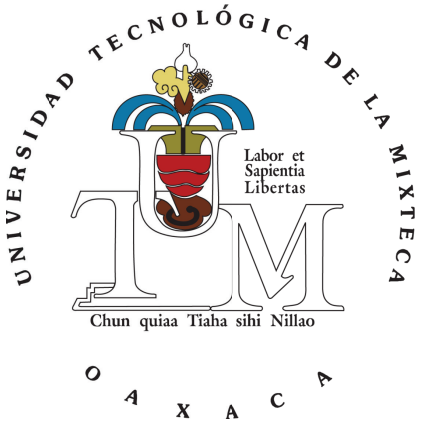

UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"CONTROL SERVOVISUAL DE UN ROBOT MANIPULADOR DE 3 GRADOS DE LIBERTAD"

TESIS:

PARA OBTENER EL GRADO DE
MAESTRA EN ROBÓTICA

PRESENTA:
ING. KARINA RAMÍREZ LÓPEZ

DIRECTOR DE TESIS:
DR. OSCAR DAVID RAMÍREZ CÁRDENAS

CODIRECTOR DE TESIS:
DR. JESÚS LINARES FLORES

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO. JUNIO DE 2026

Dedicatoria

*A **Dios**, por todas las oportunidades y las personas maravillosas que ha puesto en mi camino.*

*A mis padres, **Minerva** y **Máximo**, por ser para mí un ejemplo de esfuerzo, trabajo y fortaleza; por darme siempre todo cuanto estuvo a su alcance con amor y dedicación, y por nunca limitar mis sueños. Gracias por creer en mí en todo momento y por acompañarme incondicionalmente en cada etapa de mi vida.*

*A mis hermanas: **Rosario**, **Andrea** y **Sandra**, por estar siempre para mí, por cuidarme y consentirme cada vez que lo necesito. Los momentos con ustedes son un abrazo a mi corazón.*

*A mi hermano **Jesús**, por guiarme cuando todo se complicaba y por enseñarme que hay retos que puedo enfrentar sola.*

*A mi **Lalito**, por todo su amor, su compañía y su incondicional apoyo; por permanecer a mi lado incluso en los momentos más difíciles y por comprenderme siempre. Gracias por llenar mis días de felicidad y detalles, por cada recuerdo construido juntos y por hacer cada momento especial. Las palabras no son suficientes para expresar lo maravilloso que eres y lo importante que has sido en este camino.*

*A mis compañeros de la maestría que se convirtieron en amigos para toda la vida: **Exon**, **Osmar** y **Eicarl**. Por estar para mí y apoyarme en todo. Contar con su apoyo me ha llevado a este momento.*

*A **Mayra**, mi gran amiga y una persona a quien admiro profundamente por ser un ejemplo para mí. Gracias por llegar a mi vida en un momento tan complicado, por escucharme con paciencia, brindarme tu apoyo incondicional y procurar siempre mi bienestar. Tu amistad*

ha sido un gran apoyo a lo largo de este camino.

A ***Reyna*** *y* ***Jos****, gracias por su amistad, por acompañarme y apoyarme en los momentos en que más necesitaba compañía, y por regalarme sonrisas y momentos de alegría incluso en los días más grises.*

A ***Monse*** *y* ***Liz****, gracias por acompañarme en esas tardes interminables en los laboratorios. Su compañía hizo más ameno este camino.*

A todas las personas que me acompañaron en algún punto de este camino y me apoyaron para alcanzar esta meta.

Agradecimientos

Al **Dr. Oscar David Ramírez Cárdenas**, mi director de tesis, por guiarme a lo largo de este trayecto académico; por compartir generosamente sus conocimientos y experiencias, distinguiéndose siempre como un excelente profesor; y por cada uno de sus valiosos consejos. Asimismo, agradezco profundamente la confianza depositada en mí y su constante apoyo durante el desarrollo de este trabajo.

Al **Dr. Jesús Linares Flores**, por su invaluable asesoría como codirector de este trabajo, así como por la atención y dedicación brindadas para lograr su culminación.

A mis asesores de tesis: **Dr. Hugo Ramírez Leyva, Dr. Anibal Arias Aguilar, Dr. Carlos García Rodríguez y Dr. Eduardo Sánchez Soto** por sus valiosas sugerencias, observaciones y aportaciones académicas, las cuales enriquecieron significativamente el desarrollo y la calidad de este trabajo.

A la señora **Celes**, a la **Lic. Carmen** y al **técnico Omar**, por desempeñar sus funciones con eficiencia, dedicación y compromiso, contribuyendo de manera significativa para que pudiera culminar este trabajo de tesis.

A la **Universidad Tecnológica de la Mixteca**, por facilitar sus instalaciones y recursos académicos para el desarrollo de esta investigación, especialmente al Laboratorio de Posgrado, por brindar el espacio y las condiciones necesarias para la realización de este trabajo.

A la **Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI)**, por el apoyo brindado a través de la beca otorgada, la cual hizo posible el desarrollo de este trabajo de investigación.

Resumen

El presente trabajo aborda la implementación de la técnica de control servovisual en un entorno de simulación, empleando un robot manipulador de tres grados de libertad con configuración cámara en mano, cuyo diseño tridimensional fue desarrollado en la Universidad Tecnológica de la Mixteca (UTM). En capítulos posteriores se describen el diseño 3D del robot, así como sus modelos cinemático y dinámico. Posteriormente, se expone el modelo matemático obtenido para el control servovisual y se presentan los resultados alcanzados en la simulación. En particular, se realiza una comparación entre la aplicación del control servovisual en combinación con un controlador PID convencional y su implementación junto con un controlador PID+G. También se explora la integración del control servovisual en conjunto con un control por modos deslizantes integral (ISMC). Finalmente, se analiza el error de seguimiento obtenido al integrar el control servovisual basado en imagen con las 3 variantes del controlador, observando un menor error cuando se emplea el controlador PID+G.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las aplicaciones de control servovisual para robots manipuladores, han aumentado en años recientes, incluyendo el uso de entornos 3D para su implementación. Estos entornos ofrecen las herramientas y sensores más comunes en el mercado [5]; enfocando la mayoría de trabajos en robots industriales de vanguardia, cuyas restricciones son distintas a las que se presentan en prototipos de bajo presupuesto. Ante la falta de herramientas de este tipo en el área de robótica de la UTM, se busca diseñar e implementar el control servovisual para un robot manipulador de 3 grados de libertad (GDL) en el simulador Matlab-Coppelia Sim, en específico, de un modelo que se diseñará en la institución antes mencionada. Lo anterior, tiene como objetivo que el alumnado de carreras afines a la robótica pueda llevar a cabo simulaciones con el software diseñado y contribuir a una mejor comprensión de la teoría de control de robots manipuladores así como ampliar la investigación en esta área de la robótica.

Al desarrollar la técnica de control se busca reducir el error de posición combinando el control servovisual basado en imagen con un controlador PID, en comparación con los resultados obtenidos al utilizar un controlador PID+G, las ventajas de las técnicas propuestas se enuncian en el marco teórico del presente trabajo. Se muestra el desempeño de ambas técnicas de control, mediante simulaciones realizadas utilizando la herramienta Coppelia Sim.

1.1. Estado del arte

El aumento de aplicaciones para las cuales se hace uso de los robots manipuladores, requiere de la mayor cantidad posible de datos sobre el área de trabajo en la cual se desempeñan, ya que tanto el entorno como las tareas a realizar son cada vez más complejos. Teniendo en cuenta que la finalidad de la automatización de procesos es realizarlos rápidamente o minimizar el número de errores cometidos durante tales actividades, diversos investigadores [6], [7], [8], apuestan por la retroalimentación al sistema con información relevante de su entorno para cumplir con dicha finalidad.

Por ejemplo, en 1971 en el trabajo presentado por Inoue y Shirai [6], se propone el uso de una posición actual además de una posición deseada para lograr la retroalimentación. Lo anterior lo consiguen incluyendo un sensor de visión en el sistema robótico, de modo que, a través de tal sensor se obtienen datos sobre la posición del robot en distintos instantes de tiempo hasta alcanzar la posición deseada en algún momento.

En los últimos 50 años el uso de sensores de visión controlar el movimiento de un robot se ha convertido en una técnica cada vez más común. Uno de los primeros trabajos donde se menciona la retroalimentación visual para la guía de robots es el realizado en 1971 por Inoue [6]. El término control servovisual, el cual es el tema de interés de este proyecto de investigación, fue usado por primera vez en 1979, para guiar el movimiento de un robot en tiempo real [7].

Conforme han transcurrido los años, la complejidad de las tareas se ha incrementado, considerando no solamente el robot manipulador y su movimiento, sino agregando también un objeto en movimiento, el cual tiene que ser seguido y empuñado como se muestra en el trabajo de N. Houshangi [8] presentado en 1990.

Desde que se acuñó el término control servovisual, se ha desarrollado una serie de análisis sobre los problemas presentes en aplicaciones en las cuales se implementa dicha técnica, además se plantean posibles soluciones a cuestiones referentes al sensado, errores en la etapa de control y consideraciones extras a tenerse en cuenta para robots de distintas configuraciones, como se indica en el trabajo de 1996 publicado por Smith y Papanikolopoulos [9].

Según el trabajo de Şahin y Zergeroğlu de 2005 [10], es importante considerar las in-

certidumbres que se presentan durante el control de movimiento de un robot, ya sean las relacionadas con la dinámica del sistema o con los parámetros intrínsecos de las cámaras a utilizar para la obtención de la información visual. Una vez que se han realizado las consideraciones necesarias, será posible seleccionar la técnica de control adecuada para compensar dichas incertidumbres. Además, es fundamental considerar que existen dos enfoques mediante los cuales es factible obtener retroalimentación visual para un robot; control por cámara en mano y control por cámara fija. Este trabajo se enfocará en el control por cámara en mano debido a las ventajas que presenta.

En el artículo de 2008 elaborado por Wang et al. [11], se indican las ventajas de utilizar la configuración cámara en mano en el control en lazo cerrado. Entre los atributos de la configuración ya mencionada, se señalan el ahorro de tiempo y recursos que serían necesarios para calibrar el sensor de visión en comparación con los requeridos si se usara una cámara fija, también se hace referencia a la flexibilidad extra de la cual se dota al manipulador, lo cual le permite tener una vista más detallada de su espacio de trabajo.

En la Figura 1.1 se muestra el esquema de la configuración cámara en mano que se usará en el resto del trabajo. Se ejemplifica una aplicación para la inserción de piezas en una superficie con una perforación que está orientada en posiciones aleatorias a las cuales se tendrá que ajustar el manipulador.

A pesar de las constantes mejoras que se han realizado para optimizar el desempeño de los robots manipuladores guiados por control servovisual, la inclusión de determinadas consideraciones en el sistema para optimizar ciertas características generalmente repercute agregando deficiencias en otros aspectos. Por lo anterior, las contribuciones a la teoría de control son cada vez mayores, es así como Keshmiri y Xie [12], presentaron una técnica denominada planeación de trayectoria optimizada, en la cual se toman en cuenta las restricciones del sistema para evitar ir más allá de sus límites y quedar atrapado en una posición sin retorno. Esta técnica mostrada en [12], se basa en la combinación de 4 etapas para completar la propuesta de control. La primera etapa consiste en el cálculo de profundidad inicial, a continuación se realiza la planeación de trayectoria fuera de línea, posteriormente se ejecuta la trayectoria calculada y finalmente se pasa de la planeación de trayectoria a un bloque de control servovisual con el cual se compensa cualquier error obtenido durante el cálculo de trayectorias.

Según el artículo de Bae et al. [13] , de acuerdo con la configuración del lazo de retro-

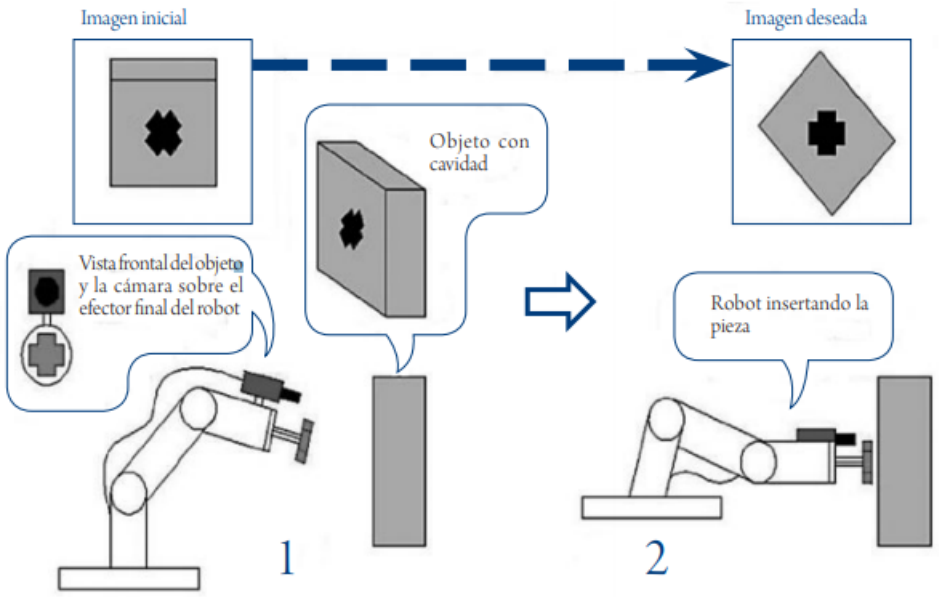

Figura 1.1: Configuración robot con cámara en mano. Tomado de [1].

alimentación de control, existen 2 tipos de sistemas de control servovisual; el directo y el indirecto. El sistema directo calcula directamente las entradas del sistema dinámico y requiere altas frecuencias de muestreo y puede ser difícil de implementar cuando el manipulador se mueve muy rápido. El sistema indirecto, en cambio, utiliza 2 lazos de control y es apropiado para casos donde la frecuencia de muestreo es baja. Además, se menciona la diferencia entre un sistema de control servovisual basado en posición y el basado en imagen. El método basado en imagen localiza y sigue una característica específica de la imagen y busca reducir la distancia entre dicho punto en la imagen y el objeto al cual se busca llegar. En el método basado en posición se hallan más complicaciones puesto que se requiere conocer el modelo geométrico de la cámara, ya que los parámetros de posición y seguimiento están definidos en 3 dimensiones. Por las ventajas que se han descrito en [13], para dicho trabajo se implementa un sistema indirecto de control servovisual con cámara en mano, mediante un lazo externo de control servovisual basado en imagen para calcular la velocidad objetivo del efector final y un lazo interno para control de posición de las articulaciones usando backstepping adaptativo más un controlador proporcional integral derivativo.

Ya que en el mundo real no siempre es fácil modelar un sistema en su totalidad, es necesario tener en cuenta técnicas que permitan controlar un sistema ante incertidumbres presentes

en el mismo. Por ejemplo en 2019 Nourisola et al. [14] proponen un esquema híbrido para mejorar el rendimiento del control servovisual basado en imagen mediante la combinación de un controlador proporcional derivativo y un controlador por modos deslizantes. Con la técnica propuesta se obtienen movimientos más rápidos, precisos y robustos ante ruidos e incertidumbres que los resultantes de usar las técnicas convencionales por separado. Las características que se comparan son el error de posición, la velocidad en las articulaciones y la suavidad de la trayectoria.

Analizando algunos de los trabajos relacionados con la presente investigación y que se han desarrollado en el ámbito local, Pacheco et al. [15] muestran el desarrollo de una plataforma experimental para propósitos de investigación y enseñanza, en el área de control de sistemas de transmisión directa de un grado de libertad.

Otro de los trabajos a destacar se refiere al publicado en 2017 donde Bedolla et al. [16] presentan un sistema hardware in the loop para emular un robot manipulador de 2 grados de libertad, utilizando un controlador digital de señales para resolver el modelo dinámico del sistema en tiempo real, además de que se muestran los movimientos del modelo 3D del robot en el ambiente virtual de LabVIEW®.

Por otra parte Aragón et al. [17] presentan la implementación en hardware in the loop del sistema carro- péndulo, cuya validación se realiza a través de Matlab®-Simulink®. Además es importante notar el desarrollo del modelo en el entorno virtual Unity3D®, en el cual se observan los movimientos del carro-péndulo y mediante el cual se dota de un mayor realismo al trabajo.

Asimismo se tiene el proyecto desarrollado por Ambrocio et al. [18], en el cual se presenta el desarrollo de un algoritmo de marcha para el robot Nao usando el método del punto de momento cero, donde se pretende mejorar la estabilidad de la marcha del mismo sobre pasto artificial. El algoritmo obtenido se valida con simulaciones en el entorno virtual de Webots®y en el propio robot.

1.2. Planteamiento del problema

Dado que la lista de procesos cuya automatización es realizada a través de la incorporación de robots manipuladores es cada vez más extensa, resulta importante dotar a estos

dispositivos de la mayor cantidad posible de datos sobre el área en la cual se desempeñan, para asegurar su correcto funcionamiento. Puesto que tanto el espacio de trabajo como las tareas a realizar por robots manipuladores son cada vez más complejos, debe tenerse en cuenta que a través de la automatización de procesos ha sido posible realizar tareas rápidamente o minimizar el número de errores cometidos durante tales actividades siempre que se le proporcione al sistema la información adecuada [19].

Aunque las técnicas existentes, con las que se controlan los sistemas robóticos han evolucionado con el paso del tiempo y los resultados obtenidos han ido mejorando poco a poco, todavía existen problemas relacionados con la información visual que puede captar un robot, esto debido a las restricciones referentes al presupuesto con el que se cuenta para completar la construcción del robot manipulador además del equipamiento del mismo con los sensores de visión. Aunado a lo anterior, la situación económica actual hace aún más difícil la adquisición de equipos para fines académicos y construir algún prototipo desde cero muchas veces tampoco es una tarea que resulte asequible sobre todo en zonas en desarrollo.

Una forma de contar con herramientas que faciliten la experimentación en estas zonas, es a través de simulaciones utilizando entornos 3D, que permitan a la comunidad científica el poder realizar prácticas adecuadas.

1.3. Justificación

Como lo menciona Aldana [20], no se puede negar que el desarrollo económico y social de los pueblos está ligado con su desarrollo científico y tecnológico. En México es difícil lograr un progreso dentro de la ciencia y la tecnología, puesto que no existe la inversión necesaria para apoyar a la comunidad científica en sus procesos de investigación. En zonas de bajos recursos el investigador tiene que hallar la manera de llevar a cabo sus proyectos utilizando los pocos recursos que tiene al alcance.

De acuerdo con Maxinez et al. [21] es necesario desarrollar la tecnología en lugares como México, para entender la basta colección de aplicaciones que se desarrollan a nivel mundial y no solo en el aspecto académico sino también en el empresarial, para entonces tener la posibilidad de competir con tecnología propia en el mercado internacional y no simplemente depender del extranjero para obtener tecnología de punta. Conseguir personal calificado

para desarrollar y manipular tecnología reciente, requiere de una educación integral donde se conjunte teoría y práctica de calidad, suele suceder que se cuenta con la teoría, sin embargo, el equipo necesario para la práctica tiene un alto costo.

Considerando los factores anteriores, es importante desarrollar modelos en entornos 3D que capaciten al estudiante de forma adecuada para comprender la tecnología actual.

Aunque en México existen avances relacionados con el control servovisual aplicado a los robots manipuladores, en la UTM aún no se desarrollan proyectos de dicha índole, por lo cual, llevar a cabo un trabajo de investigación referente al desarrollo de un simulador para el control servovisual de un robot manipulador en esta institución, contribuiría de buena manera al área de investigación de la robótica. Asimismo, les permitiría a los estudiantes de licenciatura experimentar con un software funcional que se asemeje a la realidad y así lograr una mejor comprensión de la teoría de control y robótica de manipuladores abordada en sus cursos académicos.

Cabe señalar que se busca desarrollar un simulador 3D que integra el diseño 3D de un robot manipulador de 3 GDL, utilizando el entorno Matlab-CoppeliaSim ya que debido a la situación académica y económica actual, es necesario desarrollar herramientas que faciliten la asimilación de conocimientos y el proceso de investigación en el área de la robótica, sin la necesidad de adquirir costosos equipos además, el uso tanto de Matlab como CoppeliaSim se ha extendido ampliamente dentro de la comunidad perteneciente al área de la robótica, por lo cual existe un gran acervo bibliográfico que facilita la comprensión del funcionamiento del entorno, además de que posibilita la manipulación del mismo sin mayores complicaciones. Asimismo, se pretende sentar las bases para que en trabajos futuros sea posible construir un prototipo basado en el diseño desarrollado en la presente investigación y que de este modo sea posible contar tanto con el simulador como con el prototipo del robot manipulador en la institución, los cuales servirán como herramientas para desarrollar prácticas adecuadas.

1.4. Hipótesis

"Será posible disminuir el error de posición de un robot manipulador de 3 grados de libertad, utilizando el control PID+G en comparación con el error obtenido utilizando un PID clásico".

1.5. Objetivos

1.5.1. Objetivo general

Diseñar un robot manipulador de 3 GDL e implementar la técnica de control servovisual utilizando un simulador 3D.

1.5.2. Objetivos específicos

- Diseñar un robot manipulador en 3D en SolidWorks®.
- Simular el control de la posición de un robot manipulador utilizando Matlab-CoppeliaSim.
- Implementar la técnica de control servovisual indirecto utilizando la configuración ojo en mano.
- Comparar el error de posición encontrado al aplicar los controladores PID, ISMC y PID+G.

1.6. Limitaciones

- El presente trabajo se enfoca en el seguimiento de un objeto en movimiento para lograr enfocarlo y no se contempla la captura del mismo.
- La retroalimentación del sistema está dada por solo una cámara colocada en el efector final del robot debido a la técnica de control servovisual que se pretende utilizar.
- Se trabaja con un manipulador de 3 grados de libertad, ya que el aumentar dichos grados de libertad aumentaría la complejidad del proyecto, el cual no puede exceder un periodo de 2 años para su finalización en tiempo y forma. Con respecto al modelo dinámico, se toma como base el trabajo desarrollado en años anteriores en [22] y [2].

A continuación se presentan en el capítulo 2 los fundamentos teóricos y las bases necesarias para desarrollar este trabajo de tesis, en el capítulo 3 se presentan el modelo 3D del

robot manipulador, así como el modelo cinemático y dinámico. En el capítulo 4 se muestran los pasos desarrollados para llevar a cabo el control servovisual del manipulador y posteriormente en el capítulo 5 se presentan los resultados obtenidos al simular diferentes trayectorias para evaluar el comportamiento del sistema. Para finalizar, en el caítulo 5 se presentan las conclusiones y trabajos futuros.

Capítulo 2

Fundamentos teóricos

El presente capítulo tiene como finalidad mostrar las bases fundamentales sobre las cuales se desarrolla el actual trabajo de tesis. Se abordan conceptos básicos referentes a los grupos principales dentro de la clasificación de robots, también se explora información acerca de los componentes esenciales para el funcionamiento de los robots manipuladores, y aspectos esenciales referentes al control servovisual.

2.1. Robots

Según [23] el término robot apareció por primera vez en 1921, en la obra teatral Rosum's Universal Robots del autor Karel Capek, dado que la palabra robota"significa fuerza de trabajo o servidumbre, es decir que el término se asocia a la idea de trabajo y producción, en un inicio como una forma de reproducir los movimientos además de potenciar las habilidades humanas y posteriormente para desarrollar procesos industriales automatizados libres de interacción humana. Principalmente, los robots conforman 2 grandes grupos: los robots móviles y los manipuladores.

2.1.1. Robots manipuladores

Se menciona en [4] que un robot manipulador es en esencia un brazo articulado reprogramable y multifuncional, diseñado para mover materiales, piezas, herramientas o dispositivos

esenciales, mediante movimientos variados, donde una de sus principales ventajas es la reprogramabilidad, la cual dota al manipulador de la capacidad de ser útil y adaptarse a diversas tareas.

Puesto que el presente trabajo se centra en los robots manipuladores, a continuación se describen clasificaciones de robots manipuladores de acuerdo a su modo de operación y a su estructura y tipo de articulaciones.

2.1.2. Robots manipuladores de acuerdo a su modo de operación

Según la forma de manejar lo robots manipuladores se ha desarrollado la siguiente clasificación:

Robot teleoperado

En [24] se menciona que existen diversas ocasiones en las cuales se requiere evitar el empleo de personas para la realización de tareas que podrían resultar peligrosas. Cuando se expone la integridad física de las personas para completar una tarea, resulta útil contar con robots teleoperados, los cuales, son aquellos controlados por un usuario a distancia. Dada su gran utilidad, se han empleado en diversos campos.

Robot adaptativo

Conforme a [25], se trata del robot que se adapta al entorno de trabajo para cumplir su tarea programada, esto se logra gracias a sensores, visión artificial, redes neuronales, entre otros. Actualmente aplicados a la industria de la manufactura.

Robot secuencial

En [25] se menciona que en este tipo de robots solo se puede controlar una serie de movimientos con base en dos puntos establecidos (Point to Point: PTP). Los manipuladores de tipo neumático son un claro ejemplo de este tipo de control.

Robot controlado por trayectoria

Según [25] estos dispositivos, son controlados con base en una programación de trayectoria previa, lo cual permite que el robot siga una trayectoria continua dentro de su área de trabajo, usado en la industria para trabajos cíclicos.

2.1.3. Robots manipuladores de acuerdo a su estructura

De acuerdo con [26], dependiendo del tipo de articulaciones presentes en la estructura del manipulador, existe la siguiente clasificación:

Robot cartesiano

Utiliza tres ejes de movimiento lineal (Articulación prismática o lineal) perpendiculares entre sí. Esta configuración da lugar a robots de alta precisión, con velocidad y capacidad de carga constante en todo su alcance, gran capacidad de carga, amplia zona de trabajo y simplificación del sistema de control. En comparación con las configuraciones que incluyen articulaciones de rotación, presentan una mala relación entre su volumen de trabajo y el espacio que ocupan en planta.

Robot Cilíndrico

Cuando el brazo de un robot tiene una articulación de rotación y dos prismáticas, es decir, si la primera articulación prismática del tipo cartesiano, es reemplazada por una articulación de rotación con su eje girado 90° respecto al eje z, los puntos que puede alcanzar pueden ser convenientemente especificados con coordenadas cilíndricas, es decir, ángulo β, altura y radio z. Un robot con este tipo de brazo se denomina robot cilíndrico, cuyo brazo se mueve por medio de, β y z, es decir, tiene una rotación de base, una elevación y un alcance, respectivamente. Puesto que las coordenadas del brazo pueden asumir cualquiera de los valores entre los límites superior e inferior especificados, su efector final puede moverse en un volumen limitado, que es una sección de corte dentro del espacio entre los dos cilindros concéntricos.

Polar o esférico

Cuando el brazo de un robot es capaz de cambiar su configuración moviendo sus dos articulaciones rotacionales y su articulación prismática, es decir, cuando la segunda articulación prismática a lo largo de la altura y del tipo cilíndrico es reemplazada por una articulación rotacional con su eje girado 90° respecto al eje z, se denomina brazo de robot esférico o polar; la posición del brazo se describe convenientemente por medio de las coordenadas esféricas q, f y z. Los movimientos del brazo representan la rotación de la base, los ángulos de elevación y el alcance, respectivamente.

Robot Scara

Es un robot con dos articulaciones R y una P, con las dos R se controla la posición respecto al plano X-Y y con la P la coordenada Z. Es rápido, barato y preciso, pero solo tiene accesibilidad a zonas de trabajo que estén en planos perpendiculares a su eje vertical. Se emplea fundamentalmente en operaciones de ensamblado o inserción de componentes electrónicos y en otros trabajos similares. Es originario de Japón y es allí donde más se emplea, su inconveniente inicial era la potencia de cálculo necesaria para determinar posiciones por combinación de giros, pero este problema se ha resuelto para este y otros robots.

Robot angular o antropomórfico

Tiene sus tres principales articulaciones de tipo rotacional, con lo cual emplea las coordenadas angulares para determinar las posiciones de su elemento terminal. Se llama antropomórfico por que simula los movimientos de un brazo humano, el primer eje se corresponde con el cuerpo, el segundo con el brazo, el tercero con el antebrazo y el resto de con la muñeca-mano; la primera articulación se corresponde con el giro de la cintura, la segunda con el del hombro, la tercera con el del codo y el resto están en la muñeca.

Puesto que el tipo de robot utilizado para el desarrollo del presente trabajo es el antropomórfico, enseguida se presentan algunas aplicaciones del mismo.

En la Figura 2.1 se muestran las principales configuraciones de robos manipuladores:

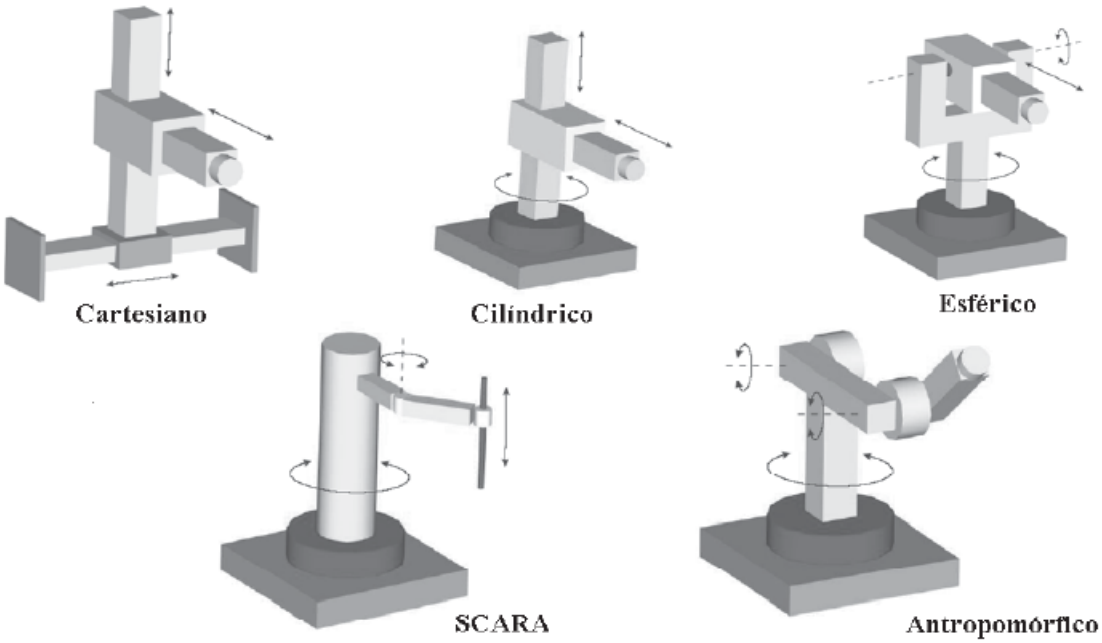

Figura 2.1: Clasificación de robots manipuladores de acuerdo con el tipo de articulación. Tomado de [2]

Aplicaciones del manipulador antropomórfico

Según información de [27] los robots manipuladores generalmente se utilizan dentro de la industria manufacturera, dentro de células de trabajo para completar tareas como:

- Realizar operaciones de carga y descarga a máquinas-herramientas.
- Carga y descarga de materiales.
- Empaquetado (en tarimas o pallets).
- Ensamble forma automatizado.
- Soldadura de punto o de arco.
- Pintura.
- Aplicación de diversos tipos de resinas.

- Eliminación del exceso de material de piezas.
- Remaches y estampados.
- Cortes.
- Incluso tienen aplicaciones médicas como la cirugía.

Es relevante conocer los componentes esenciales que permiten el óptimo funcionamiento y control de un robot manipulador.

2.1.4. Componentes del robot manipulador

- Eslabones: Conforme a [28] se refiere a la secuencia de cuerpos rígidos unidos mediante las articulaciones.
- Articulaciones: En [23] se menciona que es el punto de unión de 2 eslabones y estas permiten el movimiento relativo del robot entre eslabones sucesivos.

  Existen 2 tipos principales de articulaciones:

  - Rotacionales: Permiten a los eslabones rotar alrededor del eje de la articulación.
  - Prismáticas: Dota a los eslabones de un movimiento de traslación a lo largo del eje de la articulación.
- Actuadores: De acuerdo con [23] estos dispositivos generan la fuerza o par necesarios para dotar de movimiento al robot. Pueden ser actuadores neumáticos, hidráulicos o motores eléctricos.

En el actual trabajo de investigación se manejan 3 articulaciones, a saber:

- Hombro: Guarda similitud con el hombro humano determina la altura a la cual se posicionará el efector final.
- Codo: Determina el alcance máximo del efector final.
- Muñeca: Es la articulación encargada de determinar la orientación del efector final.

  Según la información presentada en [28]

2.1.5. Grados de libertad

Siguiendo la información de [28] en una cadena cinemática abierta un grado de libertad se refiere a cada movimiento provisto por cada una de las articulaciones con las que cuenta el robot, es decir el número de grados de libertad es igual al número de articulaciones.

En cambio,en una cadena cinemática cerrada, el número de grados de libertad es menor que el número de articulaciones dependiendo de las restricciones aplicadas al sistema.

El número de grados de libertad también se relaciona con el número de actuadores, en conformidad con [29] existen:

- Sistemas mecánicos completamente actuados: Son aquellos que poseen igual número de actuadores que de grados de libertad.

- Sistemas mecánicos subactuados: Poseen menor número de actuadores que de grados de libertad.

2.2. Control servovisual

Una vez que se conocen los conceptos fundamentales correspondientes a la parte mecánica del robot manipulador, es relevante tener conocimiento de las bases concernientes al control servovisual que se pretende aplicar en el robot manipulador.

Es importante definir a qué se refiere el término control servovisual, según [30] y [31], el control servovisual en el área de manipuladores, se refiere al proceso de utilizar información visual captada a través de sensores, para controlar la posición del efector final con respecto de un objetivo, el cual puede estar fijo o en movimiento.

A continuación se presentan algunas de las clasificaciones más importantes dentro del control servovisual. De acuerdo con la configuración de retroalimentación del lazo de control existen 2 tipos de sistemas servovisuales: directo e indirecto. El sistema de tipo directo utiliza solo un lazo de control, en cambio, el sistema indirecto, usa un lazo dual.

Sistema servovisual directo

De acuerdo con [13] y [30], en el sistema servovisual directo, el controlador basado en visión calcula directamente la entrada del sistema dinámico y únicamente utiliza información visual para estabilizar al sistema. Este sistema requiere una frecuencia de muestreo alta y puede resultar difícil utilizar esta clase de sistemas cuando el robot se mueve rápidamente.

Sistema servovisual indirecto

Para los sistemas indirectos, se calcula la ley de control de referencia que el controlador basado en visión envía a un controlador inferior del sistema dinámico, es decir, el sistema de visión envía las entradas para el controlador de las articulaciones y posteriormente, a través de la retroalimentación de las articulaciones, el robot se estabiliza internamente. A diferencia del sistema directo, el sistema indirecto tiene una estructura de lazo dual, lo cual es adecuado para sistemas de visión con frecuencias de muestreo bajas [13], [30].

2.2.1. Sensor de visión

Los sensores de visión conforman una parte fundamental para lograr una implementación adecuada del control servovisual, por lo cual es importante conocer los fundamentos correspondientes a estos sensores.

Un sensor de visión es un dispositivo que imita el sentido de visión humano y posibilita la medición sin contacto del entorno para retroalimentar el lazo de control del sistema, además pueden obtenerse características relevantes de la imagen como formas, colores, o dimensiones [10], [30].

En al área concerniente al control servovisual aplicado a robots manipuladores, se utilizan principalmente dos formas de colocar las cámaras o sensores de visión con respecto al espacio de trabajo del robot, a saber: cámara fija y cámara en mano.

Cámara fija

Como su nombre lo indica, una o varias cámaras están montadas en un punto fijo en el espacio de trabajo del robot, la imagen captada no depende del movimiento del robot.

Una de las más grandes desventajas de la configuración cámara fija es que el campo de visión puede quedar bloqueado por el propio movimiento del robot, además de que el área a visualizar puede ser muy reducida si el sensor es de baja calidad [30].

Cámara en mano

La configuración cámara en mano se refiere al hecho de montar una o varias cámaras en el efector final del robot manipulador, por lo tanto, la imagen depende del movimiento del brazo robótico, la ventaja de esta configuración es que se puede acceder a todo el espacio de trabajo del robot conforme este vaya desplazando su efector final [30].

Dado que se puede obtener una mayor cantidad de información visual al contar con un campo de visión más extenso, para el presente trabajo se utilizará la configuración de cámara en mano.

Los esquemas servovisuales pueden ser clasificados dependiendo de cómo se utilizarán los datos captados por los sensores. Por ejemplo: control servovisual basado en imagen, en posición y control por píxeles, estos esquemas de control se abordan en las secciones posteriores.

2.2.2. Control servovisual basado en imagen

Usa datos de la imagen en la retroalimentación del lazo de control. Este método de control tiene como objetivo posicionar y rastrear el objetivo en el plano de la imagen reduciendo el error de distancia entre las características de la imagen actual y el objetivo [13], [32].

Si el objeto está fijo con respecto al marco base, se puede formular un sistema servovisual basado en imágenes estipulando que el vector de los parámetros de características del objeto tiene un valor constante deseado s_d correspondiente a la pose deseada de la cámara. Por lo tanto, se asume implícitamente que existe una pose deseada x_d, de modo que la pose de la cámara pertenece al espacio de trabajo del manipulador [28].

