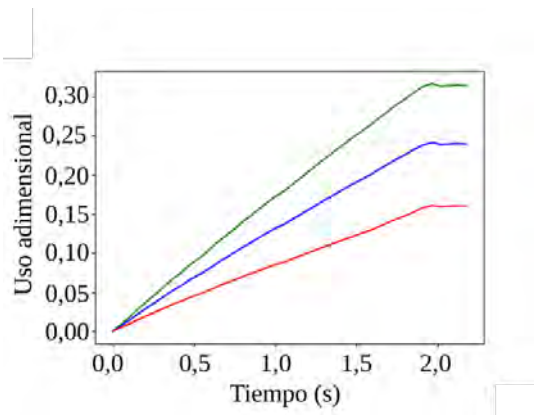
Es así como se descarta la hipótesis inicial que indica que al desincentivar las articulaciones que están cerca de sus límites, se induce indirectamente el uso de aquellos grados de libertad que no tienen límites o que están lejos de ellos, de forma que el brazo se mantendrá en un estado de alta manipulabilidad. Una explicación clara de esto es que el rango en que se desincentiva una articulación no es lo suficientemente grande como para evitar que el robot entre en un estado desfavorable para el resto de la cadena durante lo que resta de la simulación. Es decir, cuando se inhabilita un grado de libertad por estar cerca de su límite, es porque el robot ya se encuentra en una postura inadecuada, y de la que no se puede recuperar únicamente cambiando los pesos de la cadena.

El cambio consecuente es evitar que se llegue a estados como estos, y la forma más sencilla de lograrlo es fomentando que la posición del efector final se ubique siempre en la parte frontal relativa al cuerpo del robot. Para ello se debe procurar que el frente del robot se dirija siempre en dirección al objetivo.

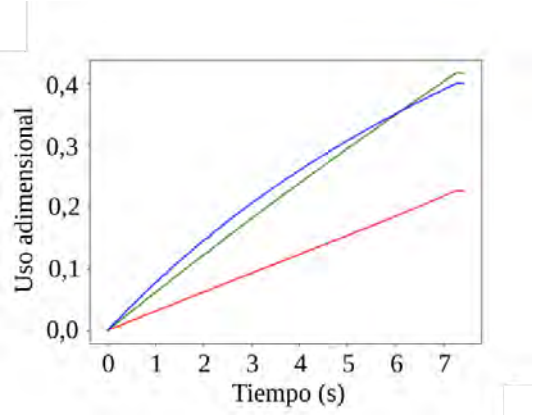
5.9. Pruebas de giro

5.9.1. Giro ante un mismo ángulo inicial de desviación

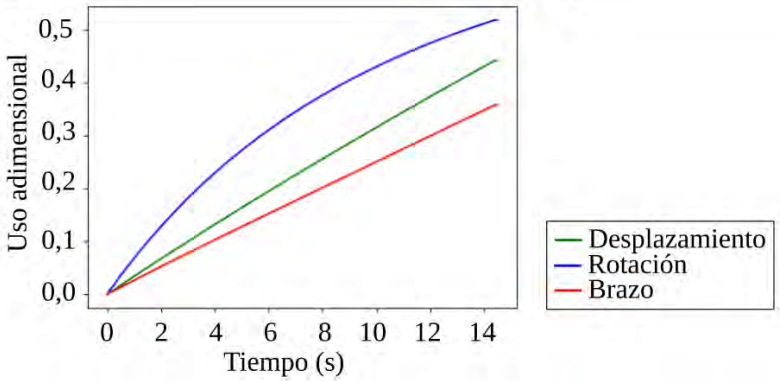
Mediante un estudio más preciso del comportamiento que tiene el grado de libertad rotacional de la base, se determina que, contrario al planteamiento original, la cadena cinemática no siempre tiende a moverse de forma que el frente del robot se mantenga en la dirección del objetivo. Esta primer suposición se basa en la hipótesis de que el algoritmo de VIK, ante tantos grados redundantes, va a tender a aquellas posturas donde hay mayor manipulabilidad. Esto tiene asidero en el hecho de que dentro del algoritmo se integra un método que por medio de un SVD favorece los resultados que alejan a la matriz jacobiana de las singularidades. Ahora bien, los resultados obtenidos por [4], donde se determina que la zona donde el robot tiene mayor capacidad de manipular es al frente, se realizaron para ambos brazos. Estudios



(a) Un metro de distancia



(b) Tres metros de distancia



(c) Cinco metros de distancia

Figura 5.8: Pruebas para determinar el uso de los tipos de movimiento. Se muestra un promedio adimensional del desplazamiento de cada tipo de articulación. En verde se muestra el desplazamiento en el plano XY; en azul la rotación, y en verde el brazo. Las pruebas corresponden a: (a) la pose final es equivalente a poner la palma de la mano hacia el cuerpo, con los dedos girados levemente hacia abajo; en (b) la *pose* del objetivo es muy similar a la inicial.

Fuente: Elaboración propia

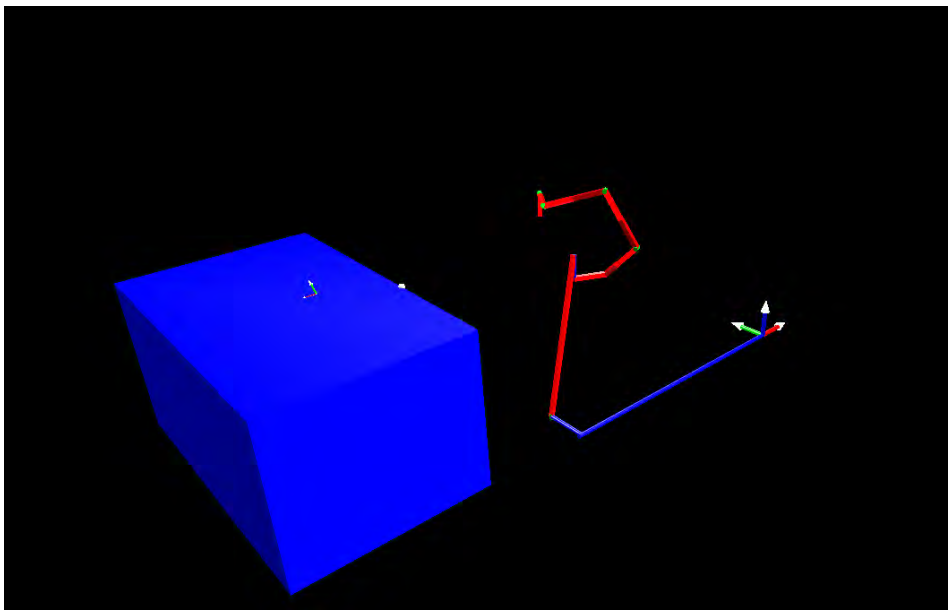
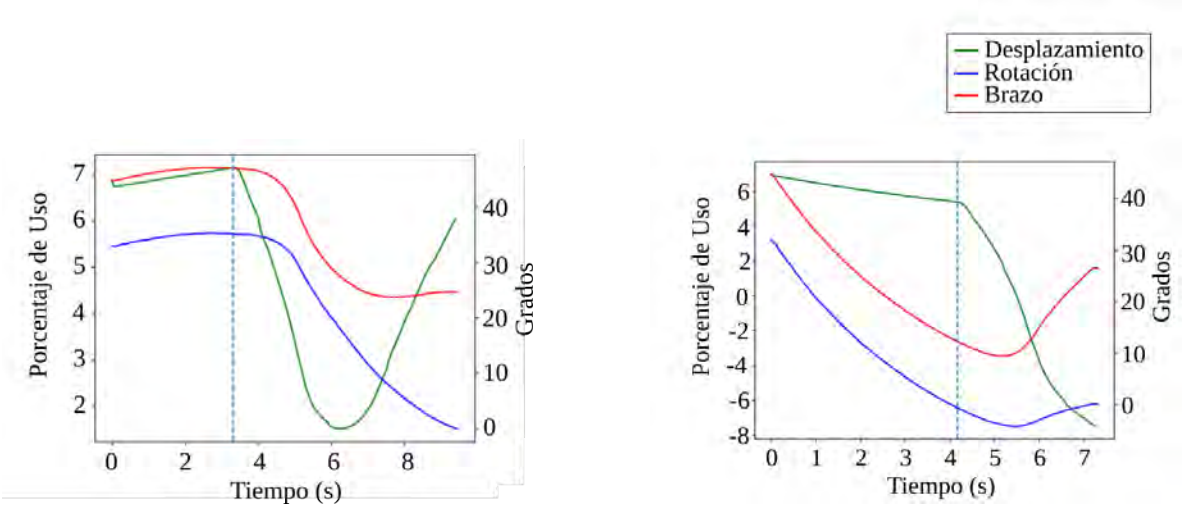


Figura 5.9: Si no se fomenta el giro del robot, se producen situaciones como la de esta imagen, donde el robot intenta pasa su brazo por encima de su cuerpo para alcanzar un objetivo que se encuentra detrás de sí.

