UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Implementación de algoritmos de realidad mixta en una interfaz humano-máquina háptica para el procedimiento de anestesia epidural asistido por robot teleoperado

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Marcos López Magaña

DIRECTOR DE TESIS

M.I. Daniel Haro Mendoza

Ciudad Universitaria, Cd. Mx., 2026

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado IMPLEMENTACION DE ALGORITMOS DE REALIDAD MIXTA EN UNA INTERFAZ HUMANO- MAQUINA HAPTICA PARA EL PROCEDIMIENTO DE ANESTESIA EPIDURAL ASISTIDO POR ROBOT TELEOPERADO que presenté para obtener el titulo de INGENIERO ELÉCTRICO ELECTRÓNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

MARCOS LOPEZ MAGAÑA
Número de cuenta: 316037325

Agradecimientos

Agradezco a mi familia que me ha apoyado a lo largo de toda la carrera y la vida, gracias a ellos es posible la compleción de este trabajo. También agradezco a todos los que me han ayudado a completar la carrera.

Agradezco profundamente a la Universidad Nacional Autónoma de México (UNAM), por su valioso respaldo institucional. Asimismo, extiendo mi sincero agradecimiento a la Dirección General de Asuntos del Personal Académico (DGAPA) por el apoyo brindado para la realización de este trabajo, a través de los proyectos UNAM-DGAPA-PAPIIT IT103025: “Integración de modelos de inteligencia artificial multimodal en una plataforma robótica para la ejecución autónoma de tareas de manipulación de objetos”, y UNAM-DGAPA-PAPIME PE110923: “Desarrollo de un laboratorio de robótica remota para implementar prácticas de programación de algoritmos de planificación y navegación en bancos de prueba físicos”.

Índice

Capítulo 1 Introducción

La evolución de las tecnologías médicas en las últimas décadas ha revolucionado los procedimientos quirúrgicos, ofreciendo nuevas oportunidades para mejorar la precisión y la seguridad en el campo de la medicina. La integración de tecnologías avanzadas como la realidad aumentada y los sistemas robóticos ha permitido realizar intervenciones quirúrgicas con un grado de exactitud sin precedentes. Los robots quirúrgicos proporcionan una visión detallada y maniobrabilidad precisa, lo que facilita procedimientos complejos con mínimas incisiones y reduce el riesgo de errores humanos. Esta transformación tecnológica no solo mejora los resultados quirúrgicos, sino que también acelera la recuperación del paciente y reduce las complicaciones postoperatorias, contribuyendo a una mayor calidad en la atención médica [1] [2].

Uno de los procedimientos médicos que podría beneficiarse considerablemente de estos avances es la anestesia epidural. Este procedimiento es esencial antes de cirugías en miembros inferiores y durante el trabajo de parto. Requiere que un anestesiólogo inserte una aguja Tuohy con precisión a través de un espacio intervertebral restringido por las vértebras, guiándose únicamente por la sensación táctil y su conocimiento de las estructuras anatómicas. La precisión en la inserción es crucial para evitar complicaciones graves como hematomas, dolor por pérdida de líquido cefalorraquídeo y lesiones en la estructura neural del paciente. Actualmente, este procedimiento se realiza manualmente, lo que hace que la habilidad y experiencia del anestesiólogo sean determinantes para el éxito del mismo [3].

1.2 Motivación

La necesidad de mejorar la precisión en la anestesia epidural surge de los riesgos asociados con el procedimiento manual. La inserción correcta de la aguja Tuohy es un desafío debido a la limitada visibilidad y el espacio restringido en el que se realiza. Las complicaciones derivadas de la punción incorrecta no solo afectan la salud del paciente, sino que también representan un costo significativo para las instituciones médicas debido a la gestión de eventos adversos y posibles litigios por mala praxis.

Entre las complicaciones de este método se encuentra la punción dural accidental con un porcentaje de ocurrencia de 0.5–1.5% variando según el estudio, la cual puede provocar bloqueo nervioso extendido, hipotensión, frecuencia cardíaca fetal anormal, analgesia inadecuada y anestesia general tras el fallo de la anestesia neuraxial [4]. Otras complicaciones pueden incluir daño neurológico (0,016-0,56%), hematoma epidural (0,0004-0,03%) y absceso epidural (0,01-0,05%). Junto con estas complicaciones, es común que se prolongue el tiempo de hospitalización [5] [6] [7].

En México, el costo de un procedimiento de anestesia epidural puede variar entre $300 y $1,000 USD por inyección [8]. Menos complicaciones significan una reducción en los gastos relacionados con la gestión de eventos adversos y los posibles costos legales asociados con reclamaciones por mala praxis médica, que pueden oscilar entre $15,000 y $45,000 USD anualmente para los hospitales, dependiendo de varios factores. Esto contribuye a la eficacia en el costo de las operaciones hospitalarias [9].

Al mejorar la eficiencia y precisión de los procedimientos de anestesia epidural, los hospitales pueden potencialmente aumentar su capacidad para realizar más procedimientos dentro del mismo período de tiempo. Esto podría llevar a una mayor generación de ingresos a través del aumento del flujo de pacientes y la reducción de los tiempos de espera.

Por lo tanto, se considera a la integración de tecnologías emergentes como la realidad mixta y los sistemas robóticos como una solución prometedora para estos problemas al proporcionar guías visuales precisas y retroalimentación táctil durante el procedimiento, lo que puede mejorar la exactitud y reducir los riesgos asociados.

1.3 Planteamiento de la investigación

La problemática actual en la práctica del procedimiento de anestesia epidural incluye varios desafíos significativos:

1.3.1 Dependencia de la experiencia del anestesiólogo

La habilidad del anestesiólogo para realizar el procedimiento de anestesia epidural varía según su experiencia y práctica, destacando las siguientes características:

- **Habilidad manual**: La anestesia epidural requiere una habilidad manual precisa y una experiencia considerable. Los anestesiólogos deben insertar una aguja Tuohy en el espacio epidural guiándose por la resistencia que sienten al atravesar los tejidos y no por una imagen directa. La destreza manual y la experiencia son cruciales para la precisión del procedimiento [10].
- **Variabilidad en la experiencia**: La calidad del procedimiento puede variar dependiendo de la experiencia del anestesiólogo. Anestesiólogos menos experimentados pueden tener más dificultades para realizar la punción de manera precisa [10].

1.3.2 Riesgos y complicaciones asociadas

La anestesia epidural, aunque efectiva para el manejo del dolor, presenta riesgos asociados a la técnica de inserción que pueden tener consecuencias significativas para el paciente entre los cuales destacan:

- **Punciones accidentales**: Entre los riesgos que están asociados a la dificultad de la inserción precisa y el potencial de error durante el procedimiento, se encuentran los hematomas, dolor de cabeza por la pérdida de líquido cefalorraquídeo, o incluso lesiones en la estructura neural del paciente [7].
- **Lesiones neurológicas**: La inserción incorrecta de la aguja puede llevar a daños en las estructuras neurales, lo que puede resultar en efectos adversos graves para el paciente, como déficits neurológicos o dolor crónico [6].

1.3.3 Limitaciones en la visualización y control

Entre las limitaciones del procedimiento destaca la falta de visión creando dependencia táctil del anestesiólogo para su realización, se pueden clasificar las siguientes problemáticas:

- **Ausencia de imágenes directas**: Durante el procedimiento, los anestesiólogos no tienen acceso a imágenes en tiempo real del área anatómica donde se realiza la punción. Esto significa que deben confiar en su habilidad y experiencia para evitar estructuras críticas y guiar la aguja correctamente [3].
- **Falta de retroalimentación objetiva**: Los anestesiólogos dependen en gran medida de su percepción táctil para guiar la aguja, sin tener una retroalimentación objetiva sobre la posición exacta de la aguja en relación con las estructuras anatómicas [3].

1.3.4 Variabilidad anatómica

Anatomía paciente a paciente: La variabilidad en la anatomía de los pacientes, como la presencia de diferentes tamaños y formas de los espacios intervertebrales, puede complicar aún más la inserción precisa de la aguja. Esta variabilidad puede hacer que el procedimiento sea más desafiante y aumentar el riesgo de complicaciones [11].

1.3.5 Falta de estándares uniformes

Procedimientos no estandarizados: La práctica de la anestesia epidural puede no estar estandarizada en todos los centros médicos, lo que puede llevar a diferencias en la técnica y en la calidad del procedimiento entre diferentes profesionales y lugares [11].

1.3.6 Capacitación y formación

Necesidad de formación continua: Dado que la técnica de anestesia epidural es compleja y requiere habilidad avanzada, es esencial una formación continua y capacitación adecuada para mantener la competencia en la realización del procedimiento.

Estas problemáticas, de cómo se realiza actualmente el procedimiento, resaltan la necesidad de mejoras en las técnicas y herramientas disponibles para la realización de la anestesia epidural, con el objetivo de reducir los riesgos, mejorar la precisión y ofrecer una mayor seguridad para los pacientes.

Los problemas de **dependencia de la experiencia del anestesiólogo, riesgos y complicaciones asociadas, falta de estándares uniformes**, se generan debido a la técnica actual de anestesia epidural realizada manualmente, sin embargo el uso de un robot puede presentar una solución a estas problemáticas al mejorar la estabilidad de la inserción de aguja durante el procedimiento, estandarizar la metodología y abrir la oportunidad de utilizar algoritmos colaborativos para asistir al anestesiólogo [1] [3].

En este trabajo se propone abordar las problemáticas mencionadas anteriormente mediante la integración de algoritmos de realidad mixta en una interfaz de control robótico para el procedimiento de anestesia epidural. Haciendo uso del dispositivo háptico Touch® de 3D Systems® se pretende ofrecer al anestesiólogo una retroalimentación táctil precisa

durante la inserción de la aguja Tuohy. La realidad mixta se empleará para establecer guías visuales que eviten el contacto con estructuras óseas y la médula espinal, optimizando la precisión del procedimiento. Esta combinación de tecnologías pretende transformar la práctica de la anestesia epidural, haciendo el procedimiento más seguro y eficiente.

Por otro lado, los desafíos que tienen que ver con: **limitaciones en la visualización y control, variabilidad anatómica, capacitación y formación** no serán abordados en esta tesis, sin embargo, quedan como posibles trabajos a futuro.

1.4 Hipótesis

La integración de algoritmos de realidad mixta en una interfaz humano-máquina con retroalimentación háptica para teleoperación robótica mejora significativamente la precisión y seguridad del procedimiento de anestesia epidural. Donde el uso de guías visuales, restricciones cinemáticas y retroalimentación táctil reduce los errores clínicos y las complicaciones asociadas con la inserción de la aguja Tuohy, optimizando el proceso y mejorando los resultados para los pacientes.

1.5 Objetivo y Metas de la Investigación

El objetivo de este trabajo es crear restricciones cinemáticas para un robot teleoperado desde una Interfaz Humano-Máquina (IHM) con realidad mixta y retroalimentación háptica en el procedimiento de anestesia epidural.

Las metas del trabajo son:

- Desarrollar e implementar una interfaz humano-máquina que utilice realidad mixta para visualizar restricciones cinemáticas durante el procedimiento de anestesia epidural e integrar retroalimentación háptica para proporcionar al usuario una percepción táctil de la interacción entre la aguja y los tejidos.

- Evaluar la efectividad de los algoritmos de realidad mixta y la interfaz robótica en la inserción precisa de la aguja Tuohy y en la reducción de riesgos y errores clínicos.

- Documentar los resultados y experiencias para guiar futuras investigaciones y desarrollos en la teleoperación robótica y procedimientos quirúrgicos asistidos por robot.

-Establecer la comunicación y teleoperación entre el robot y la interfaz humano-máquina.

-Definir restricciones cinemáticas para el efector del robot teleoperado y fijarlas en la posición requerida en el área de trabajo haciendo uso de un marcador que será identificado mediante un sistema de seguimiento.

1.6 Limitaciones de la Investigación

Las limitaciones de la investigación incluyen que los resultados son validados en un modelo anatómico de pruebas y el entorno es distinto a uno clínico. De igual forma se considera una anatomía de paciente invariable en el banco de pruebas.

1.7 Contribuciones

Definición de restricciones cinemáticas virtuales como algoritmos de realidad mixta para el procedimiento de anestesia epidural asistido por robot teleoperado.

El trabajo propuesto presenta significativas e innovadoras aportaciones en el ámbito de la anestesia epidural, comenzando por la mejora en la precisión del procedimiento a través de la integración de algoritmos de realidad mixta, que proporcionan guías visuales para evitar el contacto con estructuras críticas como huesos y la médula espinal, lo que optimiza la inserción de la aguja Tuohy. Además, el uso del dispositivo háptico Touch® de 3D Systems® ofrece retroalimentación táctil precisa, mejorando la percepción del anestesiólogo y reduciendo la dependencia de la experiencia manual. Estas innovaciones pueden disminuir los riesgos y complicaciones asociados con la anestesia epidural, estableciendo un marco más uniforme que aborda la falta de estándares en la técnica, lo que podría conducir a prácticas más seguras en diferentes entornos clínicos. Asimismo, la incorporación de realidad mixta y retroalimentación háptica en la formación de anestesiólogos facilitará un aprendizaje más efectivo en un entorno controlado, mientras que esta investigación también contribuye al avance de la teleoperación robótica en procedimientos quirúrgicos, promoviendo el uso de tecnologías emergentes para mejorar los resultados clínicos y la eficiencia en la atención médica.

Como un trabajo derivado, se utilizaron los algoritmos de realidad mixta en conjunto con el dispositivo háptico para establecer un sistema donde la aguja tuohy se monta en el dispositivo Touch®, mediante el cual se proporciona retroalimentación háptica durante el proceso de inserción manual de la aguja en el espacio epidural, además de incluir las guías visuales mediante realidad mixta. El artículo “Development of a haptic interface with mixed reality guidance for needle insertion in epidural anesthesia” que describe dicho trabajo, fue presentado en el octavo simposio internacional en sistemas multicuerpo y mecatrónica (MuSMe 2025) y será publicado en las memorias del congreso “**Multibody Mechatronic Systems**” de la editorial Springer.

1.8 Descripción del documento

En la sección de marco teórico, se describirán todos los términos y conceptos relevantes para la comprensión total de este trabajo de tesis. Una vez concluida, en la sección de antecedentes se abordan trabajos previos que tienen relevancia y comparten similitudes con este proyecto, de forma que se sientan las bases para su realización.

La sección de materiales y metodología describe detalladamente la forma en que se configuro el sistema de teleoperación y los elementos que lo componen para posteriormente hacer una descripción del entorno de pruebas y como se realizaron estas mismas.

Finalmente, la sección de resultados y análisis presenta los resultados de las pruebas y describe el significado de ellos relacionado a la hipótesis del proyecto. Posteriormente la sección de conclusión y discusión presenta que se concluye del trabajo en base a los resultados obtenidos y se mencionan áreas de oportunidad, así como propuestas de trabajos a futuro.

Capítulo 2 Marco teórico.

En este capítulo se abordarán los temas relacionados a este trabajo de tesis los cuales incluyen la teleoperación robótica, realidad extendida, retroalimentación háptica y anestesia epidural.

2.1 Teleoperación robótica

La teleoperación de un robot, también conocida como telerobótica, consiste en un usuario que actúa como supervisor en comunicación intermitente con una computadora en la que envía comandos y recibe información del área de trabajo, el estado de la tarea a realizar e información recibida de sensores. De esta manera el robot es subordinado del usuario y ejecuta su tarea con comandos recibidos del operador sumados a su propio comportamiento artificial. La técnica de teleoperación es importante para procesos cuyas tareas son suficientemente variables, de forma que el usuario debe monitorearlas y alterar constantemente los movimientos del robot [12].

El esquema de teleoperación se componen como indica la Figura 1 de un maestro, que incluye una interfaz humano-máquina mediante la cual un usuario introduce comandos, una computadora encargada de tomar los comandos y procesarlos según este configurado el sistema y un esclavo que recibe los comandos procesados de la computadora y los lleva a cabo en el ambiente de trabajo [12].

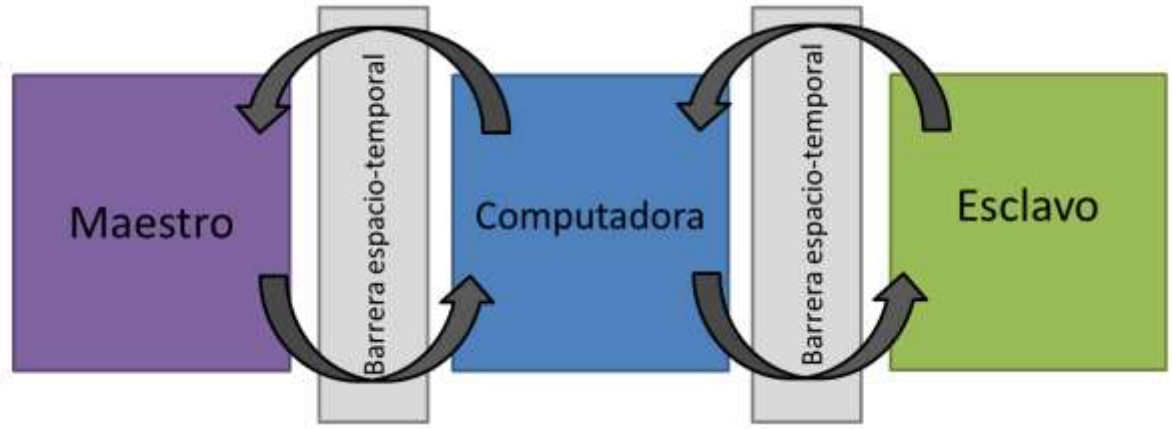


Figura 1 Esquema de teleoperación robótica

El sistema robótico teleoperado hace uso de sensores y actuadores para ayudar a la movilidad del robot y la manipulación del entorno de trabajo. Un término relacionado es el de telerobot, el cual refiere a un sistema capaz de realizar tareas de manera autónoma por períodos cortos, siempre bajo la supervisión de un humano que puede reprogramarlo y controlarlo en un proceso conocido como control supervisado [13].

La teleoperación, dependiendo del intercambio de información y el grado de las tareas ejecutada por la computadora se puede clasificar en 4 modos mostrados en el siguiente esquema [14]:

:

$$Teleoperación \begin{cases} 1.\ Sin\ autonomía \begin{cases} 1a\ Teleoperación\ Directa \\ 1b\ Teleoperación\ Directa\ Computarizada \end{cases} \\ 2.\ Semiautónoma \begin{cases} 2a\ Teleoperación\ con\ Control\ Compartido \\ 2b\ Teleoperación\ con\ Control\ Supervisor \end{cases} \end{cases}$$

En el cual la teleoperación directa refiere a la ejecución exacta de los comandos del usuario por parte del esclavo sin ningún tipo de procesamiento computacional. La teleoperación directa computarizada Es aquella que presenta cálculos para hacer más efectiva la operación del robot. Ambos modos entran en la clasificación de teleoperación sin autonomía.

Los modos de teleoperación que se presentan en un sistema semiautónomo son la teleoperación de control compartido, refiriéndose a aquella en la que el control directo se ve afectado por el control que ejerce la computadora y la teleoperación con control supervisor, en la cual el maestro solo genera comandos de alto nivel para ser procesados por la computadora y ejecutados por el esclavo [14].

El uso de sistemas robóticos teleoperados es de utilidad en diversos casos como es el ejemplo de:

- Ambientes donde sea difícil o riesgoso tener la presencia de humanos, como en tareas de búsqueda y rescate luego de desastres. En este contexto un robot teleoperado es capaz de moverse por lugares donde a una persona se le dificulta debido a su propio tamaño o toxicidad en el ambiente. De igual forma los robots se pueden considerar como prescindibles [15].

- Aplicaciones submarinas. En este contexto el control de movimiento es la función más básica que se puede implementar para llevar un vehículo subacuático a algún área de interés. Sin embargo, un entorno de navegación complejo como puede ser un área que no ha sido previamente explorada y con obstáculos, presenta un obstáculo para un movimiento completamente autónomo por lo que el uso de la teleoperación de un usuario para el vehículo es una opción viable en estas aplicaciones [16].

- Aplicaciones industriales en las cuales una tarea presenta dificultades para que una persona la realice debido a la demanda física de esta y a la vez es lo suficientemente compleja como para ser automatizada completamente y obtener resultados constantes. De esta forma un esquema de teleoperación es indicado para realizar estas tareas y a la vez evitar que los trabajadores se encuentren físicamente presentes en la línea de producción [17].

- Aplicaciones quirúrgicas. La cirugía robótica ofrece ventajas clínicas para los pacientes y las instituciones de salud. Entre los beneficios se encuentran mayor precisión, incisiones más pequeñas, menor fatiga en procedimientos prolongados, reducción de la pérdida de sangre, menos dolor, tiempos de recuperación más rápidos y menos complicaciones. Estos factores contribuyen a estancias hospitalarias más cortas, menor necesidad de transfusiones y medicamentos para el dolor, con esta técnica los pacientes experimentan muy baja mortalidad y

morbilidad. La capacidad de los brazos robóticos para mantenerse estables y proporcionar fuerza constante es especialmente útil para pacientes obesos o con anatomía complicada, y permite la transición entre personal quirúrgico durante procedimientos extensos [1].

"La teleoperación robótica permite que un usuario humano controle a distancia un robot, supervisando sus acciones y ajustando su comportamiento en tiempo real. Este tipo de interacción es posible gracias a la interfaz humano-máquina, que conecta al operador con el sistema robótico. En las siguientes secciones, se explorará en detalle el papel de cada parte que compone la teleoperación robótica. Primeramente, se abordará el papel del maestro, donde el usuario introduce los comandos que serán procesados y ejecutados por el sistema en el entorno de trabajo."

2.1.1 Lado del maestro

Desde el lado del maestro en el sistema de teleoperación, el usuario ingresa los comandos mediante una interfaz humano-máquina al sistema para ser interpretados y acatados por el esclavo. En este bloque, es importante la retroalimentación de información para poder seguir realizando la tarea de teleoperación al ajustar los comandos a las variables del espacio de trabajo y del actuador.

Interfaz humano-máquina

La parte del maestro en un esquema de teleoperación incluye la interfaz humano máquina, la cual es un sistema interactivo al que el usuario proporciona instrucciones de control y recibe información de vuelta que le asiste en la tarea de teleoperación, por lo tanto estos sistemas, proporcionan al usuario un control supervisado del robot.

Las interfaces humano-maquina son un elemento de gran importancia para el control preciso del robot teleoperado, ya que funcionan como la conexión entre el usuario y el sistema. Por ello es necesaria una forma precisa de obtener la información del operador, para ello a menudo, es necesario que los comandos ingresados por el usuario a la interfaz sean analógicos, con el fin de ingresar instrucciones que serían complicadas de expresar con símbolos [12].

La interfaz humano-máquina también se encarga de interpretar y procesar las señales recopiladas por el robot. Estas señales se almacenan y procesan para que el usuario las reciba en su interfaz, ayudándolo en la tarea de teleoperación. Este proceso generalmente se realiza a través de una interfaz gráfica, donde el usuario puede visualizar la información, y también es común el uso de señales auditivas para complementar la retroalimentación [12].

Los componentes de la Interfaz Humano-Máquina se pueden caracterizar de la siguiente forma:

1. **Dispositivos de Entrada**: Incluyen controles analógicos como joysticks, pedales o paneles de control que permiten al usuario enviar comandos precisos. De igual forma se toman en cuenta los controles digitales como un teclado o botones para tareas menos complicadas.

2. **Dispositivos de Salida**: La interfaz humano-máquina utiliza gráficas para mostrar información visual sobre el estado del robot y su entorno. Esto puede incluir mapas, imágenes de cámaras, gráficos de datos y alertas. De igual forma se pueden tener salidas de otro tipo como son las auditivas (sonidos o tonos) para alertar al usuario sobre eventos importantes o condiciones del robot. Para una operación efectiva, es crucial que la interfaz proporcione retroalimentación en tiempo real. Esto incluye la visualización de datos recopilados por los sensores del robot, como temperatura, presión y otras métricas relevantes, lo que ayuda al usuario a tomar decisiones informadas.
3. **Procesamiento de Señales**: Las señales recopiladas por el robot se procesan y se traducen en información comprensible para el usuario. Este procesamiento puede involucrar la conversión de datos crudos en representaciones visuales y auditivas que reflejen con precisión el estado y el entorno del robot.

La interfaz humano-máquina para cirugía asistida por robot debe incluir controles para manipular el movimiento del actuador y alguna forma de visualizar el área de trabajo. En el caso del sistema Sistema Quirúrgico da Vinci ® (el más ampliamente utilizado) la interfaz cuenta con joysticks para controlar los instrumentos sobre el paciente, un visualizador de estéreo que proporciona una imagen en 3D del área quirúrgica y permite ver hasta dos imágenes auxiliares e incluye comunicación de audio bidireccional. También incluye un panel táctil que permite seleccionar funciones del sistema y ajustar configuraciones como ángulo de la cámara y escalado de movimientos, pedales para manejo de cámara y energía del sistema y botones de emergencia así como para ajustar la posición del usuario.

Interfaz humano-máquina háptica

La háptica es término utilizado para describir interacciones cutáneas y kinestésicas, siendo las primeras táctiles y las segundas de fuerza. La retroalimentación kinestésica se relaciona con cómo sentimos el peso y la tensión a través de nuestros músculos y articulaciones, requiere de actuadores capaces de proporcionar fuerza y torque, lo que suele implicar equipos mecánicos externos. La retroalimentación táctil, por otro lado, se refiere a la percepción de sensaciones finas como vibración, presión y textura a través de la piel, ayudándonos a distinguir entre características superficiales. Ambas en conjunto forman las sensaciones del sentido de tacto. Otra forma de retroalimentación háptica es mediante la temperatura, especialmente en áreas térmicamente sensibles. Un cambio rápido de temperatura, aunque sea de poca magnitud es capaz de producir incomodidad al usuario [18]. Para este trabajo, solo se considerarán las interacciones kinestésicas, en las que se toman en cuenta las fuerzas que afectan un solo punto y el usuario obtiene retroalimentación háptica mediante un dispositivo en la interfaz [19].

Para conseguir retroalimentación háptica se pueden utilizar dispositivos de distintos principios de funcionalidad como son:

- Hidráulicas: Pueden generar fuerzas, deformaciones y vibraciones efectivas al hacer uso de los fluidos electrorreológicos (ER), que se controlan mediante un campo eléctrico, de esta forma pueden proporcionar una retroalimentación

kinestésica. Sin embargo, estos sistemas presentan problemas de tiempo de respuesta debido a la baja velocidad del flujo del líquido, y pueden requerir voltajes altos para provocar el flujo, lo que plantea problemas de seguridad en aplicaciones portátiles.

- Neumáticas: Funcionan con gas comprimido para causar deformaciones en la interfaz, se basan en la energía potencial del aire comprimido para crear interfaces táctiles y kinestésicas. Los beneficios de los sistemas neumáticos incluyen su bajo costo, alta capacidad de respuesta, ligereza, flexibilidad en el diseño, ausencia de líneas de recirculación y facilidad de ajuste de presión y velocidad. Además, son adecuados para entornos limpios y tienen una alta relación potencia-peso, lo que los hace seguros y eficientes.

