Plataforma de Simulación de Trayectorias Quirúrgicas en Neurocirugía Utilizando Robots Colaborativos.

Leonardo Alberto Paz Paz
Jan Carlos Alvira Meneses

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, diciembre de 2023

Plataforma de Simulación de Trayectorias Quirúrgicas en Neurocirugía Utilizando Robots Colaborativos.

Leonardo Alberto Paz Paz
Jan Carlos Alvira Meneses

Trabajo de grado para optar por el título de
Ingeniero en Automática Industrial

Director: PhD. Oscar Andrés Vivas Albán

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, diciembre de 2023

Agradecimientos

Leonardo Alberto Paz Paz

En primer lugar, estoy agradecido con Dios por esta vida que se me ha permitido tener. Agradezco a mis padres por su paciencia y apoyo incondicional, ellos fueron mi motor para salir adelante en este proceso. A mi hermana y sobrino y a mi familia en general por su amabilidad.

También quiero agradecer a mi compañero de tesis Jan Carlos por su don de gente y optimismo frente a los momentos más difíciles a lo largo de la carrera. Fuiste un soporte esencial durante todo este tiempo.

Quiero agradecer de forma especial a mi director de tesis, el PhD. Óscar Andrés Vivas Albán, por darnos la oportunidad de trabajar bajo su orientación. Sus palabras de aliento y consejos fueron fundamentales para salir adelante con este proyecto, pero sobre todo su paciencia y confianza fue una gran motivación. También agradezco al PhD. José María Sabater Navarro por su calidez al recibirnos y especialmente por sus ideas que ayudaron a darle forma al proyecto. En este sentido, el grupo nBio de la Universidad Miguel Hernández de Elche, España, tiene mi agradecimiento por su contribución desinteresada cada vez que tuvimos dificultades, y agradezco especialmente a Juan David Romero quien nos hizo sentir como en casa.

Agradezco a mis compañeros Juan Sebastián Montenegro Bravo y Juan David Ruiz Flórez, por sentar las bases de este trabajo con el desarrollo de la plataforma y por su ayuda en un momento crucial.

Finalmente, agradezco a la Universidad del Cauca, a sus profesores que son ejemplo de calidad humana y profesional, y a mis compañeros de estudio, todos ustedes fueron muy importantes para mi crecimiento personal.

Jan Carlos Alvira Meneses

Estoy profundamente agradecido con Dios por darme una segunda oportunidad y poder culminar una nueva etapa en mi vida. A mis padres que son la razón de mi existencia, me inspiraron a pensar diferente, hacer las cosas a mi manera, haciéndome entender que a medida que el mundo cambia nosotros también debemos hacerlo, entendiendo que donde termina un camino, comienza nuestro propio viaje. Sin ellos jamás lo hubiera logrado. Además, deseo expresar mis agradecimientos a mis hermanos que siempre estuvieron ahí para apoyarme.

También quiero extender mis más sinceros a agradecimientos a mi compañero y amigo Leonardo Alberto Paz Paz por sus arduas horas de trabajo, dedicación, esfuerzo, creatividad para que este trabajo tuviera un norte. Su profesionalismo, entusiasmo y entereza por la investigación han sido fundamentales para superar los desafíos y lograr los objetivos propuestos. Además, debo expresar mis magnos agradecimientos a Don Arquímedes, y a la señora Cruz por el apoyo inconmensurable y fundamental que nos brindaron en este proceso, Su bondad y empatía son verdaderamente inspiradoras.

Debo agradecer de manera especial a mi director de tesis, el PhD. Oscar Andrés Vivas Alban por darme la oportunidad de incursionar en el maravilloso mundo de la investigación, además de guiarnos cuando algunas veces estábamos perdidos. Sus valiosas y oportunas sugerencias fueron fundamentales en la realización de todo el proyecto. También quiero reconocer y agradecer a el PhD. José María Sabater Navarro por permitirnos hacer nuestra estancia de investigación en los laboratorios de la Universidad Miguel Hernández, además por sus asesorías, conocimientos rigurosos y precisos que enriquecieron nuestro trabajo. Mi agradecimiento se extiende a los miembros del grupo nBio por acogernos y brindarnos un gran apoyo tanto como técnico y académico. También quiero agradecer a Juan David Romero por toda su invaluable ayuda que nos brindó en nuestra estancia.

Por último y no menos importante debo agradecer a la Universidad del Cauca y a todo su talento humano quien nos instruyó en todo este camino, nos ayudó a crecer tanto personal, profesional y académicamente. A los compañeros que estuvieron ahí en toda esta lucha. A Juan David Ruiz y Juan Sebastián Montenegro por cimentar las bases de este proyecto, estamos seguros de que serán unos grandes profesionales.

Ahora es el momento de volver a donde pertenecemos a los brazos de los luchadores y campeones, para cargar hacia adelante juntos, con un nuevo poder y responsabilidad a nuestro alcance, obligándonos a siempre ser mejores aquí y más allá. El tiempo ha llegado para terminar lo que empezamos.

Resumen

En las últimas décadas se han presentado avances significativos en el área de la robótica médica. Sin embargo, el difícil acceso a hardware, software y entrenamiento relacionado a estos equipos entorpece su alcance e implementación. Por esta razón, el software de simulación se está convirtiendo en una alternativa importante para conducir nuevas investigaciones. Una revisión del estado del arte fue llevada a cabo para arrojar luz en este sentido.

El presente trabajo de grado es una proyección de un trabajo anterior, cuyo objetivo era desarrollar una plataforma en Unity, basada en ROS, capaz de manipular un robot UR3e bajo tres modos de control. Con el fin de encaminar la plataforma hacia un enfoque médico-quirúrgico como dispositivo de navegación, fue necesario modificar el trabajo original en uno de sus modos de control. De esta manera, es posible crear trayectorias con gran presición al interior de objetos 3D que representan partes de la anatomía humana. Para este trabajo se utilizó un *phantom* de cráneo y una herramienta similar a un endoscopio para llevar a cabo las pruebas de laboratorio.

Los cambios hechos en la plataforma y los experimentos conducidos serán descritos detalladamente en las subsecuentes páginas. Por otro lado, los resultados confirman el cumplimiento de los objetivos propuestos y las conclusiones alientan a continuar explorando las posibilidades de la plataforma bajo este enfoque.

Palabras clave: Neurocirugía, neuronavegación, simulación robótica, trayectorias quirúrgicas, registro, Unity, URSim, ROS, UR3e.

Abstract

In recent decades, significant advances have been made in the area of medical robotics. However, difficult access to hardware, software and training related to this equipment hinders their reach and implementation. For this reason, simulation software is becoming an important alternative to conduct new research. A review of the state of the art was carried out to shed light in this regard.

This bachelor´s degree is a projection of a previous work, whose objetive was to develop a platform in Unity, ROS-based, capable of manipulating a UR3e robot under three control modes. In order to direct the platform towards a medical-surgical approach as a navigation device, it was necessary to modify the original work in one of its control modes. In this way, it is posible to créate highly precise trajectories inside 3D objects that represent parts of the human anatomy. For this work, a skull phantom and a tool similar to an endoscope were used to carry out the laboratory tests.

Changes made to the platform and experiments conducted will be described in detail on subsequent pages. On the other hand, results confirm fulfillment of the propose objectives and the conclusions encourage to continue exploring the possibilities of the platform under this approach.

Keywords: Neurosurgery, neuronavigation, robotic simulation, surgical trajectories, registration, Unity, URSim, ROS, UR3e.

Índice general

Índice de figuras

Índice de Tablas

1. Introducción

En la actualidad, la cirugía robótica es un área interdisciplinaria que combina elementos tanto de la medicina como la ingeniería. Su creciente estudio, desarrollo y aplicación muestran la gran acogida de esta tecnología. La complejidad cada vez mayor de las cirugías, la precisión y exactitud de los instrumentos y la disminución de los tiempos operatorios, hacen que cada vez más hospitales consideren adquirir robots quirúrgicos y plataformas de entrenamiento [1]. La cirugía laparoscópica, por ejemplo, ha presentado grandes avances en este campo debido a que actualmente el esquema de control de los robots quirúrgicos es la teleoperación (a diferencia de la cirugía ortopédica o la neurocirugía, donde la autonomía supervisada ha sido aplicada exitosamente). Gracias a la implementación de estos sistemas robóticos en el quirófano, los cirujanos pueden realizar intervenciones más cortas, con un volumen de cirugías mucho mayor [2]. Actualmente, los métodos de entrenamiento en cirugía robótica incluyen modelos de animales [3], modelos de cadáveres [4], modelos de mesa y modelos robóticos [5]. Las tendencias indican que los modelos futuros estarán enfocados hacia plataformas tecnológicas y simuladores virtuales [6], [7], [8].

El acceso a las plataformas de entrenamiento de cirugía robótica virtuales son una gran alternativa de formación que ha demostrado ser más eficiente que los métodos de entrenamiento tradicionales [9]. Los simuladores quirúrgicos laparoscópicos virtuales, por ejemplo, presentan una curva de aprendizaje más acelerada; el desarrollo de habilidades motoras y de coordinación se ven potenciadas bajo el uso de esta tecnología. La mayoría de simuladores y estudios han sido llevados a cabo alrededor de plataformas quirúrgicas como el *daVinci Trainer (Intuitive Surgical, Inc., Sunnyvale, California, USA)* o el *Xperience (Surgical Science, Goteborg, Sweden)*; debido a su robustez y perfecta integración entre función y forma [10], [11].

En cuanto a los simuladores neuroquirúrgicos, existen dos enfoques: los simuladores físicos y los simuladores virtuales. En el primer caso, se tiene un entorno pedagógico de bajo costo, orientado a facilitar el aprendizaje experimental para el entrenamiento en cirugía robótica guiada por imágenes. En este caso se simula la inserción y extracción de una aguja, similar a una biopsia guiada por imagen. El entorno de entrenamiento está

compuesto por un fantasma impreso en 3D (también llamado *phantom*), una CT del fantasma, un entorno de realidad virtual que incluye información háptica, un sistema robótico de 1 DOF para inserción y extracción y un *Novint Falcon*, un manipulador de 3DOF con retroalimentación háptica [5]. Lo particular de este entorno, es que posee tanto una parte física como una virtual, razón por la que de todos los simuladores es el único en el que hay que hacer un proceso de registro/emparejamiento; no obstante, no está enfocado hacia la cirugía de base de cráneo (transorbital, transnasal, o transoral) sino a la cirugía intracraneal. Para el segundo enfoque hay varios ejemplos, uno de ellos es el simulador de entrenamiento quirúrgico para cirugía endoscópica de base de cráneo asistida por robot [12]. Este simulador aprovecha la configuración de teleoperacion maestro-esclavo para replicar estos procedimientos en un entorno de realidad virtual para fines de capacitación de cirujanos. Otro es un simulador para cirugía endonasal asistida por robot; este fue usado como medio de validación de unas trayectorias para acceder a los senos paranasales, reconstruidas a partir de una CT volumétrica [13]. Y finalmente, se tiene un simulador de realidad virtual de neurocirugía endonasal asistida por robot; donde se propone un algoritmo de retroalimentación háptica que combina dispositivos virtuales de guía y región prohibida para mejorar la seguridad en dicho procedimiento [14].

Debido a lo anterior, se desea crear una alternativa software de bajo costo y acceso libre, cuya utilidad como simulador de trayectorias neuroquirúrgicas con abordaje endonasal haya sido validada previamente por medio de las técnicas de registro/emparejamiento con objetos del mundo real. Debido a que el número de universidades con acceso a robos colaborativos o *"Cobots"*, está creciendo cada día, y especialmente la línea de *Cobots* UR *(Universal Robots, Odense, Dinamarca)* está ganando popularidad [15], es posible crear nuevas trayectorias, añadir nuevos procedimientos quirúrgicos e incluso validar los resultados con fantasmas creados a partir de otros modelos 3D.

Con esto en mente, se desea utilizar el software Unity como entorno virtual de pruebas, el software URSim como entorno virtual de validación, y el framework ROS como plataforma de comunicación. Gracias a un proyecto presentado por estudiantes de la Universidad del Cauca, la plataforma que reúne estas características ya ha sido desarrollada, por lo que no será necesario preocuparse por aspectos técnicos como la comunicación o la interfaz de usuario. Sin embargo, varios cambios deben realizarse para cumplir con los objetivos de este trabajo, siendo el principal la planeación y ejecución de trayectorias

neuroquirúrgicas desde el simulador; para lo cual se cuenta además con un cráneo virtual y un cráneo impreso en 3D, provenientes de una CT, donde se podrán realizar las pruebas necesarias.

Gracias al convenio interinstitucional establecido entre la Universidad del Cauca (Popayán, Colombia), y la Universidad Miguel Hernández (Elche, España), se realizó una estancia de investigación de tres meses en los laboratorios del grupo nBio, donde se llevaron a cabo actividades que permitieron validar el uso de una plataforma software en la planeación y ejecución de trayectorias neuroquirúrgicas con un robot UR3e real.

La estructura de esta monografía se divide en cinco capítulos que agrupan las temáticas principales del presente proyecto. El capítulo uno es una introducción al contexto que incentiva el desarrollo de este trabajo. El capítulo dos presenta una revisión del actual estado del arte. El capítulo tres se divide en ocho partes que abarcan desde los conceptos generales, pasando por el reacondicionamiento de la plataforma, hasta la ejecución de las pruebas experimentales. En el capítulo cuatro se detallan los resultados de las pruebas y finalmente el capítulo cinco presenta las conclusiones y los trabajos futuros.

2. Estado del arte

En esta sección se tratará el tema de los simuladores quirúrgicos asistidos por robots. Con esto en mente, se quiere investigar las características de una plataforma que permita simular trayectorias neuroquirúrgicas y ejecutarlas en el mundo real. Es decir, se quiere conocer la efectividad de la plataforma en la simulación de trayectorias quirúrgicas usando un robot colaborativo UR3 en un entorno virtual. Para lograrlo, se planteó incluir en el mundo virtual un cráneo proveniente de una resonancia magnética o tomografía computarizada CT, realizar el registro entre el *phamton* del cráneo real y el cráneo virtual, calcular el error de registro y posteriormente validar las trayectorias planeadas.

En total se encontraron 79 documentos relacionados con el tema de investigación que cumplían con los criterios de inclusión: Artículos cercanos al tema específico a tratar, no menores al año 2000 y en idioma inglés o español. Estos textos pueden ser agrupados con base en las características de cada documento, según las siguientes 4 temáticas:

- Introducción a la cirugía robótica.
- Registro/emparejamiento de imágenes médicas.
- Procedimientos quirúrgicos intracraneales y de base de cráneo.
- Plataformas de simulación neuroquirúrgicas.

Los textos de “Introducción a la cirugía robótica” se caracterizan por abordar el tema de manera amplia; desde los tipos de cirugías asistidas por robots, hasta las ventajas y desventajas de dichos sistemas. Los textos de “registro/emparejamiento de imágenes médicas” se caracterizan por analizar las herramientas desarrolladas en años recientes en la adquisición, segmentación y registro de imágenes médicas, particularmente las innovaciones en hardware o software utilizadas en la reducción del error de registro. Los textos de “procedimientos quirúrgicos intracraneales y de base de cráneo” se caracterizan por clasificar la neurocirugía desde diferentes enfoques y dar a conocer los avances en el plano práctico actual. Por último, los textos de “plataformas de simulación neuroquirúrgicas” se caracterizan por presentar los avances referentes a los simuladores virtuales neuroquirúrgicos y su capacidad de emparejamiento en el mundo real.

Hasta hace algunos años el potencial de los robots era desconocido en la medicina, debido en gran parte a que su implementación se dio principalmente en la industria de bienes de producción; los robots podían ejecutar tareas sencillas y repetitivas con gran precisión, pero tareas de mayor complejidad eran prácticamente imposibles de realizar. Sin embargo, el desarrollo de la instrumentación, capacidad de procesamiento, software y redes de comunicación más avanzadas, ha permitido la inserción de la robótica en un amplio espectro de áreas de la actividad humana [16]; desde el entretenimiento, la vida doméstica y la educación, hasta la industria militar, energética o automotriz, etc. Por supuesto la medicina no es la excepción, pero al tratarse de una de las ciencias de la vida, requiere de especial cuidado e investigación rigurosa.

La robótica en la medicina tiene diferentes enfoques; robots asistenciales *(desinfectant-bots* [17], *companion-bots* [18]), sustitutos robóticos de telepresencia [19], *robotic nurse* [20]), robots para rehabilitación (prótesis [21] y ortesis [22]), robots de almacenaje y distribución de medicamentos (*pharmabots* [23]), microrobots (*endoscopy-bot* [24], *pill robot* [25]), robots de entrenamiento clínico [26], etcétera. Y finalmente, la robótica quirúrgica. Por otro lado, la inteligencia artificial y especialmente, la visión artificial, están revolucionando el panorama de la medicina, ya sea generando diagnósticos más precisos en menos tiempo que un médico [27], o prediciendo nuevos brotes de alguna epidemia [28] gracias al análisis de miles de datos de forma simultánea.

Cabe aclarar que los procedimientos quirúrgicos han ido evolucionando desde las cirugías abiertas, pasando por las cirugías mínimamente invasivas, hasta las cirugías robóticas. La cirugía mínimamente invasiva ha demostrado ser superior a la cirugía abierta, gracias a una menor pérdida de sangre, un excelente efecto cosmético, menos dolor y tiempo de recuperación y estadías hospitalarias más cortas. No obstante, este procedimiento también presenta unas desventajas como una percepción visual limitada, destreza distal reducida, coordinación mano-ojo invertida y sensibilidad háptica reducida [29]. Por otro lado, la cirugía mínimamente invasiva asistida por robots supera estos obstáculos al permitir amplificar las capacidades de los cirujanos a niveles nunca antes vistos (ergonomía, destreza, precisión, cirugías a distancia) [30].

Existen diferentes técnicas quirúrgicas que requieren diferentes sistemas robóticos; por ejemplo, los procedimientos endoscópicos (laparoscopia, toracoscopia) o ”*keyhole”* se enfocan en la manipulación bimanual, orientación visual y tele operación. Los procedimientos neuroquirúrgicos y ortopédicos se enfocan en un registro preciso y

trayectorias intraoperativas restringidas, mientras que los procedimientos percutáneos se enfocan en la compacidad de los instrumentos [29]. Las disciplinas donde esta tecnología está siendo aplicada actualmente, son:

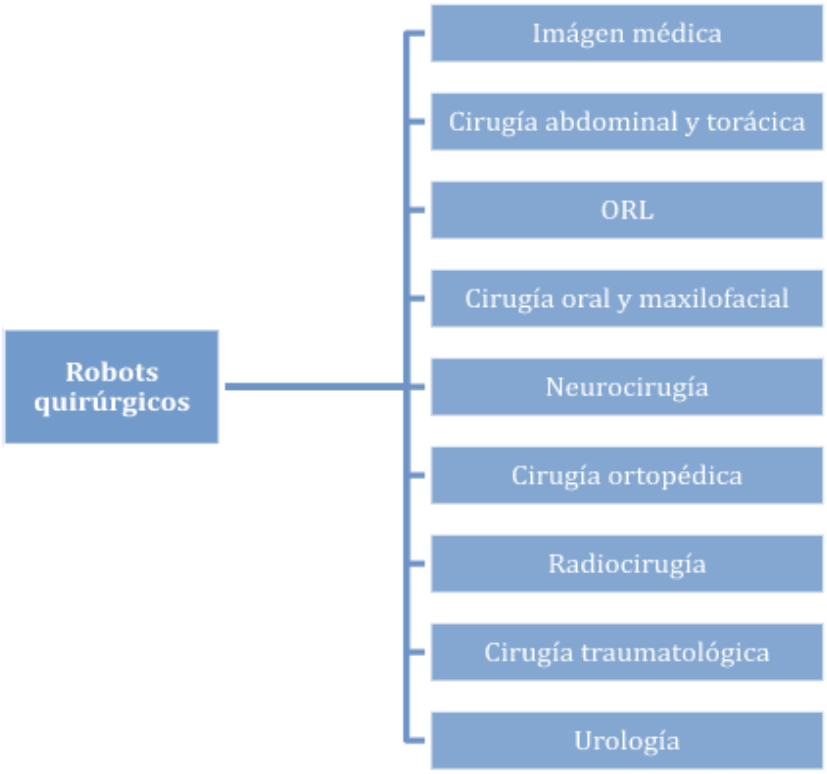


Figura 2.1: Disciplinas donde se aplican cirugías con robots [31], [32].

El potencial clínico de la cirugía asistida por robot (CAR) se empezó a vislumbrar desde mucho antes de ser introducido el computador a los hospitales, centros de investigación y programas de formación académica. La cirugía asistida por computador (CAC) ha permitido mejoras tanto de seguridad como precisión en la sala de operaciones. Por ejemplo, en la cirugía ortopédica asistida por ordenador (CAOS), el taladrado de huesos largos es una tarea compleja, que el cirujano debe realizar con gran precisión para evitar el calentamiento excesivo del hueso [31].

El desarrollo de la neurocirugía guiada por imágenes fue un avance significativo en el tratamiento de lesiones intracraneales. La planificación quirúrgica se realiza mediante la adquisición de imágenes de resonancias magnéticas o tomografías computarizadas. Sin embargo, estas imágenes no reflejan los cambios que pueden ocurrir durante el procedimiento real debido a la naturaleza dinámica del cuerpo humano [32]; fenómenos

como distorsión en la imagen o el *brain shift* (deformaciones y desplazamientos del cerebro), así lo demuestran [33]. Por esta razón, la adquisición de imágenes durante la intervención quirúrgica proveería al neurocirujano de la información necesaria para actuar eficazmente frente a las condiciones intraoperatorias cambiantes. Así pues, el establecimiento de correspondencia entre imágenes preoperatorias e intraoperatorias es de suma importancia para muchos procedimientos clínicos, tales como fusión de imágenes, creación de atlas de órganos o monitoreo de tumores.

Este establecimiento de correspondencia se conoce como registro de imágenes, y consiste en el proceso de transformación de diferentes *datasets* de imágenes (de un mismo punto anatómico del cuerpo humano) en un solo sistema de coordenadas, de modo que los *datasets* de imágenes queden emparejados. Este procedimiento es necesario al analizar imágenes provenientes de diferentes puntos de vista, tiempos, sensores o modalidades de adquisición. Hasta hace poco el registro se realizaba de forma manual por un especialista [[34]. En cambio, en el registro automático las imágenes deben ser alineadas por medio de una serie de transformaciones geométricas, las cuales mapean la ubicación de una imagen hacia una nueva ubicación proveída por una imagen de referencia. Las transformaciones pueden ser rígidas, afines, proyectivas (elásticas) o no rígidas [35], pues las imágenes multimodales de un paciente son obtenidas en diferentes momentos. La planificación pre-quirúrgica, la visualización y orientación durante el procedimiento y la evaluación posterior al procedimiento son los componentes centrales de la cirugía guiada por imágenes. Así, el cirujano tiene mejor visualización y navegación [38], [36].

Sin embargo, el proceso de emparejamiento nunca va a realizarse con una coincidencia de las imágenes del 100 %. Este problema se llama error de registro, y se han desarrollado los siguientes métodos para abordarlo:

- Métodos computacionales de predicción y corrección [37], [32].
- Proyección de patrones a cuadros en superficies junto con algoritmos de registro rígido [38].
- Aplicaciones basadas en inteligencia artificial [39], visión de máquina [40], *machine learning* [41] y *deep learning* [34] [42].
- Enfoques no paramétricos para el registro elástico de imágenes multimodales (entropía, varianza) [43].

- Aplicación de técnicas topográficas ópticas (espectroscopia funcional de infrarrojo cercano, tomografía de difusión óptica, imagen de topografía óptica) [44].
- Corrección manual por medio de gestos [45].
- Realidad aumentada [45], [46], [47], [46], [47], [48].

Pero no solo se ha presentado un gran avance en las técnicas de registro y rastreo, también se han diseñado y desarrollados prototipos robóticos tipo serie [49], paralelo e incluso un instrumento robótico portátil [50], con el fin de desarrollar tareas como insertar herramientas quirúrgicas tales como agujas de biopsia, herramientas de succión, catéteres, entre otras. Estos dispositivos son supervisados por el cirujano en el posicionamiento preciso y navegación de la aguja durante cirugías estereotácticas intracraneales. También, aunque menos frecuente, se han hecho incursiones en el área con prototipos teleroboticos [51]. Por otro lado, entre los robots comerciales neuroquirúrgicos más conocidos, se pueden encontrar los siguientes: El sistema NeuroMate (*Integrated Surgical Systems, Davis, CA)*, o el sistema ROSA *(MedTech Surgical Inc., Newark, NJ)*. Ambos sistemas permiten realizar cirugías guiadas por imagen, tanto *frameless* como *framebased*; ya sea para ejecutar biopsias, para ubicar electrodos para DBS [52] o procedimientos como LITT, SEEG, RNS, entre otros. Además, estos sistemas cuentan con diferentes métodos de registro y rastreo (localización) como:

- Para el *NeuroMate*, registro con fiducias (marco estereotáctico como marcador) y sistema de rastreo con infrarrojo, o registro con marcadores (tornillos) y sistema de rastreo con infrarrojo [53]. Para el ROSA, registro con fiducias (tornillos en cráneo) y registro con escáner láser del rostro [54].

La cirugía en el área craneal incluye operaciones del complejo fronto-cigomático-maxilar, la cavidad nasal, los senos paranasales, el oído y la base del cráneo, que tienen una proximidad cercana a estructuras altamente críticas como nervios, vasos, el ojo, el órgano coclear y laberíntico, o el cerebro [55]. Los cirujanos han desarrollado dos enfoques para acceder a la base del cráneo: El enfoque transesfenoidal endonasal y, más recientemente, el enfoque endonasal endoscópico expandido. Sin embargo, operar de esta forma con herramientas no articuladas es técnicamente muy difícil por lo que no se ha adoptado ampliamente [56] debido a los requerimientos de ángulos mínimos entre la instrumentación robótica (efecto de embudo), el ángulo de entrada cerrado y el tamaño de

la instrumentación [57]. Un caso particular de cirugía transesfenoidal endonasal endoscópica asistida, fue el uso de un brazo robótico llamado *iArmS* (*Denso Design*, Japon), para sostener la mano no dominante del cirujano que sujetaba el endoscopio; su uso redujo la fatiga y estabilizó la mano durante el procedimiento, permitiendo que el temblor de la imagen de video disminuyera [58]. Los resultados muestran que el reposabrazos inteligente parece ser seguro y eficaz, mejorando la precisión considerablemente.

Por otro lado, la cirugía neuro endoscópica transorbital es una técnica relativamente nueva, que no solo permite acceder al contenido de la órbita ocular, sino también al compartimiento intracraneal, incluyendo la fosa craneal anterior, la fosa media y el seno cavernoso lateral. Gracias a esta técnica es posible diseñar nuevas trayectorias quirúrgicas, mejorando la visualización y el tratamiento de lesiones [59]. Finalmente, se ha hecho un primer acercamiento a la cirugía robótica transoral, por medio del daVinci Xi [60].

Como se puede observar, la utilidad de los dispositivos robóticos continúa expandiéndose en neurocirugía y a pesar de las mejoras en maniobrabilidad y visualización 3D, los ejemplos exitosos de implementación robótica para la neurocirugía endonasal endoscópica o de la base del cráneo siguen siendo escasos; limitaciones en cuanto a tamaño y herramientas específicas serían la causa [61].

A continuación, se presenta un resumen general de los temas discutidos anteriormente:

Tabla 2.1: Clasificación de neurocirugías robóticas, arquitecturas de control y métodos de registro / localización.

Classification					Control Architecture	Registration Methods				Tracking Methods
Neuro-surgery	Microscopic	Intra-cranеal / Skull base	Open surgery		Shared [72]	Not needed				Not needed
	Stereotaxic	Intra-cranеal	Frame-based stereotaxic		Shared	Optical R.	Pair-point R.	Marker-based R.	Frame-mounted fiducials [64]	Optical T. [64]
			Frame-less stereotaxic (Image-based neuronavigation)		Supervised and Shared	Mechanic R.	Marker-based R.	Skin-mounted fiducials [66]		Mechanical T. [20]
						Optical R.	Pair-point R.	Marker-based R.	Bone-mounted fiducials [66]	Acoustic T. [20]
									Skin-mounted fiducials [73]	Optical T. (passive, active, checkerboard) [20]
								Marker-less R.	Anatomical landmarks [73]	
							Surface-matching R. [73,74, 73]			
						Magnetic R. [20]				Electromagnetic T. [20]
						Hybrid R. [20]				Glass-fiber T. [20]
	Endoscopic	Skull base	Endoscopic surgery	Transorbital	Shared	Not needed				Not needed
				Transnasal (Endonasal)	Shared [29]	Not needed				Not needed
				Transoral	Shared	Not needed				Not needed

Fuente: Autores

Por otro lado, los textos [2] y [6] tienen en común que se enfocan en encontrar parámetros de medición estándar de habilidades adquiridas con el uso de simuladores en el área de la laparoscopia principalmente, mientras que en [7] y [8] se enfocan en explorar nuevas herramientas de desarrollo de simuladores quirúrgicos. Con base en la literatura encontrada, se pudo establecer que el método preferido de estudio de factibilidad y viabilidad de cualquier nueva herramienta software o hardware, es realizar pruebas en estudiantes de medicina y profesionales de la salud, para conocer de primera mano la eficacia de las herramientas desarrolladas. La metodología más idónea es identificar dos grupos de control, uno que utilice la herramienta y otro que continúe una formación tradicional, con el fin de que después de un tiempo determinado se evalúen las capacidades de cada grupo (velocidad, precisión, cansancio, etc.) en la ejecución de procedimientos quirúrgicos similares.

De la información obtenida a partir de los textos consultados, se logró identificar un vacío en cuanto a la implementación de software capaz de realizar trayectorias neuroquirúrgicas endonasales con capacidad de emparejamiento con el mundo real; tanto como herramienta de investigación para desarrollar proyectos encaminados hacia la implementación de un control compartido robot-cirujano, como un control autónomo

supervisado en procedimientos quirúrgicos sencillos. Por otro lado, el robot de preferencia a utilizar en las plataformas de entrenamiento existentes es el da Vinci, razón por la que escoger un robot más flexible en funcionalidad y más ampliamente distribuido en universidades y centros de investigación, como el UR3, da un nuevo enfoque en la materia.

3. Materiales y métodos

3.1. Descripción del proyecto original

El proyecto original “Plataforma software para la manipulación de objetos 3D para un robot UR [110]”, fue concebido a partir de la integración de diferentes herramientas software y hardware. Las herramientas software pueden catalogarse como principales y secundarias. Las principales son: la plataforma Unity, el *framework* ROS y el simulador URSim; mientras que las secundarias son: el protocolo de comunicación *Websocket*, las librerías *Rosbridge* y *RosSharp* y la extensión *ExternalControl*. Por otro lado, las herramientas hardware hacen referencia a los ordenadores, la red (cables Ethernet, conmutadores, enrutadores) y finalmente el robot. A continuación, se describirá brevemente cada una.

El UR3e es un robot colaborativo (COBOT) de 6 DOF fabricado por la empresa Universal Robots (Odense-Dinamarca) que, gracias a sus características de seguridad, trabajo cooperativo, facilidad de programación-control y costo, es ideal para uso en ambientes educativos, de investigación, e incluso productivos. Este robot tiene un alcance de 500 mm, una carga útil de 3 kg, un peso de 11.2 kg y una exactitud de 0.1 mm; además, cuenta con una pantalla digital (*Teach Pendant*) con la que se puede interactuar de forma directa con el robot. El software que funciona como interfaz gráfica entre el robot y el operador, se llama Polyscope y, aunque no es posible generar trayectorias complejas en este, permite la comunicación con otras aplicaciones relevando el control del robot.

Figura 3.1: UR3e con *Teach Pendant* [129].

Unity es una plataforma de desarrollo de videojuegos multiplataforma en 2D-3D, realidad aumentada y realidad virtual. Cuenta con motores gráficos y de física capaces de crear animaciones y simulaciones de gran calidad. El editor del programa tiene 4 paneles principales: La ventana de jerarquía, la ventana del inspector, la ventana del proyecto y las vistas del proyecto. El lenguaje de programación de Unity es C#.

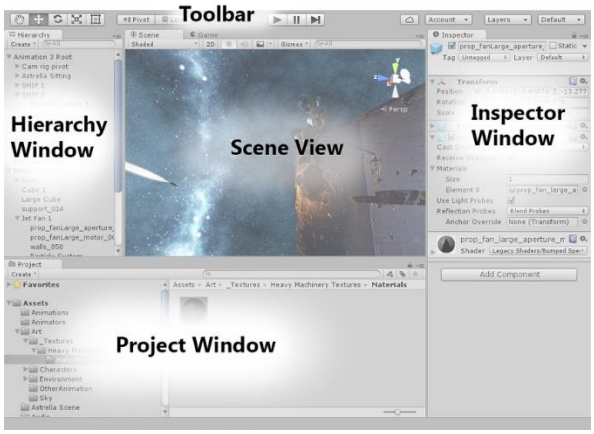


Figura 3.2: Paneles del editor de Unity.

ROS (*Robot Operating System*) es un *middleware* y *framework*, especializado en el desarrollo de aplicaciones para robots. Su arquitectura de comunicación está basada en canales (tópicos), que permiten la suscripción y publicación de estructuras de datos (mensajes) entre procesos individuales (nodos). Estos nodos pueden ejecutarse de forma distribuida en diferentes ordenadores. Los mensajes de ROS son archivos con extensión ".msg".

Para establecer un flujo de datos bidireccional entre Unity y los nodos de ROS, es necesario crear un canal de comunicación con la librería Rosbridge por medio del protocolo TCP/IP *WebSocket*. Los mensajes enviados desde y hacia Unity, son convertidos de un formato a otro por la librería RosSharp.

URSim es un simulador capaz de representar características cinemáticas de los robots UR (series CB y serie-e). Este programa permite realizar pruebas sin necesidad de involucrar el robot real.

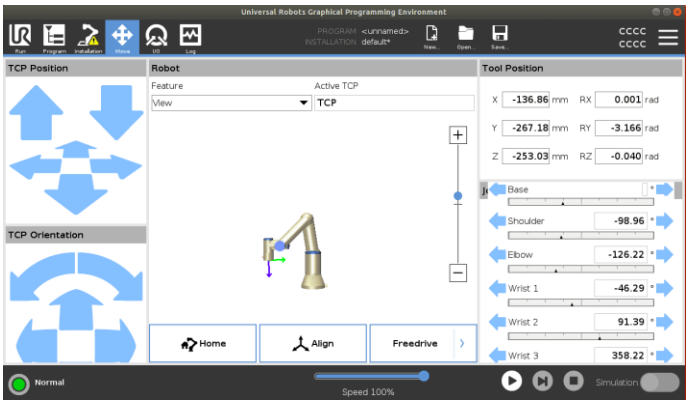


Figura 3.3: Interfaz URSim.

La extensión (plugin) *ExternalControl* es un accesorio de *Polyscope*, que se encuentra también en el simulador. Esta herramienta permite la manipulación del robot por dispositivos externos, a través de una red ethernet.

Ahora, el objetivo del proyecto original era crear una aplicación en Unity capaz de manipular un robot UR3e. En ella se simula un entorno virtual con un gemelo digital, que reproduce de forma precisa los movimientos del manipulador real, y un menú con el panel de conexión y los 3 modos de control, con los que el usuario podrá interactuar.

En el panel de conexión el usuario puede establecer comunicación ya sea con el robot real o con URSim, por medio de la dirección IP.

Figura 3.4: Panel de conexión.

El panel de control articular cuenta con 6 deslizadores con un rango de -360 a 360 unidades, cuyo valor define el movimiento rotacional de cada articulación del robot. El usuario puede crear, guardar y cargar listas de valores articulares para crear trayectorias.

Figura 3.5: Panel de control articular.

El panel de control cartesiano permite ingresar por teclado 3 posiciones y 3 orientaciones, para ubicar el efector final del robot en un punto especifico del espacio tridimensional. El usuario puede crear, guardar y cargar listas de posiciones y orientaciones para crear trayectorias. Esta opción cuenta con un deslizador que permite controlar la velocidad del robot.

Figura 3.6: Panel de control cartesiano.

El panel de control por trayectoria libre permite definir de forma gráfica la trayectoria que debe realizar el efector final del robot, por medio de la manipulación con el ratón de una línea rosa ubicada en el centro del escenario. El usuario puede crear, guardar y cargar las trayectorias generadas por la línea rosa. Esta opción cuenta con un deslizador que permite controlar la velocidad del robot.