Fuente: Elaboración propia

del KUKA LBR de forma aislada, como [45], muestran que en realidad la se encuentra cerca del eje normal a la base del brazo.

La tendencia de las pruebas se puede resumir con la figura 5.10. Aquí se grafica en verde un promedio del porcentaje de proximidad de las articulaciones con sus respectivos límites; en azul se muestra el ángulo de desviación entre el ángulo del en cada uno de los casos, se puede identificar dos distintos comportamientos, diferenciados claramente por el momento donde se habilita el uso de los brazos (representado por una línea punteada). Antes de este punto, el robot se desplaza y rota, moviéndose sin cambio de postura hacia el objetivo. Para el caso (b), donde la *pose* final es muy sencilla todos los parámetros disminuyen, es decir, el robot y la mano se orientan hacia el objetivo; empero, para (b) sucede lo contrario, donde los parámetros de desviación del robot aumentan, indicando que se acomoda para avanzar de forma lateral, llevando la mirada hasta casi 50° del atractor.
El segundo periodo es más complejo. La curvatura de las gráficas está relacionada con el cambio lineal del peso de la base hacia el peso del brazo (ver la figura 4.7), y su punto de inflexión coincide con el momento en que se le da un peso nulo al primero.
Otro aspecto de gran importancia es el comportamiento de las desviaciones del brazo y del robot. 5.6.2. La desviación del robot es también un sinónimo de qué tan cerca está la mirada del robot hacia la *pose* final; puede verse que en ambos casos termina alrededor de los 25°, y que aumenta después de que se encuentra con la mesa. Este resultado significa que el robot, al verse sin los grados de libertad de desplazamiento, acomoda su cuerpo en la posición que le dé más alcance. Esto es igual al comportamiento del humano, quien al tener los brazos en los costados, gana parte del ancho de su tronco al girarse.



(a) (b)

Figura 5.10: Pruebas de acercamiento, con el objetivo en las coordenadas (2,-2) y el borde la mesa con una inclinación de 50°, a 40 cm del objeto. El peso de la rotación de la base se mantuvo en 1.0 a lo largo de toda la simulación. En (a) la pose final es equivalente a poner la palma de la mano hacia el cuerpo, con los dedos girados levemente hacia abajo; en (b) la *pose* del objetivo es muy similar a la inicial.

Fuente: Elaboración propia

5.9.2. Giro ante ángulos de desviación grandes

Hasta ahora se ha hablado de cómo el sistema de control actúa ante un mismo escenario con distintas *poses* finales y cómo esto influye en la forma en que la base gira y se desplaza. Es necesario complementar esa información estudiando qué sucede con el robot si se varía las condiciones con objetivos dispersos a su alrededor.
Los resultados son variados. Un ejemplo claro son los casos simples, donde la mesa se encuentra perpendicular a la trayectoria directa (es decir, tangencial al perímetro circundante), aquí el desenlace es positivo para la mayoría de los casos, sin embargo, conforme la desviación inicial se acerca a los 180°, se vuelven más ocurrentes las advertencias de límite de articulaciones, y consecuentemente la incapacidad de cumplir la tarea. A su vez, se aumenta la tendencia de avanzar en lateralmente, e incluso en retroceso, lo que claramente está interrelacionado con el problema anterior. Otro aspecto que habla en contra de este comportamiento, es que incumple uno de los principales criterios de diseño estipulados en la sección 4.1.2, el cual establece que, en la medida de lo posible, se debe fomentar que el robot avance hacia el frente, de forma que se tenga retroalimentación visual en todo momento.

5.10. Giro inicial

El resultado de las pruebas descritas atrás hace clara la necesidad de diseñar un método que fomente el giro del robot, con el fin de solucionar dos problemas: (i) la tendencia pobre a avanzar de manera frontal y (ii) los movimientos desfavorables y de apariencia poco natural que lleva a cabo el brazo. A pesar de que ambos están estrechamente relacionados con el giro del robot, divergen en el hecho de que el segundo depende, más que todo, en cuánto se voltee inicialmente, mientras que el primero requiere además de estímulo constante de la rotación. Estos dos incentivos de giro

5.10.1. Relación con el control de cuerpo completo

El sistema de coordinación del cuerpo completo depende de la forma en que el algoritmo del VIK solucione las velocidades impuestas por el campo vectorial. El único cambio que se está realizando para ejercer control sobre él es restringiéndole cuáles variables puede usar, y en qué medida. Es decir, cambiando las dimensiones de la matriz jacobiana y ponderando el efecto de las variables. Lo que resulte a partir de este punto depende del algoritmo y de las condiciones. Por tanto, con este método, no se pueden inducir velocidades que obedezcan a otros propósitos si no es modificando el campo vectorial o afectando el comportamiento del efector final.
Las pruebas han demostrado que el giro no entra en acción aún cuando intuitivamente parece ser el movimiento más óptimo. Esto porque la conducta que muestra el sistema carece de planificación y responde únicamente a lo que el campo de vectores indique para cada posición. En otras palabras, es un problema espacial y no temporal.
Ante esta situación existen dos posibles opciones:

1. Restringir el sistema hasta forzar el comportamiento deseado: si el se anulan los suficientes grados de libertad, se puede indeterminar el sistema, de forma que aunque no tenga una solución su resultado sea favorable. Un caso como esto es restringir todas las articulaciones salvo la de rotación, ante esta situación la solución con el menor error es girar hacia el objetivo. Sin embargo, esto significa que la desviación del brazo cerá cercana a cero, pero no necesariamente la del robot, Para ello sería necesario posicionar el efector final al frente del robot, lo cual complica la estrategia.

2. Utilizar un método distinto para ejercer el giro del robot: el sistema de control ofrece distintos métodos de control cinemático (ver descripción del puerto `/bridge` en la sección 5.5), algunos de los cuales se pueden usar incluso de forma simultánea, como el VF y el espacio nulo. Hay dos alternativas:

   a) Utilizar el control por campo vectorial junto al de espacio nulo para generar una velocidad nueva para la rotación de la base que no afecte la tarea anterior. Esto requeriría plantear todo un sistema matemático nuevo que no se ha implementado antes para el robot. Pero tiene la ventaja de que permitiría un control continuo de la orientación a lo largo de todo el proceso.

   b) Deshabilitar el control por campo vectorial y usar únicamente el de posición de articulaciones para girar la base de manera que se logre exactamente la orientación deseada. Este método es considerablemente más sencillo de implementar y hace que el sistema sea mucho más predecible, repetible y fácil de estudiar, ya que reduce el problema de cuerpo completo a acercarse a una mesa que se encuentra al frente.