- Piezoeléctricas: Haciendo uso del efecto piezoeléctrico inverso, estos dispositivos transforman señales eléctricas en deformaciones mecánicas generando una sensación táctil al usuario. Esta tecnología tiene un alto tiempo de respuesta, sin embargo es limitada por la debilidad de la sensación generada.

- Electromagnéticas: Estos actuadores consisten en un imán permanente que se deforma rápidamente dentro de un campo electromagnético creado por una bobina. El campo magnético, conocido como la fuerza de Lorentz, atrae o repela el imán cuando la corriente pasa a través de la bobina. La fuerza del actuador depende de factores como el número de vueltas de la bobina, su tamaño, la distancia entre la bobina y el imán, y el tamaño del imán. Estos actuadores ofrecen una retroalimentación táctil más fuerte y una mayor frecuencia de vibración con una respuesta rápida en comparación con los actuadores piezoeléctricos y los sistemas neumáticos o hidráulicos.

- Termoeléctricas: Estas interfaces están compuestas por pares termoeléctricos (semiconductores tipo p y n) conectados en serie entre dos capas de electrodos. El calor se transfiere desde el disipador de calor al punto de unión, elevando la temperatura en la superficie superior del dispositivo, mientras que una corriente eléctrica en dirección opuesta disminuye la temperatura en la superficie superior. Las interfaces hápticas termoeléctricas pueden transmitir información a través de distribuciones de temperatura en píxeles térmicos, además de regular la temperatura corporal.

La retroalimentación háptica proporciona sensaciones táctiles realistas, como masa y textura por lo que es capaz de asistir al usuario en tareas de diseño asistido por computadora y simulaciones virtuales. También es útil para la formación en procedimientos quirúrgicos o actividades en entornos peligrosos, al ofrecer retroalimentación de fuerza en procedimientos remotos [18].

Una interfaz humano-máquina háptica es un sistema mecatrónico que modula la interacción física que experimenta el usuario. Las interfaces hápticas suelen incluir partes eléctricas, mecánicas y computacionales que trabajan en conjunto. Al procesar la información de entrada proveniente del usuario mediante sensores y computarla, se le otorga a este una retroalimentación física mediante los actuadores de la interfaz. De esta

forma, se le permite al usuario tener una interacción física con un entorno remoto o virtual. Las interfaces hápticas se pueden clasificar según su diseño en tres tipos [20]:

- Fijadas sobre una superficie, como es el caso de dispositivos que generan fuerzas de torque en un punto fijo para que un usuario las sienta en un dispositivo.

- Montadas directamente al usuario, como puede ser el caso de un guante o alguna prenda equipada con estimuladores piezoeléctricos o tecnología electrotáctil [21] capaz de proporcionar retroalimentación háptica para interacciones físicas de la computadora al usuario [22].

- De superficie, que consisten en solo una superficie bidimensional que puede cambiar su textura y fricción que experimenta el usuario.

Una vez definido el bloque del maestro en un esquema de teleoperación con una interfaz háptica que permite la captura de comandos dados por el usuario a la vez de proporcionar retroalimentación táctil, en la siguiente sección se procede a dar explicación al bloque de la computadora el cual procesa los comandos ingresados y genera las señales para retroalimentación háptica.

2.1.2 Computadora

En el esquema de teleoperación, la computadora modera la interaccion que tiene el maestro con el esclavo, procesando los comandos introducidos según la configuración del sistema antes de hacerlos llegar al actuador y de igual forma procesa la información proveniente de los sensores del actuador para mandarla a la interfaz humano-máquina.

Ente las formas de procesamiento que puede realizar la computadora se incluyen los filtros para suprimir los movimientos involuntarios del operador como es el caso de temblores. Un ejemplo es “ [23]” se hace uso de técnicas de aprendizaje máquina para detectar los movimientos de temblor del operador y suprimirlos con control de ganancias antes de que el esclavo los ejecute.

En la computadora también se pueden añadir algoritmos de control como parámetros de PID al actuador para garantizar la estabilidad de la teleoperación. Otra posibilidad se encuentra en añadir procesamiento mediante redes neuronales para gestionar incertidumbres debido a la variabilidad del entorno.

Hay varias formas en que la computadora puede afectar la experiencia de teleoperación, sin embargo, para este trabajo nos centraremos en las que caen bajo el termino de realidad extendida.

Realidad extendida

La realidad extendida (XR, por sus siglas en inglés) es un término general que abarca todas las formas de tecnologías inmersivas, que incluye la realidad virtual (VR, por sus siglas en inglés), realidad mixta (MR, por sus siglas en inglés) y realidad aumentada (AR, por sus siglas en inglés). Donde, la VR busca sumergir completamente al usuario en un entorno digital, separándolo del mundo físico. La AR tiene como objetivo superponer elementos digitales (imágenes, información, objetos) sobre el mundo real. Finalmente, la

MR combina elementos del mundo real y virtual que pueden interactuar entre sí en tiempo real [24].

Las aplicaciones de XR ofrecen portabilidad al no depender en sí mismas de componentes de gran tamaño para su función, a menudo hacen uso de tecnologías portátiles, como los dispositivos de visualización montados en la cabeza tales como los cascos de realidad virtual y las gafas inteligentes. Estas aplicaciones también cuentan con la ventaja de poder ser reproducidas fácilmente, permitiendo una mayor estandarización, accesibilidad y replicabilidad en la experiencia que ofrecen [25].

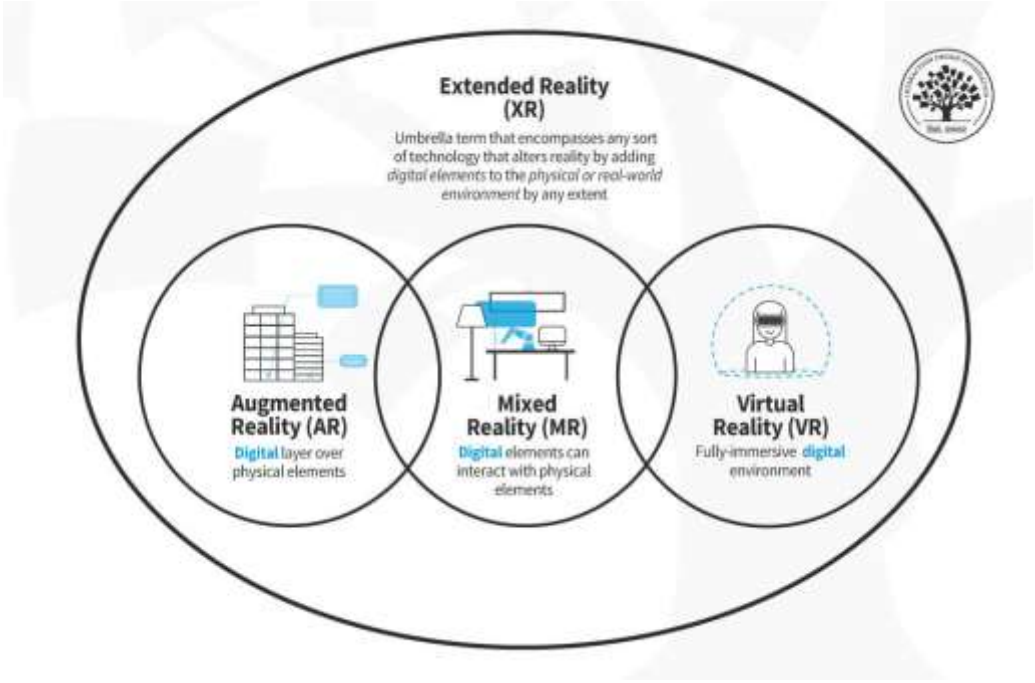


Figura 2 Diagrama de realidad extendida [24]

En las siguientes secciones se desglosarán las tecnologías que componen el termino de realidad extendida.

Realidad virtual

La realidad virtual también como Realidad Artificial, Ciberespacio o Realidad Sintética. Es una experiencia sensorial creada por computadora que sumerge completamente al usuario, haciéndole difícil distinguir entre lo virtual y lo real. Utiliza gráficos, sonidos e imágenes para reproducir versiones electrónicas de situaciones reales. Aunque la VR no es un computador en sí, utiliza tecnología informática para sintetizar realidades y recrear nuestra relación con el mundo físico en un nuevo plano. Los sistemas actuales de VR ofrecen principalmente experiencias visuales generadas por diseño asistido por computadora (CAD) y otros sistemas de gráficos y animación. Los investigadores están trabajando en dispositivos de interfaz para añadir sonido y tacto a las experiencias de VR, y en el futuro, se podría lograr una conexión directa entre la computadora y el cerebro [26].

Un avance significativo en VR fue el desarrollo de visores montados en la cabeza con dos pequeñas pantallas estereoscópicas. En este sistema el movimiento se simula ajustando

la óptica en el campo de visión en respuesta a dispositivos de entrada, o al movimiento de ciertas partes del cuerpo, como la cabeza o las manos, por lo que requiere de sensores capaces de captar los movimientos de estas extremidades. Al girar la cabeza, la escena virtual se ajusta en consecuencia, creando la sensación de estar inmerso en el mundo artificial generado por la computadora. A medida que el usuario mueve la cabeza para mirar alrededor, las imágenes en las gafas se ajustan para crear una ilusión de movimiento, haciendo que el usuario se sienta en movimiento mientras el mundo virtual permanece estático [26].

Realidad aumentada

La Realidad Aumentada es una tecnología que mejora la visualización del mundo real al combinar elementos reales con virtuales generados en la computadora creando una mezcla entre la realidad y el mundo virtual. Entre las extensas aplicaciones de esta tecnología se incluyen cirugía, terapia, reconocimiento de objetos, visualización de bases de datos, rehabilitación y filtros en aplicaciones de cámara [27].

El objetivo principal de AR es crear modelos virtuales almacenados en una base de datos. Estos modelos se recuperan, procesan y superponen sobre la escena real. Los componentes esenciales de AR incluyen sensores, dispositivos de entrada, procesadores y pantallas. En el funcionamiento de AR, se toma una escena real, la computadora la procesa y se combinan datos relevantes con la escena inicial para ser mostrados al usuario ya sea mediante una pantalla o dispositivos como gafas, lentes de contacto, etc [27].

En las ventajas de esta tecnología se encuentran la visualización personalizada siendo que la AR permite a los usuarios modificar y visualizar objetos en contextos personalizados, mejorando la perspectiva y el sentido de presencia en el mundo real. Otra ventaja es la facilidad de uso siendo que esta tecnología es accesible y fácil de usar, especialmente en teléfonos celulares. Sin embargo también posee desventajas como lo son la complejidad y costo ya que crear una experiencia AR con visualización natural es complicado y costoso, con altos gastos de mantenimiento que pueden no ser accesibles para el uso cotidiano. De igual forma se puede mencionar que la falta de dispositivos proyectores y problemas de rendimiento, como errores de alineación, pueden limitar la efectividad de AR [27].

La AR en colaboración con un robot teleoperado puede ser de utilidad al mejorar la interacción y la eficiencia en el movimiento del actuador al superponer objetos virtuales en el mundo real, sin embargo el campo de visión limitado, la estabilidad de rastreo deficiente (especialmente en presencia de oclusiones) e interfaces de usuario rudimentarias pueden suponer limitaciones al uso de esta técnica. Se prevé que el uso de inteligencia artificial para ajustar automáticamente los sistemas de rastreo y sensores, permita una colocación precisa de objetos virtuales en entornos AR. Esto en conjunto con una mejora de las interfaces de usuario AR y plataformas de simulación que puedan replicar con precisión los escenarios simulados en el entorno real podrían resolver problemas de integración y usabilidad respecto a esta tecnología [28].

Realidad mixta

La definición de este término puede variar según los expertos del tema, los problemas para precisar qué es la MR resultan en varias definiciones contradictorias. Entre las definiciones que existen, se pueden rescatar las siguientes formas de explicar la MR [29]:

- Continuidad: La definición de continuidad clasifica a la MR como un punto medio en el espectro entre AR y VR, siendo un entorno virtual con algunos elementos reales (similar a como AR es el entorno real con algunos elementos virtuales).

- Sinónimo: Se clasifica a la MR como un sinónimo de AR. Esto se ha hecho en ocasiones con el propósito de dar un nombre novedoso a algunas aplicaciones de AR.

- Colaboración: Esta definición requiere de dos usuarios ya que clasifica a la MR como la interacción entre un usuario de AR y uno de VR que pueden estar separados, de forma que el entorno local del usuario de AR es visualizado en el mundo virtual reconstruido de VR.

- Combinación: En esta definición indica que es la combinación de AR y VR en un dispositivo híbrido diseñado para soportar tanto AR como VR, permitiendo a los usuarios alternar entre ver el mundo real con objetos virtuales añadidos (AR) y ser completamente inmersos en un entorno virtual (VR). Ejemplos incluyen ciertos modelos de visores como Microsoft HoloLens y Meta Quest Pro, que tienen capacidades tanto de AR como de VR.

- AR fuerte: Bajo esta clasificación, la MR es una versión con más capacidades de AR, en la cual hay interacciones de parte de un usuario con objetos virtuales y a la vez los objetos virtuales interactúan con el ambiente real. Esto suele significar que la tecnología de MR está ligada a alguna pieza de hardware para las funcionalidades necesarias. Bajo esta definición se puede considerar a la MR como una evolución de la AR.

En esta tesis, se describirá a la MR basándonos en la definición de AR fuerte, es decir como la integración y la interacción bidireccional entre elementos virtuales y el entorno real, en la cual los elementos virtuales no solo se superponen al entorno real, sino que también pueden interactuar físicamente con él. Para ello es necesario un actuador en el mundo real capaz de integrar los algoritmos de realidad mixta a su comportamiento y los efectos que tiene sobre el ambiente, dicho de otra forma, un robot.

Al hacer uso de la MR en una aplicación de teleoperación, se puede lograr que el usuario tenga una percepción mejorada del área de trabajo al apoyarse en los objetos virtuales visibles y a la vez los efectos que dichos objetos tienen sobre el actuador, le apoyen en su tarea [30]. La MR se encuentra en el dominio de la computadora en el esquema de teleoperación, de esta forma, la imagen que recibe el usuario del área de trabajo se ve previamente procesada para incluir la superposición de los elementos virtuales generados en la computadora y a su vez puede recibir retroalimentación de la interacción con estos mediante la interfaz humano-máquina. Del lado del esclavo, los comandos que recibe también deben de procesar previamente las interacciones con los elementos virtuales

antes de ser ejecutados por el actuador. Por lo tanto al aplicar la tecnología de MR se tiene una forma de teleoperación colaborativa.

Esta tecnología emergente ofrece aplicaciones prometedoras en diversos campos, entre los cuales se incluye la medicina, donde puede utilizar junto con la cirugía asistida por robot para mejorar la precisión y seguridad en procedimientos quirúrgicos [31] [32].

Al comprender la función del bloque de computadora en el esquema de teleoperación y como se integrará la realidad mixta en este trabajo, se procede en la siguiente sección a describir el bloque del esclavo.

2.1.3 Lado del esclavo

El esclavo en el sistema de teleoperación, refiere al robot que seguirá los comandos ingresados del lado del maestro. Este puede seguir un sistema directo o colaborativo dependiendo del grado de autonomía. En las siguientes secciones se abordan los distintos grados de autonomía que se pueden implementar en el sistema robótico.

Teleoperación directa

Los sistemas de teleoperación directa suelen están diseñados para seguir el control ejercido por el operador de manera constante por lo que en este sistema el esclavo no presenta ningún tipo de autonomía. El usuario se encarga de operar el robot durante todo el tiempo que esté en funcionamiento y debe asegurarse que la tarea se esté ejecutando de manera correcta [12].

El operador se apoya de sistemas de retroalimentación con el fin de lograr una presencia remota (telepresencia). La calidad de la percepción sensorial se denomina posteriormente como transparencia. La calidad de la telepresencia que experimenta el usuario tiene un efecto directo sobre el rendimiento de las tareas y el grado de intuición que tiene el control en un esquema de teleoperación directa [33].

El esclavo en un sistema de teleoperación sin autonomía, puede funcionar de forma directa el cual la teleoperación directa refiere a la copia exacta de los comandos que el usuario ingresa en la interfaz. Por otro lado en la teleoperación directa computarizada, los comandos ingresados son procesados antes de que el esclavo los ejecute para hacer más efectiva la operación del robot [14].

El procesamiento de los comandos puede ser usado para elevar grado de autonomía del sistema robótico, surgiendo la teleoperación colaborativa.

Teleoperación colaborativa

En un esquema de teleoperación colaborativa, el esclavo presenta cierto grado de autonomía en la que, a pesar de depender de los comandos del maestro para la tarea de teleoperación, es capaz de asistirlo al tener cierta influencia sobre su movimiento. Esto con el fin de hacer más rápida y segura la tarea de teleoperación ya que puede ayudar al usuario a evitar singularidades, prevenir colisiones y en a mantener al robot sobre la trayectoria correcta necesaria para realizar su tarea. La teleoperación colaborativa se puede presentar de varios tipos dependiendo del tipo de asistencia que se le presentara al usuario en forma de autonomía por parte del esclavo [12] [34].

En un esquema de control de teleoperación es posible definir la movilidad del esclavo de forma autónoma mediante restricciones virtuales. Estas pueden ser de dos tipos:

- Restricciones virtuales de guía: en el caso que apoyan a mantener al actuador en una superficie o trayectoria específica del espacio de trabajo. Las restricciones virtuales de guía se pueden imponer de forma en que el movimiento del robot es influenciado activamente por la restricción virtual. Este tipo de guía es conocido como de impedancia y aunque es de utilidad, puede hacer que el esclavo tenga movimientos no deseados.
  Por otro lado las guías también pueden ser de tipo admitancia en la cual la velocidad a la que el robot es atraído a la guía es proporcional a la fuerza que ejerce el usuario, por lo tanto se puede experimentar un movimiento más suave si así se desea y el actuador permanecerá inmóvil si el usuario no aplica movimientos [35].

- Restricciones virtuales de región prohibida: Para prevenir colisiones o pérdida de postura mediante la imposición de restricciones virtuales para definir regiones prohibidas para el movimiento del robot, estas restricciones no tienen efecto sobre el actuador cuando este se encuentra fuera de la región de interés.
  Se pueden clasificar en restricciones de impedancia o admitancia igual que las de guía. En el caso de impedancia, la restricción proporciona una penalización oponiéndose al movimiento del robot provocando que el usuario deba ejercer más esfuerzo en traspasar la región prohibida. Por otro lado en el caso de admitancia, el esclavo ignora los comandos que lo inducirían dentro de la región prohibida por lo que nunca la traspasa [35].

Otra forma de teleoperación colaborativa es mediante la técnica de teleoperación mixta, la cual refiere al tipo de teleoperación en la que el usuario tiene control directo sobre el actuador y a la vez la computadora realiza una captura de movimientos para posteriormente ser capaz de realizar ciertas sub-tareas de manera autónoma. Este sistema es también conocido como aprendizaje por demostración, es utilidad para asistir al usuario en los casos donde se presentan interrupciones intermitentes a la comunicación [36].

Una vez definidos los niveles de colaboración que puede presentar el sistema robótico, se procede a explicar el uso de estos sistemas en el área de la medicina a lo largo de la siguiente sección.

2.2 Robótica medica

La cirugía robótica ofrece ventajas clínicas para los pacientes y las instituciones de salud. Entre los beneficios se encuentran mayor precisión, incisiones más pequeñas, menor fatiga en procedimientos prolongados, reducción de la pérdida de sangre, menos dolor, tiempos de recuperación más rápidos y menos complicaciones. Estos factores contribuyen a estancias hospitalarias más cortas, menor necesidad de transfusiones y medicamentos para el dolor, con esta técnica los pacientes experimentan muy baja mortalidad y morbilidad. La capacidad de los brazos robóticos para mantenerse estables y proporcionar fuerza constante es especialmente útil para pacientes obesos o con

anatomía complicada, y permite la transición entre personal quirúrgico durante procedimientos extensos [1].

El uso de sistemas robóticos en procedimientos médicos, se puede clasificar en teleoperación, que es el esquema utilizado en esta tesis, y de mano, que compone a los sistemas donde el operador tiene contacto directo con el actuador para realizar el procedimiento requerido. A continuación se elabora en cada uno de los esquemas.

De mano

Una forma de colaboración robótica con un cirujano es aquella donde el usuario sostiene el efector del sistema robótico con sus propias manos, lo que le permite control instantáneo sobre el mientras que el dispositivo puede realizar funciones como control de precisión o seguimiento de restricciones cinemáticas impuestas [37]. Algunas aplicaciones para este tipo de robots médicos son:

- Supresión de temblor por parte del cirujano: son de utilidad para aplicaciones que requieren de alta precisión como por ejemplo la microcirugía retinal. Estos sistemas detectan su propio movimiento y compensa movimientos no deseados realizados por el cirujano [38]. Aunque estos dispositivos son más compactos y menos intrusivos en comparación con los sistemas quirúrgicos fijos, enfrentan varios desafíos. Entre ellos, la distinción entre movimiento intencional y erróneo del operador, el aumento del peso y la inercia debido a los sensores y mecanismos, y la necesidad de operar en un amplio rango de frecuencias [37].

- Sistemas de guía: En esta aplicación se puede usar el sistema robótico de mano para guiar la dirigir la herramienta quirúrgica manejada por el cirujano hacia o alejándola de un punto, trayectoria o superficie predefinidos (restricciones activas) [39]. Además, se ha implementado retroalimentación de fuerza en dispositivos portátiles no anclados para que el operador pueda experimentar la interacción entre la herramienta quirúrgica y la restricción activa a través de una pantalla táctil [39].

- Sistemas articulados: utilizados para cirugía de mínima invasión, estos dispositivos incorporan mecatrónica en instrumentos quirúrgicos articulados, lo cual les permite ser menos intrusivos, tener más ergonomía, reducir su peso y ofrecer la posibilidad de implementar guías por imagen para localizar el efector quirúrgico [37].

- Control de fuerza: Estos sistemas utilizan sensores para medir la fuerza que se aplica en la herramienta quirúrgica y ajustar en tiempo real para mantener la fuerza deseada. Esto es esencial en entornos como la cirugía cardíaca, donde el tejido se mueve de forma rítmica y el control de la fuerza es crítico para evitar daño y asegurar precisión [37].

Teleoperado

Los sistemas robóticos teleoperados utilizados en cirugía asistida por robot, son configurados de tal forma que reciben comandos en un sistema maestro-esclavo de parte

del cirujano. Para propósitos médicos, la teleoperación ofrece posibilidades de proveer servicios de salud e intervención médica a distancia, permitiendo a los especialistas de salud realizar sus labores desde una ubicación remota y en un ambiente de trabajo más cómodo. Aun con esto, en telecirugías común que el cirujano y el esclavo estén ubicados en la misma habitación. El sistema se puede dividir en dos sitios: en el que opera el cirujano y se tienen los componentes necesarios para la manipulación del sistema robotico, y el sitio distante donde se encuentra el paciente y el sistema esclavo junto con un equipo de personal de apoyo.

Las aplicaciones de telecirugía se pueden dividir en larga distancia y corta distancia. En las aplicaciones de larga distancia es popular el sistema de ultrasonido MELODY que permite al maestro realizar un examen ecográfico de manera remota. El sistema se compone de la estación del maestro, la del esclavo y la comunicación entre ambas [40].

Los sistemas de corta distancia tienen utilidad en una amplia variedad de procedimientos como son:

- **Inserciones de agujas**: Estos sistemas robóticos permiten realizar procedimientos guiados por imágenes. Por ejemplo, un sistema presentado por [41] utiliza un manipulador con 7 grados de libertad, donde el extremo del manipulador tiene sensores para controlar la inserción de agujas y un sistema de rastreo magnético para monitorear la posición de la punta de la aguja.

- **Guía por MRI**: En terapias y procedimientos diagnósticos bajo guía MRI, como biopsias o inyección local de medicamentos, los robots compatibles con MRI. permiten manipular agujas en tiempo real para tratamientos de tumores hepáticos [42].

- **Guía por CT**: Otros sistemas robóticos asistidos por CT, utilizan un brazo robótico ligero montado en una plataforma móvil para controlar la alineación de las agujas, con datos de imagen 2D y 3D transferidos a una estación de planificación robótica [40].

- **Cirugía mínimamente invasiva**: Los robots para este tipo de cirugía realizan intervenciones a través de pequeñas incisiones, mejorando la precisión y reduciendo el trauma. Estos sistemas uelen tener múltiples brazos para diversas tareas, como sujeción de herramientas o cámaras laparoscópicas, con control remoto a través de consolas de mando.

- **Sistemas para acceso de puerto único**: Estos sistemas están diseñados para realizar intervenciones a través de una única incisión. Ejemplos incluyen robots con estructuras flexibles tipo serpiente y mecanismos de curvatura continua [43].

- **NOTES (por sus siglas en inglés Cirugía Endoscopida Transluminal por Orificio Natural):** Los sistemas robóticos para NOTES utilizan orificios corporales naturales para acceder a los órganos internos [40].

- **Intervenciones vasculares**: Un robot para intervenciones vasculares combina un manipulador con un dispositivo de imagen rotacional para guiar la colocación de catéteres [44].

- **Otras especializaciones**: entre otras aplicaciones se encuentran las de cirugía oral y ortopédica que también se pueden auxiliar de sistemas robóticos teleoperados para mejorar el procedimiento [40].

En este trabajo se hace uso de un sistema robótico teleoperado. Una vez definido el esquema, se procede a dar contexto sobre el procedimiento de nuestra elección, al ser este el de anestesia epidural, se hará una revisión sobre los tipos de anestesia antes de profundizar en la anestesia epidural en las siguientes secciones.

2.3 Anestesia [45]

Un procedimiento de anestesia es aquel en el que se hace uso de una sustancia química para bloquear temporalmente la capacidad del cerebro de reconocer un estímulo de dolor. Este tipo de procedimientos se suelen usar para manejar el dolor y el malestar durante intervenciones quirúrgicas, diagnósticas o terapéuticas.

Los procedimientos de anestesia se pueden clasificar en cuatro tipos:

- **Anestesia Local:** Este es el procedimiento más común y es utilizado en casi todas las especialidades de la medicina. Es utilizado para producir la perdida de sensibilidad en regiones pequeñas en el paciente. Este método bloquea los nervios más superficiales que permiten la sensación de dolor en la piel, evitando esta función.
  Aunque este tipo de anestesia se puede aplicar mediante el uso de geles o líquidos sobre la superficie, como se suele hacer en la faringe antes de ingresar instrumentos por esta cavidad, lo común es que se realice al inyectar lidocaína en la región de interés.

- **Anestesia general:** Este tipo de anestesia es utilizada generalmente para procedimientos quirúrgicos de mayor complejidad y extensos, en donde es necesario bloquear los receptores nerviosos de diferentes capas y tejidos en el paciente y por lo tanto no es viable hacerlo mediante varias aplicaciones de anestesia local. Cuando se le aplica anestesia general a un paciente, este es inconsciente y pierde la capacidad de realizar cualquier movimiento.