Aunque existen algunas ventajas al utilizar el control servovisual basado en posición, el esquema servovisual basado en imagen resulta más sencillo de implementar, ya que no se requiere altas velocidades de procesamiento ni modelos matemáticos complicados, por lo cual en el trabajo actual se utilizará un esquema basado en imagen.

En el presente proyecto se implementa el control servovisual con cámara en mano basado en imagen, por lo cual a continuación se muestran conceptos fundamentales para la implementación de esta técnica.

Una característica de la imagen es cualquier variable que incluya relaciones entre componentes estructurales de la imagen tales como líneas, puntos, áreas o parámetros cuantitativos asociados a ellas. Se define un parámetro característico de la imagen como cualquier cantidad real valuada que se puede calcular desde una o más características de la imagen. Algunos de los parámetros característicos más ampliamente conocidos son las coordenadas de un punto en la imagen, la distancia entre dos puntos en el plano de la imagen y la orientación de la línea que los conecta, y el área de la superficie proyectada.

Para trabajar con las características de la imagen, es necesario conocer los parámetros de la cámara. Debe tenerse en cuenta que se utilizan parámetros como la distancia focal f, el vector de posición de la característica del objeto ${}^cP(X_c, Y_c, Z_c)$, la proyección de la característica del objeto (x_i, y_i) y la proyección óptica central del eje óptico (x_p, y_p).

Según [33], la relación entre el objeto y un punto de su imagen está dada por:

$$(x_i - x_p) = x_s = \frac{fX_c}{s_y Z_c} \tag{2.1}$$

$$(y_i - y_p) = y_s = \frac{fY_c}{s_x Z_c} \tag{2.2}$$

Donde s_x y s_y son las dimensiones horizontal y vertical de los píxeles.

2.2.3. Modelo de Cámara Pinhole

El modelo de cámara pinhole es una aproximación simplificada que describe la proyección de una escena tridimensional en un plano de imagen bidimensional. En este modelo, se considera que la cámara consiste en un orificio (pinhole) a través del cual los rayos de luz pasan y se proyectan sobre el plano de imagen.

La proyección de un punto tridimensional en la cámara se realiza trazando una línea recta desde el punto hasta el plano de imagen como se muestra en la Figura 2.2. La intersección de esta línea con el plano de imagen representa la proyección del punto en la imagen. La

posición del punto en el plano de imagen se determina por las coordenadas de píxel (u, v), mientras que su posición tridimensional se especifica mediante las coordenadas (X, Y, Z) en el sistema de coordenadas de la escena.

El modelo matemático de la proyección en el cámara pinhole se puede describir mediante las siguientes ecuaciones [3]:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.3}$$

Donde (u, v) son las coordenadas del píxel en la imagen, (X, Y, Z) son las coordenadas tridimensionales del punto en la escena, y f_x, f_y, c_x, c_y son los parámetros intrínsecos de la cámara que representan las distancias focales y el punto principal.

2.2.4. Plano discreto de la imagen

En una cámara digital, el plano de la imagen es una cuadrícula W × H de elementos sensibles a la luz llamados fotositos que corresponden directamente a los elementos de imagen (o píxeles). Las coordenadas de los píxeles componen un vector de 2 enteros no negativos (u,v) y por convención el origen está en la esquina superior izquierda del plano de la imagen.

Los píxeles son uniformes en tamaño y están centrados en una cuadrícula regular, de modo que la coordenada del píxel está relacionada con la coordenada del plano de la imagen mediante:

$$u = \frac{x}{\rho_w} + u_0 \tag{2.4}$$

$$v = \frac{y}{\rho_h} + v_0 \tag{2.5}$$

Donde ρ_w y ρ_h son el ancho y la altura de cada píxel, respectivamente, y (u_0 , v_0) son las coordenadas del píxel del punto en el que el eje óptico se cruza con el plano de la imagen respecto al nuevo origen.

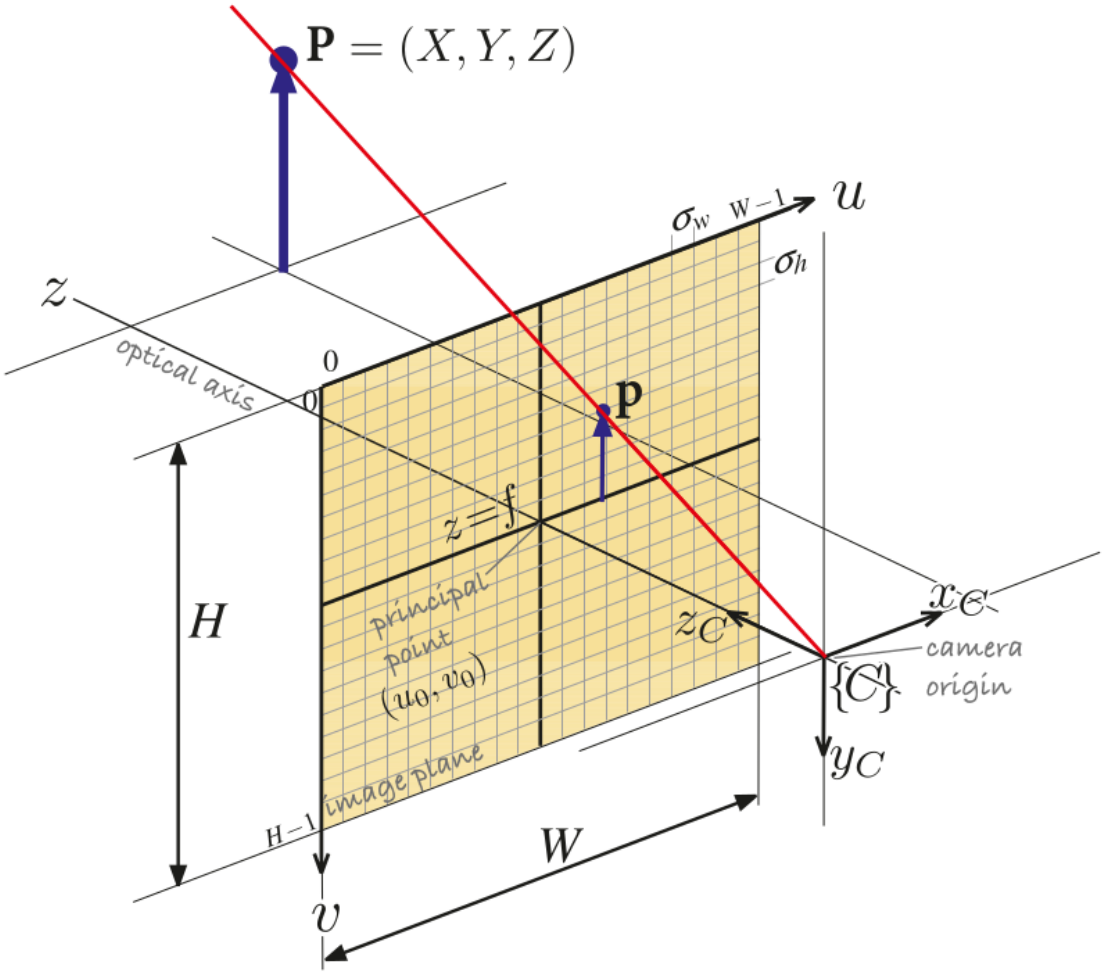

Figura 2.2: Modelo de proyección central del plano de la imagen y pixeles. Tomado de [3].

La proyección en perspectiva puede escribirse en términos de las coordenadas de píxel [3]:

$$\tilde{p} = \begin{bmatrix} 1/\rho_w & 0 & u_0 \\ 0 & 1/\rho_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} .^C\tilde{P} \tag{2.6}$$

Donde $\tilde{p} = (\tilde{u}, \tilde{v}, \tilde{w})$ es la coordenada homogénea del punto en el mundo $\mathbf{P}$, en coordenadas de píxel.

La proyección de perspectiva también puede ser escrita en forma de función como:

$$p = P(\mathbf{P}, \mathbf{K}, \xi_C) \tag{2.7}$$

Donde $\mathbf{P}$ es el vector de coordenadas del punto en el mundo, $\mathbf{K}$ es la matriz de parámetros de la cámara, e integra los parámetros intrínsecos, los cuales son características innatas de la cámara o sensor, tales características son: f, ρ_w, ρ_h, u_0 y v_0. ξ_C es la pose de la cámara y se compone por un mínimo de 6 parámetros-los parámetros extrínsecos- que describen la traslación y rotación de la cámara [33].

La derivada de la proyección de perspectiva con respecto al tiempo es:

$$\dot{p} = J_p(\mathbf{P}, \mathbf{K}, \xi_C)v \tag{2.8}$$

Donde $v = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)\ \epsilon\ \mathbb{R}^6$ es la velocidad espacial de la cámara.

J_p es un pseudo-Jacobiano, pero dado que se ha utilizado la derivada con respecto a la pose $\xi\ \epsilon\ \mathbf{SE}(3)$, en lugar de un vector, es llamada técnicamente una matriz de interacción. Sin embargo, en el ámbito del control servovisual, comúnmente es llamado Jacobiano de la imagen.

Considere que una cámara se mueve con una velocidad v=(v, ω) en el marco de referencia del mundo y observa un punto $\mathbf{P}$ en el mundo con coordenadas respecto de la cámara $\mathbf{P} = (X, Y, Z)$.

La velocidad del punto con respecto al marco de referencia de la cámara es:

$$\dot{\mathbf{P}} = -\omega \times \mathbf{P} - v \tag{2.9}$$

La velocidad también puede ser escrita en forma escalar como:

$$\begin{aligned} \dot{X} &= Y\omega_z - Z\omega_y - v_x \\ \dot{Y} &= Z\omega_x - X\omega_z - v_y \\ \dot{Z} &= X\omega_y - Y\omega_x - v_z \end{aligned} \tag{2.10}$$

La proyección de perspectiva para coordenadas normalizadas imagen-plano es:

$x = \frac{X}{Z}$, $y = \frac{Y}{Z}$

Y la derivada temporal está descrita por:

$$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2}, \dot{y} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} \tag{2.11}$$

Sustituyendo la ec. 2.10, junto con X=xZ y Y=yZ, en la ec. 2.11 se puede escribir en forma matricial:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.12}$$

Lo cual relaciona la velocidad espacial de la cámara con la velocidad de la característica en coordenadas normalizadas de la imagen.

ahora, es necesario establecer la relación entre las coordenadas normalizadas en el plano de la imagen con las coordenadas de píxel a través de:

$u = \frac{f}{\rho_u}x + u_0$, $v = \frac{f}{\rho_v}y + v_0$

Reescribiendo lo anterior se tiene:

$$x = \frac{\rho_u}{f}\bar{u}, y = \frac{\rho_v}{f}\bar{v} \tag{2.13}$$

Donde $\bar{u} = u - u_0$ y $\bar{v} = v - v_0$ son las coordenadas de píxel con respecto al punto principal. La derivada temporal es:

$$\dot{x} = \frac{\rho_u}{f}\dot{\bar{u}}, \dot{y} = \frac{\rho_v}{f}\dot{\bar{v}} \tag{2.14}$$

Y sustituyendo 2.13 y 2.14 en 2.12 tiene que:

$$\begin{bmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \end{bmatrix} = \begin{bmatrix} \frac{-f}{\rho_u Z} & 0 & \frac{\bar{u}}{Z} & \frac{\rho_v \bar{u}\bar{v}}{f} & \frac{-f^2+\rho_u^2\bar{u}^2}{\rho_u f} & \frac{\rho_v \bar{v}}{\rho_u} \\ 0 & \frac{-f}{\rho_v Z} & \frac{\bar{v}}{Z} & \frac{-f^2+\rho_v^2\bar{v}^2}{\rho_v f} & \frac{-\rho_u \bar{u}\bar{v}}{f} & \frac{-\rho_u \bar{u}}{\rho_v} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.15}$$

Finalmente para el caso en el cual ρ_u=ρ_v=ρ, se puede expresar la distancia focal en píxeles $f' = f/\rho$, formando así la matriz Jacobiana de la imagen.

2.3. Jacobiano de la imagen

El Jacobiano de la imagen es una matriz que relaciona las velocidades de un punto en la imagen con las velocidades de un punto en el espacio tridimensional. Este Jacobiano es fundamental para el control servovisual, ya que proporciona información sobre cómo deben cambiar las coordenadas de píxel en respuesta a los cambios en las coordenadas tridimensionales [3].

$$L_s = \begin{bmatrix} \frac{-f'}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{f'} & \frac{-f'^2+\bar{u}^2}{f'} & \bar{v} \\ 0 & \frac{-f'}{Z} & \frac{\bar{v}}{Z} & \frac{f'^2+\bar{v}^2}{f'} & \frac{-\bar{u}\bar{v}}{f'} & -\bar{u} \end{bmatrix} \tag{2.16}$$

La matriz jacobiana de la imagen tiene interesantes propiedades, puesto que no depende en absoluto de las coordenadas del mundo X o Y, en cambio, depende de las coordenadas del

plano de la imagen (u,v). Sin embargo, las 3 primeras columnas dependen de la profundidad del punto denominada Z.

La primera columna indica el movimiento en dirección z, a lo largo de un rayo que se dirige al punto objetivo, y no representa movimiento en la imagen, tampoco existe rotación alrededor de z, como se indica en la columna 4. Las columnas 2 y 3 son más complejas, dado que combinan rotación y traslación.

Se puede considerar el movimiento de dos puntos al calcular 2 jacobianos, como se muestra a continuación:

$$\begin{bmatrix} \dot{u_1} \\ \dot{v_1} \\ \dot{u_2} \\ \dot{v_2} \end{bmatrix} = \begin{bmatrix} L_s(p_1, Z_1) \\ L_s(p_2, Z_2) \end{bmatrix} \xi \tag{2.17}$$

Y así se obtiene una matriz de 4x6, la cual tendrá un espacio nulo con dos columnas. Ahora, considerando el movimiento de 3 puntos de la imagen, se tiene que:

$$\begin{bmatrix} \dot{u_1} \\ \dot{v_1} \\ \dot{u_2} \\ \dot{v_2} \\ \dot{u_3} \\ \dot{v_3} \end{bmatrix} = \begin{bmatrix} L_s(p_1, Z_1) \\ L_s(p_2, Z_2) \\ L_s(p_3, Z_3) \end{bmatrix} \xi \tag{2.18}$$

De forma explícita:

$$L_s = \begin{bmatrix} \frac{-f'}{Z_1} & 0 & \frac{\bar{u_1}}{Z_1} & \frac{\bar{u_1}\bar{v_1}}{f'} & \frac{-f'^2+\bar{u_1}^2}{f'} & \bar{v_1} \\ 0 & \frac{-f'}{Z_1} & \frac{\bar{v_1}}{Z_1} & \frac{f'^2+\bar{v_1}^2}{f'} & \frac{-\bar{u_1}\bar{v_1}}{f'} & -\bar{u_1} \\ \frac{-f'}{Z_2} & 0 & \frac{\bar{u_2}}{Z_2} & \frac{\bar{u_2}\bar{v_2}}{f'} & \frac{-f'^2+\bar{u_2}^2}{f'} & \bar{v_2} \\ 0 & \frac{-f'}{Z_2} & \frac{\bar{v_2}}{Z_2} & \frac{f'^2+\bar{v_2}^2}{f'} & \frac{-\bar{u_2}\bar{v_2}}{f'} & -\bar{u_2} \\ \frac{-f'}{Z_3} & 0 & \frac{\bar{u_3}}{Z_3} & \frac{\bar{u_3}\bar{v_3}}{f'} & \frac{-f'^2+\bar{u_3}^2}{f'} & \bar{v_3} \\ 0 & \frac{-f'}{Z_3} & \frac{\bar{v_3}}{Z_3} & \frac{f'^2+\bar{v_3}^2}{f'} & \frac{-\bar{u_3}\bar{v_3}}{f'} & -\bar{u_3} \end{bmatrix} \tag{2.19}$$

Esta matriz será de rango completo, no singular, mientras que los puntos no sean coincidentes o colineales.

2.3.1. Inversa del Jacobiano de la imagen

Anteriormente se ha analizado cómo se mueven los puntos en la imagen a consecuencia del movimiento de la cámara, sin embargo, comúnmente es más útil abordar el problema inverso: ¿qué movimiento de la cámara se necesita para mover las características de la imagen a una velocidad deseada? Para el caso de 3 puntos $(u_i, v_i), i = 1, ..., 3$ y sus velocidades correspondientes (u_i, v_i) se puede invertir la ecuación correspondiente a la velocidad del punto en el plano de la imagen:

$$\xi = \begin{bmatrix} L_s(p_1, Z_1) \\ L_s(p_2, Z_2) \\ L_s(p_3, Z_3) \end{bmatrix}^{-1} \begin{bmatrix} \dot{u_1} \\ \dot{v_1} \\ \dot{u_2} \\ \dot{v_2} \\ \dot{u_3} \\ \dot{v_3} \end{bmatrix} \tag{2.20}$$

De esta forma se calcula la velocidad requerida de la cámara [4].

Dada la velocidad de las características de la imagen, se puede calcular el movimiento requerido de la cámara, ¿cómo se determina esta velocidad de las características? El método más simple, consiste en utilizar un controlador lineal:

$$\dot{p^*} = \lambda(p^* - p) \tag{2.21}$$

Dicho controlador, conduce a las características hacia los valores deseados p^* en el plano de la imagen. Combinando la ecuación 2.21 con la ecuación 2.20 para obtener la velocidad requerida de la cámara se tiene:

$$\xi = \lambda \begin{bmatrix} L_s(p_1, Z_1) \\ L_s(p_2, Z_2) \\ L_s(p_3, Z_3) \end{bmatrix}^{-1} \begin{bmatrix} \dot{u_1} \\ \dot{v_1} \\ \dot{u_2} \\ \dot{v_2} \\ \dot{u_3} \\ \dot{v_3} \end{bmatrix} (p^* - p) \tag{2.22}$$

Este controlador conducirá a la cámara de tal forma que los puntos característicos se moverán hacia la posición deseada en la imagen. Es importante notar que no se ha requerido en ningún momento de la pose de la cámara o del objeto, todo ha sido calculado a partir de lo que puede ser medido en el plano de la imagen.

Para el caso general en el cual el número de puntos es $N > 3$, podemos calcular los jacobianos para todas las características y resolver para el movimiento de la cámara usando la pseudo-inversa.

$$\xi = \lambda \begin{bmatrix} L_s(p_1, Z_1) \\ \vdots \\ L_s(p_N, Z_N) \end{bmatrix}^+ (p^* - p) \tag{2.23}$$

El jacobiano de la imagen es una aproximación de primer orden de la relación entre el movimiento de la cámara y el movimiento del plano de la imagen. Se puede alcanzar la convergencia más rápido al utilizar una aproximación de segundo orden y esto puede obtenerse de forma simple como se muestra a continuación:

$$\xi = \frac{\lambda}{2} \left(\begin{bmatrix} L_s(p_1, Z_1) \\ \vdots \\ L_s(p_N, Z_N) \end{bmatrix}^+ + \begin{bmatrix} L_s(p_1^*, Z_1^*) \\ \vdots \\ L_s(p_N^*, Z_N^*) \end{bmatrix}^+ \right) (p^* - p) \tag{2.24}$$

Al obtener la media de la pseudoinversa de los jacobianos de la imagen en los estados actual y deseado.

Los datos de las características de la imagen obtenidos del sistema de visión se definen como $s = \{x_1, y_1, \ldots, x_N, y_N\}$, $\dot{s}$ representa la velocidad de las características y la salida del control servovisual está definida como $\xi = [\dot{x}_c, \dot{y}_c, \dot{z}_c, \omega_{xc}, \omega_{yc}, \omega_{zc}]^T$, que representa la velocidad de la cámara.

2.4. Modelo del robot manipulador

Ya que se tienen las nociones más relevantes sobre el control servovisual, es necesario conocer aspectos referentes al modelo del robot manipulador, hablando del ámbito CAD, cinemático y dinámico.

Es importante abordar primero el modelo 3D elaborado utilizando software CAD, ya que

con base en este se desarrollan los modelos cinemático y dinámico.

2.4.1. CAD

CAD es el acrónimo en inglés correspondiente a Computer Aided Design, es decir, diseño asistido por computadora. Se refiere a las tecnologías de computación que se aplican en la creación, modificación,análisis y optimización de un diseño. Por lo tanto, cualquier programa de computadora que incorpore gráficos por computadora y un programa de aplicación que facilite las funciones de ingeniería en el proceso de diseño se conoce como software CAD [34].

En el actual trabajo de investigación el software CAD a utilizar es conocido como SolidWorks®. Es necesario realizar un modelo CAD ya que a través de este es posible ubicar espacialmente los componentes del robot para posteriormente calcular las ecuaciones dinámicas y cinemáticas correspondientes, teniendo en cuenta las distancias entre los puntos de referencia y la posición de los componentes de interés, además de realizar simulaciones para validar el diseño antes de poder construir el modelo físico.

2.4.2. Modelo dinámico

El modelo dinámico se refiere a la relación matemática que vincula las variables de entrada y salida del sistema. Esta caracterización matemática se describe por medio de ecuaciones diferenciales. El modelo matemático del sistema a controlar se realiza de forma analítica, es decir, basado en las leyes de la física que rigen el comportamiento del sistema.

Las ecuaciones dinámicas de un robot manipulador pueden obtenerse a partir de las ecuaciones de movimiento de Newton. El inconveniente que presenta este método es que el análisis se complica notablemente cuando aumenta el número de articulaciones del robot, ya que se requiere analizar más parámetros cada vez. En estos casos es conveniente emplear las ecuaciones de movimiento de Lagrange.

El uso de las ecuaciones de Lagrange para el modelado dinámico del robot manipulador de tres grados de libertad se reduce a cuatro etapas:

1. Cálculo de la energía cinética.

2. Cálculo de la energía potencial.
3. Cálculo del lagrangiano.
4. Desarrollo de las ecuaciones de Lagrange [22].

Las ecuaciones del modelo dinámico se presentan en capítulos posteriores.

2.4.3. Modelo cinemático

El modelo cinemático describe el movimiento del robot con respecto a un sistema de referencia sin considerar las fuerzas y torques que lo causan [35], [4]. Este análisis es fundamental, ya que al asignar cualquier tarea a un brazo robótico es necesario conocer los puntos que alcanzará su efector final dependiendo de las posiciones articulares, ya que generalmente se busca mover el efector final de un punto A a un punto B.

La cinemática directa consiste en determinar la posición y orientación del extremo del robot a partir de sus posiciones articulares y de los parámetros geométricos de sus eslabones. Por otra parte, la cinemática inversa calcula la configuración articular que el robot debe adoptar para alcanzar una posición y orientación conocidas [36]. Sin embargo, antes de obtener el modelo cinemático del robot, es necesario conocer conceptos fundamentales como sistesmas de referencia, rotación y traslación homogénea entre otros.

Sistemas de coordenadas o de referencia

Para desarrollar el análisis cinemático del manipulador, primero se establecen diversos sistemas de referencia que permiten representar posiciones y orientaciones de cuerpos rígidos, así como las transformaciones entre dichos sistemas. Posteriormente, se estudian las operaciones de rotación y traslación, y se introduce el concepto de transformaciones homogéneas.

Se puede considerar un cuerpo rígido como un punto p en el espacio, para conocer la posición y orientación de este punto es necesario definir sistemas de referencia como o_0 y o_1 mostrados en la Figura 2.3.

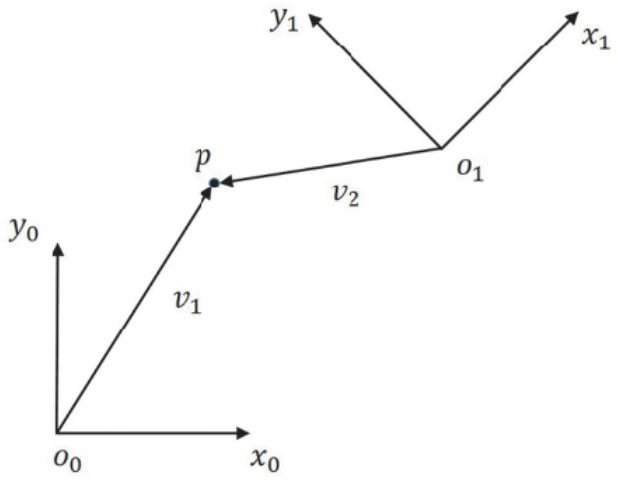

Figura 2.3: Sistemas de referencia o_0, o_1 y un punto p en el espacio. Tomado de [4]

Es importante notar que, dependiendo del sistema de referencia elegido, la representación en coordenadas del punto p cambiará. Para facilitar la identificación del marco de referencia con respecto al cual se trabaja, se utilizará un superíndice para denotarlo. Por lo tanto, la representación de p con respecto a o_0 se define como p^0, mientras que p^1 indica que se está utilizando el sistema de coordenadas o_1.

Matrices de Rotación y Traslación Homogéneas

Una vez definidos los marcos de referencia, es posible pasar de uno a otro mediante una secuencia de transformaciones básicas (rotaciones y traslaciones) que dependen de las características geométricas de cada eslabón, relacionando así un sistema de coordenadas con otro [37]. Las rotaciones y traslaciones definidas por la tabla de parámetros de Denavit-Hartenberg, (en adelante, DH) pueden representarse en su forma matricial utilizando las matrices homogéneas de rotación y traslación.

La matriz de rotación homogénea alrededor del eje x, está definida como:

$$Rot_{x,\alpha_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.25}$$

La matriz de traslación homogénea a lo largo de x se expresa de la siguiente forma:

$$Tras_{x,a_i} = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.26}$$

La matriz de rotación homogénea en el eje z se muestra a continuación:

$$Rot_{z,\theta_i} = \begin{pmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.27}$$

Finalmente, la matriz de traslación homogénea a través de z está definida como:

$$Tras_{z,d_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.28}$$

Estas operaciones de rotación y traslación dan lugar a una matriz de transformación de paso A_i, la cual está definida como el producto de las rotaciones y traslaciones mostradas anteriormente. De acuerdo con la convención de D-H cada matriz de transformación A_i se obtiene como el producto de las 4 mátrices básicas de transformación de las ecs. 2.28, 2.27, 2.26 y 2.25

$$A_i = Rot_{z,\theta_i} Tras_{z,d_i} Tras_{x,a_i} Rot_{x,\alpha_i} \tag{2.29}$$

Ahora, se denota A_i como la matriz de transformación homogénea de paso que contiene la posición y orientación de un sistema de referencia con respecto a otro cuya forma se muestra en la ec. 2.30 :

$$A_i = \begin{pmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{pmatrix} \tag{2.30}$$

Posteriormente se representa la posición y orientación del efector final con respecto al marco base con un vector o_n^0 y la matriz de rotación de tamaño 3x3 llamada R_n^0, y se define

la matriz de transformación homogénea total como:

$$H = \begin{pmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{pmatrix} \tag{2.31}$$

Donde T_n^0:

$$H = T_n^0 = A_1(q_1)A_2(q_2)...A_n(q_n) \tag{2.32}$$

Cinemática directa

Denavit y Hartenberg propusieron un algoritmo para obtener el modelo cinemático directo. Se trata de un método matricial sistemático que permite asignar sistemas de coordenadas a cada eslabón de un mecanismo y, con ello, describir la cinemática completa, representando la posición y orientación del efector final como un producto matricial de transformaciones homogéneas.

Un robot manipulador con **n** articulaciones tendrá **n + 1** eslabones, ya que cada junta conecta dos eslabones. Las juntas se enumeran de 1 a **n** y los eslabones de 0 a **n**, comenzando por la base. Siguiendo la convención de Denavit–Hartenberg, la junta **i** conecta al eslabón **i-1** con el eslabón **i**; al accionar la articulación **i**, el eslabón **i** se mueve. A cada junta se asocia una variable q_i: θ_i si la junta es revoluta o d_i si la junta es prismática. Posteriormente, se asigna un marco de referencia a cada eslabón, es decir, se definen los ejes x_i, y_i, z_i y el origen o_i del sistema asociado al eslabón i [28], [37].

Se siguieron los pasos del algoritmo DH para obtener la cinemática directa del manipulador [4].

Las ecuaciones cinemáticas del robot manipulador se presentan en apartados posteriores.

Cinemática inversa

La cinemática inversa (CI) tiene como objetivo determinar el vector de variables articulares $q = [q_1\ q_2\ q_3]^T$ a partir de una posición deseada del efector final $p_d = [x_d\ y_d\ z_d]^T$ [38]. En términos generales, si la cinemática directa se expresa como

$$p = f(q),$$

la CI consiste en resolver el problema inverso

$$q = f^{-1}(p_d).$$

En la práctica, $f^{-1}(\cdot)$ no suele ser una función única, ya que pueden existir múltiples configuraciones articulares que alcanzan el mismo punto, o bien posiciones no alcanzables debido a restricciones geométricas y límites articulares.

Particularidades en robots de 3 GDL. En un manipulador de 3 GDL, típicamente formado por tres articulaciones revolutas, la CI se enfoca principalmente en alcanzar una **posición** en el espacio. A diferencia de un robot de 6 GDL, no siempre es posible especificar de manera independiente posición y orientación del efector final; por tanto, el planteamiento usual consiste en obtener q para lograr p_d dentro del espacio de trabajo del robot.

Solución geométrica. Cuando la geometría del robot lo permite, la CI puede resolverse de forma analítica mediante trigonometría, en el presente proyecto se utilizó este método para obtener la CI. En este enfoque se construye un triángulo con longitudes equivalentes a los eslabones y la distancia desde el origen hasta el punto objetivo [38]. Con ello se obtiene un conjunto de ecuaciones que permiten calcular los ángulos articulares. Este método es computacionalmente eficiente y adecuado para aplicaciones en tiempo real; sin embargo, requiere un modelo geométrico claro y puede presentar más de una rama de solución.

2.5. Cinemática de velocidad (El jacobiano)

La cinemática de velocidad se refiere a las ecuaciones que establecen la relacion entre las velocidades articulares y las velocidades linear y angular del efector final. Esta relación generalmente se representa por una matriz llamada Jacobiano [38]. El jacobiano es la matriz equivalente de derivar un vector con respecto a otro vector. Si $y = f(x)$ y $x\epsilon\mathbb{R}^n$, con $y\epsilon\mathbb{R}^m$, entonces el Jacobiano es una matriz de $m \times n$.

Sea $\mathbf{p}$ la posición del efector final y $\mathbf{q}$ el vector de variables articulares; dado que $\mathbf{p}$ y $\mathbf{q}$ son ambos vectores, si se obtiene la derivada de $\mathbf{p}$ con respecto a $\mathbf{q}$, el resultado será una matriz, la matriz jacobiana:

$$\frac{d\mathbf{p}}{d\mathbf{q}} = \mathbf{J}(\mathbf{q}) \tag{2.33}$$

Para obtener la relación entre la velocidad articular y la velocidad del efector final de la ec. 2.33 se tiene que:

$$d\mathbf{p} = \mathbf{J}(\mathbf{q})d\mathbf{q} \tag{2.34}$$

Y al dividir ambos lados entre dt se obtiene:

$$\frac{d\mathbf{p}}{dt} = \mathbf{J}(\mathbf{q})\frac{d\mathbf{q}}{dt} \tag{2.35}$$

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.36}$$

El jacobiano mapea la velocidad de las coordenadas articulares a coordenadas cartesianas del efector final y es una función de las coordenadas articulares.

El Jacobiano analítico, denotado como J_a, se calcula tomando las derivadas parciales de las coordenadas del efector final con respecto a las coordenadas articulares. A continuación, se presenta la fórmula para calcular el Jacobiano analítico:

$$J_a = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \\ \frac{\partial \alpha}{\partial q_1} & \frac{\partial \alpha}{\partial q_2} & \cdots & \frac{\partial \alpha}{\partial q_n} \\ \frac{\partial \beta}{\partial q_1} & \frac{\partial \beta}{\partial q_2} & \cdots & \frac{\partial \beta}{\partial q_n} \\ \frac{\partial \gamma}{\partial q_1} & \frac{\partial \gamma}{\partial q_2} & \cdots & \frac{\partial \gamma}{\partial q_n} \end{bmatrix}$$

Donde:

-J_a es el Jacobiano analítico.

-$q_1, q_2, \ldots, q_n$ son las coordenadas articulares (ángulos de las articulaciones) del robot manipulador.

-x, y, z son las coordenadas espaciales del efector final.

-α, β, γ son los ángulos de Euler que representan la orientación del efector final en el espacio.

2.5.1. Jacobiano inverso

El Jacobiano inverso permite obtener velocidades articulares a partir de una velocidad deseada del efector final [38]. En su forma básica se expresa como:

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1}\,\dot{\mathbf{p}} \tag{2.37}$$

Sin embargo, no siempre es posible calcular $\mathbf{J}^{-1}$ (por ejemplo, en singularidades o cuando $\mathbf{J}$ no es cuadrada), por lo que suele emplearse la pseudoinversa.

Aspectos más importantes

- **Existencia del inverso:** $\mathbf{J}^{-1}$ existe únicamente si $\mathbf{J}$ es cuadrada y $\det(\mathbf{J}) \neq 0$; cerca de singularidades, el cálculo del inverso puede amplificar errores y provocar comandos articulares muy grandes.

- **Pseudoinversa (Moore–Penrose):** cuando $\mathbf{J}$ no es invertible o no es cuadrada, se usa $\mathbf{J}^{\dagger}$ para obtener una solución de mínimos cuadrados:

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{\dagger}\,\dot{\mathbf{p}} \tag{2.38}$$

2.5.2. Singularidades

Para encontrar las singularidades de un robot manipulador utilizando el Jacobiano analítico, se utiliza el concepto de determinante del Jacobiano. Las singularidades ocurren cuando el determinante del Jacobiano se vuelve cero o muy cercano a cero, lo que indica que el robot está perdiendo la capacidad de moverse en ciertas direcciones.

El determinante del Jacobiano analítico (J_a) se utiliza para determinar las singularidades de un robot manipulador. Una singularidad es una configuración de las articulaciones en la cual el robot pierde la capacidad de moverse en ciertas direcciones en el espacio de trabajo. Esto significa que no es posible obtener una solución única para las velocidades del efector final en esas direcciones.

La singularidad ocurre cuando el determinante del Jacobiano es igual a cero o muy cercano a cero [4]. Matemáticamente, se puede expresar como:

$$\det(J_a) = 0$$

Donde: $\det(J_a)$ es el determinante del Jacobiano analítico.

Para encontrar las singularidades, se debe resolver la ecuación anterior para las coordenadas articulares $(q_1, q_2, \ldots, q_n)$ del robot manipulador. Las soluciones de esta ecuación representarán las configuraciones singulares en las cuales el robot pierde su movilidad.

Es importante tener en cuenta que las singularidades pueden tener implicaciones significativas en la planificación y el control del movimiento del robot. Cuando se acerca a una singularidad, el robot puede volverse inestable o tener movimientos no deseados. Por lo tanto, es importante identificar y gestionar las singularidades en la programación y el control del robot manipulador.

El análisis de singularidades es una parte crucial de la cinemática y el control de robots manipuladores y es fundamental para garantizar un funcionamiento seguro y eficiente del robot en su espacio de trabajo.

2.6. Simulador 3D

Con el surgimiento de diversas librerías gráficas se puede desarrollar interfaces interactivas en tres dimensiones de una forma sencilla sin necesidad de extensos conocimientos de programación. El modelado en 3D consiste en el desarrollo de una representación visual de un objeto o conjunto de objetos mediante un ordenador para observar el modelo final desde cualquier ángulo [22].

La primera necesidad para el desarrollo de la simulación es el dibujar el robot manipulador en tres dimensiones usando SolidWorks® para posteriormente ser exportado a Coppelia Sim.

La importancia del modelado en 3D se encuentra en la facilidad con la cual el usuario puede ver el comportamiento dinámico del robot a estudiar antes de implementarlo físicamente, logrando detectar anomalías o mejoras que serían difíciles de observar con solo datos numéricos o gráficas.

Para completar el desarrollo del simulador 3D se requiere tener conocimientos sobre los siguientes aspectos.

2.7. CoppeliaSim

CoppeliaSim (anteriormente V-REP) es un simulador orientado al desarrollo y validación de algoritmos de robótica. Permite construir escenas con robots, sensores y objetos, y ejecutar simulaciones con motor físico para evaluar el comportamiento del sistema antes de implementarlo en un entorno real [39]. Su principal ventaja es que integra modelado, sensado y control en un mismo entorno, lo que facilita el prototipado rápido de aplicaciones robóticas.

2.7.1. Descripción general

Una simulación en CoppeliaSim se organiza en una *escena*, la cual típicamente contiene:

- **Modelos robóticos**: manipuladores, robots móviles y mecanismos articulados.
- **Sensores**: cámaras, sensores de proximidad y otros dispositivos de medición virtual.
- **Entorno**: objetos de interacción (piezas) y obstáculos (mesas, paredes, etc.).
- **Lógica de control**: scripts internos y/o controladores externos que envían comandos y reciben datos.