De entre las tres opciones presentadas arriba la primera tiene la única ventaja de que no hace uso solo de los pesos, pero con el costo de que necesario forzar la postura inicial. La estrategia $2a$ es posiblemente la más efectiva, puesto que ofrece la posibilidad de controlar el giro en todo momento, sin embargo significa el diseño de un módulo considerablemente complejo. Finalmente $2b$, aunque implica dividir el proceso de control en dos etapas, es quien generaría (en muchos de los casos) los movimientos que se asemejan a más los del humano, el cual usualmente gira antes de mover su brazo o desplazarse.

5.10.2. Implementación dentro del ciclo de control

Debido a que depende solamente del ángulo de desviación del robot, el cuál es un parámetro que se usa ya para otros propósitos, no requiere de cálculos adicionales. La secuencia de control es la siguiente: (i) se calcula el ángulo de desviación α, (ii) se cambia a control por posición de articulaciones, (iii) se le indica a la articulación de rotación que gire α, (iv) se verifica que se haya completado el movimiento viendo el estado del la variable q, (v) se envía a VFCLIK el objetivo final y (vi) se cambia a control por campo vectorial. El orden de los últimos dos pasos es importante, ya que si no se agrega el nuevo atractor, la cadena regresará al que tenía inicialmente y el robot se moverá de forma indeseada.

5.11. Pruebas finales

Al probar el sistema diseñado en el visualizador de PYROVITO se puede ver que el método control de cuerpo completo se desempeña muy bien en la mayoría de los caso, y lo hace con movimientos muy naturales. No obstante, con el fin de poder determinar su efectividad cuantitativamente y comprender cuáles son sus debilidades, es necesario someterlo a pruebas sistematizadas.

5.11.1. Escogencia de parámetros

Tal como se explica al inicio de este capítulo, la forma en que se restringe el problema en cuestión, y los modelos escogidos para representarlo, hacen que los parámetros implicados sean pocos. Sin embargo, sigue siendo necesario considerar solo ciertos de ellos a la hora de realizar pruebas para verificar el funcionamiento del sistema. A continuación se muestra una selección:

1. **Orientación del objetivo**: tal como indican los resultados discutidos en la sección 5.9.1, el efecto que tiene esta medida es de suma importancia ya que afecta todo el cuerpo del robot. Ahora bien, debido a que se compone a su vez de tres distintas variables, todas con rangos muy amplios, se decide asociarlas a poses requeridas por objetos comunes en los campos de trabajo en los que se desea que trabaje el ARCOS-bot. Se determinan tres distintas opciones suponiendo el uso de una mano como elemento final de la cadena, las cuales pueden verse en la figura 5.11. Aquí se explican sus posibles aplicaciones:

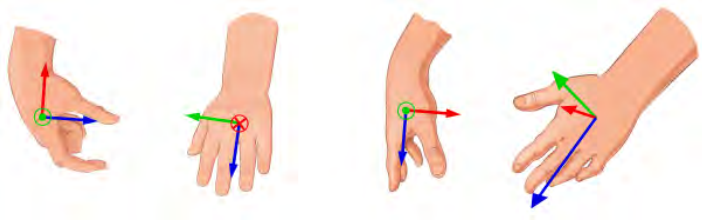


Figura 5.11: Las tres distintas *poses* que se tomaron como parámetro. Corresponden a los nombres de *asa*, *tomar* y *botón*, respectivamente. A la derecha puede verse cómo corresponden los ejes del marco de referencias al efector final del robot si fuese una mano antropomórfica.

Fuente: Elaboración propia basado en https://www.freepik.com/vectors/hand»Hand vector created by ddraw

- La palma girada hacia el cuerpo, con el pulgar hacia arriba. Esta es la postura utilizada para tomar herramientas con un asa, que son muy comunes en ambientes de cocina o comedores, como una jarra, pichel y teteras.
- La palma dirigida hacia la izquierda, con el pulgar hacia arriba. Dejando la posición de los dedos como una variable aparte, esta opción puede ser usada para empujar objetos con la palma, o con la punta de los dedos (equivalente a presionar botones).
- La palma dirigida hacia abajo con los dedos hacia el frente. Es tal vez, la posición preferida para tomar objetos pequeños que reposan sobre una superficie, y por tanto, también para depositarlos.

Existen, sin duda, muchas otras opciones, pero se considera que estas representan la gran mayoría de los casos. Es importante mencionar que, con el fin de aislarla de la orientación de la mesa, y así evitar dependencia entre parámentros, esta se define relativa a la *pose* inicial del efector final del robot.

2. **Distancia del objeto al borde de la mesa**: esto se traduce a qué tan dentro en la superficie plana sobre la que está el objeto que eventualmente se manipulará. Esta medida depende ampliamente de las prácticas usuales del humano y a las dimensiones usuales de sus espacios de trabajo. En general se recomienda que la profundidad de las superficies de trabajo no excedan los 60 cm de profundidad [46], aunque en cocinas y comedores es posible encontrarlas de hasta un metro, sin embargo, por las dimensiones del cuerpo humano, es poco probable posicionar objetos a tal distancia. Considerando estas medidas y tomando en cuenta que el robot tiene un límite físico que no tiene sentido sobre pasar, se decide tomar como valores de 15 y 45 cm, tal como se ve en la figura 5.12.

3. **Orientación de la mesa**: es importante resaltar, que *a priori* se sabe que el algoritmo no es capaz de desplazar el robot cuando la recta del borde de la mesa interseca la circunferencia de seguridad del robot. Esto sucederá con mayor facilidad con ángulos grandes. Con el fin de evaluar el sistema en el amplio espacio de posibilidades restantes se eligió un rango cercano a las zonas conflictivas: -45°, -30°, 0°, 30° y 45°. Es necesario incluir los ángulos negativos debido a que el estudio se está realizando para el brazo derecho, por lo que no hay simetría. En la figura 5.13 se muestra una representación de los ángulos positivos.

Figura 5.12: El parámetro de distancia del objetivo al borde de la mesa toma los valores 15 cm y 45 cm

Fuente: Elaboración propia

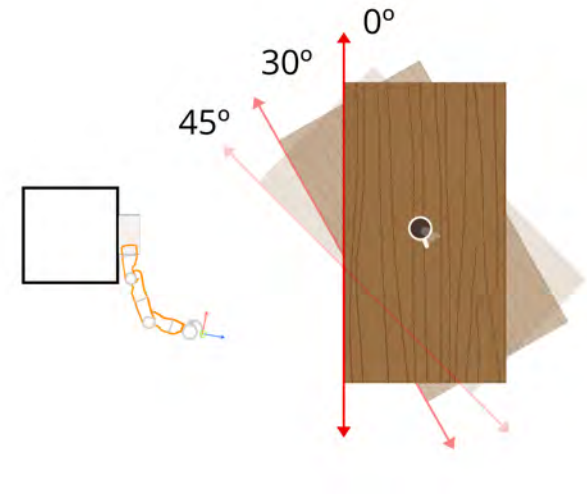


Figura 5.13: El parámetro de orientación de la mesa toma los valores -45°, -30°, 0°, 30° y 45°. Aquí se representan los positivos. Nótese como este parámetro no modifica la posición ni orientación del objetivo.

Fuente: Elaboración propia

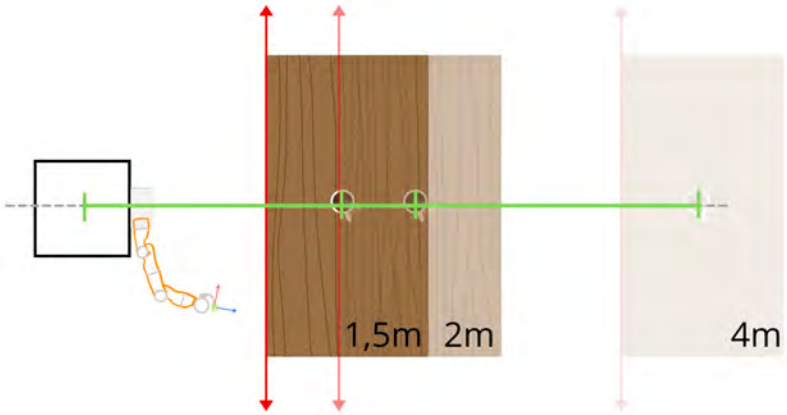


Figura 5.14: El parámetro de distancia hasta el objetivo toma los valores de 1,5 m; 2 m y 4 m. La medida se realiza desde el origen del robot y hasta el objetivo, la mesa se muestra para referencia, ya que su posición y orientación se definen mediante los parámetros anteriores.