- **Anestesia Raquídea:** Este procedimiento consiste en la administración de un fármaco anestésico en el espacio subaracnoideo o intradural, el cual contiene líquido cefalorraquídeo junto con nervios y vasos de la medula. Esto tiene el efecto de bloqueo de estímulos nerviosos en el paciente.
  Para la aplicación de este tipo de anestesia, es necesario ingresar la aguja de anestesia entre las vértebras del paciente hasta llegar al espacio intradural, por lo que se debe evitar el contacto con estructuras óseas. Cabe recalcar que si el paciente experimenta dolor durante este procedimiento, se debe parar la inyección debido a un posible daño a una raíz.

- **Anestesia Epidural:** Al igual que el procedimiento de anestesia raquídea, este tipo de anestesia requiere de la inserción de una aguja en un espacio intervertebral del paciente. Este es uno de los procedimientos más requeridos antes de realizar cirugías de miembros inferiores así como en trabajos de parto. El procedimiento se describirá a detalle en la siguiente sección.

2.3.1 Anestesia epidural [45]

El procedimiento de anestesia epidural requiere que un anestesiólogo inserte una aguja Tuohy en el espacio epidural del paciente guiándose únicamente por su imagen mental de las estructuras anatómicas internas.

El espacio epidural se encuentra entre el ligamento amarillo y la duramadre. Este está ocupado por nervios, grasa y vasos sanguíneos. Para llegar al espacio epidural, el anestesiólogo debe atravesar la aguja por seis tejidos blandos, los cuales son piel, tejido adiposo, músculo, ligamento suprapinoso, ligamento interespinoso y ligamento amarillo. Esta tarea la realiza apoyándose en su percepción de resistencia en los tejidos del paciente, siendo esto lo que le indica en que espacio anatómico del paciente se encuentra actualmente la punta de la aguja Tuohy.

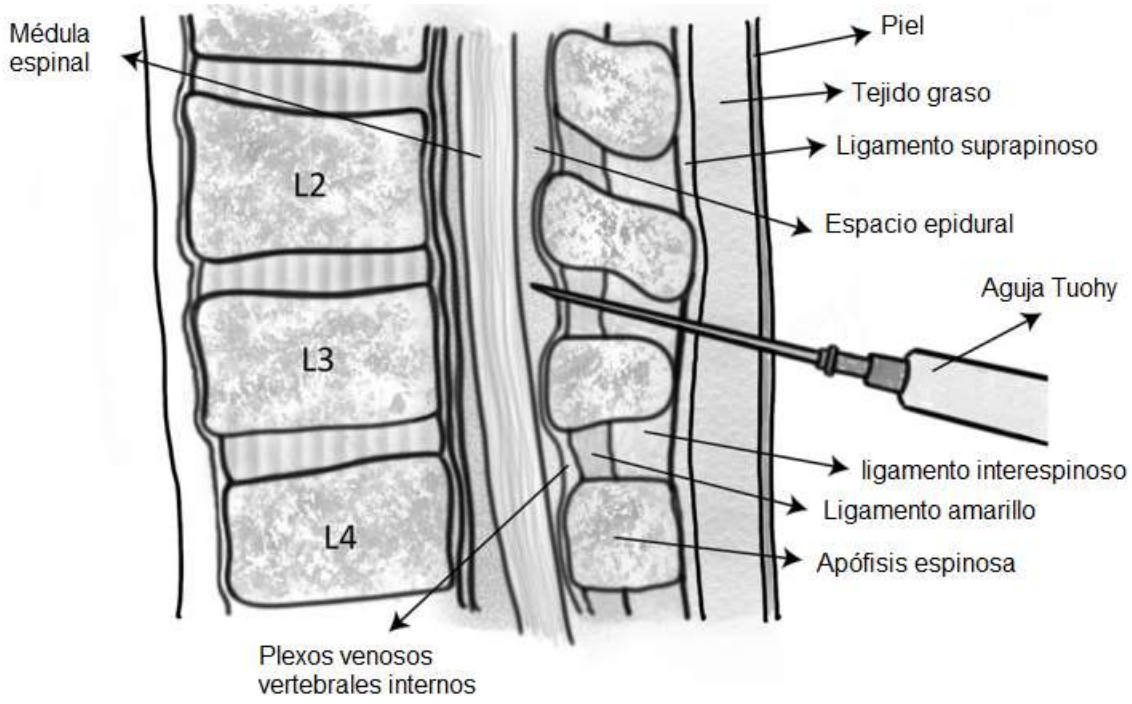


Figura 3 Anatomía del espacio intervertebral [3]

Para la realización de este procedimiento el paciente debe ser posicionado en posición de decúbito lateral o de sedestación (Figura 4) para permitir la fácil inserción de la aguja. El punto de inserción será determinado por la zona objetivo a anestesiar, donde lo más común son las punciones torácicas y lumbares.

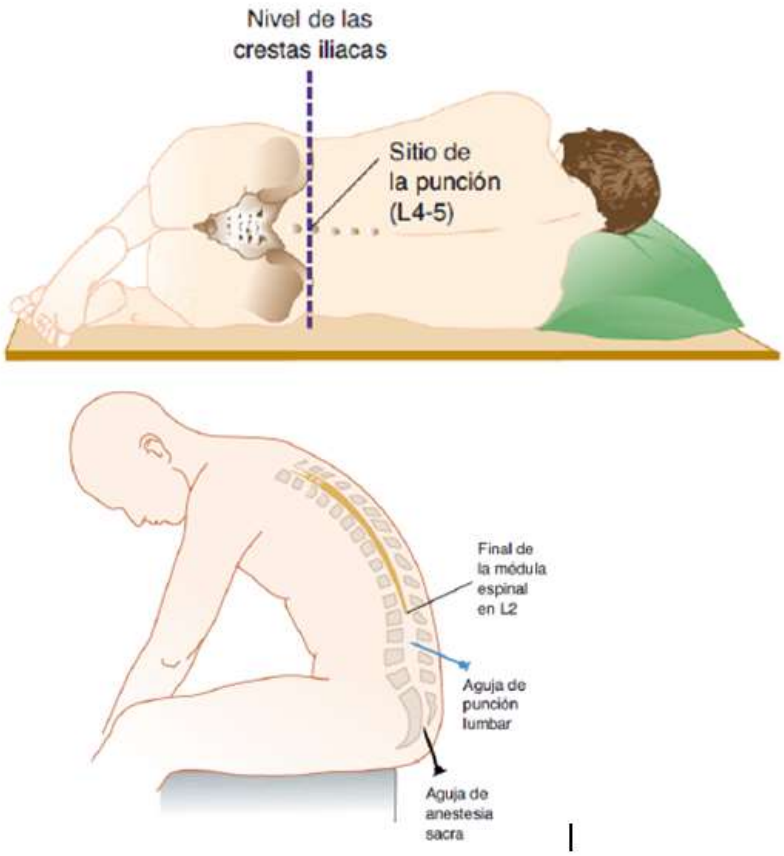


Figura 4 Modelos de inserción de aguja [45]

Previamente al procedimiento de anestesia, es necesario esterilizar y aplicar anestesia local a la zona de punción. Posteriormente se introduce la aguja Tuohy con conexión a una jeringa de baja resistencia, donde esta puede contener suero fisiológico o simplemente aire. Apoyándose en su sensación del tacto, el anestesiólogo debe seguir introduciendo la aguja mientras atraviesa las seis capas anteriormente descritas.

Una vez ingresado al espacio epidural, la resistencia en la aguja se vuelve nula y el aire o suero contenido en la jeringa es vaciado en este espacio, señalizando que la aguja se encuentra en la zona indicada. Posteriormente se hace uso de la aguja para introducir el fármaco anestésico y el catéter en caso de ser necesario. Posteriormente se retira la aguja y se cubre la punción con un apósito en caso de no haber ingresado un catéter.

En esta sección se definió el marco teórico del trabajo, incluyendo el esquema de teleoperación y el procedimiento de anestesia epidural. En la siguiente sección se realiza una revisión bibliográfica acerca de los temas tratados en el marco teórico.

3 Antecedentes

El desarrollo de tecnologías para asistir procedimientos quirúrgicos ha tenido avances considerables en las últimas décadas, incluyendo en el uso de sistemas robóticos, retroalimentación háptica y realidad extendida. En esta sección se detallaran los antecedentes relevantes respecto a las tecnologías que se usan en este trabajo, estructurándolos en torno a cuatro ejes: robótica médica, interfaces hápticas, realidad mixta/aumentada y teleoperación colaborativa.

3.1 Robótica médica y cirugía mínimamente invasiva

Remontándose al desarrollo de sistemas robóticos como ROBODOC y PROBOT hasta la consolidación del sistema da Vinci®, la robótica médica ha revolucionado la cirugía mínimamente invasiva al mejorar la precisión de un procedimiento, ayudando a reducir el trauma quirúrgico y mejorar la accesibilidad a zonas anatómicas complejas. Los sistemas robóticos, suelen emplear esquemas de teleoperación de tipo maestro-esclavo. Actualmente, aún presentan limitaciones y áreas de oportunidad como son la retroalimentación táctil y la estandarización de procedimientos. La cirugía mínimamente invasiva ha demostrado reducir complicaciones, tiempos de recuperación y costos hospitalarios [46].

3.1.2 Interfaz humano-máquina y retroalimentación háptica

La interfaz humano-máquina para cirugía asistida por robot debe incluir controles para manipular el movimiento del actuador y alguna forma de visualizar el área de trabajo. En el caso del sistema Sistema Quirúrgico da Vinci ® (recordando que es el mas ampliamente utilizado) la interfaz cuenta con joysticks para controlar los instrumentos cosbre el paciente, un visualizador de estéreo que proporciona una imagen en 3D del área quirúrgica y permite ver hasta dos imágenes auxiliares e incluye comunicación de audio bidireccional. También incluye un panel táctil que permite seleccionar funciones del sistema y ajustar configuraciones como ángulo de la cámara y escalado de movimientos, pedales para manejo de cámara y energía del sistema y botones de emergencia así como para ajustar la posición del usuario.

Se han realizado avances en el campo para manipular sistemas robóticos de formas alternativas como lo son el control por voz en el sistema ZEUS en la que el cirujano da comandos de la posición en la que quiere que se mueva el endoscopio. En los sistemas de EndoAssist y Freehand se desarrolló una forma de control mediante la orientación de cabeza del cirujano para controlar la orientación del endoscopio y en el sistema Senhance se implementó un rastreador óptico con el cual el cirujano podía controlar los movimientos del laparoscopio [47].

La interfaz humano-máquina para robots de uso médico puede complementarse con actuadores que permitan al usuario recibir retroalimentación háptica, los avances en esta área se abordan en la siguiente sección.

En el caso de la robótica médica, las interfaces humano-máquina han evolucionado significativamente, varios sistemas de control se han incorporado, como son joysticks, sistemas de voz, seguimiento óptico y visualización estereoscópica [40][47]. Un reto

persistente es la pérdida de sensibilidad táctil que ocurre al interactuar con el entorno quirúrgico mediante un robot, es decir la falta de retroalimentación háptica, se han realizado investigaciones con el propósito de suplir esta carencia. Los estudios han demostrado que los cirujanos prefieren recibir información vibratoria o kinestésica durante la manipulación remota [48].

Diversas interfaces hápticas han sido propuestas, como son: el sistema de fijación de pedículo [49], que guía la inserción sin fluoroscopía; la interfaz CombX [50], que aumenta los grados de libertad de manipulación; y el sistema Robossis [51], que mejora la alineación ósea en fracturas femorales. También se han adaptado plataformas como el da Vinci® para integrar retroalimentación háptica simple mediante joysticks y vibraciones [52]. Sin embargo, muchos de estos desarrollos no han sido implementados en entornos clínicos reales.

3.2 Realidad aumentada y mixta

La realidad aumentada (RA) y la realidad mixta (RM) permiten superponer información visual sobre el entorno anatómico del paciente, mejorando la planeación preoperatoria y la ejecución intraoperatoria. Se ha experimentado con el uso de estas tecnologías en en cirugía dental, maxilofacial, espinal y retroperitoneal, de forma que su utilización logra reducir el sangrado, mejorar la precisión y optimizar tiempos operatorios [27], [28]. En el caso de cirugía asistida por robot, estas tecnologías suelen ser adaptadas a interfaces estereoscópicas, de forma que elementos como vasos o tumores son destacados, y proporcionan alertas de proximidad a zonas de riesgo.

Pese a su potencial, la RA aún enfrenta desafíos como distracciones visuales, errores de alineación y dificultades para anotar en entornos tridimensionales. Se han propuesto mejoras a los métodos de alineación, como son soluciones con marcadores ópticos o magnéticos para una mejor calibración dinámica.

3.3 Teleoperación colaborativa

El concepto de teleoperación colaborativa extiende la teleoperación tradicional mediante la incorporación de asistencia al movimiento del instrumento quirúrgico, esta viene en este caso como límites, que pueden definirse a partir de modelos anatómicos, imágenes preoperatorias o geometrías básicas. Algunos sistemas como los empleados en cirugía de base de cráneo [30], transcraneal [55], oral [58], y nefrolitotomía percutánea [59], han demostrado que es posible crear guías que asisten activamente al operador, reduciendo errores durante el procedimiento.

Es posible la visualización de restricciones mediante RA [59] o implementarse como superficies de contacto con el propósito de retroalimentar fuerza háptica [56], [57]. En algunos casos, como en [55] dónde las restricciones también actúan como barreras de seguridad, evitando penetraciones en zonas sensibles en cirugía transcraneal. La literatura también reporta sistemas para anestesia raquídea que siguen trayectorias esperadas de inserción [3], lo cual es relevante para el presente trabajo.

La revisión de literatura, demuestra avances importantes, pero también las áreas de oportunidad, como son: uso limitado de realidad mixta con interacción física, baja

integración entre restricciones virtuales, retroalimentación háptica y visualización aumentada y escasas aplicaciones con procedimientos anestésicos. Para facilitar el análisis transversal de estos antecedentes, en la Tabla 3.1 se presenta una comparativa crítica de trabajos revisados.

Referencia	Procedimiento	Tecnología utilizada	Tipo de asistencia	Validación	Limitaciones	Aporte relevante
[49]	Fijación de tornillos en columna	Interfaz háptica sin sensores de fuerza	Retroalimentación kinestésica	Simulación	Sin sensores de fuerza	Elimina fluoroscopía intraoperatoria
[50]	Cirugía asistida genérica	Fusión de dos TouchX	Interfaz háptica multigrado	Pruebas funcionales	Costosa, compleja	Incrementa DOF hápticos
[51]	Fractura de fémur	Interfaz háptica tipo exoesqueleto	Control de fuerza y torque	Resultados positivos	Sin integración visual	Mejor alineación ósea
[52]	Cirugía laparoscópica	Joysticks + vibración; ROS	Retroalimentación háptica simple	Parcial	Integración limitada	Demuestra viabilidad técnica
[30]	Base de cráneo	Brazo robótico + guía anatómica	Guía háptica en zonas sensibles	Conceptual	Sin MR ni teleoperación	Aplicación en neurocirugía delicada

4 Metodología y materiales

Para cumplir el objetivo de este trabajo, primero es necesario realizar la configuración de un robot teleoperado para ser capaz de realizar el procedimiento de anestesia epidural, y la interfaz humano-máquina desde la cual se realizará la tarea de teleoperación. La interfaz deberá ser establecida de tal forma que permita la implementación de los algoritmos de realidad mixta, los cuales se incorporarán en la etapa final.

4.1 Configuración de sistema robótico

En la Figura 5 se muestran los bloques que conforman al sistema de teleoperación.

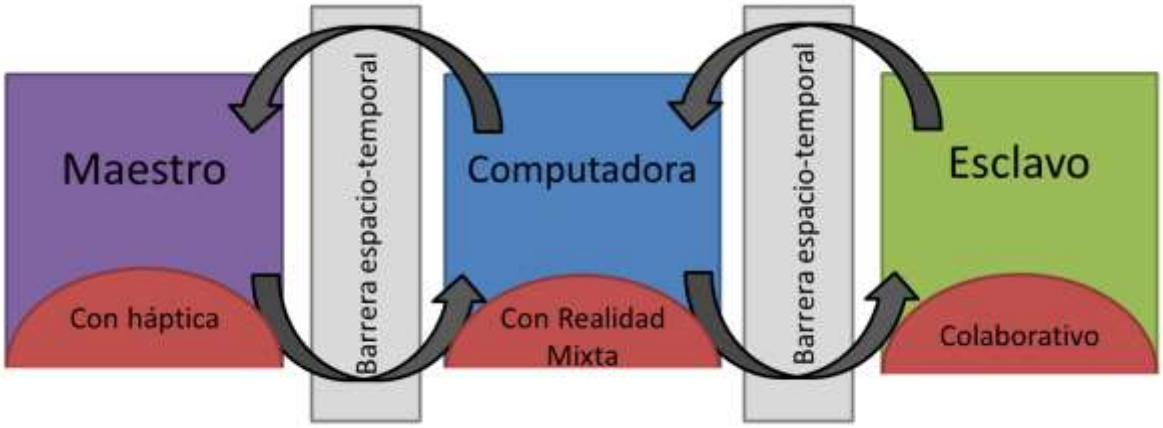


Figura 5 Diagrama de teleoperación robótica con interfaz háptica y realidad mixta

Del lado del maestro se requiere de la integración de tecnología háptica a la interfaz humano-máquina para que el anestesiólogo pueda de esta forma tener información sobre el contacto del efector final de robot (que es la aguja Tuohy) con elementos físicos y virtuales. Los elementos virtuales son generados en la computadora como algoritmos de realidad mixta que pueden ser visualizados e interactuar con la interfaz háptica. Finalmente, para el experimento el robot es de tipo colaborativo ya que los algoritmos de realidad mixta restringen sus movimientos mediante los elementos virtuales de realidad mixta. En la Figura 6 se observa la implementación física del sistema de teleoperación.

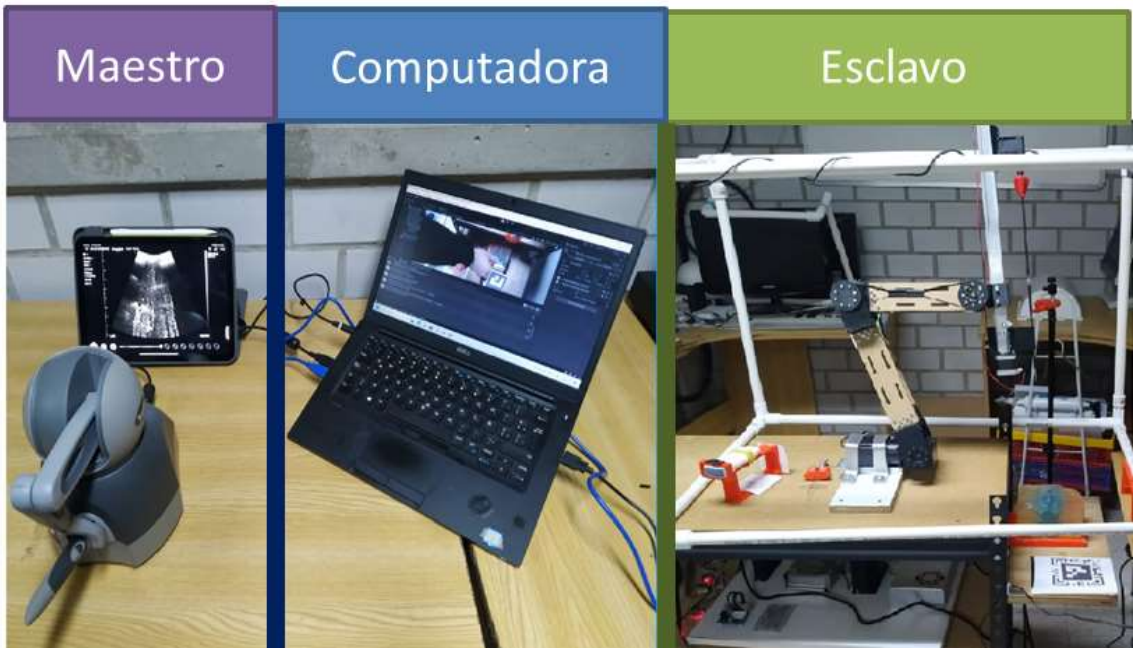


Figura 6 Elementos físicos del esquema de teleoperación

El método utilizado para configurar cada uno de estos bloques se explica a detalle en las siguientes subsecciones.

4.1.1 Lado del esclavo

El robot utilizado en este trabajo pertenece al *Mechatronics Research Group* (MRG), dirigido por el Dr. Víctor Javier González Villela, del Departamento de Ingeniería Mecatrónica de la Facultad de Ingeniería. Este sistema fue previamente diseñado y construido como parte de otros proyectos de investigación desarrollados en el laboratorio. El robot originalmente contaba con un sistema de control básico, limitado a pruebas de posicionamiento estático. En este trabajo, no se aborda el diseño mecánico ni la construcción del robot; sin embargo, se llevó a cabo una implementación independiente del control electrónico, que integra el movimiento de servomotores, un motor a pasos y la comunicación entre microcontroladores, con el fin de dotar al sistema de una funcionalidad dinámica y ejecutable en el tiempo actual. Asimismo, se realizaron las modificaciones necesarias para adaptar una aguja Tuohy al robot, permitiendo la ejecución de los experimentos requeridos para validar los objetivos planteados.

Descripción y configuración mecánica del robot.
El robot utilizado en esta tesis tiene como propósito controlar el movimiento de una aguja Tuohy durante procedimientos médicos asistidos por robot. Cuenta con tres grados de libertad (3-DoF): dos rotacionales y uno prismático. La configuración mecánica, ilustrada en la Figura 7, consiste en cinco cuerpos rígidos conectados mediante tres juntas rotacionales y una junta prismática.

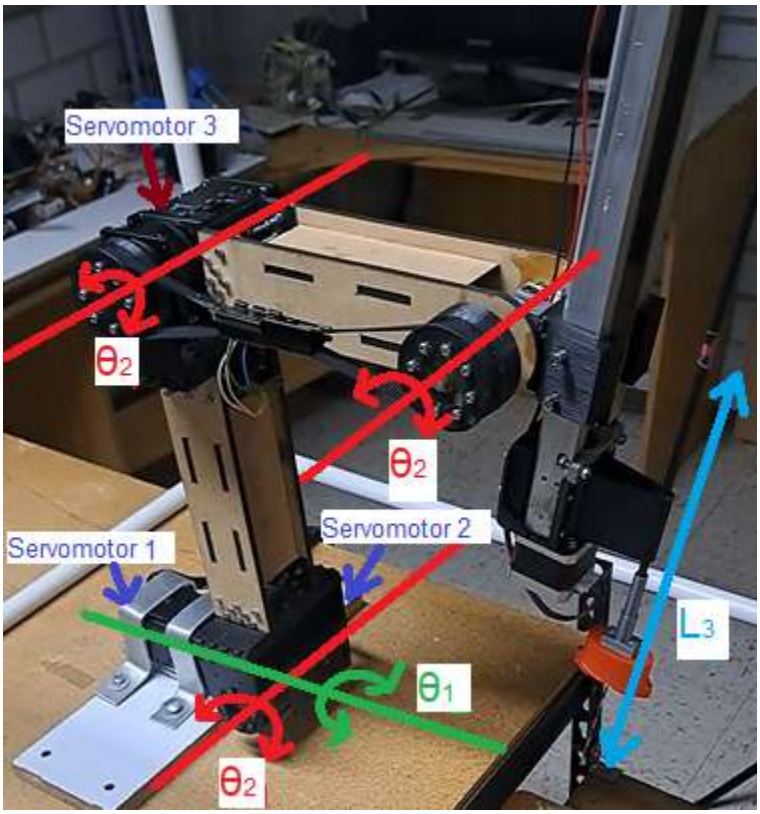


Figura 7 Configuración mecánica del robot

El sistema rotacional del robot está compuesto por tres servomotores DYNAMIXEL PRO. El servomotor DYNAMIXEL 1 permite la rotación completa del brazo robótico alrededor de su base, mientras que los servomotores DYNAMIXEL 2 y 3 están sincronizados por software para controlar el movimiento coordinado de los eslabones que conforman un mecanismo tipo paralelogramo. En esta configuración, el servomotor 3 transmite su movimiento a una articulación rotacional pasiva mediante una polea dentada y una banda, lo que asegura la conservación de la geometría del paralelogramo durante el movimiento.

Este paralelogramo constituye un subconjunto cinemático de cuatro eslabones articulados que mantienen su forma geométrica mediante la sincronización entre los actuadores y la transmisión mecánica (la banda y polea). Esta coordinación permite que el extremo del robot conserve una orientación constante con respecto a un punto fijo en el espacio, lo cual es fundamental para establecer un **Centro Remoto de Movimiento (RCM)**.

Un RCM (Centro remoto de movimiento, por sus siglas en ingles al ser un término estandarizado) es un punto fijo en el espacio alrededor del cual el efector final puede rotar sin generar desplazamientos laterales. En la cirugía mínimamente invasiva, el instrumento quirúrgico está restringido a cuatro grados de libertad a través del puerto de incisión los cuales son tres movimientos rotacionales de los cuales uno es sobre su mismo eje y translación a lo largo del eje longitudinal. Los mecanismos de (RCM) proporcionan los dos grados de libertad rotacionales, permitiendo que el instrumento quirúrgico pivote alrededor del puerto de incisión.

En este trabajo, el RCM se genera gracias a la coordinación del paralelogramo mecánico formado por los eslabones del brazo robótico y su sincronización mediante los servomotores DYNAMIXEL 2 y 3 garantizando que el punto de pivote de la aguja permanezca fijo en el espacio durante la ejecución del procedimiento. Dicho punto se encuentra en el vástago móvil del actuador lineal, precisamente donde está montada la aguja Tuohy.

El movimiento lineal de la aguja se logra mediante un motor a pasos Nema 17 que acciona una articulación prismática a través de un sistema de tornillo sin fin. La aguja Tuohy está instalada sobre un vástago móvil, el cual fue integrado mecánicamente mediante un acoplamiento diseñado como parte de las modificaciones realizadas en esta tesis, permitiendo así la ejecución precisa de los experimentos requeridos.

Finalmente, el área de trabajo de la aguja es un volumen cónico, quedando restringida por el RCM, el cual depende directamente de dos variables angulares asociadas al sistema rotacional (controladas por DYNAMIXEL 1, 2 y 3) y de una variable lineal correspondiente al sistema prismático. Esta configuración asegura la ejecución controlada del procedimiento simulado bajo condiciones geométricamente precisas.

Control electrónico del robot.