2.7.2. Puntos importantes para desarrollar simulaciones robóticas

- **Dinámica y parámetros físicos**: definir correctamente masa, inercia, fricción y colisiones es clave para obtener un comportamiento realista y evitar inestabilidades numéricas.
- **Articulaciones y modos de actuación**: las juntas pueden configurarse para control en posición, velocidad o esfuerzo/torque; además conviene establecer límites, velocidades máximas y parámetros de control.
- **Cinemática directa e inversa (IK)**: es útil configurar cadenas cinemáticas e IK para tareas de seguimiento de trayectorias, posicionamiento y manipulación.
- **Sensado virtual (visión y otros)**: configurar resolución, frecuencia de muestreo y (si aplica) ruido permite evaluar algoritmos de percepción en condiciones cercanas al caso real.

- **Scripting y automatización**: mediante scripts (por ejemplo, en Lua) se pueden automatizar experimentos, generar trayectorias, reiniciar escenas y registrar mediciones.
- **Interfaz con programas externos**: la comunicación con controladores externos (por ejemplo, Python/C++/MATLAB o middleware robótico) permite ejecutar el algoritmo fuera del simulador y acercarse a una arquitectura real.
- **Registro de datos y reproducibilidad**: fijar condiciones iniciales, documentar parámetros de la escena y guardar variables (trayectorias, lecturas de sensores, errores) facilita comparar controladores y validar mejoras.

2.8. Etapa de Control

En la etapa de control debe diseñarse una ley de control capaz de reducir los errores a cero, o a un valor mínimo que se considere tolerable. Ya que en el presente trabajo se pretende comparar los resultados del error de posición al utilizar un controlador clásico, en esta sección se utiliza como ejemplo el PID, PID+G y modos deslizantes integral (ISMC).

2.9. Técnicas de control a comparar

A continuación se presentan las principales características de las técnicas de control que se aplicarán para minimizar el error de posición de un robot manipulador de 3 grados de libertad. Aunque se considera aplicar un controlador clásico, en el siguiente apartado se revisan las características del controlador PID, ya que según [40] es el controlador más usado en la industria debido al buen desempeño que presenta. Posteriormente se analiza la planitud diferencial, ya que es una técnica que no ha sido aplicada ampliamente en robots manipuladores, por lo cual resulta interesante indagar sobre las posibles ventajas que se podrían obtener tras su aplicación en los robots antes mencionados.

2.9.1. PID

De acuerdo con [40] el control proporcional integral derivativo (PID) es la estrategia de control más utilizada en la última década. Se estima que más del 90 % de los lazos de control emplean control PID, a menudo con la ganancia derivativa configurada a cero (control PI).

desde los años cincuenta, una gran parte del esfuerzo académico e industrial se ha centrado en mejorar el control PID, principalmente en las áreas de reglas de ajuste, esquemas de identificación y técnicas de adaptación.

Los tres términos de un controlador PID cumplen tres requisitos comunes de la mayoría de los problemas de control. El término integral produce un error nulo de estado estable en el seguimiento de un punto de ajuste constante, un resultado comúnmente explicado en términos del modelo interno y demostrado utilizando el teorema del valor final. El control también permite el rechazo completo de perturbaciones constantes. Mientras que el control integral filtra el ruido de frecuencia más alta , es lento en respuesta al error de de corriente. Por otro lado, el término proporcional responde inmediatamente al error actual, pero normalmente no puede lograr la precisión del punto de ajuste deseado sin una ganancia inaceptablemente grande.

Para plantas con tiempo muerto significativo, los efectos de acciones de control previas están pobremente representados en el error actual. Esta situación puede dar lugar a grandes errores transitorios cuando se utiliza control PI. La acción derivativa combate este problema al basar una parte del control en una predicción de errores futuros.

2.9.2. PID con compensación de gravedad -PID+G

El control PID con compensación de gravedad (en adelante, PID+G) es una estrategia ampliamente utilizada en manipuladores, donde el controlador PID se encarga de corregir el error de seguimiento y un término adicional compensa el efecto de la gravedad sobre las articulaciones. La idea central es separar el problema en dos partes: (i) **compensar** una perturbación dominante y conocida (la gravedad) y (ii) **regular** el error residual mediante un controlador de realimentación [41].

Planteamiento del control en espacio articular

Para un robot manipulador, el modelo dinámico típico en coordenadas articulares se presenta en la ec. 2.39:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau \tag{2.39}$$

donde $q \in \mathbb{R}^n$ es el vector de posiciones articulares, $M(q)$ es la matriz de inercia, $C(q,\dot{q})\dot{q}$

agrupa los términos centrífugos y de Coriolis, $g(q)$ es el vector de torques debidos a la gravedad, $f(\dot{q})$ es el vector de fricción y τ es el vector de torques aplicados [41].

En el caso del presente trabajo ($n = 3$), el objetivo es que el robot siga una referencia articular $q_d(t)$. Para ello se define el error en la ec. 2.40:

$$e(t) = q_d(t) - q(t) \tag{2.40}$$

y sus derivadas $\dot{e}(t)$ y $\int e(t)dt$.

Ley de control PID + G

Una forma común de control PID+G consiste en aplicar el torque de control de la ec. 2.41:

$$\tau = K_p e + K_d \dot{e} + K_i \int e\, dt + g(q) \tag{2.41}$$

donde K_p, K_d y K_i son matrices (usualmente diagonales) de ganancias proporcional, derivativa e integral, respectivamente. El término $g(q)$ actúa como una **prealimentación** (*feedforward*) que cancela la componente gravitacional del modelo.

Interpretación y ventajas

El término $g(q)$ reduce el esfuerzo que debe realizar el PID para sostener el brazo frente a la gravedad, lo cual suele traducirse en:

- menor error estacionario y menor demanda de acción integral,
- reducción de sobreimpulsos asociados a grandes ganancias proporcional/integral,
- mejor desempeño en trayectorias lentas o movimientos donde la gravedad domina.

En términos prácticos, si el modelo de gravedad es razonablemente correcto, el controlador opera sobre un sistema con menores perturbaciones.

2.9.3. Control por Modos Deslizantes Integral -ISMC

Fundamentación

El control por modos deslizantes integral (en adelante, ISMC) es una extensión del control por modos deslizantes clásico (SMC) desarrollado por Utkin [42]. Su objetivo principal es garantizar invariancia ante incertidumbres desde el instante inicial, eliminando la fase de alcanzabilidad característica del SMC convencional.

En el SMC clásico, la robustez se garantiza únicamente después de que el sistema alcanza la superficie de deslizamiento. En contraste, el ISMC construye una superficie que contiene explícitamente la condición inicial del sistema, asegurando que el movimiento deslizante exista desde $t = 0$ [43].

Modelo del sistema

Considérese el siguiente sistema no lineal incierto:

$$\dot{x}(t) = f(x,t) + B(x,t)u(t) + d(x,t) \tag{2.42}$$

donde:

- $x \in \mathbb{R}^n$ es el vector de estado,
- $u \in \mathbb{R}^m$ es la entrada de control,
- $d(x,t)$ representa perturbaciones acotadas.

Se asume que las perturbaciones satisfacen:

$$\|d(x,t)\| \leq \Delta \tag{2.43}$$

Superficie de deslizamiento integral

En ISMC, la superficie de deslizamiento se define como:

$$\sigma(t) = Gx(t) - Gx(0) - \int_0^t G\big(f(x(\tau),\tau) + B(x(\tau),\tau)u_0(\tau)\big)\, d\tau \tag{2.44}$$

donde:

- $G \in \mathbb{R}^{m \times n}$ es una matriz de diseño,
- $u_0(t)$ es un control nominal estabilizante.

Esta construcción garantiza:

$$\sigma(0) = 0 \tag{2.45}$$

Por lo tanto, el sistema inicia directamente sobre la superficie de deslizamiento, eliminando la fase de alcanzabilidad [44].

Ley de control

La ley de control ISMC se compone de dos términos:

$$u(t) = u_0(t) + u_{sw}(t) \tag{2.46}$$

donde el término discontinuo se define como:

$$u_{sw}(t) = -K \operatorname{sign}(\sigma(t)) \tag{2.47}$$

con $K > \Delta$ para satisfacer la condición de alcanzabilidad.
Considérese la función candidata de Lyapunov:

$$V = \frac{1}{2}\sigma^\top \sigma \tag{2.48}$$

Si se garantiza:

$$\dot{V} = \sigma^\top \dot{\sigma} \leq -\eta \|\sigma\| \tag{2.49}$$

con $\eta > 0$, entonces la estabilidad en el sentido de Lyapunov queda asegurada [43].

Capítulo 3

Modelo del brazo robótico

En este capítulo, se muestra el modelo 3D del brazo manipulador, el cuál se diseñó utilizando SolidWorks®, también se describen los pasos necesarios para obtener las ecuaciones correspondientes a la cinemática directa e inversa del robot manipulador, tomando como base los parámetros de Denavit-Hartenberg. Además, se muestra el procedimiento a través del cual se obtienen las ecuaciones del modelo dinámico del sistema, haciendo uso de las ecuaciones de Euler-Lagrange.

El modelo 3D del brazo manipulador se diseñó en el software SolidWorks®versión 2020, los resultados que se obtuvieron se muestran en la Figura 3.1.

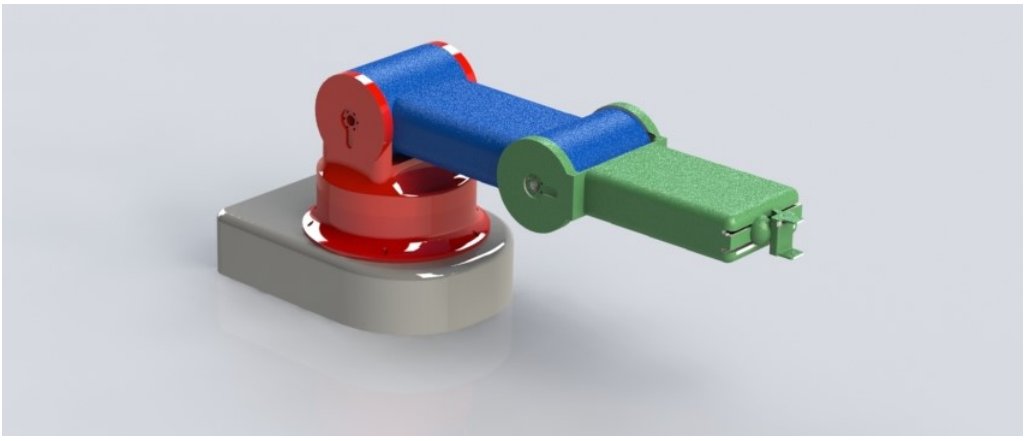

Figura 3.1: Modelo 3D obtenido en SolidWorks®.

Deben considerarse las especificaciones del software y el equipo de cómputo utilizados para realizar las simulaciones; se empleó Matlab 2020 y Coppelia 4.10 en una PC con 1 Tb

de disco duro, 16 Gb de memoria RAM y tarjeta gráfica RTX 4060.

3.1. Modelo cinemático

En esta sección se presentan la cinemática directa e inversa del robot manipulador, cuyo modelo 3D se muestra en la Figura 3.1.

Es importante recordar que el robot de interés cuenta con 3 juntas revolutas; cada junta o articulación cuenta con un grado de libertad, el cual se refiere al ángulo de rotación θ. Además está integrado por 3 eslabones denominados L1, L2 y L3, cuyas dimensiones se expresan en metros.

La Figura 3.2 muestra el diagrama de las articulaciones del robot de interés y sus respectivos marcos de referencia, los cuales se utilizaron para determinar la tabla de parámetros Denavit-Hartenberg.

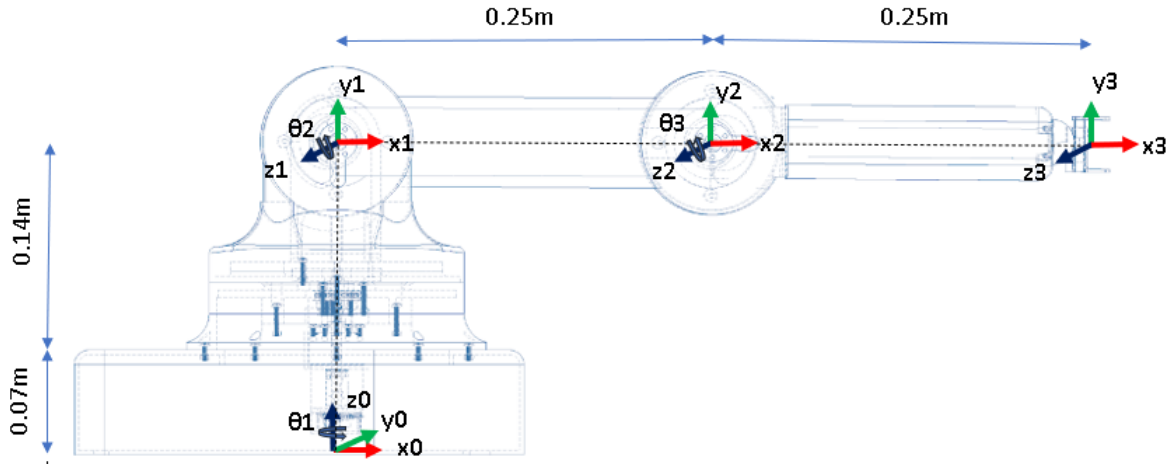

Figura 3.2: Marcos de referencia asociados a las articulaciones del robot

3.1.1. Cinemática directa

A continuación se presenta el modelo cinemático directo del robot manipulador de 3 grados de libertad. Utilizando la convención de Denavit-Hartenberg (D-H).

Para obtener la tabla de parámetros de DH, es importante establecer una convención consistente para asignar los sistemas de coordenadas (Punto de inicio o Home). La figura 3.3 muestra el Home seleccionado y la tabla de parámetros de DH se muestra en la tabla 3.1.

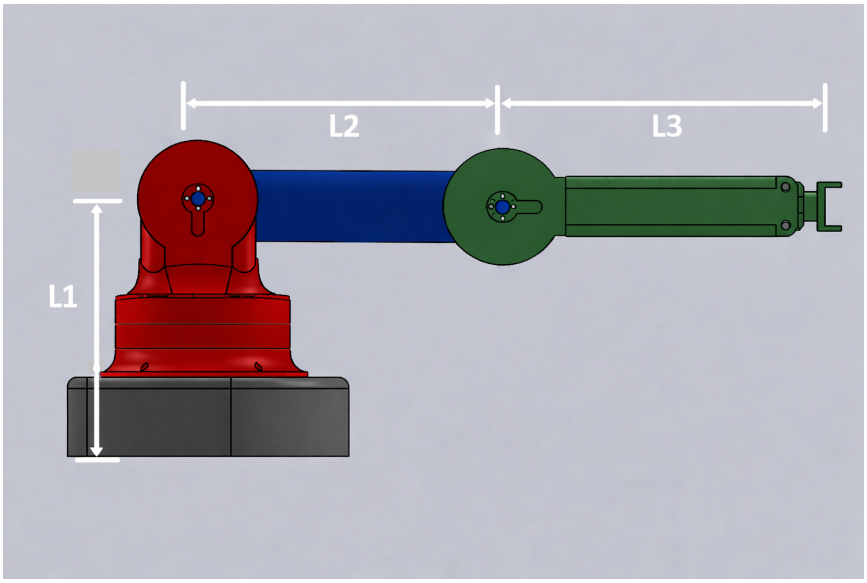

Figura 3.3: Posición inicial del robot manipulador para la obtención del modelo cinemático.

Tabla 3.1: Parámetros Denavit-Hartenberg

Articulación i	$\theta_i(z)$	$d_i(z)$	$a_i(x)$	$\alpha_i(x)$
1	θ_1	L_1	0	90
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0

En términos generales, la Tabla 3.1 muestra las rotaciones y desplazamientos a través de los ejes z y x, que serán necesarios para alcanzar cada uno de los marcos de referencia asociados a las articulaciones del robot.

La matriz de transformación homogénea para la primera articulación se obtiene al realizar el producto matricial de la ec. 2.29 y al sustituir los parámetros de la primera fila de la Tabla 3.1 se obtiene el siguiente resultado:

$$A_1^0 = \begin{pmatrix} cos(\theta_1) & 0 & sin(\theta_1) & 0 \\ sin(\theta_1) & 0 & -cos(\theta_1) & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

El resultado obtenido para la matriz de transformación homogénea de la segunda articu-

lación es el siguiente:

$$A_2^1 = \begin{pmatrix} cos(\theta_2) & -sin(\theta_2) & 0 & L_2cos(\theta_2) \\ sin(\theta_2) & cos(\theta_2) & 0 & L_2sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

La matriz de transformación homogénea correspondiente a la tercera articulación se presenta enseguida:

$$A_3^2 = \begin{pmatrix} cos(\theta_3) & -sin(\theta_3) & 0 & L_3cos(\theta_3) \\ sin(\theta_3) & cos(\theta_3) & 0 & L_3sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

Como parte final de la cinemática directa, se realiza el producto matricial de las transformaciones homogéneas correspondientes a las 3 articulaciones del robot y los valores obtenidos en las 3 primeras filas de la cuarta columna corresponderán a las coordenadas halladas al introducir determinados ángulos para la rotación de las articulaciones.

El producto matricial de las transformaciones homogéneas es el siguiente:

$$T_3^0 = A_1^0 A_2^1 A_3^2 \tag{3.4}$$

Para comprobar la cinemática directa se programó el código 3.1 Matlab® para obtener la matriz de la ec. 3.5, dicho código se presenta a continuación:

Código 3.1: Cinemática directa en Matlab

```
1  MATLAB
2  %Autor: Karina Ramirez Lopez
3  %Tema:  Cinematica Directa
4  %Codigo: Obtencion de la matriz de Transformacion Homogenea
5  syms L1 L2 L3 th1 th2 th3
6  %Matriz de tranformacion homogena A0_1
7      A1=[cos(th1) 0 sin(th1) 0;
8          sin(th1) 0 -cos(th1) 0;
9          0 1 0 L1;
```

```
10          0 0 0 1];
11
12      A2=[cos(th2) -sin(th2) 0 L2*cos(th2);
13          sin(th2) cos(th2) 0 L2*sin(th2);
14          0 0 1 0;
15          0 0 0 1];
16
17      A3=[cos(th3) -sin(th3) 0 L3*cos(th3);
18          sin(th3) cos(th3) 0 L3*sin(th3);
19          0 0 1 0;
20          0 0 0 1];
21
22  T=A1*A2*A3; %Matriz de Transformacion total
23  T=simplify(T)    % Resultado simplificado
24  \label{Cdi}
```

El resultado del producto matricial de la ec. 3.4 se muestra en la ec. 3.5

$$T_3^0 = \begin{pmatrix} c_{23}c_1 & -s_{23}c_1 & s_1 & c_1(L_3c_{23} + L_2c_2) \\ c_{23}s_1 & -s_{23}s_1 & -c_1 & s_1(L_3c_{23} + L_2c_2) \\ s_{23} & c_{23} & 0 & L_3s_{23} + L_2s_2 + L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

Donde se utilizan las expresiones $c_1 = cos(\theta_1)$, $c_2 = cos(\theta_2)$, $s_1 = sin(\theta_1)$, $s_2 = sin(\theta_2)$, $c_{23} = cos(\theta_2 + \theta_3)$ y $s_{23} = sin(\theta_2 + \theta_3)$, para reducir la ecuación.

Para comprobar el resultado, se ejecuta el código para la obtención de la matriz de transformación homogénea en Matlab® y el resultado se muestra en la Figura 3.4.

```
>> Matriz_de_Transformacion_Homogenea

T =

[ cos(th2 + th3)*cos(th1), -sin(th2 + th3)*cos(th1),  sin(th1), cos(th1)*(L3*cos(th2 + th3) + L2*cos(th2))]
[ cos(th2 + th3)*sin(th1), -sin(th2 + th3)*sin(th1), -cos(th1), sin(th1)*(L3*cos(th2 + th3) + L2*cos(th2))]
[       sin(th2 + th3),         cos(th2 + th3),       0,      L1 + L3*sin(th2 + th3) + L2*sin(th2)]
[                    0,                      0,       0,                                           1]
```

Figura 3.4: Comprobación de la cinemática directa.

3.1.2. Cinemática inversa

Para hallar la cinemática inversa, en este caso se realizará el procedimiento denominado método geométrico, para el cual se tomará como base la Figura 3.5.

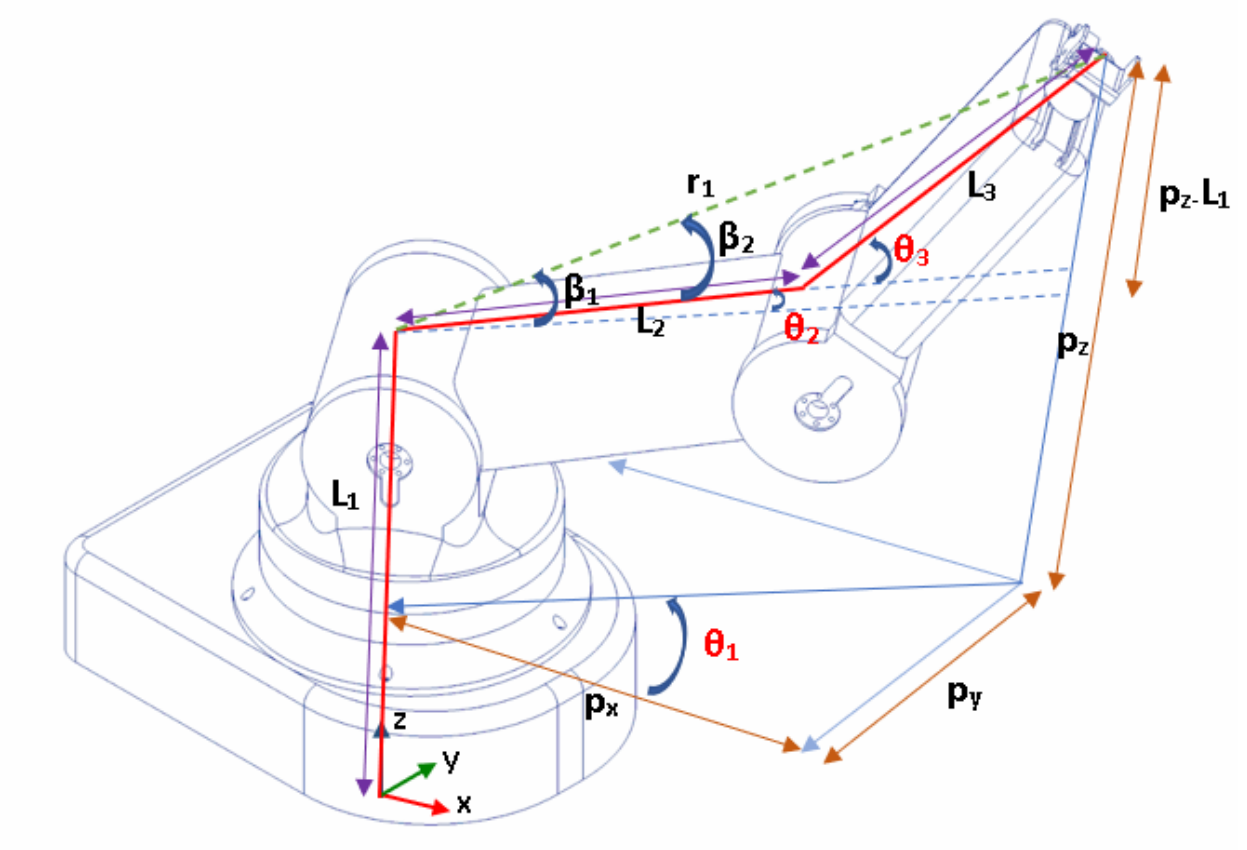

Figura 3.5: Cinemática inversa por el método geométrico

En la Figura 3.5 se coloca al robot de forma que los ángulos de giro de las articulaciones puedan obtenerse a partir de relaciones geométricas e igualdades trigonométricas. Se comienza por las relaciones correspondientes a rp, con las cuales será posible hallar θ_1. Todas las ecuaciones correspondientes a la cinemática inversa se desarrollan a continuación:

$$r_p = \sqrt{p_x^2 + p_y^2}$$

$$p_x = r_p \cos(\theta_1)$$

$$p_y = r_p \operatorname{sen}(\theta_1)$$

por lo tanto, θ_1 estará dado por:

$$\theta_1 = \arctan(py/px) \tag{3.6}$$

$$r_1 = \sqrt{(p_z - L_1)^2 + r_p^2} \tag{3.7}$$

Posteriormente se tiene que para r_1, las ecuaciones correspondientes son:

$$r_1^2 = L_2^2 + L_3^2 - 2L_2L_3\cos(180 - \theta_3) \tag{3.8}$$

tomando en cuenta que:

$$\cos(180 - \theta_3) = -\cos(\theta_3) \tag{3.9}$$

Y sustituyendo (3.9) en (3.8) se tiene:

$$r_1^2 = L_2^2 + L_3^2 + 2L_2L_3\cos(\theta_3)$$

a partir de lo anterior se tiene:

$$\cos(\theta_3) = \frac{r_1^2 - L_2^2 - L_3^2}{2L_2L_3} = M \tag{3.10}$$

por lo cual, θ_3 se define como:

$$\theta_3 = \arccos(M) \tag{3.11}$$

Posteriormente, se parte de las relaciones correspondientes a β_1 para poder hallar a θ_2.

$$\tan(\beta_1) = \frac{P_z - L_1}{r_p} \tag{3.12}$$

$$\beta_1 = \arctan(\frac{P_z - L_1}{r_p})$$

$$\beta_1 = \beta_2 + \theta_2$$

dado que se busca hallar θ_2, una vez que se cuenta con β_1, es necesario hallar β_2.

$$L_3^2 = L_2^2 + r_1^2 - 2L_2r_1\cos(\beta_2)$$

$$\cos(\beta_2) = \frac{L_3^2 - L_2^2 - r_1^2}{-2L_2 r_1} = M_1 \tag{3.13}$$

al despejar β_2 se tiene que:

$$\beta_2 = \arccos(M_1) \tag{3.14}$$

Finalmente se tiene que θ_2 se obtiene a partir de la diferencia entre β_1 y β_2:

$$\theta_2 = \beta_1 - \beta_2 \tag{3.15}$$

En resumen, las relaciones correspondientes a la cinemática inversa del robot manipulador de 3 grados de libertad serán dadas por las ecs. 3.6, 3.15 y 3.11. Para comprobar las ecuaciones obtenidas para la cinemática inversa del robot manipulador, se programó el código 3.2 en Matlab® .

Código 3.2: Cinemática inversa en Matlab

```
MATLAB
%Cinematica inversa
    L1=210;
    L2=250;
    L3=250;

    px=250;
    py=0;
    pz=420;

    th1=atan2d(py,px);
    rp=sqrt(px^2 + py^2);
    r=sqrt(px^2 + py^2 + pz^2);
    r1=sqrt(rp^2 + (pz-L1)^2);
    m=(r1^2 -L2^2-L3^2)/(2*L2*L3);
    c=sqrt(1-m^2);

    th3=acosd(m);
    a=(pz-L1);
    b1=atan2d(a,rp);
    m1=(-(L3)^2 + (L2^2) + (r1^2))/(2*L2*r1);
    b=sqrt(1-m1^2);
```

```
23      th2=b1-acosd(m1);
```

En la Figura 3.6 se muestran los resultados:

```
>> inversa
th1 =
   0
th3 =
   90
th2 =
   0
```

Figura 3.6: Resultados en Matlab®.

Jacobiano del manipulador

En las secciones anteriores se dedujeron las ecuaciones correspondientes a la cinemática directa e inversa del robot manipulador. En esta sección se determinarán las relaciones existentes entre las velocidades lineal y angular del efector final y las velocidades articulares. La velocidad articular y la velocidad del efector final están relacionadas a través de la matriz jacobiana del manipulador, la cual es una función de la configuración del robot.

Cada elemento del Jacobiano se calcula tomando la derivada parcial de la coordenada correspondiente del efector final con respecto a una coordenada articular específica. El resultado es una matriz $6 \times n$ que relaciona las velocidades de las articulaciones con las velocidades lineales y angulares del efector final en el espacio de trabajo.

Al obtener las derivadas correspondientes, el jacobiano analítico se forma como se muestra a continuación:

$$J_a = \begin{pmatrix} -S_1(L_3C_{23} + L_2C_2) & -C_1(L_3S_{23} + L_2S_2) & -L_3C_1S_{23} \\ -C_1(L_3C_{23} + L_2C_2) & -S_1(L_3S_{23} + L_2S_2) & -L_3S_1S_{23} \\ 0 & L_2C_2 + L_3C_{23} & L_3C_{23} \end{pmatrix} \tag{3.16}$$

Al calcular el determinante del jacobiano analítico se obtiene la siguiente ecuación, la cual se iguala a 0 para así obtener las singularidades.

$$\det(J_a) = -S_1(L_3C_{23} + L_2C_2)[-S_1(L_3S_{23} + L_2S_2)(L_3C_{23}) + L_3S_1S_{23}(L_2C_2 + L_3C_{23})]$$

$$+ \mathrm{C}_1(L_3S_{23}+L_2S_2)[C_1(L_3C_{23}+L_2C_2)L_3C_{23}] - L_3C_1S_{23}[C_1(L_3C_{23}+L_2C_2)(L_2C_2+L_3C_{23})]$$

(3.17)

Al resolver la ecuación 3.1.2 se obtienen las singularidades para θ_2 y θ_3 como se muestra a continuación:

$$\theta_2 = tan^{-1}[\frac{(\frac{L2}{L3} + cos(\theta_3))}{sin(\theta_3)}] \tag{3.18}$$

$$\theta_3 = 0, \pi \tag{3.19}$$

3.2. Modelo dinámico

3.2.1. Ecuaciones de Euler-Lagrange

En el presente trabajo se empean las ecuaciones de movimiento de Lagrange, debido a sus consideraciones puramente energéticas, lo cual facilita el análisis para cualquier sistema mecánico. Para el uso de las ecuaciones de Lagrange es necesario seguir tres etapas:

a) Cálculo de la energía cinética y potencial..

b) Cálculo del lagrangiano.

c) Resolver las ecuaciones para cada grado de libertad.

Para el caso del robot manipulador de interés, una vez que se cuenta con el modelo cinemático, se procede a obtener el modelo dinámico a partir del método de Euler-Lagrange, siguiendo las cuatro etapas listadas anteriormente.

Para el cálculo de la energía cinética se tiene que esta puede ser tanto rotacional como traslacional y esta forma de energía puede estar en función tanto de la posición como de la velocidad, es decir, $K(q(t), \dot{q}(t))$.

Con respecto a la energía potencial, esta se debe a fuerzas conservativas como las fuerzas ejercidas por resortes y por la gravedad, esta energía se expresa en términos de la posición y se expresa como $U(q(t))$.

Posteriormente, el lagrangiano está definido como la diferencia entre la energía cinética y la energía potencial:

$$L(q(t), \dot{q}(t)) = K(q(t), \dot{q}(t)) - U(q(t)) \tag{3.20}$$

Finalmente, las ecuaciones de movimiento para un sistema de n grados de libertad se expresan como:

$$\frac{d}{dt}\left(\frac{\partial L(q, \dot{q})}{\partial \dot{q}_i}\right) - \frac{\partial L(q, \dot{q})}{\partial q_i} = \tau_i \tag{3.21}$$

Donde τ_i con $i = 1, ..., n$, son las fuerzas o pares ejercidos por los actuadores en cada articulación además de fuerzas no conservativas como la fricción, resistencia al movimiento de un objeto dentro de un fluido y de forma general las que dependen del tiempo o de la velocidad. En este caso, dado que se trabaja con un sistema completamente actuado, se tendrán igual número de ecuaciones dinámicas como de grados de libertad y debe tenerse en cuenta que no se considera el término de fricción.

El diagrama utilizado para obtener el modelo dinámico del brazo robótico se presenta en la Figura 3.7:

3.2.2. Modelado del robot manipulador de 3 grados de libertad

Se obtuvieron las ecuaciones de Euler-Lagrange del sistema mostrado en la Figura 3.3. Por ello, inicialmente se analiza la energía cinética de cada eslabón.

1. Energía cinética: está definida por sus componentes rotacional y traslacional para cada

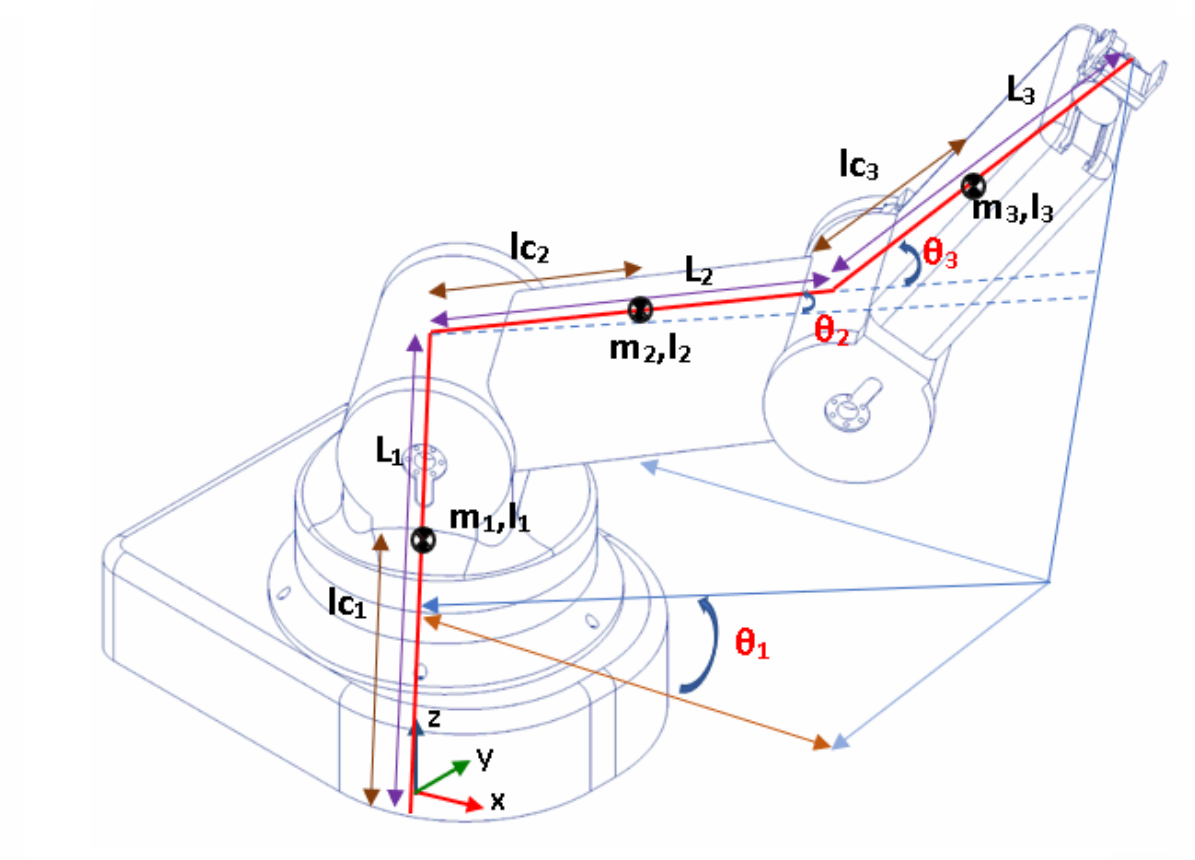

Figura 3.7: Diagrama usado para la obtención del modelo dinámico

eslabón del brazo.

$$K = \frac{1}{2}mv^2 + \frac{1}{2}I\omega^2 \tag{3.22}$$

Donde m es la masa del eslabón, v es la velocidad lineal, I es el momento de inercia y ω es la velocidad rotacional.