Fuente: Elaboración propia

4. **Distancia hasta el objetivo**: ante el hecho que el sistema de control le da la capacidad al robot de girar antes de acercarse al objetivo, es más sencillo considerar este parámetro, ya que se no hay necesidad de dar una posición en y, sino solo en x. Ahora bien, a pesar de que este factor no representa un factor decisivo para una persona que ejecute una tarea similar a esta, sí lo es para este sistema de control, debido a que como se explicó en la sección 5.10.1, este es un problema espacial, y por tanto el resultado final va a depender de la posición inicial del objetivo con respecto al robot.
   Tal como se ve en la figura 5.14, esta medida se hace del centro del cuerpo hasta la *pose* solicitada, y por tanto, si se quiere que el robot requiera desplazarse, debe entonces ser mayor a la suma a la longitud del brazo, que es de alrededor de 1,3 m. El extremo opuesto, se limita sabiendo que los espacios de trabajo usualmente son reducidos, y los movimientos libres de obstáculos, como el que se considera aquí, son cortos. Se decide entonces usar un valor máximo de 4 m. La lista se completa con un valor medio de 2 m.

5. **Postura inicial del brazo**: al igual que se argumenta con la distancia del objetivo, los factores que impliquen cambios en la trayectoria pueden tener efectos muy importantes en el resultado. Además este parámetro si se define una posición inicial baja, crece la posibilidad de que la mano tenga que desplazarse cerca de la mesa, y que pueda querer atravesar la superficie. Es importante evaluar que no suceda. Se definen entonces dos variaciones, una con el brazo a una altura media, y otra con el brazo en un posición baja. Se dejan por fuera los casos con la mano desplazada hacia la izquierda o derecha, puesto que son posturas poco probables para iniciar una tarea.

El único factor importante que no se incluye en la lista anterior es la altura de la mesa, la cual define, además, qué tan alto está el objetivo, por lo que no cabe duda que tiene efecto en el modelo. No obstante, con el fin de disminuir el número de variables involucradas se decide

dejarlo por fuera, y dejar un valor crítico en compensación. Las alturas de las superficies para trabajo de pie (como cocinas o talleres) ronda en valores de 90 cm, por lo que se usa 1 m en las pruebas.

5.11.2. Ejecución de la pruebas

Para facilitar la ejecución y asegurar la correcta asignación de los parámetros, las pruebas se corrieron de forma automatizada, para lo cual fue necesario adaptar el ciclo de control del algoritmo. Además de esto se agregaron funciones para la guardar datos y figuras.
Para reiniciar el estado del robot después de cada uno de los escenarios, se incluyó en el algoritmo una secuencia que utiliza control de posición de articulaciones, para regresarlo siempre a una misma postura inicial. La siguiente combinación de parámetros comienza cuando se verifica que las articulaciones están estáticas. Un mecanismo similar se utiliza para identificar el cumplimiento o fallo de cada una simulaciones: el primero se alcanza cuando las 16 variables de la matriz homogénea del efector final coinciden con exactitud de una milésima con la del objetivo. Por último, se clasifica una simulación como fallida cuando el ángulo de desviación del robot no ha variado en dos milésimas de radian en los últimos cinco ciclos. Se utilizó este parámetro debido a que es el cambia con mayor rapidez al acercase a la mesa y no se ve afectado por las singularidades características de este estado, como sí pasa con la desviación del brazo y el porcentaje de uso de las articulaciones.

5.12. Resultados y discusión

5.12.1. Postura inicial

El hecho más contundente de los pruebas es la efectividad nula que tiene el sistema para cumplir la tarea cuando el efector final está abajo. De las 90 combinaciones que inician con esta postura, todas llevaron a que el final de la cadena atravesara la superficie de la mesa, aunque algunas con mayor violencia que otras. Este resultado no es raro, sino que es geométricamente esperable, ya que si solo hay un atractor, el campo vectorial será radial y las trayectorias rectas. Dado esto, es imposible que haya una línea de campo que comience en el hemisferio inferior que llegue al origen desde arriba, todos deben atravesar el plano horizontal en algún punto, ya sea cerca o lejos. Por tanto, lejos de reprobar el algoritmo de control, esta condición simplemente indica que, al igual que sucede con el giro inicial, se requiere que el robot asuma una postura inicial antes de desplazarse. Nuevamente, esto se asemeja al comportamiento humano, el cual tiene la tendencia a plegar los codos, subiendo las manos y acercándolas al cuerpo, a la hora de girar. Conductualmente esto tiene sentido, puesto que reduce el riesgo de colisionar, reduce la energía la inercia rotacional y ofrece un punto de partida que resulta más cómodo para la siguiente tarea de manipulación.

5.12.2. Correlación entre parámetros

Los resultados del otro bloque de pruebas, donde la postura inicial es la que el ARCOS-Bot tiene por defecto, son positivos. De las 90 combinaciones, 71 lograron el objetivo (un 79 %).
Con el fin de poder analizar mejor el peso de cada una de las características se agruparon los datos en un diagrama de correlación (figura 5.15), donde se muestra con un 1 aquellos ensayos exitosos, con cero los fallidos y con -1 donde el efector final colisionó con la mesa (ninguno, por la razón análoga a la que llevó a que todo el primer bloque fallara). Se acomodaron los

parámetros de manera que verticalmente estén aquellos qué tienen que ver con el desplazamiento de robot (la distancia) y con la configuración del brazo (la *pose* final). Horizontalmente se acomodan los que definen el modelo de la mesa, es decir, la inclinación y la distancia del borde al objeto.
Además en ambos ejes se agruparon de forma que el parámetros que se considera más importante se ubicaran en el exterior, de manera que los colores reflejen su efecto. Las sumatorias en los extremos derecho e inferior permiten ver cuáles condiciones tienen un peso mayor en la efectividad del sistema.

Orientación de la mesa y *pose* del objetivo

Sin lugar a dudas son aquellos que están relacionados con la mesa los que tienen un efecto mayor. La orientación de la mesa tiene el efecto más consistente (tiene consecuencias a lo largo de todos los otros parámetros) con cambios de hasta un 50 %. Es interesante notar que cada una de las orientaciones para la mano tiene afectaciones distintas según se rota la mesa. La forma llamada *tomar* se ve casi que igualmente perjudicada por ángulos grandes tanto a izquierda como a derecha, la forma *botón* es la más sensible a ángulos negativos, y la forma *asa* es más bien perjudicada por ángulos positivos. La razón detrás de esto se discute brevemente en la sección 5.9: el robot gira y se desplaza distinto con tan solo cambiar la orientación de su efector final, aún cuando todas las demás condiciones permanezcan. Hay dos aspectos muy importantes detrás:

- El robot no solo está modelado como una cadena cinemática, sino que físicamente lo es. Por tanto, factores que afecten uno de sus eslabones, limitarán la capacidad de todo el conjunto.
- La capacidad de los últimos eslabones del brazo para lograr orientaciones no es tan alta. Esto es cierto incluso para la muñeca humana. Ciertas orientaciones implicarán el uso de todo el brazo, y algunas otras de todo el cuerpo. Justo ahí es donde reside una de las ventajas de los manipuladores móviles.