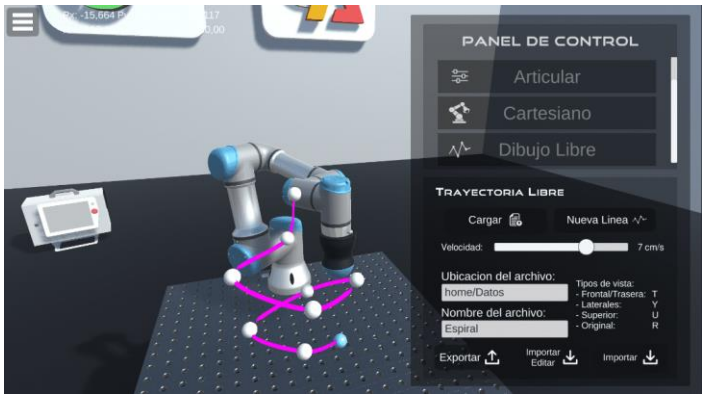


Figura 3.7: Panel de control por trayectoria libre.

3.2. Proyecto actual

El proyecto actual se enfoca particularmente en el modo de control por trayectoria libre. El objetivo principal de este trabajo consiste en utilizar esta herramienta como un dispositivo de neuronavegación. Para ello se realizaron modificaciones específicamente en Unity, tanto en los *scripts*, como en los *assets* y la propia interfaz. Las contribuciones más importantes fueron:

- Cambiar de una orientación estática del efector final, a una orientación dinámica.

- Implementar un sistema robusto de cámaras, para crear tomas dinámicas desde un punto en el espacio 3D.

Por otra parte, durante la parte experimental fueron necesarios otros elementos, que ayudaron a realizar las diferentes pruebas llevadas a cabo en el laboratorio. Estos elementos son:

- Un modelo 3D de un cráneo y un *phantom* de cráneo, ambos reconstruidos a partir de una tomografía computarizada de un paciente real.

- Una herramienta que se puede acoplar al efector final del robot real, con el correspondiente modelo 3D para el gemelo digital.

Finalmente, la pieza central del laboratorio es la estación de trabajo; una estructura de metal en la que se pueden realizar diferentes pruebas experimentales. Está compuesta por los siguientes elementos:

- 2 robots UR5.
- 1 robot UR3e.
- 3 pantallas táctiles (*Teach Pendant*).
- 3 cajas de control.
- 1 panel de cámaras *Optitrack*.

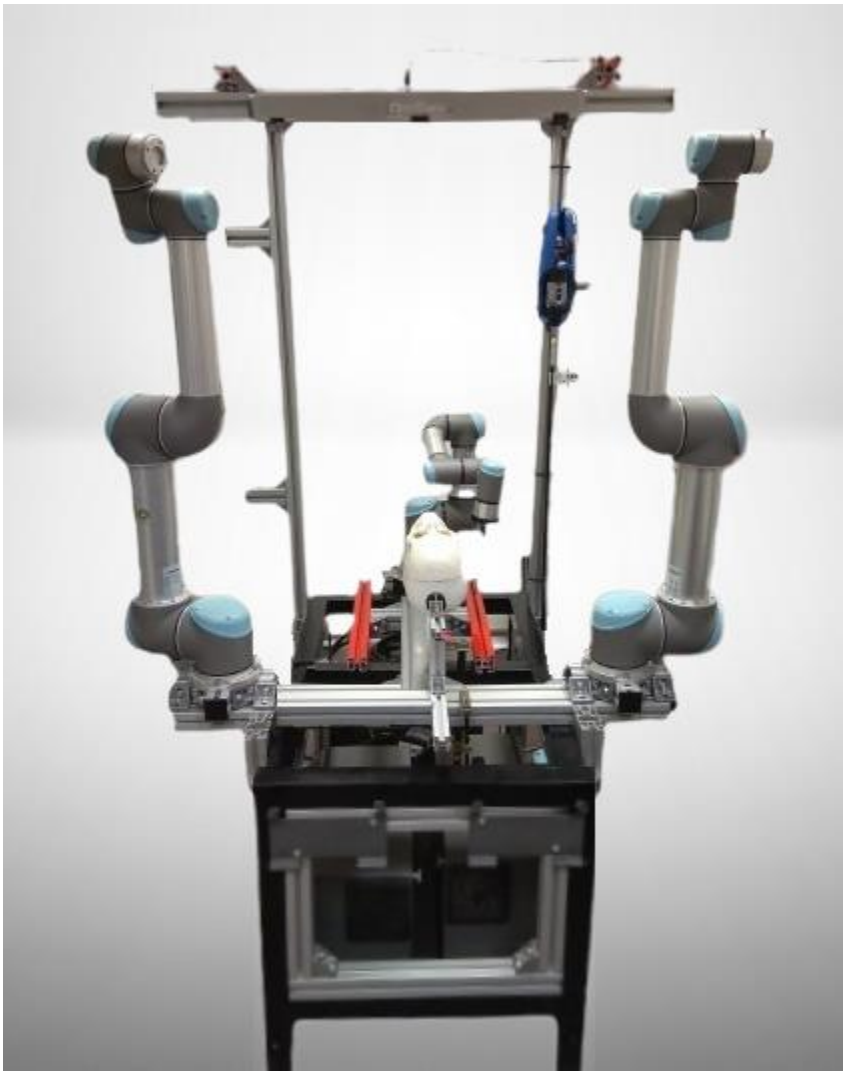


Figura 3.8: Estación de trabajo.

Como se puede observar, los robots están montados y asegurados a estructuras metálicas tipo riel (V-slot). Estos serán utilizados más adelante para montar el marco que sostiene el *phantom* de cráneo. Es importante señalar que de la estación de trabajo solo se hará uso del robot UR3e y del panel de cámaras *Optitrack*.

3.3. Segmentación y escalamiento del cráneo digital.

3.3.1. Segmentación

La segmentación de imágenes médicas es el método gráfico que recibe como entrada una imagen digital en tonos de grises ya sea una tomografía computarizada o una resonancia magnética, que representa una determinada región anatómica, cuya salida está constituida por un conjunto de regiones poligonales catalogadas según un criterio determinado [62]. El proceso de segmentación implica la separación de las partes constituyentes, tejidos blandos y los huesos, en una imagen médica. Esta se utiliza para aislar una estructura 2D o 3D y dibujar las estructuras anatómicas de forma separada. El proceso puede ser manual o automático. En el caso de la segmentación manual, el usuario utiliza herramientas para dibujar las estructuras anatómicas en la imagen médica. En el caso de la segmentación automática, se utilizan algoritmos para identificar las estructuras anatómicas en la imagen médica [63]. La segmentación precisa de las imágenes médicas es un paso importante en el correcto diagnóstico y planificación de tratamientos médicos [64].

La segmentación de imágenes médicas se puede realizar utilizando software especializado como 3DSlicer [65], [66].

3D *Slicer*

3D *Slicer* es una plataforma de software libre para la visualización, procesamiento, segmentación, registro y análisis de imágenes médicas, biomédicas, u otras imágenes como mallas 3D; además para la planificación y navegación de procedimientos guiados por imágenes [67]. Es un software gratuito disponible en diferentes sistemas operativos como Debian, Linux, Mac, Windows, y contenedores Docker [66]. También se puede desplegar atreves de puentes implementado un nodo (ROS) *Robot Operating System* llamado ROS-IGTL-Bridge, utiliza el protocolo OpenIGTLink para establecer una conexión de red TCP/IP entre el entorno ROS y el software informático externo de imágenes médicas. El nodo exporta mensajes de ROS a software externo y viceversa a través de la red, lo que permite compartir datos de forma transparente entre dispositivos basados en ROS y plataformas informáticas de imágenes médicas [68], como se muestra en la (Figura 3.9).

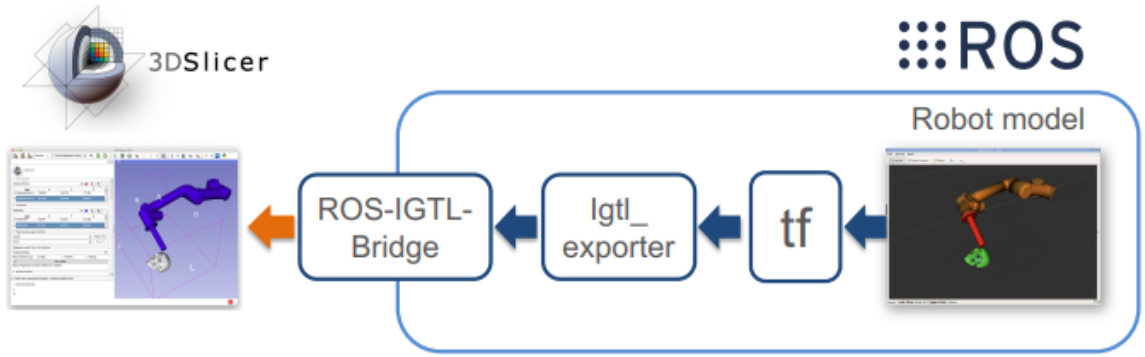


Figura 3.9: ROS-*IGTL-Bridge* [68].

La arquitectura de 3D Slicer sigue una estructura modular, dividiéndose en 3 capas, la capa básica, capa de algoritmos y la capa de aplicación [69], como se muestra en la (Figura 3.10). El enfoque de su desarrollo software se basa en código abierto como VTK, ITK, CTK, Cmake, Qt y Python, los principales lenguajes de programación son Python y C++.

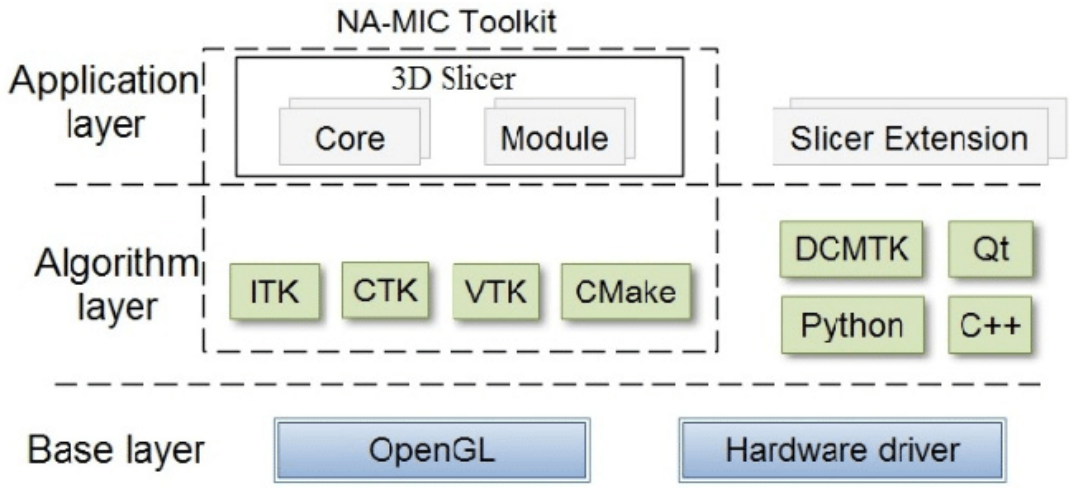


Figura 3.10: Arquitectura 3D Slicer [69].

Para el trabajo de segmentación se propusieron tres fases:

1. Cargar los archivos DICOM (*Digital imaging* and *comunication* in medicine) a 3D Slicer
2. Realizar la segmentación utilizando los métodos que nos brinda 3D Slicer.
3. Exportar el cráneo segmentado.

Preparación de las imágenes tipo DICOM

Digital imaging and *comunication* in medicine (DICOM) es un estándar que promete una representación e intercambio estandarizados de imágenes médicas e información relacionada de varias fuentes radiológicas y de forma de onda [70]. Gracias a este estándar se solucionan los problemas de interoperabilidad automática entre los sistemas informáticos de imágenes biomédicas [71].

El estándar DICOM se ha popularizado en el ambiente médico, ya que este tipo de archivo es la solución en imágenes médicas, entendiendo que las imágenes generadas de diferentes maneras como la tomografía computarizada (TC), ultrasonidos, medicina nuclear, resonancia magnética (RM) son de este tipo. Además, son muy importantes los archivos DICOM pues tienen la particularidad de asociar datos del paciente tales como la hora y fecha de adquisición, nombre del paciente y tipo de modalidad entre otros. A través del formato DICOM se estandariza el suministro de la información clínica optimizando el flujo de trabajo de cualquier hospital [72].

En 3D Slicer se utiliza el módulo DICOM que permite cargar archivos de este formato, el procedimiento se realizó sobre una tomografía computarizada de un paciente de edad adulta, el cual contiene 290 imágenes transversales del cráneo a una distancia entre imágenes de 1,25mm (Figura 3.11).

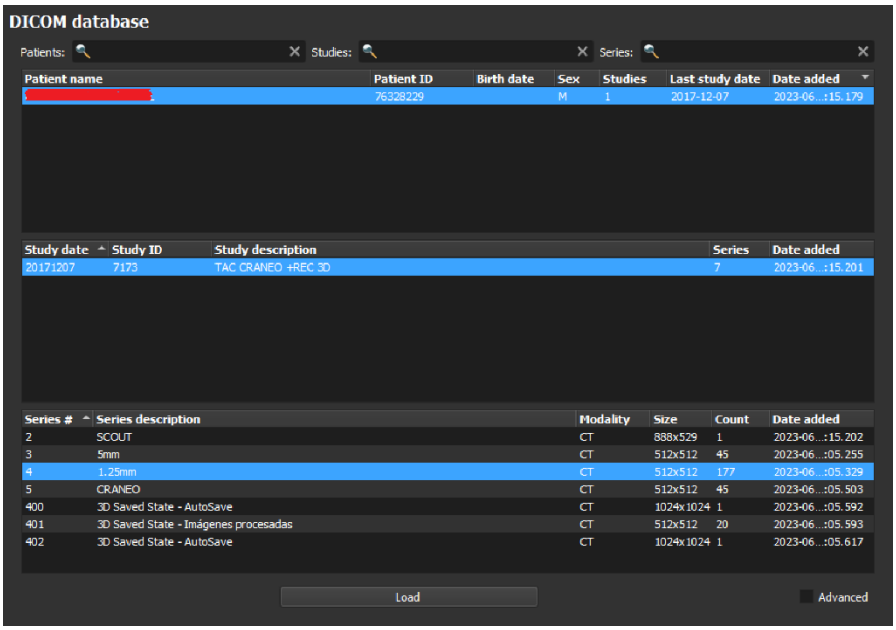
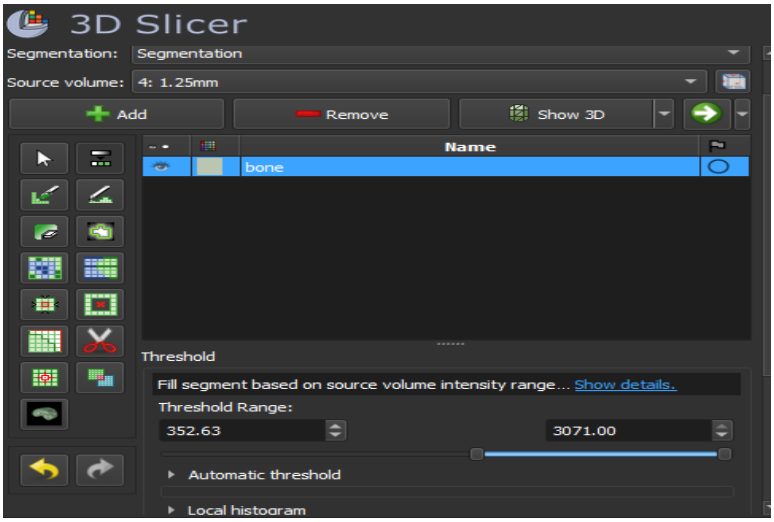


Figura 3.11: Carga de archivos DICOM en Slicer 3D.

Segmentación utilizando 3D Slicer

Una vez cargada las imágenes en la plataforma de 3D Slicer, el paso a seguir es dirigirse al módulo *segmentation*, donde se despliega un menú con diferentes herramientas. Al seleccionar en *add* se agrega una nueva segmentación y se puede utilizar la herramienta *threshold*, al dar clic despliega un cuadro con una barra deslizante donde se puede indicar el rango del umbral que se quiere segmentar, se establece un límite en el cual los tejidos, músculos, piel y órganos no se representen. El objetivo principal en este procedimiento es la segmentación del cráneo (Figura 3.12).



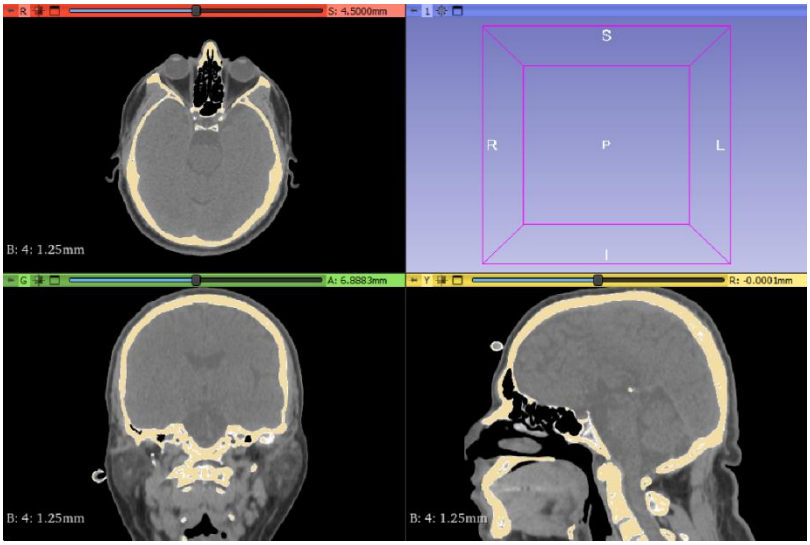


Figura 3.12: Procedimiento *Threshold* a la TC

Al ajustar el contraste en la segmentación y aplicar los cambios ya se puede obtener el objeto 3D al dar clic *Show 3D* (Figura 3.13).

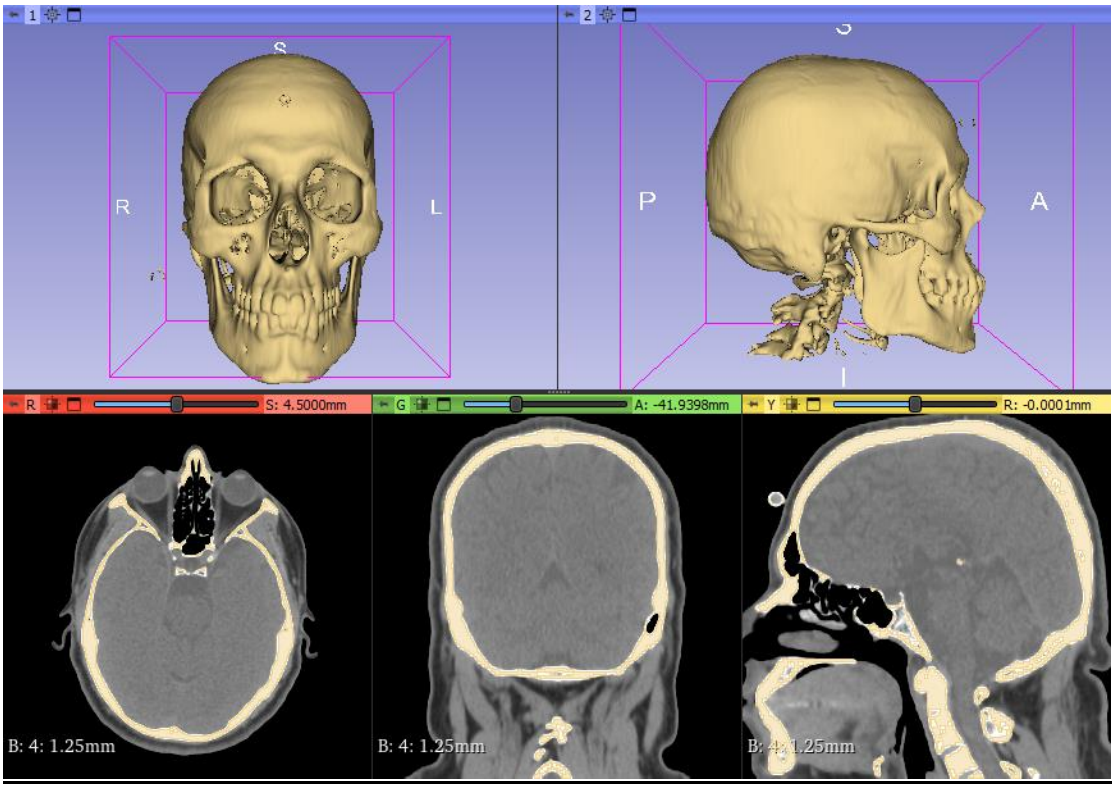


Figura 3.13: Vista del cráneo segmentado en Slicer 3D.

Exportar el cráneo

Después de obtener el cráneo segmentado en 3D Slicer se exporta a un formato reconocible, en este caso del tipo .OBJ. Para esta parte se selecciona la flecha verde que está al lado de *Show 3D,* este desplegara un menú donde se puede seleccionar el formato de preferencia (Figura 3.14).

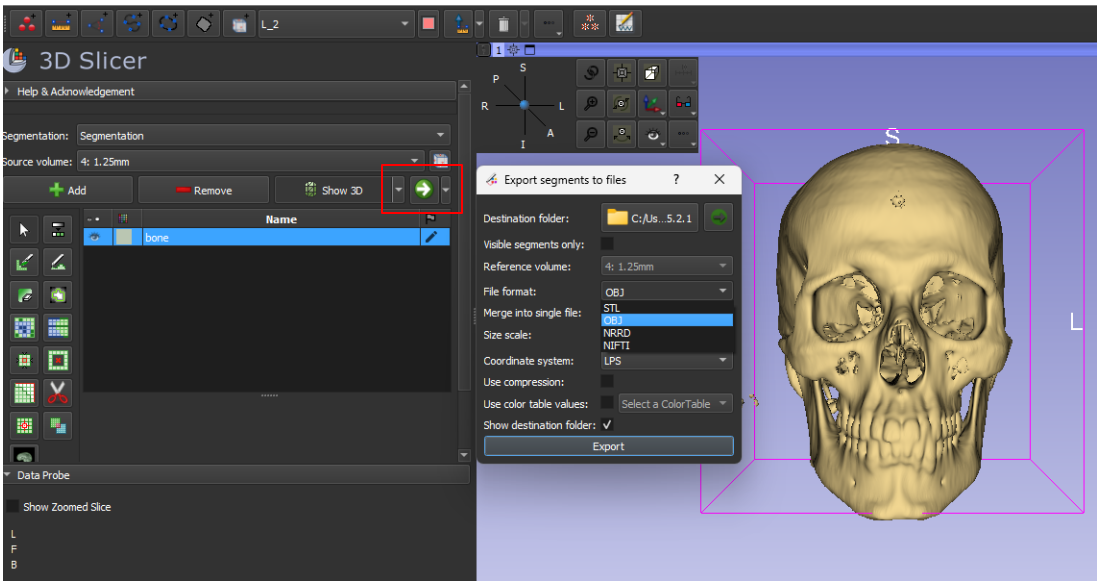


Figura 3.14: Exportar cráneo segmentado.

3.3.2. **Escalamiento.**

La escalabilidad de objetos 3D se refiere a la capacidad de cambiar el tamaño y la proporción de un modelo digital sin afectar su calidad, precisión o detalles. El escalamiento es un tipo de transformación que se puede hacer en diferentes dimensiones tanto en 2D, 3D, 4D o ND.

Escalamiento 2D.

El escalamiento en 2D implica el cambio en sus dimensiones de una figura geométrica plana donde cada punto $p = (x_1, x_2)$ es transformado por la multiplicación de dos factores s_1 y s_2 a lo largo de los ejes X_1 y X_2 respectivamente. De esta forma, las coordenadas del nuevo punto $p' = (x_1', x_2')$ se obtienen de [73]:

$$x_1' = (x_1 * s_1)$$
$$x_2' = (x_2 * s_2)$$

Sea $s = (s_1, s_2)$ el vector de los factores de escalamiento y $S(s)$ la matriz de escalamiento. El escalamiento de un punto p en 2D se puede expresar de la siguiente manera, como un producto matricial:

$$[x_1' , x_2' \ 1] = [x_1, x_2 \ 1] \cdot \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.\ 1)$$

Escalamiento 3D.

Teniendo en consideración la idea anterior, el escalamiento a 3D supone un cambio en el tamaño de un poliedro, donde cada punto $p = (x_1, x_2, x_3)$ es transformado por la multiplicación de 3 factores de escalamiento s_1, s_2 y s_3 a lo largo de los ejes X_1 , X_2 y X_3 respectivamente, de esta forma, las coordenadas del nuevo punto $p' = (x_1', x_2', x_3')$ se obtienen:

$$x_1' = (x_1 * \ s_1)$$
$$x_2' = (x_2 * \ s_2)$$
$$x_3' = (x_3 * \ s_3)$$

Sea $s = (s_1, s_2, s_3)$ el vector de los factores de escalamiento y $S(s)$ la matriz de escalamiento. El escalamiento de un punto p en 3D se puede expresar de la siguiente manera, como un producto matricial:

$$[x_1' , x_2', x_3' ,1] = [x_1, x_2 , x_3, 1] \cdot \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.\ 2)$$

La (figura 1.7) representa el efecto del escalamiento en 3D de una figura cualquiera para casos prácticos con $s_1 = 2,\ s_2 = 2.5,\ s_3 = 1.5$.

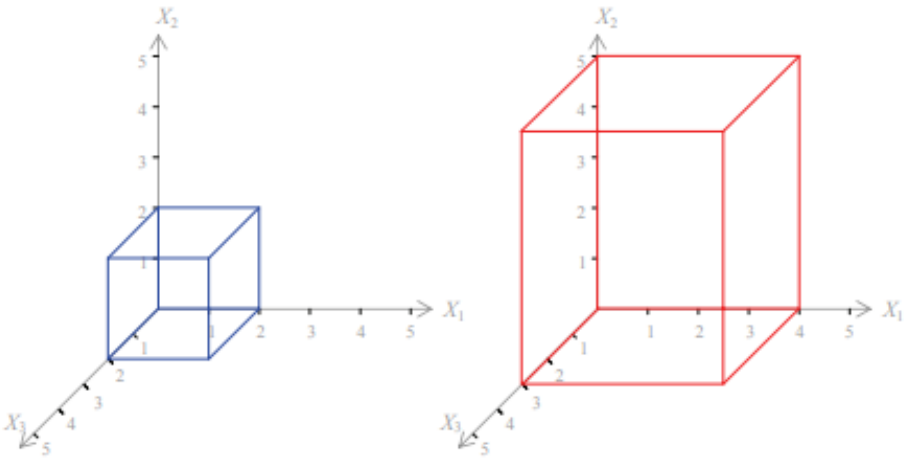


Figura 3.15: Ejemplo de escalamiento 3D [73].

Existen muchos programas que se pueden usar para escalar imágenes en 3D. Entre ellos:

HOLODICOM:

Figura 3.16: *HoloDicom* [74].

Es una herramienta que permite la visualización de una pila de imágenes DICOM, dirigida a las gafas HoloLens 2 [75]. Contiene funciones que permiten la manipulación de volumen, segmentación 2D y una interfaz de usuario muy intuitiva, tomadas directamente de tomografías computarizadas o resonancias magnéticas [74].

Blender:

Figura 3.17: Logo Blender [76].

Es un programa multiplataforma que funciona perfectamente en sistemas Linux, Windows, macOS entre otros, donde permite al usuario crear animaciones de alta calidad de modelos y datos en 3D [77]. Se usa en videojuegos y en el entretenimiento, también ha sido muy útil para generar visualizaciones científicas de alta calidad [78], como lo son datos astronómicos que adoptan una gran cantidad de formas, catálogos, cubos de datos, imágenes y simulaciones. Blender brinda a estos profesionales una herramienta útil para la animación, renderizado, y generación de modelos de datos astronómicos en gráficos tridimensionales [79].

Las características que resaltan de Blender son las siguientes [76]:

- Una potente interfaz de programación de aplicaciones (API) en Python, puede programarse para cargar datos de simulaciones numéricas.
- Control total de la cámara, el campo de visión y el renderizado.
- Creación y visualización en 3D, imágenes fijas o animaciones, edición de video, VFX, texturizado, y algunas simulaciones.
- El requisito de memoria y unidad es relativamente pequeño en comparación con otras suites de creación 3D, al emplear OpenGL que proporciona una experiencia consistente utilizando primitivas geométricas simples para crear escenas tridimensionales complejas.
- Simulaciones dinámicas para *softbodies*, partículas y fluidos.
- Capacidad para integrar gráficos creados por computador en un video con la posición, orientación y escala correcta en relación con los objetos presentes en la toma, a esto se le denomina *match moving*.
- Acepta una gran cantidad de formatos como TGA, JPG, STL, SGI, TIFF además de leer ficheros *Inventor*.
- Arquitectura de alta calidad, permitiendo un flujo de trabajo de creación rápido y eficiente.
- Edición de audio y sincronización de video.
- Características comunes en los juegos como detección de colisiones, recreaciones dinámicas y lógicas.

Los archivos que se obtienen de 3D Slicer tienen una escala diferente a la que se va a utilizar posteriormente, razón por la cual debe utilizarse Blender. Para hacer el escalado con precisión se debe tener una medida de referencia, la herramienta que nos entrega dicha precisión es la misma que analiza las imágenes primitivas en formato DICOM: 3D Slicer. Para hacer el escaldo se propusieron dos fases: medición y escalado.

Medición de referencia.

Para obtener una alta precisión en la medición del cráneo, la craneometría, que se define como la medición científica de las dimensiones del cráneo, hace la medida tomando puntos de referencia anatómicos. Estos puntos de referencia pueden establecerse utilizando diferentes métodos como la medición física directa o el uso de imágenes 2D de rayos X, o la tomografía computarizada [80].

Si se utiliza la tomografía computarizada, se importan las imágenes en el software 3D Slicer y se realiza la medición en los planos axial, coronal, sagital con la herramienta de medición que proporciona el software. Para una correcta medición como lo indica la literatura, se deben utilizar puntos de referencia anatómicos, para este procedimiento se utilizaron los siguientes:

En el plano axial se realiza la medición desde la parte superior del hueso nasal, también denominado hueso propio de la nariz. Este es un hueso par colocado a cada lado de la línea media y que se ubica justo en la parte superior de la nariz humana [81], hasta la parte externa del hueso occipital, hueso se encuentra ubicado en la parte posterior de la cabeza. Su función principal es la de proteger el cerebelo y los lóbulos occipitales del cerebro y para proporcionar unión a varios músculos y ligamentos [82].

En la (Figura 3.18) se representa dicha medición.

Figura 3.18: Medición plano axial.

En cuanto al plano coronal, la medición se hace desde el hueso cigomático derecho al izquierdo. Es importante resaltar que este hueso está a ambos lados de la cara, forma la mejilla y una parte de la cavidad del ojo, se encarga de formar el rostro humano, uniéndose con la mandíbula y los huesos cercanos a las orejas, la frente y el cráneo. Estos huesos protegen los nervios y los vasos sanguíneos de la cara y en ellos se fijan los músculos que ayudan a mover la mandíbula [83].

En la (Figura 3.19) se representa la correspondiente medición.

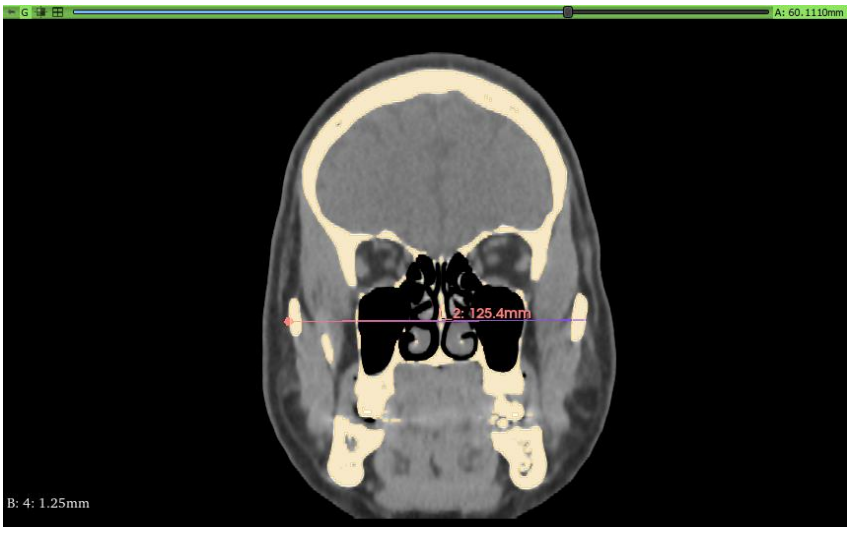


Figura 3.19: Medición plano axial.

Por último, se tiene la medición en el plano sagital que va desde la sínfisis del mentón, localizada en la parte anterior de la mandíbula. La mandíbula es un hueso impar, medio y simétrico, articula con los temporales y es el único hueso móvil de la cabeza, totalmente externo dibuja la parte inferior del rostro [84]. La medición se realiza hasta la unión del hueso parietal o como se conoce, borde sagital. Es el más ancho y largo de todos los bordes, cabe resaltar que el hueso parietal tiene cuatro bordes. Los dos huesos parietales se articulan entre sí mediante los bordes sagitales para formar la sutura sagital [85].

En la (Figura 3.20) se representa dicha medición.

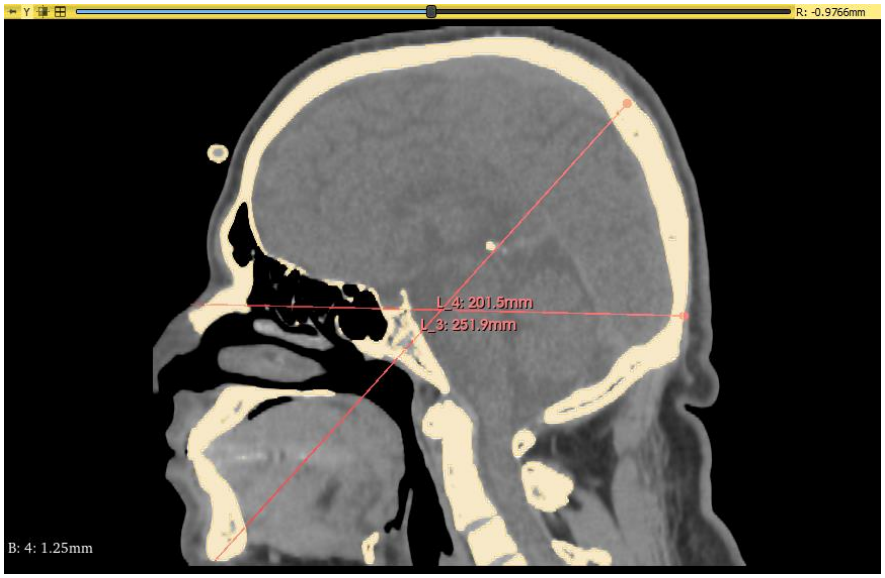


Figura 3.20: Medición plano sagital.

La medición del cráneo tridimensional queda de la siguiente forma (ver Figura 3.21): la altura del cráneo es de 251.9mm, el ancho corresponde a 125.4mm y de profundidad 201.5mm.

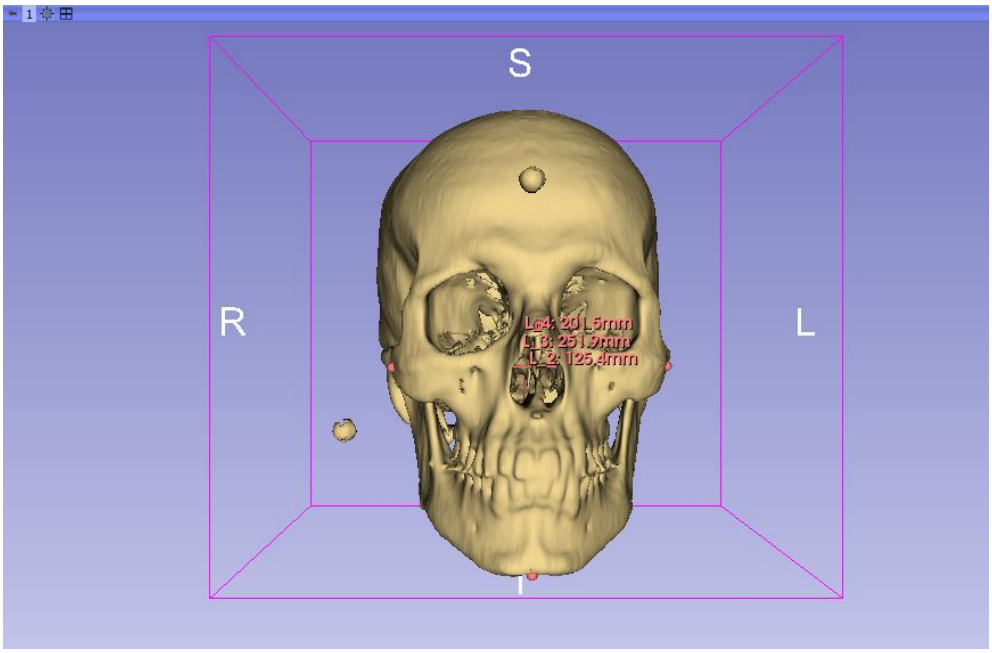


Figura 3.21: Resultado medición cráneo 3D..