La energía cinética total del sistema según la ec. 3.22 está dada por [41]:

$$\begin{aligned} K =& \frac{1}{2}m_2(lc_2^2(\dot{q_1}^2 - \dot{q_1}^2 sin^2(q_2) + \dot{q_2}^2)) + \frac{1}{2}m_2 I_{y2} cos^2(q_1)cos^2(q_2)\dot{q_2}^2 \\ &+ \frac{1}{2}m_3\,[[(cos(q_1)(L_2\dot{q_2}sin(q_2) + lc_3 sin(q_2+q_3)(\dot{q_2}+\dot{q_3})) \\ &+ \dot{q_1}sin(q_1)(lc_3 cos(q_2+q_3) + L_2 cos(q_2))]^2 + [sin(q_1)(L_2\dot{q_2}sin(q_2) \\ &+ lc_3 sin(q_2+q_3)(\dot{q_2}+\dot{q_3})) - \dot{q_1}cos(q_1)(lc_3 cos(q_2+q_3) + L_2 cos(q_2))]^2 \\ &+ (lc_3 cos(q_2+q3)(\dot{q_2}+\dot{q_3}) + L_2\dot{q_2}cos(q_2))^2 + (I_{y3}cos^2(q_1)cos^2(q_3)\dot{q_2}^2))] \end{aligned} \tag{3.23}$$

2. Energía potencial: dado que el robot manipulador almacena energía potencial gravitacional, su ecuación correspondiente es:

$$U = mgh \tag{3.24}$$

Teniendo como resultado que la energía potencial total del sistema es:

$$U = m_1 g lc_1 + m_2 g(L_1 + lc_2 sin(q_2)) + m_3 g(L_1 + L_2 sin(q_2) + lc_3 sin(q2+q3)) \tag{3.25}$$

3. Lagrangiano: se calcula a través de la diferencia entre las ecs. (3.31) y (3.32) como se muestra en (3.26):

$$\begin{aligned} L =& \frac{1}{2}m_2(lc_2^2(\dot{q_1}^2 - \dot{q_1}^2 sin^2(q_2) + \dot{q_2}^2)) + \frac{1}{2}m_2 I_{y2} cos^2(q_1)cos^2(q_2)\dot{q_2}^2 \\ &+ \frac{1}{2}m_3\,[[(cos(q_1)(L_2\dot{q_2}sin(q_2) + lc_3 sin(q_2+q_3)(\dot{q_2}+\dot{q_3})) \\ &+ \dot{q_1}sin(q_1)(lc_3 cos(q_2+q_3) + L_2 cos(q_2))]^2 + [sin(q_1)(L_2\dot{q_2}sin(q_2) \\ &+ lc_3 sin(q_2+q_3)(\dot{q_2}+\dot{q_3})) - \dot{q_1}cos(q_1)(lc_3 cos(q_2+q_3) + L_2 cos(q_2))]^2 \\ &+ (lc_3 cos(q_2+q3)(\dot{q_2}+\dot{q_3}) + L_2\dot{q_2}cos(q_2))^2 + (I_{y3}cos^2(q_1)cos^2(q_3)\dot{q_2}^2))] \\ &- m_1 g lc_1 - m_2 g(L_1 + lc_2 sin(q_2)) - m_3 g(L_1 + L_2 sin(q_2) + lc_3 sin(q2+q3)) \end{aligned} \tag{3.26}$$

4. Ecuaciones de movimiento: El sistema de interés cuenta con 3 grados de libertad y puesto que se trata de un sistema totalmente actuado, se obtienen 3 ecuaciones de movimiento:

$$
\begin{aligned}
\tau_1 &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q_1}}\right) - \frac{\partial L}{\partial q_1} = \\
&= \frac{1}{2}\left[L_2^2 m_3 + lc_2^2 m_2 + lc_3^2 m_3 + L_2^2 m3cos(2q_2) + lc_2^2 m_2 cos(2q_2) + lc_3^2 m_3 cos(2q_2+2q_3)\right. \\
&+ 2L_2 lc_3 m3cos(q_3) + lc_3^2 m_3 cos(2q_2+2q_3) + 2L_2 lc_3 m3cos(q_3) \\
&\left.+ 2L_2 lc_3 m_3 cos(2q_2+q_3)\right]\ddot{q_1} + \frac{1}{2}\left[-2L_2^2 m_3 sin(2q_2)\dot{q_2} - 2lc_2^2 m_2 sin(2q_2)\dot{q_2}\right. \\
&- lc_3^2 m_3 sin(2q_2+2q_3)(2\dot{q_2}+2\dot{q_3}) - 2L_2 lc_3 m_3 sin(q_3)\dot{q_3} \\
&\left.- 2L_2 lc_3 m_3 sin(2q_2+q_3)(2\dot{q_2}+\dot{q_3})\right]\dot{q_1} + \dot{q_2}^2 cos(q_1) sen(q_1)[I_{y2} cos^2(q_2) + I_{y3} cos^2(q_3)]
\end{aligned} \tag{3.27}
$$

$$
\begin{aligned}
\tau_2 &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q_2}}\right) - \frac{\partial L}{\partial q_2} = \\
&= lc_2^2 m_2\ddot{q_2} + lc_3^2 m_3\ddot{q_2} + lc_3^2 m_3\ddot{q_3} + L_2^2 m_3\ddot{q_2} \\
&+ I_{y2}[-2cos(q_1)sin(q_1)cos^2(q_2)\dot{q_1}\dot{q_2} - -2cos(q_2)sin(q_2)cos^2(q_1){\dot{q_2}}^2 \\
&+ cos^2(q_1)cos^2(q_2)\ddot{q_2}] + I_{y3}[-2cos(q_1)sin(q_1)cos^2(q_3)\dot{q_1}\dot{q_2} \\
&- 2cos(q_3)sin(q_3)cos^2(q_1)\dot{q_2}\dot{q_3} + cos^2(q_1)cos^2(q_3)\ddot{q_2}] \\
&+ 2L_2 lc_3 m_3[-sin(q_3)\dot{q_2}\dot{q_3} + cos(q_3)\ddot{q_2}] + L_2 lc_3 m_3[-sin(q_3){\dot{q_3}}^2] + cos(q_3)\ddot{q_3}] \\
&- [-glc_3 m_3 cos(q_2+q_3) - gL_2 m_3 cos(q_2) - glc_2 m_2 cos(q_2) \\
&- \frac{1}{2}(L_2^2 m_3 {\dot{q_1}}^2 sin(2q_2)) - \frac{1}{2}(lc_2^2 m_2 {\dot{q_1}}^2 sin(2q_2)) - \frac{1}{2}(lc_3^2 m_3 {\dot{q_1}}^2 sin(2q_2+2q_3)) \\
&- I_{y2}{\dot{q_2}}^2 cos^2(q_1)cos(q_2)sin(q_2) - L_2 Lc_3 m_3 {\dot{q_1}}^2 sin(2q_2+q_3)
\end{aligned} \tag{3.28}
$$

$$
\begin{aligned}
\tau_3 &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q_3}}\right) - \frac{\partial L}{\partial q_3} = \\
&= lc_3\, m_3\,[lc_2\,\ddot{q_2} + lc_3\,\ddot{q_3} - L_2\, sin(q_3)\,\dot{q_2}\,\dot{q_3} + L_2\, cos(q_3)\,\ddot{q_2}] \\
&- [-g\, lc_3\, m_3\, cos(q_2+q_3) - \frac{1}{2}lc_3^2\, m_3\, {\dot{q_1}}^2\, sin(2q_2+2q_3) - \frac{1}{2}L_2\, lc_3\, m_3\, {\dot{q_1}}^2\, sin(q_3) \\
&- L_2\, lc_3\, m_3\, \dot{q_2}^2\, sin(q_3) - I_{y3}\,\dot{q_2}\, cos^2(q_1)cos(q_3)sin(q_3) - \frac{1}{2}L_2\, lc_3\, m_3\, {\dot{q_1}}^2\, sin(2q_2+q_3) \\
&- L_2\, lc_3\, m_3\,\dot{q_2}\,\dot{q_3}\, sin(q_3)]
\end{aligned} \tag{3.29}
$$

Una vez que se han obtenido las ecuaciones de movimiento del sistema, las mismas pueden

ser expresadas en su forma matricial a través de la siguiente expresión:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{3.30}$$

Donde:
q: coordenadas articulares generalizadas, M(q): matriz de masa o inercia.
$C(q,\dot{q})$: fuerzas centrífugas y de Coriolis, g(q): fuerzas de gravedad.
τ: fuerzas generalizadas
La matriz de inercias queda expresada como se muestra en la ec. 3.31

$$M(q) = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \tag{3.31}$$

Donde los elementos desde m_{11} hasta m_{33} están definidos como:

$m_{11} = \frac{1}{2}\left[L_2^2 m_3 + lc_2^2 m_2 + lc_3^2 m_3 + L_2^2 m3cos(2q_2) + lc_2^2 m_2 cos(2q_2) + lc_3^2 m_3 cos(2q_2 + 2q_3)\right]$
$+2L_2 lc_3 m3cos(q_3) + lc_3^2 m_3 cos(2q_2 + 2q_3) + 2L_2 lc_3 m3cos(q_3) + [2L_2 lc_3 m_3 cos(2q_2 + q_3)]$
$m_{22} = lc_2^2 m_2 + lc_3^2 m_3 + L_2^2 m_3 + I_{y2} cos^2(q_1) cos^2(q_2) + I_{y3} cos^2(q_1) cos^2(q_3) + 2L_2 lc_3 m_3 cos(q3)$
$m_{23} = lc_3^2 m_3 + L_2 lc_3 m_3 cos(q_3)$ $\quad m_{32} = lc_3^2 m_3 + L_2 lc_3 m_3 cos(q_3)$ $\quad m_{33} = lc_3^2 m_3$
$m_{12} = 0$ $\quad m_{13} = 0$ $\quad m_{21} = 0$ $\quad m_{31} = 0$
La matriz de Coriolis se describe en la ec. 3.32:

$$C(q,\dot{q}) = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \tag{3.32}$$

Y sus elementos desde c_{11} hasta c_{33} se expresan como:

$c_{11} = \frac{1}{2}\left[-2L_2^2 m_3 sin(2q_2)\dot{q}_2 - 2lc_2^2 m_2 sin(2q_2)\dot{q}_2 - lc_3^2 m_3 sin(2q_2 + 2q_3)(2\dot{q}_2 + 2\dot{q}_3)\right.$
$\left.-2L_2 lc_3 m_3 sin(q_3)\dot{q}_3 - 2L_2 lc_3 m_3 sin(2q_2 + q_3)(2\dot{q}_2 + \dot{q}_3)\right]$

$c_{12} = cos(q_1) sen(q_1)[I_{y2} cos^2(q_2) + I_{y3} cos^2(q_3)]\dot{q}_2 - L_2^2 m_3 sin(2q_2)\dot{q}_1 - lc_2^2 m_2 sin(2q_2)\dot{q}_1$
$-lc_3^2 m_3 sin(2q_2 + 2q_3)\dot{q}_1 - 2L_2 lc_3 m_3 sin(2q_2 + q_3)\dot{q}_1$

$$c_{13} = -lc_3^2 m_3 sin(2q_2 + 2q_3)\dot{q_1} - L_2 lc_3 m_3 sin(q_3)\dot{q_1} - L_2 lc_3 m_3 sin(2q_2 + q_3)\dot{q_1}$$

$$c_{21} = -2I_{y2} cos(q_1) sin(q_1) cos^2(q_2)\dot{q_2} - 2I_{y3} cos(q_1) sin(q_1) cos^2(q_3)\dot{q_2} + \frac{1}{2} L_2^2 m_3 sin(2q_2)\dot{q_1}$$
$$+\frac{1}{2} lc_2^2 m_2 sin(2q_2)\dot{q_1} + \frac{1}{2} lc_3^2 m_3 sin(2q_2 + 2q_3)\dot{q_1} + \frac{1}{2} L_2 lc_3 m_3 sin(2q_2 + q_3)\dot{q_1}$$

$$c_{22} = -2I_{y2} cos(q_1) sin(q_1) cos^2(q_2)\dot{q_1} - 2I_{y2} cos(q_2) sin(q_2) cos^2(q_1)\dot{q_2} - 2I_{y3} cos(q_1) sin(q_1) cos^2(q_3)\dot{q_1}$$
$$-2\mathrm{I}_{y3} cos(q_3) sin(q_3) cos^2(q_1)\dot{q_3} - 2L_2 lc_3 m_3 sin(q_3)\dot{q_3} + I_{y2} cos^2(q_1) cos(q_2) sin(q_2)\dot{q_2}$$

$$c_{23} = -2I_{y3} cos(q_3) sin(q_3) cos^2(q_1)\dot{q_2} - 2L_2 lc_3 m_3 sin(q_3)\dot{q_2} - L_2 lc_3 m_3 sin(q_3)\dot{q_3}$$

$$c_{31} = \frac{1}{2} lc_3^2 m_3 sin(2q_2 + 2q_3)\dot{q_1} + \frac{1}{2} L_2 lc_3 m_3 sin(q_3)\dot{q_1} + \frac{1}{2} L_2 lc_3 m_3 sin(2q_2 + q_3)\dot{q_1}$$

$$c_{32} = -L_2 lc_3 m_3 sin(q_3)\dot{q_3} + L_2 lc_3 m_3 sin(q_3)\dot{q_3} + L_2 lc_3 m_3 sin(q_3)\dot{q_2} + I_{y3} cos^2(q_1) cos(q_3) sin(q_3)\dot{q_2}$$

$$c_{33} = -L_2 lc_3 m_3 sin(q_3)\dot{q_2} + L_2 lc_3 m_3 sin(q_3)\dot{q_2} = 0$$

Finalmente, el vector de gravedad se describe en la ec. 3.33

$$g(q) = \begin{pmatrix} 0 \\ m_2 g lc_2 cos(q_2) + m_3 g[L_2 cos(q_2) + lc_3 cos(q_2 + q_3)] \\ m_3 g lc_3 cos(q_2 + q_3) \end{pmatrix} \tag{3.33}$$

Y los parámetros del modelo dinámico son los siguientes:

Eslabón 1	**Eslabón 2**	**Eslabón 3**
L1=0.1421 m	L2=0.25 m	L3=0.25 m
lc1=0.095592	lc2 = 0.12228	lc3 = 0.10860
m1=0.924702	m2=0.448505	m3= 0.7157
I1= 0.003325	I2= 0.014271	I3= 0.002046

Tabla 3.2: Tabla de parámetros del modelo dinámico.

Donde L_i es la longitud total de cada eslabón, lc_i es la distancia a cada centro de masa, m_i es la masa de cada eslabón, además I_i hace referencia a cada momento de inercia.

3.3. Control PID+G del brazo manipulador

La ley de control proporcional integral derivativo más compensación de gravedad se muestra en la ec. 3.34.

$$\tau = k_p e(t) + k_i \int_0^t e(\lambda) \cdot d\lambda + k_d \frac{de}{dt} + g(q) \tag{3.34}$$

El control PID+G consiste de un término proporcional al error de posición $k_p\tilde{q}$ más un término de inyección de amortiguamiento a través de la velocidad de movimiento denominado acción de control derivativo $k_v\dot{q}$ el cual tiene el efecto de un amortiguador, absorbiendo los sobreimpulsos que tenga la respuesta transitoria, eliminando las oscilaciones y picos, además, se utiliza la acción de control integral, la cual permite eliminar el error en estado estacionario. Para sistemas mecánicos que se mueven en un plano vertical o en general en el espacio tridimensional existe el efecto gravitacional por lo que se requiere realizar la compensación de este fenómeno g(q).

3.3.1. Implementación de la Ley de Control

Para llevar a cabo la implementación de la ley de control, se utilizaron 5 bloques principales en Simulink®, en la Figura 3.9 se muestra el diagrama desarrollado en el cual se implementa el controlador PID + G. La Figura 3.9 consta del bloque para generación de trayectorias deseadas en coordenadas cartesianas, el área para convertir las coordenadas cartesianas a articulares, el bloque con la ley de control, además del bloque que contiene el modelo de SimMechanics®, el cual fue exportado desde SolidWorks® sobre el cual se aplica el controlador y finalmente una sección de señales donde se visualizan las trayectorias deseadas y las alcanzadas por el robot, además del comportamiento de las entradas de control.

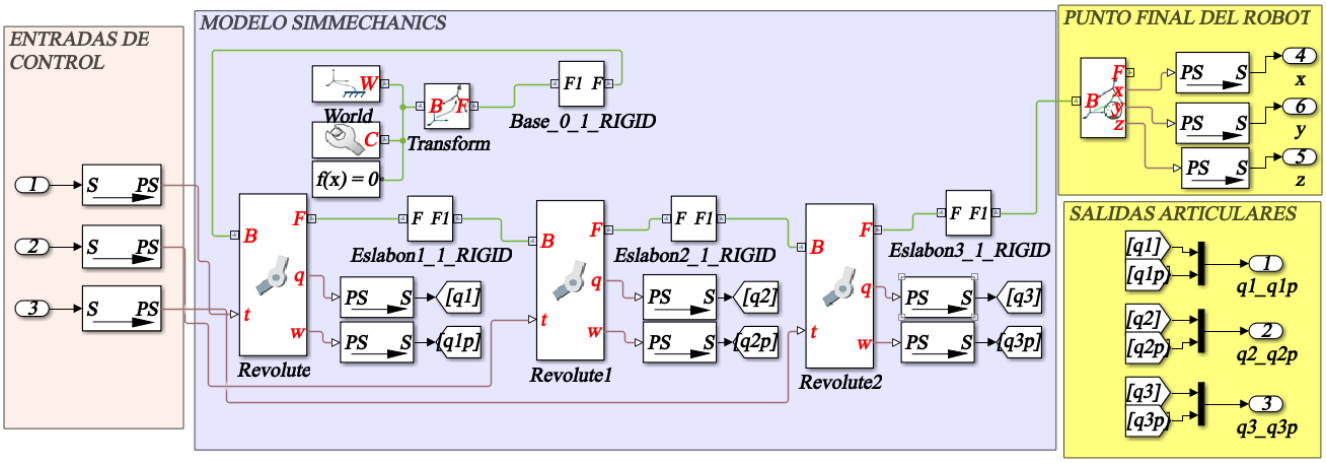

Figura 3.8: Modelo del robot exportado a SimMechanics®

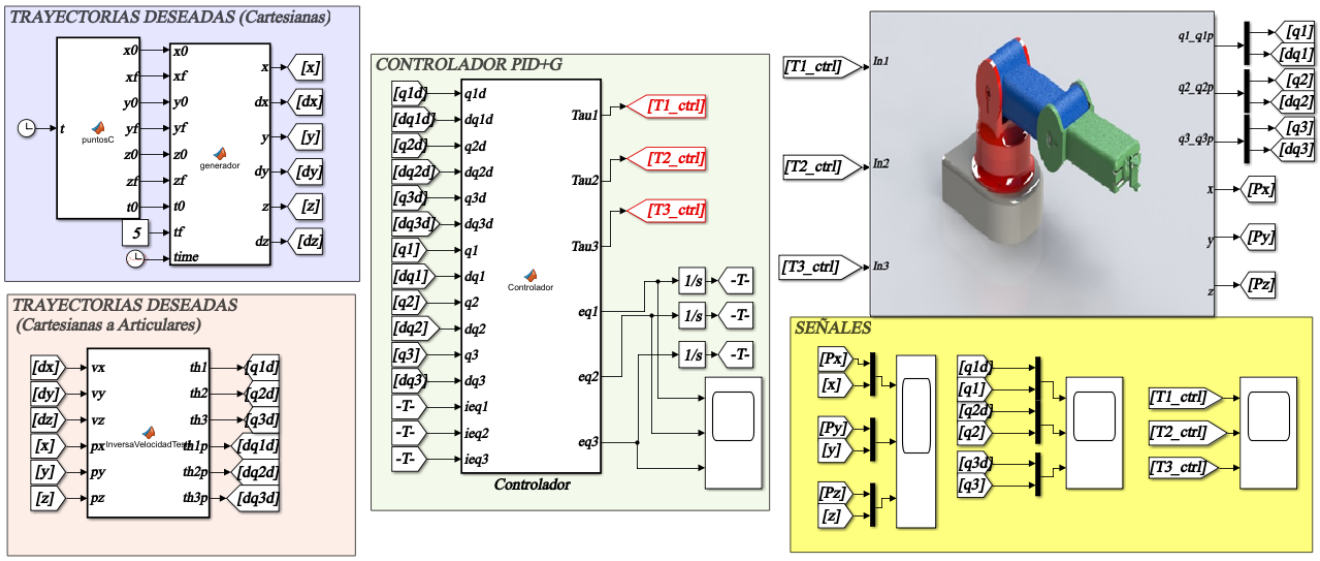

Figura 3.9: Modelo del control del robot en Simulink®

3.3.2. Resultados de la simulación

Esta sección muestra el comportamiento del robot al implementar el controlador PID+G.

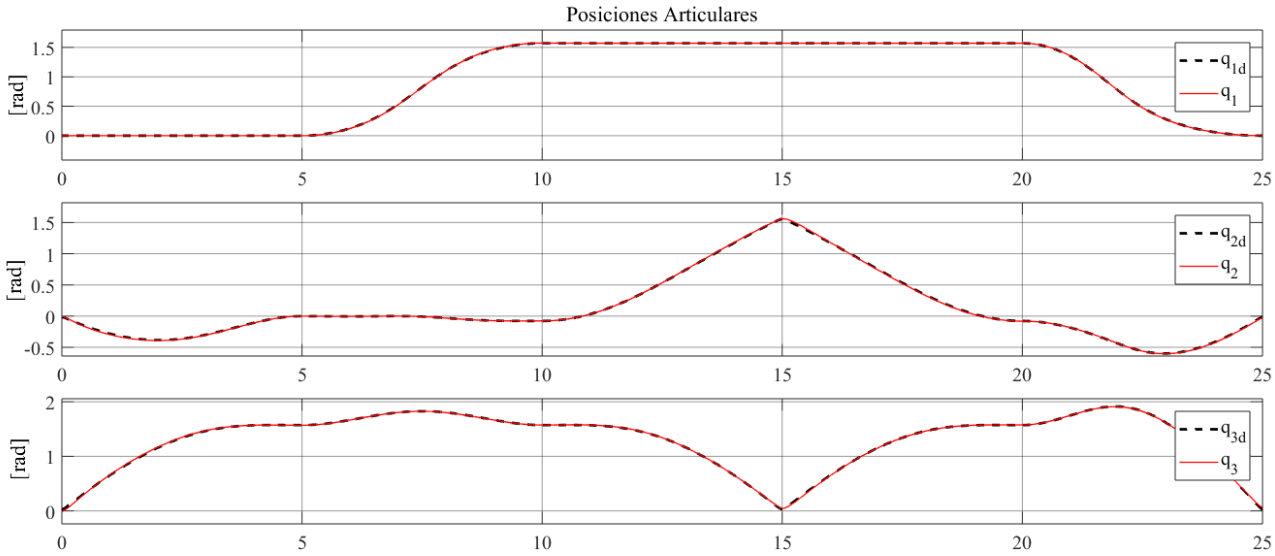

Figura 3.10: Posiciones articulares

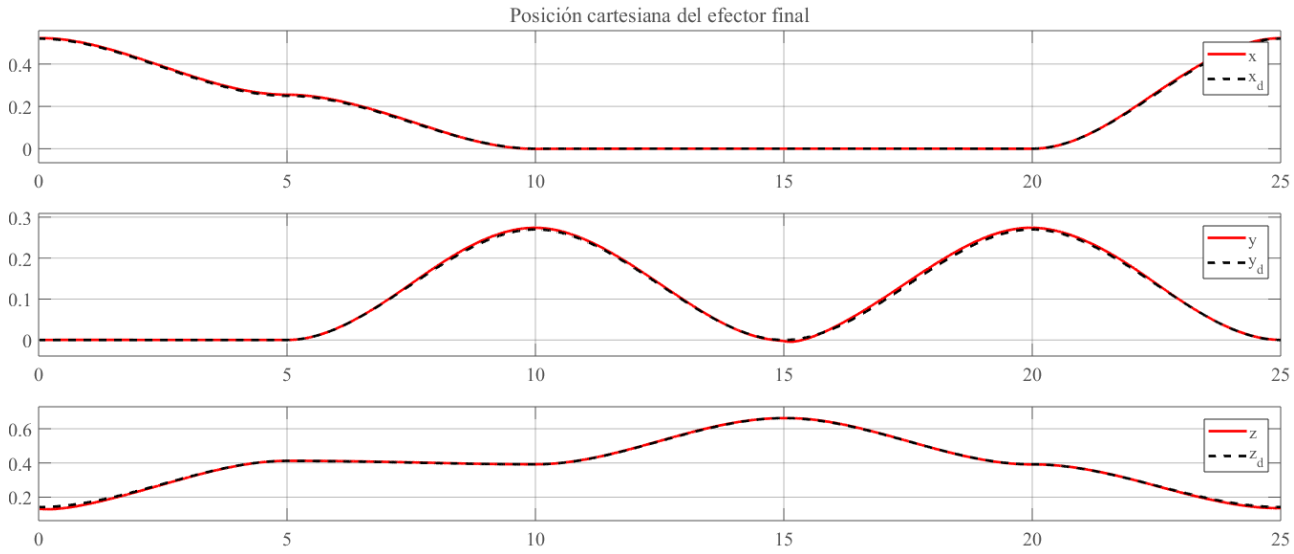

Figura 3.11: Posiciones cartesianas

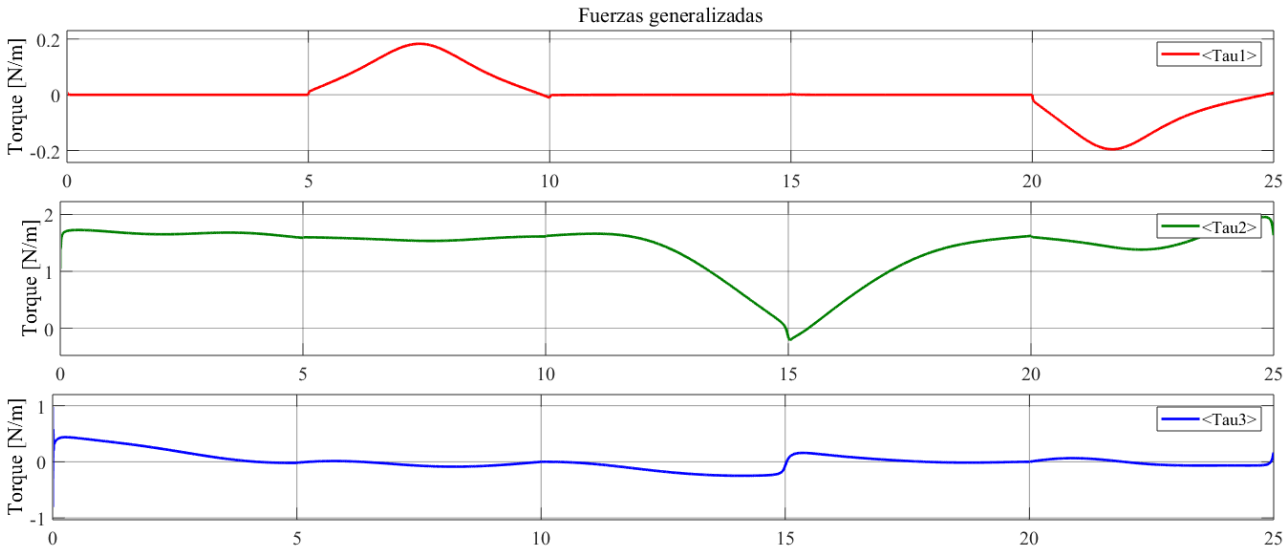

Figura 3.12: Torque

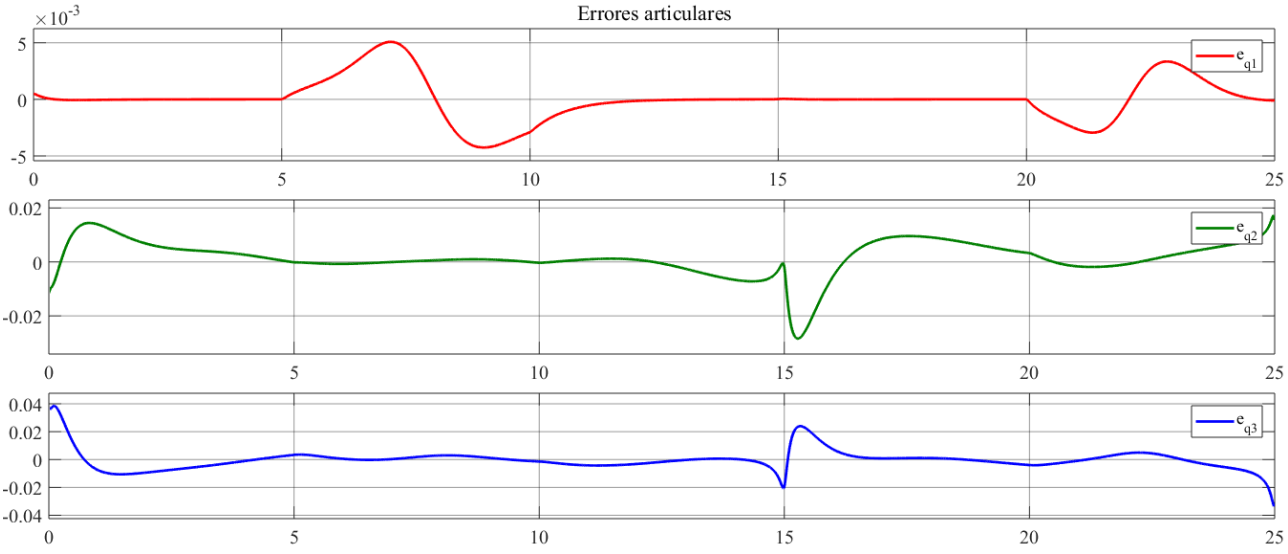

Figura 3.13: Errores articulares

En las Fig. 3.10 se presentan las trayectorias seguidas por las variables articulares, en comparación con la trayectoria deseada. Se puede notar que la posición articular deseada y

la posición leída se sobreponen ya que el error está en el orden de las milésimas. En la Fig. 3.11 se muestran las posiciones cartesianas actuales, en x, y, z del efector final, junto con las trayectorias de referencia, teniendo también en este caso gráficas superpuestas, debido al mínimo error de seguimiento existente.

3.3.3. Conclusiones de la implementación del controlador PID+G

La implementación del controlador PID+G permite mejorar el seguimiento de trayectorias articulares al reducir la carga que impone la gravedad sobre el lazo de realimentación. En particular, al incluir el término $g(q)$, el controlador PID se enfoca principalmente en compensar la dinámica residual y las perturbaciones, lo que suele traducirse en menores errores de seguimiento y en una menor demanda de acción integral.

Asimismo, el desempeño del esquema depende de manera directa de la fidelidad del modelo gravitacional y de la correcta sintonización de K_p, K_d y K_i. Por lo tanto, es recomendable incorporar saturación de torque y un procedimiento sistemático de ajuste de ganancias, con el fin de asegurar estabilidad, evitar oscilaciones y mejorar la robustez ante incertidumbres del modelo. En este caso la sintonización de ganancias se realizó de forma eurística.

Capítulo 4

Control servovisual en Matlab-CoppeliaSim

En la actualidad, los robots manipuladores se utilizan en una amplia gama de aplicaciones industriales, desde la fabricación hasta la logística y la medicina. Sin embargo, muchos de estos robots dependen en gran medida de la información proporcionada por sensores visuales para realizar tareas con precisión y adaptarse a entornos cambiantes. Aquí es donde entra en juego el control servovisual, que combina la capacidad de percepción visual con el control de movimiento del robot para lograr resultados más eficientes y flexibles.

En este capítulo, se abordará el control servovisual basado en imágenes, que es una técnica ampliamente utilizada en robótica para guiar el movimiento de robots manipuladores utilizando información visual. Se presentarán las principales características de esta técnica, los pasos a seguir para completar el método, el modelo de cámara pinhole, que es comúnmente utilizado para representar la proyección de imágenes en un plano de imagen. Además, se explicará cómo obtener el Jacobiano de la imagen, que es una herramienta importante para el control servovisual.

4.1. Pasos en el desarrollo del control servovisual

El desarrollo del control servovisual para el presente trabajo, implica varios pasos fundamentales que se describen a continuación:

1. Adquisición de datos visuales: Este paso implica la captura de datos visuales utilizando

1 cámara empotrada en el efector final. Se obtiene información de una imagen 2D.

2. Procesamiento de imágenes: Una vez que se han adquirido los datos visuales, se realiza un procesamiento de imágenes para extraer características relevantes y obtener información útil. En este caso se realizó detección de colores, rojo, verde amarillo y azul. Además se implementó la detección de formas, en específico del círculo.

3. Mapeo: Se debe realizar mapeo de los datos visuales a coordenadas en el sistema de coordenadas del robot.

4. Detección y seguimiento de objetos: Una vez que se ha realizado el procesamiento de imágenes y el mapeo a coordenadas del robot, se pueden detectar y seguir objetos en el entorno. Esto implica la identificación de objetos de interés y el seguimiento de su movimiento en las sucesivas imágenes capturadas.

5. Generación de comandos de control: Con la detección y seguimiento de objetos, se generan comandos de control para guiar el movimiento del robot. Estos comandos se basan en la información visual proporcionada para el cálculo del torque necesario para activar las articulaciones del robot.

6. Control del robot: Los comandos de control generados se envían al controlador del robot manipulador para ejecutar el movimiento necesario. El controlador del robot interpreta los comandos y genera las señales de control adecuadas para las articulaciones y los actuadores del robot.

7. Bucle de retroalimentación: El control servovisual implica un bucle de retroalimentación constante, donde la información visual se utiliza para ajustar y corregir el movimiento del robot. A medida que se capturan nuevas imágenes y se actualiza la información visual, se generan nuevos comandos de control para mantener el objetivo deseado.

4.2. Implementación del control servovisual

Una vez obtenido el jacobiano de la imagen L_s en la Sección 2.3, es necesario definir la matriz que será utilizada en el caso de estudio del presente trabajo. En este contexto, cabe señalar que los centros de los círculos detectados poseen coordenadas u_i y v_i. Asimismo,

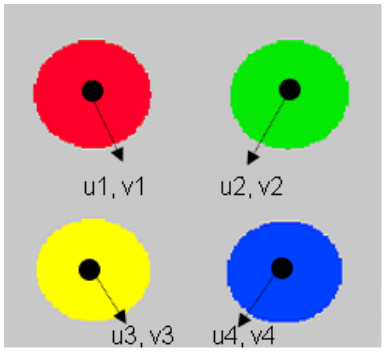

Figura 4.1: Características de la imagen

dado que se emplearán las coordenadas de los centros correspondientes a cuatro círculos detectados en la imagen, como se muestra en la Figura 4.1, el jacobiano de la imagen adopta la forma expresada en la ecuación 4.1. Donde $\bar{u} = u_i - u_0$ y $\bar{v} = v_i - v_0$ son las coordenadas de píxel con respecto al punto principal, f' es la distancia focal, u_i y v_i son las coordenadas de los centros de los círculos detectados y Z_i es la profundidad del origen del robot hasta el centro de cada círculo.

$$L_s = \begin{bmatrix} \frac{-f'}{Z_1} & 0 & \frac{\bar{u_1}}{Z_1} & \frac{\bar{u_1}\bar{v_1}}{f'} & \frac{-f'^2+\bar{u_1}^2}{f'} & \bar{v_1} \\ 0 & \frac{-f'}{Z_1} & \frac{\bar{v_1}}{Z_1} & \frac{f'^2+\bar{v_1}^2}{f'} & \frac{-\bar{u_1}\bar{v_1}}{f'} & -\bar{u_1} \\ \frac{-f'}{Z_2} & 0 & \frac{\bar{u_2}}{Z_2} & \frac{\bar{u_2}\bar{v_2}}{f'} & \frac{-f'^2+\bar{u_2}^2}{f'} & \bar{v_2} \\ 0 & \frac{-f'}{Z_2} & \frac{\bar{v_2}}{Z_2} & \frac{f'^2+\bar{v_2}^2}{f'} & \frac{-\bar{u_2}\bar{v_2}}{f'} & -\bar{u_2} \\ \frac{-f'}{Z_3} & 0 & \frac{\bar{u_3}}{Z_3} & \frac{\bar{u_3}\bar{v_3}}{f'} & \frac{-f'^2+\bar{u_3}^2}{f'} & \bar{v_3} \\ 0 & \frac{-f'}{Z_3} & \frac{\bar{v_3}}{Z_3} & \frac{f'^2+\bar{v_3}^2}{f'} & \frac{-\bar{u_3}\bar{v_3}}{f'} & -\bar{u_3} \\ \frac{-f'}{Z_4} & 0 & \frac{\bar{u_4}}{Z_4} & \frac{\bar{u_4}\bar{v_4}}{f'} & \frac{-f'^2+\bar{u_4}^2}{f'} & \bar{v_4} \\ 0 & \frac{-f'}{Z_4} & \frac{\bar{v_4}}{Z_4} & \frac{f'^2+\bar{v_4}^2}{f'} & \frac{-\bar{u_4}\bar{v_4}}{f'} & -\bar{u_4} \end{bmatrix} \tag{4.1}$$

Se puede hacer uso de la ecuación 4.2 para el cálculo de la velocidad de las características usando la velocidad de la cámara:

$$\dot{s} = L_s \xi_c^c \tag{4.2}$$

Dado que el problema de interés es obtener la velocidad del efector final necesaria para alcanzar la velocidad a la que se mueven las características de la imagen, podemos reescribir

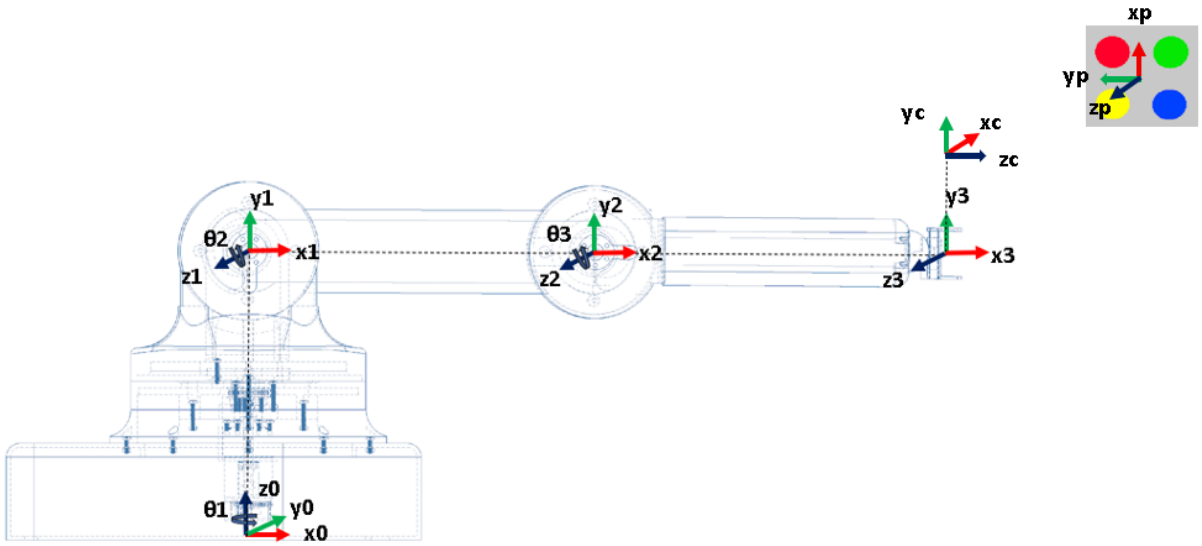

Figura 4.2: Marcos de referencia del sistema de control servovisual

la ecuación anterior como:

$$\xi_c^c = L_s^{-1}\dot{s} \tag{4.3}$$

Debe notarse que ξ_c^c es la velocidad de la cámara vista desde el marco de referencia de la cámara, sin embargo, el problema requiere calcular la velocidad del efector final visto desde el marco de referencia del origen. Por lo tanto, se aplican las siguientes transformaciones:

$$\xi_e^0 = T_n^0 T_c^n \xi_c^c \tag{4.4}$$

Donde T_c^n es una tranformación homogénea que permite expresar la ξ_c^c en el marco de referencia del efector final, en este caso simplemente se aplica una rotación alrededor de y, posteriormente T_n^0 traslada esta velocidad expresada con referencia al marco del efector final, hacia el marco de referencia del origen. Para ilustrar de mejor forma las transformaciones realizadas, en la Figura 4.2 se muestran los marcos de referencia del sistema de control servovisual.