En la Figura 8 se muestra el diagrama del sistema electrónico encargado de controlar al robot. En el distinguen dos microcontroladores ESP32: el ESP32 maestro, ubicado en la parte superior, y el ESP32 esclavo, en la parte inferior, ambos se comunican mediante UART haciendo uso de sus puertos 1 y 3.
En este sistema, el ESP32 maestro se encarga del control de los tres servomotores DYNAMIXEL PRO (IDs 1, 2 y 3), mediante una señal TTL y un convertidor de TTL a RS485 el cual permite la comunicación con los motores. Estos servomotores están interconectados en red serial y representados en la figura como un único bloque denominado *“servomotors”*, el cual incluye las entradas de comunicación y una fuente de alimentación externa. El servomotor 3 acciona una polea con banda para ejecutar movimientos coordinados con el servomotor 2.
Por otro lado, el ESP32 esclavo es responsable de controlar un motor a pasos Nema 17 a través de un driver TB6600. Este driver recibe dos señales: un tren de pulsos que determina los pasos del motor y una señal digital que define la dirección de giro. También se incluye en el sistema un interruptor de final de carrera conectado al microcontrolador esclavo para referencia de posición en el eje prismático. El driver se alimenta con una fuente de corriente directa y está conectado a las bobinas del motor a pasos.

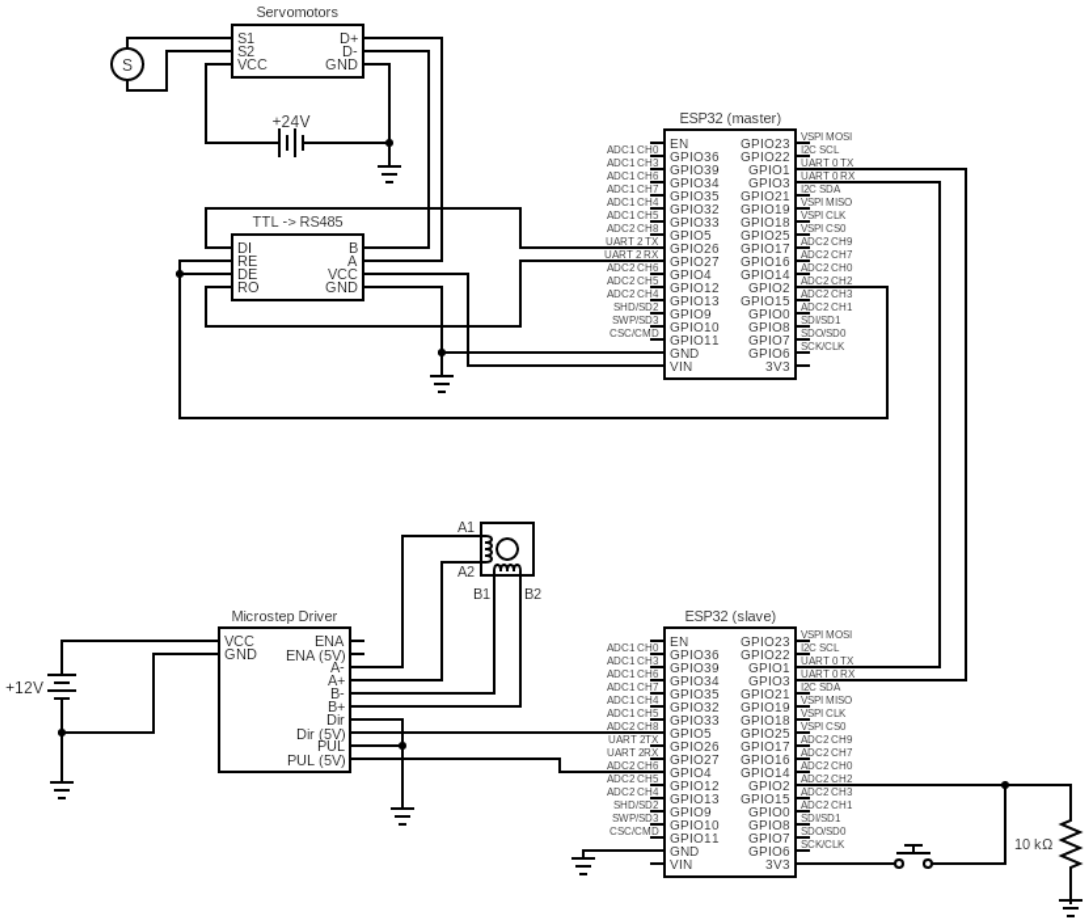


Figura 8 Diagrama de conexiones eléctricas de los motores (diagrama creado en www.circuit-diagram.org)

Arquitectura y flujo de control

El sistema robótico está dividido en dos subsistemas independientes, lo que permite desacoplar los movimientos rotacionales del robot y el movimiento del actuador lineal. Esta organización facilita la operación simultánea y eficiente de ambos mecanismos.

ESP32 Maestro

El ESP32 maestro recibe instrucciones de posición desde una computadora externa mediante protocolo UDP. Estas instrucciones corresponden a ángulos y desplazamientos preprocesados y se envían a través del puerto 1234. El microcontrolador interpreta los datos y actualiza las posiciones de los servomotores usando la librería Dynamixel2Arduino.

El flujo general de ejecución del ESP32 maestro es el siguiente:

INICIALIZAR
- Configurar comunicación I2C (Wire)
- Inicializar puerto UART para comunicación con servomotores
- Configurar parámetros de comunicación para servomotores Dynamixel
- Conectar a la red Wi-Fi usando SSID y contraseña

```
  Escuchar en puerto UDP 1234

  CONFIGURAR SERVOMOTORES
  Para cada servomotor (ID 1, 2, 3):
    Si el servomotor responde al ping
      Configurar torque y modo de operación para el servomotor

  ESCUCHAR PAQUETES UDP
  Si se recibe un paquete UDP:
    Leer datos del paquete y asignarlos a las variables:
      De ángulos y distancias requeridas
    Si L1 ≠ -100:
      Enviar L1 al motor a pasos mediante I2C

Función CALCULAR Y MOVER SERVOMOTORES
  Limitar los ángulos de los servos a un rango de -45 a 45 grados (para T2)
  Mapear T2 a un valor de servo adecuado (angulo1)
  Mapear T2 a otro valor de servo (angulo2)
  Mapear T1 a un valor de servo (angulo3)

  Si T1 ≠ -100: (si el ángulo 1 es -100 el valor de ángulos es ignorado y se está en modo inserción unicamente)
    Enviar angulo1 a DXL_ID_2 (Servo 2)
    Enviar angulo2 a DXL_ID_3 (Servo 3)
    Enviar angulo3 a DXL_ID_1 (Servo 1)

LOOP
  Continuar con la ejecución del ciclo:
    Recibir paquetes UDP
    Calcular y mover servomotores
```

ESP32 Esclavo

El segundo microcontrolador ESP32 se dedica exclusivamente al control del eje prismático. Al recibir una distancia L1 por UART desde el maestro, este microcontrolador:

- Interpreta el valor.
- Calcula la cantidad de pasos necesarios según la resolución configurada (6400 pasos por vuelta).
- Acciona el motor a pasos en el sentido indicado usando la librería AccelStepper.

De esta forma el sistema lineal queda desacoplado del sistema rotacional, evitando la interferencia entre ambos subsistemas (Figura 8).

Cálculo de la cinemática inversa.

Para controlar con la posición de la punta de la aguja Tuohy, es necesario calcular los desplazamientos angulares y lineales que deben realizar los actuadores del robot. Este procedimiento, conocido como cálculo de la cinemática inversa, se basa en la posición deseada de la punta de la aguja con respecto al Centro Remoto de Movimiento (RCM), que actúa como punto fijo de pivote durante el procedimiento.

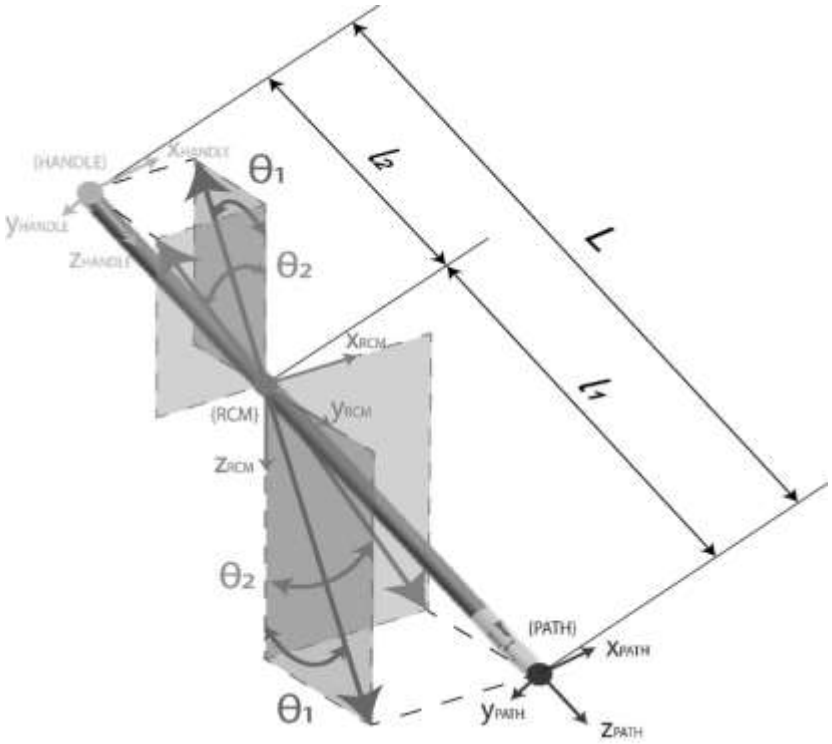


Figura 9 Cinemática inversa del efector final

En la Figura 9 se muestra cómo los ángulos θ_1, θ_2 y las disntacias l_1, l_2 dependen de las coordenadas cartesianas del efector final (X_{path},Y_{path},Z_{path}) respecto a la posición espacial del RCM fijo, ubicado (X_{RCM}, Y_{RCM}, Z_{RCM}), el cual para simplicidad del análisis se establecerá en (0,0,0).

El ángulo θ_a, que representa la rotación en el eje Z, se calcula como:

$$\theta_a = atan2(Y_{path}, Z_{path}) \quad (1)$$

Regresando al ángulo correspondiente a la tangente de $\frac{Y_{path}}{Z_{path}}$ pero en un rango de $-\pi$ a π, lo cual la hace una función adecuada para coordenadas cartesianas.

El ángulo θ_b, correspondiente a la rotación en el eje X, se obtiene de forma análoga:

$$\theta_b = atan2(X_{path}, Z_{path}) \quad (2)$$

La distancia l_1, que representa la distancia euclidiana entre el efector final y el RCM, se define como:

$$l_1 = \sqrt{{X_{path}}^2 + {Y_{path}}^2 + {Z_{path}}^2} \quad (3)$$

La distancia l_2, que corresponde a la porción restante del instrumento por encima del RCM, se determina a partir de la longitud total L del instrumento:

$$l_2 = L - l_1 = L - \sqrt{{X_{path}}^2 + {Y_{path}}^2 + {Z_{path}}^2} \quad (4)$$

A partir de estos parámetros intermedios, la Figura 10 permite derivar la cinemática inversa específica del modelo robótico utilizado.

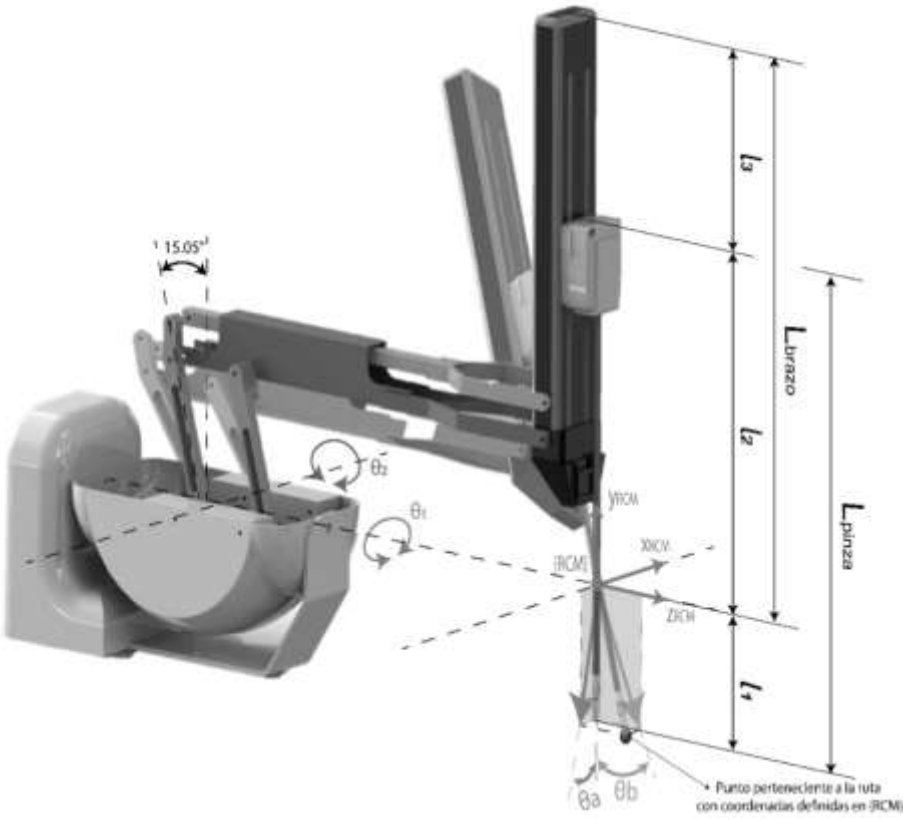


Figura 10 Cinemática inversa del modelo de robot Da Vinci

Se establece la siguiente correspondencia entre los ángulos del modelo y los obtenidos anteriormente:

$$\theta_a = \theta_2 \quad (5)$$

$$\theta_b = \theta_2 - 15° \quad (6)$$

La resta de 15° en θ_b se debe a la inclinación del primer eslabón respecto a la base, mientras que la aguja se mantiene perpendicular a esa base, lo cual es esencial para preservar el RCM mediante la geometría del paralelogramo.

Las distancias l_2, l_1 permanecen constantes, pero se introduce un nuevo parámetro l_3, definido como la distancia entre el punto inicial del instrumento y el extremo inferior del bloque que guía el tornillo sin fin. Este parámetro se calcula como:

$$l_3 = l_{brazo} - l_2 \tag{7}$$

Donde l_{brazo} es la distancia del inicio del bloque al RCM y es una distancia constante de los parámetros del robot. Escrito de otra forma:

$$l_3 = l_{brazo} - L - \sqrt{X_{path}{}^2 + Y_{path}{}^2 + Z_{path}{}^2} \tag{8}$$

Es importante señalar que, debido a que la aguja Tuohy está montada lateralmente respecto al eje del último eslabón (y no sobre el mismo eje), el efector final se encuentra desplazado una distancia fija en el eje Z, como se ilustra en las figuras correspondientes.

Los cálculos e imágenes de cinemática inversa fueron desarrollados por el M.I. Daniel Haro en clase, de forma que solo son implementadas en este trabajo. Los cálculos permiten determinar la configuración articular necesaria para alcanzar cualquier punto dentro del espacio de trabajo definido por el RCM. La cinemática inversa no se calcula dentro del microcontrolador, sin embargo, constituye la base sobre la que se gobierna el movimiento del robot. En la siguiente sección se describe el bloque de cómputo maestro, responsable de generar las instrucciones articulares.

4.1.2 Lado de la computadora

Rol del sistema de cómputo.

En el esquema de teleoperación desarrollado en este trabajo, el sistema de cómputo desempeña un papel central como intermediario entre el operador humano y el robot esclavo. Es responsable de capturar los movimientos realizados por el usuario mediante un dispositivo háptico, calcular la cinemática inversa del robot, generar la retroalimentación háptica según la interacción con objetos virtuales y coordinar la visualización del entorno tridimensional en tiempo real.

Figura 11 Mano sosteniendo el Stylus del dispositivo háptico[1]

El dispositivo háptico empleado en este sistema es un 3D Systems Touch®, el cual permite al usuario manipular un lápiz físico o stylus (Figura 11). Este stylus se encuentra articulado mecánicamente y equipado con sensores que capturan en tiempo actual su posición y orientación en el espacio tridimensional. Los datos obtenidos son procesados por la computadora y transformados en una representación virtual en el entorno gráfico, conocida como stylus virtual, la cual refleja en pantalla los movimientos del gimball en el stylus físico (Figura 12).

[1] Obtenido de magistecnologia.com.br/touch

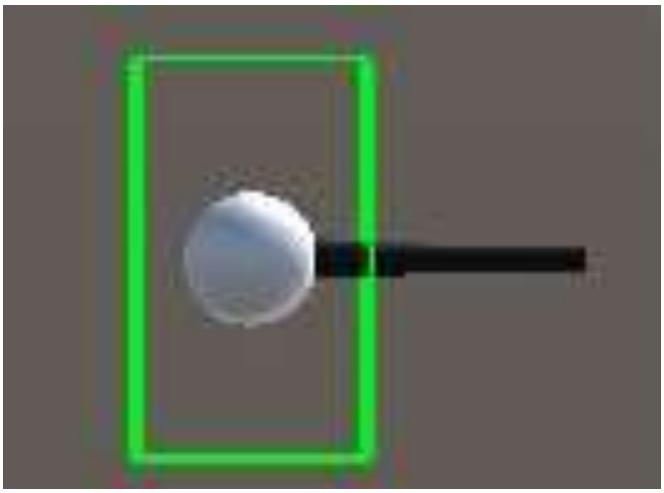


Figura 12 Representación virtual del stylus del dispositivo

Para estas funciones, se utilizó una computadora portátil Dell® Latitude 7480 equipada con un procesador Intel® Core i7 de séptima generación y 8 GB de memoria RAM, capaz de ejecutar las tareas de procesamiento, visualización y control requeridas (Figura 13).

Figura 13 Características del sistema de cómputo

Software y herramientas utilizadas

Este trabajo se implementó en el entorno de desarrollo de Unity® (versión de estudiante), un motor gráfico que permite integrar en un mismo espacio las funcionalidades de

visualización 3D, física de colisiones, comunicación con los elementos de hardware externo y programación de funciones específicas para los objetos del entorno.

Para la interacción con el dispositivo háptico Touch® de 3D Systems, se utilizó el complemento Haptics Direct[2], que permite obtener la posición y orientación de forma actualizada, además de generar retroalimentación de fuerza en el dispositivo desde el entorno de Unity®. Adicionalmente, se empleó el plugin Vuforia Engine®, que habilita la funcionalidad de realidad aumentada, con esta herramienta, se proyectan elementos virtuales sobre el entorno físico utilizando un marcador visual. Para este trabajo se utilizó un código QR como marcador, debido a su facilidad para ser reconocido por el plugin. Para captar la imagen del entorno real en donde se encuentra el marcador y proporcionar retroalimentación visual al usuario, se usa una cámara Logitech C920 (Figura 18). Esta cámara permite visualizar en pantalla lo que ocurre físicamente con el robot esclavo mientras se manipula el sistema desde el lado maestro. Toda la lógica de control, visualización, retroalimentación y comunicación fue programada en C#, utilizando el sistema de scripting nativo de Unity®.

Elementos virtuales clave

En el entorno virtual desarrollado en Unity®, se han creado objetos fundamentales para la funcionalidad del sistema de teleoperación. Uno de los más relevantes es el Haptic Actor, que representa el stylus virtual y actúa como enlace entre el movimiento físico del usuario y su visualización en el entorno gráfico. Este objeto está sincronizado con el dispositivo físico a través del Haptic Plugin (Figura 14). Este componente permite configurar diversos parámetros de interacción, como la escala entre el espacio real y el entorno virtual, la retroalimentación de fuerzas transmitidas al usuario, así como sus propiedades de visualización y colisión. Esto posibilita una interacción háptica configurable dentro del entorno virtual.

El entorno virtual también incluye un objeto fijo que representa el RCM (Remote Center of Motion) del sistema robótico. Esta esfera (Figura 15, en rojo) sirve como referencia para calcular los ángulos y desplazamientos necesarios a partir de la posición del Haptic Actor, garantizando así que los movimientos del robot cumplan con las restricciones geométricas establecidas.

Asociado al objeto de Haptic Actor se encuentra el Haptic Collider (Figura 15), el cual define la posición deseada del efector final del robot. Este objeto es el que genera la retroalimentación háptica al interactuar con otros elementos virtuales. Aunque no tiene conexión física con la aguja Tuohy, su posición se interpreta como la ubicación de la punta de dicha aguja y se utiliza como entrada para la cinemática inversa.

Adicionalmente, en el entorno virtual se incluye un objeto fijo que representa el RCM del sistema robótico (también mostrado en la Figura 15, en la esfera roja). Este objeto sirve como referencia para calcular los ángulos y desplazamientos necesarios a partir de la posición del objeto Haptic Actor, el cual se mueve alrededor de esta esfera virtual y garantiza que los movimientos del robot cumplan con las restricciones geométricas establecidas.

[2] https://assetstore.unity.com/packages/tools/integration/haptics-direct-for-unity-v1-197034

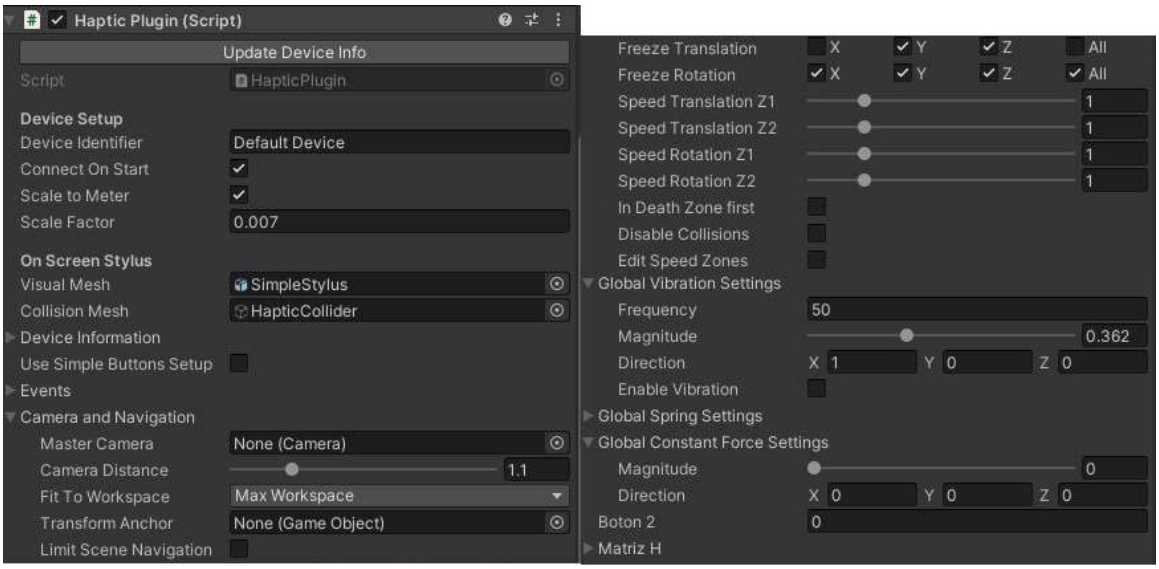


Figura 14 Configuración de Haptic Actor en Unity

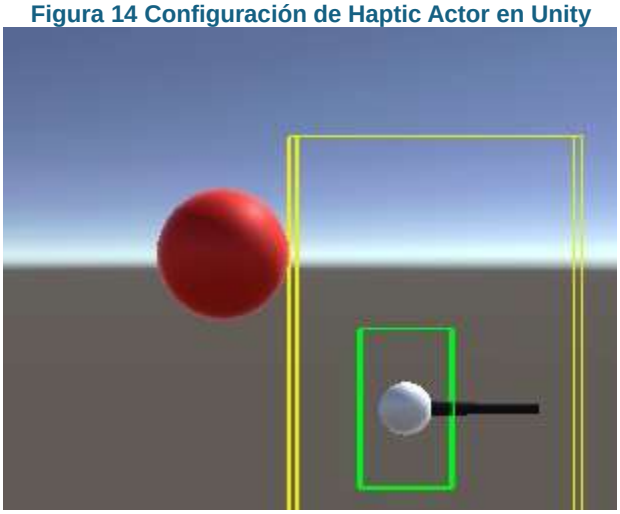


Figura 15 Objeto virtual de RCM (rojo) y stylus dictando posición del efector final (blanco)

Proceso de cálculo y comunicación

La posición del Haptic Collider respecto al RCM se utiliza como entrada para los cálculos de cinemática inversa. A partir de ella se obtienen los parámetros: θ_1: ángulo de rotación sobre el eje Z. θ_2: ángulo mecanismo del paralelogramo. L_1: desplazamiento lineal del vástago donde está montada la aguja Tuohy. Estos parámetros se obtienen utilizando funciones trigonométricas y relaciones geométricas definidas en la cinemática inversa de la sección anterior. El sistema incluye compensaciones de programación para garantizar que los movimientos sean precisos y seguros, y se establecen límites operativos que evitan colisiones y daños. La transmisión de estos comandos se realiza mediante protocolo UDP, a través de un cliente programado en Unity® que se ejecuta cada frame del entorno. Los datos solo se envían cuando el botón 1 del dispositivo háptico está presionado, asegurando que el robot solo se mueva bajo control activo del operador.

Interacción háptica y restricciones virtuales

Para asistir al usuario en la tarea de teleoperación, se implementaron restricciones virtuales que simulan límites físicos dentro del espacio de trabajo. Estas restricciones se configuran agregando la propiedad de Haptic Material a los objetos, se proporciona el efecto de interacción háptica al colisionar con el objeto de Haptic Collider previamente definido, para esto es necesario añadir la propiedad de mesh collider para crear una malla que defina el modelo del colisionador del objeto en cuestión y la propiedad de mesh render para la visualización del objeto (Figura 16). De esta forma el usuario puede sentir en forma de retroalimentación de fuerza las colisiones y límites definidos en el entorno virtual, lo que mejora el control y la precisión de la tarea Para este trabajo, dichos objetos servirán como restricciones virtuales que limiten el movimiento del robot teleoperado.

Estas se implementan agregando propiedades como Haptic Material y Mesh Collider a los objetos, lo que permite crear superficies hápticas detectables por el Haptic Collider. También se asigna la propiedad Mesh Render para su visualización (Figura 16). Así, el usuario puede sentir colisiones y límites definidos en el entorno virtual, lo que mejora el control y la precisión de la tarea. En este trabajo, estas restricciones definen zonas seguras y guían el movimiento del robot teleoperado.