En este caso, el ángulo con el que se encuentra la mesa va a determinar qué tan antes se desactive el uso de la base (por su cercanía con el borde), lo que consecuentemente obliga el brazo a estirarse más, y termina comprometiendo la orientación de su efector final. Esto se puede verificar mediante la figura 5.16, donde se comparan las gráficas de desviación de un mismo escenario, una para 0° y para -45°. Aquí las líneas de desviación del brazo y del robot se mantienen paralelas en el primer caso, pero divergen en el segundo caso, esto quiere decir que cuando la mesa esta perpendicular a la trayectoria, el robot no tiene necesidad de rotar más allá de los usual para alcanzar el objetivo; pero cuando está inclinada sí. La figura 5.17 comprueba que esto, con valores de uso de la rotación y del brazo mucho mayores a partir del segundo seis (punto donde el robot con la mesa en -45° ya no puede usar más su desplazamiento).
Todo esto denota una debilidad del sistema para aquellos casos donde se requiere que la base sea capaz de desplazarse a lo largo del borde de la mesa.

		Asa			Botón			Tomar				
		1.5	2	4	1.5	2	4	1.5	2	4		
-45°	15 cm	1	1	1	1	1	0	1	1	1	8	9
	45 cm	0	0	1	0	0	0	0	0	0	1	
-30°	15 cm	1	1	1	1	1	1	1	1	1	9	17
	45 cm	1	1	1	1	1	0	1	1	1	8	
0°	15 cm	1	1	1	1	1	1	1	1	1	9	18
	45 cm	1	1	1	1	1	1	1	1	1	9	
30°	15 cm	1	1	1	1	1	1	1	1	1	9	18
	45 cm	1	1	1	1	1	1	1	1	1	9	
45°	15 cm	1	1	0	1	1	1	1	1	1	8	10
	45 cm	0	0	0	0	0	1	0	0	1	2	
		8	8	8	8	8	7	8	8	9		
		24			23			25				

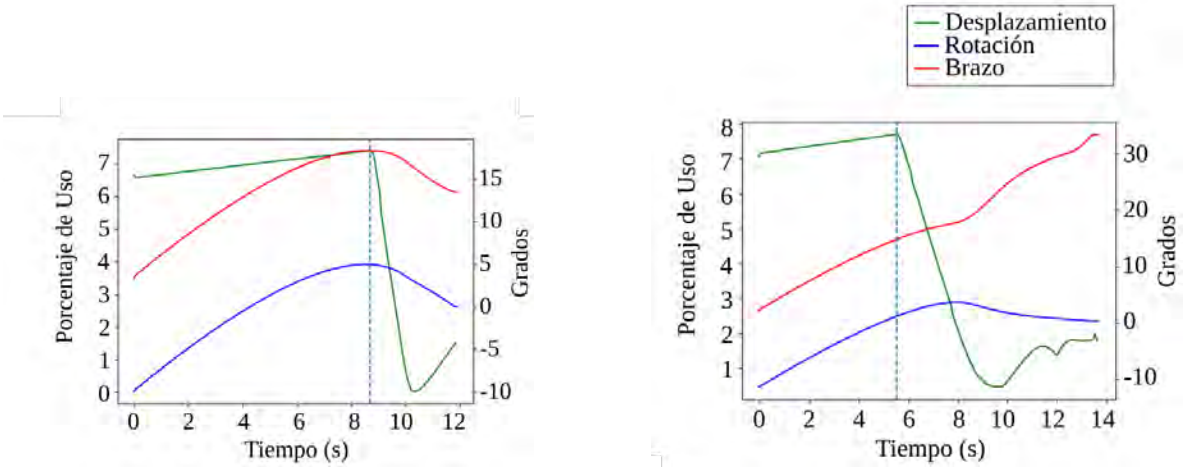
(a) Color agrupado en parámetros

		Asa			Botón			Tomar				
		1.5	2	4	1.5	2	4	1.5	2	4		
-45°	15 cm	1	1	1	1	1	0	1	1	1	8	9
	45 cm	0	0	1	0	0	0	0	0	0	1	
-30°	15 cm	1	1	1	1	1	1	1	1	1	9	17
	45 cm	1	1	1	1	1	0	1	1	1	8	
0°	15 cm	1	1	1	1	1	1	1	1	1	9	18
	45 cm	1	1	1	1	1	1	1	1	1	9	
30°	15 cm	1	1	1	1	1	1	1	1	1	9	18
	45 cm	1	1	1	1	1	1	1	1	1	9	
45°	15 cm	1	1	0	1	1	1	1	1	1	8	10
	45 cm	0	0	0	0	0	1	0	0	1	2	
		8	8	8	8	8	7	8	8	9		
		24			23			25				

(b) Color para caso individual

Figura 5.15: Diagramas de correlación entre los parámetros de las pruebas. Un 1 indica éxito y un 0 fallo

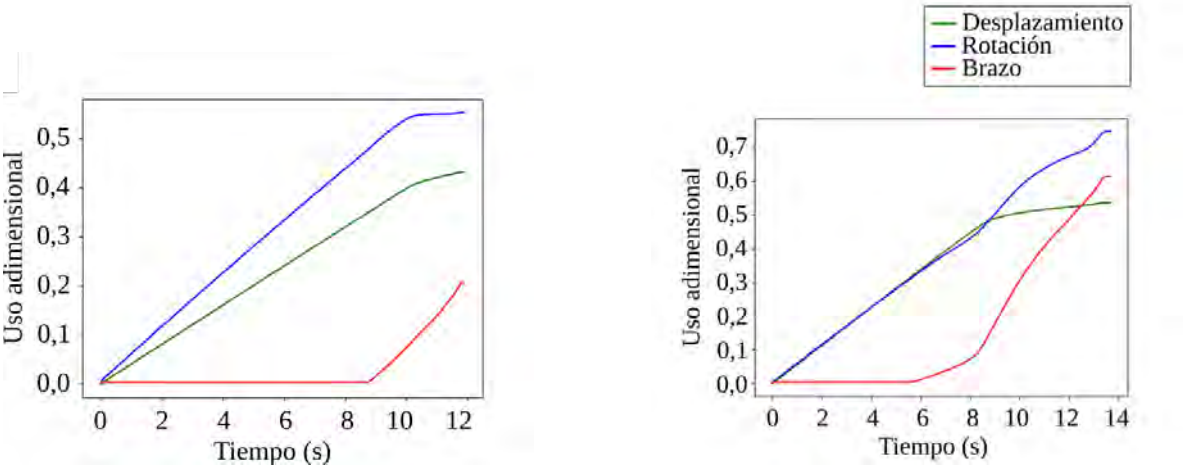
Fuente: Elaboración propia



(a) 0° (éxito) (b) -45° (fallo)

Figura 5.16: Gráficas de desviación para una *pose* final tipo *asa*, con el objetivo a 15 cm del borde de la mesa y a cuatro metros de distancia.

Fuente: Elaboración propia



(a) 0° (éxito) (b) -45° (fallo)

Figura 5.17: Gráficas de uso de articulaciones para una *pose* final tipo *asa*, con el objetivo a 15 cm del borde de la mesa y a cuatro metros de distancia.

Fuente: Fuente: Elaboración propia

Ubicación del objetivo sobre la mesa

La distancia al borde es el parámetro que tiene el efecto más rotundo, pero solo en ciertas condiciones. Para las 3 inclinaciones internas menores solo cambia el resultado de un 2 % de los casos, pero en los 45 y -45° es el responsable de una diferencia casi absoluta. Aparte de este hecho, hay que notar que si se ignora este parámetro (colocando todos los objetos a 15 cm del borde) solo habrían dos fallos, tal como se puede ver en la figura 5.15b. Se puede decir entonces, que es la combinación del ángulo de la mesa y la profundidad del objeto, el factor que en mayor medida define el éxito del sistema.