Conocida la velocidad requerida del efector final para igualar la velocidad de las características de la imagen, se puede usar la ecuación de cinemática diferencial del robot manipulador:

$$\xi_n^0 = J\dot{q} \tag{4.5}$$

Se debe tener en cuenta que las variables de interés a controlar son las velocidades articulares, por lo tanto, de la ec. 4.5 se tiene:

$$\dot{q} = J^{-1}\xi_n^0 \tag{4.6}$$

Y sustituyendo 4.4 en 4.6 obtenemos:

$$\dot{q} = J^{-1}T_n^0 T_c^n L_s^{-1}\dot{s} \tag{4.7}$$

Para acuñar la ley de control, en primer lugar, se define el error de posición como la diferencia entre la variable de estado real y la referencia, con el propósito de cuantificar la desviación del sistema respecto al comportamiento objetivo. En este caso, s(t) corresponde al vector de coordenadas de los círculos en la imagen en el instante t y $s_d(t)$ corresponde al vector de coordenadas de los círculos cuando la imagen está centrada.

$$e(t) = s(t) - s_d(t) \tag{4.8}$$

A continuación, se deriva el error con respecto al tiempo para relacionar las velocidades real y deseada del sistema.

$$\dot{e}(t) = \dot{s}(t) - \dot{s}_d(t) \tag{4.9}$$

Posteriormente, con el objetivo de garantizar que el error converja asintóticamente a cero, se impone una dinámica deseada de primer orden para el error. Esta elección asegura estabilidad exponencial del equilibrio $e(t) = 0$, siempre que la ganancia de control sea positiva (Slotine & Li, 1991).

$$\dot{e}(t) = -k\, e(t), \quad k > 0 \tag{4.10}$$

Seguidamente, al sustituir la definición del error en la dinámica deseada, se obtiene una relación directa entre la diferencia de velocidades y el error de posición. Esta expresión indica cómo debe ajustarse la velocidad real del sistema en función de la desviación existente respecto a la trayectoria deseada.

$$\dot{s}(t) - \dot{s}_d(t) = -k\left(s(t) - s_d(t)\right) \tag{4.11}$$

Finalmente, al despejar la velocidad real del sistema, se obtiene la ley de control en velocidad. Dado que la velocidad deseada es 0, la expresión final solo considera un término correctivo proporcional al error, lo que permite compensar desviaciones y garantizar un comportamiento estable del sistema.

$$\dot{s}(t) = -k\left(s(t) - s_d(t)\right) \tag{4.12}$$

Sin embargo, dado que la velocidad deseada para las características de la imagen es 0, es posible reducir la expresión 4.12 como se muestra a continuación:

$$\dot{s} = -ke_s \tag{4.13}$$

Lo cual es un error de posición multiplicado por una ganacia:

$$e_s = (s - s_d) \tag{4.14}$$

Finalmente, al sustituir 4.13 en 4.7, se obtiene:

$$\dot{q} = -kJ^{-1}T_e^0 T_c^e L_s^{-1} e_s \tag{4.15}$$

Cuyo diagrama de control se muestra en la Fig. 4.3.

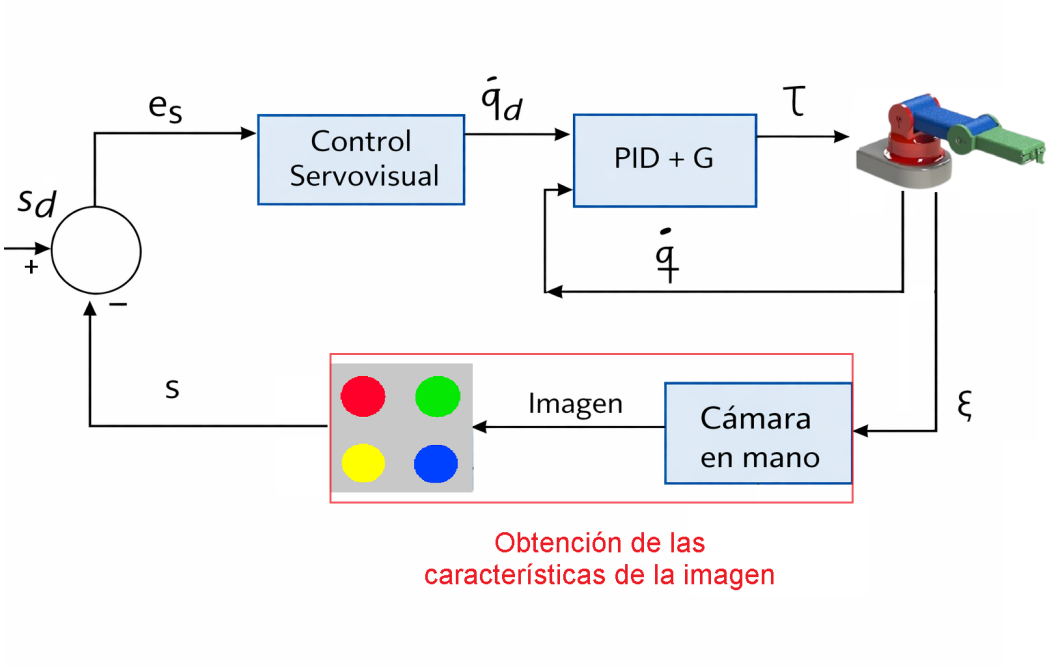

Figura 4.3: Diagrama de bloques del control servovisual

4.3. Configuración del entorno Matlab-Coppelia Sim

1. Para iniciar con la simulación del control servovisual del manipulador, en primer lugar es necesario tener en cuenta la exportación de ensambles desde SolidWorks hacia CoppeliaSim para desarrollar simulaciones robóticas avanzadas con geometrías reales. Asimismo, la correcta definición de juntas, masas y sistemas de coordenadas resulta fundamental para obtener simulaciones dinámicas precisas.

   Esto requiere:

   - SolidWorks URDF Exporter.

   Posteriormente, en CoppeliaSim: $File- > Import- > URDF$

   Se recomienda:

   - Reducir complejidad geométrica.
   - Usar mallas simplificadas.

- Verificar escalas.
- Definir correctamente ejes articulares.
- Configurar adecuadamente masas e inercias.

2. Posteriormente, debe establecerse la comunicación entre Matlab y Coppelia Sim, lo cual es posible al utilizar la *Legacy Remote API*, ya que utiliza una arquitectura cliente-servidor basada en sockets TCP/IP. Según Coppelia Robotics [39], esta API permite controlar simulaciones, leer sensores y enviar comandos articulares desde aplicaciones externas.

   En Matlab, la conexión se realiza utilizando:

```
1 sim = remApi('remoteApi');
```

   donde:

   - `remApi` carga la librería de comunicación.
   - `remoteApi` corresponde al archivo dinámico de la API.

   La comunicación entre el software utilizado, sigue la arquitectura mostrada en la Figura 4.4.

Figura 4.4: Arquitectura cliente-servidor de la Legacy Remote API.

3. Ahora, para configurar Coppelia Sim, debe verificarse que el servidor remoto esté habilitado. El archivo: remoteApiConnections.txt, se encuentra normalmente en:

   `CoppeliaSim/programming/legacyRemoteApiConnections.txt`

   Debe contener:

```
portIndex1_port             = 19997
portIndex1_debug            = false
portIndex1_syncSimTrigger   = true
```

Enseguida, iniciar CoppeliaSim.

4. A continuación, en Matlab agregar las librerías de la API remota.

   La carpeta normalmente es:

   `CoppeliaSim/programming/legacyRemoteApi/remoteApiBindings/matlab/matlab`

   Agregar la ruta:

```
addpath('C:/CoppeliaSim/programming/
legacyRemoteApi/remoteApiBindings/matlab/matlab')
```

   Después, cargar la librería remota:

```
sim = remApi('remoteApi');
```

   Esto crea el objeto de comunicación con CoppeliaSim.

   Antes de iniciar una nueva conexión, es recomendable cerrar conexiones existentes.

```
sim.simxFinish(-1);
```

   El parámetro `-1` indica cerrar todas las conexiones activas.

   Posteriormente, se establece la conexión TCP/IP.

```
clientID = sim.simxStart(...
    '127.0.0.1', ...
    19997, ...
    true, ...
    true, ...
```

```
6       5000, ...
7       5);
```

donde:

- 127.0.0.1: dirección local.
- 19997: puerto de comunicación.
- 5000: tiempo máximo de espera.
- 5: número de reintentos.

La conexión se verifica mediante:

```
1 if (clientID > -1)
2
3     disp('Conexion exitosa')
4
5 else
6
7     disp('Error de conexion')
8
9 end
```

Una vez conectado, puede iniciarse la simulación.

```
1 sim.simxStartSimulation(...
2     clientID,...
3     sim.simx_opmode_blocking);
```

4.4. Código Matlab

El control servovisual se programó en Matlab, utilizando 2 funciones principales: control_cinematico y camara1.

En control_cinematico se realiza el control para mandar el robot al punto inicial o home. Y la función cámara, obtiene las características de la imagen, obtiene los parámetros del centro de cada círculo detectado y coloca las coordenadas en un vector s.

4.5. Control cinemático

La primera etapa del control servovisual consiste en posicionar al robot manipulador en su posición home durante 10 segundos utilizando el Código 4.1 mostrado a continuación, para que tenga un punto de inicio con acceso a la vista del objeto a enfocar.

Código 4.1: Control cinemático en Matlab

```
MATLAB
clc;
clear;
close all;
sim = remApi('remoteApi');
sim.simxFinish(-1);

clientID = sim.simxStart('127.0.0.1',19999,true,true,5000,5);
k = 5;

global q1 q2 q3;

if (clientID > -1)

    %% ====== Handles ======
    [¬, j1]  = ...
        sim.simxGetObjectHandle(clientID,'j1',sim.simx_opmode_blocking);
    [¬, j2]  = ...
        sim.simxGetObjectHandle(clientID,'j2',sim.simx_opmode_blocking);
    [¬, j3]  = ...
        sim.simxGetObjectHandle(clientID,'j3',sim.simx_opmode_blocking);
    [¬, Ref] = ...
        sim.simxGetObjectHandle(clientID,'Ref',sim.simx_opmode_blocking);

    %% ====== Streaming ======
    sim.simxGetObjectPosition(clientID,Ref,-1,sim.simx_opmode_streaming);
    sim.simxGetJointPosition(clientID,j1,sim.simx_opmode_streaming);
    sim.simxGetJointPosition(clientID,j2,sim.simx_opmode_streaming);
    sim.simxGetJointPosition(clientID,j3,sim.simx_opmode_streaming);

    % Tiempo de Coppelia via float signal
```

```
28      sim.simxGetFloatSignal(clientID,'simulationTime', ...
            sim.simx_opmode_streaming);
29
30      % ====== VELOCIDAD REAL (PID+g) desde Coppelia ======
31      sim_jointfloatparam_velocity = 2012; % joint velocity
32      sim.simxGetObjectFloatParameter(clientID, j1, ...
            sim_jointfloatparam_velocity, sim.simx_opmode_streaming);
33      sim.simxGetObjectFloatParameter(clientID, j2, ...
            sim_jointfloatparam_velocity, sim.simx_opmode_streaming);
34      sim.simxGetObjectFloatParameter(clientID, j3, ...
            sim_jointfloatparam_velocity, sim.simx_opmode_streaming);
35
36      pause(0.1);
37
38      %% ====== Logs ======
39      i = 0;
40      time = [];
41      q_hist = [];
42      qp_hist = [];
43      x_hist = [];
44      xref_hist = [];
45      err_hist = [];
46      dt_hist = [];
47
48      % Nuevos logs para velocidad real y error de velocidad
49      qdot_real_hist = [];
50      qdot_err_hist  = [];
51
52      x_des = [0.5; 0.0; 0.2];
53
54      T_end = 150;
55      Ts = 0.05;
56
57      useSimTime = true;
58
59      tic;
60      while (toc < T_end)
61
62          [retref, ref_v] = sim.simxGetObjectPosition(clientID,Ref,-1,...
63          sim.simx_opmode_buffer);
```

```
64          [retj1, q1] = ...
               sim.simxGetJointPosition(clientID,j1,sim.simx_opmode_buffer);
65          [retj2, q2] = ...
               sim.simxGetJointPosition(clientID,j2,sim.simx_opmode_buffer);
66          [retj3, q3] = ...
               sim.simxGetJointPosition(clientID,j3,sim.simx_opmode_buffer);
67
68          % Leer tiempo del simulador
69          [retts, simTime] = ...
               sim.simxGetFloatSignal(clientID,'simulationTime', ...
               sim.simx_opmode_buffer);
70
71          if retts ~= sim.simx_return_ok
72              simTime = toc;
73              useSimTime = false;
74          end
75
76          if (retj1 ~= sim.simx_return_ok) || ...
77             (retj2 ~= sim.simx_return_ok) || ...
78             (retj3 ~= sim.simx_return_ok)
79              pause(Ts);
80              continue;
81          end
82
83          % ====== Leer VELOCIDAD REAL (PID+g) ======
84          [retv1, qdot1_real] = sim.simxGetObjectFloatParameter(clientID, ...
               j1, sim_jointfloatparam_velocity, sim.simx_opmode_buffer);
85          [retv2, qdot2_real] = sim.simxGetObjectFloatParameter(clientID, ...
               j2, sim_jointfloatparam_velocity, sim.simx_opmode_buffer);
86          [retv3, qdot3_real] = sim.simxGetObjectFloatParameter(clientID, ...
               j3, sim_jointfloatparam_velocity, sim.simx_opmode_buffer);
87
88          if (retv1 ~= sim.simx_return_ok) || ...
89             (retv2 ~= sim.simx_return_ok) || ...
90             (retv3 ~= sim.simx_return_ok)
91              pause(Ts);
92              continue;
93          end
94
95          %% ====== Cinematica ======
```

```
96          [J, t] = Jacobiano(q1,q2,q3);
97
98          % Referencia
99          xref = x_des;
100         xref = ref_v(:);    % referencia del simulador
101
102         err = t(:) - xref(:);
103         Jv = J(1:3,1:3);
104
105         %% ====== Control ======
106         qp = -k * pinv(Jv) * err;
107
108         % Saturacion
109         qp_max = 5;  % rad/s
110         qp = max(min(qp, qp_max), -qp_max);
111
112         % Enviar se ales (qdot deseada)
113         sim.simxSetFloatSignal(clientID,'dato1',qp(1),...
114         sim.simx_opmode_oneshot);
115         sim.simxSetFloatSignal(clientID,'dato2',qp(2),...
116         sim.simx_opmode_oneshot);
117         sim.simxSetFloatSignal(clientID,'dato3',qp(3),...
118         sim.simx_opmode_oneshot);
119
120         %% ====== Guardar ======
121         i = i + 1;
122         time(i,1) = simTime;
123         q_hist(i,:) = [q1 q2 q3];
124         qp_hist(i,:) = qp(:).';
125         x_hist(i,:) = t(:).';
126         xref_hist(i,:) = xref(:).';
127         err_hist(i,:) = err(:).';
128
129         % Guardar velocidad real + error de velocidad
130         qdot_real_hist(i,:) = [qdot1_real qdot2_real qdot3_real];
131         qdot_err_hist(i,:)  = qdot_real_hist(i,:) - qp_hist(i,:); % ...
              real - deseada
132
133         if i > 1
134             dt_hist(i-1,1) = time(i) - time(i-1);
```

```
135         end
136
137         pause(Ts);
138     end
139
140     sim.simxFinish(clientID);
```

4.6. Procesamiento de imágenes en Matlab

Para implementar el procesamiento de imágenes se utiliza el lenguaje M, propio de Matlab, el cual es un lenguaje claro y conciso con buen soporte para diversas aplicaciones, incluyendo procesamiento de imágenes.

Una vez que el robot se encuentra en su posición inicial, se activa el sensor de visión colocado en el efector final del manipulador con identificador único denominado "*vision_sensor*" y se obtiene la imagen en modo streaming para iniciar con su procesamiento y detectar los patrones de colores.

Después de leer y almacenar la imagen, esta se convierte de formato RGB a HSV, para que el procesamiento a realizar tenga mayor robustez ante cambios de iluminación.

Posteriormente se aplica una máscara a la imagen HSV, para todos los colores de interés. En este caso se toman en cuenta los umbrales para los colores rojo, amarillo, azul y verde.

Una vez que se obtiene la máscara, se aplica la función imfindcircles, la cual forma parte de las librerías de Matlab, que detecta círculos en una imagen binaria utilizando la Transformada de Hough Circular. Se define el rango de radios de los círculos a buscar en la imagen, en este caso, se buscan círculos con radios entre 6 y 25 píxeles. También se ajusta la sensibilidad del detector, un valor cercano a 1 aumenta la probabilidad de detectar círculos tenues o parcialmente visibles, pero también puede incrementar los falsos positivos. Esta función devuelve el radio y las coordenadas del centro del círculo para los colores rojo, amarillo, verde y azul; las coordenadas de los centros son las características de interés en la imagen, por lo cual se almacenan en una matriz s de dimensiones 2x4. El código 4.2 en Matlab que se muestra a continuación contiene el procedimiento para procesar las imágenes obtenidas del sensor de visión.

Código 4.2: Procesamiento de la imagen y obtención de características

```
MATLAB
function [s2, qd, e_points_hist, pos_cam_hist, t] = camara1()

clc; clear; close all;

%% ====== ESTILO GLOBAL (para TODAS las figuras) ======
set(groot, 'defaultFigureColor', 'w');
set(groot, 'defaultAxesFontSize', 20);
set(groot, 'defaultAxesLineWidth', 2);
set(groot,'defaultFigureWindowState','maximized');
set(groot, 'defaultAxesTickDir', 'out');
set(groot, 'defaultAxesTickLength', [0.015 0.015]);
set(groot, 'defaultLineLineWidth', 3);
set(groot, 'defaultTextFontSize', 36);
set(groot, 'defaultAxesFontName', 'Times New Roman');
set(groot, 'defaultTextFontName', 'Times New Roman');
set(groot,'defaultAxesTitleFontSizeMultiplier',2);


%% ====== CARPETA RESULTADOS ======
folderName = 'Resultados_Figuras';
if ¬exist(folderName,'dir'), mkdir(folderName); end
timestamp = datestr(now,'yyyy_mm_dd_HHMMSS');

%% ====== CONEXION ======
sim = remApi('remoteApi');
sim.simxFinish(-1);

clientID = sim.simxStart('127.0.0.1',19999,true,true,5000,5);

s2 = [];
qd = [];
e_points_hist = [];
pos_cam_hist = [];
t = [];

global q1 q2 q3

if clientID ≤ -1
```

```
40      disp('fall  la conexi n');
41      sim.delete();
42      return
43  end
44
45  sim.simxStartSimulation(clientID, sim.simx_opmode_oneshot);
46
47  %% ====== HANDLES ======
48  [¬, cam]    = sim.simxGetObjectHandle(clientID,'Vision_sensor',...
49  sim.simx_opmode_blocking);
50  [¬, plano] = ...
        sim.simxGetObjectHandle(clientID,'Plane',sim.simx_opmode_blocking);
51
52  [¬, j1] = sim.simxGetObjectHandle(clientID,'j1',sim.simx_opmode_blocking);
53  [¬, j2] = sim.simxGetObjectHandle(clientID,'j2',sim.simx_opmode_blocking);
54  [¬, j3] = sim.simxGetObjectHandle(clientID,'j3',sim.simx_opmode_blocking);
55
56  % ========= ARRANQUE DE TRAYECTORIA DESDE MATLAB =========
57  % Forzamos flanco 0 -> 1 para que el script en Coppelia reinicie t0 y ...
        arranque
58  sim.simxSetIntegerSignal(clientID,'startTrajectory',0,...
59  sim.simx_opmode_oneshot);
60  pause(0.05);
61  sim.simxSetIntegerSignal(clientID,'startTrajectory',1,...
62  sim.simx_opmode_oneshot);
63  % ========================================================
64
65
66  %% ====== STREAMING ======
67  sim.simxGetVisionSensorImage2(clientID, cam, 0, sim.simx_opmode_streaming);
68  sim.simxGetObjectPosition(clientID, plano, cam, sim.simx_opmode_streaming);
69  sim.simxGetObjectPosition(clientID, cam, -1, sim.simx_opmode_streaming);
70
71  sim.simxGetObjectPosition(clientID, plano, -1, ...
        sim.simx_opmode_streaming); % para 3D
72  sim.simxGetJointPosition(clientID, j1, sim.simx_opmode_streaming);
73  sim.simxGetJointPosition(clientID, j2, sim.simx_opmode_streaming);
74  sim.simxGetJointPosition(clientID, j3, sim.simx_opmode_streaming);
75
76  % tiempo del simulador (float signal)
```

```
77 sim.simxGetFloatSignal(clientID, 'simulationTime', ...
       sim.simx_opmode_streaming);
78 sim.simxGetIntegerSignal(clientID, 'ctrlMode', sim.simx_opmode_streaming);
79
80 % ====== VELOCIDAD REAL desde Coppelia ======
81 sim_jointfloatparam_velocity = 2012; % joint velocity
82 sim.simxGetObjectFloatParameter(clientID, j1, ...
       sim_jointfloatparam_velocity, sim.simx_opmode_streaming);
83 sim.simxGetObjectFloatParameter(clientID, j2, ...
       sim_jointfloatparam_velocity, sim.simx_opmode_streaming);
84 sim.simxGetObjectFloatParameter(clientID, j3, ...
       sim_jointfloatparam_velocity, sim.simx_opmode_streaming);
85
86 % ====== TORQUE / FUERZA REAL EN LA ARTICULACION ======
87 sim.simxGetJointForce(clientID, j1, sim.simx_opmode_streaming);
88 sim.simxGetJointForce(clientID, j2, sim.simx_opmode_streaming);
89 sim.simxGetJointForce(clientID, j3, sim.simx_opmode_streaming);
90
91 pause(0.1);
92
93 %% ====== PARAMETROS ======
94 %Leimiscata 45
95 %Ruedas 300
96 %150 corazon
97
98 T_end = 150;    % tiempo de finalizacion
99 Ts    = 0.05;   % muestreo (50 ms)
100
101 %% ====== LOGS ======
102 k = 0;
103
104 pos_plano_hist  = [];
105 pos_cam_hist    = [];
106 qd              = [];
107 t               = [];
108 e_points_hist   = [];
109 e_global_hist   = [];
110
111 % nuevos logs (velocidades):
112 qdot_des_hist   = [];   % daros mandados
```

```
qdot_real_hist  = [];   % lo que reporta Coppelia
qdot_err_hist   = [];   % real - deseada
dt_hist         = [];   % dt desde simulationTime (o toc fallback)
useSimTime      = true;

% nuevo log (torques):
tau_hist        = [];   % torque (N m) / fuerza (N) reportado por el joint

tic;
while toc < T_end

    %% ====== IMAGEN ======
    [retImg, ~, img] = sim.simxGetVisionSensorImage2(clientID, cam, 0, ...
        sim.simx_opmode_buffer);
    if retImg ~= sim.simx_return_ok || isempty(img)
        pause(Ts);
        continue;
    end

    imagenHSV = rgb2hsv(img);

    % M scaras HSV
    mR1 = (imagenHSV(:,:,1) >= 0    & imagenHSV(:,:,1) <= 0.05) & ...
        (imagenHSV(:,:,2) >= 0.5) & (imagenHSV(:,:,3) >= 0.5);
    mR2 = (imagenHSV(:,:,1) >= 0.95 & imagenHSV(:,:,1) <= 1.00) & ...
        (imagenHSV(:,:,2) >= 0.5) & (imagenHSV(:,:,3) >= 0.5);
    mR  = mR1 | mR2;

    mG = (imagenHSV(:,:,1) >= 0.25 & imagenHSV(:,:,1) <= 0.40) & ...
        (imagenHSV(:,:,2) >= 0.5) & (imagenHSV(:,:,3) >= 0.5);
    mB = (imagenHSV(:,:,1) >= 0.55 & imagenHSV(:,:,1) <= 0.65) & ...
        (imagenHSV(:,:,2) >= 0.5) & (imagenHSV(:,:,3) >= 0.5);
    mY = (imagenHSV(:,:,1) >= 0.15 & imagenHSV(:,:,1) <= 0.20) & ...
        (imagenHSV(:,:,2) >= 0.5) & (imagenHSV(:,:,3) >= 0.5);

    [cR, ~] = imfindcircles(mR, [6 25], 'Sensitivity', 0.8);
    [cG, ~] = imfindcircles(mG, [6 25], 'Sensitivity', 0.8);
    [cB, ~] = imfindcircles(mB, [6 25], 'Sensitivity', 0.8);
    [cY, ~] = imfindcircles(mY, [6 25], 'Sensitivity', 0.8);

```

```
147     %% ====== JOINTS Y DISTANCIA ======
148     [retj1, q1] = sim.simxGetJointPosition(clientID, j1, ...
            sim.simx_opmode_buffer);
149     [retj2, q2] = sim.simxGetJointPosition(clientID, j2, ...
            sim.simx_opmode_buffer);
150     [retj3, q3] = sim.simxGetJointPosition(clientID, j3, ...
            sim.simx_opmode_buffer);
151
152     if (retj1 ≠ sim.simx_return_ok) || (retj2 ≠ sim.simx_return_ok) || ...
            (retj3 ≠ sim.simx_return_ok)
153         pause(Ts);
154         continue;
155     end
156
157     [retPC, posPlanoCam] = sim.simxGetObjectPosition(clientID, plano, ...
            cam, sim.simx_opmode_buffer);
158     if retPC ≠ sim.simx_return_ok || isempty(posPlanoCam)
159         pause(Ts);
160         continue;
161     end
162     distancia_z = norm(posPlanoCam);
163
164     %% ====== TIEMPO SIMULACION ======
165     [retts, simTime] = sim.simxGetFloatSignal(clientID, ...
            'simulationTime', sim.simx_opmode_buffer);
166     if retts ≠ sim.simx_return_ok || isempty(simTime)
167         simTime = toc;
168         useSimTime = false;
169     end
170
171     %% ====== CONTROL SERVO-VISUAL ======
172     okR = ¬isempty(cR); okG = ¬isempty(cG); okB = ¬isempty(cB); okY = ¬...
            isempty(cY);
173
174     if okR && okG && okB && okY
175         s2 = [cR(1,:); cG(1,:); cB(1,:); cY(1,:)];
176         [qpc2, es] = Jiimagen_copia(s2, q1, q2, q3, distancia_z);
177     else
178         s2   = zeros(4,2);
179         es   = zeros(8,1);
```

```
180         qpc2 = [0;0;0];
181         warning('Faltan c rculos (R:%d G:%d B:%d Y:%d). qpc2 = [0 0 ...
                0].', okR, okG, okB, okY);
182     end
183
184     % qdot deseada
185     qpc2 = double(qpc2(:)).';  % 1x3
186
187     % Enviar se ales
188     sim.simxSetFloatSignal(clientID,'dato1', qpc2(1), ...
            sim.simx_opmode_oneshot);
189     sim.simxSetFloatSignal(clientID,'dato2', qpc2(2), ...
            sim.simx_opmode_oneshot);
190     sim.simxSetFloatSignal(clientID,'dato3', qpc2(3), ...
            sim.simx_opmode_oneshot);
191
192     %% ====== LEER VELOCIDAD REAL  ======
193     [retv1, qdot1_real] = sim.simxGetObjectFloatParameter(clientID, j1, ...
            sim_jointfloatparam_velocity, sim.simx_opmode_buffer);
194     [retv2, qdot2_real] = sim.simxGetObjectFloatParameter(clientID, j2, ...
            sim_jointfloatparam_velocity, sim.simx_opmode_buffer);
195     [retv3, qdot3_real] = sim.simxGetObjectFloatParameter(clientID, j3, ...
            sim_jointfloatparam_velocity, sim.simx_opmode_buffer);
196
197     if (retv1 ≠ sim.simx_return_ok) || (retv2 ≠ sim.simx_return_ok) || ...
            (retv3 ≠ sim.simx_return_ok)
198         pause(Ts);
199         continue;
200     end
201
202     qdot_real = [qdot1_real, qdot2_real, qdot3_real];
203
204     %% ====== LEER TORQUE / FUERZA REAL  ======
205     [rett1, tau1] = sim.simxGetJointForce(clientID, j1, ...
            sim.simx_opmode_buffer);
206     [rett2, tau2] = sim.simxGetJointForce(clientID, j2, ...
            sim.simx_opmode_buffer);
207     [rett3, tau3] = sim.simxGetJointForce(clientID, j3, ...
            sim.simx_opmode_buffer);
208
```

```
209     if (rett1 ≠ sim.simx_return_ok) || (rett2 ≠ sim.simx_return_ok) || ...
            (rett3 ≠ sim.simx_return_ok)
210         pause(Ts);
211         continue;
212     end
213
214     tau = [tau1, tau2, tau3];
215
216     %% ====== POSICIONES  ======
217     [¬, q1p] = sim.simxGetJointPosition(clientID, j1, ...
            sim.simx_opmode_buffer);
218     [¬, q2p] = sim.simxGetJointPosition(clientID, j2, ...
            sim.simx_opmode_buffer);
219     [¬, q3p] = sim.simxGetJointPosition(clientID, j3, ...
            sim.simx_opmode_buffer);
220
221     [¬, pos_cam]   = sim.simxGetObjectPosition(clientID, cam,   -1, ...
            sim.simx_opmode_buffer);
222     [¬, pos_plano] = sim.simxGetObjectPosition(clientID, plano, -1, ...
            sim.simx_opmode_buffer);
223
224     %% ====== GUARDAR LOGS ======
225     k = k + 1;
226
227     t(k,1)    = simTime;
228     qd(k,1:3) = [q1p q2p q3p];
229
230     pos_plano_hist(k,:) = pos_plano(:).';
231     pos_cam_hist(k,:)   = pos_cam(:).';
232
233     E = reshape(es(:), 2, []).';
234     e_points_hist(k,:) = sqrt(sum(E.^2,2)).';
235     e_global_hist(k,1) = norm(es);
236
237     % velocidades
238     qdot_des_hist(k,:)  = qpc2;
239     qdot_real_hist(k,:) = qdot_real;
240     qdot_err_hist(k,:)  = qdot_real_hist(k,:) - qdot_des_hist(k,:);
241
242     % torque / fuerza
```

```
243     tau_hist(k,:) = tau;
244
245     %Se Lee el modo de control
246     [retcm, ctrlMode] = sim.simxGetIntegerSignal(clientID, 'ctrlMode', ...
            sim.simx_opmode_buffer);
247     if retcm ~= sim.simx_return_ok
248         ctrlMode = 1; % fallback (por si no llega)
249     end
250
251     switch ctrlMode
252         case 0, controlador = 'PID';
253         case 1, controlador = 'PID+G';
254         otherwise, controlador = 'ISMC';
255     end
256     ctrlTag = ['  - (' controlador ')'];
257
258     % dt real desde time
259     if k > 1
260         dt_hist(k-1,1) = t(k) - t(k-1);
261     end
262
263     pause(Ts);
264 end
265
266 %% ====== CERRAR ======
267 sim.simxSetIntegerSignal(clientID,'startTrajectory',0,...
268 sim.simx_opmode_oneshot);
269 sim.simxFinish(clientID);
270 sim.delete();
271
272 %% ========================
273 %          GRAFICAS
274 % ========================
275
276 %% ====== NORMALIZACION DEL ERROR (OPCION 1: % DE LA DIAGONAL) ======
277 W = 320; H = 320;
278 diag_px = sqrt(W^2 + H^2);
279
280 e_points_pct = 100 * (e_points_hist / diag_px);    % Nx4  (%)
281 e_global_pct = 100 * (e_global_hist / diag_px);    % Nx1  (%)
```

4.7. Control servovisual en Matlab-CoppeliaSim

En esta fase se toma la matriz de coordenadas de los centros de los círculos encontrados en la imagen y se pasa como parámetro a la función Jiimagen donde se guardan como vector de tamaño 8x1 antes de calcular el jacobiano de la imagen denominado L_s, posteriormente se aplica la cinemática diferencial inversa para hallar las velocidades deseadas para alcanzar la velocidad de la imagen y finalmente retorna el vector de velocidades deseadas sd a la función camara1, donde se envían como señal a CoppeliaSim para finalmente calcular el torque deseado aplicando un controlador PID+G. El código 4.3 empleado se muestra enseguida.

Código 4.3: Control servovisual en Matlab

```
MATLAB
%Autor: Karina Ramirez Lopez
function [qp, es] = Jiimagen_copia(s,q1,q2,q3,z)
    %Constantes
    k = 0.02;
    k_CI = 0.5;%ganancia del control en la cinematica inversa
    tz = 0;
    ty = 0;
    tx = 0;
    L1 = 0.209;
    L2 = 0.249976;
    L3 = 0.25384;
    sd = [133;127;
            168;127;
            168;162;
            133;162];

    z=1.0;
    z1 = z;
    z2 = z;
    z3 = z;
    z4 = z;

    lam = 1;
    u1d = sd(1);
    v1d = sd(2);
```

```
27      u2d = sd(3);
28      v2d = sd(4);
29      u3d = sd(5);
30      v3d = sd(6);
31      u4d = sd(7);
32      v4d = sd(8);
33
34      lsd = [-lam/z1,           0,  u1d/z1,             u1d*v1d/lam, ...
           -(lam^2 + u1d^2)/lam,  v1d;
35                             0, -lam/z1,  v1d/z1, (lam^2 + v1d^2)/lam, ...
                                          -u1d*v1d/lam, -u1d;
36                 -lam/z2,           0,  u2d/z2,             u2d*v2d/lam, ...
                      -(lam^2 + u2d^2)/lam,  v2d;
37                             0, -lam/z2,  v2d/z2, (lam^2 + v2d^2)/lam, - ...
                                           u2d*v2d/lam, -u2d;
38                 -lam/z3,           0,  u3d/z3,             u3d*v3d/lam, ...
                      -(lam^2 + u3d^2)/lam,  v3d;
39                             0, -lam/z3,  v3d/z3, (lam^2 + v3d^2)/lam, ...
                                          -u3d*v3d/lam, -u3d;
40                 -lam/z4,           0,  u4d/z4,             u4d*v4d/lam, ...
                      -(lam^2 + u4d^2)/lam,  v4d;
41                             0, -lam/z4,  v4d/z4, (lam^2 + v4d^2)/lam, ...
                                          -u4d*v4d/lam, -u4d;];
42
43      u1 = s(1,1);
44      v1 = s(1,2);
45      u2 = s(2,1);
46      v2 = s(2,2);
47      u3 = s(3,1);
48      v3 = s(3,2);
49      u4 = s(4,1);
50      v4 = s(4,2);
51
52      ls = [-lam/z1,           0,  u1/z1,             u1*v1/lam, -(lam^2 ...
           + u1^2)/lam,  v1;
53                             0, -lam/z1,  v1/z1, (lam^2 + v1^2)/lam, ...
                                          -u1*v1/lam, -u1;
54                 -lam/z2,           0,  u2/z2,             u2*v2/lam, ...
                      -(lam^2 + u2^2)/lam,  v2;
55                             0, -lam/z2,  v2/z2, (lam^2 + v2^2)/lam, - ...
```

```
                                        u2*v2/lam, -u2;
56              -lam/z3,           0,  u3/z3,             u3*v3/lam, ...
                   -(lam^2 + u3^2)/lam,  v3;
57                        0, -lam/z3,  v3/z3, (lam^2 + v3^2)/lam, ...
                                        -u3*v3/lam, -u3;
58              -lam/z4,           0,  u4/z4,             u4*v4/lam, ...
                   -(lam^2 + u4^2)/lam,  v4;
59                        0, -lam/z4,  v4/z4, (lam^2 + v4^2)/lam, ...
                                        -u4*v4/lam, -u4;];
60
61
62      plsd=inv(lsd'*lsd)*lsd';
63      pls=inv(ls'*ls)*ls';
64      l=(pls+plsd)/2;
65
66      s_v = [s(1,1);
67                  s(1,2);
68                  s(2,1);
69                  s(2,2);
70                  s(3,1);
71                  s(3,2);
72                  s(4,1);
73                  s(4,2)];
74      es = sd  - s_v ;
75
76     Vc_c =k*l*(es);%velocidad de la camara
77
78     R  = [cos(pi/2), 0, sin(pi/2);
79        0,       1, 0;
80       -sin(pi/2), 0, cos(pi/2)];
81     V3_3 = [R, zeros(3); zeros(3), R]*Vc_c;%velocidad del efector final ...
          visto desde el efector final
82     R0_3 = [cos(q2 + q3)*cos(q1), -sin(q2 + q3)*cos(q1),  sin(q1); ...
          cos(q2 + q3)*sin(q1), -sin(q2 + q3)*sin(q1), -cos(q1); sin(q2 + ...
          q3), cos(q2 + q3), 0];
83
84     V3_0 = [R0_3, zeros(3); zeros(3), R0_3]*V3_3;
85
86      [J, t] = Jacobiano(q1, q2, q3);
87      inv_J = inv(J'*J)*J';
```

```
88      qp = -k_CI* inv_J * V3_0;
89  end
```

4.8. Controladores en CoppeliaSim

Una vez que se obtienen las velocidades articulares deseadas s_d para alcanzar la velocidad de las características de la imagen, se envía el vector de velocidades como una señal tipo double desde Matlab, y se recibe en Coppelia, donde las velocidades articulares deseadas se pasan como parámetro a la función del controlador, con la cual se calculará el vector de torque necesario para activar las articulaciones, esta señal que se calcula de forma continua se establece como la fuerza objetivo para cada articulación, las cuales deberán estar configuradas en modo $'fuerza'$. El programa para los controladores se muestra en el código 4.4.