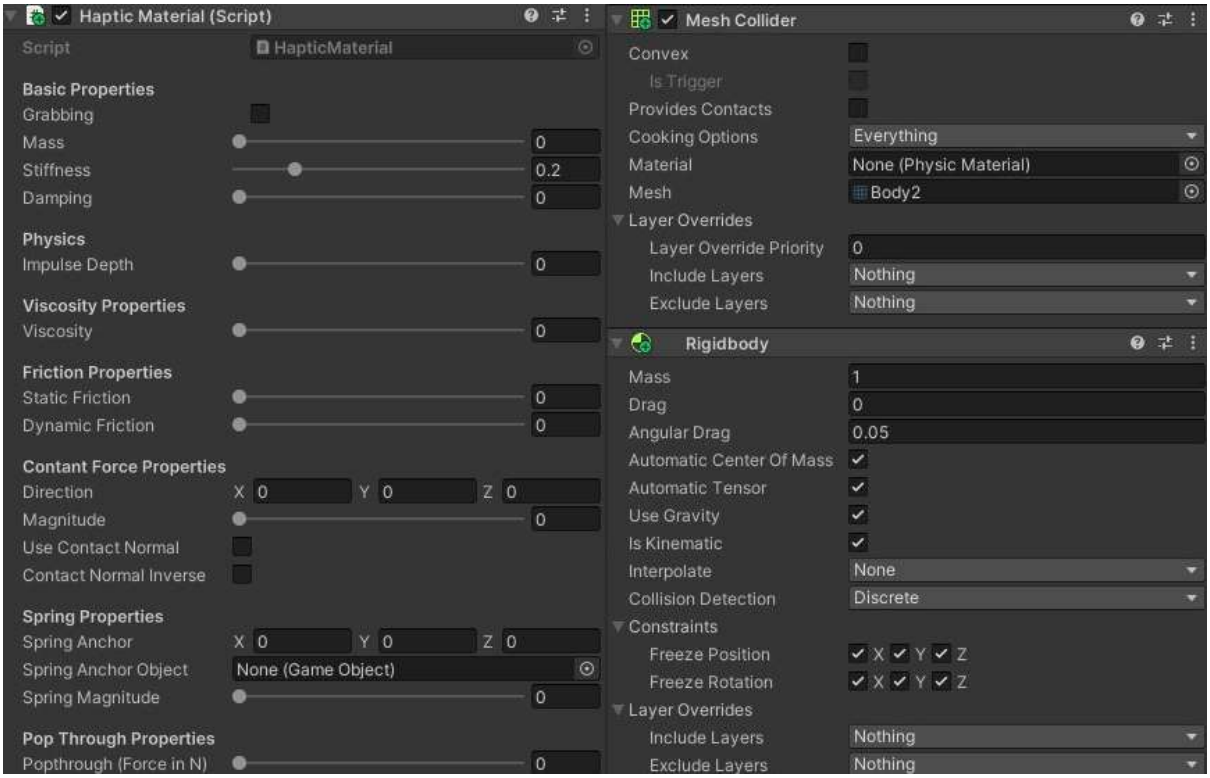


Figura 16 Parámetros para habilitar retroalimentación háptica en objetos virtuales

Realidad aumentada con Vuforia engine®

La funcionalidad de realidad aumentada se implementó mediante el complemento Vuforia Engine®[3], que permite anclar objetos virtuales al mundo físico utilizando un marcador visual. En este proyecto se empleó un código QR como marcador, captado por una cámara Logitech C920. Esto permitió superponer visualmente los objetos de Haptic Actor, el RCM y restricciones virtuales creados en Unity® en el espacio físico en el que se encuentra el robot. En Unity®, se configuró una AR Camera que reemplaza la cámara principal, y se definió el marcador como un Image Target. Los objetos virtuales fueron agregados como hijos de este marcador, de forma que su posición se encuentra asociada al marcador, asegurando su correcta alineación durante la ejecución. Se estableció una relación de escala en la cual una unidad de Unity® equivale a 10 cm en el mundo real, lo cual permite superponer los objetos virtuales de manera precisa en el espacio de trabajo.

Como asistencia visual y háptica, se diseñan dos elementos virtuales posicionados de manera continúa que restringen el movimiento de la aguja (Figura 17), con el objetivo de evitar que entre en contacto con las vértebras del modelo y asegurar que solo se inserte en una zona determinada:

- Elemento en forma de cono: Restringe la trayectoria de la aguja antes de penetrar el modelo anatómico.
- Elemento en forma de cilindro: Limita la profundidad máxima a la que la aguja puede insertarse.

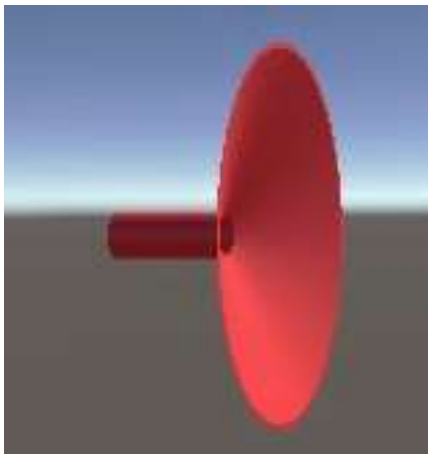


Figura 17 Visualización de restricciones cinemáticas virtuales

Estos elementos virtuales se ajustan de manera que se adaptan a la anatomía del paciente y proporcionan las restricciones necesarias al efector final del robot para asistir en la realización de la anestesia epidural.

[3] https://developer.vuforia.com/downloads/sdk

Figura 18 Cámara utilizada con el entorno de Unity®

Figura 19 Marcador para rastreo de imagen en Unity®

4.1.3 Lado del maestro

Función general del maestro

En el lado del maestro dentro del sistema de teleoperación se encuentra el dispositivo háptico Touch® de 3D Systems, conectado directamente a la computadora. Esta interfaz permite al usuario controlar remotamente la posición y orientación de la aguja quirúrgica montada en el robot esclavo. Mediante el stylus del dispositivo háptico, el operador puede ejecutar movimientos en un entorno tridimensional, mientras que recibe retroalimentación háptica de fuerzas a través del dispositivo cuando el stylus interactúa con restricciones virtuales definidas del entorno de trabajo. Con las capacidades de este dispositivo, el usuario experimenta una interacción física facilitando la tarea de teleoperación.

Especificaciones técnicas del dispositivo háptico

El Touch® de 3D Systems es un dispositivo háptico de seis grados de libertad (6-DoF), de los cuales tres tienen la capacidad de generar retroalimentación de fuerza. Está

compuesto por un brazo articulado que sostiene un stylus, el cual registra en tiempo real los movimientos del usuario. Este stylus también cuenta con dos botones programables, a los cuales se les pueden asignar funciones personalizadas.

El dispositivo se conecta a la computadora mediante puerto USB y envía continuamente información sobre la posición y orientación del stylus. La captura de estos datos es fundamental para calcular la cinemática inversa del sistema esclavo, así como para simular colisiones, resistencias o restricciones cinemáticas en el entorno virtual.

Cinemática directa del stylus

En la Figura 20, se muestra el dispositivo háptico Touch® de 3D Systems junto con una representación de sus grados de libertad y dimensiones articulares. En dicha imagen se ilustran las rotaciones asociadas a sus articulaciones, y se destacan aquellas capaces de proporcionar retroalimentación de fuerza, identificadas con la letra "A".

La posición y orientación del stylus en el espacio tridimensional, se obtiene con un modelo de cinemática directa, que se desarrolla a detalle en [60]. El modelo se construye a partir de una cadena de transformaciones homogéneas que relacionan espacialmente las distintas articulaciones del dispositivo, permitiendo sincronizar el movimiento físico del stylus con su representación virtual (Haptic Actor) en Unity®.

Las matrices de transformación homogénea utilizadas permiten representar de forma conjunta las rotaciones y traslaciones en un sistema de coordenadas unificado. En la Ecuación (9) se muestra la transformación T_{Junta3}^{Base}, que describe la posición y orientación del gimbal respecto a la base, la cual se ubica en el área central del dispositivo.

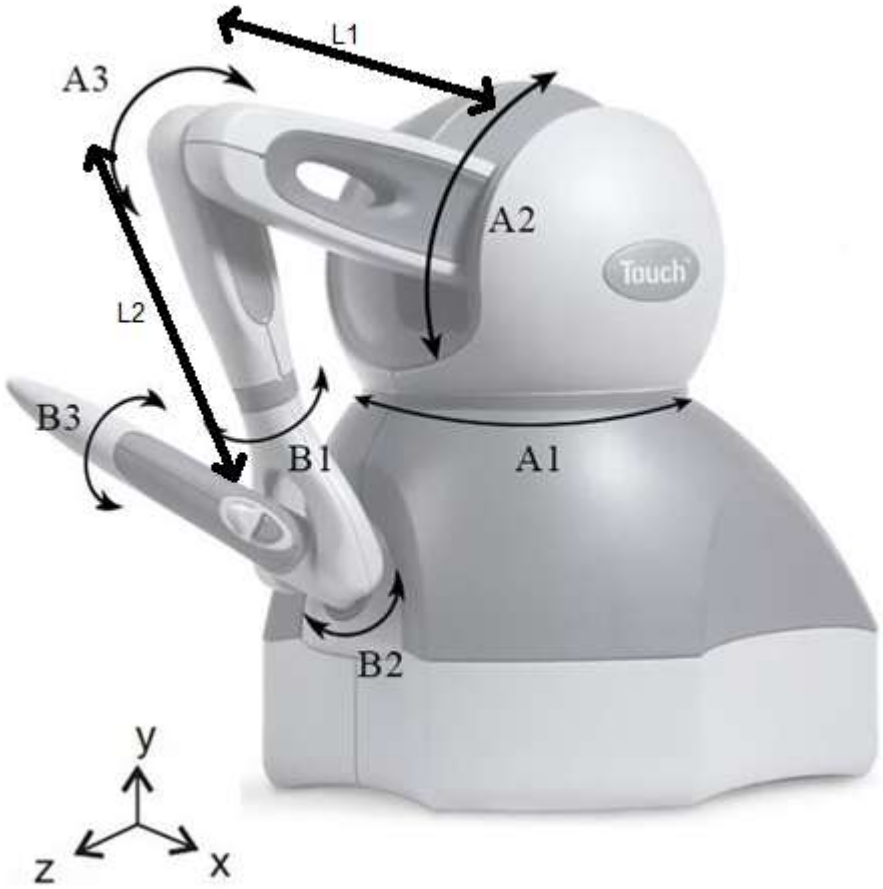


Figura 20 Dispositivo háptico touch de 3D systems con representación de movimientos y longitudes de sus juntas. Imagen obtenida de [61] y modificada para los propósitos de esta tesis.

$$T_{Junta3}^{Base} = \begin{bmatrix} C_{A1} & S_{A1}S_{A3} & -C_{A3}S_{A1} & -S_{A1}(L_1C_{A2} + L_2S_{A3}) \\ 0 & C_{A3} & S_{A3} & L_1S_{A2} + L_2C_{A3} \\ S_{A1} & -C_{A1}S_{A3} & C_{A1}C_{A3} & C_{A1}(L_1C_{A2} + L_2S_{A3}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

Una segunda transformación de relevancia es T_{Punta}^{Junta3} (Ecuación 10), que representa la posición y orientación de la punta del stylus respecto al gimbal. En esta matriz, L_3 corresponde a la distancia física entre la junta 3 y la punta del stylus:

$$T_{Punta}^{Junta3} = \begin{bmatrix} C_{B1} & -S_{B1}S_{B2} & C_{B2}S_{B1} & -L_3C_{B@}S_{B1} \\ 0 & C_{B2} & S_{B2} & -L_3S_{B2} \\ -S_{B1} & -C_{B1}S_{B2} & C_{B1}C_{B2} & -L_3C_{B1}C_{B2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

El producto de estas dos matrices permite obtener la posición y orientación absoluta de la punta del stylus en el espacio, en relación con la base del dispositivo. Aunque este modelo no se utiliza directamente para el control del robot esclavo, resulta de utilidad en el contexto de la interacción háptica, ya que permite calcular con precisión la orientación de la punta del stylus virtual y mantener la coherencia entre el movimiento físico del usuario y su representación en el sistema.

Para un análisis más detallado del desarrollo matemático y cinemático del dispositivo háptico, se recomienda consultar la fuente original [60].

4.2 Preparación del sistema para la aplicación

En esta sección se describirán las adaptaciones del sistema de teleoperación para la ejecución del procedimiento de anestesia epidural. Los aspectos abordados incluyen el montaje físico, la alineación virtual, la detección del punto de inserción, la aplicación de retroalimentación háptica, la integración de visión por computadora, realidad aumentada y simulación de fuerzas.

4.2.1 Montaje, posicionamiento del Robot y modelo anatómico

El robot se encuentra montado y fijado en una estructura que garantiza estabilidad y precisión durante la ejecución del procedimiento. El efector final queda orientado hacia una zona libre bajo la base de este, donde se ubica una mesa de trabajo, facilitando la inserción descendente de la aguja sobre esta área (Figura 21).

En esa área de trabajo, se colocará el modelo anatómico (phantom) del área de columna donde se insertará la aguja. En este caso se emplea un phantom que simula las vértebras L1 y L2. Para la creación de este modelo, se imprimió en 3D los modelos de las vértebras y se puso en un recipiente para sumergirse en gel balístico, de esta forma se obtiene la impresión 3D inmersa en el gel (Figura 22). Este phantom se coloca manualmente cerca del robot, al alcance del efector final, procurando alinear el espacio intervertebral con la línea de acción de la aguja Tuohy.

Figura 21 Acoplamiento de aguja

Siendo el objetivo insertar la aguja Tuohy hasta alcanzar el espacio epidural, se acopla un adaptador específico para la aguja Tuohy, el cual se coloca en el efector final del robot **Figura 21**.

Figura 22 Modelo anatómico de pruebas

Detección del punto de inserción mediante visión y ultrasonido Detección del punto de inserción mediante visión y ultrasonido

Para la determinación del punto de inserción, se utiliza un sistema de visión por computadora que detecta marcadores de tipo ArUco mediante una cámara Logitech C920 (Figura 23). El espacio de trabajo está referenciado mediante un marcador base, y el dispositivo de ultrasonido tiene un marcador acoplado, cuya posición y orientación son rastreadas en tiempo actual (Figura 24).

Figura 23 Uso del dispositivo de ultrasonido en el modelo

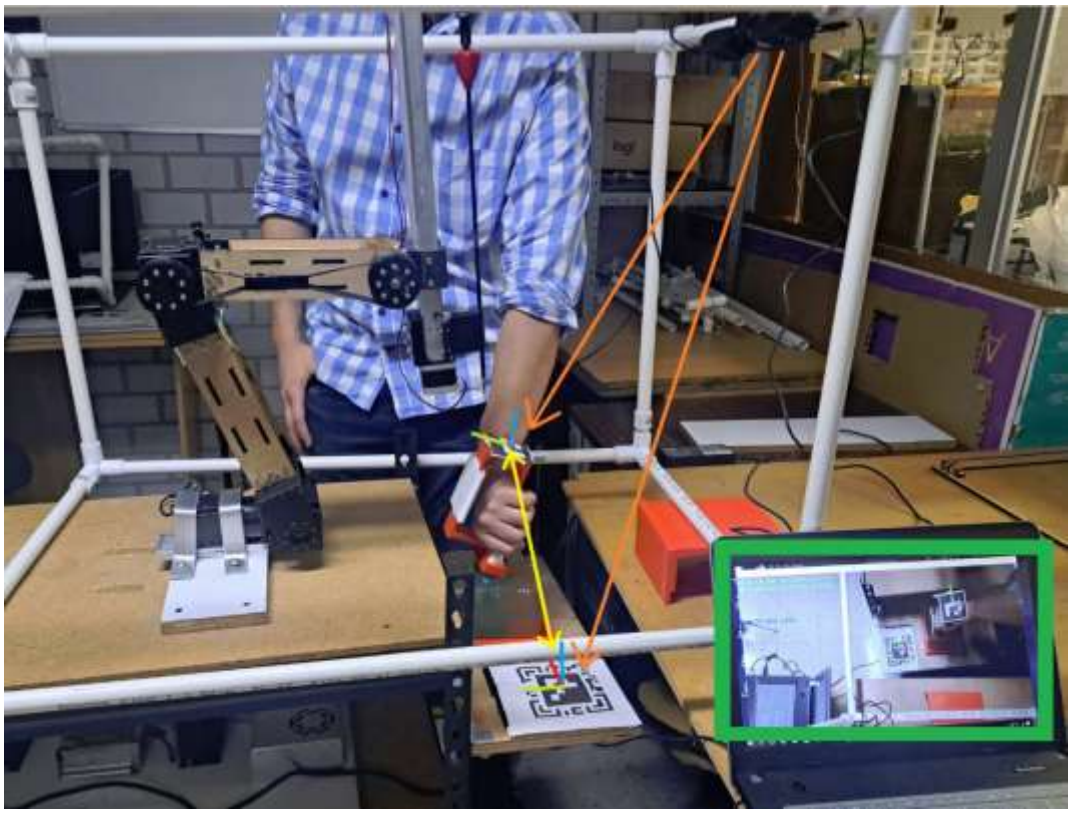


Figura 24 rastreo de marcadores para determinar coordenadas de punto de inserción

Como en el programa de rastreo se cuenta con las coordenadas y la orientación de los marcadores respecto a la cámara, a partir de los datos del rastreo, se calculan las coordenadas y orientación del punto de inserción en relación con el marcador del espacio de trabajo. Estos valores se calculan utilizando una resta de vectores y la matriz de rotación de los marcadores Figura 24. Este proceso es realizado en un programa de Python, en este caso se utiliza una computadora exclusiva para el algoritmo de visión computacional. El pseudocódigo que describe el proceso de rastreo de marcadores y envío de datos está a continuación:

```
INICIAR
    Configuración de comunicación UDP:
        Definir la dirección IP y puerto UDP
        Crear socket UDP

    Configuración de marcadores ArUco:
        Definir los IDs de los marcadores a detectar
        Definir el tamaño de los marcadores en metros
        Definir el tamaño de la imagen de video (ancho y alto)

    Cargar parámetros de la cámara (matriz de cámara y coeficientes de distorsión)
    Inicializar la captura de video desde la cámara (definir resolución)

    Bucle principal:
        Mientras el video esté activo:
            Leer un fotograma desde la cámara
```

```
        Si la lectura del fotograma es exitosa:
            Detectar los marcadores ArUco en el fotograma
            Si se detectan marcadores:
                Dibujar los marcadores detectados en el fotograma
                Estimar la pose de los marcadores:
                    Calcular vectores de rotación (rvecs) y traslación (tvecs)

                Inicializar diccionarios para almacenar posiciones y rotaciones de los marcadores

                Para cada marcador detectado:
                    Si el ID del marcador está en la lista de marcadores relevantes:
                        Guardar la posición (tvec) y rotación (rvec) del marcador
                        Ajustar el vector de rotación (invertir ciertos valores)
                        Calcular la matriz de rotación usando el vector de rotación (rvec)

                        Dibujar el sistema de ejes en el marcador
                        Mostrar las coordenadas (x, y, z) del marcador en el fotograma

                Si se detecta el marcador ID0 (origen):
                    Obtener su posición
                    Si también se detecta el marcador ID1:
                        Obtener su posición
                        Calcular la distancia entre ID0 e ID1 usando la fórmula de distancia euclidiana
                        Calcular las diferencias de posición en X, Y y Z (ajustando Z)
                        Mostrar la distancia en el fotograma

                        Calcular la matriz de rotación entre el marcador ID0 y el marcador ID1

                        Empaquetar la matriz de rotación y las distancias calculadas en un mensaje binario
                        Enviar el mensaje a través de UDP

                Mostrar el fotograma procesado con las detecciones y resultados

        Si se presiona la tecla 'q', salir del bucle

    Liberar recursos:
        Liberar la captura de video
        Cerrar las ventanas de OpenCV
FIN
```

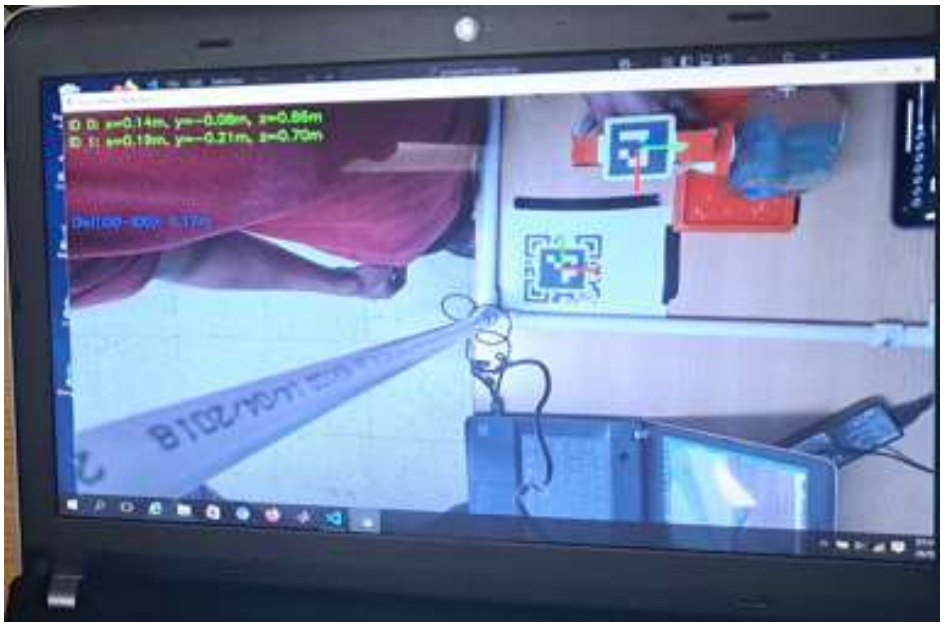


Figura 25 Sistema de visión para determinar posición y orientación de marcadores

Los datos se envían constantemente a Unity®, donde se puede visualizar la posición cambiante de los objetos virtuales según el posicionamiento del dispositivo de ultrasonido (Figura 25). El marcador del ultrasonido se coloca sobre el phantom hasta identificar el espacio intervertebral en la imagen ecográfica. Una vez que se establece el dispositivo sobre el espacio intervertebral en la orientación adecuada y es comprueba mediante las imágenes de ultrasonido obtenidas, se da el comando de detener el software de visión. El programa de Unity almacena los últimos datos recibidos y los utiliza para posicionar las restricciones virtuales en el espacio de trabajo (Figura 26). El código de Unity® que recibe los datos y ajusta los objetos virtuales se encuentra en los anexos, el pseudocódigo que lo describe es el siguiente:

```
INICIALIZAR
    Definir puerto de entrada
    Definir variable del objeto virtual a manipular como GameObject
FUNCION Start
    Crear un nuevo hilo y ejecutar InitInputServer en ese hilo

FUNCION Update
    Convertir la rotación desde quaternion a ángulos de Euler
    Ajustar los valores de rotación invirtiendo su signo donde sea necesario
    Aplicar la rotación a GameObject con los ángulos de Euler
    Ajustar la posición de GameObject usando newX, newY, newZ multiplicados por 10

FUNCION InitInputServer
    Crear un array de bytes para recibir datos
    Inicializar matrices de rotación (rotationM) y otras matrices de compensación
    Crear un punto de conexión UDP en el puerto listenPort
    Usar un UdpClient para escuchar el puerto y recibir datos en un bucle
    Mientras runTask sea verdadero:
        Recibir datos desde el cliente UDP
        Construir la matriz de rotación (rotationM) con los datos recibidos
```

Convertir la matriz de rotación en un quaternion (rotationQuat)
Extraer los valores de posición (newX, newY, newZ) de los datos recibidos

Cabe mencionar que, para este trabajo en específico, la cámara se encuentra alineada en medida de lo posible sobre el marcador del área de trabajo, esto con el fin de simplificar las operaciones computacionales, de forma que la rotación del marcador de ultrasonido respecto al marcador del entorno, es la misma que la rotación respecto a la cámara Figura 24.

De igual forma se colocan las cámaras correspondientes a cada programa en una posición donde se tenga una imagen clara del marcador y a la vez no estorben en el área de trabajo, para ello se hace uso de una estructura de prisma en la cual se pueden montar las cámaras para obtener una vista clara del área de trabajo por arriba Figura 24.

Posicionamiento y Escalado de los Objetos Virtuales (Cono y Cilindro)

Una vez obtenidos los valores deseados, se envían a Unity, donde se utilizan para posicionar las restricciones virtuales en el espacio de trabajo. Con los datos del punto de inserción capturados, los elementos virtuales (cono y cilindro) se colocan en el entorno 3D en Unity. El cono se sitúa fuera del phantom, mientras que el cilindro inicia en su superficie y termina justo en el espacio epidural. Estos objetos definen la ruta segura de inserción. Los objetos virtuales se ajustan de manera que se adaptan a la anatomía del paciente y proporcionan las restricciones necesarias al efector final del robot para asistir en la realización de la anestesia epidural (Figura 26).

El código de Unity que recibe los datos y ajusta los objetos virtuales se encuentra en los anexos, el pseudocódigo que lo describe es el siguiente:

```
INICIALIZAR
   Definir puerto de entrada
   Definir variable del objeto virtual a manipular como GameObject
FUNCION Start
   Crear un nuevo hilo y ejecutar InitInputServer en ese hilo

FUNCION Update
   Convertir la rotación desde quaternion a ángulos de Euler
   Ajustar los valores de rotación inviertiendo su signo donde sea necesario
   Aplicar la rotación a GameObject con los ángulos de Euler
   Ajustar la posición de GameObject usando newX, newY, newZ multiplicados por 10

FUNCION InitInputServer
   Crear un array de bytes para recibir datos
   Inicializar matrices de rotación (rotationM) y otras matrices de compensación
   Crear un punto de conexión UDP en el puerto listenPort
   Usar un UdpClient para escuchar el puerto y recibir datos en un bucle
   Mientras runTask sea verdadero:
      Recibir datos desde el cliente UDP
      Construir la matriz de rotación (rotationM) con los datos recibidos
      Convertir la matriz de rotación en un quaternion (rotationQuat)
      Extraer los valores de posición (newX, newY, newZ) de los datos recibidos
```

Otro factor a considerar, es que, debido a las características mecánicas del robot, es necesario alinear las restricciones virtuales con el RCM. Considerando que este ya está representado de manera virtual en Unity, el usuario alinea los objetos virtuales con el RCM físico del robot desplazando el dispositivo de ultrasonido. Para asistir a una correcta alineación, se incluye una guía virtual con forma de línea vertical que cruza el centro del cono. Esta línea desaparece cuando colisiona con el RCM virtual, indicando visualmente al usuario que los elementos están alineados Figura 26.

Figura 26 Objeto virtual de asistencia para alineación de restricciones

El pseudocódigo para la guía de alineación se describe a continuación:

```
Inicio

Definir una variable para el objeto Sphere (esfera)

Función OnTriggerEnter(Colisión con otro objeto):
    Si el objeto con el que ha colisionado es la esfera (Sphere):
        Desactivar la visibilidad del objeto actual (deshabilitar MeshRenderer)

Función OnTriggerExit(Al salir de la colisión con otro objeto):
    Si el objeto con el que dejó de colisionar es la esfera (Sphere):
        Activar la visibilidad del objeto actual (habilitar MeshRenderer)

Fin
```

Una vez que el programa de Unity está activo, mostrando las restricciones virtuales en la posición correcta dentro de la interfaz de realidad aumentada sobre el modelo anatómico, se puede avanzar a la etapa intraoperatoria. El dispositivo de ultrasonido seguirá siendo de utilidad para obtener una imagen del ultrasonido en tiempo actual durante la ejecución

del procedimiento, permitiendo al usuario observar la progresión de la aguja y recibir asistencia visual.