Escalado con Blender.

Al tener las medidas de referencia se puede aplicar el escalamiento utilizando Blender, agregando las dimensiones en los ejes correspondientes.

El objetivo de utilizar Blender, además de hacer el escalamiento, es también adecuar las unidades de medida dado que el cráneo virtual que proviene de 3D *Slicer* está en milímetros (mm) y la que se necesita posteriormente en Unity es en metros (m). Blender facilita el trabajo porque primero su unidad de medida es en metros (m) y además se puede exportar en un formato reconocible para Unity como es .OBJ.

Blender cuenta con atajos que son muy importantes pues las vastas herramientas que tiene el programa los hace necesarios. Para ingresar a la pestaña **transformación** como lo muestra la (Figura 3.22), solo se da clic en la pestaña objeto, donde se desplegará un menú, la opción que se requiere está de primero. O la alternativa de atajo es oprimir la tecla S para ir directo a la pestaña de escala o T a la de transformación.

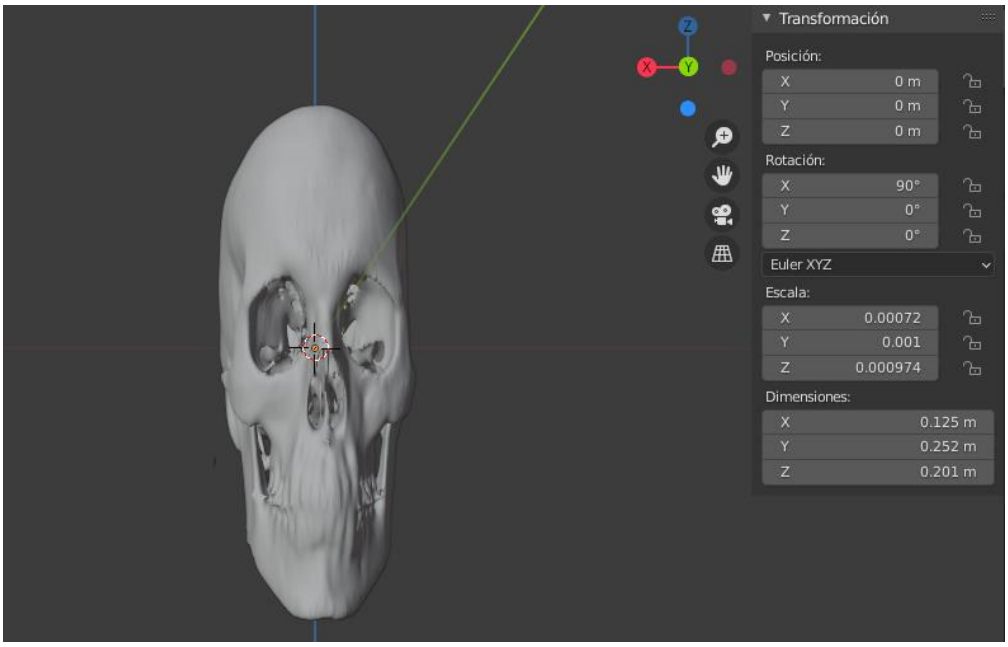
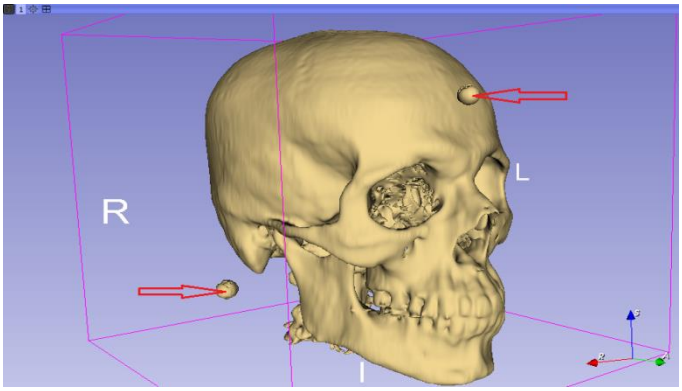


Figura 3.22: Escalado cráneo 3D Blender.

Estando el escalado completo el paso a seguir es refinar el objeto 3D, en donde se eliminan partes que quedaron al hacer la segmentación. Estas ciertas deformaciones no son propias del cráneo y no son relevantes como las que se observan en 3D *Slicer*, en la Figura 3.23 se muestran con una flecha roja. Utilizando el modo esculpir se puede modificar la forma de un modelo, su flujo de trabajo no trata con elementos individuales (vértices, bordes, caras) sino que manipula la geometría de la región de influencia, en este caso el pincel [86]. El proceso de esculpido que se realiza es el de raspar.



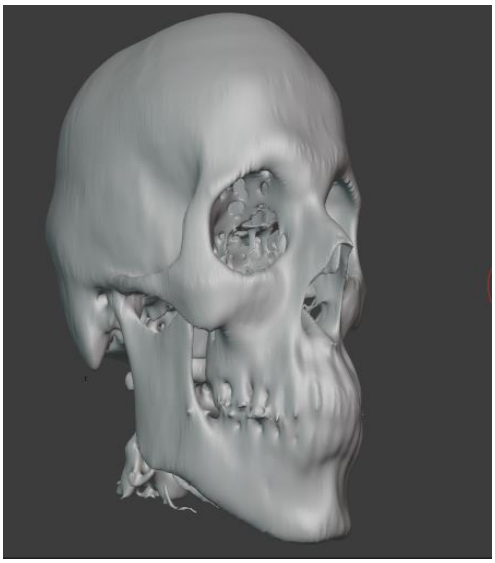


Figura 3.23: Comparación cráneo 3D Slicer y Blender.

3.3.3. Incorporar el cráneo digital a Unity

Unity permite cargar una infinidad de archivos que pueden ser efectos de sonido, animaciones, videos, imágenes, y por supuesto modelos en BD en formatos *fbx* y *obj*. También archivos propios de aplicaciones 3D como Blender en su formato *Blend* o 3D Studio Max, en formato Max, entre los más conocidos.

Para incorporar el cráneo digital en la escena de Unity se tiene dos alternativas, las más intuitiva es arrastrar el archivo *.obj* que contiene el cráneo digital en la sección Project—*Assets* --- *Object*. El resultado debe ser como se muestra en la (Figura 3.24). Ya teniendo el objeto en el proyecto este se debe colocar en la jerarquía, para que se visualice en la escena (Figura 3.25).

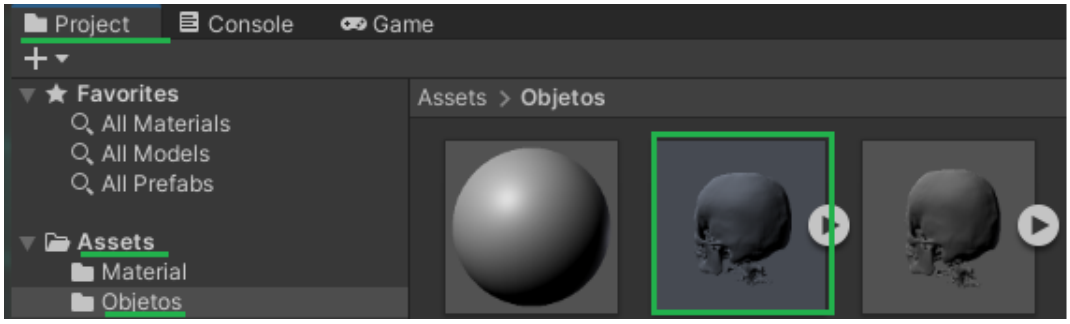


Figura 3.24: Carga del cráneo digital a Unity.

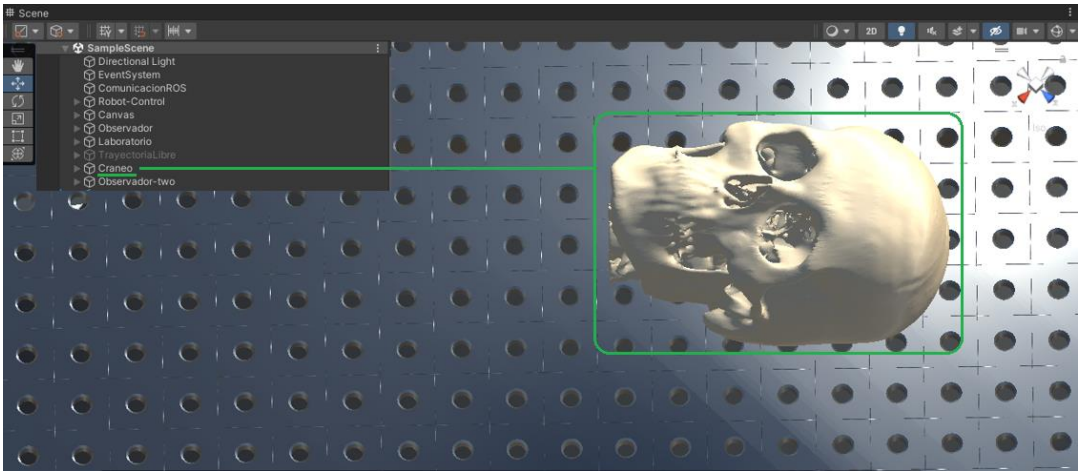


Figura 3.25: Carga del cráneo digital en la escena.

3.4. *Phantom* de cráneo.

El *phantom* de cráneo se obtuvo de la impresión 3D de un modelo reconstruido a través de los s*lices* de una tomografía computarizada de un paciente real, por medio del software Slicer3D. Cabe destacar que el resultado final, el *phantom*, va a presentar diferencias considerables en comparación al material original, ya que este se ha obtenido por medio del procesamiento de datos en diferentes etapas y bajo diferentes programas de computadora. Desde la misma técnica de adquisición de datos (tomografía computarizada), hasta la conversión a diferentes formatos (formato de imagen DICOM, formato 3D OBJ, formato de impresión 3D STL), siempre habrá imprecisiones (*noise data*) que afectarán la calidad de la impresión [87]. Sin embargo, el error de mayor impacto es el correspondiente a la etapa de generación del modelo 3D en *Slicer*, ya que se pueden inducir errores por el mismo manejo de la herramienta en el proceso de segmentación, debido a que es una tarea manual. Además de los errores inherentes al manejo del software, se deben considerar los correspondientes al hardware, ya que en la etapa de impresión del *phantom*, la calidad del material, la resolución de la impresora (impresoras convencionales manejan una resolución de 0.01mm a 0.5mm), el proceso de deformación y compresión en el enfriamiento del material, son factores que se deben tener en cuenta también. Sin embargo, estos errores, son irrelevantes teniendo en cuenta que no son el foco de este estudio.

Antes de proceder a explicar las características del *phantom* y la anatomía del cráneo humano, es importante explicar el concepto de ejes y planos anatómicos. Los planos anatómicos son planos imaginarios o superficies planas que cortan o seccionan al cuerpo en su posición anatómica [88]; mientras que los ejes anatómicos son las líneas imaginarias sobre las que se desplazan dichos planos. Tanto los ejes como los planos son ortogonales entre sí. La importancia de esta terminología radica en que permite una comunicación precisa entre profesionales de la salud, especialmente en el área de imágenes médicas.

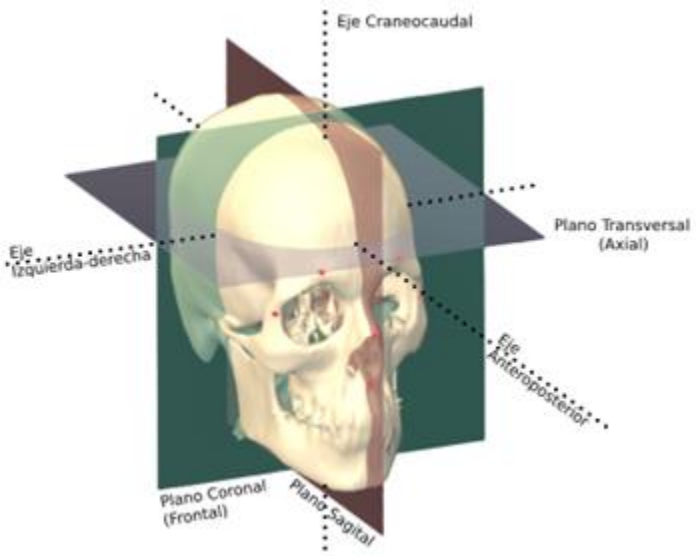


Figura 3.26: Ejes y planos anatómicos.

Las características básicas del *phantom*, como medidas o volumen, se describen a continuación. Cada medida puede ser verificada por separado en las imágenes de las (Figuras 3.27,3.28, 3.29), ya que en estas se representan dos caras de un mismo plano anatómico. La forma en que se logra esto es proyectando una línea recta desde el punto anatómico más sobresaliente del *phantom*, hasta la escuadra que se ubica en el borde de la figura.

Las medidas del *phantom* son:

- Altura: 25.1 cm.
- Ancho: 12.5 cm.
- Profundidad: 20.1 cm.

Gracias a las medidas descritas anteriormente, se puede calcular un volumen en forma de paralelepípedo, de $6306.3\ cm^2$.

Como se puede observar las medidas coinciden con aquellas descritas por el software Slicer3D; por lo que de ahora en adelante cada programa de computadora que manipule de alguna forma el objeto virtual del cráneo debe ajustarse para manejar estas mismas dimensiones.

Figura 3.27: Vista anterior y posterior del *phantom.*

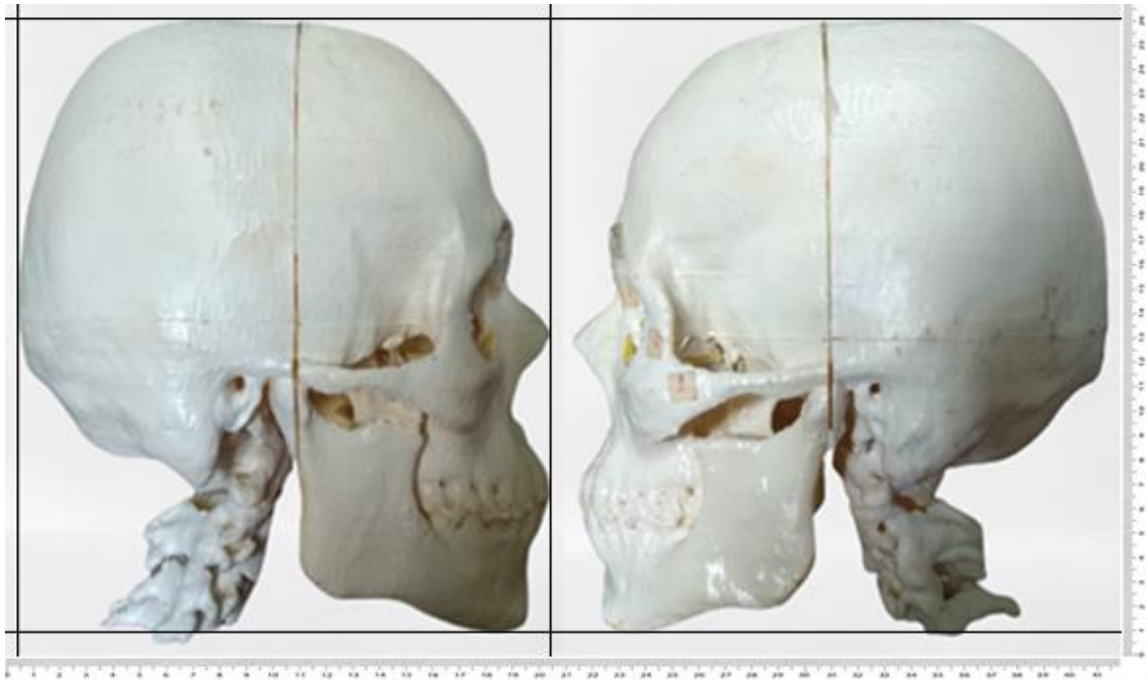


Figura 3.28: Vista lateral derecha y lateral izquierda del *phantom.*

Figura 3.29: Vista superior e inferior del *phantom*.

3.4.1. **Anatomía del cráneo humano**

Con el fin de ubicar los puntos anatómicos más fácilmente en este trabajo, se realizaron diferentes mapas anatómicos del cráneo humano, usando como modelo base el mismo *phantom*. La ubicación, tamaño y forma de los huesos, o regiones demarcadas, es aproximado y solo se tomará como referencia.

El cráneo está formado por entre 22 a 29 huesos, unidos por medio de suturas (Figuras 3.33, 3.34). El cráneo se divide en dos partes principales: el neurocráneo y el viscerocráneo (Figuras 3.30, 3.31). El neurocráneo como su nombre lo indica, contiene al cerebro, mientras que el viscerocráneo o esqueleto facial, contiene los tejidos blandos del rostro. Además, el neurocráneo o cavidad craneal, se puede dividir en la calvaría y la base del cráneo (Figura 3.32).

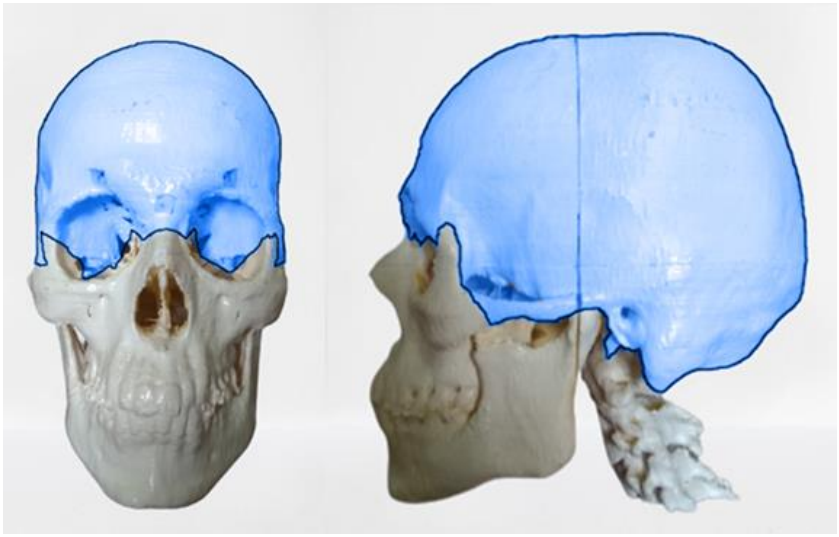


Figura 3.30: Neurocráneo.

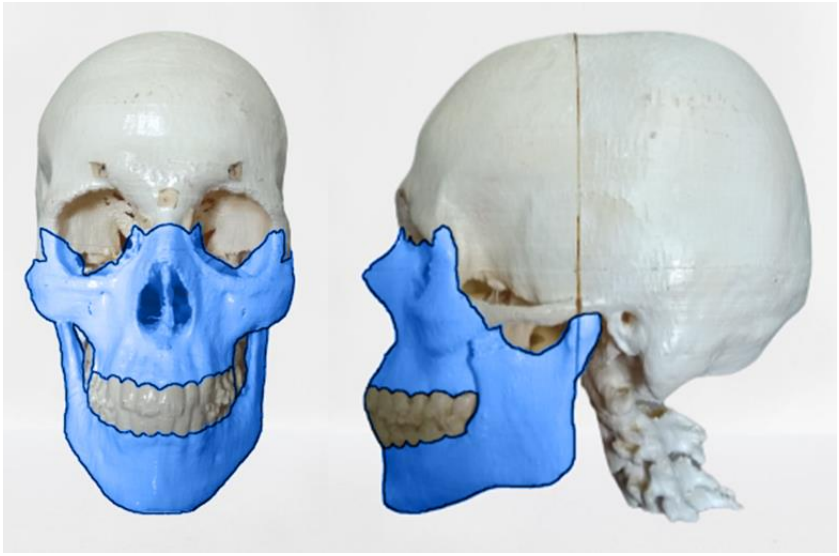


Figura 3.31: Viscerocráneo.

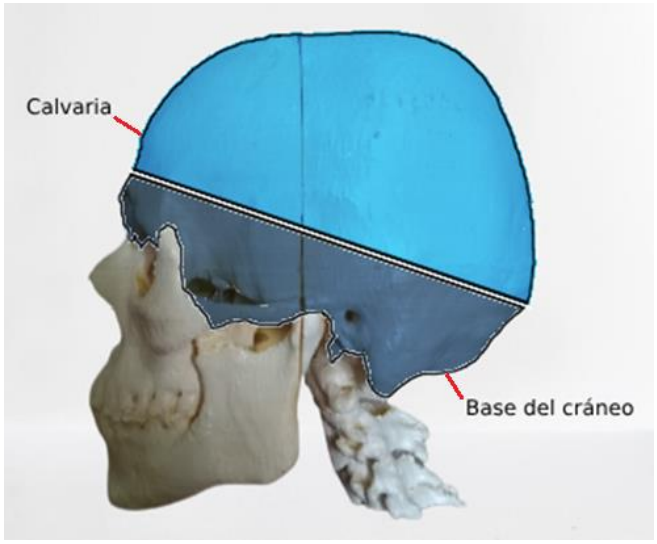


Figura 3.32: Cavidad craneal.

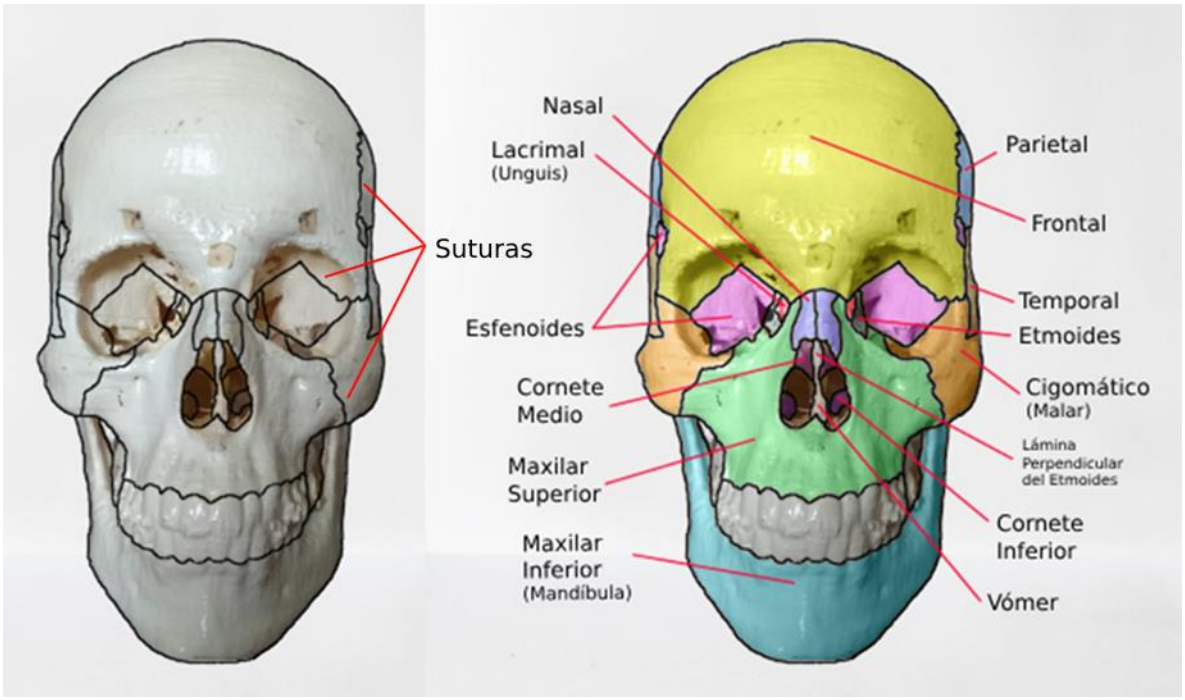


Figura 3 33: Vista anterior de los huesos del cráneo.

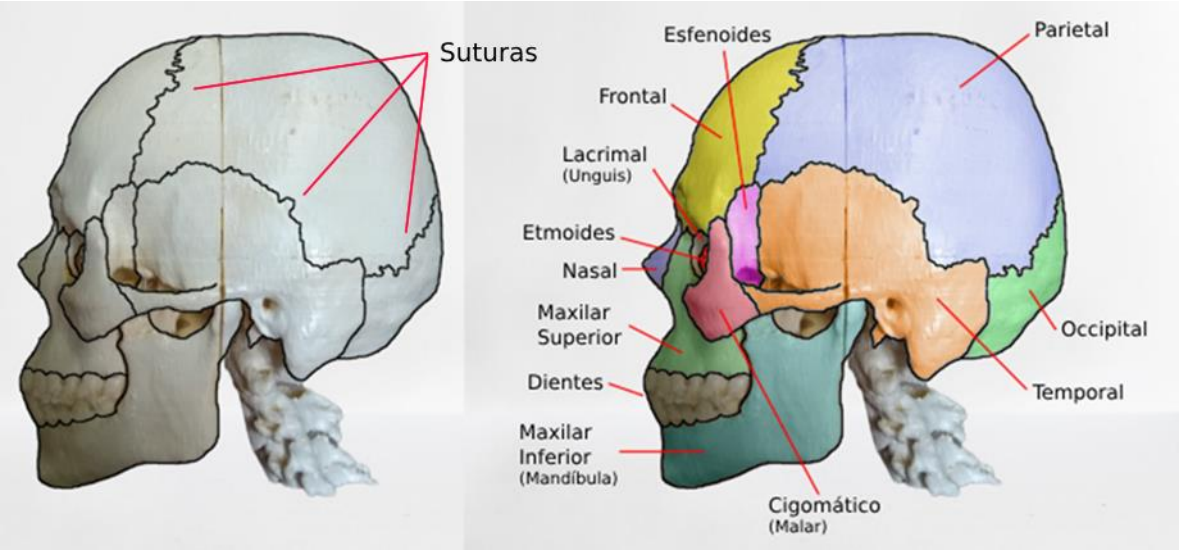


Figura 3.34: Vista lateral de los huesos del cráneo.

3.4.2. **Punto de acceso a la base del cráneo.**

Las cirugías de la base del cráneo pueden ser abordadas desde dos técnicas quirúrgicas principalmente: la cirugía endoscópica endonasal o la craneotomía. En el primero se interviene al paciente a través de las fosas nasales, mientras que en el segundo se realiza una apertura del cráneo directamente [89]. Dentro de las cirugías endoscópicas endonasales, existe un tipo particular de intervención: la cirugía transesfenoidal transnasal, cuyo objetivo principal es crear un acceso directo al área antero-medial del encéfalo, donde se ubica la glándula pituitaria o hipófisis (Figura 3.35). Para este trabajo se tomó un enfoque transesfenoidal, razón por la que, como su nombre lo indica, la herramienta quirúrgica debe desplazarse a través del seno esfenoidal (Figura 3.36). Como el *phantom* posee las estructuras óseas de un cráneo real, el seno esfenoidal se encuentra presente en este también (Figura 3.37). Con el objetivo de recorrer una trayectoria transesfenoidal con la herramienta del robot, es necesario realizar un orificio en esta área, como en una intervención real. Cabe destacar que este orificio no está presente en el modelo virtual.

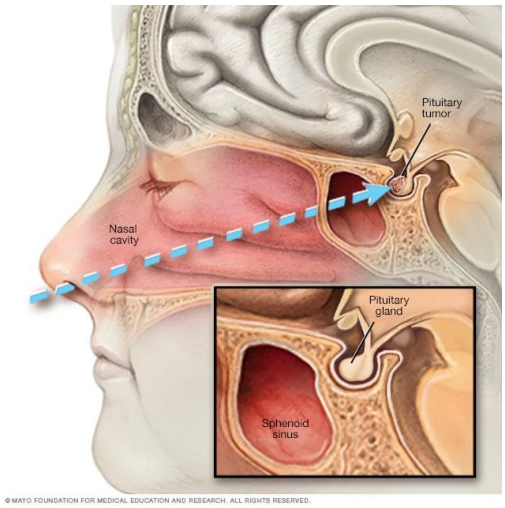


Figura 3.35: Áreas implicadas en una cirugía transesfenoidal transnasal [90].

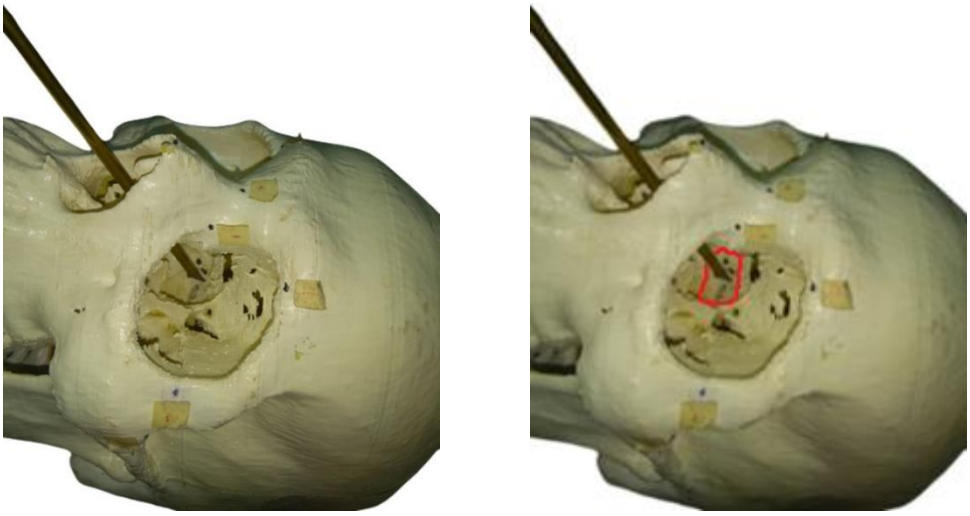


Figura 3.36: Pared anterior del seno esfenoidal.

Para realizar este orificio se utilizó un taladro ya que, aunque el *phantom* fue elaborado a partir de resinas plásticas, la profundidad del área a perforar dificultaba la tarea con otras herramientas. Con el fin de realizar diferentes trayectorias dentro de la cavidad craneal, el orificio debía ser lo suficientemente amplio como para permitir pivotear el endoscopio, pero no tanto como para afectar otras áreas del *phantom.*

Figura 3.37: Orificio del seno esfenoidal realizado.

Una vez realizado el orificio, se pudo verificar que el tamaño y la forma eran los adecuadas para poder realizar diferentes trayectorias desde cada una de las fosas nasales. Se utilizó una escala que se aproxima a las medidas reales del *phantom*, con el fin de poder calcular el área perforada en el seno esfenoidal. El área del orificio es aproximadamente de $195\ mm^2$. Como se mencionó anteriormente, el orificio se expandió deliberadamente por la naturaleza experimental de este trabajo, algo que no ocurriría en una intervención real, donde la destrucción de tejido debe ser mínima.

Por otro lado, si se considera una cirugía transorbital (cavidad ocular) para operar la glándula pituitaria, se debe tener en cuenta que, para el mismo orificio del seno esfenoidal, se presenta una reducción del área de aproximadamente 51% debido al ángulo de acceso. Aunque no se alcanzaron a realizar pruebas con trayectorias transorbitales, las figuras (3.38 y 3.39) muestran que el alcance y la maniobrabilidad sobre esta región se verían limitados, por lo que sería necesario perforar desde otro ángulo.

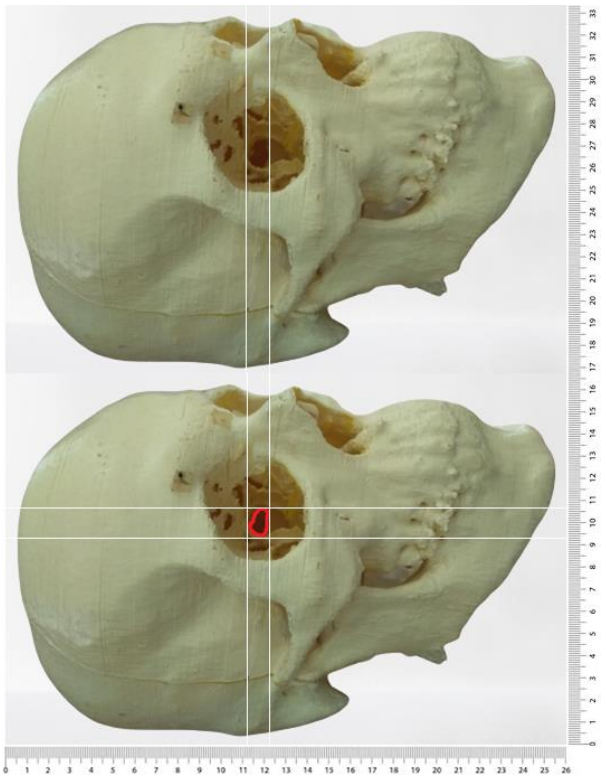


Figura 3.38: Orificio del seno esfenoidal desde la cavidad ocular derecha.

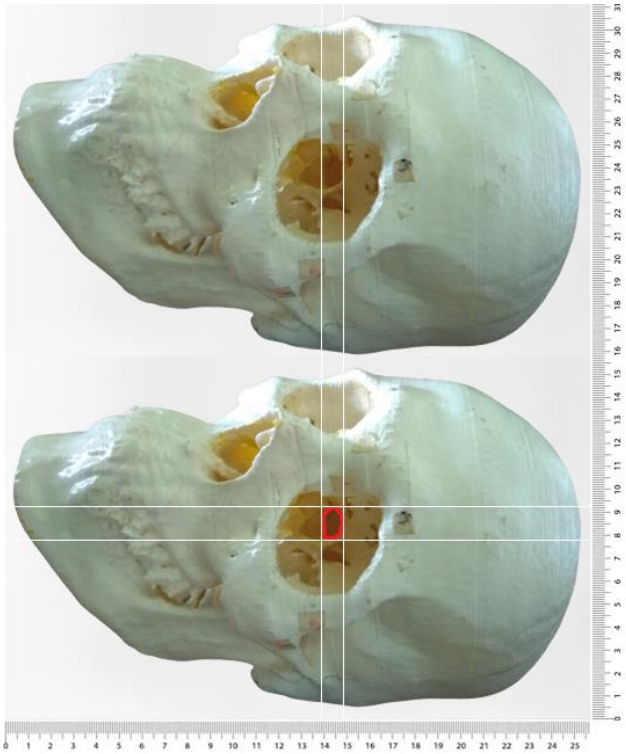


Figura 3.39: Orificio del seno esfenoidal desde la cavidad ocular izquierda.

3.4.3. Fijador *phantom* de cráneo.

Posicionamiento del paciente.

El posicionamiento del paciente es una parte integral de la planificación operativa [91], y más cuando se debe realizar un procedimiento quirúrgico. El posicionamiento dependerá de qué tipo de cirugía se vaya a realizar, en este caso un procedimiento neuroquirúrgico, específicamente un abordaje endonasal a un *phantom* de cráneo.

Teniendo esto en consideración, el paciente suele colocarse en posición decúbito supina, donde se encuentra tendido boca arriba en posición horizontal, con la espalda en contacto con la superficie y las extremidades inferiores extendidas [92], como lo indica la (Figura 3.40).

Figura 3.40: Posición decúbito supina.

Fijación *phantom* de cráneo.

La cabeza puede colocarse con una variedad de dispositivos de fijación dependiendo de la cirugía. Los dispositivos más comunes que se utilizan son la abrazadera de cabeza Mayfield Kess (Figura 3.41), el armazón de cabeza Sugita (Figura 3.41B), y el reposacabezas tipo herradura (Figura 3.41C) [93], cada dispositivo con un propósito particular.