Código 4.4: Controladores PID+G, PID y ISMC en CoppeliaSim

```
1
2  # ========================
3  # Controladores
4  # ========================
5
6  def pid_vel(qp, qp_des, tau_lim, ie, ea, dt, kp_vec, ki_vec, kd_vec):
7      kp = np.array(kp_vec, dtype=float).reshape((3,1))
8      ki = np.array(ki_vec, dtype=float).reshape((3,1))
9      kd = np.array(kd_vec, dtype=float).reshape((3,1))
10
11     e  = qp_des - qp
12     de = (e - ea) / dt
13     ie = ie + e * dt
14     ea = e.copy()
15
16     tau = kp * e + kd * de + ki * ie
17     tau = saturar(tau, tau_lim)
18     return tau, ie, ea
19
20
21 def pidg_vel(q, qp, qp_des, tau_lim, ie, ea, dt, kp_vec, ki_vec, kd_vec):
```

```
22      tau_pid, ie, ea = pid_vel(qp, qp_des, tau_lim, ie, ea, dt, kp_vec, ...
            ki_vec, kd_vec)
23
24      L2  = 0.25
25      lc2 = 0.12228
26      lc3 = 0.10860
27      g   = 9.81
28      m2  = 0.448505
29      m3  = 0.7157
30
31      aux1 = (m2 * g * lc2 * np.cos(q[1, 0]) +
32              m3 * g * L2  * np.cos(q[1, 0]) +
33              m3 * g * lc3 * np.cos(q[1, 0] + q[2, 0]))
34      aux2 = m3 * g * lc3 * np.cos(q[1, 0] + q[2, 0])
35
36      gravedad = np.array([[0.0], [aux1], [aux2]], dtype=float)
37
38      tau = tau_pid + gravedad
39      tau = saturar(tau, tau_lim)
40      return tau, ie, ea
41
42
43  def ismc_vel(qp, qp_des, m, k1, k2, int_sigma, e_prev, dt, tau_lim):
44      m  = np.array(m,  dtype=float).reshape((3,1))
45      k1 = np.array(k1, dtype=float).reshape((3,1))
46      k2 = np.array(k2, dtype=float).reshape((3,1))
47
48      e  = qp_des - qp
49      ep = (e - e_prev) / dt
50
51      sigma = ep + m * e
52      int_sigma = int_sigma + sigma * dt
53
54      sw = np.sign(sigma + k2 * int_sigma)
55      u = k2 * sigma + k1 * sw
56
57      u = saturar(u, tau_lim)
58      return u, e, sigma, int_sigma, e
```

Capítulo 5

Resultados

En este capítulo se analiza el comportamiento del manipulador al implementar los controladores propuestos. Las pruebas experimentales consisten en programar una trayectoria en el plano de la imagen, con el propósito de que el efector final la siga, minimizando el error de seguimiento. A continuación se presentan los resultados de las pruebas empleando 2 trayectorias diferentes, en adelante se hará referencia a la trayectoria con forma de corona como "trayectoria A", y la trayectoria con forma de cuadrado como "trayectoria B".

5.1. Resultados para el seguimiento de la trayectoria A con el controlador PID+G

En la Figura 5.1 se presenta la trayectoria de referencia con forma de corona definida en el plano de la imagen, así como la trayectoria efectivamente seguida por el efector final al implementar el controlador PID+G. Los resultados evidencian que el sistema es capaz de reproducir satisfactoriamente la geometría propuesta, manteniendo una correspondencia adecuada entre la trayectoria deseada y la trayectoria ejecutada por la cámara. En la representación gráfica, la línea roja punteada corresponde a la trayectoria de referencia en el plano imagen, mientras que la trayectoria en tonos azul y amarillo representa el movimiento seguido por la cámara; la transición cromática permite visualizar la evolución temporal del desplazamiento.

Asimismo, la trayectoria evaluada constituye un escenario de seguimiento particularmente demandante debido a las variaciones abruptas presentes en su geometría, especialmente en

las regiones correspondientes a los picos de la corona. Estas discontinuidades locales implican cambios rápidos en la dirección de movimiento y generan mayores exigencias dinámicas sobre el sistema de control. A pesar de ello, el controlador PID+G logra conservar la estabilidad del movimiento y mantener errores de seguimiento acotados, lo que sugiere una adecuada capacidad de compensación frente a las no linealidades dinámicas del manipulador.

Por otra parte, la Figura 5.2 presenta las señales de control asociadas a los tres grados de libertad del robot manipulador. Se observa que los torques articulares se encuentran aproximadamente en un intervalo comprendido entre -8 Nm y 1 Nm, comportamiento que refleja el esfuerzo requerido para ejecutar una trayectoria con cambios bruscos de curvatura. En particular, la articulación 2 presenta las mayores magnitudes de torque, lo cual resulta consistente con la configuración mecánica del manipulador, ya que dicha articulación soporta el peso combinado de los eslabones subsecuentes y contribuye de manera significativa al movimiento global del sistema. Este comportamiento confirma que la carga dinámica no se distribuye uniformemente entre las articulaciones y que el controlador incrementa la acción de control en los grados de libertad con mayor demanda mecánica. En la gráfica, el torque τ_1 correspondiente a la articulación 1 se representa en color azul, τ_2 en color naranja y τ_3 en color amarillo.

Posteriormente, la Figura 5.3 muestra el error individual asociado a cada característica visual empleada durante el proceso de servovisión. El análisis independiente de cada punto permite evaluar de manera específica la contribución de cada característica al error total del sistema, proporcionando una interpretación más detallada del desempeño del controlador visual. Los resultados indican que el error permanece en un rango aproximado entre 0 y 12 %, lo cual puede considerarse aceptable dadas las limitaciones cinemáticas del robot manipulador utilizado. Es importante destacar que el sistema no posee control explícito de orientación, por lo que no es posible garantizar una coincidencia exacta entre la orientación deseada y la orientación real de la cámara durante toda la trayectoria. En consecuencia, parte del error observado se asocia directamente con dicha restricción estructural y no exclusivamente con deficiencias del controlador.

De igual forma, la Figura 5.4 presenta la comparación entre las velocidades articulares deseadas y las velocidades reales obtenidas durante la ejecución del controlador PID+G. Se aprecia que las señales reales reproducen adecuadamente la tendencia de las referencias deseadas, manteniendo un comportamiento dinámico consistente incluso ante cambios abruptos de velocidad requeridos por la trayectoria con forma de corona. Las velocidades articulares

oscilan aproximadamente entre -3 y 3 rad/s, lo que evidencia la necesidad de respuestas dinámicas rápidas para seguir correctamente la trayectoria propuesta. Además, la similitud entre las curvas deseadas y reales sugiere que el controlador proporciona una capacidad adecuada de seguimiento dinámico y contribuye a reducir desfases significativos entre referencia y respuesta.

Finalmente, la Figura 5.5 ilustra el error de velocidad correspondiente a cada articulación del manipulador. Se observa que las mayores desviaciones se concentran principalmente en las articulaciones q_2 y q_3, comportamiento atribuible a la mayor sensibilidad dinámica de dichos grados de libertad y al acoplamiento mecánico existente entre las articulaciones del sistema. En particular, la articulación q_2 se encuentra sometida a mayores esfuerzos gravitacionales y dinámicos, mientras que la articulación q_3 presenta variaciones más pronunciadas debido a su participación directa en los movimientos de ajuste fino del efector final. No obstante, a pesar de estas variaciones, los errores permanecen acotados y no comprometen la estabilidad general del sistema ni la capacidad de seguimiento de la trayectoria visual, lo que confirma la efectividad del controlador PID+G en tareas de servovisión robótica con trayectorias complejas.

5.2. Resultados para el seguimiento de trayectoria A con el controlador PID

En la Figura 5.6 se presenta la trayectoria de referencia programada en el plano de la imagen, así como la trayectoria efectivamente seguida por la cámara al implementar el controlador PID. Los resultados muestran un buen desempeño del sistema ya que es capaz de reproducir de manera adecuada la trayectoria deseada, manteniendo una correspondencia aceptable entre la trayectoria de referencia y la trayectoria ejecutada. La curva azul representa el desplazamiento realizado por la cámara durante la tarea de seguimiento visual, evidenciando un comportamiento continuo y estable a lo largo de la trayectoria.

Asimismo, la trayectoria con forma de corona representa un caso de seguimiento complejo debido a las variaciones abruptas presentes en su geometría, particularmente en las regiones con cambios pronunciados de curvatura. Estas condiciones incrementan las exigencias dinámicas del sistema, ya que requieren modificaciones rápidas en las velocidades articulares y en las acciones de control para conservar el seguimiento de la referencia. A pesar de ello, el

controlador PID mantiene un desempeño estable y permite que el manipulador siga la trayectoria sin presentar divergencias significativas, lo que demuestra una adecuada capacidad de respuesta frente a cambios en la referencia visual.

Por otra parte, en la Figura 5.7 se muestra la señal de control correspondiente a los tres grados de libertad del robot manipulador. Se observa que los torques articulares oscilan aproximadamente entre $1\,\mathrm{Nm}$ y $-8\,\mathrm{Nm}$, magnitudes que reflejan el esfuerzo mecánico requerido para ejecutar la trayectoria propuesta. En particular, la articulación 2, representada en color naranja, presenta las mayores demandas de torque, comportamiento consistente con la estructura dinámica del manipulador, ya que esta articulación soporta el peso de los eslabones subsecuentes y participa de manera importante en la generación del movimiento global del sistema. Este resultado evidencia que la distribución de esfuerzos no es uniforme entre las articulaciones y confirma que el seguimiento de trayectorias con cambios bruscos de dirección incrementa significativamente la acción de control requerida.

Posteriormente, en la Figura 5.8 se presenta la evolución del error porcentual asociado a cada una de las características visuales utilizadas durante el proceso de servovisión. El análisis individual de cada punto permite identificar la contribución específica de cada característica al error total del sistema, proporcionando una evaluación más detallada del desempeño del controlador. Los resultados muestran que el error permanece aproximadamente en un intervalo entre $0\,\%$ y $13\,\%$, observándose una tendencia de convergencia durante el seguimiento. Aunque el error no converge exactamente a cero, este comportamiento resulta coherente con las limitaciones cinemáticas del manipulador, particularmente debido a la ausencia de control explícito de orientación. En consecuencia, parte de las desviaciones observadas se relaciona directamente con las restricciones estructurales del sistema y no únicamente con el desempeño del controlador PID.

De igual forma, la Figura 5.9 presenta una comparación entre las velocidades articulares deseadas y las velocidades reales obtenidas en cada articulación del manipulador. Se aprecia que las señales reales reproducen adecuadamente la tendencia de las referencias deseadas, manteniendo un comportamiento dinámico estable incluso en las regiones donde la trayectoria exige cambios rápidos de dirección. Las velocidades articulares oscilan aproximadamente entre $-3\,\mathrm{rad/s}$ y $2.5\,\mathrm{rad/s}$, lo cual denota la capacidad del sistema para responder ante exigencias dinámicas relativamente altas. Además, la cercanía entre las velocidades deseadas y reales indica que el controlador PID logra mantener una regulación adecuada, reduciendo desfases significativos entre la referencia y la respuesta del sistema.

Finalmente, en la Figura 5.10 se muestra el error de velocidad obtenido durante el seguimiento de la trayectoria. Los resultados indican que el error permanece acotado aproximadamente entre $-0.5\,\text{rad/s}$ y $0.5\,\text{rad/s}$, lo cual sugiere que el controlador mantiene una respuesta dinámica estable durante toda la ejecución de la trayectoria. Las variaciones observadas se incrementan principalmente en las regiones donde la trayectoria presenta cambios bruscos de curvatura, debido a la necesidad de generar aceleraciones articulares más rápidas. Sin embargo, dichas desviaciones no comprometen la estabilidad general del sistema ni la capacidad de seguimiento de la trayectoria visual, confirmando que el controlador PID proporciona un desempeño satisfactorio para tareas de servovisión robótica con trayectorias complejas.

5.2.1. Resultados para el seguimiento de la trayectoria A con el controlador ISMC

En la Figura 5.11 se presenta la trayectoria de referencia con forma de corona en color rojo, junto con la trayectoria efectivamente seguida por la cámara en color azul mediante la implementación del controlador por modos deslizantes integral (ISMC). A diferencia de los resultados obtenidos con los controladores PID y PID+G, en este caso el sistema no logra mantener un seguimiento satisfactorio de la trayectoria durante toda la ejecución. Se observa que, aproximadamente después de los 6 segundos, la cámara pierde la referencia visual y no consigue recuperarla, provocando la interrupción de la tarea de seguimiento.

Asimismo, este comportamiento evidencia que el controlador ISMC presenta limitaciones importantes frente a trayectorias con variaciones geométricas abruptas y cambios rápidos de dirección, como ocurre en la trayectoria tipo corona. La pérdida de referencia sugiere que la acción de control no logra compensar adecuadamente las exigencias dinámicas y las perturbaciones asociadas al movimiento del manipulador. En consecuencia, el sistema pierde estabilidad en el seguimiento visual y el efector final deja de responder correctamente a la referencia deseada. Estos resultados indican que, bajo las condiciones evaluadas, el controlador ISMC presenta un desempeño inferior respecto a los controladores PID y PID+G para tareas de servovisión con trayectorias complejas.

Por otra parte, en la Figura 5.12 se muestran las variaciones de las señales de control correspondientes a los tres grados de libertad del manipulador. Durante la etapa inicial del movimiento, los torques evolucionan de acuerdo con las exigencias dinámicas impuestas por la trayectoria de referencia, evidenciando que el controlador intenta compensar las aceleraciones

y cambios de dirección requeridos. Sin embargo, entre los segundos 4 y 6 se observa una disminución progresiva de las señales de control, coincidiendo con la pérdida de la referencia visual. Posteriormente, los torques tienden a valores cercanos a cero debido a que el efector final pierde estabilidad y permanece prácticamente inmóvil tras la caída del sistema de seguimiento. Este comportamiento confirma que, una vez perdida la referencia visual, el controlador deja de generar acciones de control significativas.

Posteriormente, en la Figura 5.13 se presenta el error de posición asociado a cada uno de los puntos característicos utilizados en el proceso de servovisión. Los resultados muestran errores comprendidos aproximadamente entre 0 y 30 píxeles, magnitudes considerablemente superiores a las obtenidas con los controladores PID y PID+G. Este incremento en el error refleja la incapacidad del controlador ISMC para mantener la alineación adecuada entre las características visuales deseadas y las características observadas durante toda la trayectoria. Además, el aumento progresivo del error antes de la pérdida de referencia evidencia una degradación gradual en el desempeño dinámico del sistema, particularmente en las regiones de mayor complejidad geométrica de la trayectoria.

De igual forma, la Figura 5.14 muestra la comparación entre las velocidades articulares reales y las velocidades deseadas. Durante los primeros segundos de operación se aprecia una correspondencia parcial entre ambas señales, lo que indica que inicialmente el controlador es capaz de reproducir el comportamiento dinámico requerido por la trayectoria. No obstante, después del segundo 4 las velocidades articulares comienzan a decrecer progresivamente hasta converger a valores cercanos a cero. Este comportamiento ocurre como consecuencia directa de la pérdida de la referencia visual y de la posterior inmovilización del efector final. Por lo tanto, el sistema deja de ejecutar el seguimiento dinámico de la trayectoria y permanece en un estado estacionario no deseado.

Finalmente, en la Figura 5.15 se presenta el error de velocidad asociado a cada articulación del manipulador. Durante la fase inicial, la mayor desviación se registra en la tercera articulación, alcanzando valores aproximados entre $1.5\,\mathrm{rad/s}$ y $-2\,\mathrm{rad/s}$. Este comportamiento indica que la articulación q_3 presenta una mayor sensibilidad dinámica frente a las exigencias de la trayectoria y a las acciones de control generadas por el controlador ISMC. Posteriormente, a partir del segundo 4, el error disminuye progresivamente hasta aproximarse a cero. Sin embargo, esta reducción no representa una mejora en el desempeño del controlador, sino que se debe a la pérdida de movimiento del manipulador tras la pérdida de referencia visual.

En consecuencia, los resultados demuestran que el controlador ISMC no logra garantizar un seguimiento robusto y estable para trayectorias visuales complejas, especialmente en escenarios donde existen cambios bruscos de curvatura y elevadas demandas dinámicas.

5.2.2. Resultados para el seguimiento de trayectoria B con el controlador PID+G

La Figura 5.16 presenta la trayectoria cuadrada programada en el plano de la imagen, junto con la trayectoria efectivamente seguida por el efector final mediante la implementación del controlador PID+G. La trayectoria de referencia se representa mediante una línea roja, mientras que la trayectoria seguida por la cámara se ilustra con una línea cuyo gradiente de color, de azul a amarillo, permite identificar la evolución temporal del movimiento. Los resultados muestran que el efector final es capaz de seguir satisfactoriamente la trayectoria propuesta, manteniendo una correspondencia adecuada entre la referencia y la trayectoria ejecutada.

Asimismo, la trayectoria cuadrada representa un caso de seguimiento exigente debido a la presencia de cambios abruptos de dirección en las esquinas de la figura geométrica. Estas discontinuidades generan transiciones rápidas en las velocidades y aceleraciones articulares, incrementando las demandas dinámicas sobre el sistema de control. A pesar de estas condiciones, el controlador PID+G mantiene la estabilidad del movimiento y permite que el manipulador conserve un seguimiento adecuado de la trayectoria visual, evidenciando una capacidad apropiada para compensar las perturbaciones dinámicas asociadas a cambios bruscos de curvatura.

Por otra parte, la salida de control se presenta en la Figura 5.17. Se observa que la mayor exigencia dinámica recae nuevamente sobre la articulación 2, cuyos valores de torque oscilan aproximadamente entre $-8\,\text{Nm}$ y $-1\,\text{Nm}$. Este comportamiento resulta consistente con la estructura mecánica del manipulador, ya que dicha articulación soporta el peso de los eslabones subsecuentes y participa de manera significativa en la generación del movimiento global del sistema. En la articulación 3, los torques varían aproximadamente entre $-2\,\text{Nm}$ y $0\,\text{Nm}$, mientras que en la articulación 1 los valores se encuentran en un rango cercano a $-2\,\text{Nm}$ y $2\,\text{Nm}$. Estas diferencias evidencian que la distribución de esfuerzos no es uniforme y que las articulaciones centrales presentan mayores demandas de control .

Además, en la Figura 5.18 se presenta el error de seguimiento asociado a cada uno de los