Modelo de Fuerzas Simuladas del Tejido

Para esta aplicación, se adapta un modelo de fuerzas en función de la distancia de inserción de la aguja simulando las fuerzas que se experimentan al atravesar los diferentes tejidos hasta llegar al espacio epidural. El modelo de estas fuerzas es extraído de [62] en el cual son captadas con un sensor de fuerzas al hacer una punción vertical hasta el espacio epidural en una muestra anatómica real. Para su aplicación, las fuerzas siguen el modelo en un programa que toma como entrada su posición y las calcula en base a ella (ver anexos). La simulación incluye un conjunto de fuerzas graficadas en la Figura 27:

- Fuerza creciente: A medida que la aguja penetra más profundamente, la fuerza de simulación aumenta.
- Picos de fuerza: Se simulan dos picos de resistencia correspondientes a la penetración del ligamento supraespinoso (fuerza de menor magnitud) y del ligamento flavum (fuerza de mayor magnitud).

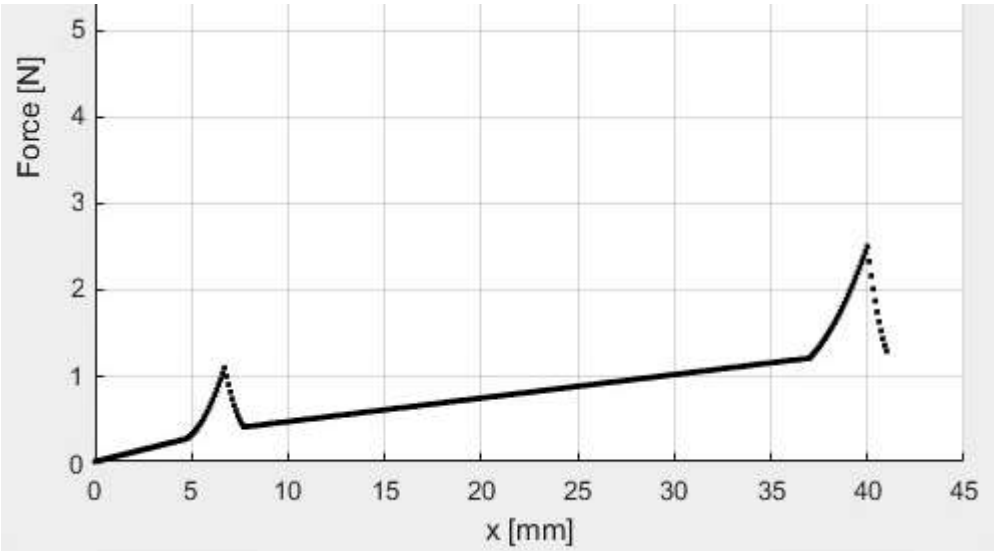


Figura 27 Gráfica de fuerzas de simulación contra distancia recorrida

Estas fuerzas se activan cuando el operador presiona el botón secundario y su magnitud depende de la distancia recorrida desde el inicio de este modo. Las fuerzas se aplican en dirección contraria a la del movimiento de la aguja, lo que depende de la orientación del usuario en la interfaz del dispositivo háptico. Al incorporar estas simulaciones de fuerzas, el operador puede sentir las variaciones de resistencia durante el proceso de inserción, lo que le ayuda a guiarse por el progreso del procedimiento. El pseudocódigo para las fuerzas de simulación se describe a continuación:

INICIALIZAR
Definir parámetros de entrada como longitud de inserción y picos de fuerza (picoA, picoC).
Inicializar valores para los segmentos A y C del modelo matemático según los parámetros del usuario (ubicación de los picos).

INICIAR (al iniciar el script):
Calcular inicio y fin de los picos de fuerza usando la longitud de inserción y magnitud.
Calcular los valores dependientes de la entrada del usuario para el modelo matemático

AL INICIAR:
Guardar la posición inicial del actuador cuando se presiona el botón de inicio (SetStart).

AL TERMINAR:
Detener las fuerzas de simulación cuando se presiona el botón de fin (EndForce).

FUNCION "calcForce":
Entrada: dist (distancia desde el inicio)

SI dist < 0:
Retornar 0 (sin fuerza)

Calcula las fuerzas para el segmento A (si la distancia está en el rango de inicioA a finA):
Si dist está entre inicioA y finA:
- Calcular las fuerzas de inserción para el segmento A dependiendo de la distancia.

Calcula la fuerza de corte (Fcut) y fricción (Ffriction) para el segmento B:
- La fuerza de corte es una función de la distancia hasta el picoA.
- La fricción aumenta conforme se acerca al picoC.

Calcula las fuerzas para el segmento C (si la distancia está en el rango de inicioC a finC):
Si dist está entre inicioC y finC:
- Calcular las fuerzas de inserción para el segmento C dependiendo de la distancia.

Sumar las fuerzas A, B (fricción y corte), y C:
- La fuerza B se ajusta para no desentonar con las fuerzas A y C.
Retornar la fuerza total.

AL ACTUALIZAR:
Obtener la posición actual del estilete (punta).
Calcular la distancia entre el punto inicial (inicioX, inicioY, inicioZ) y la posición actual del actuador.
Convertir la distancia a milímetros y calcular la fuerza de inserción usando "calcForce".
Ajustar la fuerza calculada según la matriz de transformación (para aplicar la fuerza en la dirección correcta).
Aplicar la fuerza al dispositivo háptico usando el plugin.

Las fuerzas del modelo de simulación en función a la distancia recorrida se suman a las fuerzas obtenidas por colisión con objetos virtuales para obtener la retroalimentación háptica en tiempo actual que experimenta el usuario mediante el dispositivo háptico.

Integración del sistema

Todos los módulos descritos se integran en una arquitectura funcional que permite las tareas de:

- Captura de imágenes con marcadores.
- Cálculo de posición y orientación del dispositivo de ultrasonido.
- Transmisión de datos por UDP.

- Posicionamiento de objetos virtuales en Unity®.
- Simulación de fuerzas hápticas durante la inserción.

El diagrama de la Figura 28 resume la interacción entre los scripts y módulos que conforman el sistema.

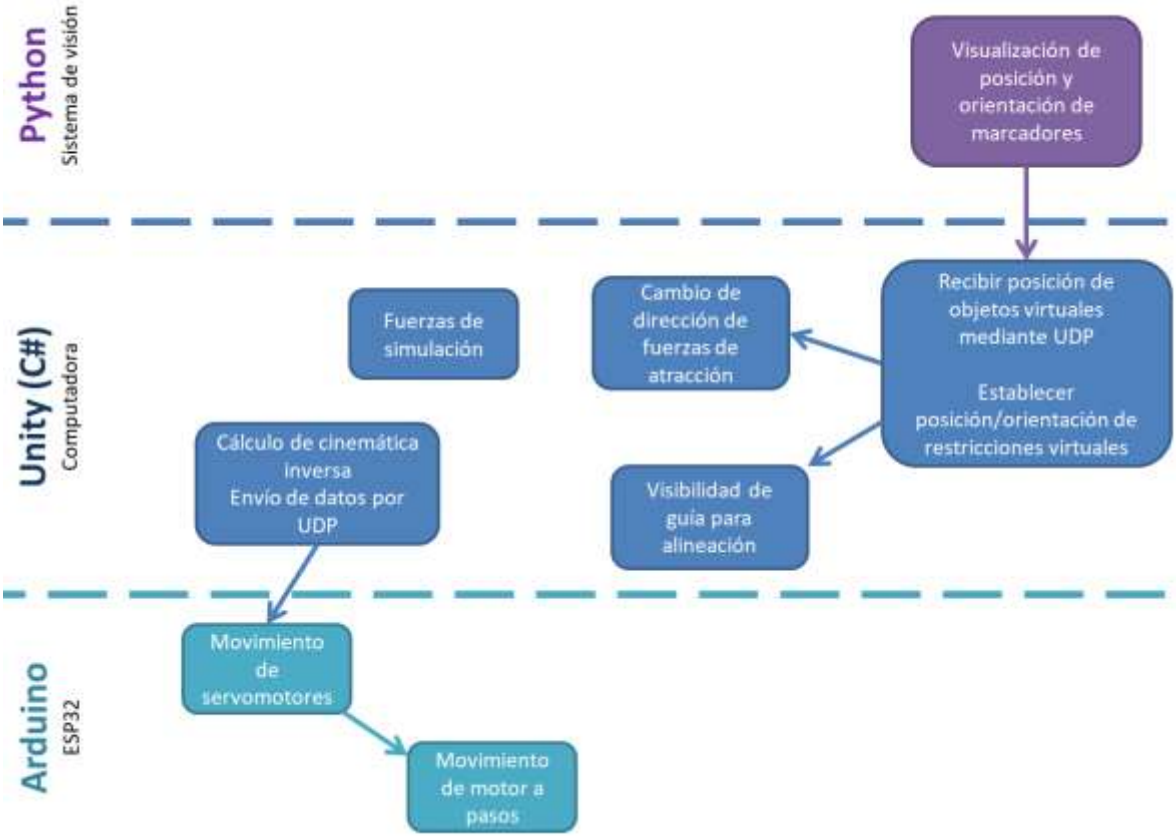


Figura 28 Diagrama de interacción de programas

Una vez descritas las adaptaciones al modelo de teleoperación para poder ejecutar la tarea de insertar una aguja Tuohy en el espacio epidural, se procede a hacer los experimentos.

División de la tarea en etapas

Para facilitar la ejecución del procedimiento, se definieron dos etapas operativas según el avance de la intervención:

- Etapa de Orientación: el operador presiona el botón 1 del dispositivo háptico para activar la alineación automática del robot, que posiciona la aguja según el punto y el ángulo de inserción determinados previamente mediante visión y ultrasonido.

- Etapa de Inserción: una vez alineada la aguja, el operador ejecuta la inserción lineal hasta alcanzar la profundidad deseada. Durante esta fase, se activa la retroalimentación háptica basada en el modelo de simulación de tejidos.

Con el sistema ya configurado, se procede a la validación experimental, descrita en la siguiente sección.

4.3 Experimento

En esta sección se describirá paso por paso la realización de la prueba de anestesia epidural asistida por robot teleoperado sobre el modelo anatómico de columnas previamente descrito. La prueba se divide en dos etapas: la preoperatoria y la intraoperatoria. La primera consiste en la preparación del espacio de trabajo y del equipo necesario para realizar la tarea, mientras que la segunda es la fase en la que se ejecuta la tarea.

Etapa Preoperatoria

Al inicio de esta etapa, se posiciona el phantom con las vértebras L1 y L2 en el área de trabajo y se inicializa el programa de Unity (para visualización y retroalimentación háptica) y el de visión computacional (para detección de marcadores) en sus respectivas computadoras. Se espera a que ambos programas detecten correctamente el marcador del área de trabajo antes de proseguir. Ambos sistemas fueron configurados para comunicarse vía red local mediante protocolo UDP, verificando que las direcciones IP, puertos y el estado del firewall permitieran una transmisión fluida de datos.

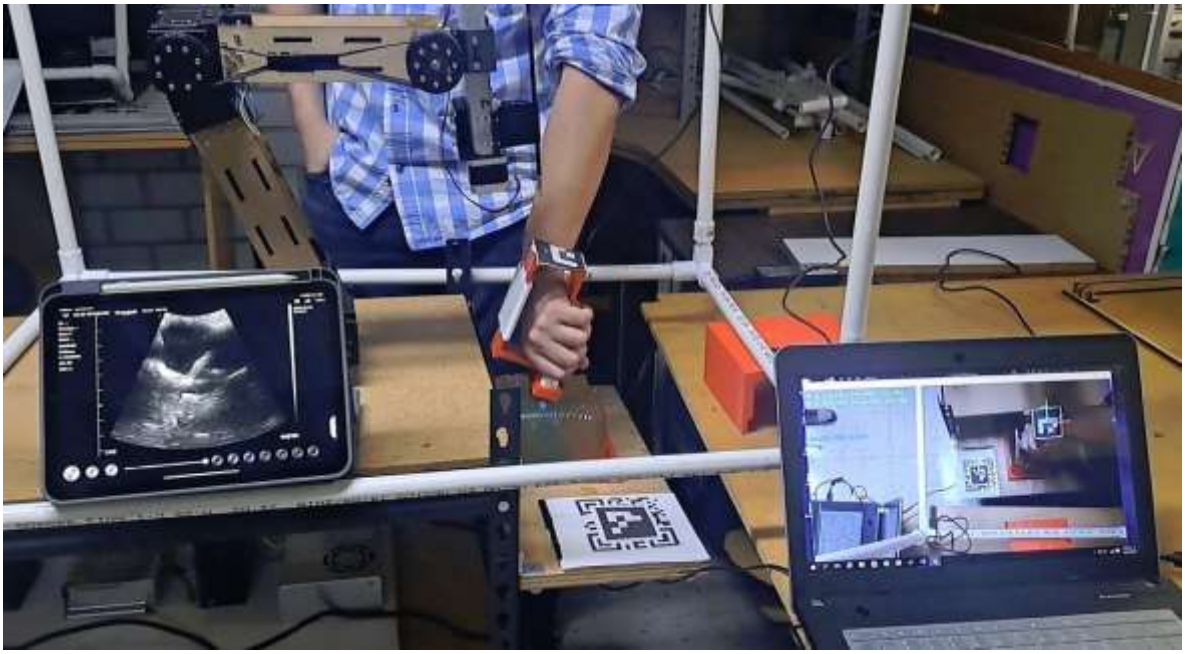


Figura 29 Uso de ultrasonido para determinar punto de inserción

El usuario emplea el ultrasonido con un marcador visual Apriltag para localizar el espacio intervertebral (Figura 29). El sistema define coordenadas, ángulo y profundidad. Una vez que se establece el dispositivo sobre el espacio intervertebral en la orientación adecuada y es comprueba mediante las imágenes de ultrasonido obtenidas, se da el comando de

detener el software de visión. Cuando esto sucede, en el programa de Unity se le quedan asignados los datos al objeto de guía virtual. En el este caso, el modelo a utilizar fue el mostrado en la Figura 22 y este cuenta con una profundidad de 24 mm hasta el inicio de la zona epidural y se determinó un ángulo de inserción de 15° (Figura 30).

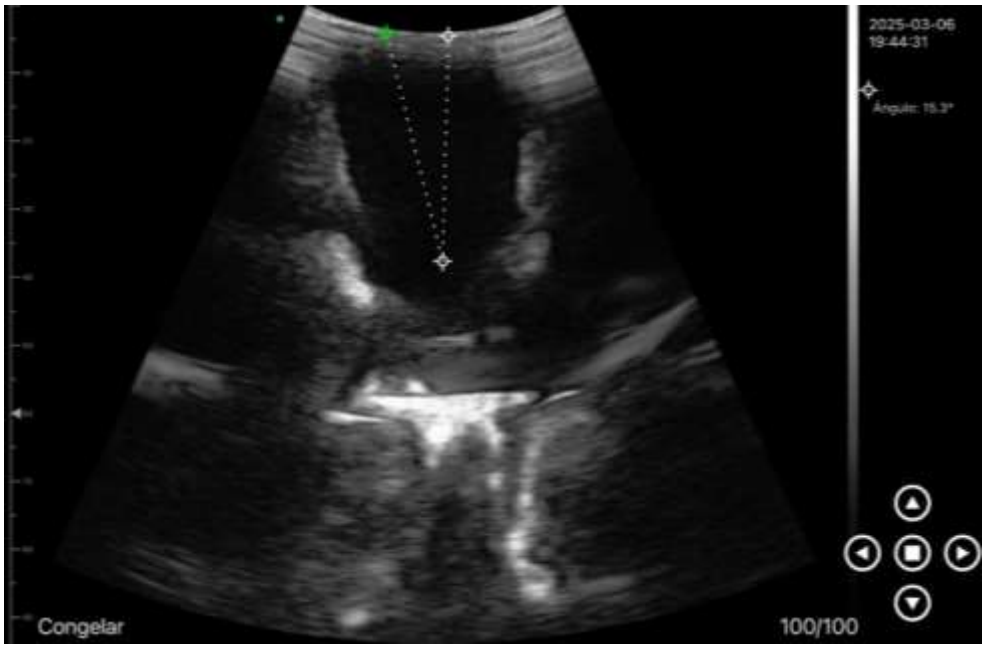


Figura 30 Profundidad del espacio epidural y ángulo de inserción en el modelo

Etapa Intraperatoria

Con el punto y ángulo de inserción definidos, se procedió a ejecutar el procedimiento. El operador activó el botón 1 del stylus del dispositivo háptico para que el robot realizara la alineación automática de la aguja. Una vez alineada, el operador soltó el botón 1 y presionó el botón 2 para entrar en la etapa de inserción, controlando únicamente el actuador lineal del robot.

La visualización de Unity y la imagen de ultrasonido permiten supervisar la ejecución en tiempo actual teniendo un control sobre la penetración de la aguja en el modelo a la vez que experimenta fuerzas de simulación (Figura 31). Se realizaron los siguientes escenarios experimentales para comprobar la efectividad de las restricciones virtuales:

- Inserción con solo modelo de fuerzas de tejido (sin objetos virtuales): en este caso se desactivaron los objetos virtuales que imponen restricciones para realizar la inserción de la aguja, de forma que el usuario depende únicamente de las fuerzas de simulación para determinar la posición de la aguja.
- Inserción con modelo de fuerzas y elementos virtuales (cono y cilindro): para esta prueba se activan los objetos de restricción virtual, por lo que se además de las fuerzas de simulación, se tiene una guía visual que indica la trayectoria y profundidad hacia el espacio epidural en el modelo y retroalimentación háptica al llegar a ese punto.

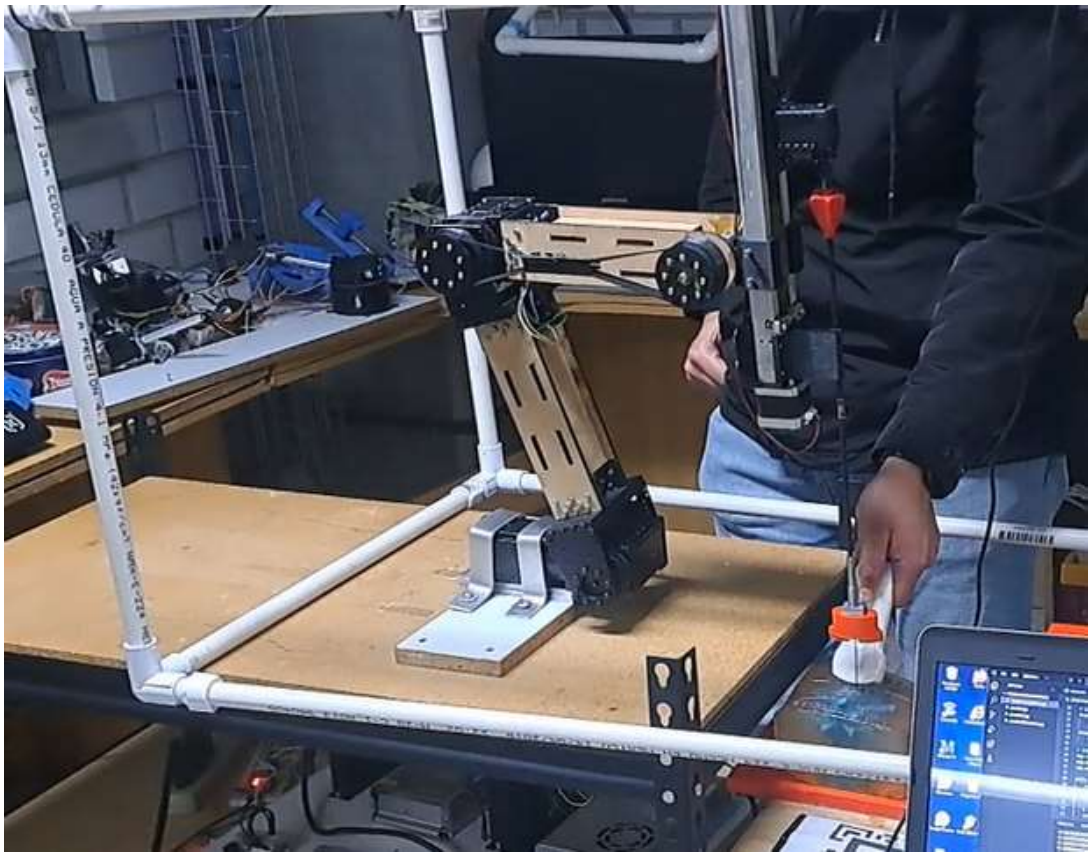


Figura 31 Inserción de aguja en el modelo

5 Resultados y Análisis

En este capítulo se presentan los resultados obtenidos durante la evaluación experimental del sistema de teleoperación diseñado para asistir la inserción de una aguja Tuohy en el espacio epidural mediante el empleo de restricciones virtuales y retroalimentación háptica desarrollado en esta tesis. Es importante recalcar que las pruebas fueron realizadas por un usuario sin experiencia previa en procedimientos anestésicos, para valorar la capacidad del sistema para guiar a operadores novatos en una tarea técnica de alta precisión.

En la etapa previa a la inserción, se determinó mediante ultrasonido que la distancia entre la superficie del modelo anatómico y el inicio de su espacio epidural es de 24 mm. Este valor fue utilizado como referencia para evaluar la precisión de las inserciones ejecutadas en los diferentes escenarios experimentales.

Una vez completada la alineación automática del robot hacia el punto y ángulo de inserción definidos, el operador realizó dos pruebas independientes: una con ayuda de objetos virtuales que establecen restricciones cinemáticas que generan retroalimentación háptica al alcanzar la profundidad límite, y otra sin dichas restricciones, en la cual el usuario se guía exclusivamente con la percepción de fuerzas simuladas.

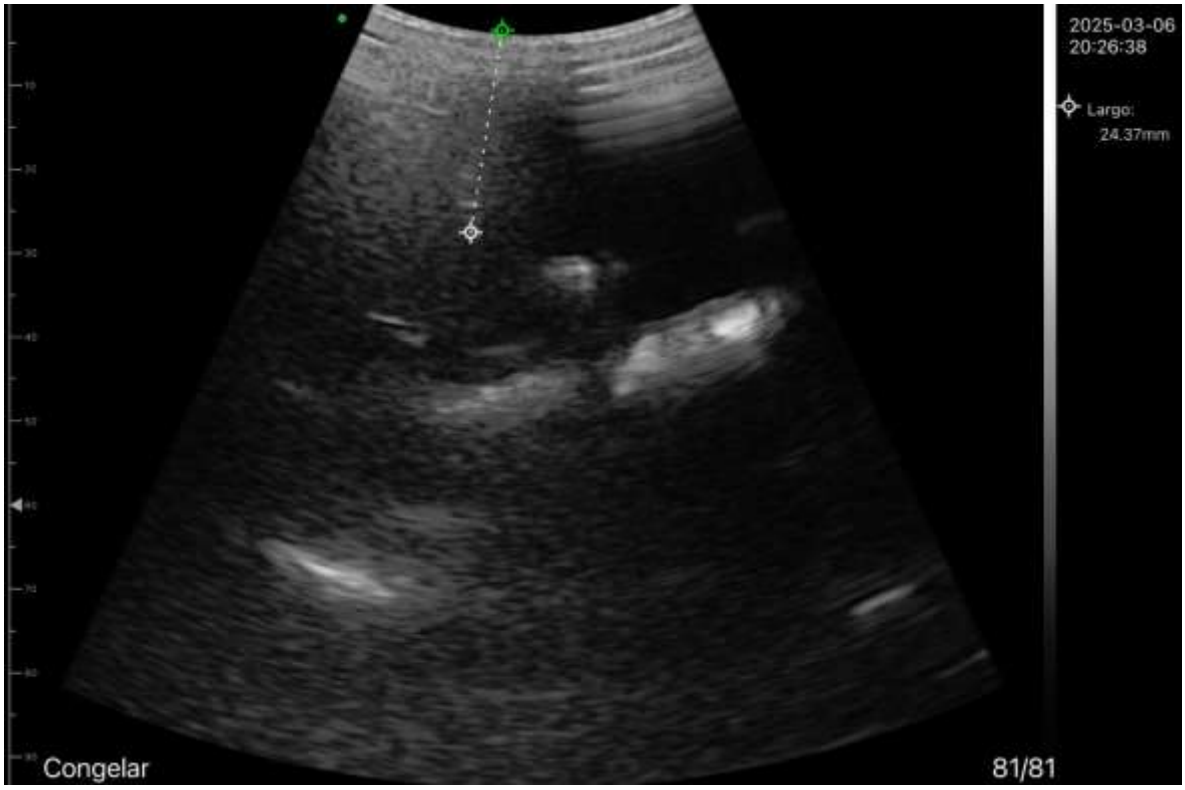


Figura 32 Inserción de aguja con restricciones virtuales

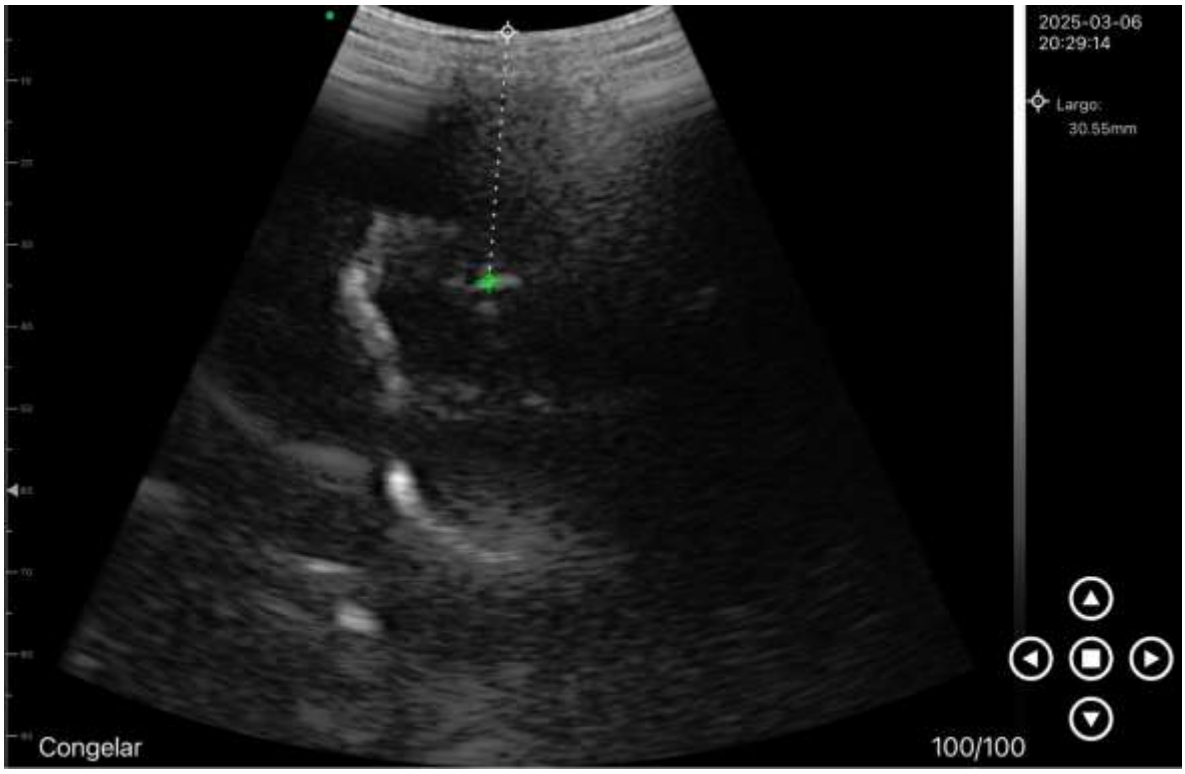
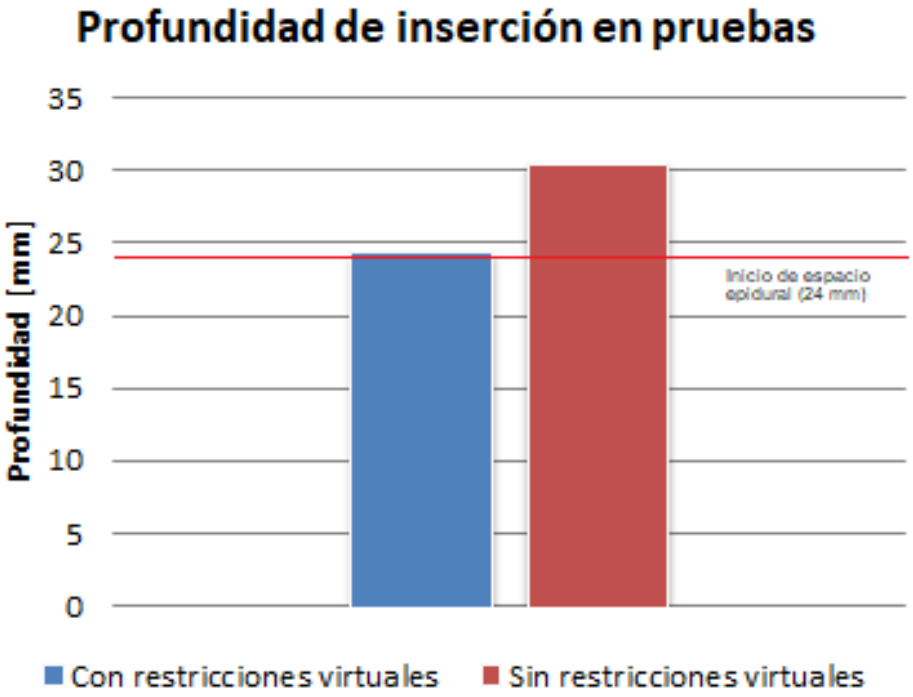


Figura 33 Inserción de aguja sin restricciones virtuales

En la prueba con ayuda de objetos virtuales (figura 32), la inserción alcanzó una profundidad de 24.37 mm, lo que representa 0.37 mm más de la distancia real al espacio epidural. Esto demuestra que la restricción cinemática actuó como una barrera efectiva, previniendo un sobrepaso de la aguja que pudieran comprometer estructuras internas en un caso real. Además, la retroalimentación háptica emitida al alcanzarse el límite virtual facilitó al usuario identificar el punto correcto de detención.