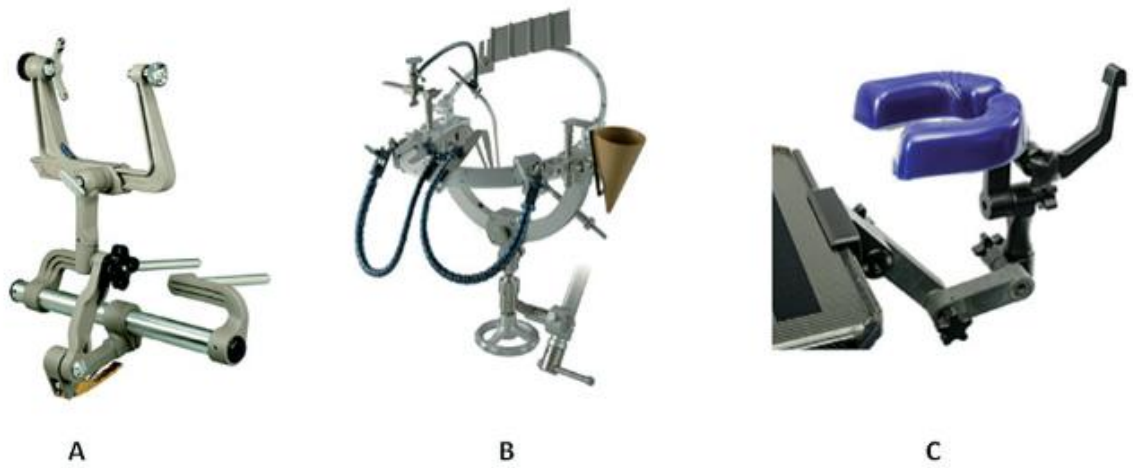


Figura 3.41: Dispositivos de fijación de la cabeza [93]

Con base en estos instrumentos de fijación se decidó elaborar un dispositivo en concreto que nos facilitara un posicionamiento supino adecuado del *phantom*. El dispositivo fue elaborado con piezas de perfiles V-slot 20X20 como se muestra en la (Figura 3.42). El dispositivo se compone de un mecanismo *quick release* que permite acoplar o desacoplar el *phantom*, además tiene un sistema de rotación y bloqueo que permite girar el *phantom* cada 5 grados.

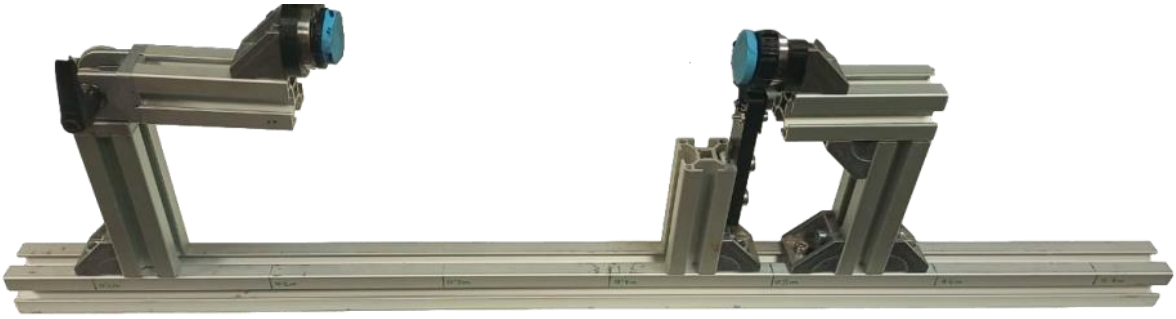


Figura 3.42: Fijador de *phantom* de cráneo.

La fijación del *phantom* de cráneo en el dispositivo diseñado se puede observar en la (Figura 3.43). La fijación se hace desde la parte inferior del cráneo, exactamente en las vértebras cervicales que sobresalen del *phantom*, hasta la parte superior del cráneo en el hueso parietal del mismo. El dispositivo ejerce un mínimo de presión para que el *phantom* quede estático en esa posición.

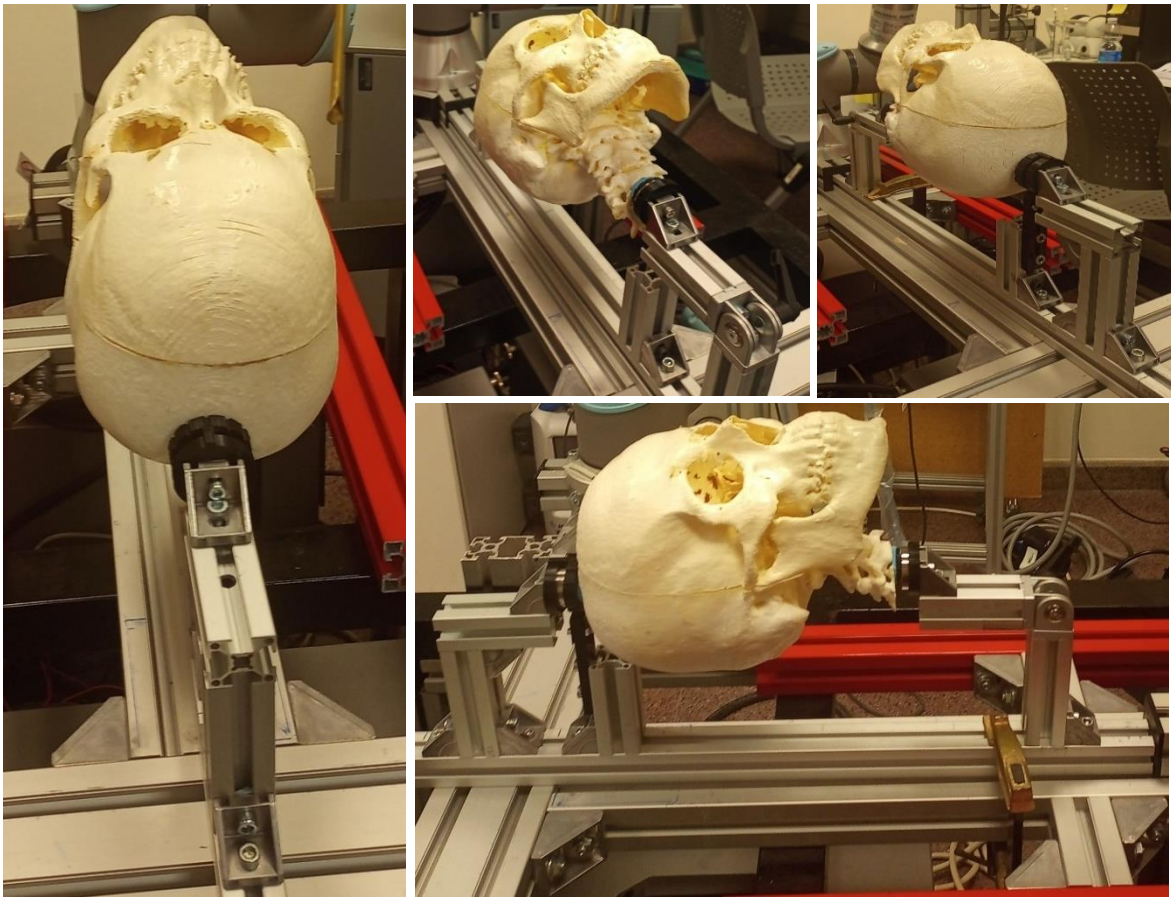


Figura 3.43: Fijación del *phantom* de cráneo en el dispositivo.

3.5. **Herramienta endoscópica.**

Para hablar del endoscopio primeramente se debe aclarar el concepto de endoscopia, técnica la cual permite la inspección, manipulación y posibles tratamientos a órganos internos mediante dispositivos que permiten una mejor visualización a distancia de los órganos objetivos. Esto sin necesidad de hacer incisiones grandes que afecten la integridad del paciente, de manera que se inserten en áreas donde los órganos huecos se conecten con el exterior a través de orificios naturales, como la uretra, la vagina, el recto, el canal auditivo, la garganta y la faringe [94], o por pequeñas incisiones.

El endoscopio convencional se compone principalmente de haces de fibra óptica o cámaras de chip CCD distales para transmitir una imagen óptica, todo empaquetado en

un tubo largo y flexible. Las cámaras están adheridas en la punta del instrumento, las cuales envían imágenes al monitor, lo cual permite ver a los médicos el interior del cuerpo. Un endoscopio de mayor diámetro también contiene dispositivos para dirigir sus puntas distales, y muchos permiten el despliegue de instrumentos para la manipulación quirúrgica y disección de tejido, como también los haces de fibra óptica pueden transmitir rayos láser que cortan, cauterizan o vaporizan tejido [95]. El endoscopio es controlado por el médico usando manubrios que mueven la punta del instrumento en diferentes direcciones [94], aplicando fuerzas en sus extremos proximales que están ubicados fuera del paciente.

Tipos de Endoscopios:

Los endoscopios se clasifican en 3 tipos: rígidos, flexibles y de cápsula.

Endoscopio rígido: Es un tubo rígido dotado de lentes cilíndricas y un cuerpo de acero inoxidable sellado por soldadura láser. El endoscopio también tiene una cámara y un canal de instrumentos para pasar herramientas a través del tubo [96]. Las principales características de este dispositivo son:

- Obtiene una mejor imagen.
- Alta resolución.
- Mejor transmisión de luz.
- Vista más amplia.

Los endoscopios rígidos se utilizan cuando se realizan inspecciones en áreas que no son muy sinuosas como el acceso endosanal a la base del cráneo, abordajes transfenoidal [97], conductos auditivos, y el aparato urinario [98].

Figura 3.44: Endoscopio rígido [96].

Endoscopio flexible: También se le denomina fibroscopio por estar compuestos por fibra óptica, su funcionamiento es idéntico a un endoscopio rígido, con la diferencia que suele ser más frágil al ser más delgado y flexible, características que lo hacen idóneos para trabajos en áreas sinuosas como al tránsito por la laringe y la columna vertebral.

Pese a su avanzada tecnología existe una mayor pérdida de luz que con los sistemas rígidos, lo que significa pobre luminosidad y baja calidad de imagen [99].

Figura 3.45: Endoscopio flexible [32].

Cápsula endoscópica: Es un procedimiento que involucra una capsula que debe ser ingerida, esta se impulsa a través del tracto gastrointestinal por peristalsis, sin necesidad de aplicar una fuerza de empuje que le permita recorrer el sistema digestivo [100]. La cápsula es bien pequeña, sus dimensiones son de (11-23 mm), carece de alambres externos, cables y haces de luz, pero tiene incorporado un cámara miniatura a color y una fuente de luz (Figura 3.46). La transmisión de video se realiza mediante telemetría en banda UHF a antenas que están adheridas al cuerpo, estas permiten la captura de imágenes.

Una de las características principales de este método es que no hay necesidad que el paciente esté recluido en un centro médico, puede seguir su vida normal mientras la capsula toma imágenes y video sin dolor, al intestino delgado [100].

Esta técnica ha cogido fuerza, ya que permite tomar imágenes de tramos que hasta el momento eran desconocidos e imposibles de captar. La principal utilidad de esta cápsula no reutilizable es el estudio del intestino delgado para poder determinar hemorragias de origen desconocido, anemia crónica, y enfermedades inflamatorias intestinales [101].

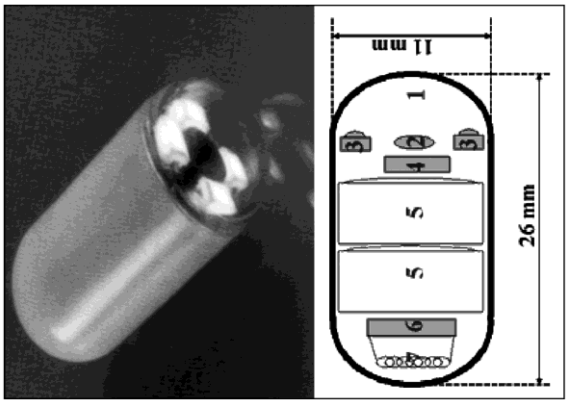


Figura 3.46: Estructura de la cápsula endoscópica. 1. Objetivo. 2. Lente. 3. Indicadores luminosos. 4. Captador de imágenes. 5. Baterías. 6. Transmisor ASIC. 7. Antena [101].

3.5.1. **Elaboración de la herramienta**

El diseño de la herramienta se realizó en *Fusión 360* (Figura 3.47), y consta de dos piezas: un conector (Figura 3.48 (a)), que se sujeta al efector final del robot mediante tornillos (permitiendo un acople más firme), y un cilindro recto, que se inserta en el conector. El conector (longitud total 53 mm) posee en la parte superior un cuello externo de 40 mm de altura (Figura 3.48 (b) con un agujero en el medio (Figura 3.48 (c)), donde se introduce el cilindro. El conector al ser una pieza más robusta, puede ser reproducido en una impresora 3D, mientras que el cilindro al tener una forma larga y delgada debe ser recreado a partir de otro material menos frágil.

Figura 3.47: Pieza de endoscopio diseñada en *Fusión 360*.

a) b) c)

Figura 3.48: Pieza que se ajusta al efector final del robot.

Por otro lado, el cilindro consiste en un tubo recto de aluminio de 19.34 cm de longitud (Figura 3.49), similar a un endoscopio rígido, que se inserta en el conector diseñado en *Fusion 360.* Esta medida se escogió con base a un catálogo de ventas de instrumentos quirúrgicos especializados en cirugías endonasales [102], donde las herramientas ópticas tienen una longitud media de entre 18 cm y 30 cm. La pieza cilíndrica debía ser lo suficientemente larga como para sobrepasar el seno esfenoidal, pero no tanto como para colisionar con la parte interna del hueso parietal (véase sección 3.4.1 y 3.4.2). Además, convenía tener en cuenta la distancia que se pierde al ensamblar esta pieza con el conector (2 cm aprox.), así como la distancia que se pierde al acoplar la herramienta al robot (6 mm aprox.), ya que el alcance de la herramienta se ve reducido. El conjunto ensamblado mide un total de 21.34 cm, mientras la longitud útil es de 16 cm *(*Figura 3.50).

Figura 3.49: Longitud del cilindro recto.

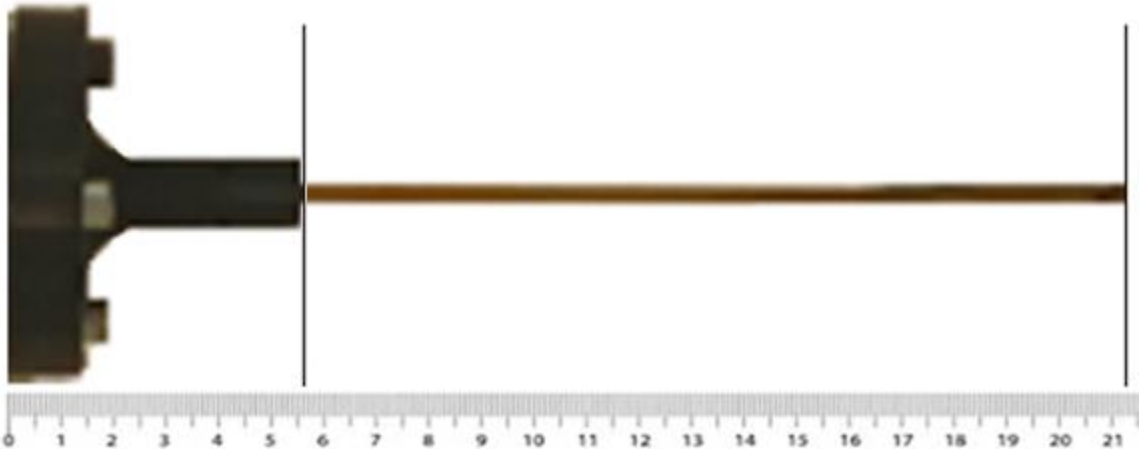


Figura 3.50: Longitud total más longitud útil.

En este trabajo se menciona frecuentemente la herramienta acoplada al robot como un endoscopio, ya que es un término adecuado por su función dentro de la aplicación, además del evidente parecido físico. El conector se llamará base de la herramienta o base del endoscopio.

3.5.2. **Modelo digital de la herramienta endoscópica.**

Para integrar el modelo 3D de la herramienta endoscópica a la plataforma, se debe utilizar el mismo software en el que se diseñó la pieza, para hacer la conversión de formato. La pieza diseñada en *Fusión 360* se puede exportar a formato URDF (*Universal Robotic Description Format*), que es un formato XML para representar un modelo de robot en un entorno de simulación [103]. El formato da una descripción del robot sobre sus propiedades dinámicas, cinemáticas y visuales, el archivo tiene una estructura tipo árbol, donde se alternan elementos de enlace y unión [104] (ver anexo A.2).

Fusión 360 no entrega propiamente un archivo URDF sino una versión simplificada del mismo en formato XACRO, lenguaje de macros XML que crea archivos más cortos y legibles. Para hacer la conversión al formato requerido seguimos las siguientes instrucciones:

1. La carpeta que genera *Fusión 360* con las piezas en formato XACRO se pega en la siguiente dirección *catkin_ws/src.*

2. Por terminal de Ubuntu se ubica la siguiente ruta: */home/catkin_ws/src/TCP_UR_endo_completo_1_description/urdf* y se ejecutan las líneas de comando como se indica en (Código 2.1). Para verificar si la conversión fue exitosa se debe ver cómo (Figura 3.51).

Código 2.1 Conversión URDF

```
$ cd /home/catkin_ws/src/TCP_UR_endo_completo_1_description/urdf
## conversión de .xacro to urdf
$ rosrun xacro xacro TCP_UR_endo_completo.xacro --inorder > TCP_UR_endo_completo.urdf
```

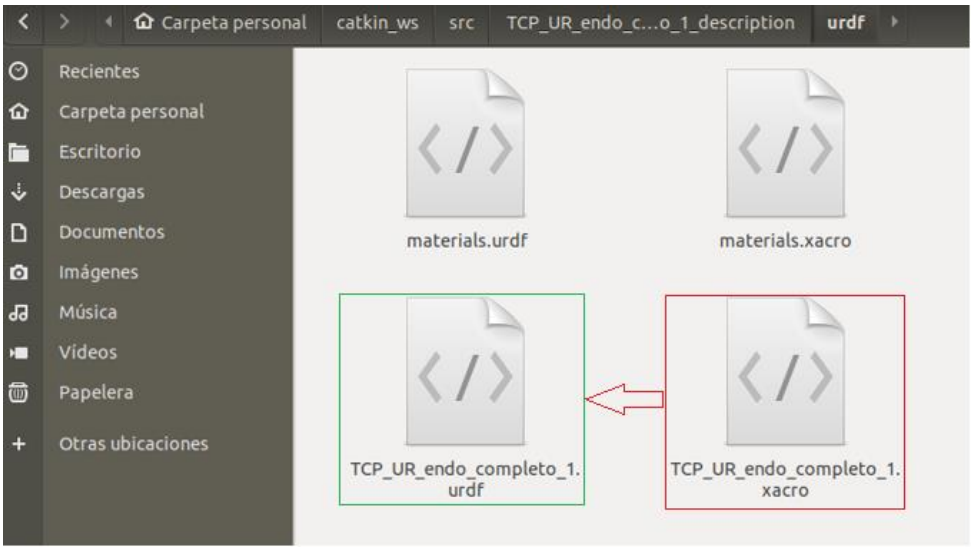
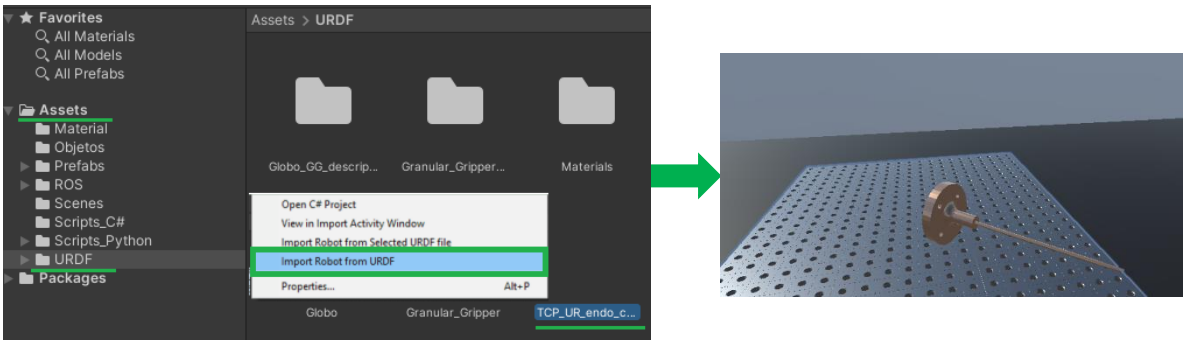


Figura 3.51: Verificación de conversión xacro a urdf.

Para finalizar la carpeta *TCP_UR_endo_completo_1_description y* el archivo que se obtuvo se pegan dentro de la carpeta *Assets* que genera Unity.

3.5.3. Cargar herramienta endoscópica a Unity.

Para incorporar la herramienta a *Unity* nos dirigimos al apartado de *Assets* como se indica en la (Figura 3.52(a)). Accedemos a URDF, se selecciona el archivo y se oprime click derecho para acceder a la opción *Import Robot from URDF,* una vez se haya dado click la herramienta debe mostrarse en la jerarquía del proyecto y en la escena (Figura 3.52(b)).



a) b)

Figura 3.52: Cargar herramienta a la escena en Unity.

Luego se debe ajustar la herramienta en dos secciones. Primero se reacomoda la pieza en la jerarquía del proyecto, luego se lo hace en la escena, se debe encajar la pieza en la última articulación del robot digital, procurando hacer un acople lo más preciso posible, como lo indica la (Figura 3.53).

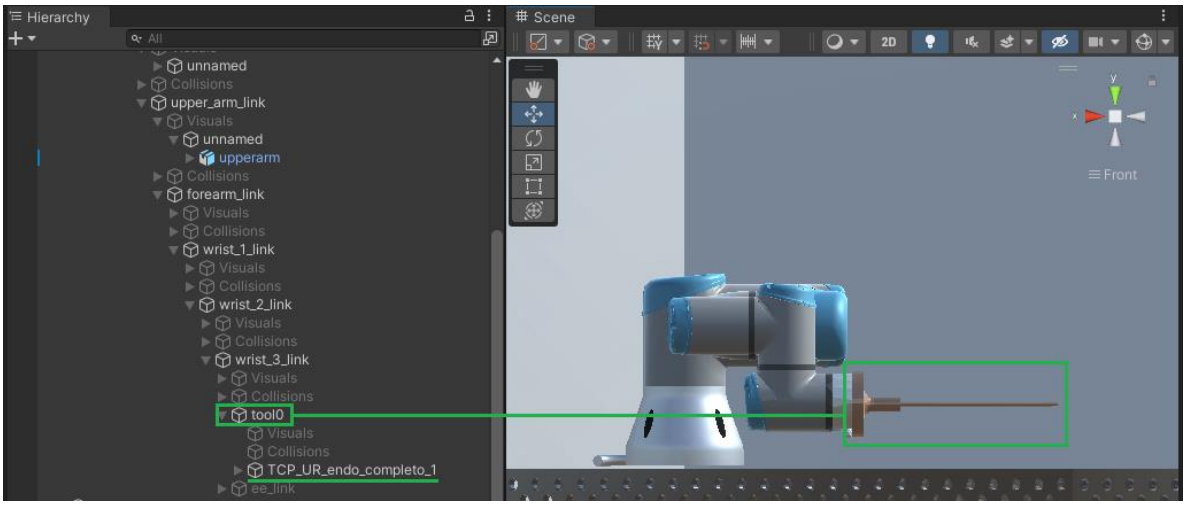


Figura 3.53: Acople de la herramienta al robot digital en Unity.

3.5.4. **Calibración de la herramienta.**

Antes de realizar el proceso de registro, es necesario primero llevar a cabo una tarea de suma importancia: calibrar o ajustar la herramienta; así como se debe realizar el registro del cráneo virtual y el *phantom*, de forma similar se debe hacer el registro del endoscopio virtual y el endoscopio real. Si no se realiza dicha tarea, el proceso de registro del cráneo tendrá errores de principio a fin, ya que la adecuada sincronización de la herramienta es lo que más adelante permitirá el emparejamiento de estos objetos. En este caso en particular se debe implementar esta tarea debido a que el modelo 3D del endoscopio (.urdf), es un objeto que se añade posteriormente al modelo (.urdf) del robot en Unity; es decir, no hace parte del diseño original donde tendría una articulación más precisa. El problema radica en que Unity, al no ser o no tener características de un software de diseño paramétrico, no permite ajustar la herramienta al eslabón final (tercera articulación de la muñeca) del robot de forma precisa. Esta es una operación básica en programas como *SolidWorks*, donde ensamblar dos componentes es muy sencillo usando la función “Relaciones de posición”, que permite crear relaciones geométricas que definen las direcciones permisibles del movimiento lineal o rotacional de las piezas. Entonces, es necesario lograr que las caras que entran en contacto, tanto de la base del endoscopio como del último eslabón sean coplanares, es decir, compartan un mismo plano; y que el endoscopio y el último eslabón sean concéntricos, es decir, compartan un mismo eje.

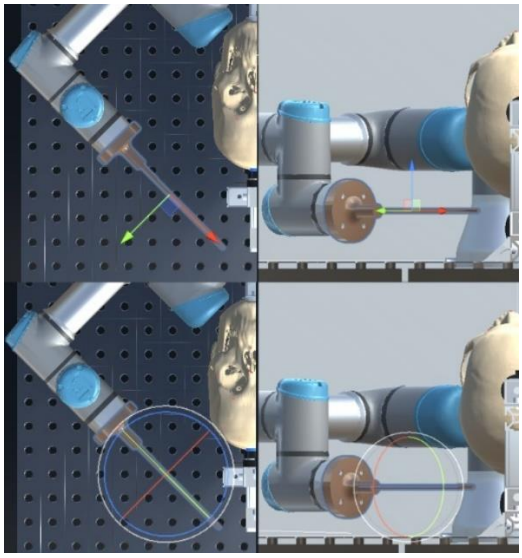
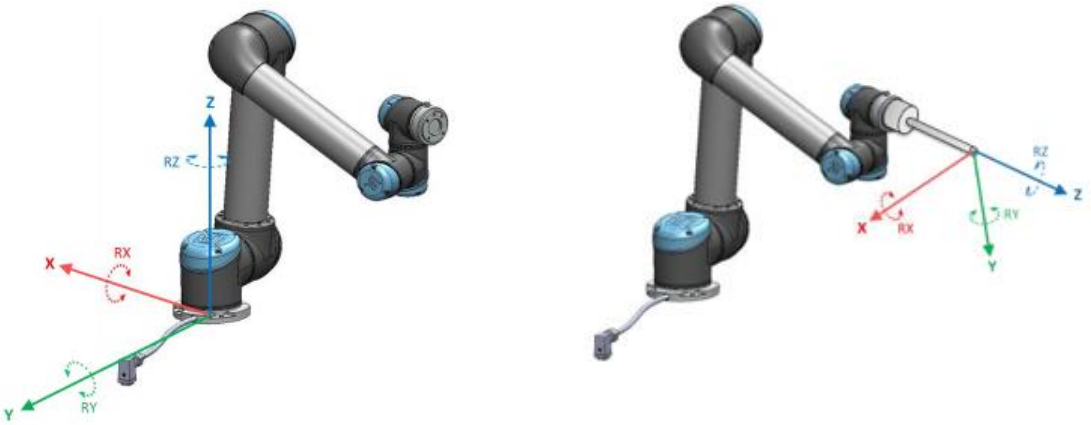


Figura 3.54: *Gizmos* de traslación y rotación.

La forma más rápida de ajustar la herramienta respecto al robot es por medio de los *gizmos* de traslación y rotación del objeto o, en otras palabras, posicionar este objeto de

forma manual. Sin embargo, el verdadero problema es lograr que el TCP de la herramienta en Unity y el TCP de la herramienta real estén ubicado a la misma distancia. Por supuesto, se trató de replicar las medidas de los modelos 3D en el laboratorio, pero existen variables del mundo real que terminarán afectando estas medidas. Por ejemplo, en el proceso de ensamblaje de las diferentes partes de la herramienta y del acople de la herramienta al robot (véase sección 3.5.1), se pierden milímetros de la longitud total de la pieza. Por lo anterior mencionado, es preciso utilizar el *teach pendant* del robot para, no sólo encontrar la distancia del TCP[1] de la herramienta real, sino para calibrar esta herramienta con su gemelo digital en Unity.



Sistema de referencia de la base (a) Sistema de referencia de la herramienta (b)

Figura 3.55: Sistemas de referencia del robot [105].

A fin de encontrar la coordenada en Z del nuevo TCP (Figura 3.55(b)) se utiliza únicamente el robot real. Para ello, se enciende el robot y al cargar la interfaz en el TP, en la pestaña "*Move*" se selecciona la opción "*Base*" en el menú "*Feature*", de modo que la posición y orientación del TCP sea calculada respecto a la base del robot (Figura 3.55(a)). Esta característica permite conocer de forma inmediata la altura o distancia sobre el eje Z de la punta de la herramienta al TCP original, utilizando una séptima fiducia o fiducia auxiliar. Cabe mencionar que solo es necesario calcular el offset en el eje Z ya que, al tratarse de una herramienta concéntrica al último eslabón del robot, no existe un offset en X e Y. Con este propósito en mente, se ubica el endoscopio en la superficie de los rieles

[1] *TCP: Tool Center Points*

sobre los que se ha fijado el robot (con el botón de movimiento libre del TP[2]) y se observa la distancia en Z en el TP (Figura 3.57). Es crucial que los rieles estén al mismo nivel que la base del robot y que la herramienta esté alineada sobre el eje Z (Figura 3.56), puesto que debe ser perpendicular a la superficie para hallar la altura exacta. Para mayor seguridad, se verifica este valor a un lado y al otro del robot (Figura 3.58).

Es oportuno indicar que el software *PolyScope* ofrece un método para hallar el TCP de cualquier herramienta, en casos donde se presenta un *offset* en alguno de los tres ejes XYZ. Este consiste en rotar manualmente (con el botón de movimiento libre del TP) la punta de la herramienta sobre un punto fijo en el espacio, de modo que la cinemática inversa del robot pueda calcular las coordenadas del nuevo TCP.

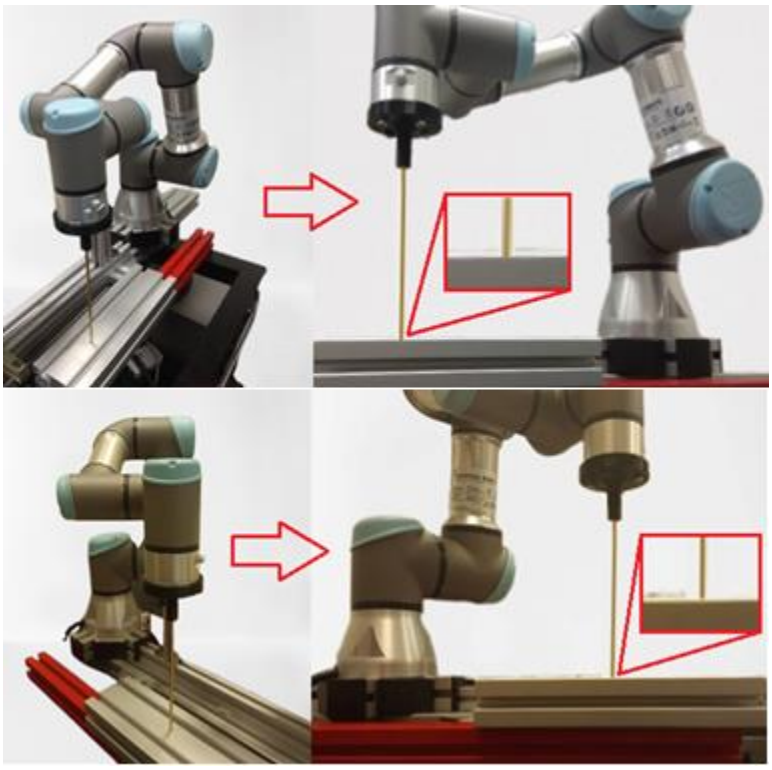


Figura 3.56: Séptima fiducia.

[2] *TP: Teach Pendant*

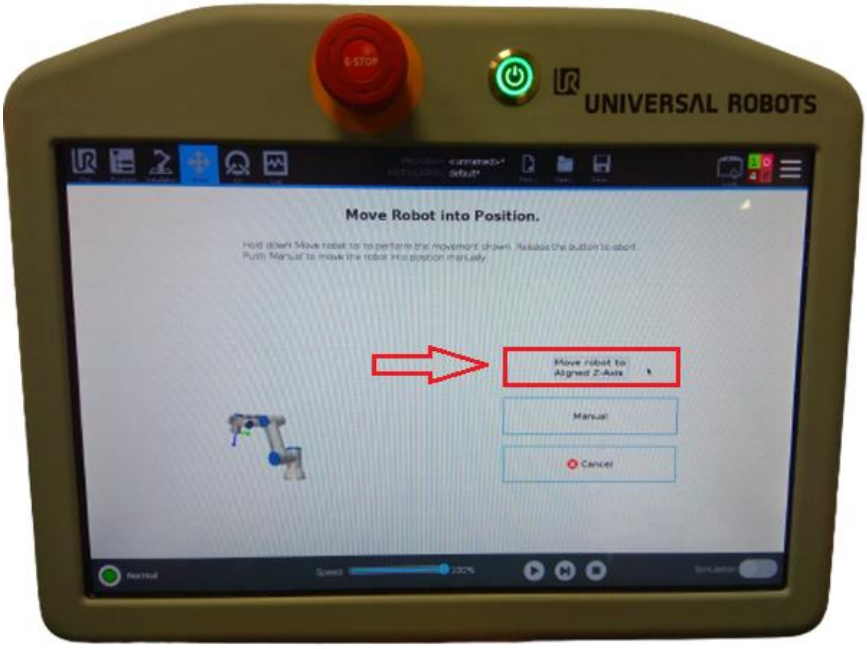
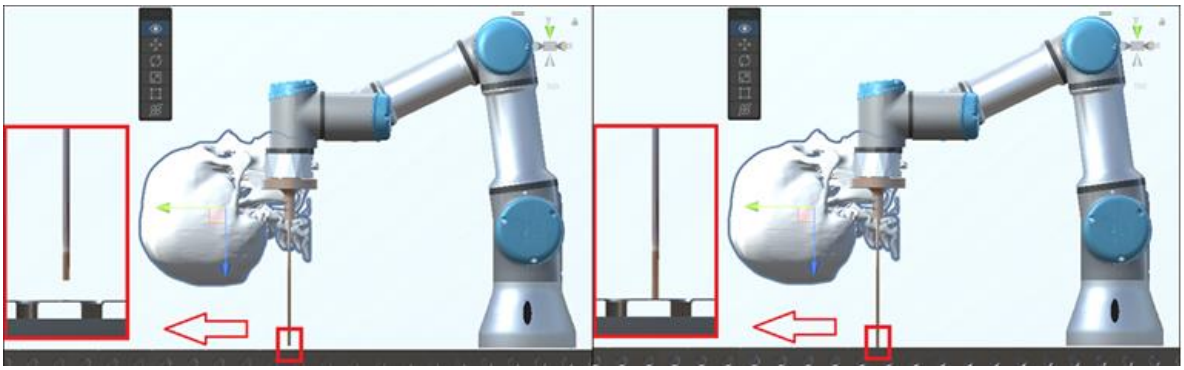


Figura 3.57: Alineación de TCP con el eje Z.

Figura 3.58: Coordenada en Z del nuevo TCP.

Una vez encontrada la coordenada en Z del nuevo TCP, se debe calibrar la herramienta en Unity con la herramienta real, de modo que coincidan visualmente. Primero, se establece conexión entre la aplicación y el robot real, de forma que el movimiento de uno se vea reflejado en el otro. Aunque la herramienta real está en contacto con la superficie

de los rieles (del paso anterior), es evidente que la herramienta en Unity no (Figura 3.59(a)). Esta distancia o *gap* que se aprecia en la pantalla es la distancia que se debe corregir. Para ello se utilizaron los *gizmos* de posición de la pieza para moverla en su eje X (Figura 3.59) y ajustar el TCP, hasta que finalmente coincida con la herramienta real (Figura 3.59(b)). Fuera de esto se debe alinear, en la medida de lo posible, el centro de la herramienta con el centro del último eslabón del robot para lograr concentricidad entre estos elementos; algo bastante difícil de lograr al tratarse de una tarea que debe realizarse de forma tentativa, como se explicó al principio de esta sección.



(a) *Gap* de la herramienta. (b) Herramienta calibrada.

Figura 3.59: Calibración del endoscopio en Unity.

Finalmente, como el TCP del robot tiene un sistema de coordenadas que puede ser modificado en posición u orientación en el *teach pendant* (Figura 3.60), se ingresa el valor hallado en la Figura 3.58, que corresponde al TCP de la herramienta acoplada al robot (204.7 mm en Z). Esta opción se encuentra en la pestaña "*Installation",* en el menú "*General"*, opción "*TCP*", recuadro "*Tool Center Point"*. Si el valor se ha ingresado correctamente, la representación visual del sistema de coordenadas del TCP debe cambiar. A continuación, para verificar que la distancia en Z del nuevo TCP es correcta, se vuelve a ubicar la herramienta real sobre la superficie de los rieles (con el botón de movimiento libre del TP), se alinea sobre el eje Z y se observa en la pestaña *"Move"*, en el recuadro "*Tool Position*", el valor en Z. El valor debe ser lo más cercano a 0 posible, como se observa en la Figura 3.61.