5.12.3. Lejanía de la mesa

Este factor registra un único efecto notable -en los 4 m- pero no es consistente con respecto a los demás parámetros. En la figura 5.15b se observa que para algunas *poses* finales, según la posición del objeto sobre la mesa, resulta beneficioso o perjudicial. Nuevamente el efecto ocurre solo para los ángulos de mesa mayores (con una sola excepción para *botón* en -30°). La principal forma en la que la lejanía de la mesa puede afectar es en cómo el robot navega a lo largo del campo vectorial. Esto posiblemente incide en cuál termina siendo el punto donde la base se topa con la mesa, y por tanto, en cuánto trabajo recae en el brazo.

5.12.4. Condiciones óptimas de trabajo

Con el fin de determinar la causalidad de los fallos se ejecutan nuevas corridas donde se modifican los dos parámetros más decisivos (ángulo de la mesa y ubicación del objetivo dentro de la mesa) de forma individual. Los resultados se ven en las figuras 5.18, donde se cambia la distancia poco a poco dejando la inclinación en 45° y -45°; y en 5.19, donde se hizo el caso análogo: para la misma distancia de 45 cm, se disminuyeron paulatinamente los grados de la mesa. Al proceder de esta forma también se quiere ver la sensibilidad del sistema ante dichos cambios y encontrar un rango de configuraciones donde el sistema actúe idealmente.

A continuación se describen los aspectos más importantes que se obtienen de analizar estas simulaciones:

- Hay una tendencia a la mejora muy clara en ambas pruebas, lo que indica que los parámetros guardan una estrecha correlación. Si uno de ellos hubiera estado subordinado al otro, se habrían observado cambios únicamente en uno de los dos grupos de datos.
- La distancia al borde de la mesa muestra irregularidad en las *poses* de *botón* y *tomar*, es decir, en ocasiones aumentan los fallos aunque se disminuya el parámetro. Es interesante recordar que este factor prácticamente no mostró efecto en otras inclinaciones menores a 45° y -45°. Parece ser que esta combinación de factores no solo limita al robot, sino que lo lleva a un estado inestable, donde es difícil predecir el resultado.
- La influencia de estos parámetros es muy distinta según cuál sea la *pose* final con la que se esté trabajando. La forma *asa* es la más delicada de todas, la única que sufre con los ángulos positivos y se ve muy perjudicada sea que el ángulo es cercano a 45°, o con que la distancia al borde sea cercana a los 45 cm. Es decir, la causa del problema el la *pose*.
- Para ambas pruebas se ve la poca interdependencia de los parámetros con la lejanía de la mesa.

		Asa			Botón			Tomar				
		1.5	2	4	1.5	2	4	1.5	2	4		
-45°	45 cm	0	1	1	1	0	0	1	1	0	5	27
	40 cm	0	1	1	0	0	0	0	1	0	3	
	38 cm	0	1	1	1	0	0	1	1	0	5	
	35 cm	0	1	1	1	1	0	1	1	0	6	
	33 cm	1	1	1	1	1	0	1	1	1	8	
45°	45 cm	0	0	0	1	1	1	1	1	1	6	26
	40 cm	0	0	0	0	1	1	0	0	1	3	
	38 cm	0	0	0	1	1	1	1	0	1	5	
	35 cm	0	0	0	1	1	1	1	1	1	6	
	33 cm	0	0	0	1	1	1	1	1	1	6	
		1	5	5	5	3	1	5	6	2		
		11			9			13				

(a) Color agrupado en parámetros

		Asa			Botón			Tomar				
		1.5	2	4	1.5	2	4	1.5	2	4		
-45°	45 cm	0	1	1	1	0	0	1	1	0	5	27
	40 cm	0	1	1	0	0	0	0	1	0	3	
	38 cm	0	1	1	1	0	0	1	1	0	5	
	35 cm	0	1	1	1	1	0	1	1	0	6	
	33 cm	1	1	1	1	1	0	1	1	1	8	
45°	45 cm	0	0	0	1	1	1	1	1	1	6	26
	40 cm	0	0	0	0	1	1	0	0	1	3	
	38 cm	0	0	0	1	1	1	1	0	1	5	
	35 cm	0	0	0	1	1	1	1	1	1	6	
	33 cm	0	0	0	1	1	1	1	1	1	6	
		1	5	5	5	3	1	5	6	2		
		11			9			13				

(b) Color para caso individual

Figura 5.18: Diagramas de correlación en pruebas donde se modifica únicamente la distancia entre el objetivo y el borde de la mesa. Un 1 indica éxito y un 0 fallo

Fuente: Elaboración propia

		Asa			Botón			Tomar			
		1.5	2	4	1.5	2	4	1.5	2	4	
-45°	45 cm	0	1	1	1	0	0	1	1	0	5
-40°	45 cm	0	1	1	1	0	0	1	1	0	5
-35°	45 cm	1	1	1	1	1	0	1	1	1	8
-30°	45 cm	1	1	1	1	1	0	1	1	1	8
30°	45 cm	1	1	1	1	1	1	1	1	1	9
35°	45 cm	0	0	0	1	1	1	1	1	1	6
40°	45 cm	0	0	0	1	1	1	1	1	1	6
45°	45 cm	0	0	0	1	1	1	1	1	1	6
		3	5	5	7	5	3	7	7	5	
		13			15			19			

Figura 5.19: Diagramas de correlación en pruebas donde se modifica únicamente la el ángulo de inclinación de la mesa. Un 1 indica éxito y un 0 fallo

Fuente: Elaboración propia

- Los valores a partir de los cuales se ve mayor mejoría son 38 y 35°/-35°, por lo que se les usa como parámetros para una nueva prueba.

La figura 5.20 muestra los resultados al fijar los parámetros relacionados con la mesa en los valores óptimos. El resultado es sumamente positivo (un 93 % de éxito), aunque sigue marcando errores, todos ellos para 4 m, los cuales demuestran una alta correlación entre la distancia de la mesa con el tipo de *pose* final que se solicite.

		Asa			Botón			Tomar				
		1.5	2	4	1.5	2	4	1.5	2	4		
-35°	38 cm	1	1	1	1	1	0	1	1	1	8	16
	35 cm	1	1	1	1	1	0	1	1	1		
	33 cm	1	1	1	1	1	0	1	1	1	8	
-30°	38 cm	1	1	1	1	1	0	1	1	1	8	17
	35 cm	1	1	1	1	1	1	1	1	1		
	33 cm	1	1	1	1	1	1	1	1	1	9	
30°	38 cm	1	1	0	1	1	1	1	1	1	8	17
	35 cm	1	1	1	1	1	1	1	1	1		
	33 cm	1	1	1	1	1	1	1	1	1	9	
35°	38 cm	1	1	0	1	1	1	1	1	1	8	16
	35 cm	1	1	0	1	1	1	1	1	1		
	33 cm	1	1	0	1	1	1	1	1	1	8	
		12	12	8	12	12	8	12	12	12		
		32			32			36				