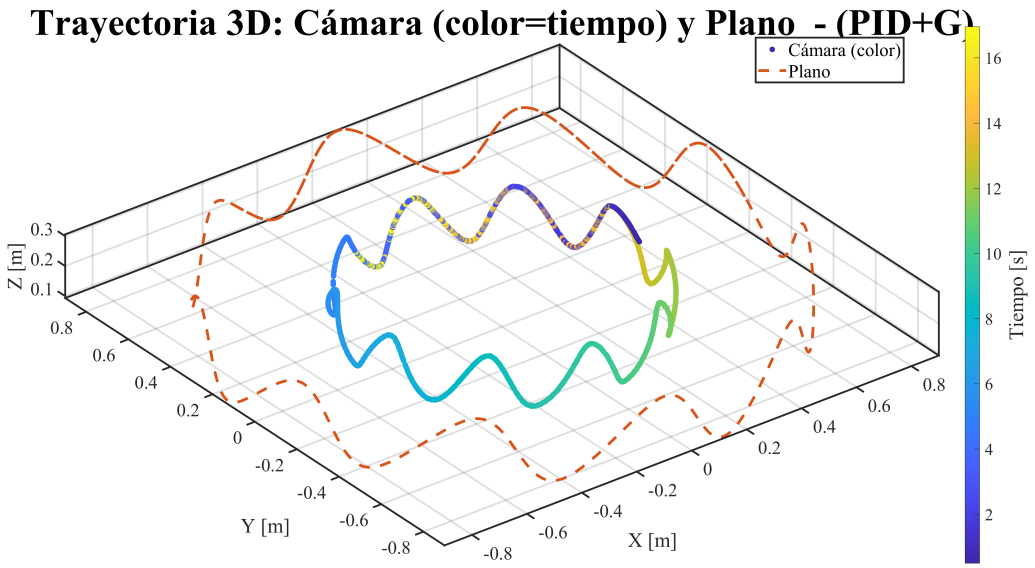

Figura 5.1: Seguimiento de trayectoria corona con PID+G

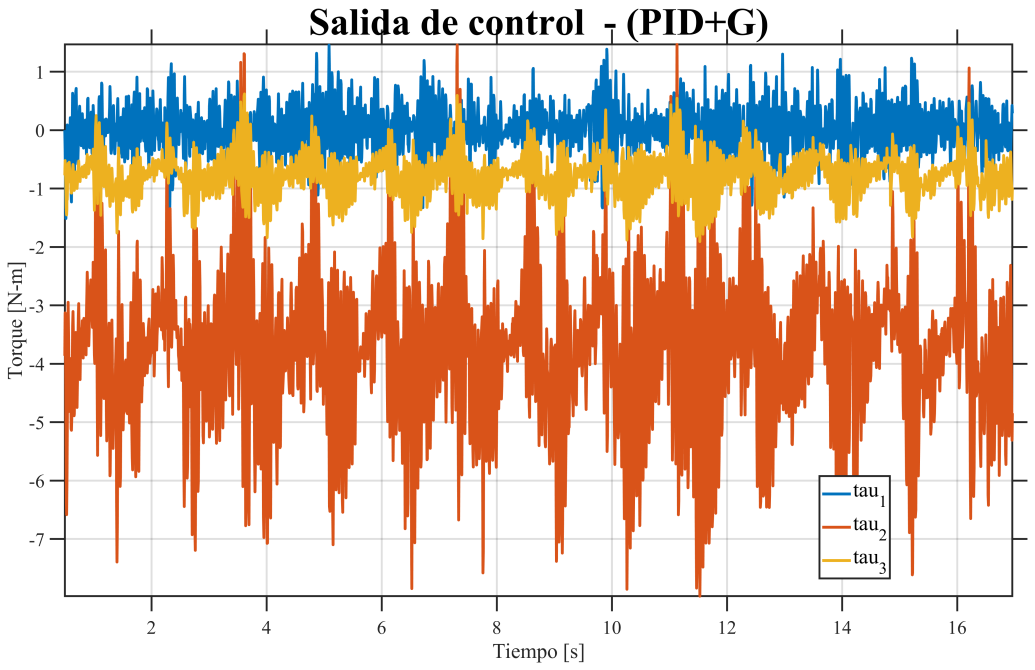

Figura 5.2: Salida de control - Torque PID+G

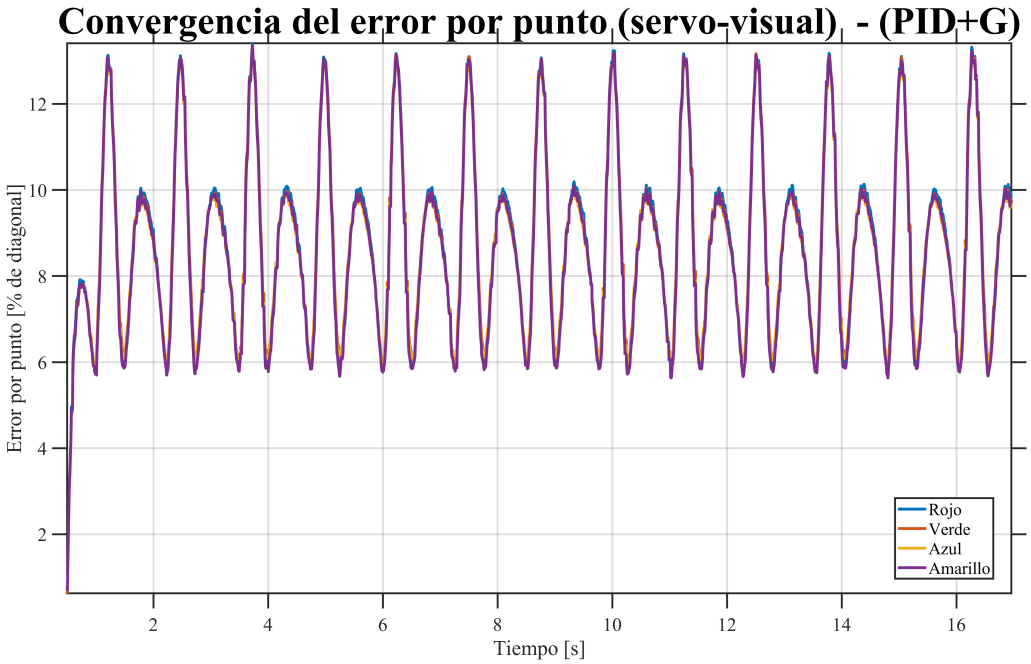

Figura 5.3: Error de posición por punto -PID+G

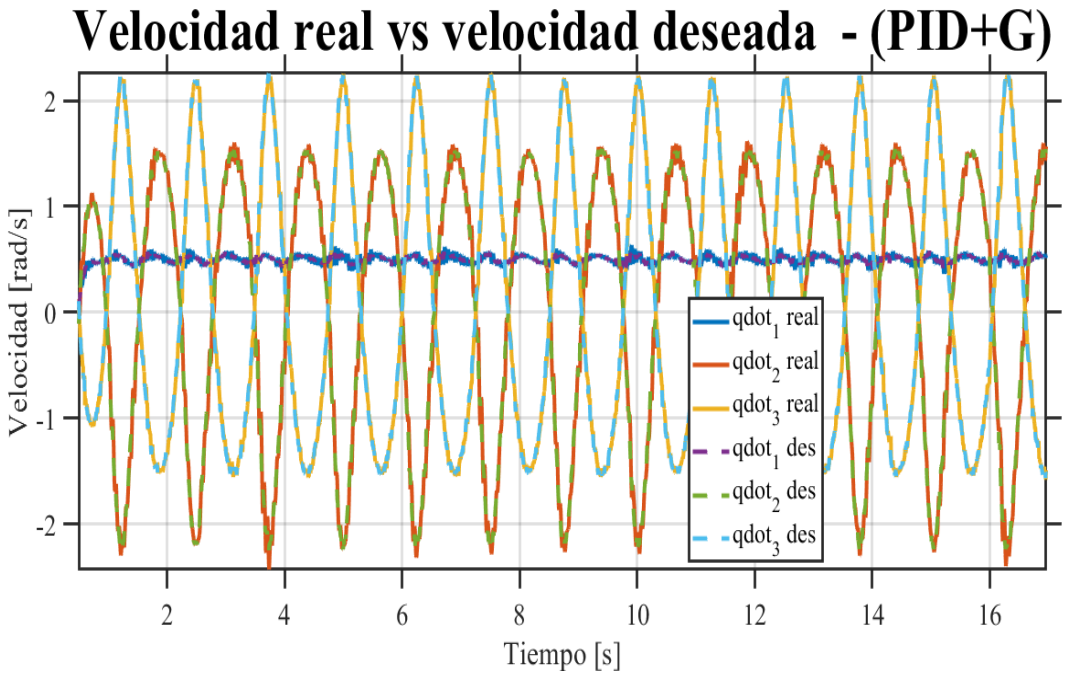

Figura 5.4: Velocidad real vs velocidad deseada -PID+G

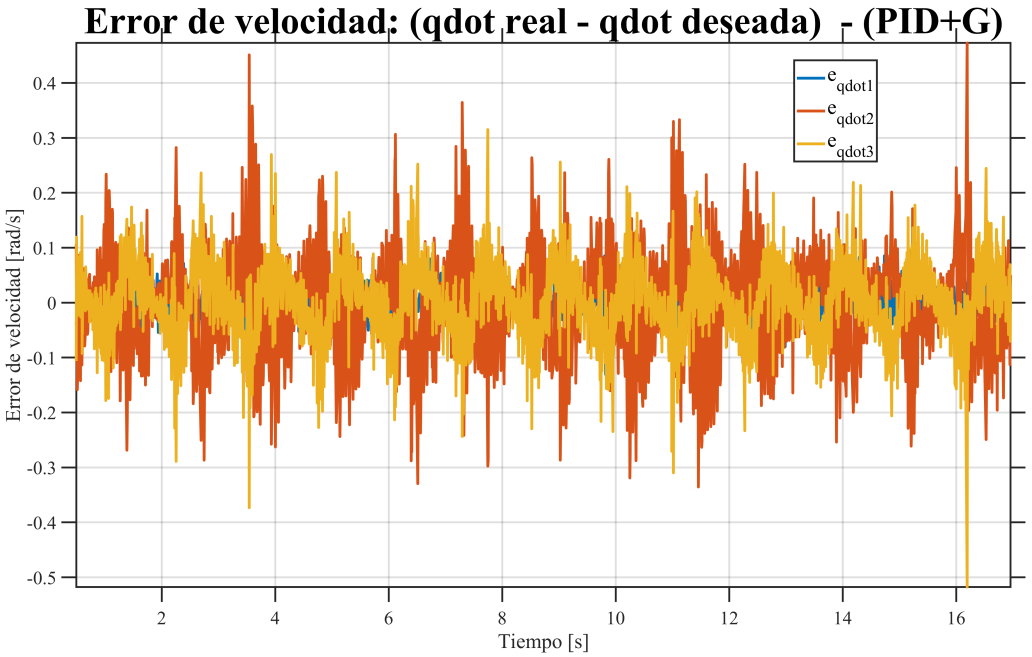

Figura 5.5: Error de velocidad articular-PID+G

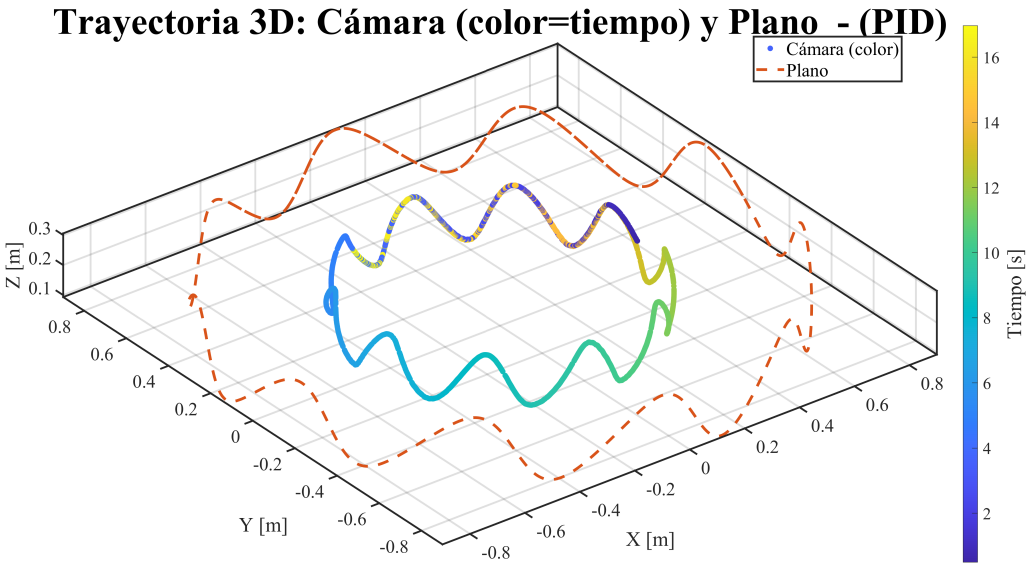

Figura 5.6: Seguimiento de trayectoria corona con PID

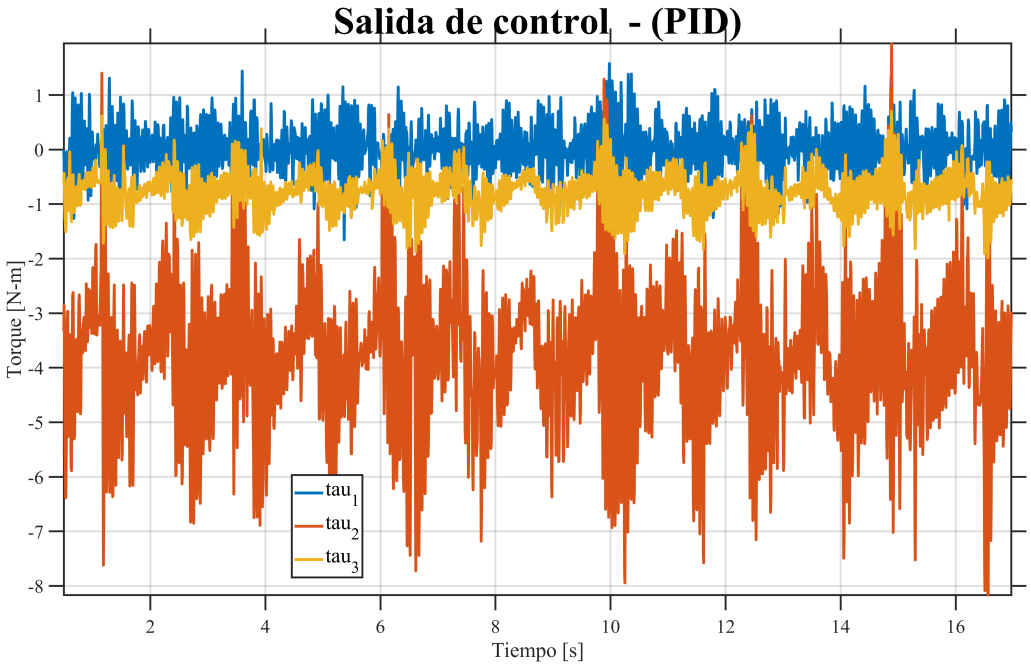

Figura 5.7: Salida de control - Torque PID

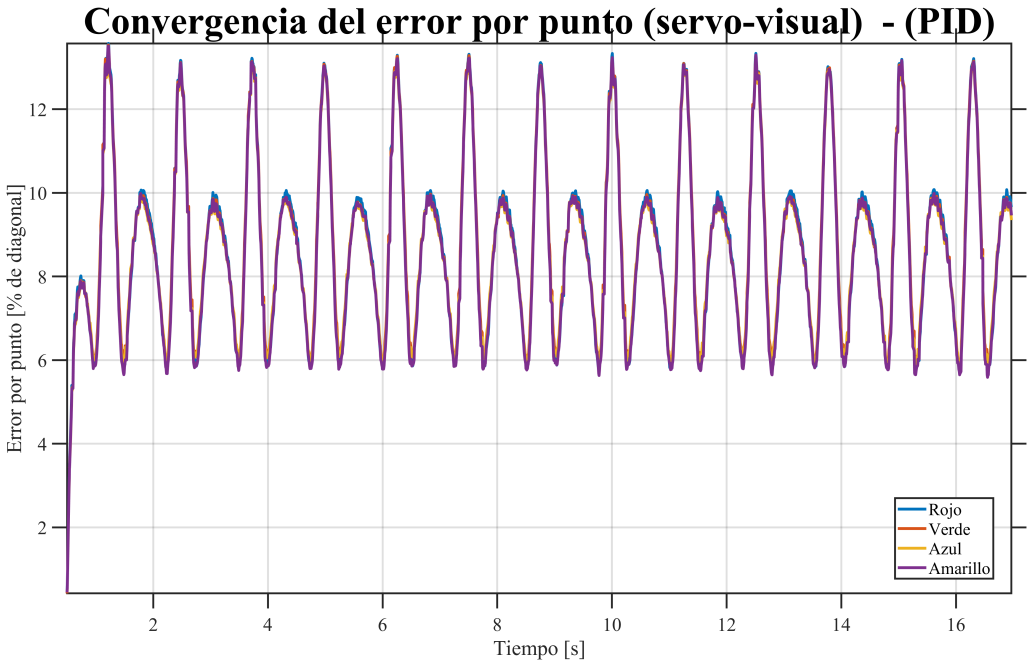

Figura 5.8: Error de posición por punto- PID

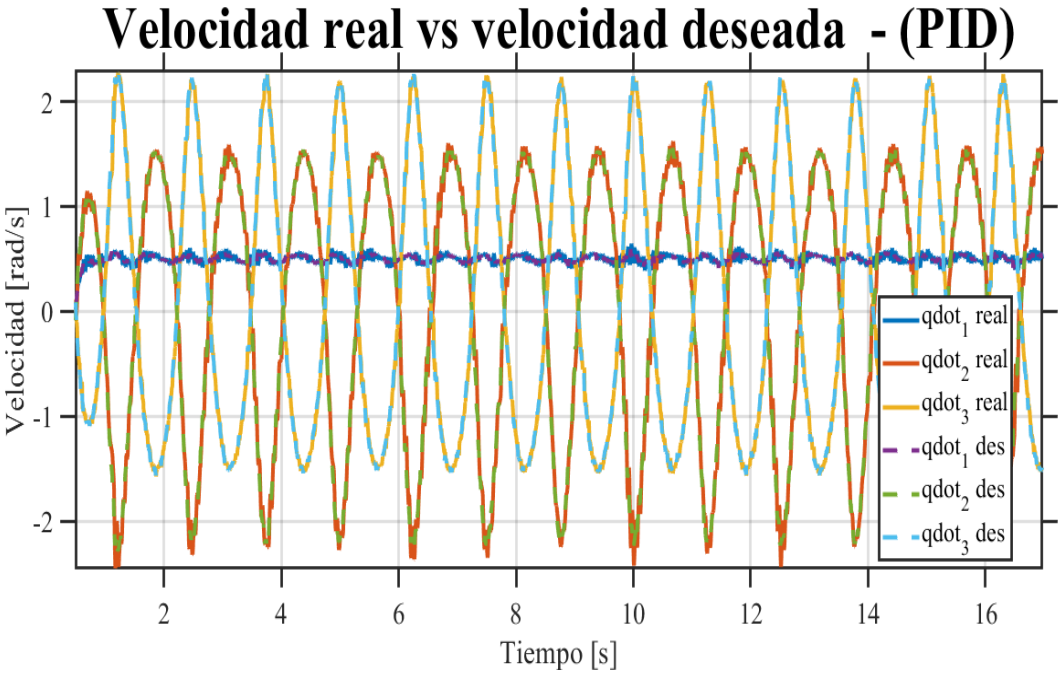

Figura 5.9: Velocidad real vs velocidad deseada- PID

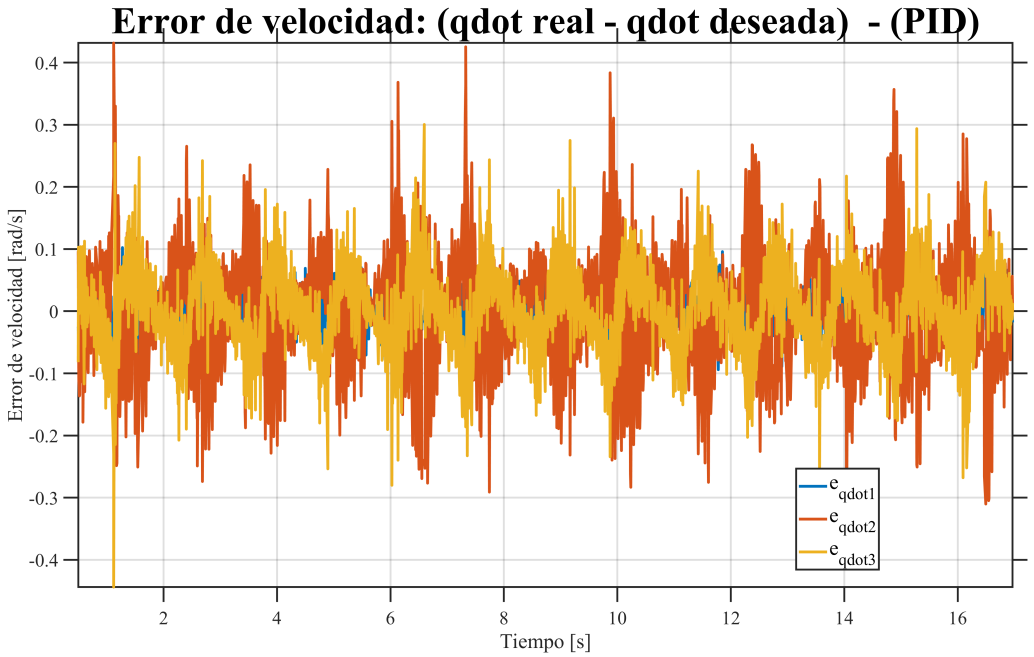

Figura 5.10: Error de velocidad articular -PID

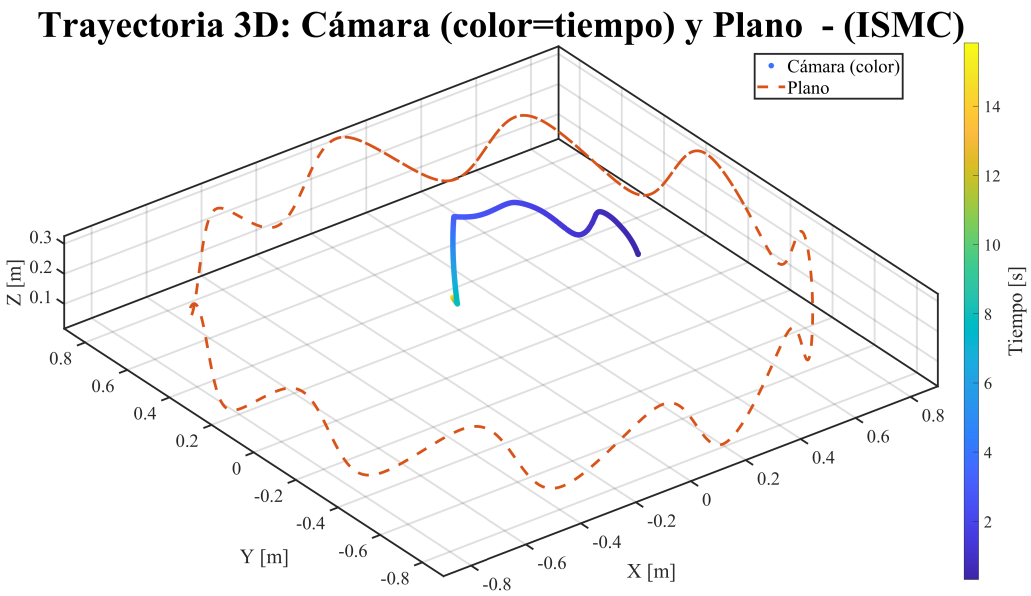

Figura 5.11: Seguimiento de trayectoria corona con ISMC

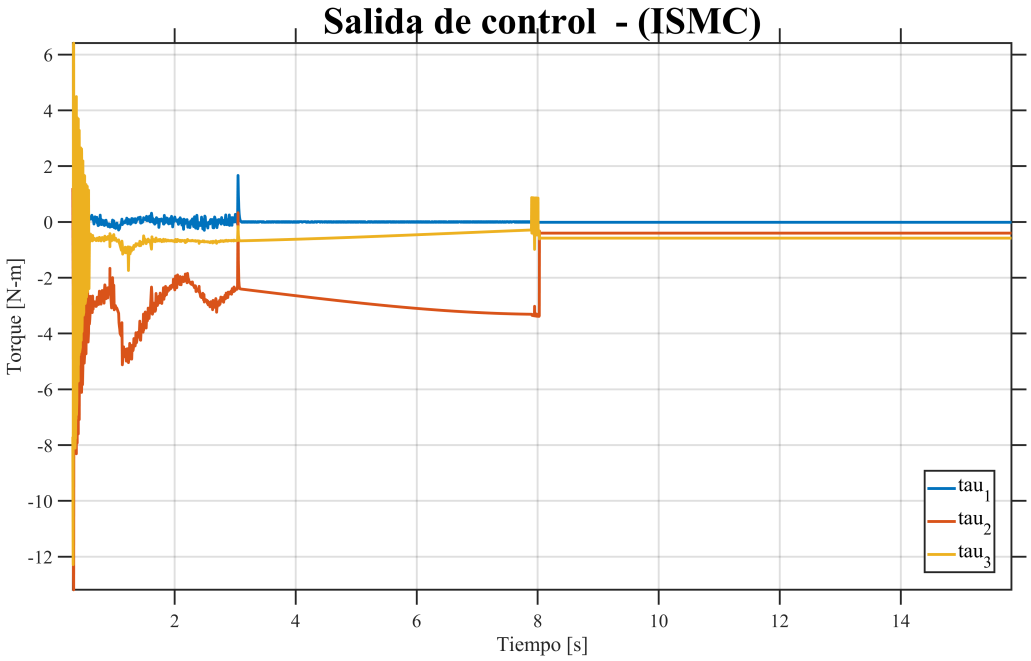

Figura 5.12: Salida de control - Torque ISMC

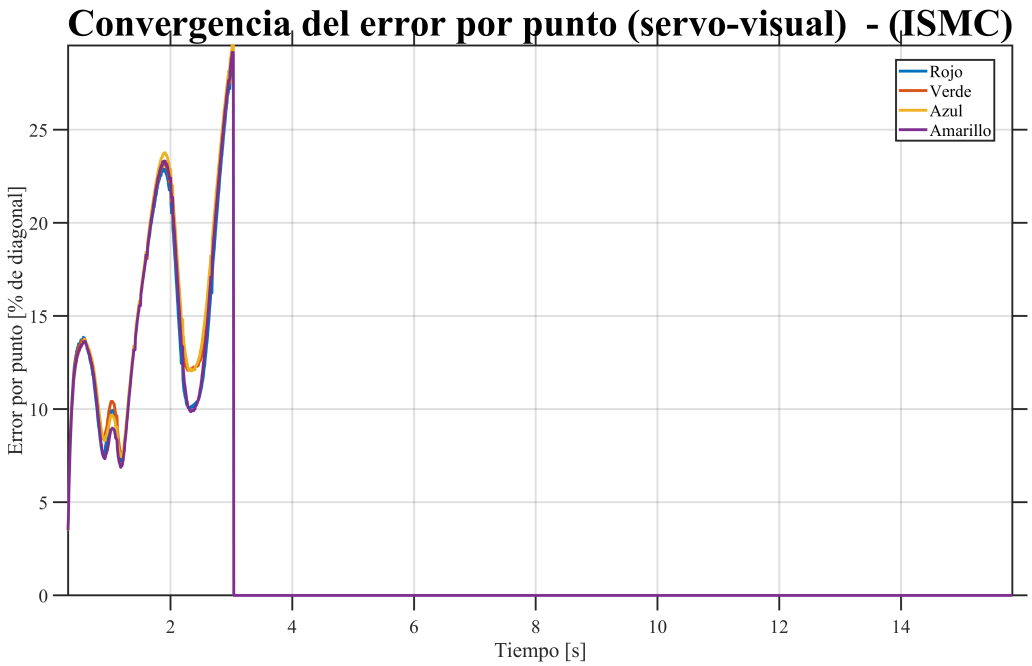

Figura 5.13: Error de posición por punto con ISMC

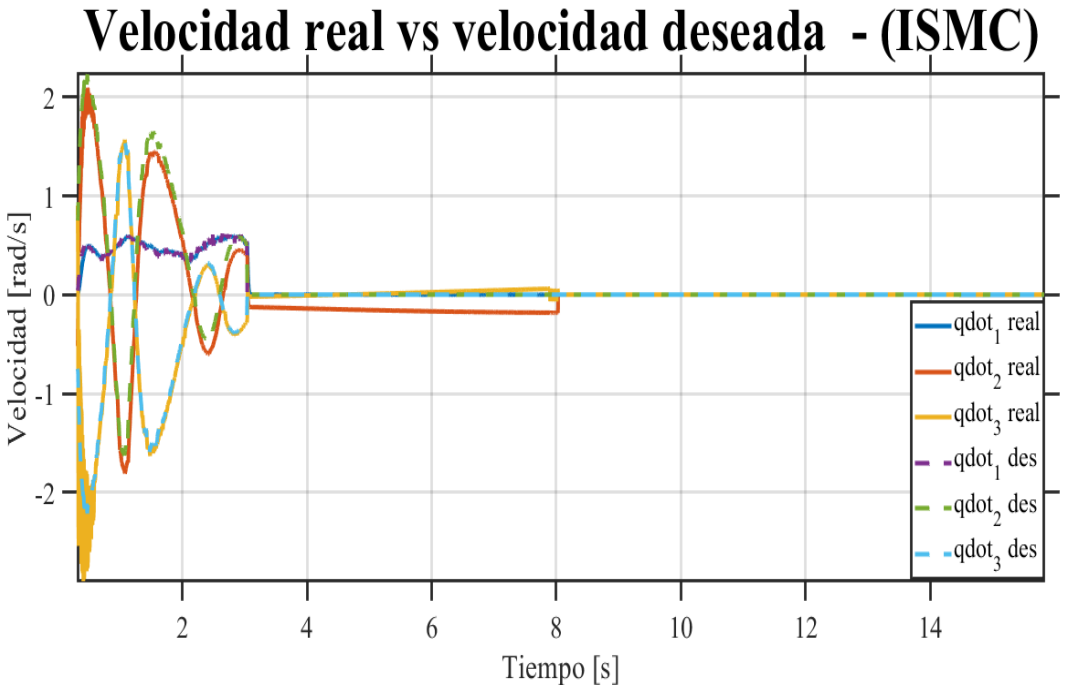

Figura 5.14: Velocidad real vs velocidad deseada ISMC

puntos característicos reconocidos en la imagen. Este análisis individual permite identificar la contribución específica de cada característica visual al error total del sistema. Los resultados muestran que el mayor error se registra en el seguimiento del punto azul, alcanzando aproximadamente un 16.5 %, mientras que el menor error corresponde al punto rojo, con un valor máximo cercano al 14 %. Aunque los errores no convergen exactamente a cero, los valores obtenidos pueden considerarse aceptables debido a las limitaciones cinemáticas del manipulador y a la complejidad geométrica de la trayectoria cuadrada. En particular, las mayores desviaciones tienden a presentarse en las esquinas de la trayectoria, donde los cambios abruptos de dirección incrementan las exigencias dinámicas del sistema de servovisión.

De igual forma, la Figura 5.19 muestra una comparación entre las velocidades articulares reales y las velocidades deseadas. Se aprecia que ambas señales presentan comportamientos similares durante la mayor parte del seguimiento, lo cual indica que el controlador PID+G logra reproducir adecuadamente la dinámica requerida por la trayectoria programada. Este resultado demuestra que el sistema mantiene una respuesta estable incluso en presencia de cambios bruscos asociados a la geometría cuadrada. No obstante, se observa que las mayores discrepancias se concentran en las articulaciones 2 y 3, donde el error alcanza valores aproximados entre 0 % y 16 %. Este comportamiento puede atribuirse a la mayor sensibilidad dinámica de dichos grados de libertad y al acoplamiento mecánico existente entre las articulaciones del manipulador.

Finalmente, la Figura 5.20 presenta el error de velocidad obtenido durante la ejecución de la trayectoria. Los resultados muestran que esta variable oscila aproximadamente entre $-0.3\,\text{rad/s}$ y $0.3\,\text{rad/s}$, manteniéndose acotada durante todo el seguimiento. Asimismo, se observa que las mayores desviaciones se concentran en las articulaciones 2 y 3, particularmente en los instantes donde la trayectoria presenta cambios bruscos de dirección. Sin embargo, dichas variaciones no comprometen la estabilidad general del sistema ni la capacidad de seguimiento visual del manipulador, lo que confirma la efectividad del controlador PID+G para tareas de servovisión robótica con trayectorias geométricamente complejas.

5.2.3. Resultados para el seguimiento de trayectoria B con el controlador PID

La Figura 5.21 presenta el seguimiento de la trayectoria tipo B mediante la implementación del controlador PID. Los resultados muestran un comportamiento muy similar al

obtenido con el controlador PID+G, ya que el sistema logra completar satisfactoriamente toda la trayectoria sin presentar pérdida de la referencia visual. La trayectoria seguida por el efector final mantiene una correspondencia adecuada con la trayectoria programada, lo que evidencia que el controlador PID posee la capacidad de conservar la estabilidad del sistema y garantizar el seguimiento visual durante toda la ejecución de la tarea.

Asimismo, la trayectoria tipo B implica cambios continuos de dirección y variaciones de curvatura que generan exigencias dinámicas importantes sobre el manipulador. A pesar de estas condiciones, el controlador mantiene una respuesta estable y evita divergencias significativas durante el seguimiento. Este comportamiento sugiere que la estrategia de control implementada es capaz de compensar adecuadamente las variaciones dinámicas producidas por el movimiento del manipulador y las perturbaciones asociadas al proceso de servovisión.

Por otra parte, en la Figura 5.22 se presenta la salida de control correspondiente a los tres grados de libertad del robot manipulador. Se observa que la mayor variación de torque se concentra en la articulación 2, con valores comprendidos aproximadamente entre $-10\,\mathrm{Nm}$ y $0\,\mathrm{Nm}$. Este resultado concuerda con la configuración dinámica del manipulador, ya que dicha articulación soporta el peso de los eslabones subsecuentes y contribuye significativamente a la generación del movimiento global del sistema.
Posteriormente, la segunda mayor variación se registra en la articulación 1, cuyos valores oscilan aproximadamente entre $-2\,\mathrm{Nm}$ y $2\,\mathrm{Nm}$. Finalmente, la articulación 3 presenta las menores demandas de torque, con variaciones comprendidas entre $-2\,\mathrm{Nm}$ y $0\,\mathrm{Nm}$. Estas diferencias evidencian que las exigencias dinámicas no se distribuyen uniformemente entre las articulaciones y que el controlador incrementa la acción de control principalmente en los grados de libertad con mayor carga mecánica.

Además, el error de seguimiento asociado a las características visuales se presenta en la Figura 5.23. El análisis individual de cada punto característico permite identificar la contribución específica de cada elemento visual al error total del sistema. Los resultados muestran que los mayores errores corresponden al seguimiento de los puntos azul y verde, mientras que el menor error se obtiene en el seguimiento del punto rojo. Este comportamiento puede atribuirse a la posición relativa de las características dentro del plano imagen y a las variaciones de perspectiva generadas durante el movimiento del efector final. Asimismo, las desviaciones observadas tienden a incrementarse en las regiones donde la trayectoria presenta mayores cambios de dirección, debido a las mayores exigencias dinámicas requeridas para mantener el seguimiento visual.

De igual forma, la Figura 5.24 muestra una comparación entre las velocidades articulares reales y las velocidades deseadas. Se aprecia que ambas señales presentan comportamientos altamente similares durante la mayor parte del seguimiento, resultado que coincide con el desempeño observado en el controlador PID+G. Las velocidades articulares oscilan aproximadamente entre $-2\,\mathrm{rad/s}$ y $2\,\mathrm{rad/s}$, evidenciando que el sistema responde adecuadamente a las demandas dinámicas impuestas por la trayectoria tipo B. Además, la cercanía entre las curvas reales y deseadas indica que el controlador PID logra minimizar desfases importantes y mantener un seguimiento dinámico estable en cada articulación del manipulador.

Finalmente, el error de velocidad se presenta en la Figura 5.25. Los resultados muestran valores comprendidos aproximadamente entre $-0.5\,\mathrm{rad/s}$ y $0.5\,\mathrm{rad/s}$, manteniéndose acotados durante toda la ejecución de la trayectoria. Se observa que las mayores desviaciones se concentran en las articulaciones 2 y 3, comportamiento asociado a la mayor sensibilidad dinámica de dichos grados de libertad y al acoplamiento mecánico existente entre las articulaciones del manipulador.

No obstante, las variaciones observadas no comprometen la estabilidad general del sistema ni la capacidad de seguimiento visual, lo que confirma que el controlador PID proporciona un desempeño adecuado para tareas de servovisión robótica con trayectorias geométricamente complejas.

5.2.4. Resultados para el seguimiento de trayectoria B con el controlador ISMC

La Figura 5.26 presenta el seguimiento de la trayectoria tipo B utilizando el controlador por modos deslizantes integral (ISMC). Los resultados muestran que el sistema no logra completar satisfactoriamente la trayectoria programada, ya que, aunque durante la etapa inicial el seguimiento presenta un comportamiento aceptable, a partir de aproximadamente los 4 segundos se pierde la referencia visual. En consecuencia, el efector final deja de seguir la trayectoria deseada y el sistema entra en una condición de inestabilidad que impide recuperar el seguimiento.

Asimismo, este comportamiento evidencia que el controlador ISMC presenta limitaciones importantes frente a trayectorias con cambios continuos de curvatura y variaciones dinámi-

cas significativas, como ocurre en la trayectoria tipo B. Durante los primeros instantes de operación, el controlador es capaz de generar acciones de control suficientes para mantener parcialmente el seguimiento; sin embargo, conforme aumentan las exigencias dinámicas del movimiento, las desviaciones acumuladas provocan la pérdida progresiva de la referencia visual. Estos resultados indican que el controlador no logra compensar adecuadamente las perturbaciones dinámicas y el acoplamiento mecánico existente entre las articulaciones del manipulador, especialmente en trayectorias que requieren movimientos rápidos y continuos del efector final.

Por otra parte, en la Figura 5.27 se presenta la salida de control correspondiente a los tres grados de libertad del robot manipulador. Se observan variaciones de torque comprendidas aproximadamente entre $-15\,\mathrm{Nm}$ y $10\,\mathrm{Nm}$, magnitudes considerablemente superiores a las registradas con los controladores PID y PID+G. Este comportamiento evidencia que el controlador ISMC genera acciones de control más agresivas con el propósito de compensar las elevadas demandas dinámicas impuestas por la trayectoria. Además, las mayores fluctuaciones se concentran en las articulaciones 2 y 3, las cuales presentan una mayor sensibilidad dinámica debido a la distribución de masas. Posteriormente, una vez que se pierde la referencia visual, los valores de torque convergen a cero en todas las articulaciones, debido a que el sistema deja de ejecutar acciones efectivas de seguimiento y el efector final permanece prácticamente inmóvil.

De igual forma, la Figura 5.29 muestra la comparación entre las velocidades articulares reales y las velocidades deseadas. A diferencia de los resultados obtenidos con los controladores PID y PID+G, en este caso se observan discrepancias significativas entre ambas señales desde etapas tempranas de la trayectoria. Estas diferencias evidencian que el controlador ISMC no consigue reproducir adecuadamente la dinámica requerida por el movimiento deseado, generándose desfases importantes entre la referencia y la respuesta real del sistema. Asimismo, después de la pérdida de referencia visual, las velocidades articulares disminuyen progresivamente hasta alcanzar valores cercanos a cero, reflejando la interrupción del movimiento del manipulador.

Posteriormente, el error de velocidad se presenta en la Figura 5.30. Los resultados muestran variaciones comprendidas aproximadamente entre $-2\,\mathrm{rad/s}$ y $1\,\mathrm{rad/s}$, valores considerablemente mayores a los obtenidos con los otros esquemas de control evaluados. El mayor error se registra en la articulación 3, lo cual sugiere que este grado de libertad presenta una mayor sensibilidad frente a las perturbaciones dinámicas y a las variaciones bruscas de

movimiento requeridas por la trayectoria. Además, las elevadas oscilaciones del error indican que el controlador no logra estabilizar adecuadamente la respuesta del sistema durante toda la tarea de seguimiento visual.

Finalmente, en la Figura 5.28 se presenta el error asociado al seguimiento de las características visuales utilizadas durante el proceso de servovisión. Se observa que la pérdida del seguimiento ocurre aproximadamente a partir del segundo 3, momento en el cual el error comienza a incrementarse de manera significativa. El mayor error corresponde al punto azul, con valores comprendidos aproximadamente entre 0 % y 25 %, mientras que el resto de las características presentan comportamientos similares aunque con menores magnitudes.

Este incremento progresivo del error confirma la degradación del desempeño del controlador conforme aumenta la complejidad dinámica de la trayectoria y evidencia la incapacidad del esquema ISMC para mantener una alineación estable entre las características visuales deseadas y las observadas.

Por otra parte, la Tabla 5.1 presenta un resumen comparativo del error porcentual de seguimiento obtenido para cada trayectoria evaluada en combinación con los distintos esquemas de control implementados. El análisis comparativo permite identificar diferencias significativas en el desempeño dependiendo tanto de la geometría de la trayectoria como de la estrategia de control utilizada. En general, las trayectorias con cambios abruptos de dirección y mayores exigencias dinámicas producen incrementos considerables en el error de seguimiento, particularmente cuando se emplea el controlador ISMC.

En consecuencia, los resultados experimentales confirman que la combinación de la trayectoria tipo A con los controladores PID y PID+G proporciona el mejor desempeño global en términos de exactitud y estabilidad de seguimiento. Ambos controladores muestran una mayor capacidad para compensar las no linealidades dinámicas del manipulador y mantener errores acotados durante la ejecución de trayectorias complejas.

En contraste, el controlador ISMC presenta mayores oscilaciones, pérdidas de referencia visual y errores de velocidad superiores, lo que evidencia un desempeño menos robusto bajo las condiciones evaluadas en este trabajo.

Tabla 5.1: Resumen comparativo del error de seguimiento por trayectoria y tipo de controlador

Trayectoria	**Controlador**	**Error de seguimiento (%)**
A (Corona)	PID / PID+G	0 – 13
B (Cuadrado)	PID / PID+G	0 – 16.5
A, B	ISMC	hasta 24 – 30