Figura 3.60: Distancia en Z del nuevo TCP.

Es importante al hacer uso de la plataforma no olvidarse de ingresar el valor en Z del TCP correspondiente a la herramienta, ya que cada vez que se envía una trayectoria desde la plataforma al controlador del robot, este calcula la cinemática inversa del robot con relación a la posición y orientación del TCP que se haya configurado.

Figura 3.61: Error de calibración del nuevo TCP.

Conviene señalar que la distancia del nuevo TCP es un parámetro que se pudo calibrar en Unity bastante bien, mientras que la concentricidad del eslabón final con el endoscopio puede estar sujeta a errores. Esta diferencia (*offset*) aunque no es apreciable a simple vista, puede tener un mínimo efecto en el emparejamiento de las fiducias.

3.6. **Cámaras en Unity 3D.**

Una cámara es un objeto especial que ayuda a ver cosas en el mundo del juego. Define lo que se puede ver y cómo se ve. El componente de la cámara también define el tamaño y la forma de las cosas que podemos ver a través del dispositivo [106]. Hay dos tipos de cámaras en Unity: perspectiva y ortogonal. Las cámaras de perspectiva hacen que los objetos parezcan más pequeños cuanto más lejos están del ángulo de visión, mientras que las cámaras ortográficas no reducen el tamaño de los objetos al aumentar la distancia [107].

Las características de las cámaras convencionales que Unity ofrece se quedan cortas en rendimiento y funcionalidad, ya que para usarlas se requiere una lógica de programación extensa y, en caso de realizar cambios, se debe hacer en el código. Como alternativa se procede a utilizar un complemento gratuito denominado *Cinemachine,* el cual cuenta con una amplia gama de variantes operativas.

Cinemachine:

Cinemachine es una colección de herramientas de cámara sofisticadas que están diseñadas para ser fáciles de usar sin necesidad de codificación. Estas herramientas son dinámicas e inteligentes y pueden analizar la composición e interacción de la escena, lo que resulta en crear tomas óptimas y visualmente atractivas [108].

En lugar de crear nuevas cámaras, *Cinemachine* dirige una sola cámara de Unity en la escena con el *script CinemachineBrain* adjunto. Este permite el monitoreo de las otras cámaras ficticias denominadas cámaras virtuales, que pueden ser infinitas dependiendo de las tomas que se dese realizar.

Cinemachine tiene una función muy importante, esta consiste en guardar los cambios realizados al hacer ajustes cuando se está en modo *play* (cuando el proyecto se está

ejecutando en el editor), ya que por defecto, el programador tiene que detener el modo de reproducción para realizar los ajustes que se proponen. Esta es una característica muy conveniente ya que permite al desarrollador realizar ajustes y modificaciones de las cámaras en tiempo de ejecución

Cinemachine proporciona comportamientos fundamentales como *Composer, Transposer, Free look camera, Cinemachine 2D, Target group* y combinaciones personalizadas. Los más relevantes para estos proyectos son:

Composer: El enfoque principal de *composer* está en la orientación por procedimientos y la rotación automática de la cámara, para garantizar que los objetos deseados se mantengan en cualquier posición especifica dentro de la pantalla.

Transposer: También conocida como cámara de seguimiento, es un elemento de la cámara que se puede adherir a cualquier objeto para que lo pueda seguir, la forma en que la cámara seguirá el objeto es totalmente configurable. Para utilizar este complemento es muy simple; primero se descarga usando el *Package Manager* que ofrece Unity, menú superior *Windows* < *Package Manager*, se puede buscar en el menú desplegable *all packages.*

Para configurar las cámaras con *Cinemachine*, se elige en el menú de Unity el complemento, este automáticamente te crea un *GameObject* que está compuesto por una *Cinemachine Virtual Camera.*

La *virtual camera* tiene propiedades importantes como el *status*, puede tener tres estados *live, disabled, standby.*

Live: Indica que la cámara virtual está controlando activamente la cámara de Unity, donde se tiene adjunto un *script* de *cinemachine Brain.*

Disable: Indica que la cámara virtual no está controlando la cámara de Unity, ni apunta a ninguno de los objetivos, lo que indica que no está utilizando la potencia de procesamiento, por lo que es una buena práctica deshabilitar cámaras que no están trabajando.

Standby: Indica que la cámara virtual no está controlando actualmente la cámara de Unity, sin embargo, está a la espera del cambio de prioridad para que este inicie el control de la cámara.

Otra propiedad importante que utiliza *Cinemachine Brain* para hacer el cambio de cámaras es la prioridad (*Priority).* Entre más alta sea la prioridad de la cámara virtual esta es la que se activa dentro de las demás cámaras virtuales, dejando a la de menor prioridad en *standby.*

Se tienen estas dos propiedades que son muy potentes: *follow* y *look at.* Cada una espera una referencia de un objeto de la escena, básicamente *follow* toma el control del componente *transform* de la posición de la cámara virtual y físicamente mueva la cámara dependiendo de su objetivo de referencia. En cuanto a *look at* hará lo mismo en el componente *transform* de la rotación de la cámara virtual.

También hay otros atributos que pertenecen a los lentes de la cámara virtual que son *fiel of view*, que determina el aumento o disminución del campo de visión, el *near* y *far clip plan*, que determinan el punto inicial y final respectivamente de lo que se supone se tiene que renderizar en el espacio del tamaño de la toma.

Dentro de la camara virtual existen tres propiedades que se encargan de la lógica de la camara denominadas *body, aim y nose*. Estas especifican cómo la camara virtual va a rastrear, seguir, apuntar y girar. La propiedad *body* especifica qué lógica algorítmica utilizará para moverse. Dentro de *body* hay seis algoritmos que sirven para mover la cámara y cada uno tiene sus características:

- *Transposer:* se mueve con relación al objetivo al que sigue.
- *Do nothing:* la cámara se mantiene estática.
- *Framing Transposer*: es similar a *transposer*, solo que fija un espacio en relación con la pantalla.
- *Orbital Transposer*: es similar a *transposer,* solo que permite a la entrada del usuario ajustar dinámicamente la cámara, es decir utiliza el zoom.
- *Tracked Dolly:* se le puede dar movimiento a la camara por medio de un camino predeterminado.
- *Hard Lock to Target:* utiliza la ubicación exacta del objetivo de seguimiento.

3.6.1. **Observador II.**

Se crea una entidad denominada "*observador two"* donde se incluyen las diferentes cámaras virtuales que enfocan un ángulo de visión diferente, este además tiene dos *scripts* (Figura 3.62). Uno se encarga de ampliar o disminuir el tamaño de la imagen, a esto se le denomina *zoom*, y el otro controla el cambio de cámara dependiendo qué enfoque se requiere con teclas que se configuran para tal fin.

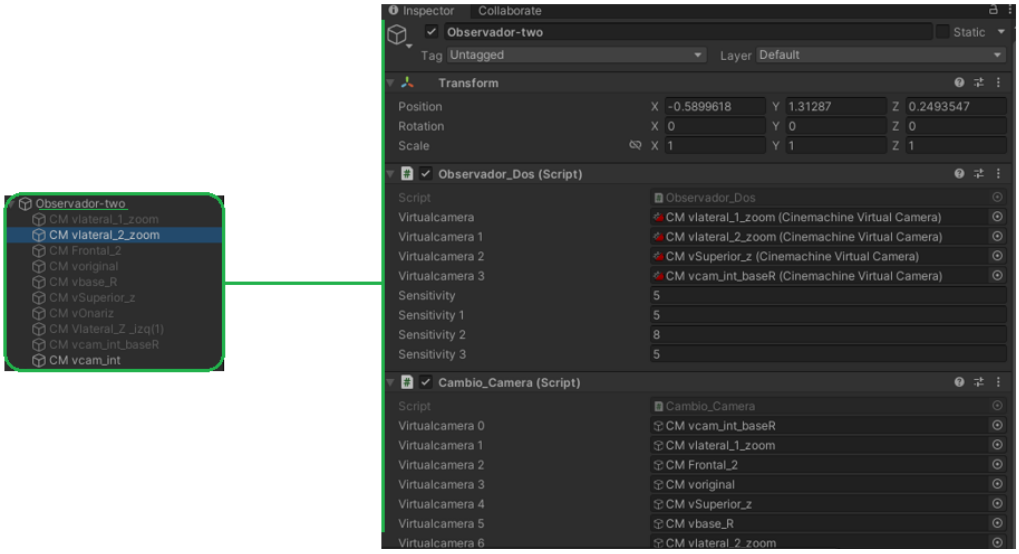


Figura 3.62: Observador Dos.

Además, se realizan modificaciones en las cámaras virtuales adicionando un *Transform* del objeto que se quiere seguir o ver en las propiedades de *follow* y *look at.* Adicionalmente, para que funcione correctamente el *zoom*, como se observa en la (Figura 3.63), se agrega a *body* un algoritmo de estilo de movimiento *Framing Transposer* y a *Aim* el algoritmo *composer*. Estos complementos acompañados de un *script*, permiten al usuario tener más movimiento en la plataforma virtual, aumentando o disminuyendo la imagen que enfoca esa cámara virtual al girar el *scroll* del *mouse*.

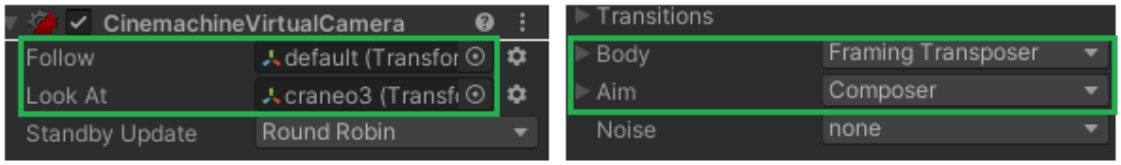


Figura 3.63: Configuración "*zoom*" de la cámara virtual.

Una de las partes más fundamentales de la cámara virtual son los planos de recorte denominados *Near* y *Far*; regulando sus valores numéricos se puede determinar dónde inicia y termina la vista de la cámara. Los planos de recorte cercano y lejano junto con los planos definidos por el campo de visión de la cámara describen lo que popularmente se conoce como cámara *frustum* [106]. La palabra *frustum* se refiere a una figura sólida que parece como una pirámide con la parte superior cortada paralela a la base (Figura 3.64). Esta es la forma de la región que puede ser vista y renderizada por una cámara de perspectiva [109].

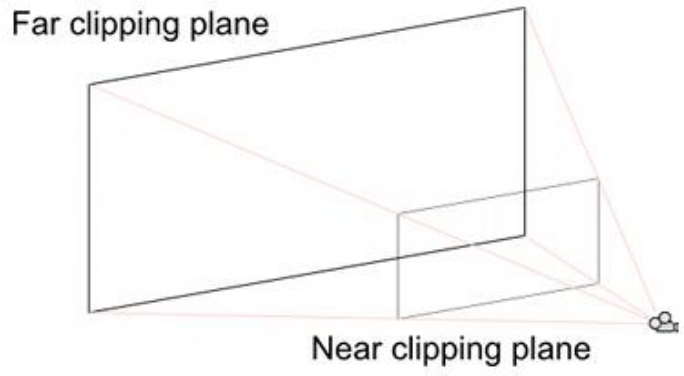


Figura 3.64: Representación del *Frustum* [109].

Teniendo en cuenta lo anterior se pueden realizar renderizados que faciliten la visualización de partes importantes del cráneo digital. En la (Figura 3.65) se puede observar una línea roja que pasa aproximadamente por la mitad de la figura digital del cráneo, esta se denomina *Near clipping plane,* es el inicio de lo que se va a visualizar.

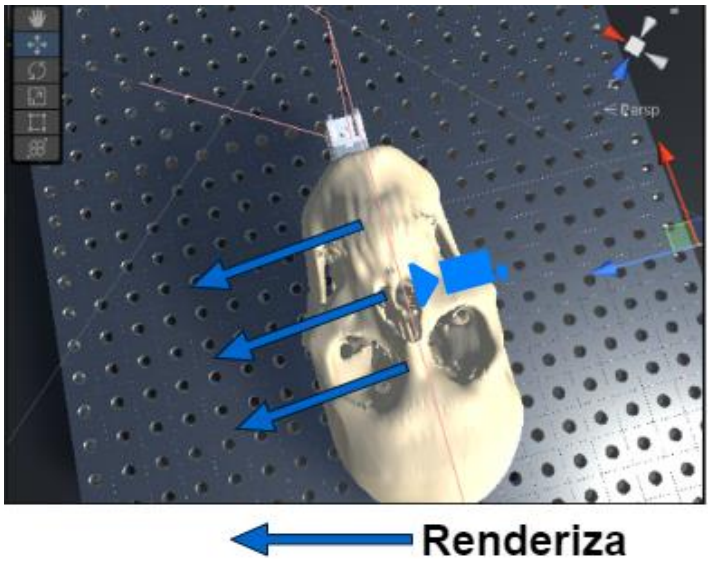


Figura 3.65: Representación del *Frustum*.

Cuando *Near clipping plane* se coloca en el centro del cráneo digital y apuntando en la dirección como se muestra en la (Figura 3.65), la vista que resulta es la (Figura 3.66), esta permite la observación de la estructura interna del cráneo. Lo que se realizó básicamente fue hacer un barrido de derecha a izquierda, no renderizando la parte de la figura que está antes de la línea *Near clipping plane.*

La secuencia inicia mostrando la figura (3.67), el usuario mueve el scroll del mouse, tratando de hacer un acercamiento utilizando el efecto *zoom* para llegar a la (Figura 3.66), haciendo un corte sagital del cráneo.

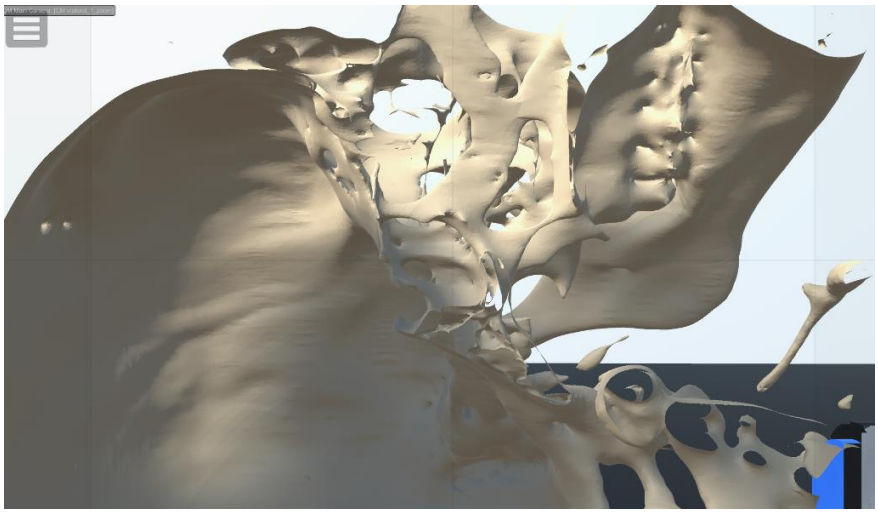


Figura 3.66: Corte sagital del cráneo virtual.

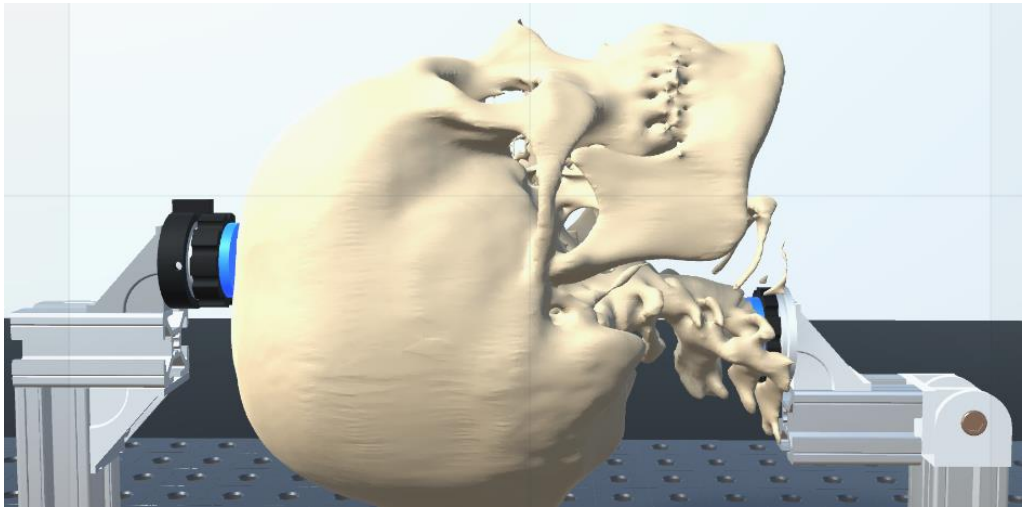


Figura 3.67: Visualización de la cámara virtual.

3.6.2. **Tipos de vista.**

Una de las modificaciones que se realizaron a la plataforma *Generador 3D de trayectorias libres de colisiones para un manipulador UR3e con pinza blanda* [110], fue el cambio en las perspectivas con diferentes cámaras virtuales, logrando una mejor interacción para el usuario en la plataforma, teniendo como marco de referencia principal el de cráneo digital. Los tipos de vista son: Superior/inferior, laterales, diagonal/anterior, posterior interna, endoscopio, vista original.

Superior/Inferior:

La vista superior enfoca el vértice o la parte más alta del cráneo virtual, teniendo en cuenta la dirección anatómica (Figura 3.68), la inferior hace lo contrario. (Figura 3.69).

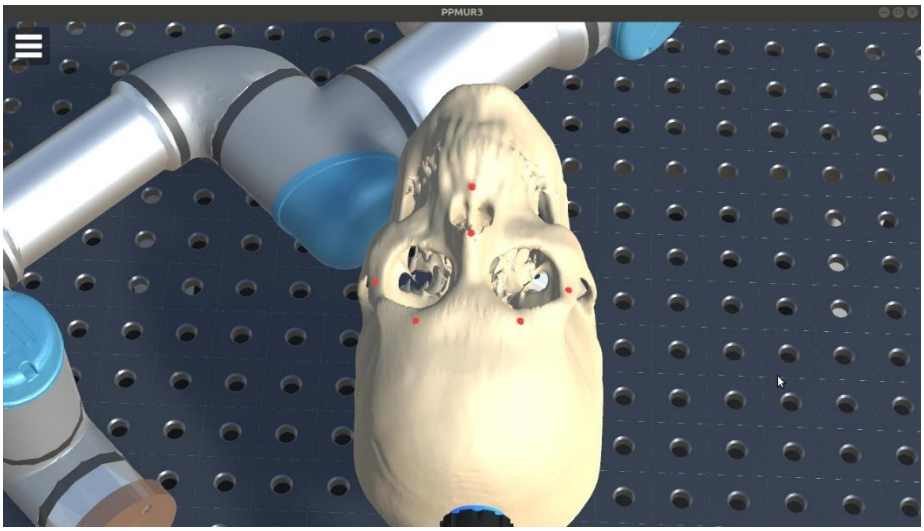


Figura 3.68: Vista superior.

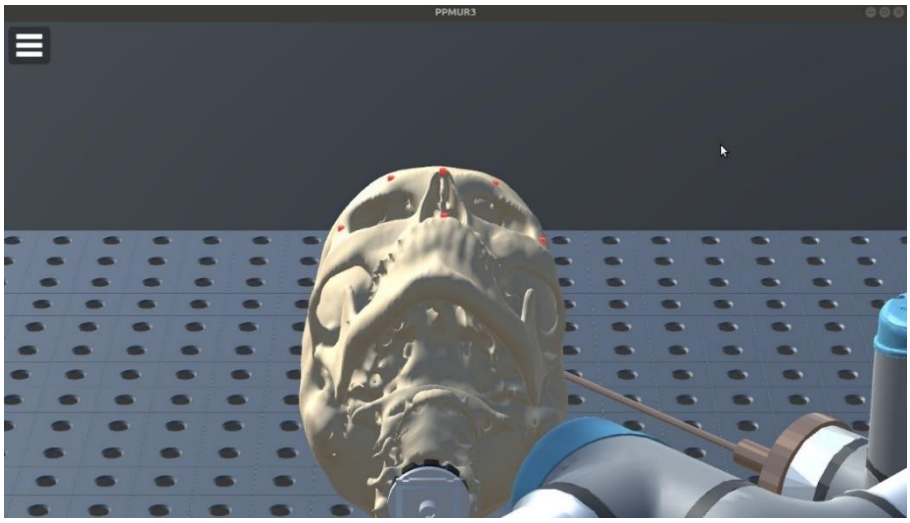


Figura 3.69: Vista inferior.

Laterales:

Esta vista entrega una perspectiva y un enfoque de profundidad. Al hacer un acercamiento de la cámara se realiza un corte de plano sagital como ya se ha mencionado anteriormente, (ver Figura 3.66) desde la derecha a izquierda del cráneo virtual (Figura 3.70).

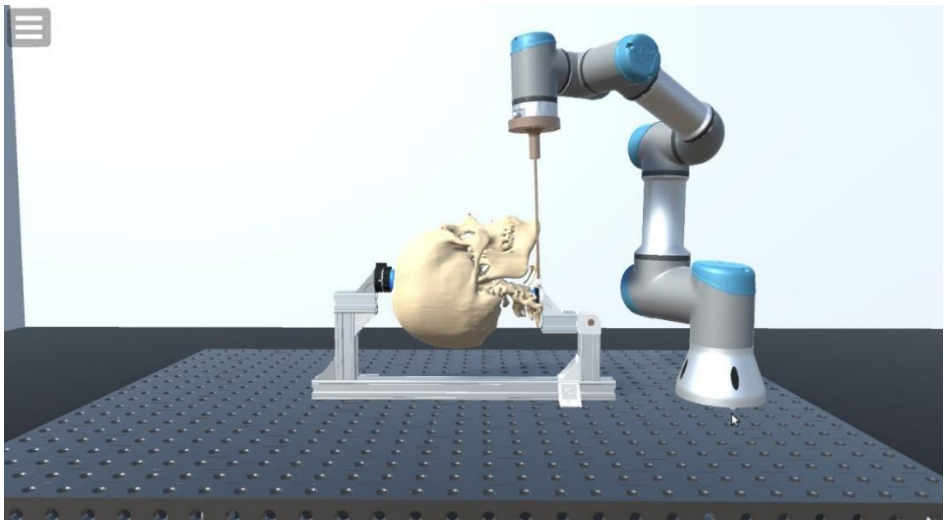


Figura 3.70: Vista lateral.

Diagonal/anterior:

La vista diagonal (Figura. 3.71) se enfoca en el plano interior de la órbita derecha del cráneo virtual, la vista anterior (Figura. 3.72), según la dirección anatómica del cráneo virtual, muestra una perspectiva que en el mundo de los videojuegos se conoce como vista superior, y muestra la escena desde arriba, como en este caso. Además ambas vistas tienen la característica de acercar y alejar (*zoom*), como también la vista anterior hace un corte de tipo coronal.

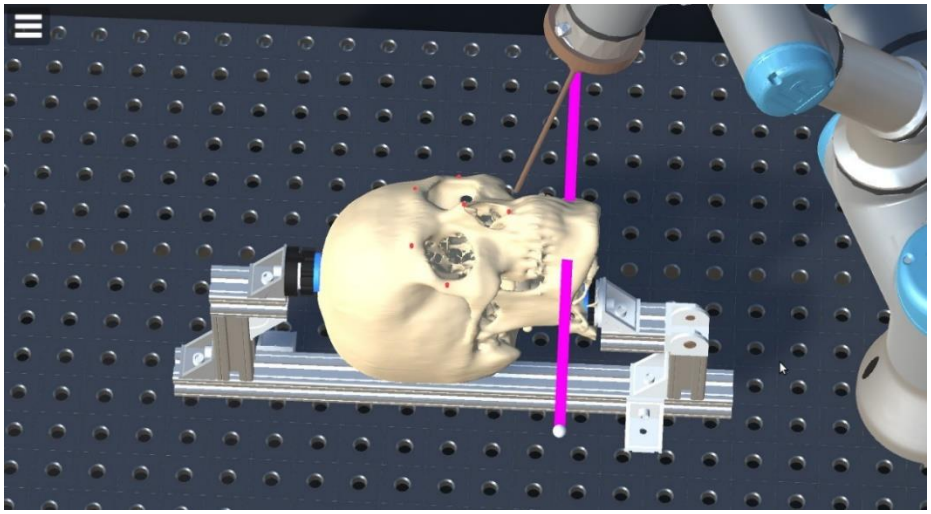


Figura 3.71: Vista diagonal.

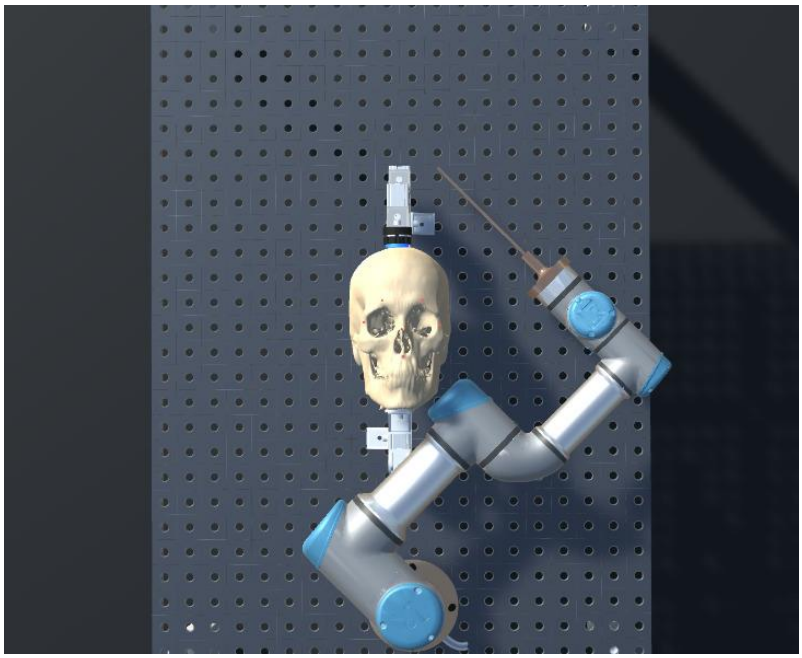


Figura 3.72: Vista anterior.

Posterior interna:

Es una vista fija (Figura. 3.73) que permite tener una perspectiva desde el interior del cráneo virtual. Hace un tipo de corte coronal con un leve ángulo de inclinación, para que permita renderizar partes importantes del mismo.

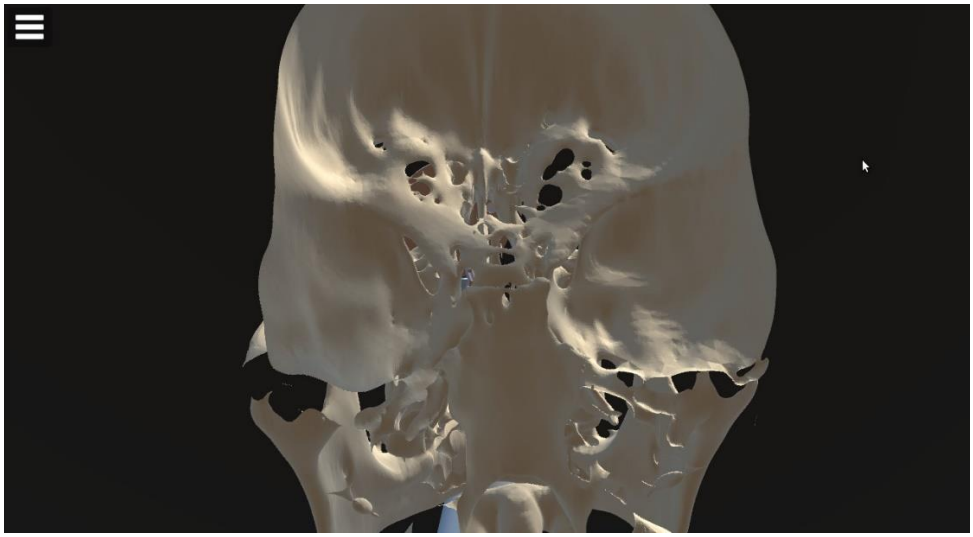


Figura 3.73: Vista posterior interna.

Vista endoscopio:

Es una cámara virtual adherida a la parte terminal de la herramienta, esta permite una perspectiva similar a la de un endoscopio, permite visualizar las trayectorias que hace el robot por las vías nasales (Figura 3.74).

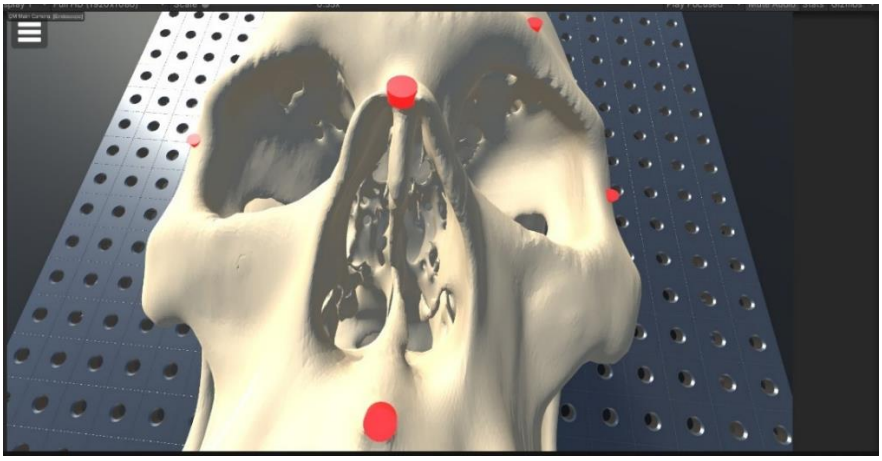


Figura 3.74: Vista endoscopio.

Vista original:

Ofrece una perspectiva completa del mundo virtual del que está compuesto la plataforma (Figura 3.75).

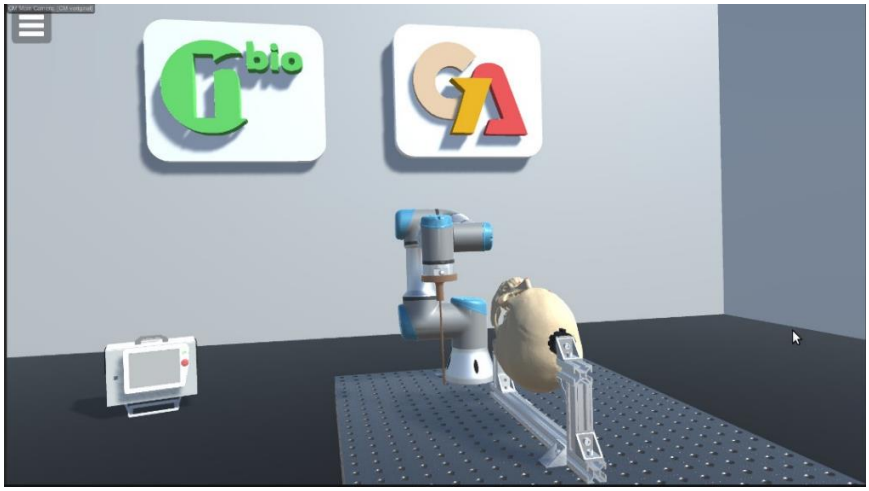


Figura 3.75: Vista original.

3.7. Registro manual.

3.7.1. Registro manual con *bounding box*.

En programas de modelado 3D como Blender, Cinema 4D, Maya, etc., de diseño paramétrico como Autocad, SolidWorks, Fusion 360, etc., motores de videojuegos como Unity, Unreal Engine, Godot, etc., o incluso en aplicaciones de procesamiento de imágenes, visión por computadora y robótica; se hace uso de una herramienta conocida como *bounding box*. En diseño 3D, el *bounding box* es una figura geométrica en forma de paralelepípedo que se utiliza para encerrar o rodear un objeto o grupo de objetos en el espacio tridimensional digital, con el fin de definir su ubicación o tamaño de modo que su manipulación sea más sencilla [111]. Las coordenadas de los vértices del paralelepípedo, que indican los valores máximos y mínimos en los ejes XYZ (los lados de esta figura siempre serán paralelos a los ejes de coordenadas XYZ), determinan sus dimensiones y, por lo tanto, su tamaño. En las siguientes figuras se puede apreciar el *bounding box* que encierra al cráneo virtual en el programa Slicer3D. Desde las vistas ortogonales solo se puede observar el paralelogramo que está más cerca de la cámara y no el que se ubica al fondo (las caras de esta figura se superponen); esto se debe a que la representación tridimensional se hace bidimensional y, por lo tanto, cada vista (anterior-posterior, superior-inferior, lateral derecha-lateral izquierda) será una proyección perpendicular de las otras dos.

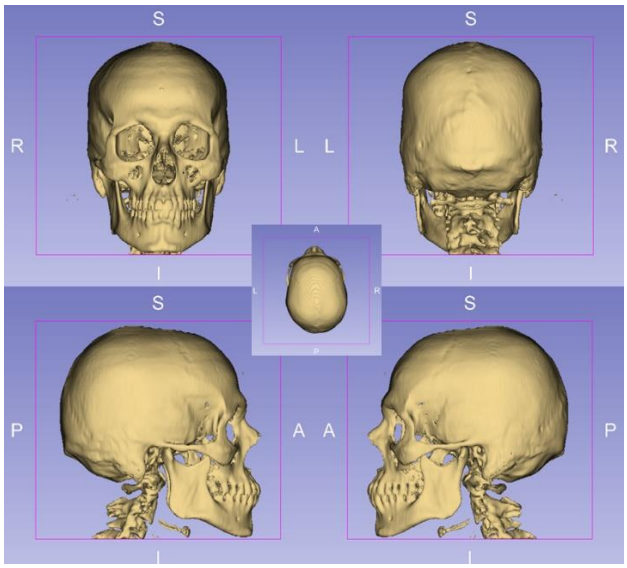


Figura 3.76: Vistas ortogonales (anterior-posterior, lateral d-i, superior) del modelo 3D.

La vista isométrica, a diferencia de la anterior, se caracteriza porque el objeto se muestra de forma tridimensional, permitiendo una visualización completa del paralelepípedo mencionado anteriormente. Cabe resaltar que si se compara la vista anterior en su proyección ortogonal e isométrica, la diferencia entre estas dos representaciones se hace bastante apreciable, ya que la perspectiva isométrica presenta mayor distorsión al tratar de representar una figura 3D en 2D.

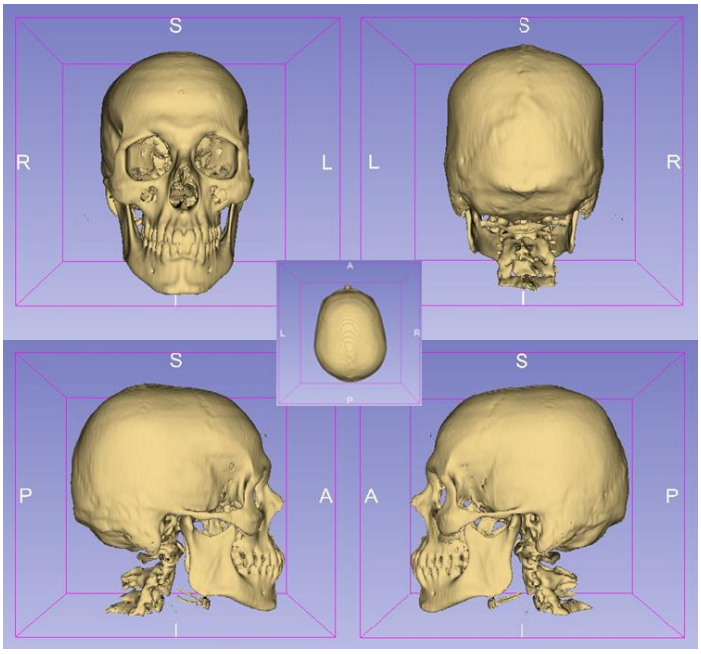


Figura 3.77: Vistas isométricas (anterior-posterior, lateral d-i, superior) del modelo 3D.

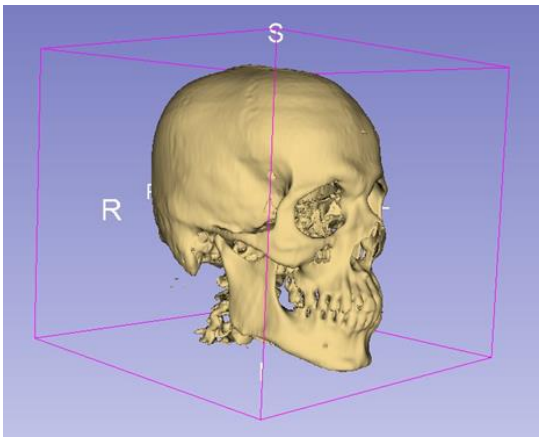


Figura 3.78: Vista isométrica diagonal del modelo 3D.