En el segundo caso sin ayuda de objetos virtuales (figura 33), el usuario continuó el avance de la aguja hasta alcanzar una profundidad de 30.55 mm, lo que implica un sobrepaso de 6.55 mm respecto al objetivo. Este error es consistente con las limitaciones esperadas al depender únicamente de sensaciones hápticas en operadores inexpertos, demostrando la dificultad para distinguir los cambios en resistencia simulada cuando se atraviesan los tejidos del modelo.

La comparación de ambos valores muestra una diferencia marcada entre resultados, con una precisión significativamente superior al emplear restricciones cinemáticas. En la figura 34 se ilustra esta diferencia mediante un gráfico de barras acompañado de una línea de referencia correspondiente a la profundidad epidural de 24 mm.



Los resultados obtenidos demuestran que el uso de elementos virtuales y restricciones cinemáticas integrados en un sistema de teleoperación mejora de manera notable la precisión de la inserción al delimitar la profundidad a la que se permite insertar la aguja, especialmente útil en usuarios con poca o nula experiencia en el procedimiento. Esto respalda el uso de la realidad mixta como herramienta de apoyo para reducir errores técnicos durante procedimientos guiados por robot.

6 Conclusiones y Discusión

Con ayuda de las pruebas realizadas, se comprobó que el sistema robótico utilizado en este trabajo es capaz de realizar el procedimiento de inserción de una aguja tuohy en el espacio epidural al hacer uso de un modelo físico para simular la anatomía de la zona intervertebral. En las pruebas realizadas, se puede notar que el uso de las restricciones virtuales, así como su visualización, asisten al usuario y le permiten llevar a cabo de forma exitosa la inserción de la aguja en la zona epidural del modelo. Al comparar el resultado contra las pruebas sin uso de restricciones virtuales, se nota que el sistema depende más de la habilidad del usuario para detener la aguja en la profundidad adecuada guiándose por las fuerzas de simulación. Estas fuerzas fueron de utilidad en ambos casos ya que con ellas el usuario puede determinar la posición en donde se encuentra la aguja durante el proceso.

Para la alineación de la aguja a la posición de inserción, se determina que, aunque la alineación automática facilita el proceso y permite al usuario enfocarse únicamente en la tarea de penetración, la forma de alineación manual ofrece una alternativa para que el usuario personalice la posición de inserción a una que él pueda considerar más favorable.

Cabe recalcar que estas observaciones son derivadas de pruebas realizadas por un usuario con baja habilidad en este procedimiento, sin embargo, esto también demuestra que el uso de este sistema robótico con la habilitación de restricciones virtuales que tienen efecto en la interfaz háptica, permite a dicho usuario con baja experiencia realizar el procedimiento de punción epidural de manera exitosa en el modelo físico utilizado.

Se concluye que para una obtención más amplia de resultados, se puede realizar una serie de pruebas extensivas para obtener una medición cuantitativa del margen de error en cada caso. De igual forma también se propone incluir a usuarios con mayor experiencia en el procedimiento para poder comparar los resultados contra los obtenidos con usuarios de baja habilidad y obtener una evaluación de la ayuda que presenta el sistema de teleoperación en el procedimiento de anestesia epidural dada por un profesional.

Otro aspecto a mejorar es la utilización de un modelo de pruebas que sea más anatómicamente preciso y presente menores deformaciones con la inserción de la aguja. Esto se debe a que el modelo usado en este proyecto se cuarteaba con las inserciones repetidas, lo cual llegaba a provocar ruido en la imagen del ultrasonido. De igual forma, para la propuesta de modelos en trabajos futuros, se debe tener preferencia por aquellos donde se pueda realizar la visualización mediante ultrasonido para obtener datos precisos de los resultados.

Junto a estas observaciones, se puede añadir que el éxito de este trabajo fue posible por el correcto funcionamiento del sistema de visión propuesto para las tareas de desplegar los objetos virtuales sobrepuestos al ambiente real y obtener la ubicación del dispositivo de ultrasonido. Sin embargo, la precisión de este era variable y los trabajos futuros se beneficiarían de un sistema mejorado de rastreo.

Bibliografía

[1] E. K. C. Kim, Robotics in General Surgery, New York: Springer, 2014.

[2] G. D., «Robotic -assisted surgery: the future is here,» *J Healthc Manag,* vol. 48, pp. 242-51, 2003.

[3] F. P.-E. D. P.-M. a. V. J. G.-V. D. Haro-Mendoza, «Needle path planning in semiautonomous and teleoperated robot-assisted epidural anaesthesia procedure: A proof of concept,» *Int. J. Med. Robotics Comput. Assisted Surg,* vol. 18, nº 6, pp. 24-34, 2022.

[4] S. P. e. al, «Complications of unintentional dural puncture during labour epidural analgesia: a 10-year retrospective observational study,» *J. Anesthesia Analgesia Crit. Care,* vol. 3, nº 1, p. 42, 2023.

[5] C. T. S. B. D. T.-J. a. J. C. N. Hollister, «Minimising the risk of accidental dural puncture with epidural analgesia for labour: a retrospective review of risk factors,» *Int. J. Obstet. Anesthesia,* vol. 21, nº 3, pp. 236-241, 2012.

[6] S. M. Nimmo, «Benefit and outcome after epidural analgesia,» *Continuing Educ. Anaesthesia Crit. Care Pain,* vol. 4, nº 2, pp. 44-47, 2004.

[7] M. Behrends, «Epidural anesthesia: Complications and side effects,» Pain Management Education at UCSF, [En línea]. Available: https://pain.ucsf.edu/neuraxial-anesthesia/epidural-anesthesia-complications-and-side-effects. [Último acceso: 16 Feb 2025].

[8] MyMediTravel, «Anesthetics Clinics in Mexico | 2024 Prices,» 2024. [En línea]. Available: https://www.mymeditravel.com. [Último acceso: 10 2 2025].

[9] J. R. Guardado, «Policy Research Perspectives: Medical Liability Claim Frequency Among U.S. Physicians,» Am. Med. Assoc., 2023. [En línea]. Available: https://www.ama-assn.org/system/files/policy-research-perspective-medic. [Último acceso: 16 Feb 2025].

[10] R. K. I. D. M. S. a. V. C. Z. Friedman, «Objective assessment of manual skills and proficiency in performing epidural anesthesia--video-assisted validation,» *Reg. Anesthesia Pain Med,* vol. 31, nº 4, pp. 304-310, 2006.

[11] N. S. S. a. K. N. Khrapov, «Epidural anesthesia in abdominal surgery,» *Messenger of Anesthesiology and Resuscitation,* vol. 19, nº 2, pp. 64-73, 2022.

[12] T. B. Sheridan, «Telerobotics,» *Automatica,* vol. 25, nº 4, p. 487–507, 1989.

[13] T. B. Sheridan, «Teleoperation, telerobotics and telepresence: A progress report,» *Control Eng. Practice,* vol. 3, nº 2, p. 205–214, 1995.

[14] O. D. Hernández, *Intuición artificial aplicada a la teleoperación,* Ciudad de México: Universidad Nacional Autónoma de México, 2014.

[15] J. C. a. R. R. Murphy, «Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center,» *IEEE Trans. Syst. Man Cybern.,* vol. 33, nº 3, p. 367–385, 2003.

[16] J. Y. X. Y. C. H. a. X. G. T. Gao, «Teleoperation system of autonomous underwater vehicle toward human-on-the-loop: Design and implementation,» *IEEE 13th Int. Conf. CYBER Technology in Automation, Control, and Intelligent Systems,* p. 142–147, 2023.

[17] J. E. S. A. M. L. G. V. G.-J. a. J. T. C. González, «Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback,» *J. Manuf. Syst,* vol. 59, p. 283–298, 2021.

[18] V. M. M. M. P. M. D. V. a. P. V. R. De Fazio, «Human–Machine Interaction through Advanced Haptic Sensors: A Piezoelectric Sensory Glove with Edge Machine Learning for Gesture and Object Recognition,» *Future Internet,* vol. 15, nº 1, p. 14, 2023.

[19] A. M. Okamura, «Methods for haptic feedback in teleoperated robot-assisted surgery,» *Ind. Robot,* vol. 31, nº 6, p. 499–508, 2004.

[20] K. J. Kuchenbecker, «Haptics and haptic interfaces,» *Encyclopedia of Robotics,* 2018.

[21] F. A. S. V. M. M. a. C. P. P. Kourtesis, «Electrotactile feedback applications for hand and arm interactions: A systematic review, meta-analysis, and future directions,» *IEEE Trans. Haptics,* vol. 15, nº 3, p. 479–496, 2022.

[22] Z. S. Z. Z. Q. S. T. H. H. L. T. C. a. C. L. M. Zhu, «Haptic-feedback smart glove as a creative human-machine interface (HMI) for virtual/augmented reality applications,» *Science Advances,* vol. 6, nº 19, 2020.

[23] J. L. Y. P. Z. L. a. C.-Y. S. C. Yang, «Personalized variable gain control with tremor attenuation for robot teleoperation,» *IEEE Trans. Syst. Man Cybern. Syst.,* vol. 48, nº 10, p. 1759–1770, 2018.

[24] G. S. R. V. a. C. P. J. Singh, «Exploring the evolving landscape of extended reality (XR) technology,» *3rd Int. Conf. Smart Generation Computing,* p. 1–6, 2023.

[25] X. X. M. Y. A. a. O. M.-P. V. R. Curran, «V. R. Curran, X. Xu, M. Y. Aydin, and O. Meruvia-Pastor,» *Med. Sci. Educ.,* vol. 33, nº 1, p. 275–286, 2022.

[26] J. Franchi, «Virtual reality: An overview,» *TechTrends,* vol. 39, p. 23–26, 1994.

[27] A. B. A. K. T. J. a. T. V. S. Saju, «Augmented reality vs virtual reality,» *Int. J. Eng. Technol. Manag. Sci,* 2022.

[28] Z. M. a. H. Varol, «Augmented Reality for Robotics: A Review,» *Robotics,* vol. 9, nº 2, p. 21, 2020.

[29] B. H. a. M. N. M. Speicher, «What is Mixed Reality?,» *Conf. Human Factors Comput. Syst,* p. 1–15, 2019.

[30] M. S. A. M. N. N. D. G. P. K. a. R. T. H. Ishida, «Haptic-assisted collaborative robot framework for improved situational awareness in skull base surgery».

[31] B. P. Y. F. a. Y. A. H. Qiu, «Surgical Instruments Motion Safety Constraint Based on Haptic Virtual Fixture,» *IEEE Int. Conf. Mechatronics Autom.,* pp. 2267-2272, 2019.

[32] H. J. M. K. -W. K. a. G. -Z. Y. K. Leibrandt, «Implicit active constraints for a compliant surgical manipulator,» *IEEE International Conference on Robotics and Automation,* pp. 276-283, 2014.

[33] B. J. Q. C. T. A. a. Y. H. Y. Zhu, «A Shared Control Framework for Enhanced Grasping Performance in Teleoperation,» *IEEE Access,* vol. 11, pp. 69204-69215, 69204-69215.

[34] P. M. a. S. Sirouspour, «A task-space weighting matrix approach to semi-autonomous teleoperation control,» *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 645-652, 2011.

[35] J. M. P. O. A. Abbott, «Haptic Virtual Fixtures for Robot-Assisted Manipulation,» *Robotics Research. Springer Tracts in Advanced Robotics,* vol. 28, 2007.

[36] H. a. S. Calinon, «Learning from demonstration for semi-autonomous teleoperation,» *Auton. Robot.,* vol. 43, p. 713–726, 2019.

[37] C. P. a. G. Yang, «Hand-Held Medical Robots,» *Ann. Biomed. Eng,* vol. 42, p. 1594–1605, 2014.

[38] K. C. V. W. T. L. C. Y. S. C. N. R. a. W. T. A. U. X. Tan, «Estimating displacement of periodic motion with inertial sensors,» *IEEE Sens. J,* vol. 11, nº 8, p. 1385–1388, 2009.

[39] S. H. F. R. y. B. P. G. J. C. a. B. D. M. Jakopec, «Preliminary results of an early clinical experience with the acrobot™ system for total knee replacement surgery,» *Med. Image Comput. Comput.-Assist. Interv,* p. 256–263, 2022.

[40] E. C. A. P. S. Avgousti, «Medical telerobotic systems: current status and future trends,» *BioMed Eng. Online,* vol. 15, nº 96, 2016.

[41] N. A. a. R. Patel, «Teleoperated master-slave needle insertion,» *Int. J. Med. Robot. Comput. Assist. Surg.,* vol. 5, nº 2, p. 398–405, 2009.

[42] N. H. R. H. M. H. a. T. D. F. Ohara, «Needle guiding robot with five-bar linkage for MR guided thermotherapy of liver tumor,» *J. Life Support Eng,* vol. 16, p. 235–236, 2004.

[43] N. R. a. M. S. L. Van Den Bedem, «Design of a slave robot for laparoscopic and thoracoscopic surgery,» *Proc. 20th Int. Conf. Med. Innov. Technol.,* pp. 1-4, 2008.

[44] J. Z. D. L. B. L. a. F. Z. C. Meng, «A remote-controlled vascular interventional robot: system structure and image guidance,» *Int. J. Med. Robot. Comput. Assist. Surg.,* vol. 9, nº 2, p. 230–239, 2013.

[45] D. H. Mendoza, «Intuición artificial aplicada a un procedimiento de anestesia raquídea teleoperado,» *Coordinación General de Estudios de Posgrado, Universidad Nacional Autónoma de México,* 2019.

[46] S. S. S. a. R. K. Shinde, «Minimally invasive gastrointestinal surgery: A review,» *Cureus,* vol. 15, nº 11, 2023.

[47] Y. Yanpei, «Development of an intuitive foot-machine interface for robotic surgery,» *arXiv,* 2019.

[48] J. K. a. K. Kuchenbecker, «Surgeons and non-surgeons prefer haptic feedback of instrument vibrations during robotic surgery,» *Surg. Endosc,* vol. 29, nº 10, p. 2970–2983, 2015.

[49] B. P. A. C. a. J. S. S. Nakdhamabhorn, «Sensorless Based Haptic Feedback Integration In Robot-assisted Pedicle Screw Insertion For Lumbar Spine Surgery: A preliminary cadaveric study,» *Comput. Struct. Biotechnol. J.*

[50] S. Z. Z. W. B. Y. a. K. X. B. Zhao, «CombX: Design and experimental characterizations of a haptic device for surgical teleoperation,» *Int. J. Med. Robotics Comput. Assist. Surg,* vol. 16, nº 1, 2020.

[51] M. P. C. M. S. S.-H. J. A. L. C. P. C. H. J. P. I. I. I. a. M. H. A.-N. F. H. Alruwaili, «Design and experimental evaluation of a leader-follower robot-assisted system for femur fracture surgery,» *Int. J. Control Autom. Syst.,* vol. 22, nº 9, p. 2833–2846, 2024.

[52] Z. K. N. D. R. a. J. D. B. S. Machaca, «Towards a ROS-based Modular Multi-Modality Haptic Feedback System for Robotic Minimally Invasive Surgery Training Assessments,» *2022 International Symposium on Medical Robotics,* pp. 1-7, 2022.

[53] E. D. M. a. G. F. N. Enayati, «Haptics in Robot-Assisted Surgery: Challenges and Benefits,» *IEEE Reviews in Biomedical Engineering,* vol. 9, pp. 49-65, 2016.

[54] T. H. A. S. E. E. a. M. B. C.F. Mackenzie, «Virtual reality and haptic interfaces for civilian and military open trauma surgery training: A systematic review,» *Injury,* vol. 53, nº 11, p. 3575–3585, 2022.

[55] H. J. M. K. -W. K. a. G. -Z. Y. K. Leibrandt, «Implicit active constraints for a compliant surgical manipulator,» *2014 IEEE International Conference on Robotics and Automation,* pp. 276-283, 2014.

[56] B. P. Y. F. a. Y. A. H. Qiu, «Surgical Instruments Motion Safety Constraint Based on Haptic Virtual Fixture,» *2019 IEEE International Conference on Mechatronics and Automation,* pp. 2267-2272, 2019.

[57] F. R. a. H. J. Chizeck, «Forbidden-region virtual fixtures from streaming point clouds: Remotely touching and protecting a beating heart,» *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 3308-3313, 2012.

[58] W. W. Y. C. Q. Z. Y. W. Y. H. a. S. W. Y. Wang, «A guiding and positioning motion strategy based on a new conical virtual fixture for robot-assisted oral surgery,» *Machines,* vol. 11, nº 1, p. 3, 2023.

[59] J. F. e. al., «Augmented Reality and Human–Robot Collaboration Framework for Percutaneous Nephrolithotomy: System Design, Implementation, and Performance Metrics,» *IEEE Robotics & Automation Magazine,* vol. 31, nº 3, pp. 25-37, 2024.

[60] B. C. S. C. S. B. L. H. a. C. C. S. Liu, «A cable linkage with remote centre of motion,» *Mechanism and Machine Theory,* vol. 105, p. 583–605, 2016.

Anexos

1 Artículo presentado en “8 th International Symposium on Multibody Systems and Mechatronics”

Development of a haptic interface with mixed reality guidance for needle insertion in epidural anesthesia," *Mechanisms and Machine Science*, vol. 195, 2025

2 Códigos Arduino

2.1 Código para ESP32 del lado esclavo (motor a pasos)

```
//incluye librería para control del motor a pasos mediante el driver
//de forma que la ESP solo tiene que enviar pulsos a la salida
#include <AccelStepper.h>
//incluye librería para comunicación mediante UART con otra ESP32
#include <Wire.h>

//se definen pines de salida de pulsos y de UART así como el que tomara la entrada del switch de final de carrera
#define pul  4
#define dir  5
#define final 2
#define MY_ADDRESS 9
#define DEBUG_SERIAL Serial

//inicializa un arreglo para recibir información por UART y variables para movimientod e motr a pasos
char buf[4];
float l3;
int p_l3;

//inicializa la comunicación con el driver del motor a paso
AccelStepper stepper(1, pul,dir);

void setup() {
  // inicia comunicación serial
  DEBUG_SERIAL.begin(115200);
  //inicia variables para movimiento del motor a pasos
  stepper.setMaxSpeed(30000);
  stepper.setSpeed(30000);
  pinMode(final, INPUT);        // pin 7 de entrada
  //empieza a mover el motor a pasos hasta que se presiona el switch de final de carrera
  while(digitalRead(final)==LOW){
    stepper.runSpeed();
  }
  //una vez en esa posición se establece el punto cero
  stepper.setCurrentPosition(0);

  //stepper.runToNewPosition(-100);
  //cabe recalcar que 100 pulsos es igual a 4mm en la configuracion actual
  //se estabblece la aceleración para el motor a pasos
  stepper.setAcceleration(64000);
```

```
  //Para las pruebas de aguja Tuohy mas exactas instalada (mide como 12.6 cm mas que las pinzas)
  stepper.runToNewPosition(-5600);

  //se establece como punto cero la posición actual
  stepper.setCurrentPosition(0);
  //inicia comunicación UART y al recibir un mensaje entra en subrutina
  Wire.begin(MY_ADDRESS);
  Wire.onReceive(receiveEvent);
  delay(5000);

}

// Subrutina de recibir un mensaje
void receiveEvent(int numBytes) {
  //se leen 4 bytes que corresponden a un número flotante. Este se almacena en la variable l3
  if(Wire.available() == 4) {
    buf[0] = Wire.read();
    buf[1] = Wire.read();
    buf[2] = Wire.read();
    buf[3] = Wire.read();
    memcpy (&l3, buf, 4);
    //se multiplica por 10 para obtener la distancia en unidades reales (m)
    l3=l3*10;
    //p_l3 = (l3*200)/0.8;
    //se traduce al número de pulsos requeridos para cubrir la distancia deseada (p_l3)
    p_l3 = (l3*6400)/0.8;
    //se mueve el motor a la posición deseada
    if(p_l3==0){
      stepper.moveTo(0);
    }
    else{
      stepper.moveTo(-p_l3);
    }
    DEBUG_SERIAL.println(l3);
    DEBUG_SERIAL.println(p_l3);
  }
}

//comando para que el motor se mueva constantemente a una posición indicada (esta se define en la subrituna
"receiveEvent")
void loop() {
  stepper.run();
}
```

2.2 Código para ESP32 del lado maestro (control de servomotores y recibimiento de instrucciones por UDP)

```
//incluye librería para comunicaión con los servomotores
#include <Dynamixel2Arduino.h>

//librerías y definiciones para comunicación serial, Wifi, UaRt
#define DXL_SERIAL Serial2
#define DEBUG_SERIAL Serial
#include "WiFi.h"
#include "AsyncUDP.h"
```

```
#include <Wire.h>
#define SLAVE_ADDRESS 9

//variables para comunicaion por UDP con computadora
const char* ssid = "TP-Link_8960";
const char* pass = "53899736";
AsyncUDP udp;
float f, f2, f3,a1=0,a2=0,a3,L1=0,T_1=0,T_2=0;

const int DXL_DIR_PIN = 2; // DYNAMIXEL Shield DIR PIN
const uint8_t DXL_ID_1 = 1;
const uint8_t DXL_ID_2 = 2;
const uint8_t DXL_ID_3 = 3;
const float DXL_PROTOCOL_VERSION = 2.0;

//union para pasar de flotante a 4 bytes y mandar al motor a pasos
union cvt {
  float val;
  unsigned char b[4];
} x;

//comunicación con servomotores
Dynamixel2Arduino dxl(DXL_SERIAL, DXL_DIR_PIN);

//This namespace is required to use Control table item names
using namespace ControlTableItem;

void setup() {

  //para comunicaion i2c con motor a pasos
  Wire.begin();

  // Use UART port of DYNAMIXEL Shield to debug.
  DEBUG_SERIAL.begin(115200);
  while(!DEBUG_SERIAL);

  // Set Port baudrate to 57600bps. This has to match with DYNAMIXEL baudrate.
  dxl.begin(1000000);
  // Set Port Protocol Version. This has to match with DYNAMIXEL protocol version.
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
  // Get DYNAMIXEL information

if(dxl.ping(DXL_ID_1)) {
  //imprime la información de los servomotores a la consola   al inicializarlos
  DEBUG_SERIAL.print("ID : ");
  DEBUG_SERIAL.print(DXL_ID_1);
  DEBUG_SERIAL.print(", Model Number: ");
  DEBUG_SERIAL.println(dxl.getModelNumber(DXL_ID_1));
  DEBUG_SERIAL.println(dxl.getPresentPosition(DXL_ID_1));
  DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_POSITION, DXL_ID_1));

  //parámetros de servomotor
  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID_1, 0);
  dxl.writeControlTableItem(OPERATING_MODE, DXL_ID_1, 3);
  dxl.writeControlTableItem(POSITION_P_GAIN, DXL_ID_1, 25);
  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID_1, 1);
```

```
}


if(dxl.ping(DXL_ID_2)) {
  DEBUG_SERIAL.print("ID : ");
  DEBUG_SERIAL.print(DXL_ID_2);
  DEBUG_SERIAL.print(", Model Number: ");
  DEBUG_SERIAL.println(dxl.getModelNumber(DXL_ID_2));
  DEBUG_SERIAL.println(dxl.getPresentPosition(DXL_ID_2));
  DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_POSITION, DXL_ID_2));

  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID_2, 0);
  dxl.writeControlTableItem(OPERATING_MODE, DXL_ID_2, 3);
  dxl.writeControlTableItem(POSITION_P_GAIN, DXL_ID_2, 25);
  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID_2, 1);
}


if(dxl.ping(DXL_ID_3)) {
  DEBUG_SERIAL.print("ID : ");
  DEBUG_SERIAL.print(DXL_ID_3);
  DEBUG_SERIAL.print(", Model Number: ");
  DEBUG_SERIAL.println(dxl.getModelNumber(DXL_ID_3));
  DEBUG_SERIAL.println(dxl.getPresentPosition(DXL_ID_3));
  DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_POSITION, DXL_ID_3));

  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID_3, 0);
  dxl.writeControlTableItem(OPERATING_MODE, DXL_ID_3, 3);
  dxl.writeControlTableItem(POSITION_P_GAIN, DXL_ID_3, 25);
  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID_3, 1);
}

  delay(1000);

  //*******Setup para comunicacion por UDP***
  WiFi.disconnect(true);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  if (udp.listen(1234)) {
    Serial.print("UDP Listening on IP: ");
    Serial.println(WiFi.localIP());
    //al recibir mensaje de UDP, lo organiza en las variables correspondientes a los datos recibidos
    udp.onPacket([](AsyncUDPPacket packet) {
      //Serial.write(packet.data(), packet.length());
      memcpy (&a1, packet.data(), 4);
      memcpy (&a2, packet.data()+4, 4);
      memcpy (&a3, packet.data()+8, 4);
      memcpy (&f, packet.data()+12, 4);
      memcpy (&f2, packet.data()+16, 4);
      memcpy (&f3, packet.data()+20, 4);
      memcpy (&L1, packet.data()+24, 4);
      memcpy (&T_1, packet.data()+28, 4);
```

```
    memcpy (&T_2, packet.data()+32, 4);

    //-100 es el valor que indica omitir el movimiento del motor a pasos, al recibir un valor distinto a ese
    //en la variable L1, se envía la posición deseada por UART a la ESP32 que controla el motor a paso
    if(L1 != -100){
      x.val = L1;
      Wire.beginTransmission(SLAVE_ADDRESS);
      Wire.write(x.b[0]);
      Wire.write(x.b[1]);
      Wire.write(x.b[2]);
      Wire.write(x.b[3]);
      Wire.endTransmission();
    }

   });
  }

}

void loop() {
  int angulo1;
  int angulo2;
  int angulo3;

  //se establece el rango de los ángulos de los servomotores 2 y 3 entre -45 y 45 grados
  if(-45>T_2){
    T_2=-45;
  }
  else if(T_2>45){
    T_2=45;
  }

  //se mapean los valores en grados a valores para los servomotores
  angulo1 = map(T_2, -45, 45, 114541, 74762);
  DEBUG_SERIAL.printf("%f\n", T_2);
  DEBUG_SERIAL.printf("Angulo junta 2 = %d\n",angulo1);

  //angulo2 = map(T_2, -25, 25, 21453, -20149);
  angulo2 = map(T_2, -45, 45, 20000, -20149);
  DEBUG_SERIAL.printf("Angulo junta 3 = %d\n",angulo2);

  angulo3 = map(T_1, -25, 25, -28998, 7525);
  DEBUG_SERIAL.printf("Angulo junta 1 = %d\n",angulo3);

  //-100 es el valor de la variable angular que indica omitir el movimiento de los servomotores
  //en caso contraio, se secriben los valores deseados a los servomotores
  if(T_1 != -100){
    dxl.writeControlTableItem(GOAL_POSITION, DXL_ID_2, angulo2);
    dxl.writeControlTableItem(GOAL_POSITION, DXL_ID_1, angulo1);


  //angulo 2 es el de enmedio, angulos 2 y 3 se mueven sincronizadamente


  //DEBUG_SERIAL.printf("ADC analog value 2 = %d\n",analogValue2);
  //angulo 3 es el motor que esta encima de los otros
```

```
    dxl.writeControlTableItem(GOAL_POSITION, DXL_ID_3, angulo3);
  }

  delay(100);
}
```