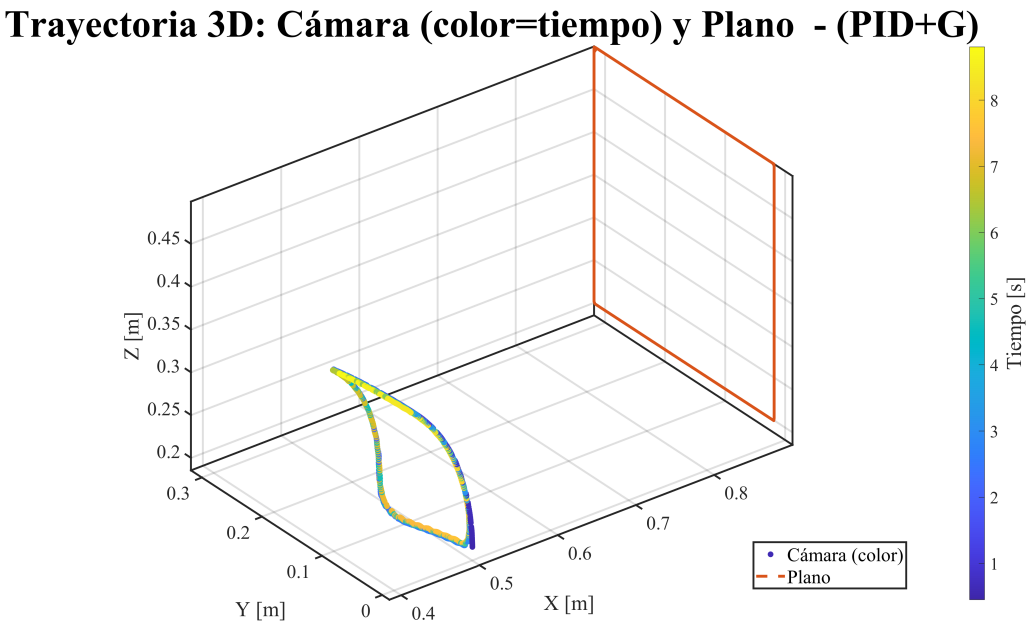

Figura 5.16: Seguimiento de trayectoria cuadrado con PID+G

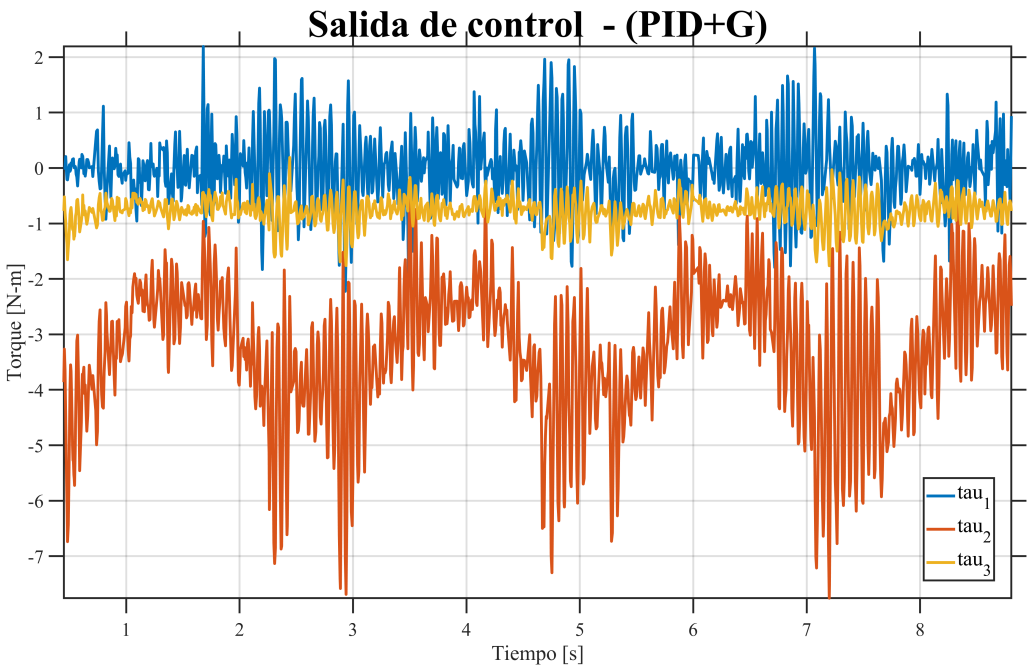

Figura 5.17: Salida de control - Torque PID+G

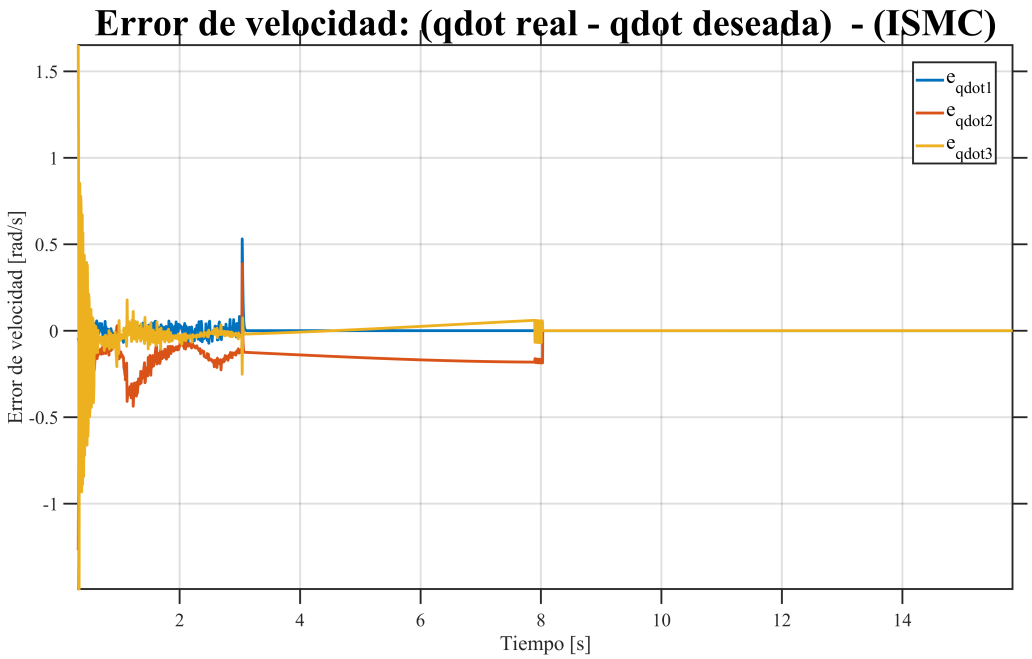

Figura 5.15: Error de velocidad articular con ISMC

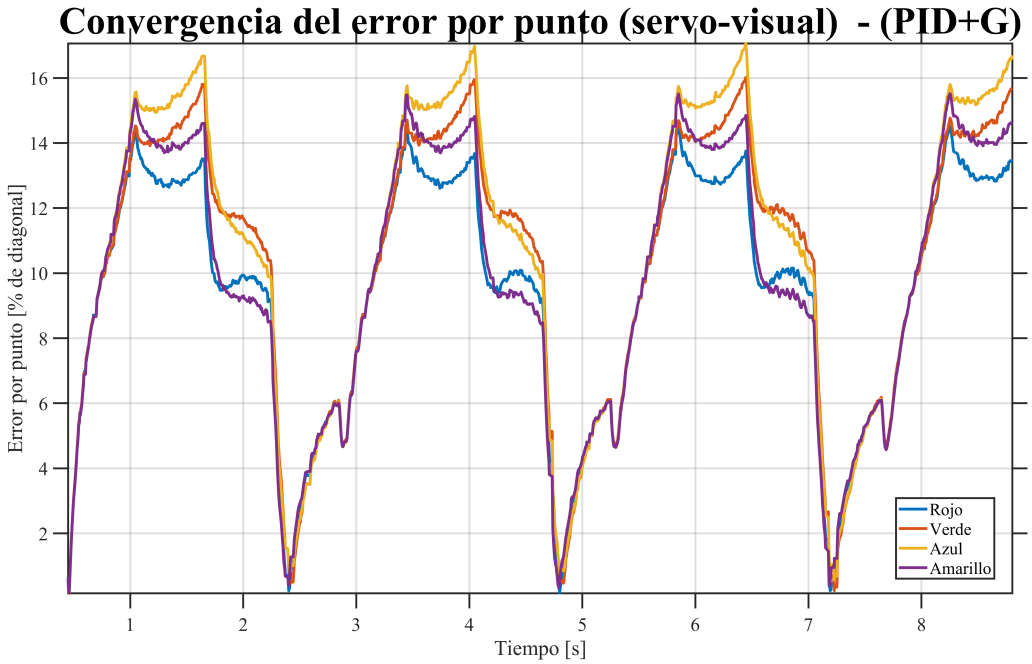

Figura 5.18: Error de posición por punto -PID+G

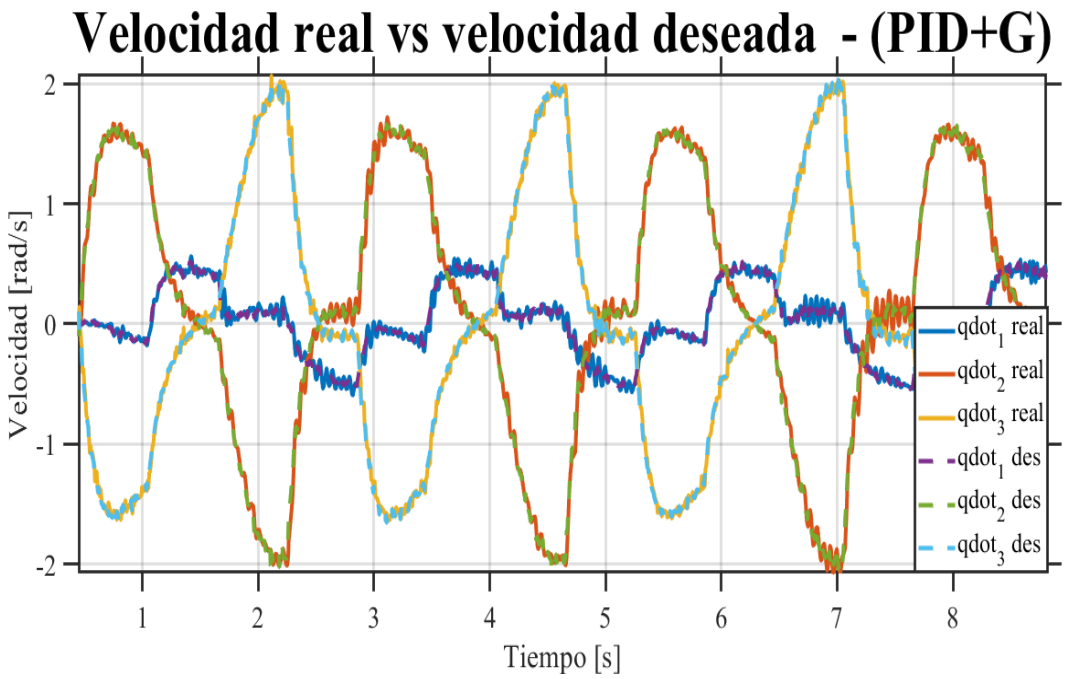

Figura 5.19: Velocidad real vs velocidad deseada -PID+G

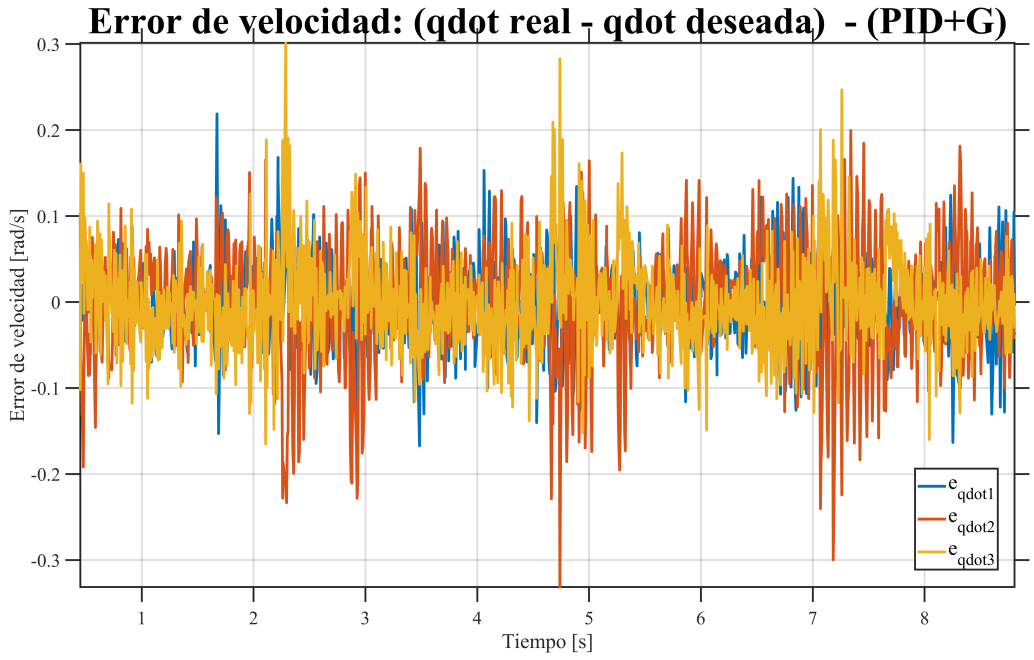

Figura 5.20: Error de velocidad articular -PID+G

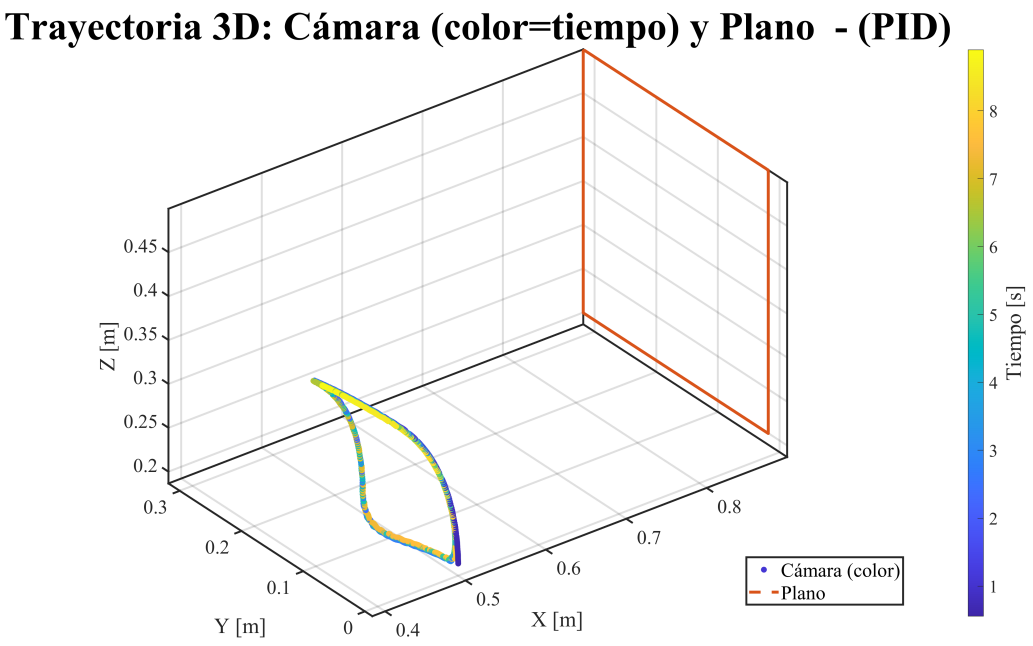

Figura 5.21: Seguimiento de trayectoria cuadrado con PID

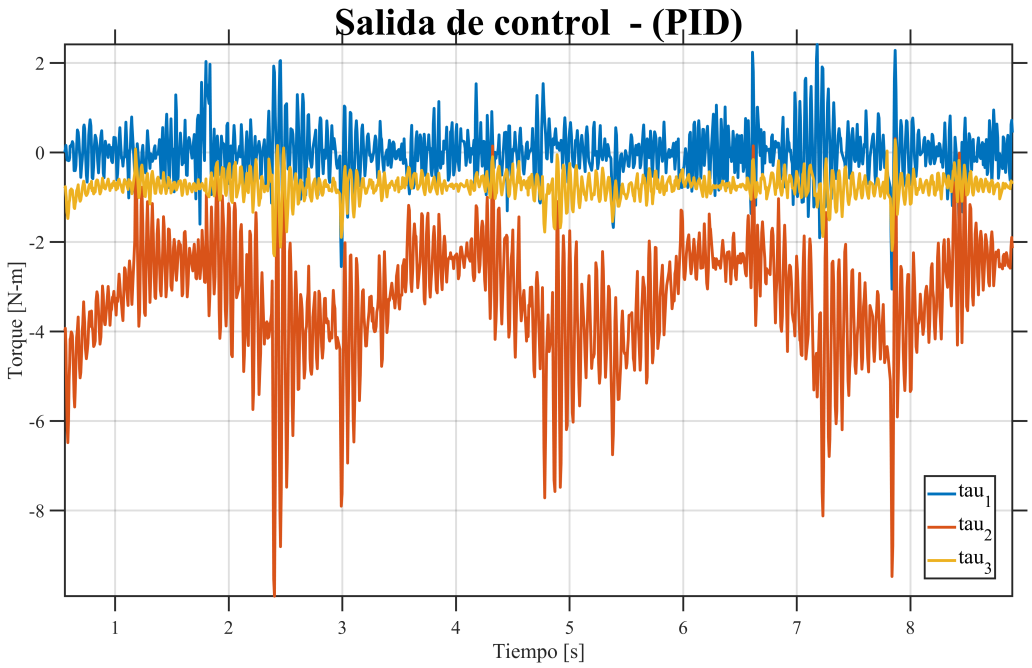

Figura 5.22: Salida de control - Torque PID

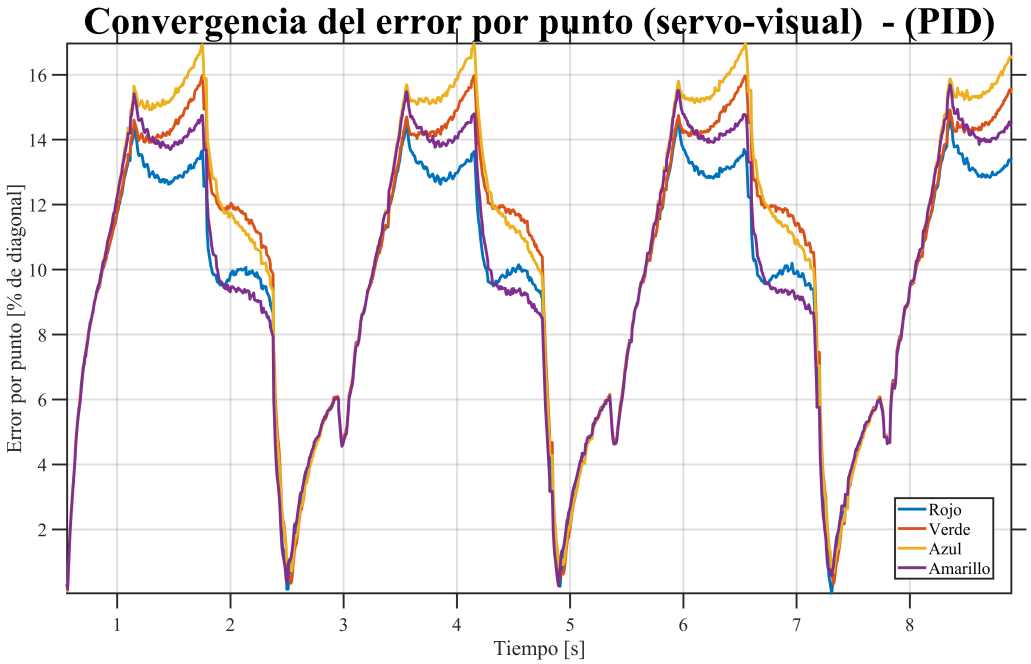

Figura 5.23: Error de posición por punto -PID

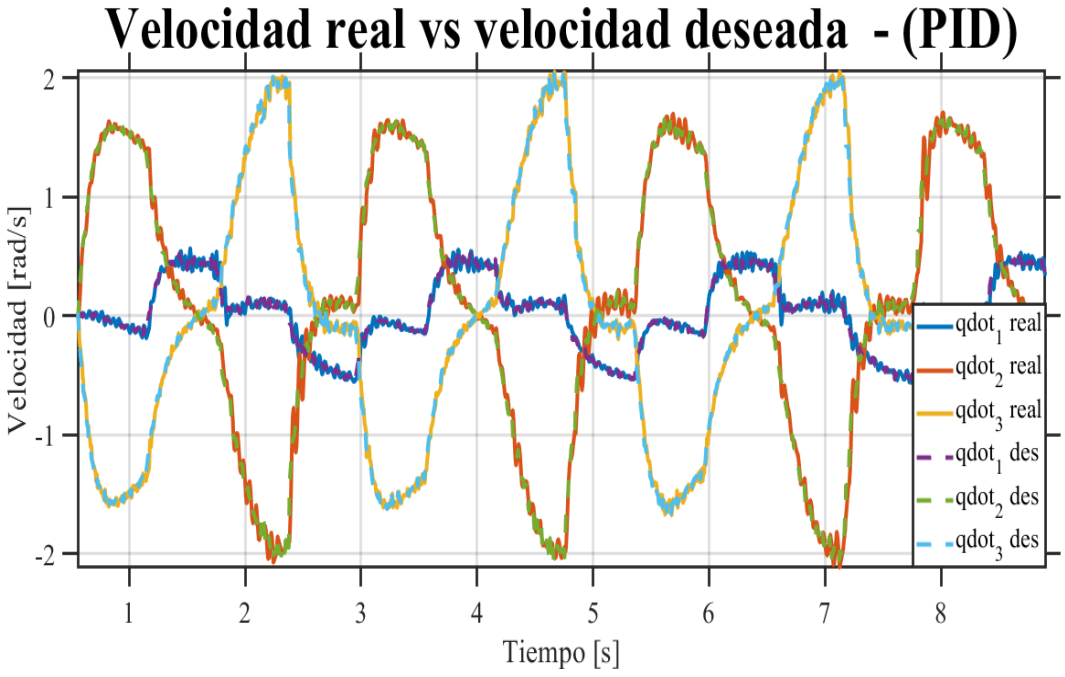

Figura 5.24: Velocidad real vs velocidad deseada -PID

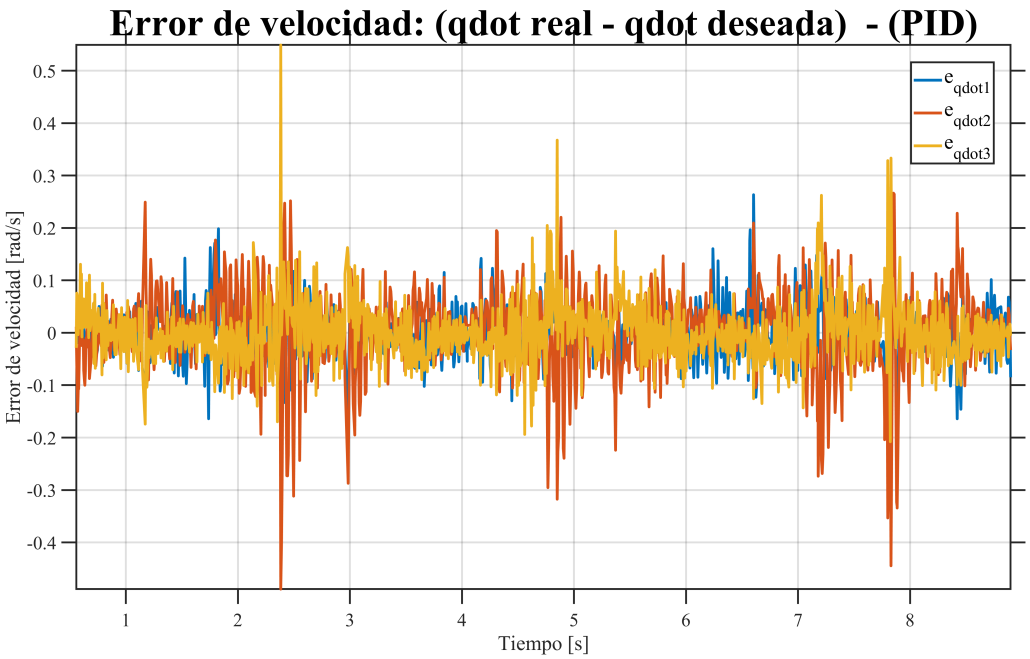

Figura 5.25: Error de velocidad articular -PID

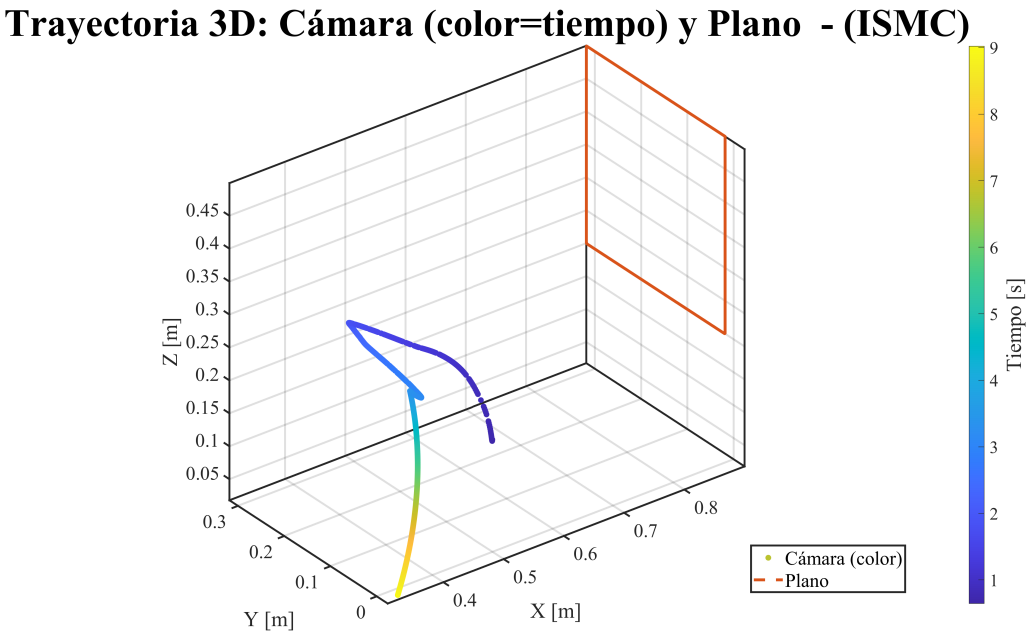

Figura 5.26: Seguimiento de trayectoria cuadrado con ISMC

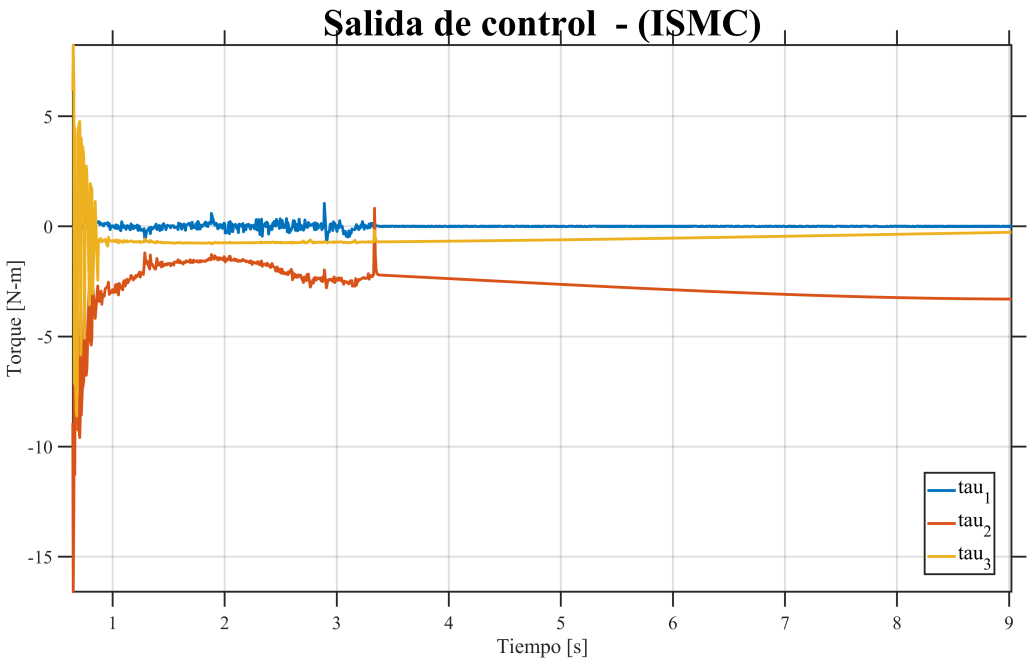

Figura 5.27: Salida de control - Torque ISMC

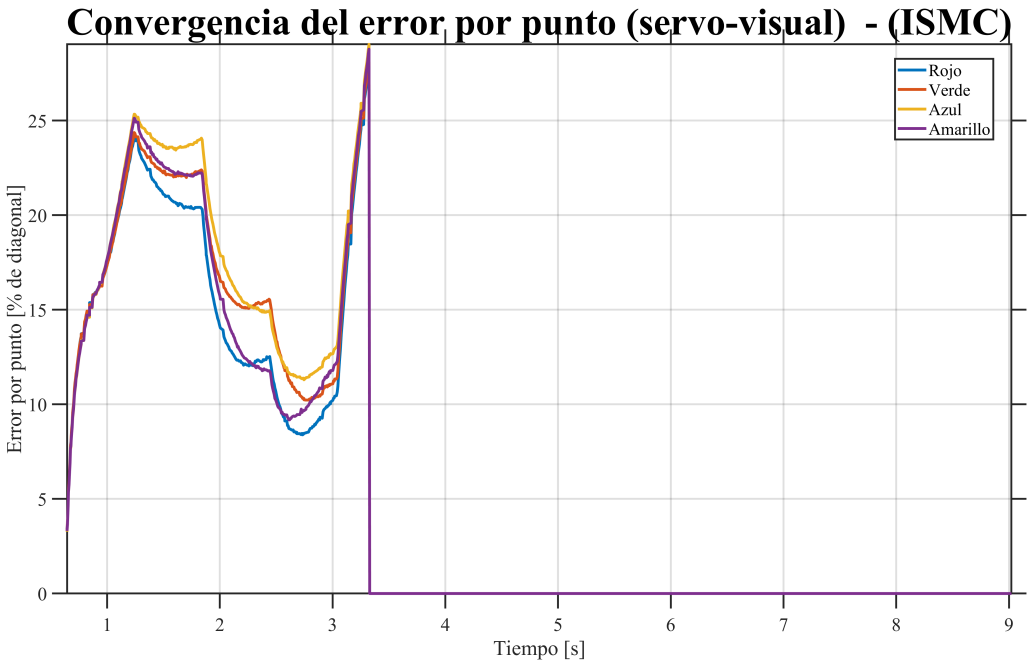

Figura 5.28: Error de posición por punto -ISMC

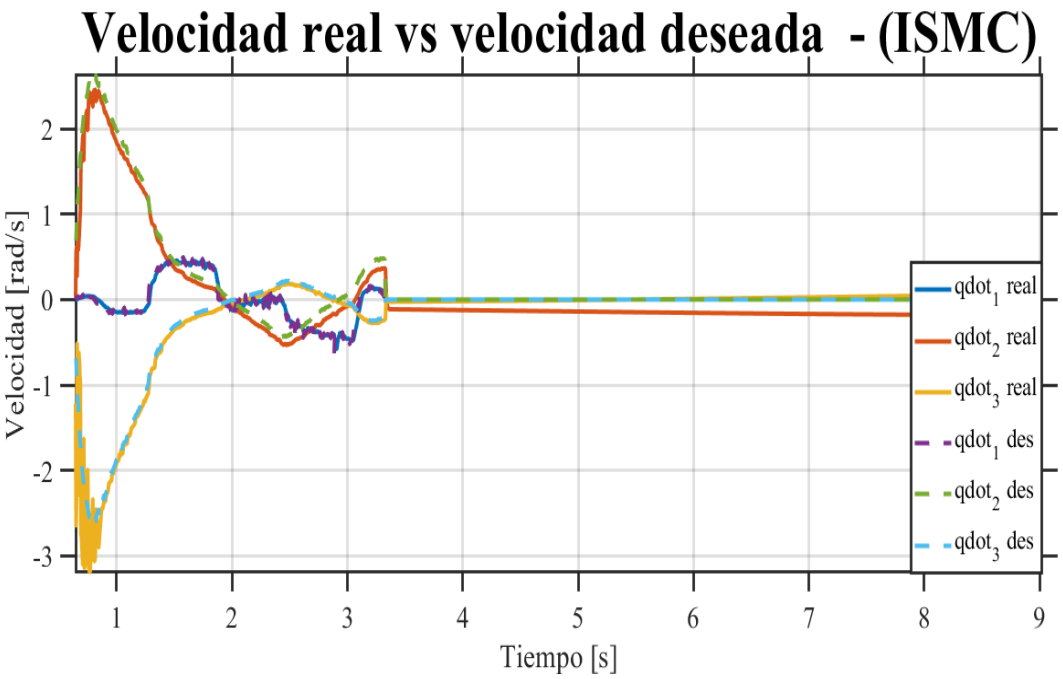

Figura 5.29: Velocidad real vs velocidad deseada -ISMC

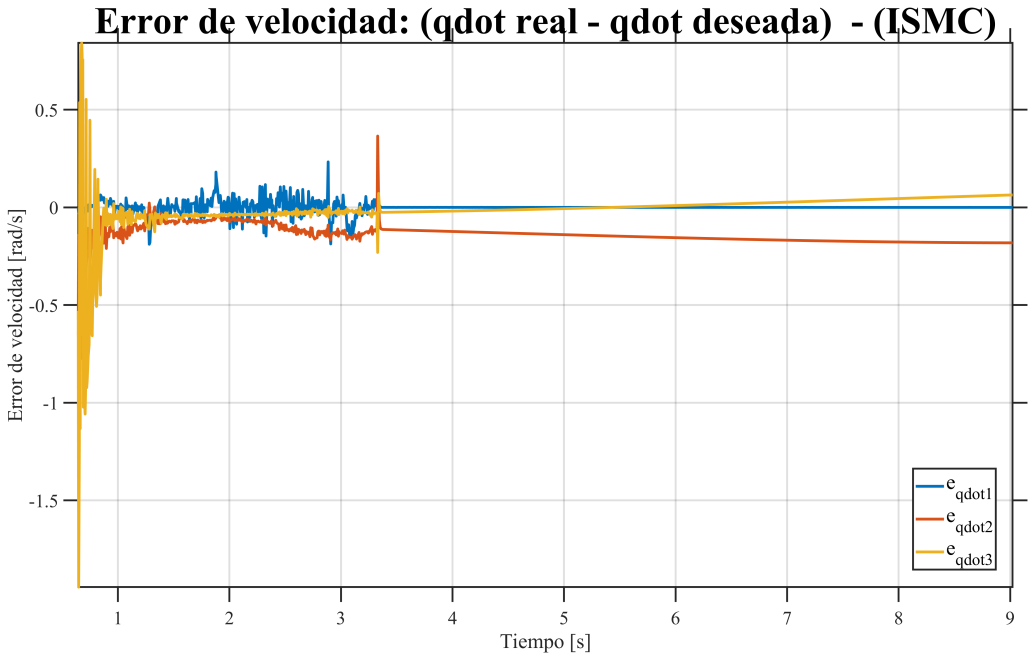

Figura 5.30: Error de velocidad articular -ISMC

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

En el presente trabajo de tesis se desarrolló e implementó la simulación tridimensional del control servovisual aplicado a un robot manipulador de tres grados de libertad con configuración cámara en mano. La plataforma propuesta permitió reproducir escenarios de seguimiento de trayectorias en el plano de la imagen, integrar el modelo cinemático y dinámico del manipulador con distintos esquemas de control, y evaluar de manera sistemática el desempeño del sistema bajo diversas condiciones geométricas y dinámicas.

A partir del análisis comparativo de las trayectorias A (corona) y B (cuadrado), se concluye que el desempeño del sistema de control servovisual depende de manera directa tanto de la complejidad geométrica de la trayectoria como de la estrategia de control implementada y del tiempo de muestreo. Las trayectorias con cambios abruptos de curvatura o discontinuidades geométricas, como la corona y el cuadrado, impusieron mayores exigencias dinámicas al manipulador, lo cual se reflejó en incrementos de torque, mayores errores transitorios y una mayor sensibilidad en las articulaciones intermedias.

En este contexto, los controladores PID y PID+G mostraron un comportamiento robusto y consistente en todas las trayectorias evaluadas. Ambos esquemas lograron completar las trayectorias sin pérdida de referencia visual, manteniendo errores porcentuales acotados y velocidades articulares dentro de rangos coherentes con la dinámica del sistema. Es importante mencionar que el tiempo de muestreo para el control servovisual y el lazo interno de control con los 3 esquemas es de 50 ms.

Es importante notar que la trayectoria A (corona) presentó errores de hasta aproximada-

mente 12–13 %, mientras que la B(cuadrado), caracterizada por discontinuidades geométricas en sus esquinas, registró errores máximos cercanos al 16.5 %. Asimismo, al emplear el controlador ISMC en distintas trayectorias, se observaron errores considerablemente mayores, alcanzando valores de hasta 24–30 %. Si bien el error obtenido puede calificarse como elevado, es necesario considerar que la referencia no puede alcanzarse de manera estrictamente exacta, dado que el sistema carece de capacidad para controlar la orientación. En consecuencia, existe una limitación estructural que impide una coincidencia total entre la trayectoria deseada y la trayectoria efectivamente seguida.

Asimismo, la inclusión de la compensación gravitacional en el esquema PID+G contribuyó a una distribución más equilibrada del esfuerzo de control, especialmente en trayectorias con mayores demandas dinámicas. Aunque en ciertos casos el desempeño global fue comparable al del PID convencional, el error de seguimiento fue menor para las trayectorias A y B, al emplear el controlador con compensación gravitacional, ya que mostró una tendencia a moderar los esfuerzos y a mantener una respuesta más estable frente a variaciones geométricas pronunciadas, lo que confirma la relevancia de incorporar términos dinámicos en el diseño del controlador.

En contraste, el controlador por modos deslizantes integral (ISMC) presentó un desempeño inferior en la mayoría de los escenarios analizados. En trayectorias con alta exigencia geométrica se registró pérdida de referencia visual antes de completar el seguimiento, mientras que en los casos donde se completó el tiempo total de simulación se observaron errores de seguimiento y esfuerzos de control significativamente mayores. Las variaciones de torque alcanzaron magnitudes superiores a las registradas con los controladores basados en PID, lo que implica un mayor consumo energético y potenciales problemas de estabilidad práctica bajo las condiciones evaluadas. Sin embargo, podría mejorarse la respuesta del controlador al realizar una resintonización de las ganancias y cambiar el tiempo de muestreo del sistema.

Adicionalmente, se identificó de manera consistente que la articulación q_2 constituye el grado de libertad más exigido dinámicamente, debido a su función estructural en el soporte de los eslabones distales. Este comportamiento fue independiente de la trayectoria y del esquema de control utilizado, lo que pone de manifiesto la influencia determinante de la configuración cinemática y de la distribución de masas en el desempeño global del sistema de servovisión.

Finalmente, se concluye que el entorno de simulación desarrollado constituye una herramienta efectiva para el análisis, validación y comparación de estrategias de control servo-

visual. Su naturaleza modular facilita la modificación de bloques, el ajuste de parámetros y el registro de variables relevantes, lo que acelera el proceso de diseño y reduce riesgos y costos asociados a pruebas experimentales directas. En consecuencia, los resultados obtenidos no sólo validan la pertinencia de los controladores PID y PID+G para el seguimiento de trayectorias visuales en manipuladores de tres grados de libertad, sino que también establecen una base sólida para futuras implementaciones experimentales y para la exploración de estrategias de control avanzadas en entornos reales.

6.2. Trabajos futuros

Como líneas de investigación futuras se plantean las siguientes acciones:

- Incorporar modelos más detallados de fricción, saturación y retardos en los actuadores, con el propósito de aproximar de manera más precisa el comportamiento dinámico del sistema real.
- Evaluar e implementar estrategias de control avanzadas, tales como esquemas robustos, adaptativos o técnicas de control libre de modelo basadas en inteligencia artificial y realizar un análisis comparativo de su desempeño respecto a los controladores actualmente empleados.
- Integrar un módulo de visión por computadora orientado al control servovisual, a fin de cerrar el lazo de control mediante mediciones obtenidas de cámaras reales conectadas al entorno de simulación a través de Matlab.
- Llevar a cabo la validación experimental en un prototipo físico, con el objetivo de cuantificar la brecha entre simulación y sistema real, así como ajustar los parámetros del modelo para mejorar su fidelidad.

Capítulo 7

Anexos

A continuación se presenta un código QR, el cual permite acceder a todos los códigos y archivos utilizados en el presente trabajo:

Figura 7.1: Escanea el código QR para acceder a los archivos.

De igual forma, se agrega el enlace de la carpeta que contiene los archivos del proyecto:

```
https://drive.google.com/drive/folders/
13g0-q8me-kNCsWjC0rBiqE4mstphTBui?usp=sharing
```

Bibliografía

[1] M. Bueno López and M. A. Arteaga Pérez, "Control servovisual de robots manipuladores: un enfoque hacia la industria," *Épsilon*, vol. 1, no. 19, pp. 9–31, 2012.

[2] F. Reyes, *Robótica-control de robots manipuladores*. Alfaomega grupo editor, 2011.

[3] P. I. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, 2nd ed., ser. Springer Tracts in Advanced Robotics. Cham, Switzerland: Springer International Publishing, 2017, vol. 118.

[4] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot modeling and control*. wiley New York, 2006, vol. 3.

[5] P. M. Khiabani, B. S. Aghdam, J. Ramezanzadeh, and H. D. Taghirad, "Visual servoing simulator by using ros and gazebo," in *2016 4th International Conference on Robotics and Mechatronics (ICROM)*. IEEE, 2016, pp. 308–312.

[6] H. Inoue and Y. Shirai, "Guiding a robot by visual feedback assembling tasks," *Eletrotechnical Laboratory Chiyoda-ku Tokyo, Japan*, 1971.

[7] J. Hill, "Real time control of a robot with a mobile camera," in *9th Int. Symp. on Industrial Robots, 1979*, 1979, pp. 233–246.

[8] N. Houshangi, "Control of a robotic manipulator to grasp a moving target using vision," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1990, pp. 604–609.

[9] C. E. Smith and N. P. Papanikolopoulos, "Vision-guided robotic grasping: Issues and experiments," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 1996, pp. 3203–3208.

[10] T. Şahin and E. Zergeroğlu, "Adaptive visual servo control of robot manipulators via composite camera inputs," in *Robot Motion and Control*. Springer, 2006, pp. 141–152.

[11] H. Wang, Y.-H. Liu, and D. Zhou, "Adaptive visual servoing using point and line features with an uncalibrated eye-in-hand camera," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 843–857, 2008.

[12] M. Keshmiri and W.-F. Xie, "Image-based visual servoing using an optimized trajectory planning technique," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 359–370, 2016.

[13] S.-H. Bae, E.-J. Kim, S.-J. Yang, J.-K. Park, and T.-Y. Kuc, "A dynamic visual servoing of robot manipulator with eye-in-hand camera," in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2018, pp. 1–4.

[14] H. Nourisola, A. Azizi, and S. S. Majidabad, "Pre-compensated proportional derivative sliding mode controller design for image based visual servoing," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*. IEEE, 2019, pp. 725–730.

[15] C. E. P. Gamboa, "Plataforma experimental para el control de un robot de transmisión directa de un gdl," Oaxaca, México, 2012.

[16] D. Bedolla Martínez *et al.*, "Emulador de un robot manipulador de 2 grados de libertad con base en un controlador digital de señales," *REPOSITORIO NACIONAL CONACYT*, 2017.

[17] M. M. Aragón, F. H. R. Leyva, and J. A. A. Aguilar, "Implementación en "hardware in the loop" del sistema carro-péndulo invertido con base en el microcontrolador hercules rm57l843 de texas instruments," *Pistas Educativas*, vol. 39, no. 128, 2018.

[18] A. D. Rocío *et al.*, "Control basado en el zmp usando bases de groebner para la marcha de un robot humanoide," *REPOSITORIO NACIONAL CONACYT*, 2019.

[19] A. C. Rangel, C. A. de Luna Ortega, S. P. F. Esquivel, and J. L. G. Ramírez, "El uso de robots en la industria: Un panorama general," in *2016 UPA Seminario de Investigación de invierno y 2do seminario de divulgación de matemáticas aplicadas*. UPA, 2016, pp. 7–13.

[20] M. Aldana, "¿qué le falta a la ciencia en México?" *Temas*, vol. 69, pp. 26–30, 2012.

[21] D. G. MAXINEZ, F. J. S. RANGEL, and L. C. GARRIDO, "Robótica, economía & educación," in *Simposio de Robótica 2014. Memorias.* UNAM, 2014, pp. 1–10.

[22] B. H. Paola, R. C. Fernando, V. L. Sergio, and R. S. Eduardo, "Diseño de un simulador 3d para robots manipuladores de 3 grados de libertad," *Misión Mecatrónica. Número*, vol. 1, 2009.

[23] A. O. Baturone, *Robótica: manipuladores y robots móviles.* Marcombo, 2005.

[24] A. C. Correa, "Sistemas robóticos teleoperados," *Ciencia e Ingeniería Neogranadina*, vol. 15, pp. 62–72, 2005.

[25] F. F. Bozzeta De la Cruz and R. A. Bruno Villena, "Diseño del control servo visual de un robot de 3 gdl aplicando rna para la identificación de elementos químicos y manipulación mediante un rayo tractor sónico," 2018.

[26] J. M. Trejo Peraza, R. M. Hernández Ortiz *et al.*, "Diseño y construcción de un prototipo de robot con tres grados de libertad para posicionamiento de objetos," 2018.

[27] M. García Bartolomé, J. M. Álvarez Ontivero, and D. Cava Jiménez, "Robotrónica: aplicaciones de la robótica," 2003.

[28] B. Siciliano and O. Khatib, *Springer handbook of robotics.* springer, 2016.

[29] M. Antonio-Cruz, C. Márquez-Sánchez, R. Silva-Ortigoza, and C. Merlo-Zapata, "Sistemas mecánicos subactuados: péndulos invertidos," *Boletín UPIITA*, no. 41, 2014.

[30] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[31] O. Nasisi and R. Carelli, "Adaptive servo visual robot control," *Robotics and Autonomous Systems*, vol. 43, no. 1, pp. 51–78, 2003.

[32] E. Cervera and P. Martinet, "Combining pixel and depth information in image-based visual servoing," in *ICAR'99-9th International Conference on Advanced Robotics*, 1999, pp. 445–450.

[33] P. I. Corke, *Visual Control of Robots: High-Performance Visual Servoing*. Taunton, Somerset, England: Research Studies Press Ltd., 1996.

[34] K. Lee, "Principles of cad/cam/cae systems," 1999.

[35] L. Andueza and I. Aguirre, "Diseño de un manipulador robótico con tres grados de libertad para fines educativos," *Ciencia e Ingeniería*, vol. 30, no. 1, pp. 3–13, 2008.

[36] J. Craig John, "Robótica, 3ª edición–ed," 2006.

[37] A. Barrientos and L. Peñin, "F., balaguer, c. y aracil, f.(2007). fundamentos de robótica."

[38] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer, 2009.

[39] Coppelia Robotics, "Coppeliasim — robot simulation software," https://www.coppeliarobotics.com, 2023, accessed: 2026-02-24.

[40] C. Knospe, "Pid control," *IEEE Control Systems Magazine*, vol. 26, no. 1, pp. 30–31, 2006.

[41] R. Kelly, V. S. Davila, and J. A. L. Perez, *Control of robot manipulators in joint space*. Springer Science & Business Media, 2006.

[42] V. I. Utkin, *Sliding Modes in Control and Optimization*. Berlin: Springer, 1992.

[43] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, *Sliding Mode Control and Observation*. New York: Springer, 2014.

[44] C. Edwards and S. K. Spurgeon, *Sliding Mode Control: Theory and Applications*. London: Taylor & Francis, 1998.