Para este caso particular, se utilizó el *bounding box* como método de registro, ya que cada objeto dentro de Unity está encerrado por un paralelepípedo, sobre el cual las propiedades del *transform* como posición o escala [112], se aplican. Las coordenadas de posición del objeto en Unity corresponden al centro del paralelepípedo que encierra al objeto virtual, en este caso el cráneo 3D. El objetivo consiste en tratar de duplicar esta figura con el *phantom* en el mundo real, hallar las correspondientes distancias desde el centro de la figura a cada eje y copiar dichos valores en el *transform* del modelo 3D en Unity, logrando el emparejamiento espacial de ambos objetos. Por supuesto, se debe garantizar que la orientación sea duplicada en Unity también; ya sea con los *gizmos* de rotación del objeto, o directamente en el *transform*. Al tratarse de un objeto 3D generado a partir de una tomografía de un paciente ubicado en posición anatómica, la orientación es relativamente fácil de obtener, ya que sólo es necesario rotar el objeto sobre uno o dos ejes con múltiplos o submúltiplos de 90°, para lograr la orientación que se aprecia en la Figura 3.79. De esta forma, tanto la posición como la orientación será la misma en ambos casos.

Para lograr este objetivo, en primer lugar se alineó el plano medio-sagital del *phantom* con el origen del sistema de coordenadas del robot, o sea, el punto central de la base circular del robot; tratando, además de que el plano fuera equidistante a los ejes XY (Figura 3.79). Conviene mencionar que los cirujanos que realizan este tipo de intervenciones quirúrgicas se ubican a un lado del paciente; no obstante, para este experimento la cara inferior del *phantom* se orienta hacia el robot, con el fin de facilitar el acceso del endoscopio a la cavidad nasal y simplificar los cálculos geométricos del *bounding box.*

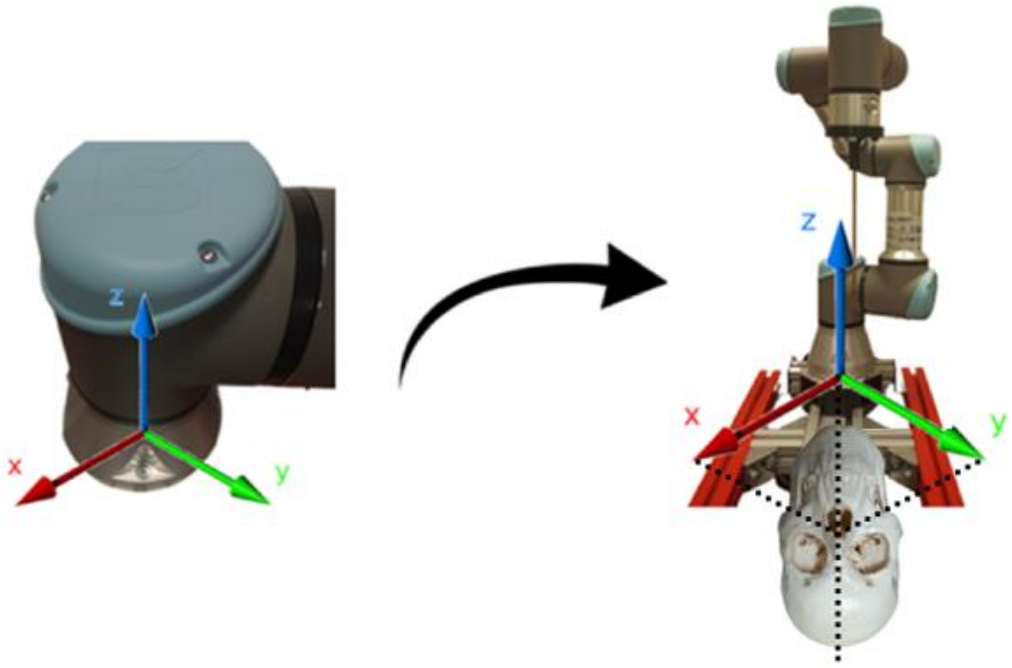


Figura 3.79: Sistema de referencia de coordenadas del robot.

Luego, como el plano medio-sagital está ubicado a la misma distancia del eje X y del eje Y, las coordenadas correspondientes a dichos ejes serán iguales. Además de estos dos valores, es necesario conocer el valor correspondiente al eje Z. Para calcular dicho valor, primero se debe conocer la altura del *phantom* con respecto a la base del robot; para ello se ubicó un riel en la parte superior del *phantom*, garantizando que estuviera totalmente paralelo al nivel de la mesa al mismo tiempo que estuviera en contacto con el punto más sobresaliente del hueso nasal. Y para calcular los valores de los ejes X e Y, se debe hallar la distancia desde la coronilla (vertex) del *phantom* hasta el punto central de la base del robot; para ello se proyectó la medida de la coronilla hasta el nivel de la mesa y se midió desde este punto hasta el perímetro de la base del robot. Todas estas medidas se consignan en la (Figura 3.81). Cabe resaltar que el *transform* del objeto 3D en Unity obtiene sus valores de los puntos medios del *bounding box* hasta el origen de coordenadas; por esta razón, tanto los valores de altura como de profundidad del *phantom* deben ser divididos sobre 2 durante los respectivos cálculos.

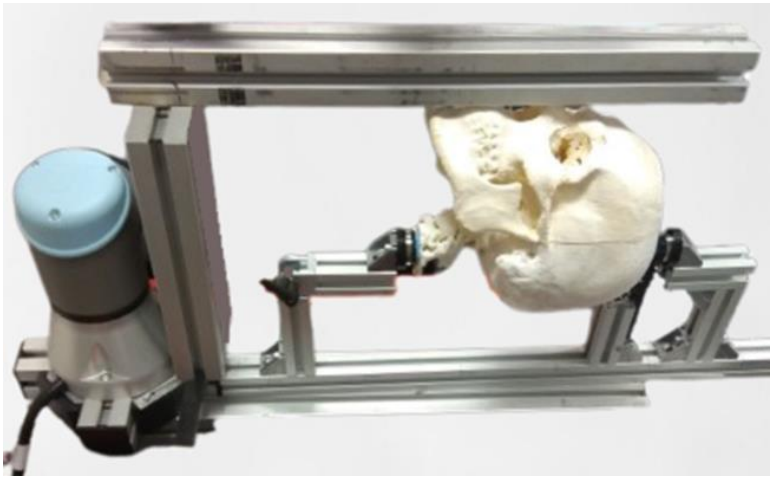


Figura 3.80: Vista diagonal del *bounding box* en el *phantom*.

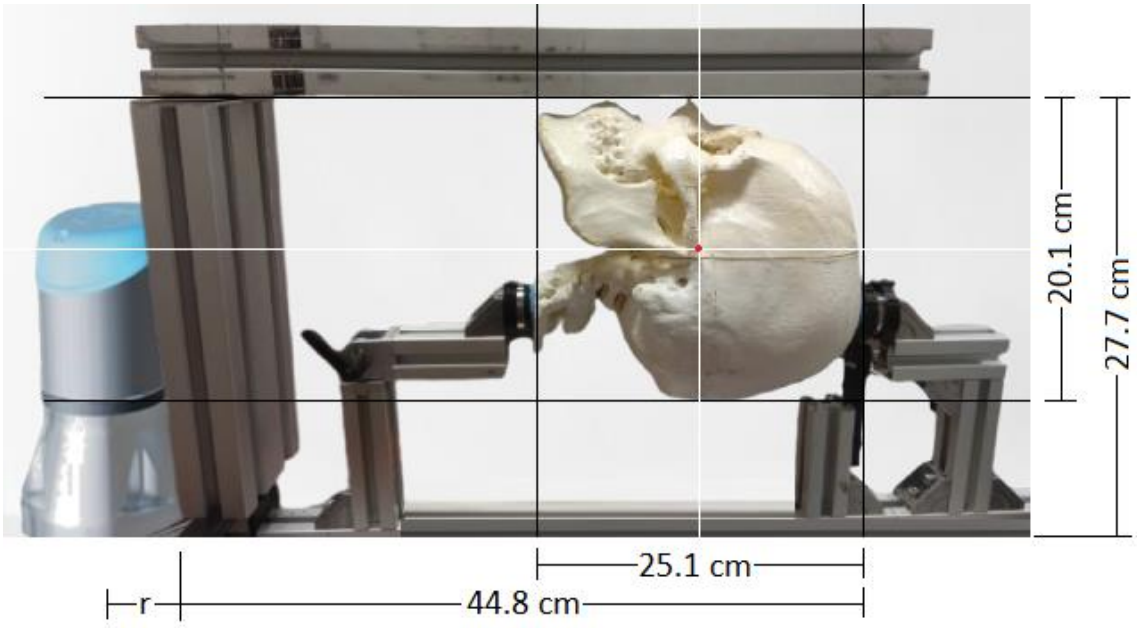


Figura 3.81: Vista lateral del *bounding box* en el *phantom* con sus medidas.

Coordenada del eje Z:

$$z = H - \frac{h}{2}$$

Donde,

H = altura total, desde el hueso nasal hasta el nivel base del robot.

h = medida de profundidad del *phantom*.

Entonces,

$$z = 27.7 - \frac{20.1}{2}$$

$$z = 17.65\ cm$$

$$z = 0.1765\ m$$

Coordenada de los ejes X e Y:

$$L_T = L + r$$

$$r = \frac{P}{2\pi}$$

Donde,

L_T = longitud total, desde la coronilla hasta el origen de coordenadas (punto central base del robot).

L = longitud desde la coronilla hasta el perímetro base del robot.

R = radio base del robot.

P = perímetro base del robot.

$$r = \frac{41.1}{2\pi}$$

$$r = 6.54\ cm \tag{3. 3}$$

$$L_T = 44.8 + 6.54$$

$$L_T = 51.34\ cm$$

Luego, se calcula la distancia desde el origen hasta el punto medio del *bounding box* (l = longitud del *phantom*).

$$L_{BB} = L_T - \frac{l}{2}$$

$$L_{BB} = 51.34 - \frac{25.1}{2} \tag{3. 4}$$

$$L_{BB} = 38.79\ cm$$

$$L_{BB} = 0.3879\ m$$

Finalmente, se calculan las coordenadas de los ejes X e Y, haciendo uso del teorema de Pitágoras.

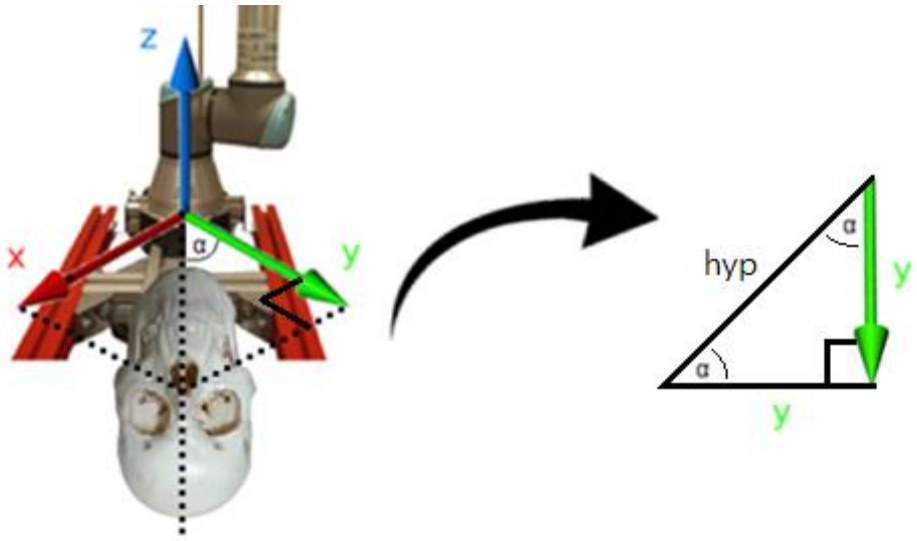


Figura 3.82: Teorema de Pitágoras aplicado al *bounding box*.

$$hyp^2 = y^2 + y^2$$

$$y = \sqrt{\frac{hyp^2}{2}}$$

$$y = \sqrt{\frac{{L_{BB}}^2}{2}} \qquad (3.\ 5)$$

$$y = \sqrt{\frac{38.79^2}{2}}$$

$$y = 27.42\ cm$$

$$y = 0.2742\ m$$

Como,

$$y = x$$

$$x = 0.2742\ m$$

De esta forma, se obtuvieron los 3 valores de posición correspondientes al cráneo virtual en Unity. Cabe mencionar que al principio de esta sección se ubicó el *phantom* en la

región donde todos los ejes son positivos para facilitar los cálculos matemáticos; pero en realidad, el *phantom* está ubicado en la región donde X e Y son negativos:

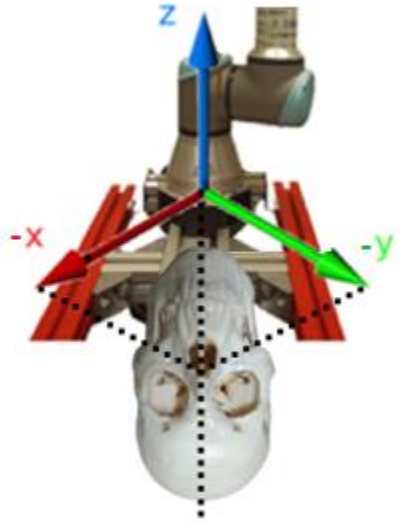


Figura 3.83: Sistema de referencia real del robot.

Entonces se tiene lo siguiente:

$$x = -0.2742\ m$$

$$y = -0.2742\ m$$

$$z = 0.1765\ m$$

En segundo lugar, como el sistema de coordenadas de Unity es diferente al del robot, se deben aplicar las siguientes conversiones:

Tabla 3.1: Conversion sistema de referencia Unity-ROS

UNITY	*ROS*
EJE X	EJE Y
EJE Y	EJE Z
EJE Z	EJE -X

Dando como resultado:

$$x = -0.2742\ m$$

$$y = 0.1765\ m$$

$$z = 0.2742\ m$$

La precisión de este método será evaluada más adelante en la sección “Error de Registro”.

3.7.2. **Registro manual con fiducias.**

El segundo método que se utilizó para realizar el registro fue con marcadores fiduciarios. Estos son dispositivos médicos que se colocan en el cuerpo o sobre el cuerpo para marcar un área para tratamiento con radioterapia o cirugía [113]. En neurocirugía y ortopedia, consisten generalmente en dispositivos que se fijan a la estructura ósea que se desea intervenir. Estos dispositivos se adhieren al paciente antes de la etapa de adquisición de datos, ya que las imágenes y modelos generados posteriormente tendrán estos puntos de referencia, facilitando en gran medida el proceso de registro y rastreo durante la intervención quirúrgica.

Dicho lo anterior, es pertinente resaltar que el cráneo del que se obtuvo la tomografía computarizada no tenía fiducias adheridas al hueso; por lo tanto, la segmentación y posterior impresión 3D no iban a contar con algún punto significativo para el proceso de registro. De haberse dado el caso, tanto el *phantom* como el modelo virtual generado tendrían una forma de emparejarse más precisa. Por esta razón, se procedió a ubicar las fiducias en los puntos anatómicos más representativos de la porción más anterior del modelo 3D (cara o rostro humano), como valles o crestas, los cuales fueron fácilmente identificables en el *phantom* posteriormente. A continuación, se explican los 2 diseños de fiducias para el modelo 3D de Unity, que varían tanto en posición, tamaño e incluso color.

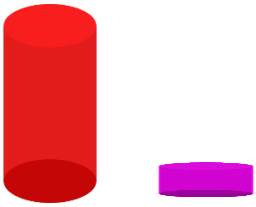


Figura 3.84: 1ro y 2do diseño de fiducias 3D.

Básicamente, las fiducias consisten en objetos tridimensionales en forma de cilindro, con características básicas de posición, orientación (fija), escala (tamaño) y material (color); pero sin características de colisión (*rigidbody*), con el fin de coincidir, en la medida de lo posible, el centro de estos objetos con la superficie del cráneo virtual (véase sección "*Bounding box*"). Esto quiere decir que la posición de la fiducia será aproximadamente el punto que intercepta la superficie del cráneo virtual. Por otro lado, dentro de la ventana de jerarquías en el editor se deben ubicar las fiducias como objetos hijo del cráneo virtual, de modo que al mover el cráneo las fiducias se muevan con él.

Para el primer diseño se definieron 5 puntos anatómicos de referencia; usualmente el número mínimo de puntos visibles a escoger es 3 [114]. Se ubicaron dos en los márgenes supraorbitales del hueso frontal, uno en la glabela del hueso frontal y dos cerca de la cara orbitaria del hueso cigomático.

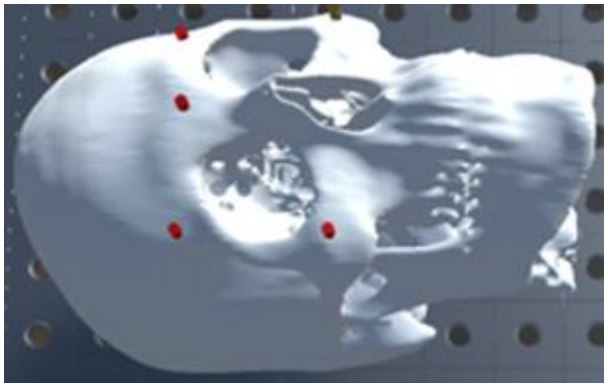


Figura 3.85: 1er modelo fiducias 3D.

De este primer diseño se hizo evidente que las fiducias no eran adecuadas tanto en tamaño como en ubicación. En ubicación porque las 3 fiducias superiores estaban parcialmente alineadas sobre 2 ejes, mientras que las 4 fiducias exteriores estaban parcialmente alineadas sobre 3 ejes (Figura 3.85). Y en tamaño, porque las fiducias sobresalían demasiado de la superficie del cráneo, razón que más adelante, aún si se tuvieran las réplicas exactas reales, posicionarlas correctamente en la superficie del *phantom* sería una tarea de considerable complejidad. Sin embargo, la razón por la que se descartó este primer modelo, más que el tamaño fue el problema de ubicación, ya que resulta particularmente inadecuado definir las posiciones de las fiducias compartiendo uno o más ejes de coordenadas. Esto se debe a que los vectores de posición (coordenadas XYZ) de cada fiducia deben ser linealmente independientes unos de otros, ya que en caso de que se utilice un algoritmo de registro, ningún vector debe ser un subproducto vectorial

de otro, esto con el fin de alcanzar una mayor precisión en el emparejamiento de las superficies del modelo 3D y del *phantom*.

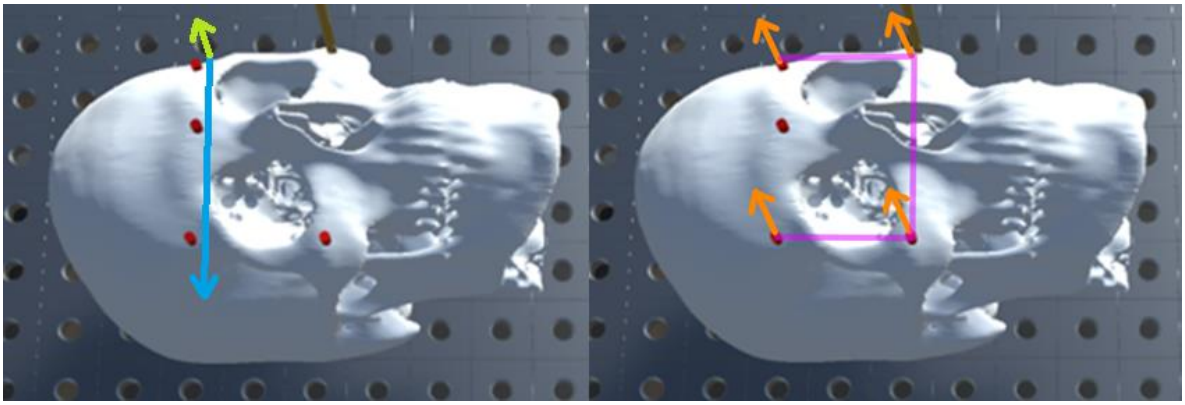


Figura 3.86: Errores de ubicación primer diseño fiducias 3D.

En el segundo diseño, se ubicaron las fiducias en 6 puntos anatómicos de referencia (Figura 3.87); dos en los márgenes supraorbitales del hueso frontal, 2 en el proceso frontal del hueso cigomático, 1 en el borde inferior-medial del hueso nasal y 1 en la espina nasal anterior del hueso maxilar superior. Además, se redujo su tamaño de modo que al ubicar el efector final del robot (endoscopio) sobre estas, se aproximara con mayor exactitud al punto que representan en el *phantom*; ya que en él las fiducias iban a ser simplemente puntos resaltados con marcador negro. Cabe mencionar que las fiducias 3D seguirían conservando su forma cilíndrica, más pequeñas como para no sobresalir demasiado sobre la superficie del cráneo, pero no tanto como para no ser posible visualizarlas. Así pues, mientras las fiducias en el *phantom* del cráneo son puntos dibujados sobre la superficie del mismo (Figura 3.88), las fiducias en el modelo 3D son más parecidas a los tornillos utilizados en los procedimientos quirúrgicos reales, por lo tanto, sobresalen de la superficie unos cuantos milímetros.

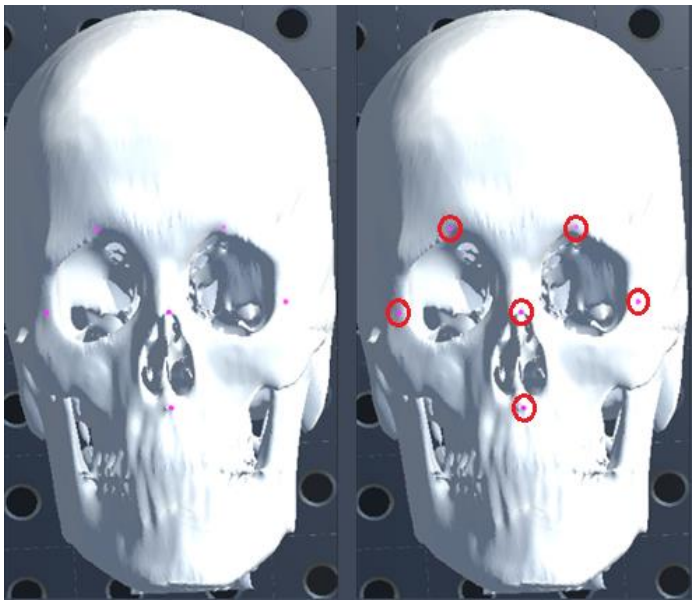


Figura 3.87: 2do modelo fiducias 3D.

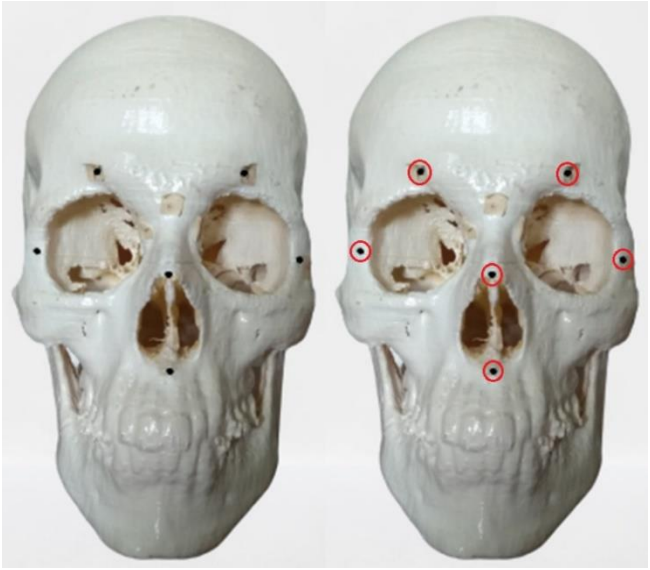


Figura 3.88: 2do modelo fiducias en el *phantom*.

Finalmente, como el endoscopio real y el endoscopio virtual coinciden visualmente en el espacio tridimensional, se procede a realizar el emparejamiento del *phantom* y del cráneo virtual con dicha herramienta. Del método de registro anterior, se tiene el *phantom* en su posición anatómica ubicado a una determinada distancia del robot (véase sección 3.7.1). Es importante no modificar esta posición, con el fin de evaluar la precisión de ambos métodos de registro.

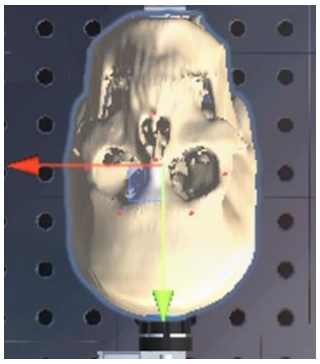


Figura 3.*89*: *Gizmos* de posición del cráneo virtual.

El proceso de registro consiste en ubicar la punta del endoscopio real (con el botón de movimiento libre del TP) sobre cada una de las fiducias definidas al principio de esta sección y una vez estén en contacto, dirigirse a la pantalla de Unity para manipular los *gizmos* de posición del cráneo virtual (Figura 3.89), hasta que la fiducia equivalente también esté en contacto con el endoscopio virtual. De esta manera, se establece una correspondencia entre el *phantom* y el cráneo virtual (Figura 3.90, Figura 3.91, Figura 3.92, Figura 3.93, Figura 3.94, Figura 3.95, Figura 3.96). Durante este proceso, puede ser necesario hacer uso de los *gizmos* de orientación del cráneo.

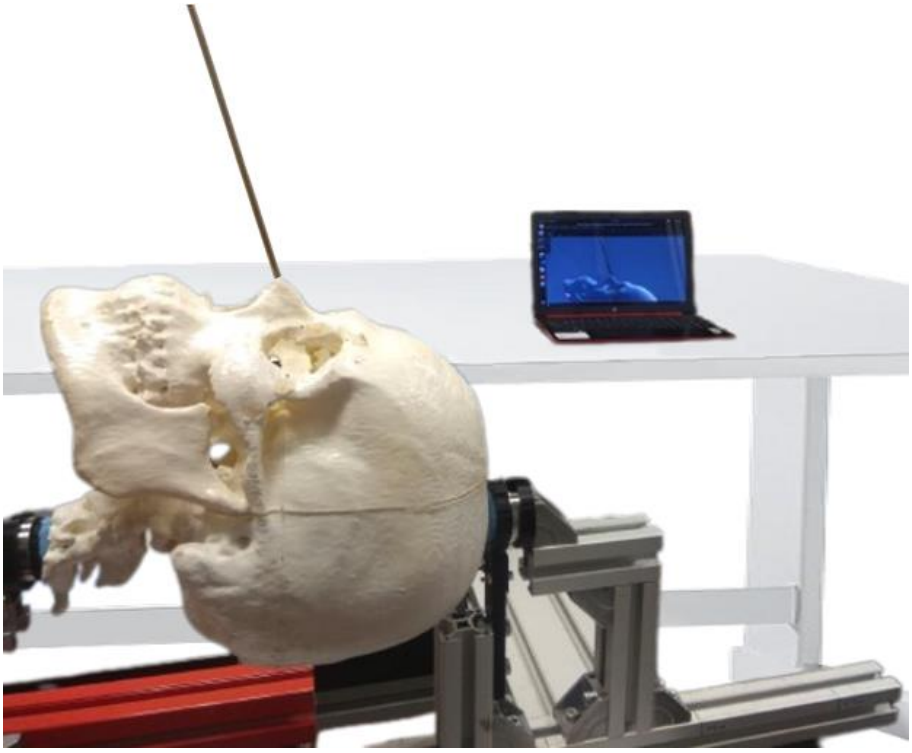


Figura 3.90: Emparejamiento con fiducias.

Es importante resaltar que cada vez que se ajusta el cráneo con relación a una fiducia, se van a alterar los ajustes hechos para las demás. Debido a ello esta tarea supone una gran cantidad de tiempo, además de que el resultado final puede no ser el más exacto, como se verá más adelante en la sección “Error de Registro”.

Figura 3.91: Emparejamiento fiducial en el borde inferior-medial del hueso nasal.

Figura 3.92: Emparejamiento fiducial en la espina nasal anterior del hueso maxilar superior.

Figura 3.93: Emparejamiento fiducial en el proceso frontal del hueso cigomático derecho.

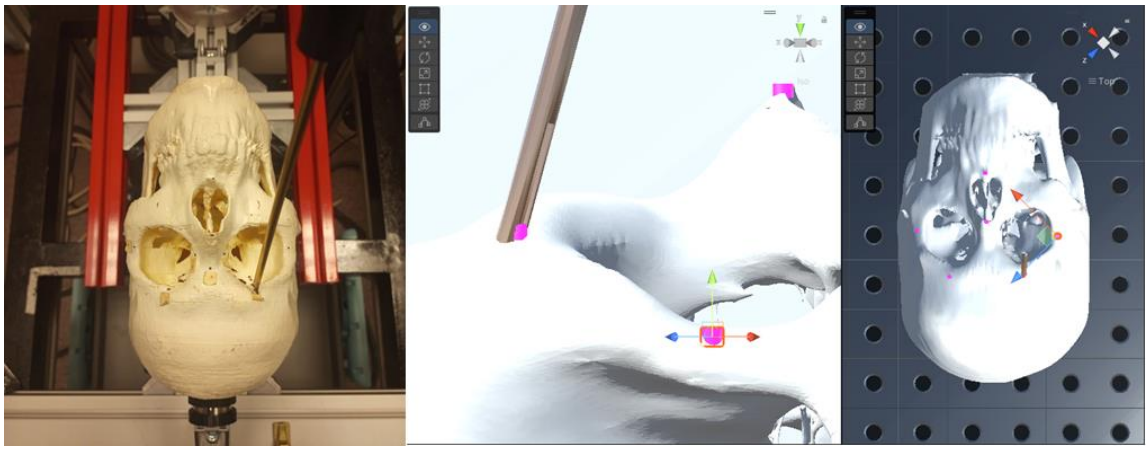


Figura 3.94: Emparejamiento fiducial en el margen supraorbital derecho del hueso frontal.

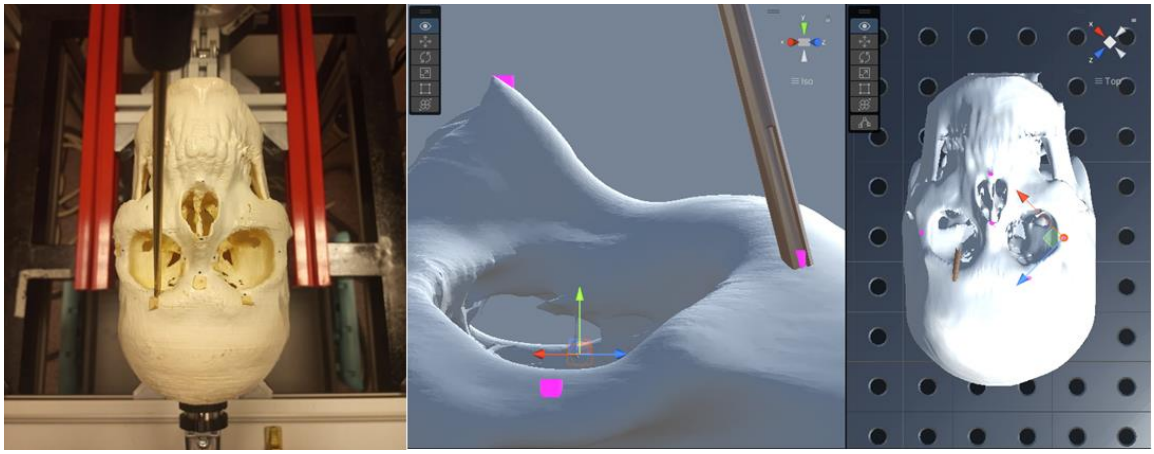


Figura 3.95: Emparejamiento fiducial en el margen supraorbital izquierdo del hueso frontal.

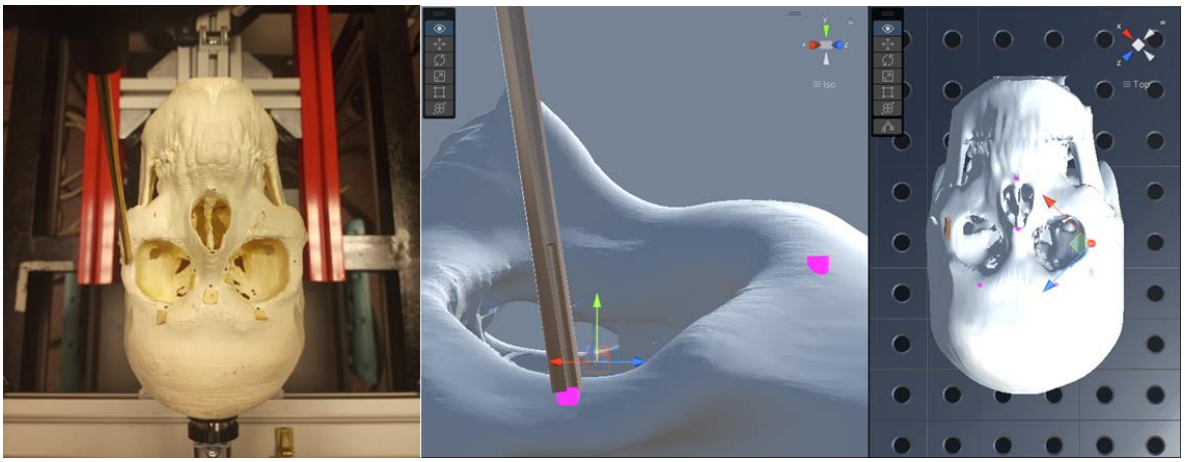


Figura 3.96: Emparejamiento fiducial en el proceso frontal del hueso cigomático izquierdo.

Una vez terminado el proceso de registro, se puede verificar que la posición del cráneo virtual hallado con este método y el hallado por medio del *bounding box* es ligeramente diferente (Figura 3.97).

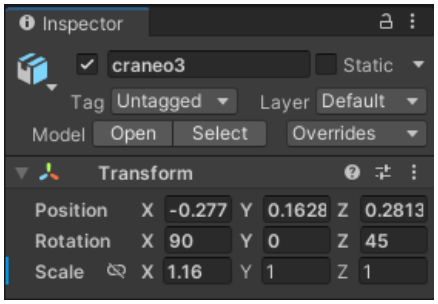


Figura 3.97: Propiedades del *Transform* del cráneo virtual.

3.7.3. **Error de registro**

El registro de la imagen del paciente es un procedimiento de suma importancia en neurocirugía. Su aplicación fue necesaria una vez las técnicas *frameless* empezaron a reemplazar el uso del marco estereotáctico. Dentro de estas técnicas se encuentran el registro con marcadores y sin marcadores. El registro con marcadores fiduciarios es uno de los más utilizados debido a su exactitud y versatilidad [115]. Actualmente es implementado con sistemas de seguimiento ópticos o electromagnéticos.

En el presente trabajo se hizo uso de dos métodos de registro, uno con *bounding box* y otro con marcadores fiduciarios; ambos realizados de forma manual. El *bounding box* no

es un método oficial, pero fue útil como punto de comparación para el registro con fiducias, que sí es un método como tal. El objetivo consistió en emparejar el *phantom* y el cráneo virtual por medio de estos dos métodos y posteriormente escoger el más adecuado.

Es importante señalar que el registro manual es un proceso largo, extenuante y sujeto a errores. Por esta razón, los equipos comerciales de navegación quirúrgica utilizan algoritmos de registro automáticos o semiautomáticos, que se encargan de realizar el registro en cuestión de minutos y con gran exactitud (<2mm [116]). Para tejidos duros, como estructuras óseas, utilizan transformaciones rígidas (rotaciones y traslaciones) o afines (escalado), mientras que para tejidos blandos utilizan transformaciones elásticas.

Para escoger el método de registro más conveniente, fue necesario calcular el error de registro o en este caso, el *fiducial registration error*. Este consiste en determinar, bajo un mismo sistema de referencia, la coincidencia de dos pares de puntos en el espacio virtual y real. Para ello se puso a disposición el sistema de rastreo y captura de movimiento 3D, *Optitrack*, para calcular la posición de las fiducias reales. Para el tutorial de configuración y uso dirigirse al Anexo A.3.