Figura 5.20: Resultado de las pruebas con los valores óptimos

Fuente: Elaboración propia

Capítulo 6

Conclusiones y recomendaciones

6.1. Conclusiones

- En cuanto a los métodos que se utilizan para el abordaje de la cinemática de cuerpo completo de los robots humanoides, se observa que la literatura expone estrategias muy diversas con el fin de favorecer ciertos factores sobre otros. El simulador del robot humanoide del ARCOS-Lab se presta para ser modificado para adoptar la mayoría de estas propuestas.
- Las técnicas para la evasión de obstáculos utilizadas en la robótica se enfocan en su mayoría en la búsqueda de rutas óptimas, el método implementado en el ARCOS-bot no sigue esa tendencia, lo que lo hace más sencillo de aplicar a cadenas cinemáticas largas. El proyecto presenta un sistema nuevo que lo complementa.
- El simulador del robot humanoide del ARCOS-Lab no incluye la base omnidireccional ni su torso, por tanto, el robot carece de capacidad para ejercer tareas de manipulación móvil. El desarrollo de esta investigación logra consumar este ampliación de capacidades, abriendo la posibilidad de desarrollar trabajos más complejos.
- Este trabajo hace un significativo aporte en el sistema de control del simulador del robot humanoide, al darle la capacidad de realizar movimientos que implican el uso del cuerpo completo, de forma que es capaz de llegar a una posición y orientación espacial solicitada, según se lo permita su geometría.
- El control de cuerpo completo desarrollado se integró con el método de evasión de obtáculos de forma que el sistema es capaz de alcanzar una amplio rango de posiciones y orientaciones sobre una superficie plana, sin colisionar con ella. Su efectividad se comprobó mediante pruebas simuladas.
- El comportamiento de la cinemática inversa de velocidades de una cadena redundante con articulaciones prismáticas va a depender de las velocidades que comanden al efector final. Para el algoritmo utilizado para este proyecto, cuando la velocidad y la articulación son paralelas, la solución va a limitarse a usar únicamente ese grado de libertad, ya resulta ser la respuesta más sencilla de todas.
- Limitar el uso de ciertas articulaciones mediante la modificación de la matriz de pesos no implica que las demás articulaciones vayan utilizarse de la forma más conveniente. Esto

es debido a que el algoritmo encargado da una solución tomando en cuenta únicamente el estado actual del sistema. Cualquier requerimiento que vaya más allá de esto debe ser parte activa del algoritmo que define los pesos. Este el caso sucedido con la rotación de la base, en la sección 4.2.3.

- El sistema de coordinación de cuerpo completo utilizado tiene la desventaja de que el movimiento de la base está completamente subordinado a la tarea que está desempeñando el efector final, y por tanto siempre va a seguir la dirección que esta tome. La única posibilidad que se tiene es determinar qué tanto se quiere que se considere su uso para lograr el movimiento de la mano. Por esta razón, si se quiere lograr una tarea cinemática paralela que requiera un comportamiento distinto, se requerirá el uso de una herramienta adicional, como velocidades en el espacio nulo.
- El sistema de control diseñado logra unificar el cuerpo completo para lograr movimientos coordinados y naturales. Además tiene la ventaja de que por sí solo saca provecho de su geometría para desempeñar la tarea que se le solicita, esto es evidente con la tendencia registrada del robot de girar su cuerpo para llegar a objetos que están muy dentro en la mesa. Un comportamiento de este tipo, para controles desacoplados, requeriría ser reproducido sintéticamente.
- La inclinación de la mesa es el factor más influyente para la efectividad del sistema de control diseñado. A partir de los valores de 45° y -45°, el sistema no puede desenvolverse correctamente, pero en el rango dentro de estos, su comportamiento es óptimo, con un porcentaje de éxito del 96 %.

6.2. Recomendaciones

- Al evaluar la solución de la cinemática inversa de velocidades de un sistema redundante es recomendable someter el sistema a trayectorias de velocidad o a campos vectoriales (o lo más similar al caso real) y no solo a una velocidad fija, ya que el comportamiento puede diferir por ser no holonómico y llevar a conclusiones erradas.
- Realizar un estudio detallado del comportamiento del algoritmo de cinemática inversa con todos los grados de libertad es un paso fundamental para poder diseñar un sistema de control basado en pesos. Los escenarios de prueba deben elegirse cuidadosamente según la tarea que se quiere llevar a cabo con el robot.
- Algunas de las tareas realizadas por el módulo desarrollado se pueden adaptar de forma que se usen las funciones disponibles en la biblioteca de utilidades de ARCOSPYU y de PYROVITO. Este cambio ayudaría a mejorar la uniformidad de este nuevo código con el existente, además de que reduciría su volumen.
- Ante la imposibilidad del sistema de alejar la base de los obstáculos es recomendable hacer uso del control por proyecciones en el espacio nulo para agregar esta capacidad. Esto supondría un complemento ideal para el control de cuerpo completo, ya que ayudaría a encontrar la posición óptima para alcanzar el objeto, capacidad actualmente carente que evitaría la mayoría de los estados fallidos en las pruebas. Este cambio puede implementarse como un método que le introduzca velocidades paralelas al borde de la mesa.

- Que la Universidad de Costa Rica, especialmente las Escuelas de Ingeniería Eléctrica y Mecánica, sigan generando, por medio del ARCOS Lab, investigación científica a partir de los avances alcanzados con este proyecto. Esto con el firme propósito de lograr la máxima capacidad del robot y el desarrollo académico e investigativo de los estudiantes.
- Favorecer de nuevos recursos económicos, materiales y científicos que empoderen el proyecto macro del robot humanoide, a un nivel de alcance tal, que el laboratorio figure con líder regional -por medio de la Universidad de Costa Rica- con nuevos alcances del robot humanoide.
- Se puede incorporar más activamente, para el crecimiento del proyecto del robot humanoide, la participación planificada de estudiantes con propuestas de tesis de licenciatura en mecánica, física y eléctrica que logren avanzar sistemáticamente en la integración y mayor optimización del robot humanoide.

Bibliografía

[1] F. Ruiz-Ugalde, G. Cheng, and M. Beetz, "Prediction of action outcomes using an object model," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010, pp. 1708–1713.

[2] F. Ruiz-Ugalde, G. Cheng, and M. Beetz, "Fast adaptation for effect-aware pushing," in *IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 614–621.

[3] D. García-Vaglio and F. Ruiz-Ugalde, "An Object Manipulation System Architecture for Humanoid Robots Based on Primate Cognition," Tech. Rep., 2018.

[4] I. Chaves-Arbaiza, D. García-Vaglio, and F. Ruiz-Ugalde, "Smart placement of a two-arm assembly for an everyday object manipulation humanoid robot based on capability maps," Tech. Rep., 2018.

[5] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed., ser. Advanced Textbooks in Control and Signal Processing. Springer, 2010.

[6] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*, 2nd ed., ser. Springer Tracts in Advanced Robotics. Springer, 2014.

[7] O. Brock, J. Park, and M. Toussaint, "Mobility and manipulation," in *Springer Handbook of Robotics*, 2nd ed., B. Siciliano and O. Khatib, Eds. Springer, 2017.

[8] A. Dietrich, *Whole-Body Impedance Control of Wheeled Humanoid Robots*, 1st ed., ser. Springer Tracts in Advanced Robotics. Springer, 2016.

[9] B. Bäuml, F. Schmidt, T. Wimböck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. Grebenstein, O. Eiberger, and G. Hirzinger, "Catching flying balls and preparing coffee: Humanoid Rollin' Justin performs dynamic and sensitive tasks," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 3443–3444.

[10] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 529–536.

[11] F. Ruiz-Ugalde, "Compact Models of Objects for Skilled Manipulation," Dr. rer. nat. thesis, Technical University of Munich, april 2015.

[12] L. de Robótica Autónoma y Sistemas Cognitivos, "ARCOS-Lab Documents," 2020, unpublished. [Online]. Available: https://git.arcoslab.org/administrative/arcoslab-doc

[13] A. Jain and C. C. Kemp, "Pulling Open Doors and Drawers: Coordinating an Omnidirectional Base and a Compliant Arm with Equilibrium Point Control," Tech. Rep., 2010.

[14] N. Ramezani and M. A. Williams, "Smooth robot motion with an Optimal Redundancy Resolution for PR2 robot based on an analytic inverse kinematic solution," in *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-December. IEEE Computer Society, dec 2015, pp. 338–345.

[15] K. Lynch and F. Park, *Modern Robotics Mechanics, Planning, and Control*, 1st ed. Cambridge University Press, May 2017.

[16] J. Angeles, *Fundamentals of Robotic Mechanical Systems Theory, Methods, and Algorithms*, 4th ed., ser. Mechanical Engineering. Springer, 2014.