3 Códigos de Unity

3.1 Código para enviar instrucciones mediante UDP

```
using UnityEngine;
using System.Collections;

using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Linq;

public class UDPSend : MonoBehaviour
{
    private static int localPort;

    public HapticPlugin hapticAct;
    public GameObject actuador;
    public GameObject rcm;
    public GameObject enviarUDP;
    public GameObject insertP;

    // prefs
    private string IP;  // define in init
    public int port;  // define in init

    // "connection" things
    IPEndPoint remoteEndPoint;
    UdpClient client;

    // gui
    string strMessage="";


    // start from unity3d
    public void Start()
    {
        init();
    }


    // init
    public void init()
    {
        // Define the endpoint from which the messages are sent.
```

```
        print("UDPSend.init()");

        // define
        IP="192.168.0.100";
        port=1234;

        // ----------------------------
        // Send
        // ----------------------------
        remoteEndPoint = new IPEndPoint(IPAddress.Parse(IP), port);
        client = new UdpClient();

        // status
        print("Sending to "+IP+" : "+port);
        print("Testing: nc -lu "+IP+" : "+port);
    }


    // sendData
    private void sendString(string message)
    {
            FiltroFIR filtro = enviarUDP.GetComponent<FiltroFIR>();
            float num = 1.03F;
            float X_local, Y_local, Z_local, L_1, theta1 = 0, theta2 = 0, Xfiltrada, Yfiltrada, Zfiltrada;
            float comp;

            hapticAct.UpdateButtonStatus();

        try
        {
                        //calculo de vector y distancias de el colisionador respecto a RCM
                        //Cabe recalcar que se divide entre 10 porque normalmente se puede mover 20 cm
alrededor del RCM, de esta forma lo ponemos a 2
                        (Xfiltrada, Yfiltrada, Zfiltrada) = filtro.updateFilter(actuador.transform.position[0],
actuador.transform.position[1], actuador.transform.position[2]);

                        X_local = (actuador.transform.position[0]-rcm.transform.position[0]);

                        Y_local = (actuador.transform.position[1]-rcm.transform.position[1]);


                        Z_local = (actuador.transform.position[2]-rcm.transform.position[2]);

                        L_1 = MathF.Sqrt((X_local*X_local) + (Y_local*Y_local) + (Z_local*Z_local));

                        //para la alineación automática, constantemente manda la misma posición fijada
            //en caso de no utilizarla, comentar las siguientes lineas:
                        X_local = (insertP.transform.position[0]-rcm.transform.position[0]);
                        Y_local = (insertP.transform.position[1]-rcm.transform.position[1]);
                        Z_local = (insertP.transform.position[2]-rcm.transform.position[2]);

            //se añaden los ángulos del gimball del dispositivo y las posiciones locales a los datos
            //a enviar a la ESP32 por medio de UDP
                        byte[] data = BitConverter.GetBytes(hapticAct.GimbalAngles[0]);
                        byte[] toSend =
data.Concat(BitConverter.GetBytes(hapticAct.GimbalAngles[1])).ToArray();
```

```
                    toSend = toSend.Concat(BitConverter.GetBytes(hapticAct.GimbalAngles[2])).ToArray();
                    toSend = toSend.Concat(BitConverter.GetBytes(X_local)).ToArray();
                    toSend = toSend.Concat(BitConverter.GetBytes(Y_local)).ToArray();
                    toSend = toSend.Concat(BitConverter.GetBytes(Z_local)).ToArray();


                    //de igual forma se calculan los ángulos theta 1 y 2 y se mandan
                    //se invierte el signo porque la camara esta viendo hacia la base del robot

                theta1 = (-MathF.Atan2(X_local,Y_local));
                        theta1 = theta1 *Mathf.Rad2Deg;
                    theta2 = (-(MathF.Atan2(Z_local,Y_local)));
                        theta2 = theta2 *Mathf.Rad2Deg;
                    theta2 = theta2-25.0f;

                    if(Y_local < 0){
                       theta1 = (MathF.Atan2(X_local,-Y_local)*Mathf.Rad2Deg);
                        theta2 = ((MathF.Atan2(Z_local,-Y_local)*Mathf.Rad2Deg));
                        theta2 = theta2-25.0f;
                    }

        //modificación en la distancia que se mueve L_1 para compensar imperfectos mecánicos del robot
                    comp = (theta1-(-25f))*(.1f-0f)/(25f-(-25f))+0f;
                    if(Y_local < 0){
          L_1= L_1-comp;
                    }

                    //esto es para si el objeto a controlar esta encima del RCM, se mueva hacia arriba el
efector final, en este caso se comenta
                    //ya que al desacoplar el efector final del movimiento normal del robot no se ocupa
                    if(Y_local > 0){
                        L_1 = -L_1;
                    }

        //establece límites superiores e inferiores a lo que se puede mover el efector final del robot para evitar
daños
                    if(L_1<-0.02f){
                        L_1=-0.02f;
                    }
                    if(L_1>1.7f){
                        L_1=1.7f;
                    }

                    //esta seccion es para separar los movimientos del actuador lineal de los de los
servomotores, boton 1 para los servos y boton 2
                    //para subir y bajar, sin ambargo ahora se decide que boton 1 tambien sube y baja y solo
el boton 2 es para quitar los servos.
                    //por lo que se comenta el else
                    if(hapticAct.boton2 == 1){
                        theta1 = -100f;
                        theta2 = -100f;
                    }

                    //se envia la distancia L1, angulos theta1 y theta2. Es necesario agregarle al codigo del
micro para que reciba mas bytes
                    toSend = toSend.Concat(BitConverter.GetBytes(L_1)).ToArray();
```

```
                    toSend = toSend.Concat(BitConverter.GetBytes(theta1)).ToArray();
                    toSend = toSend.Concat(BitConverter.GetBytes(theta2)).ToArray();

            //envía los datos por UDP
            Debug.Log(client.Send(toSend, toSend.Length, remoteEndPoint));
        }
        catch (Exception err)
        {
            print(err.ToString());
        }
    }


    private void Update()
    {
    //cada que se actualize el juego (una vez por frame), se envían los datos de la posición mediante UDP
    //recordar que este script soloe sta activo cuando el botón 1 del dispositivo esta presionado
            Matrix4x4 matrizAct;
            Matrix4x4 matrizF;
            Vector3 scale = new Vector3(100, 100, 100);
            double[] vectorF = {1,1,1};

            sendString(Convert.ToString(hapticAct.CurrentPosition));
            hapticAct.UpdateDeviceInformation();
            hapticAct.GetDeviceTransformationRaw();

            matrizAct = hapticAct.retTransform();
        matrizF = matrizAct * hapticAct.matrizH.inverse;
    }

}
```

3.2 Código para habilitar la guía visual de alineación de objetos virtuales

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class hideGuide : MonoBehaviour
{

    public GameObject Sphere;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
```

```
    // Cuando el objeto al que este script esta asignado colisiona con la esfera, deja de ser visible
    private void OnTriggerEnter(Collider other){
            if(other.gameObject == Sphere){
                    gameObject.GetComponent<MeshRenderer>().enabled = false;
            }
    }
    // Al dejar de tocar la esfera, el objeto vuelve a ser visible
    private void OnTriggerExit(Collider other){
            if (other.gameObject == Sphere){
           gameObject.GetComponent<MeshRenderer>().enabled = true;
            }
    }
}
```

3.3 Código para habilitar fuerzas de simulación en el dispositivo háptico

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NeedleForceSimultaion : MonoBehaviour
{
    //Objetos del entorno de juego
    public HapticPlugin hapticAct;
    public GameObject actuador;
    public GameObject punta;
    public GameObject cilindro;
    public GameObject image;
    public Plane gameObject ;
    //Longitud de profundidad desde la piel al espacio epidural y picos de fuerza dados por el usuario desde la
interfaz
    public float longitudMM;
    public float picoA;
    public float picoC;

    //variables del modelo matemático que describirá las fuerzas de simulación
    float inicioA;
    float finA;
    float inicioC;
    float finC;
    float theta1a = 2f;
    float theta2a = 1f;
    float theta3a = 0.2f;
    float theta4a;
    float theta5a = 0.5f;
    float theta6a;

    float theta1c = 3f;
    float theta2c = 1f;
    float theta3c = 0.5f;
    float theta4c;
    float theta5c = 0.5f;
    float theta6c;
```

```
float FpA = 1.42893f;
float FpC = 2.4308f;
float F0 = 0.751333f;
float fr = 0.054588333f;

float inicioX = 0;
float inicioY = 0;
float inicioZ = 0;

// Start is called before the first frame update
void Start()
{
   //inicialización de valores para el modelo matemático con los parámetros dados por el usuario
   inicioA = picoA - (0.04878f*longitudMM);
        finA = inicioA + (0.07317f*longitudMM);
        inicioC = picoC - (0.07317f*longitudMM);
        finC = picoC + (0.02439f*longitudMM);

   theta4a = 1-theta3a;
        theta6a = 1-theta5a;

   theta4c = 1-theta3c;
   theta6c = 1-theta5c;
}

public void SetStart()
{
   //registra la posición espacial inicial del actuador al iniciar el script (presionar el botón 2)
   inicioX = actuador.transform.position[0];
   inicioY = actuador.transform.position[1];
   inicioZ = actuador.transform.position[2];
}

public void EndForce()
{
   //al terminar el script (soltar el botón 2 y presionar el obtón 1), se desactivan las fuerzas de simulación
   double[] vectorF = {0,0,0};
   HapticPlugin.setGravityForce("Default Device", vectorF);
}

private float calcForce(float dist){

   //función para calcular la fuerza de simulación según la distancia desde el punto de inicio
   float xsa, xsc, Er_a, Er_c, El_a, El_c, Fa, Fb, Fc, Fcut, Ffriction, F;
   Fcut=0f;
   Ffriction=0f;
   F=0f;
   Fa=0f;
   Fc=0f;

   if(dist < 0){
      return 0;
   }

   //Ecuaciones segmento A
```

```
if((dist >= inicioA) && (dist < finA)){
   xsa = dist-picoA;
   if(xsa <= 0){
   El_a = -xsa/theta1a;
   Fa = (theta3a*(1-El_a) + theta4a*(1-El_a)*(1-El_a)) * FpA;
   }
   if(xsa > 0){
   Er_a = xsa/theta2a;
   Fa = (theta5a*(1-Er_a) + theta6a*(1-Er_a)*(1-Er_a)) * FpA;
   }
}
else{
   Fa = 0;
   }

//Ecuaciones segmento B

if(dist < picoA){
   Fcut = F0*(dist/picoA);
}
if(dist >= picoA){
   Fcut = F0;
}
if(dist < picoA){
   Ffriction = 0;
}
if((dist >= picoA) && (dist < picoC)){
   Ffriction = fr*(dist-picoA);
}
if(dist >= picoC){
   Ffriction = fr*(picoC-picoA);
}

//Ecuaciones segmento C

if((dist >= inicioC) && (dist < finC)){
   xsc = dist-picoC;
   if(xsc <= 0){
   El_c = -xsc/theta1c;
   Fc = (theta3c*(1-El_c) + theta4c*(1-El_c)*(1-El_c)) * FpC;
   }
   if(xsc > 0){
   Er_c = xsc/theta2c;
   Fc = (theta5c*(1-Er_c) + theta6c*(1-Er_c)*(1-Er_c)) * FpC;
   }
}
else{
   Fc = 0;
   }

//Una vez calculadas las fuerzas de cada segmento se suman
//Cabe recalcar que las fuerzas A y C corresponden a picos de fuerza mientras que la B es una
constantemente creciente
//Debido a las limitaciones de fuerza que puede otorgar el dispositivo, se decide hacer mas débil a la
fuerza B
//para que no pasen desapercibidas las fuerzas de los picos
```

```
        Fb = Fcut + Ffriction;
        F = Fa + Fb*(0.5f) + Fc;
        return F;

    }

    // Update is called once per frame
    void Update()
    {
        //constantemente se calcula la fuerza y se aplica en el dispositivo háptico
        float dist;
        int i,j;
        float force = 0f;
        //inicialización de vectores para realizar operaciones y almacenar resultados
        double[] vectorF = {0,0,0};
        double[] vectorX = {-1,-1,-1};
        double[] vectorP = {10,10,10};
        double[] res = {0,0,0};
        float[,] matrizR={{0,0,0},{0,0,0},{0,0,0}};

        //vector que guarda la posición actual del estilete
        vectorP[0] = punta.transform.position[0];
        vectorP[1] = punta.transform.position[1];
        vectorP[2] = punta.transform.position[2];

        gameObject.SetNormalAndPosition(this.transform.up, this.transform.position);
        //calculo de distancia desde punto inicial a la actual
                dist = (float)Math.Sqrt(Math.Pow(inicioX-actuador.transform.position[0], 2)+Math.Pow(inicioY-
actuador.transform.position[1], 2)+Math.Pow(inicioZ-actuador.transform.position[2], 2));

        //se calcula la fuerza en base a la distancia recorrida
        dist = dist * 100;
        force = calcForce(dist);
                        Debug.Log(force);

        //una vez calculada la fuerza se multiplica por los componentes de la matriz de transformación para verse
aplciada
        //sobre el eje Z del estilete, en este caso es en dirección contraria a donde apunta el usuario
        res[0] = hapticAct.retTransform()[0,2]100(force);
        res[1] = hapticAct.retTransform()[1,2]100(force);
        res[2] = hapticAct.retTransform()[2,2]100(force);

        //aplica las fuerzas al dispositivo
        HapticPlugin.setGravityForce("Default Device", res);
    }
}
```

3.4 Código para habilitar fuerza de atracción según la orientación de un objeto virtual

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FuerzaAtraccion : MonoBehaviour
{
    public HapticPlugin hapticAct;
```

```
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
            double[] res = {0,0,0};
            //De la matriz de rotación del objeto al que este script está asignado, obtiene los valores del vector Z
            //y los multiplica por un valor para ajustar el nivel y dirección de las fuerzas
            res[0] = gameObject.transform.localToWorldMatrix[0,2]*0.2;
            res[1] = gameObject.transform.localToWorldMatrix[1,2]*-0.2;
            res[2] = gameObject.transform.localToWorldMatrix[2,2]*0.2;

            //Aplica las fuerzas al dispositivo háptico
            HapticPlugin.setConstantForceValues("Default Device",res,0.3);
    }
}
```

3.5 Código para recibir valores mediante UDP (posición de objetos virtuales)

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using UnityEngine;

public class RemoteInputListener : MonoBehaviour
{
    private static RemoteInputListener SharedInstance;
    private volatile bool runTask = true;
    public int listenPort = 8181;
    public GameObject cubo;

    Quaternion rotationQuat;
    float newX;
    float newY;
    float newZ;

    public static RemoteInputListener GetSharedInstance()
    {
        return SharedInstance;
    }

    private void Awake()
    {
        if(SharedInstance == null)
        {
```

```
        SharedInstance = this;
        DontDestroyOnLoad(gameObject);
    }else if (SharedInstance != this)
    {
        Destroy(gameObject);
    }
}

//Para sacar quaterniones de matriz de rotación
public static Quaternion QuaternionFromMatrix(Matrix4x4 m)
{
    float trace = m.m00 + m.m11 + m.m22;

    if (trace > 0)
    {
        float s = Mathf.Sqrt(trace + 1.0f) * 2;
        return new Quaternion(
            (m.m21 - m.m12) / s,
            (m.m02 - m.m20) / s,
            (m.m10 - m.m01) / s,
            0.25f * s
        );
    }
    else if ((m.m00 >= m.m11) && (m.m00 >= m.m22))
    {
        float s = Mathf.Sqrt(1.0f + m.m00 - m.m11 - m.m22) * 2;
        return new Quaternion(
            0.25f * s,
            (m.m01 + m.m10) / s,
            (m.m02 + m.m20) / s,
            (m.m21 - m.m12) / s
        );
    }
    else if (m.m11 > m.m22)
    {
        float s = Mathf.Sqrt(1.0f + m.m11 - m.m00 - m.m22) * 2;
        return new Quaternion(
            (m.m01 + m.m10) / s,
            0.25f * s,
            (m.m12 + m.m21) / s,
            (m.m02 - m.m20) / s
        );
    }
    else
    {
        float s = Mathf.Sqrt(1.0f + m.m22 - m.m00 - m.m11) * 2;
        return new Quaternion(
            (m.m02 + m.m20) / s,
            (m.m12 + m.m21) / s,
            0.25f * s,
            (m.m10 - m.m01) / s
        );
    }
}

// Start is called before the first frame update
```

```
    void Start()
    {
        //comms
        Thread t1 = new Thread(InitInputServer);
        t1.Start();
    }

    // Update is called once per frame
    void Update()
    {
        //cada frame actualiza la posición del objeto referenciado y orientación mediante ángulos de Euler
        Vector3 rotaciones = rotationQuat.eulerAngles;
        rotaciones[1] = rotaciones[1]*-1;
        rotaciones[0] = rotaciones[0]*-1;
        cubo.transform.rotation = Quaternion.Euler(rotaciones);

        cubo.transform.position = new Vector3(-newX*10f, newZ*10f, newY*10f);

    }

    private void InitInputServer()
    {
        //inicialización de vectores y matrices para almacenar datos y realizar operaciones
        byte[] receivedData;
        Matrix4x4 rotationM = new Matrix4x4();
        Matrix4x4 xCompens = new Matrix4x4();
        Matrix4x4 Ultrasonido = new Matrix4x4();
        Vector3 posUltrason = new Vector3(0f, 0.10f, 0.0f);
        Vector3 newPos = new Vector3(0f,0f,0f);
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any, listenPort);
        IPEndPoint ipepReceive = new IPEndPoint(IPAddress.Any, listenPort);

        using (var udpClient = new UdpClient(ipep))
        {
            while (runTask)
            {
                receivedData = udpClient.Receive(ref ipepReceive);
                //Construye una matriz homogúnea de los datos recibidos (posicitn y rotación) de un marcador
respecto al marcador global
                rotationM.SetColumn(0, new Vector4((float)BitConverter.ToDouble(receivedData, 0),
(float)BitConverter.ToDouble(receivedData, 24), (float)BitConverter.ToDouble(receivedData, 48),0f));
                rotationM.SetColumn(1, new Vector4((float)BitConverter.ToDouble(receivedData, 8),
(float)BitConverter.ToDouble(receivedData, 32), (float)BitConverter.ToDouble(receivedData, 56),0f));
                rotationM.SetColumn(2, new Vector4((float)BitConverter.ToDouble(receivedData, 16),
(float)BitConverter.ToDouble(receivedData, 40), (float)BitConverter.ToDouble(receivedData, 64),0f));
                rotationM.SetColumn(3, new Vector4(0f,0f,0f,1f));

                //llama a función para obtener cuaterniones de la matriz de rotación
                rotationQuat = QuaternionFromMatrix(rotationM);

                //datos de posición
                newX = (float)BitConverter.ToDouble(receivedData, 72);
                newY = (float)BitConverter.ToDouble(receivedData, 80);
                newZ = (float)BitConverter.ToDouble(receivedData, 88);
            }
        }
```

```
    }

    private void OnDestroy()
    {
        runTask = false;
    }
}
```

4 Códigos Python

4.1 Código para detectar marcadores y calcular posición y rotación entre ellos

```
import cv2 as cv
import cv2.aruco as aruco
import numpy as np
import json
import socket
import struct

print(cv._version_)

#datos de comunicación WiFi
#UDP_IP = "192.168.1.72"
UDP_IP = "192.168.0.101"
UDP_PORT = 8181
MESSAGE = "Hello, World!"

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP

# Configuración global de marcadores
MARKER_IDS = [0, 1, 2, 3]  # IDs de los marcadores a detectar
MARKER_SIZE_METERS = 0.06
VIDEO_WIDTH, VIDEO_HEIGHT = 1280, 720
config_file = "scale_factor_config.json"

# Configuración de diccionario y parámetros de ArUco
aruco_dict = aruco.getPredefinedDictionary(aruco.DICT_4X4_50)
parameters = aruco.DetectorParameters()
detector = aruco.ArucoDetector(aruco_dict, parameters)

# Calibración de la cámara (inserta tus propios parámetros)
camera_matrix = np.array([[540, 0, VIDEO_WIDTH // 2],
                          [0, 540, VIDEO_HEIGHT // 2],
                          [0, 0, 1]], dtype=np.float32)
dist_coeffs = np.zeros(([5,1]))  # Asume que la cámara no tiene distorsión

# Cargar scale_factor desde archivo (si existe)
try:
    with open(config_file, "r") as file:
        scale_factor = json.load(file).get("scale_factor", None)
except FileNotFoundError:
    scale_factor = None

# Inicializar captura de video
cap = cv.VideoCapture(0, cv.CAP_DSHOW)  # 0 indica cámara de la computadora y 1 cámara externa
(WebCam)
```

```
cap.set(cv.CAP_PROP_FRAME_WIDTH, VIDEO_WIDTH)
cap.set(cv.CAP_PROP_FRAME_HEIGHT, VIDEO_HEIGHT)

#constantemente busca marcadores en la imagen
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Detectar marcadores ArUco en el cuadro
    corners, ids, _ = detector.detectMarkers(frame)

    if ids is not None and len(ids) > 0:
        # Dibujar los marcadores detectados
        aruco.drawDetectedMarkers(frame, corners, ids)

        # Estimar la pose de los marcadores
        rvecs, tvecs, _ = aruco.estimatePoseSingleMarkers(corners, MARKER_SIZE_METERS, camera_matrix,
dist_coeffs)

        marker_positions = {}
        marker_rotations = {}
        rotationMatrixs = {}

        #para cada marcador detectado
        for i, id in enumerate(ids.flatten()):
            if id in MARKER_IDS:
                if i < len(rvecs) and i < len(tvecs):
                    rvec, tvec = rvecs[i][0], tvecs[i][0]  # Vector de rotación y traslación
                    marker_positions[id] = tvec  # Guardar la posición del marcador
                    marker_rotations[id] = rvec  # Guardar la rotacion
                    rvec2 = (-rvec[0],-rvec[2],rvec[1])
                    #guardamatrices de rotacion
                    rotationMatrixs[id], _ = cv.Rodrigues(rvec2)

                    # Dibujar el sistema de ejes en el marcador
                    cv.drawFrameAxes(frame, camera_matrix, dist_coeffs, rvec, tvec, 0.05)

                    # Mostrar las coordenadas en metros (x, y, z)
                    x, y, z = tvec
                    cv.putText(frame, f"ID {id}: x={x:.2f}m, y={y:.2f}m, z={z/0.6:.2f}m",
                            (10, 30 + 30 * id), cv.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)


                        # Calcular y mostrar distancias entre ID0 y los demás marcadores
        #se calcularan tambie las rotaciones, en este caso en vez de utilizar la funcion para cada marcador
        #se realizara solo entre el marcador Id 1 y el 0 ya que es el correspondiente al ultrasonido
        #y al origen
        if 0 in marker_positions:
            origin = marker_positions[0]  # Posición del marcador ID0
            #busca el marcador 1
            if 1 in marker_positions:
                target = marker_positions[1]
                # Calcular distancia euclidiana
                #distance = np.linalg.norm(origin - target)
                distance= ((target[0]-origin[0])*2+ (target[1]-origin[1])2+(target[2]-origin[2])2)*0.5
```

```
            #print(distance)
            distanceX = origin[0]-target[0]
            distanceY = origin[1]-target[1]
            distanceZ = ((origin[2]-target[2])/0.6)
            print(distanceX, distanceY, distanceZ)
            cv.putText(frame, f"Dist(ID0-ID{id}): {distance:.2f}m",
                  (10, 200 + 30 * id), cv.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)
            #calcula rotaciones de cada marcador respecto al marcador 0
            M = np.matmul(rotationMatrixs[0].T,rotationMatrixs[1])
            #M2 = M.tobytes
            M2 = M.tobytes() + struct.pack('d', distanceX)+struct.pack('d', distanceY)+struct.pack('d',
distanceZ)
            #M2=M.astype(float)
            #sock.sendto(M.tobytes('C'), (UDP_IP, UDP_PORT))
            sock.sendto(M2, (UDP_IP, UDP_PORT))
      # Mostrar el video con la detección
   cv.imshow('Aruco Marker Detection', frame)

   # Salir del programa si se presiona la tecla 'q'
   if cv.waitKey(1) & 0xFF == ord('q'):
      break

# Liberar recursos
cap.release()
cv.destroyAllWindows()
```