Una vez se ha configurado correctamente el programa *Tracking Tools*, se procede a tomar la medida de las posiciones de cada una de las fiducias del *phantom*, siguiendo el siguiente orden:

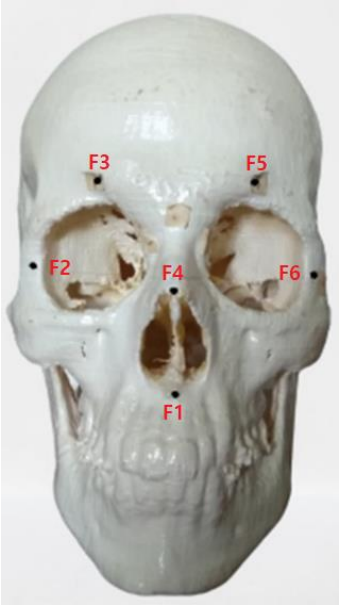


Figura 3.98: Nombre de las fiducias.

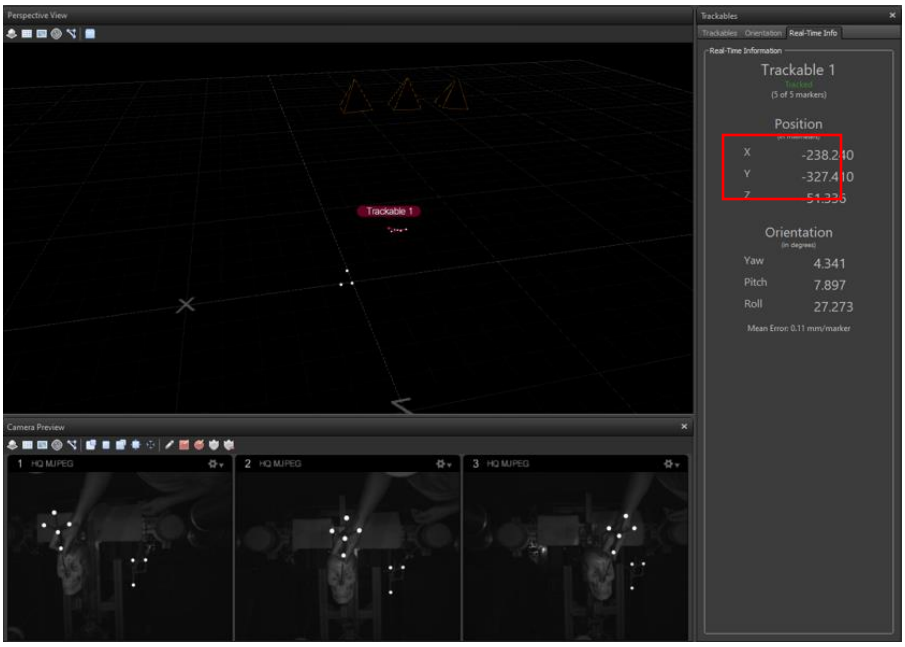


Figura 3.99: Fiducia 1 en *TrackingTools*.

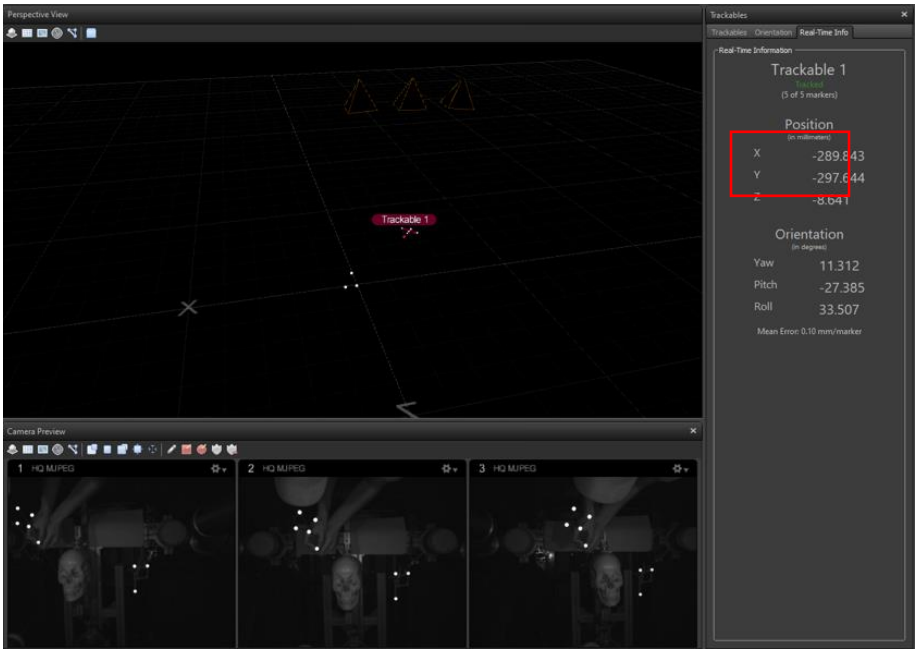


Figura 3.100: Fiducia 2 en *TrackingTools*.

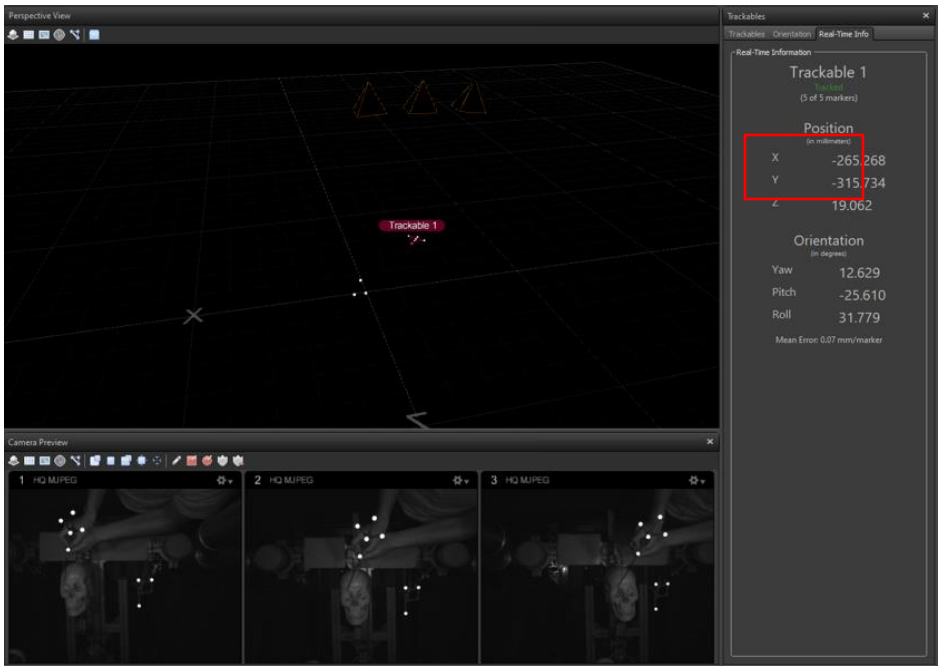


Figura 3.101: Fiducia 3 en *TrackingTools*.

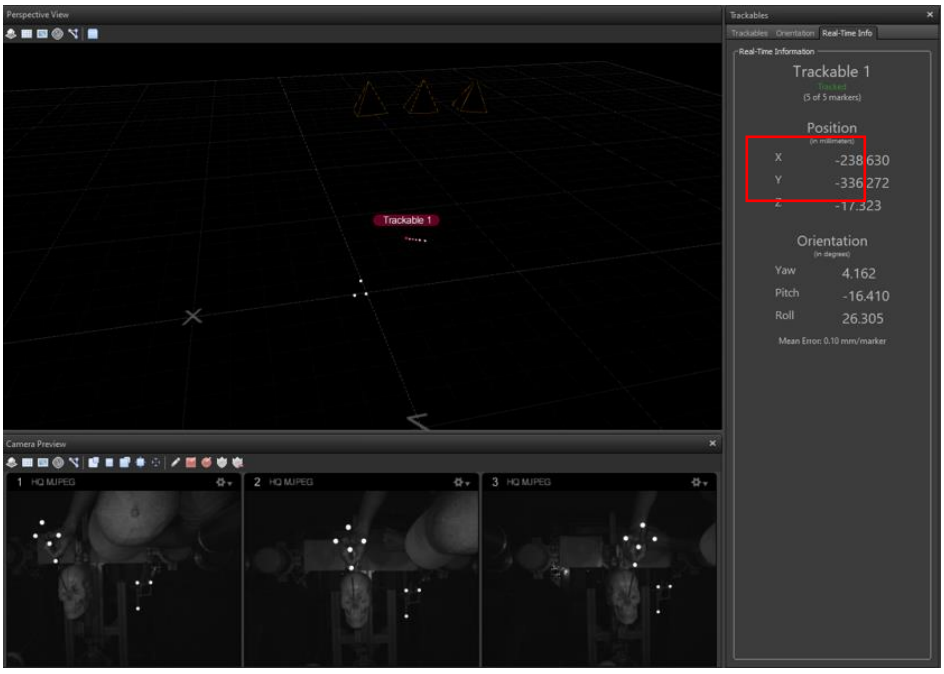


Figura 3.102: Fiducia 4 en *TrackingTools*.

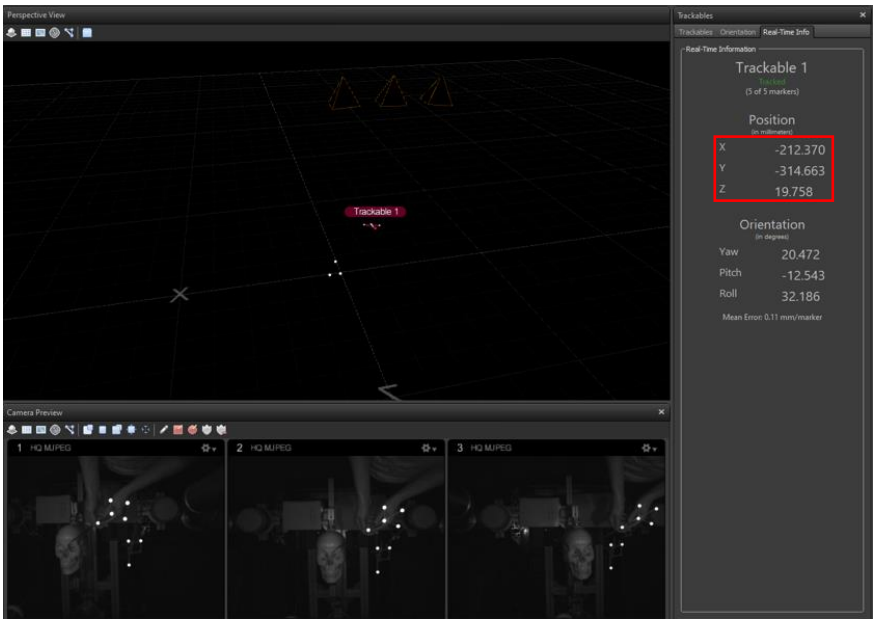


Figura 3.103: Fiducia 5 en *TrackingTools.*

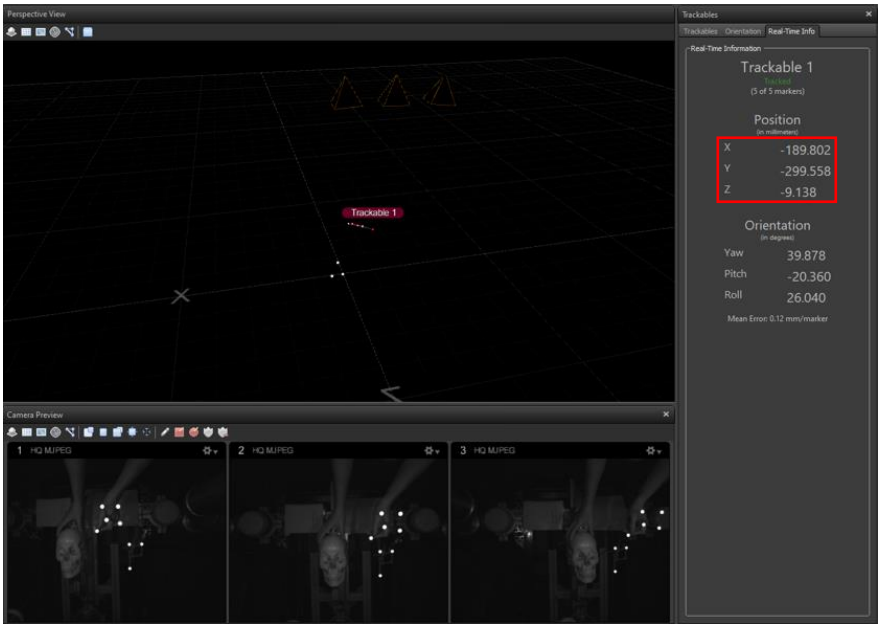


Figura 3.104: Fiducia 6 en *TrackingTools*.

Las posiciones de las fiducias en el sistema de referencia de *Optitrack* y en el sistema de referencia del robot, después de aplicar la correspondiente matriz de transformación (Anexo A.3), son las siguientes:

Tabla 3.2: Posición de las fiducias en el sistema de referencia de *Optitrack* y en el sistema de referencia del robot.

		Fiducias Reales (Optitrack)	Fiducias Reales (UR3e)
F1	**X**	-238.2400 mm	-254.9211 mm
	Y	-327.4100 mm	-252.8534 mm
	Z	-51.3360 mm	257.3803 mm
F2	**X**	-289.8430 mm	-260.6576 mm
	Y	-297.6440 mm	-321.1942 mm
	Z	-8.6410 mm	231.5275 mm
F3	**X**	-265.2680 mm	-296.1562 mm
	Y	-315.7340 mm	-316.8341 mm
	Z	19.0620 mm	252.0085 mm
F4	**X**	-238.6300 mm	-281.3241 mm
	Y	-336.2720 mm	-272.8038 mm
	Z	-17.3230 mm	269.2308 mm
F5	**X**	-212.3700 mm	-328.1815 mm
	Y	-314.6630 mm	-274.7247 mm
	Z	19.7580 mm	251.0036 mm
F6	**X**	-189.8020 mm	-319.4884 mm
	Y	-299.5580 mm	-240.2771 mm
	Z	-9.1380 mm	233.3898 mm

El siguiente paso consiste en encontrar las posiciones de las fiducias en ambos métodos de registro. Cabe aclarar que para calcular el error de registro del *bounding box*, también se debe hacer uso de las fiducias; más adelante se confirmara que las posiciones de las fiducias son diferentes debido a que la posición absoluta del cráneo es diferente en ambos casos.

A modo de comparación, una vez ingresadas las posiciones del cráneo calculadas con el método *bounding box*, se ubicó el endoscopio en las fiducias del cráneo real y se observó en Unity la diferencia entre el endoscopio y las fiducias virtuales. Esta evidente diferencia será corroborada con el error de registro.

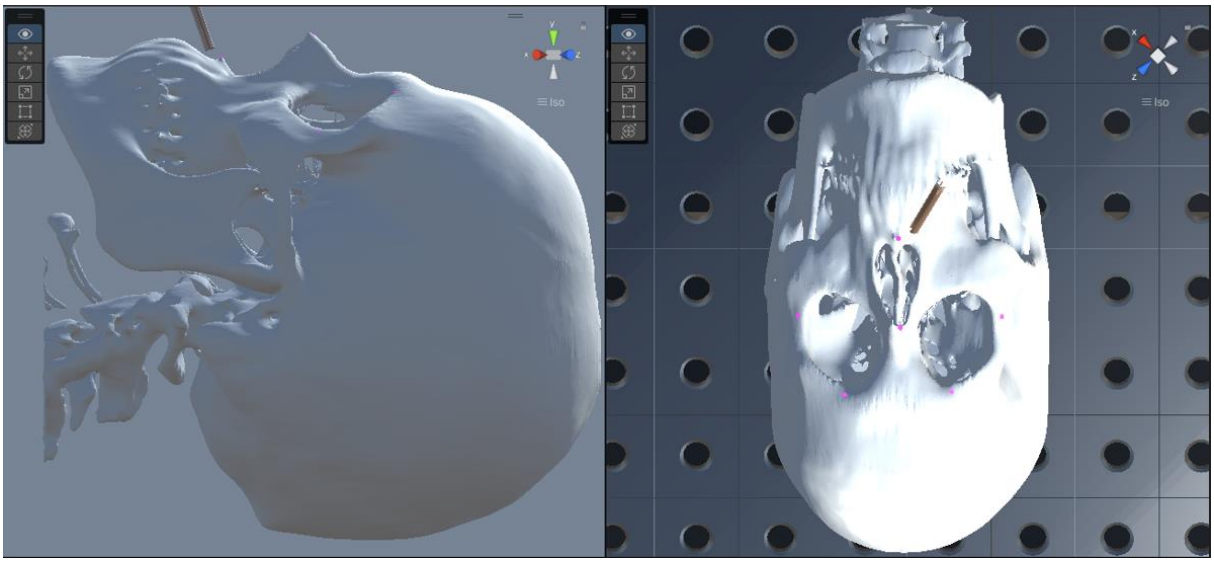


Figura 3.105: Fiducia 1 (*bounding box*).

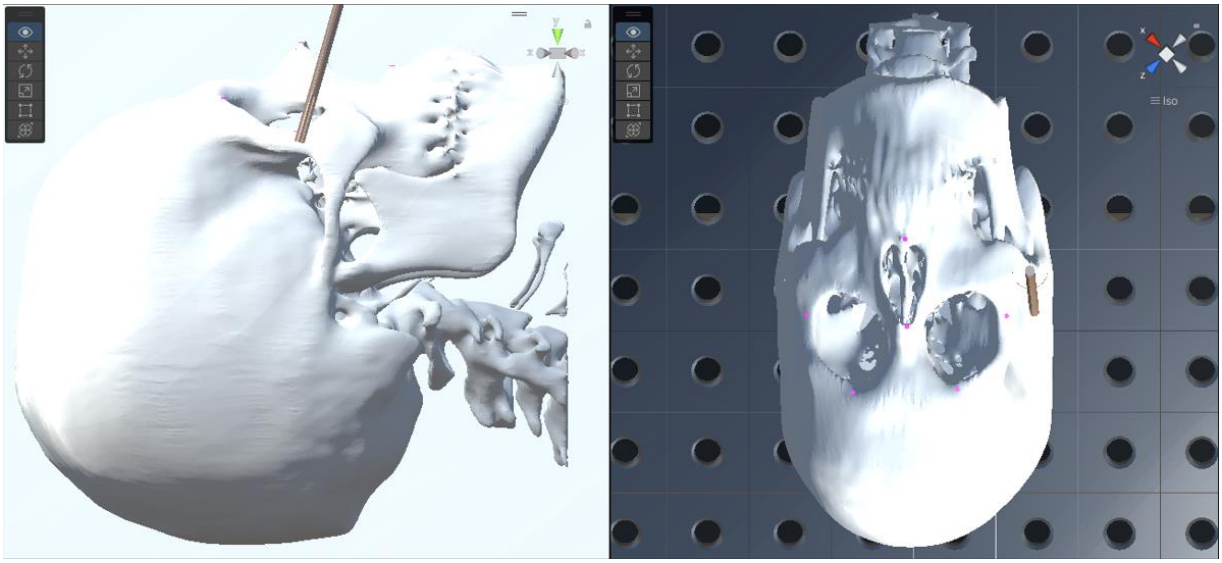


Figura 3.106: Fiducia 2 (*bounding box*).

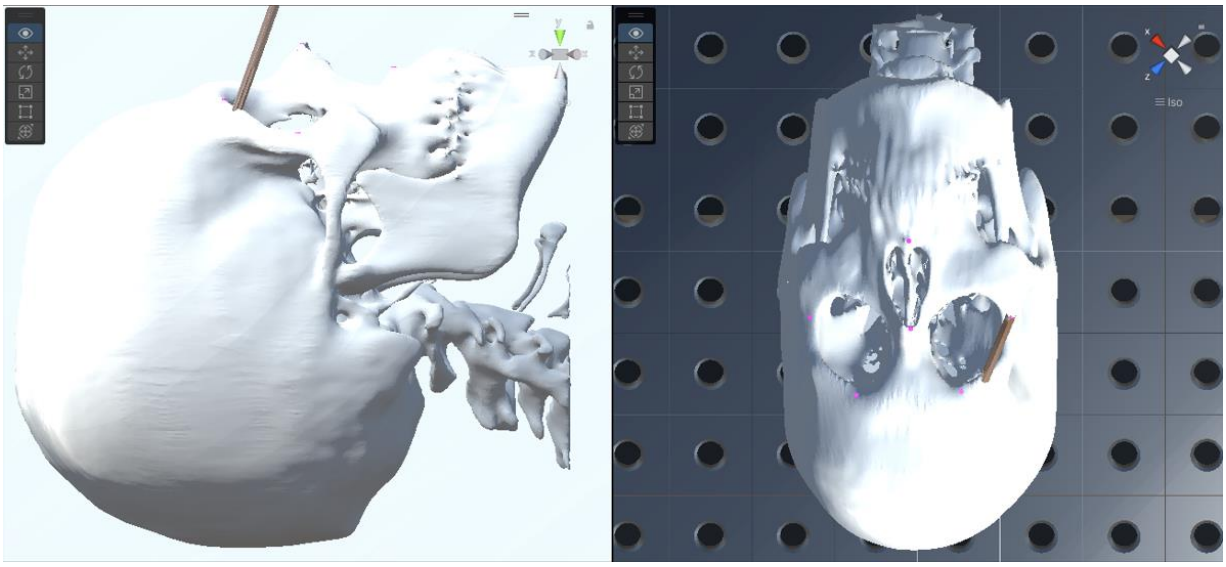


Figura 3.107: Fiducia 3 (*bounding box*).

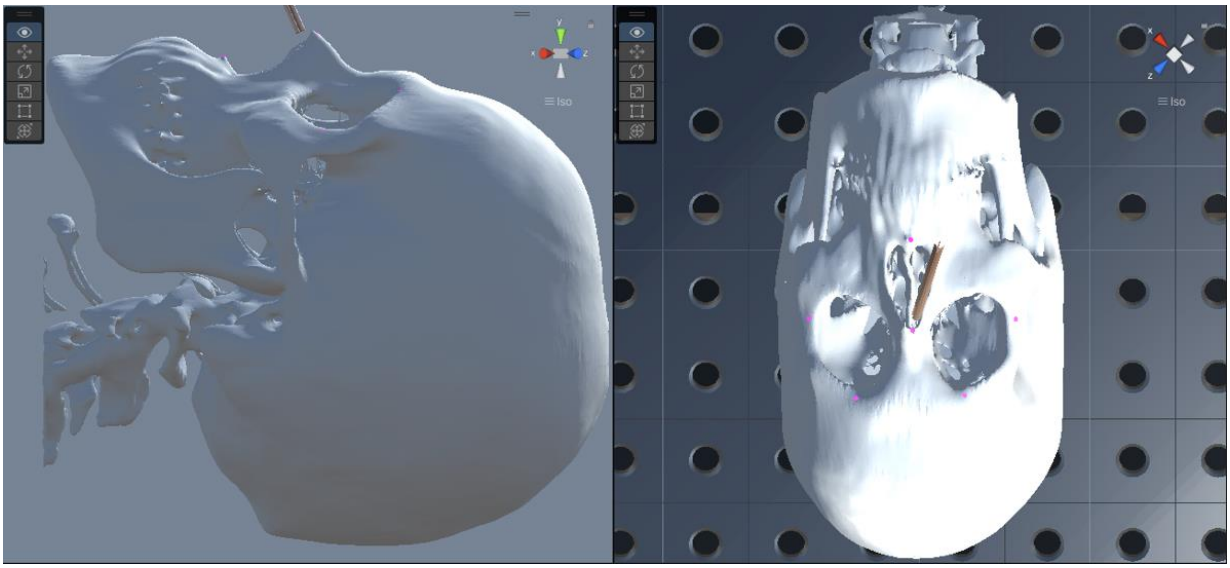


Figura 3.108: Fiducia 4 (*bounding box*).

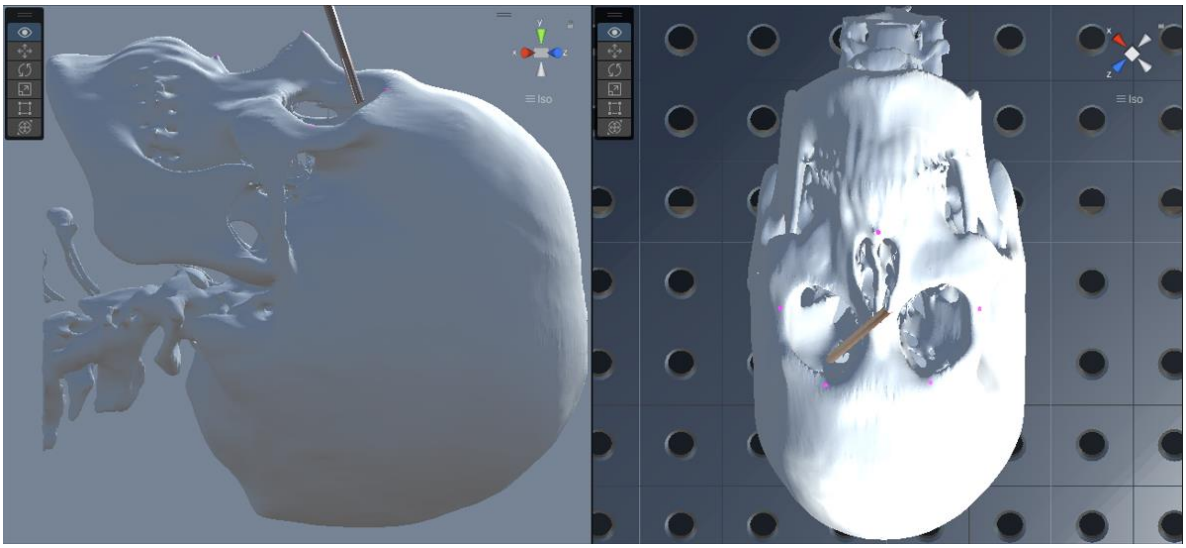


Figura 3.109: Fiducia 5 (*bounding box*).

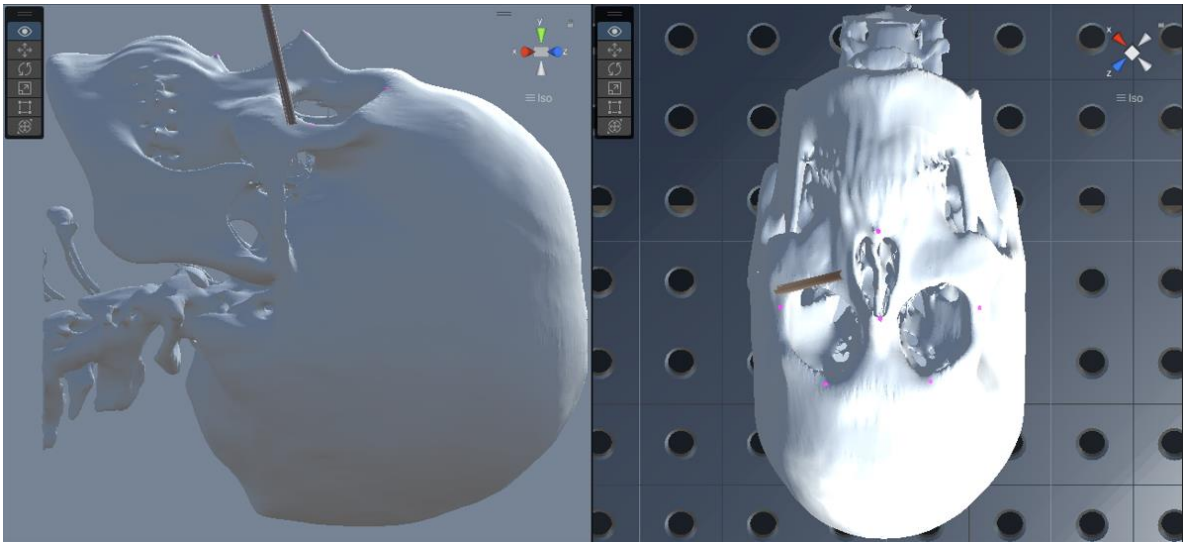


Figura 3.110: Fiducia 6 (*bounding box*).

Ahora, las posiciones que presenta cada fiducia en el *transform* son las posiciones relativas al objeto padre, el cráneo en este caso, por lo que se deben encontrar las posiciones respecto al origen del sistema de coordenadas. Para encontrar las posiciones absolutas de las fiducias, primero se debe eliminar la relación de jerarquía con el objeto padre, duplicando las 6 fiducias y seleccionando la opción "*Clear Parent*", al dar *click* derecho en cada una en el árbol de jerarquías. Una vez anotadas las posiciones, se

procede a eliminar las fiducias duplicadas, ya que son objetos independientes. Este proceso se debe repetir cuando se utilice el otro método de registro. No se debe olvidar que las posiciones de las fiducias obtenidas en Unity deben ser convertidas al sistema de coordenadas del robot con las siguientes convenciones:

Tabla 3.3: Conversión sistema de referencia Unity-ROS

UNITY	*ROS*
EJE X	EJE Y
EJE Y	EJE Z
EJE Z	EJE -X

Las posiciones absolutas XYZ (convertidas) de cada una de las fiducias, son las siguientes:

Tabla 3.4: Posición fiducias virtuales (BB)[3]

		Fiducias Virtuales (BB)
F1	**X**	-252.1750 mm
	Y	-244.4647 mm
	Z	275.8000 mm
F2	**X**	-247.3639 mm
	Y	-312.4912 mm
	Z	249.4000 mm
F3	**X**	-284.6822 mm
	Y	-315.1952 mm
	Z	264.5700 mm
F4	**X**	-276.0781 mm
	Y	-269.3521 mm
	Z	285 mm
F5	**X**	-321.9693 mm
	Y	-278.3323 mm
	Z	261.7700 mm
F6	**X**	-315.3112 mm
	Y	-239.0285 mm
	Z	244.8300 mm

[3] *BB: Bounding box*

Una vez se han eliminado las fiducias duplicadas, se ingresa la posición y orientación del cráneo obtenidas durante el registro con fiducias y se repite el proceso anterior para hallar las posiciones absolutas (y convertidas) de las mismas:

Tabla 3.5: Posición fiducias virtuales (2do método).

		Fiducias Virtuales (2do método)
F1	**X**	-259.2750 mm
	Y	-247.6647 mm
	Z	262.1000 mm
F2	**X**	-254.4639 mm
	Y	-315.6912 mm
	Z	235.7000 mm
F3	**X**	-291.7822 mm
	Y	-318.3952 mm
	Z	250.8700 mm
F4	**X**	-283.1781 mm
	Y	-272.5521 mm
	Z	271.3000 mm
F5	**X**	-329.0693 mm
	Y	-281.5323 mm
	Z	248.0700 mm
F6	**X**	-322.4112 mm
	Y	-242.2285 mm
	Z	231.1300 mm

Al tener estos resultados es posible calcular el error de registro para cada método. Con esto en consideración, primero se utiliza la diferencia euclidiana para hallar la distancia más corta entre cada par de puntos tridimensionales de un mismo par de fiducias. Esta diferencia entre las coordenadas reales y las virtuales representa la distancia en que cada par de puntos debe calibrarse [117]. Acto seguido, se utiliza la raíz cuadrada de la media cuadrática para encontrar el error de registro. Este valor representa la acumulación de errores de cada método, o en términos más generales, la exactitud del sistema de navegación (Unity) de representar los marcadores anatómicos del *phantom*.

La fórmula para la diferencia euclidiana (*root-mean-square-error*) es:

$$RMS_{error} = \sqrt{(x' - x)^2 + (y' - y)^2 + (z' - z)^2} \tag{3.6}$$

Donde x'y'z' son las coordenadas virtuales y xyz son las coordenadas reales.

La fórmula para la raíz cuadrada de la media cuadrática es:

$$TotalRMS_{error} = \sqrt{\frac{\sum_{i=1}^{n} RMS_i{}^2}{n}} \tag{3.7}$$

Donde RMS es la distancia euclidiana de cada par de fiducias y *n* es el número de fiducias.

Una vez se han realizado estos cálculos se procede a completar la siguiente tabla, para visualizar mejor los resultados:

Tabla 3.6: Comparación RMS

		Fiducias Reales	Fiducias Virtuales (BB)	Fiducias Virtuales (2do método)	Distancia Euclidiana	
					BB	Fiducias
F1	X	-254.9211 mm	-252.1750 mm	-259.2750 mm	20.425 mm	8.255 mm
	Y	-252.8534 mm	-244.4647 mm	-247.6647 mm		
	Z	257.3803 mm	275.8000 mm	262.1000 mm		
F2	X	-260.6576 mm	-247.3639 mm	-254.4639 mm	23.914 mm	9.276 mm
	Y	-321.1942 mm	-312.4912 mm	-315.6912 mm		
	Z	231.5275 mm	249.4000 mm	235.7000 mm		
F3	X	-296.1562 mm	-284.6822 mm	-291.7822 mm	17.091 mm	4.781 mm
	Y	-316.8341 mm	-315.1952 mm	-318.3952 mm		
	Z	252.0085 mm	264.5700 mm	250.8700 mm		
F4	X	-281.3241 mm	-276.0781 mm	-283.1781 mm	16.973 mm	2.789 mm
	Y	-272.8038 mm	-269.3521 mm	-272.5521 mm		
	Z	269.2308 mm	285 mm	271.3000 mm		
F5	X	-328.1815 mm	-321.9693 mm	-329.0693 mm	12.943 mm	7.465 mm
	Y	-274.7247 mm	-278.3323 mm	-281.5323 mm		
	Z	251.0036 mm	261.7700 mm	248.0700 mm		
F6	X	-319.4884 mm	-315.3112 mm	-322.4112 mm	12.242 mm	4.178 mm
	Y	-240.2771 mm	-239.0285 mm	-242.2285 mm		
	Z	233.3898 mm	244.8300 mm	231.1300 mm		
				Error de Registro	**17.732** mm	**6.558** mm

Como se puede apreciar, el error de registro del método con fiducias es aproximadamente 3 veces menor que el método con *bounding box*. Por esta razón, las posiciones y orientaciones del cráneo virtual resultantes de este método fueron las escogidas para

proseguir con el trabajo. Este valor indica que existe en promedio, una distancia aproximada entre fiducia real y fiducia virtual de 6,5 mm; un valor bastante alto si se consideran los estándares de los equipos de navegación. Por otro lado, este error también puede entenderse como la adición de pequeños errores a lo largo del proceso de registro; como errores técnicos (materiales) y humanos (mediciones, cálculos matemáticos), y no solo debido al método de registro. A pesar de todo, para el propósito de este trabajo este valor es suficiente para no tener problemas al momento de realizar las pruebas con el endoscopio al interior del phantom.

En cuanto a la discrepancia tan notoria entre las posiciones de las fiducias en el *phantom* y en el modelo virtual bajo el método de *bounding box*, se justifica por el hecho de que, aunque el paciente se ubique de tal forma que facilite la tomografía computarizada (posición anatómica), esto no significa necesariamente que la tomografía esté totalmente alineada a los ejes de los planos sagital, coronal y axial. Por lo tanto, el método del *bounding box* no es adecuado más allá de una práctica experimental, ya que, aunque el *phantom* pueda ser alineado en estos ejes de forma casi perfecta, el modelo virtual poseerá pequeñas variaciones en la orientación sobre los 3 ejes (acarreadas desde la etapa de adquisición de datos). Por otro lado, y más importante aún, el método de registro con *bounding box* no funciona, o no es el más adecuado, si se empieza a rotar el *phantom* en cualquiera de sus tres ejes; ya que medir o calcular el ángulo de inclinación de forma manual, aparte de inexacto no es muy práctico.

3.8. **Rotaciones 3D en Unity y ROS.**

El proyecto actual ha hecho uso de la plataforma Unity, el simulador URSim y el framework ROS, para crear una aplicación. Los programas Unity y URSim se caracterizan porque permiten visualizar y modificar las orientaciones de los objetos 3D en la interfaz de usuario, bajo representaciones matemáticas como los ángulos de Euler o el vector-rotación. Esto se debe principalmente a que el usuario entiende de forma intuitiva la orientación de un objeto expresado como rotaciones en 3 dimensiones. Sin embargo, Unity opera internamente haciendo uso de cuaterniones, ya que matemáticamente son más eficientes (p.e., facilitan la interpolación de dos orientaciones o evitan singularidades como el "*Gymbal Lock*") [118]. Así mismo, el *framework* ROS ha estandarizado el uso de cuaterniones para representar orientaciones [119]. Por otro lado, tanto el *teach pendant*

real como el simulado en URSim, pueden representar la orientación de dos formas: por defecto como vector-rotación (forma compacta de la notación axial-angular), o en ángulos *Roll-Pitch-Yaw* (ángulos de Euler con orden ZYX y sistema de referencia global o absoluto) [120],[121]. Cabe mencionar que a diferencia de los ángulos RPY, los ángulos de Euler de Unity tienen un orden ZXY y un sistema de referencia local o relativo al objeto 3D [122].