[17] M. Mihelj, T. Bajd, A. Ude, J. Lenarčič, A. Stanovnik, M. Munih, J. Rejc, and S. Šlajpah, *Robotics*, 2nd ed. Springer, 2019.

[18] D. García-Vaglio, "El Object Model System," may 2020, comunicación Personal.

[19] N. Hogan, "Impedance Control: An Approach to Manipulation," in *1984 American Control Conference - Conference Proceedings*, 1984, pp. 304–313.

[20] K. Gong and A. I. McInnes, "A hierarchical control scheme for coordinated motion of mobile manipulators," in *ICARA 2011 - Proceedings of the 5th International Conference on Automation, Robotics and Applications*, Wellington, New Zealand, 2011, pp. 115–120.

[21] S. Sharma and C. Scheurer, "Generalized unified closed form inverse kinematics for mobile manipulators with reusable redundancy parameters," in *Proceedings of the ASME Design Engineering Technical Conference*, vol. 5B-2017. American Society of Mechanical Engineers (ASME), nov 2017.

[22] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, vol. 1. Osaka, Japan, Japan: Publ by IEEE, 1996.

[23] K. Inoue, H. Yoshida, T. Arai, and Y. Mae, "Mobile manipulation of humanoids real-time control based on manipulability and stability," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, 2000, pp. 2217–2222.

[24] Y. Yamamoto and X. Yun, "Coordinating Locomotion and Manipulation of a Mobile Manipulator," *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1326–1332, 1994.

[25] H. Seraji, "An on-line approach to coordinated mobility and manipulation," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1. Atlanta, GA, USA, USA: Publ by IEEE, 1993, pp. 28–33.

[26] B. Bayle, J. Y. Fourquet, F. Lamiraux, and M. Renaud, "Kinematic control of wheeled mobile manipulators," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 2002, pp. 1572–1577.

[27] M. Li, Z. Yang, F. Zha, X. Wang, P. Wang, P. Li, Q. Ren, and F. Chen, "Design and analysis of a whole-body controller for a velocity controlled robot mobile manipulator," *Science China Information Sciences*, vol. 63, no. 7, p. 170204, jul 2020. [Online]. Available: http://link.springer.com/10.1007/s11432-019-2741-6

[28] M. Iskandar, G. Quere, A. Hagengruber, A. Dietrich, and J. Vogel, "Employing Whole-Body Control in Assistive Robotics," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., nov 2019, pp. 5643–5650.

[29] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-December. IEEE Computer Society, dec 2015, pp. 928–935.

[30] J. Minguez, F. Lamiraux, and J.-P. Laumond, "Motion planning and obstacle avoidance," in *Springer Handbook of Robotics*, 2nd ed., B. Siciliano and O. Khatib, Eds. Springer, 2017.

[31] A. H. Khan, S. Li, and X. Luo, "Obstacle Avoidance and Tracking Control of Redundant Robotic Manipulator: An RNN-Based Metaheuristic Approach," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4670–4680, jul 2020.

[32] G. Pajak and I. Pajak, "Planning of a point to point collision-free trajectory for mobile manipulators," in *2015 10th International Workshop on Robot Motion and Control, RoMoCo 2015*. Institute of Electrical and Electronics Engineers Inc., aug 2015, pp. 142–147.

[33] F. Ruiz-Ugalde, D. García-Vaglio, and J. Peralta. (2019, November) Tutorials: Object manipulation robot simulator, arcos-lab. Online. https://wiki.arcoslab.org/doku.php?id=tutorials:object_manipulation_robot_simulator.

[34] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," in *IEEE Journal of Robotics and Automation*, 2004.

[35] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, no. 2, pp. 47–53, 1969.

[36] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *Control Systems Technology, IEEE Transactions on*, vol. 2, pp. 123 – 134, 07 1994.

[37] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, pp. 93–101, 1986.

[38] O. Robotics. (2019) Kinematics and dynamics library, kdl. Online. http://docs.ros.org/en/indigo/api/orocos_kdl/html/index.html.

[39] D. Schinstock, T. Faddis, and B. Greenway, "Robust inverse kinematics using damped least squares with dynamic weighting," 04 1994.

[40] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 868 – 871, 1977.

[41] I. Lee, J. Oh, and H. Bae, "Constrained whole body motion planning in task configuration and time," *INTERNATIONAL JOURNAL OF PRECISION ENGINEERING AND MANUFACTURING*, vol. 19, pp. 1651–1658, 2018.

[42] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5313–5320.

[43] A. Jain and C. C. Kemp, "El-e: An assistive mobile manipulator that autonomously fetches objects from flat surfaces," *Autonomous Robots*, vol. 28, pp. 45–64, 1 2010. [Online]. Available: https://link-springer-com.ezproxy.sibdi.ucr.ac.cr/article/10.1007/s10514-009-9148-5

[44] L. Zhang, J. Pan, and D. Manocha, "Motion planning of human-like robots using constrained coordination," in *9th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS09*, 2009, pp. 188–195.

[45] O. Porges, R. Lampariello, J. Artigas, A. Wedler, C. Borst, and M. A. Roa, "Reachability and dexterity: Analysis and applications for space robotics," 05 2015.

[46] T. A. H. P. International, "International health facility guidelines: Part c ergonomics," no. 4, pp. 12–23, 2015. [Online]. Available: https://healthfacilityguidelines.com/ViewPDF/ViewIndexPDF/iHFG_part_c_ergonomics

Anexos

Anexo A.1.

```
arm_segments = [
        #Base_x
        Segment(Joint(Joint.TransX),
            Frame(Rotation.Identity(), Vector(0.0, 0.0, 0.0))),
        #Base_y
        Segment(Joint(Joint.TransY),
            Frame(Rotation.Identity(), Vector(0.0, 0.0, 0.0))),
        #Base_RZ
        Segment(Joint(Joint.RotZ),
            Frame(Rotation.Identity(), Vector(0.0, 0.0, 0.0))),
        #Base Static Transformation: from the center of the
           robot on the floor to the
        # torso's screw. At an initial height of 1.02 m from the
            floor
    Segment(Joint(Joint.None),
                        Frame(Rotation.Identity(), Vector(0.30,
                           0.0, 1.02) )),
        #Torso
        Segment(Joint(Joint.TransZ),
            Frame(Rotation.Identity(), Vector(0.0, 0.0, 0.0))),
        #Shoulders Static Transformation: from the torso's screw
            to the base of the arm
    Segment(Joint(Joint.None),
                Frame(Rotation.RotX(45.0*pi/180.0)*Rotation.RotZ
                   (30.0*pi/180.0),Vector(0.162, -0.1327, 0.0) )
                   ),
        #Arm Segments
    Segment(Joint(Joint.None),
            Frame(Rotation.Identity(), Vector(0.0, 0.0, 0.11))),
    Segment(Joint(Joint.RotZ),
            Frame(Rotation.RotX(-pi/2), Vector(0.0, 0.0, 0.20)))
               ,
    Segment(Joint(Joint.RotZ, -1),
            Frame(Rotation.RotX(pi/2), Vector(0.0, -0.20, 0.0)))
```

```
            ,
    Segment(Joint(Joint.RotZ),
            Frame(Rotation.RotX(pi/2), Vector(0, 0, .20))),
    Segment(Joint(Joint.RotZ, -1),
            Frame(Rotation.RotX(-pi/2), Vector(0, 0.2, 0))),
    Segment(Joint(Joint.RotZ),
            Frame(Rotation.RotX(-pi/2), Vector(0, 0, 0.19))),
    Segment(Joint(Joint.RotZ, -1),
            Frame(Rotation.RotX(pi/2), Vector(0, -0.078, 0.0))),
    Segment(Joint(Joint.RotZ),
        Frame(Rotation.RotZ((180.0+45)*pi/180.0)*Rotation.RotX(
           pi/2)*Rotation.RotY(pi), Vector(0.075, -0.075,
           -0.094))),
]
```