Antes de pasar a explicar el modo en que se orienta el efector final del robot, se hará una breve introducción del concepto de cuaternión, base fundamental para entender el modo en que se realizan las rotaciones de los objetos 3D en Unity o las transformaciones espaciales en ROS.

3.8.1. El cuaternión

Los cuaterniones son un sistema numérico, inventado por el matemático irlandés William Rowan Hamilton. Está compuesto por un número real y 3 números imaginarios. La parte real es un escalar, mientras que los 3 números imaginarios forman un vector. Los coeficientes de los cuaterniones son valores con un rango entre -1 y 1, y se pueden representar de dos formas; en notación vectorial o en notación numérica compleja:

$$< w + x + y + z >$$

$$\underbrace{a}_{\text{Parte Real}} + \underbrace{bi + cj + dk}_{\text{Parte Imaginaria}}$$

Así como los números complejos son una extensión bidimensional de los números reales, los cuaterniones son una extensión cuatridimensional de los números complejos. Su utilidad radica en su conveniencia para describir una orientación en el espacio tridimensional. Además, las operaciones matemáticas entre cuaterniones son inherentemente más eficientes computacionalmente que las operaciones entre matrices de rotación. Por esta razón, son usados principalmente en áreas como programación gráfica o robótica.

Multiplicación de cuaterniones

Las propiedades de la multiplicación entre cuaterniones son diferentes a las de los números complejos. Además, la propiedad conmutativa no se cumple:

$$i^2 = j^2 = k^2 = ijk = -1$$
$$ij = -ji = k$$
$$ki = -ik = j$$
$$jk = -kj = i$$

Para multiplicar dos cuaterniones:

$$q_1 = w_1 + x_1 i + y_1 j + z_1 k$$

$$q_2 = w_2 + x_2 i + y_2 j + z_2 k$$

Se resuelve multiplicando término a término, se aplican las propiedades descritas anteriormente y se agrupan los términos:

$$(w_1 + x_1 i + y_1 j + z_1 k)(w_2 + x_2 i + y_2 j + z_2 k) = (w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) +$$

$$(w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) i + (w_1 y_2 + y_1 w_2 + z_1 x_2 - x_1 z_2) j +$$
$$(w_1 z_2 + z_1 w_2 + x_1 y_2 - y_1 x_2) k \qquad (3.8)$$

Cuaterniones como rotaciones en el espacio 3D

Para facilidad de compresión se utilizará el espacio tridimensional imaginario compuesto por los ejes (i, j, k), donde solo podrán ser visualizados cuaterniones sin parte real.

La forma como se describe la rotación de un punto usando cuaterniones, es la siguiente:

Primero se define un eje de rotación cualquiera (q).

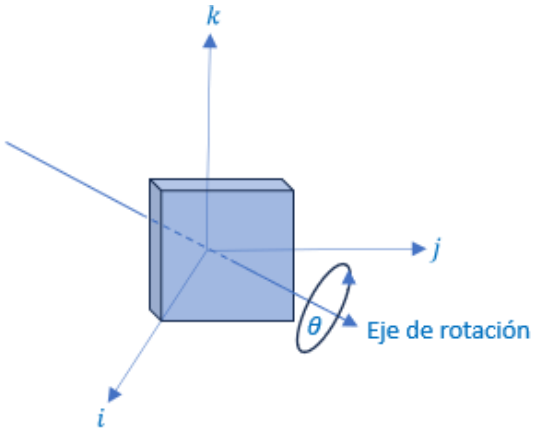


Figura 3.111: Objeto en el espacio tridimensional imaginario.

Este eje de rotación debe estar normalizado (vector unidad), es decir, la suma de sus componentes al cuadrado debe dar como resultado 1.

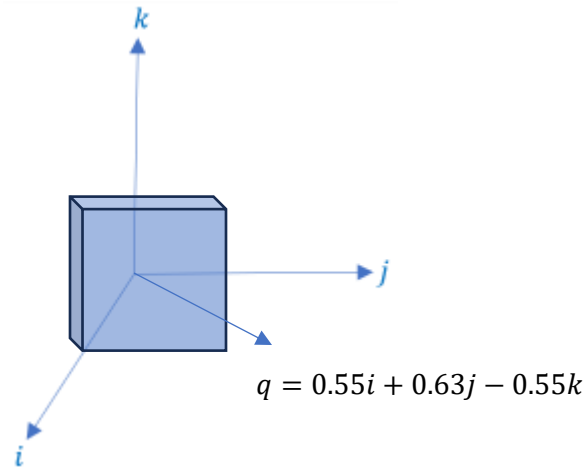


Figura 3.112: Eje de rotación representado como cuaternión.

Luego se multiplica el ángulo de rotación deseado (30°) por las coordenadas del eje de rotación, con la siguiente fórmula:

$$q_{(30°)} = \left(cos\frac{\theta}{2}\right) + \left(sin\frac{\theta}{2}\right) \cdot q \tag{3.9}$$

$$q_{(30°)} = \underbrace{\left(\cos\frac{30°}{2}\right)}_{\text{Parte real}} + \underbrace{\left(\sin\frac{30°}{2}\right) \cdot (0.55i + 0.63j - 0.55k)}_{\text{Parte imaginaria}}$$

Cuaterniones unitarios como transformaciones

Si se desea conocer la ubicación de un punto cualquiera (p), después de ser rotado θ°, se procede de la siguiente forma:

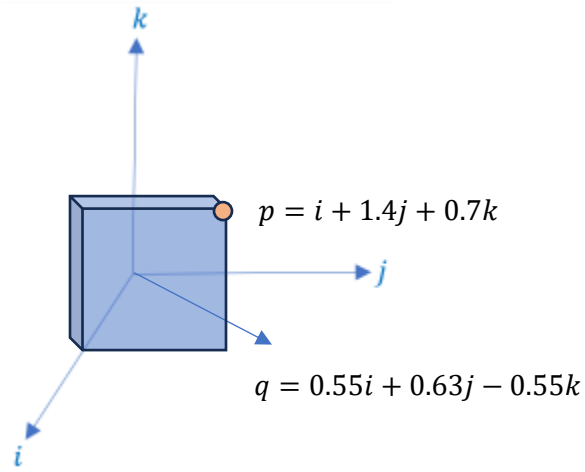


Figura 3.113: Punto representado como cuaternión.

Como las coordenadas del punto están expresadas en cuaterniones, se realiza una multiplicación de cuaterniones, con la siguiente fórmula:

$$p_{(30°)} = q_{(30°)} \cdot p \cdot {q_{(30°)}}^{-1}$$

Es importante resaltar que se multiplica (p) por (q) a la izquierda y por el inverso de (q) a la derecha, con el fin de que el punto experimente una rotación pura y no un escalado, ya que al multiplicar por el inverso la parte real del cuaternión se anula. Expandiendo la multiplicación se tiene:

$$p_{(30°)=}\left(\left(\cos\frac{30°}{2}\right)+\left(\sin\frac{30°}{2}\right)\cdot(0.55i+0.63j-0.55k)\right)\cdot(i+1.4j+0.7k)$$

$$\cdot\left(\left(cos\frac{30°}{2}\right)+\left(sin\frac{30°}{2}\right)\cdot(-0.55i-0.63j+0.55k)\right) \quad (3.10)$$

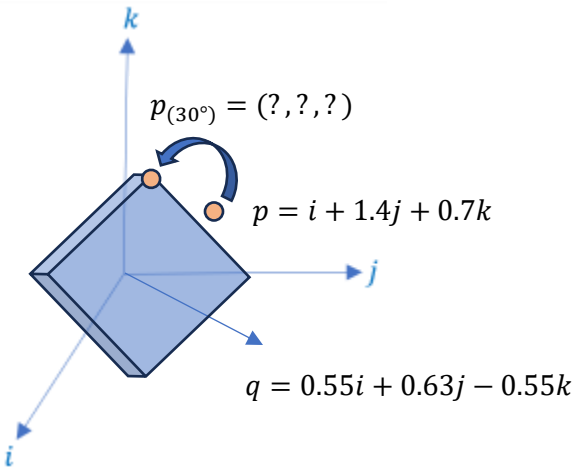


Figura 3.114: Objeto girado 30° sobre el eje de rotación.

Finalmente, el resultado de rotar 30° el punto (p) es:

$$p_{(30°)} = 0 + 1.481i + 0.861j + 0.602k$$

3.8.2. **Orientación del efector final en "Dibujo Libre"**

La aplicación desarrollada en Unity posee opciones en las que el usuario puede manipular el robot, tanto en cinemática directa como en cinemática inversa, ingresando valores por teclado o con *sliders*. Dicho esto, con el fin de facilitar aún más el uso de la herramienta software, se ha desarrollado una opción en la que el usuario puede manipular la posición y orientación del efector final de forma gráfica; es decir, no es necesario determinar las coordenadas o los ángulos de rotación ingresando valores por teclado, sino que arrastrando y soltando dos objetos dentro del juego es suficiente para indicarle al robot hacia dónde debe ir y cómo se debe orientar.

La opción tiene por nombre "Dibujo Libre" y consiste en dos esferas unidas por una línea rosa ("*line renderer*"), que pueden ser manipuladas con el *mouse* dentro del espacio tridimensional del ambiente. Anteriormente, el modo como el robot describía su trayectoria era desplazándose en línea recta (esto es, si no se han creado más esferas dentro del *line renderer*) desde la esfera A hasta la esfera B, con una orientación fija. No obstante, si

se desea usar esta aplicación en el ámbito médico-quirúrgico, es de vital importancia poder manipular la orientación también. Por ende, la nueva forma como se moverá el robot será desplazándose desde el punto A al B, orientando la herramienta acoplada al robot de acuerdo con el vector que se forma entre estos dos puntos (más información en el Anexo A.1).

Ahora bien, como se mencionó previamente, con el propósito de facilitar al usuario la manipulación de la interfaz gráfica, tanto Unity como URSim, utilizan ángulos de Euler para representar la orientación del efector final. Sin embargo, los mensajes que se envían por medio de los nodos de ROS tienen un formato que describe la orientación en cuaterniones; estos mensajes son interpretados posteriormente (convertidos de una estructura de datos a otra) por los controladores del robot o por otros nodos. A causa de esto, los *scripts* desarrollados en Unity encargados del cálculo de la orientación del efector final deben cumplir con este formato al momento de iniciar la transmisión de datos.

El *script* de Unity consiste en hallar la matriz de rotación que transforma el marco de referencia del robot respecto al nuevo marco de referencia; seguido por la conversión de la matriz de rotación a cuaterniones, con el fin de enviar estos valores por medio de ROS hacia el robot o hacia URSim. El procedimiento se detalla a continuación.

La lógica de programación de este modo de control se detalla en el diagrama del Anexo A.3.2.

3.8.2.1. Cálculo de la matriz de rotación

Para el presente ejemplo, se toman valores aleatorios para los puntos A y B.

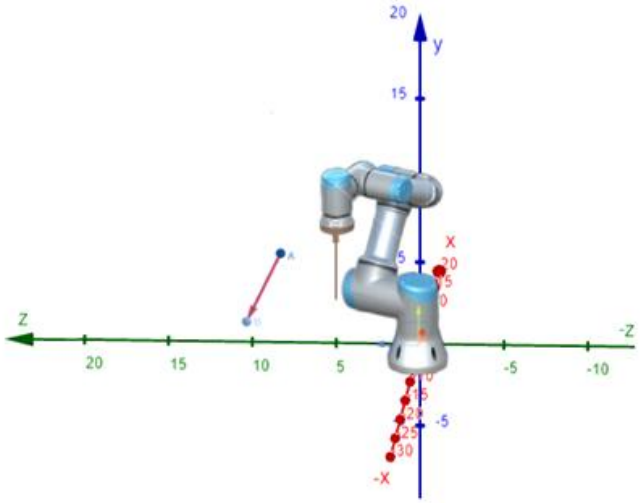


Figura 3.115: Sistema de referencia de Unity respecto al robot.

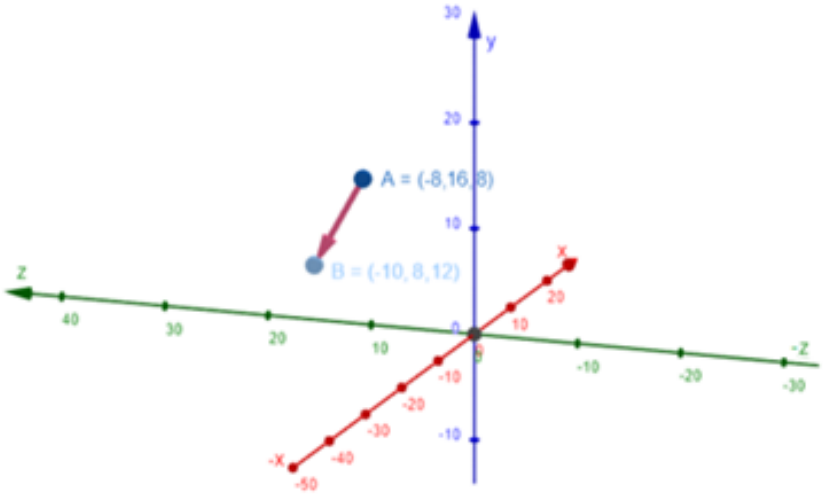


Figura 3.116: Coordenadas de los puntos A y B en Unity.

En primer lugar, se deben transformar las coordenadas de Unity a las coordenadas de ROS, aplicando las siguientes conversiones:

Tabla 3.7: Conversión sistema de referencia Unity-ROS

ROS	*UNITY*
EJE X	EJE -Z
EJE Y	EJE X
EJE Z	EJE Y

Cabe mencionar que el sistema de referencia de Unity es *"left-handed"* con "*y-up*", mientras que el sistema de referencia de ROS es "*right-handed*" con "*z-up*".

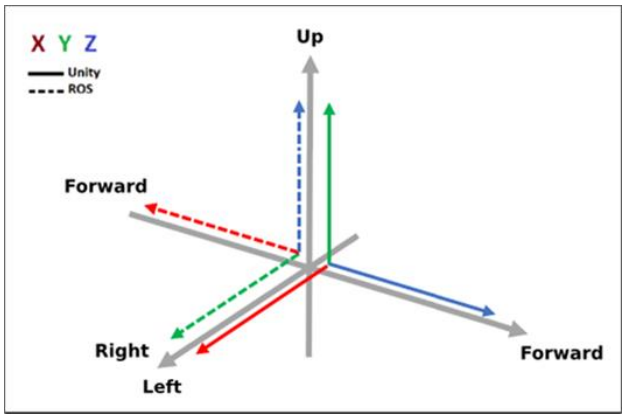


Figura 3.117: Conversión sistema de coordenadas Unity-ROS.

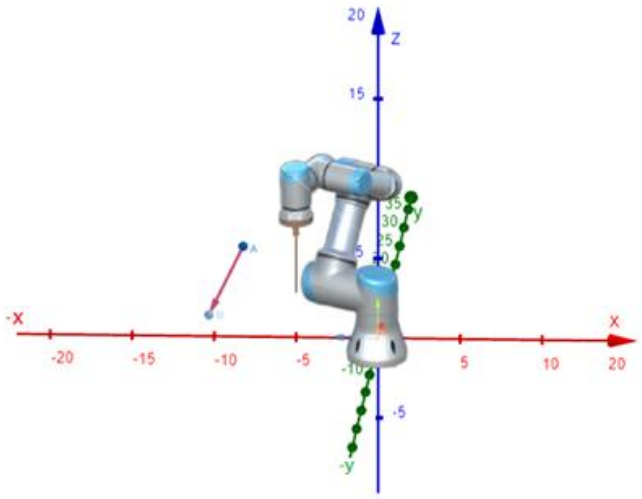


Figura 3.118: Sistema de referencia de ROS respecto al robot.

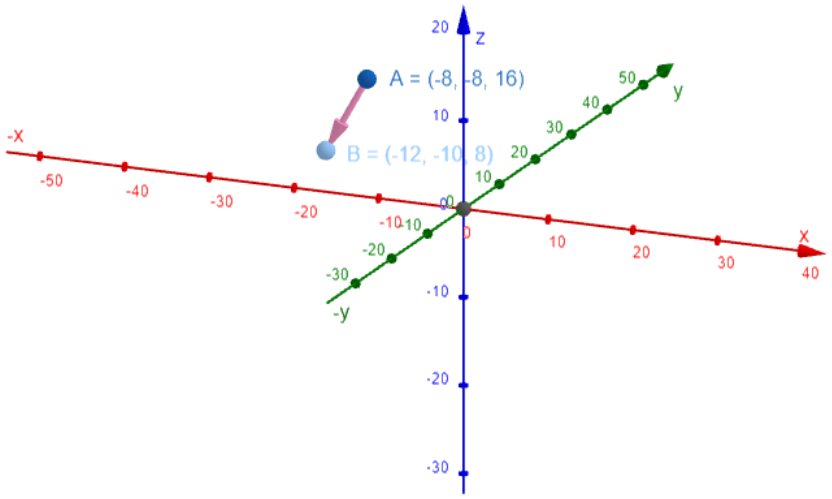


Figura 3.119: Coordenadas de los puntos A y B en ROS.

Ahora, para hallar el vector que se genera entre A y B, se deben restar las coordenadas de B menos las coordenadas de A. La diferencia de dos puntos en el espacio es equivalente a la diferencia entre los vectores que se forman desde el origen hasta A y B respectivamente [123].

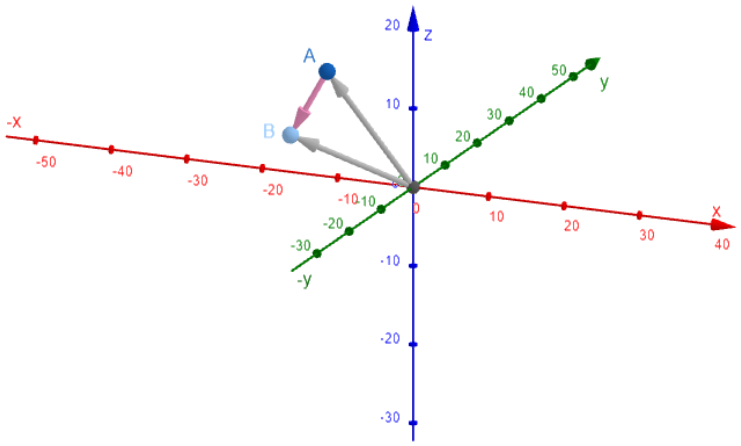


Figura 3.120: Vectores A y B.

El resultado es un vector con la misma magnitud y dirección del vector entre A y B, pero ubicado desde el origen.

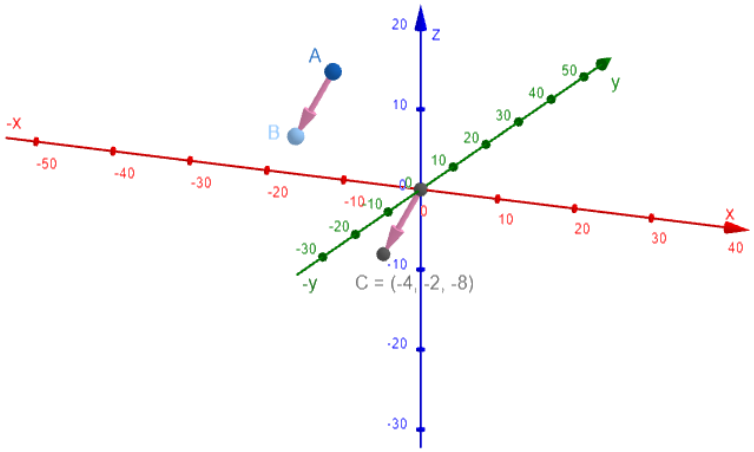


Figura 3.*121*: Diferencia entre vectores A y B.

A continuación, se procede a normalizar el vector hallado en el paso anterior (la suma de sus componentes al cuadrado debe ser igual a uno). Para ello, primero se encuentra la longitud o magnitud del vector:

$$\left|\vec{C}\right| = \sqrt{x^2 + y^2 + z^2} = \sqrt{(-4)^2 + (-2)^2 + (-8)^2} = 9.165$$

Segundo, se encuentra la dirección:

$$\vec{u} = \frac{\vec{c}}{|\vec{c}|} = -\frac{4}{9.165} - \frac{2}{9.165} - \frac{8}{9.165} = \begin{pmatrix} -0.436 \\ -0.218 \\ -0.873 \end{pmatrix} \tag{3.11}$$

El vector unitario correspondiente es:

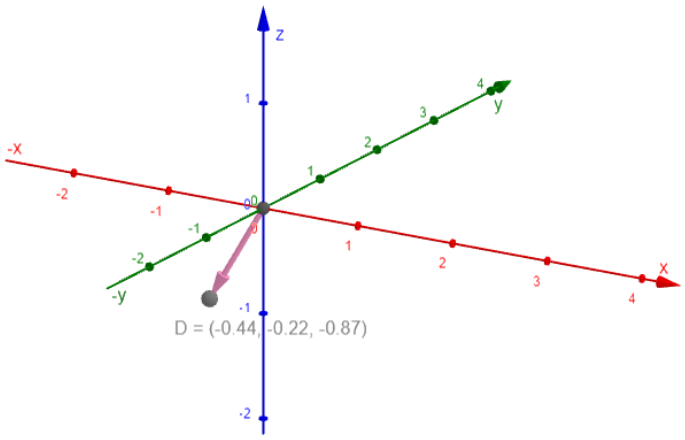


Figura 3.122: Vector C normalizado.

Desde este punto en adelante, se empieza a generar la matriz de rotación a partir de los vectores calculados. Cabe destacar que todos los vectores que componen esta matriz deben estar normalizados con el fin de representar una rotación pura. Además, es de vital importancia entender la relación entre la posición y orientación del efector final o TCP (*tool center point*) y el sistema de coordenadas de la base del robot. El TCP del robot UR3e, está orientado de la siguiente manera:

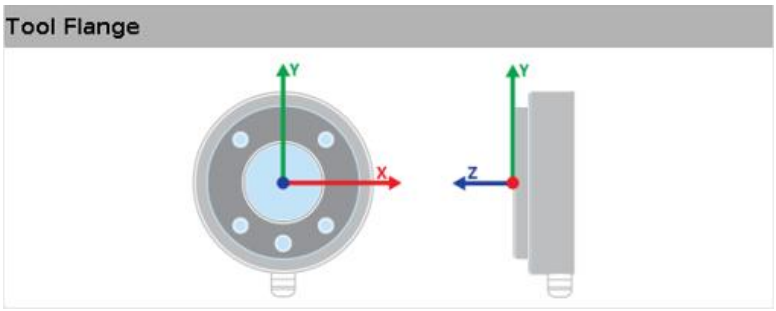


Figura 3.*123*: Sistema de coordenadas del efector final.

Como se puede observar, el eje Z es perpendicular a la cara exterior del TCP o, en otras palabras, se proyecta hacia afuera del TCP de forma concéntrica. Entonces, el vector diferencia hallado y normalizado, corresponderá al nuevo eje Z o "*Forward*" del efector final.

El siguiente paso es definir el vector que corresponderá al eje Y o "*Up*". Teniendo en cuenta el sistema de referencia de ROS, el vector corresponde a las coordenadas (0,0,1). Este vector no es perpendicular al vector "*Forward*", por lo que solo es un vector auxiliar para encontrar el vector "*Right*".

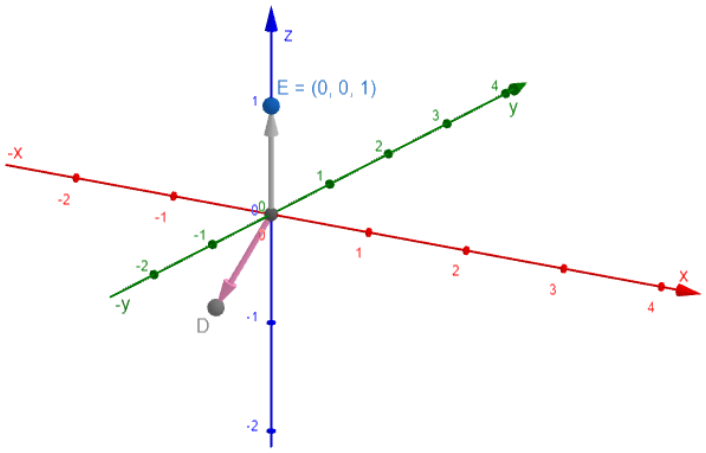


Figura 3.124: Vector "Up" auxiliar.

Con los vectores “Forward” y “Up”, se procede a realizar el producto cruz entre ambos. Es necesario recordar que todos los vectores que componen un sistema de coordenadas deben ser ortogonales entre sí.

$$(0,0,1) \times (-0.436, -0.218, -0.873) =$$

$$(0 \cdot (-0.873) - 1 \cdot (-0.218), 1 \cdot (-0.436) - 0 \cdot (-0.873), 0 \cdot (-0.218) - 0 \cdot (-0.436)) =$$

$$\begin{pmatrix} 0.218 \\ -0.436 \\ 0 \end{pmatrix} \tag{3.12}$$

El resultado corresponde al vector X o “Right”:

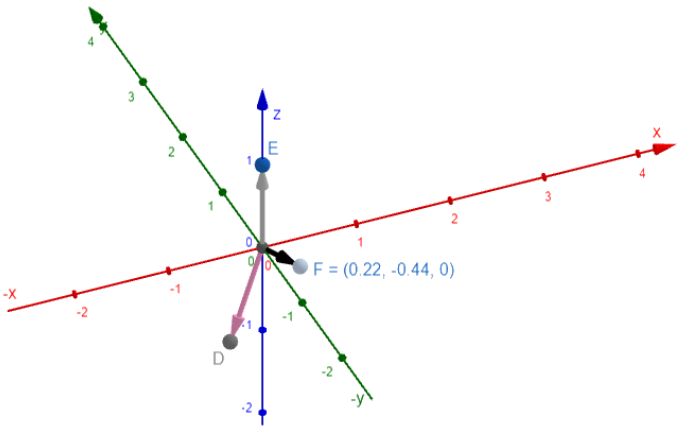


Figura 3.125: Vector “*Right*”.

Como el producto cruz fue realizado entre dos vectores que no eran ortogonales entre sí, el vector resultante no es un vector unitario, por lo que debe ser normalizado:

Longitud o magnitud del vector:

$$\left|\vec{F}\right| = \sqrt{x^2 + y^2 + z^2} = \sqrt{(0.218)^2 + (-0.436)^2 + (0)^2} = 0.487$$

Dirección:

$$\vec{u} = \frac{\vec{F}}{\left|\vec{F}\right|} = \frac{0.218}{0.487} - \frac{0.436}{0.487} + \frac{0}{0.487} = \begin{pmatrix} 0.447 \\ -0.894 \\ 0 \end{pmatrix} \tag{3.13}$$

El vector unitario correspondiente es:

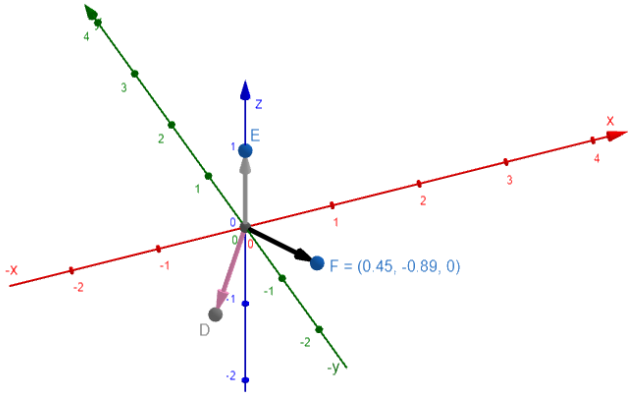


Figura 3.126: Vector F normalizado

Finalmente, con los vectores "*Forward*" y "*Right*" se realiza el producto cruz para encontrar el verdadero vector "*Up*". Como estos vectores ya están normalizados y son perpendiculares entre sí, no es necesario normalizar el vector resultante.

$$(-0.436, -0.218, -0.873) \times (0.447, -0.894, 0) =$$

$$(-0.218 \cdot 0 - (-0.873(-0.894)), -0.873 \cdot 0.447 - (-0.436 \cdot 0), -0.436(-0.894) - (-0.218 \cdot 0.447)) =$$

$$\begin{pmatrix} -0.780462 \\ -0.390231 \\ 0.48723 \end{pmatrix} \tag{3.14}$$

El vector unitario correspondiente es:

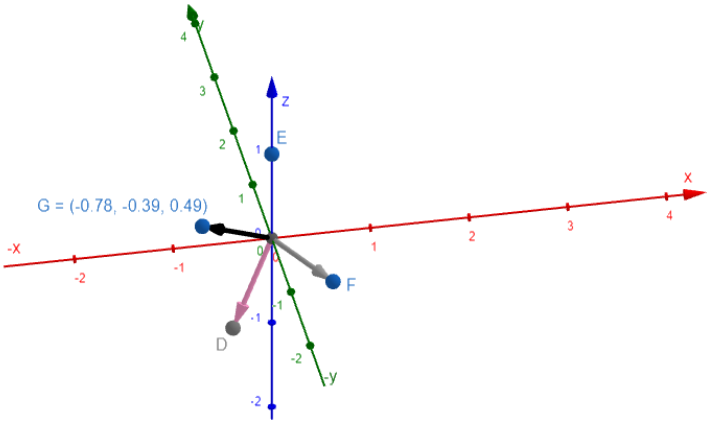


Figura 3.127:Vector "*Up*".

De esta manera se tienen todos los vectores necesarios para generar la matriz de rotación, que proporcionará la nueva orientación del efector final. El orden en que se ubican los vectores dentro de la matriz es muy importante; este depende del sistema de referencia del TCP (Figura 3.123).

Tabla 3.8: Matriz de rotación.

X	Y	Z	
0.447	−0.894	0	***Rightward***
−0.780462	−0.390231	0.48723	***Upward***
−0.436	−0.218	−0.873	***Forward***

3.8.2.2. Cuaternión equivalente

Ahora, si se tiene un cuaternión (normalizado) de la forma:

$$q = qw + qx\,\boldsymbol{i} + qy\,\boldsymbol{j} + qz\,\boldsymbol{k}, \tag{3.15}$$

la matriz que representaría la misma rotación es:

Tabla 3.9: Conversión de cuaternión a matriz de rotación.

$\boldsymbol{m00} = 1 - 2qy^2 - 2qz^2$	$\boldsymbol{m01} = 2qx \cdot qy - 2qz \cdot qw$	$\boldsymbol{m02} = 2qx \cdot qz + 2qy \cdot qw$
$\boldsymbol{m10} = 2qx \cdot qy + 2qz \cdot qw$	$\boldsymbol{m11} = 1 - 2qx^2 - 2qz^2$	$\boldsymbol{m12} = 2qy \cdot qz - 2qx \cdot qw$
$\boldsymbol{m20} = 2qx \cdot qz - 2qy \cdot qw$	$\boldsymbol{m21} = 2qy \cdot qz + 2qx \cdot qw$	$\boldsymbol{m22} = 1 - 2qx^2 - 2qy^2$

Se desea a partir de la Tabla 3.9, encontrar las ecuaciones más adecuadas para despejar los valores de los coeficientes que acompañan los términos del cuaternión. Considerando la sensibilidad a errores de redondeo debido a las operaciones en coma flotante del ordenador y que la matriz contiene información redundante, una mal escogencia del método para hallar el cuaternión podría resultar no solo en la "des-ortogonalización" de la matriz, sino en una representación poco exacta de la orientación (recordar que el rango de los coeficientes de un cuaternión está entre 1 y -1) [124].

En primer lugar, se definen cuatro ecuaciones que, a partir de los términos de la diagonal principal y la ecuación del cuaternión normalizado se define:

$$qx^2 + qy^2 + qz^2 + qw^2 = 1, \tag{3.16}$$

se calcula uno de los cuatro coeficientes del cuaternión (qw, qx, qy, o qz):

$$qw = \pm {}^1\!/_2 \sqrt{1 + m00 + m11 + m22}$$

$$qx = \pm {}^1\!/_2 \sqrt{1 + m00 - m11 - m22}$$

$$qy = \pm {}^1\!/_2 \sqrt{1 + m11 - m00 - m22} \tag{3.17}$$

$$qz = \pm {}^{1}/_{2}\sqrt{1 + m22 - m00 - m11}$$

Para sortear los problemas antedichos, se escoge el valor que evita la raíz cuadrada de cero o números negativos; en su defecto, se opta por el resultado con el valor más alto. Posteriormente, se hallan los demás coeficientes del cuaternión en base a los términos afuera de la diagonal principal. En consecuencia, se tienen 4 métodos para encontrar el cuaternión equivalente, estos se describen en la siguiente tabla:

Tabla 3.10: Conversión de matriz de rotación a cuaternión.

Se encuentra qw (de diagonal principal)	Se encuentra qx (de diagonal principal)	Se encuentra qy (de diagonal principal)	Se encuentra qz (de diagonal principal)
$qw = \frac{\pm\sqrt{1 + m00 + m11 + m22}}{2}$	$qw = \frac{m21 - m12}{4qx}$	$qw = \frac{m02 - m20}{4qy}$	$qw = \frac{m10 - m01}{4qz}$
$qx = \frac{m21 - m12}{4qw}$	$qx = \frac{\pm\sqrt{1 + m00 - m11 - m22}}{2}$	$qx = \frac{m10 + m01}{4qy}$	$qx = \frac{m02 + m20}{4qz}$
$qy = \frac{m02 - m20}{4qw}$	$qy = \frac{m10 + m01}{4qx}$	$qy = \frac{\pm\sqrt{1 + m11 - m00 - m22}}{2}$	$qy = \frac{m21 + m12}{4qz}$
$qz = \frac{m10 - m01}{4qw}$	$qz = \frac{m02 + m20}{4qx}$	$qz = \frac{m21 + m12}{4qy}$	$qz = \frac{\pm\sqrt{1 + m22 - m00 - m11}}{2}$

De la matriz de rotación obtenida en la Tabla 3.10, es evidente que el valor más alto que se puede obtener de la diagonal principal es:

$$qx = \pm {}^{1}/_{2}\sqrt{1 + m00 - m11 - m22}$$

$$qx = \frac{\pm\sqrt{1+0.447-(-0.390231)-(-0.873)}}{2} \tag{3.18}$$

$$qx = 0.8231$$

Cabe señalar que es posible escoger el valor negativo de la raíz, ya que para una misma orientación pueden existir dos cuaterniones.

Los términos restantes son:

$$qw = \frac{m21 - m12}{4qx} = \frac{-0.218 - 0.48723}{4(0.8231)} = -0.2141$$

$$qy = \frac{-0.780462+(-0.894)}{4(0.8231)} = -0.5085 \tag{3.19}$$

$$qz = \frac{0 + (-0.436)}{4(0.8231)} = -0.1324$$

Por último, el cuaternión resultante es:

$$q = -0.2141 + 0.8231i - 0.5085j - 0.1324k$$

Después de ser calculado, el cuaternión y las coordenadas de la esfera son enviados por medio de un mensaje de ROS hacia el controlador del robot, que se encargará de resolver la cinemática inversa para ubicar el efector final en la posición y dirección deseada.

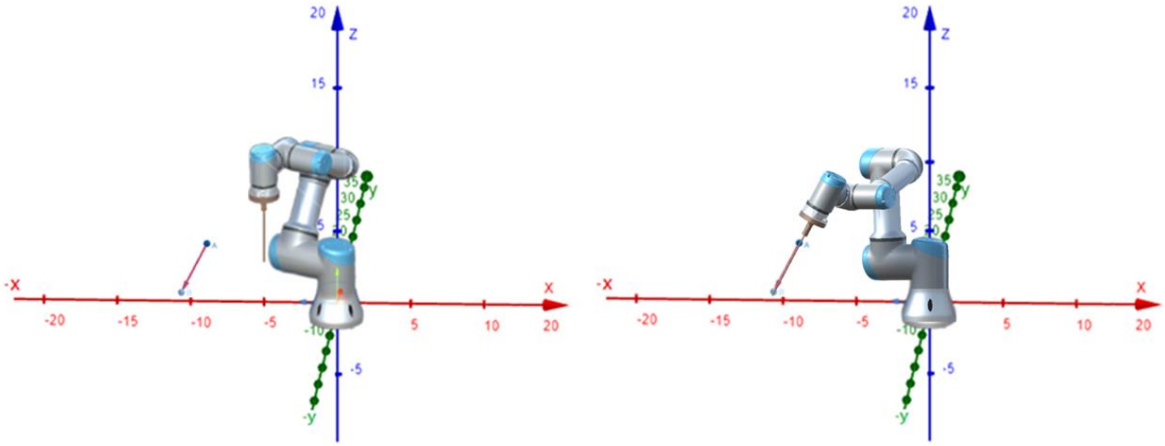


Figura 3.128: Movimiento del robot al cargar la trayectoria.

Nota:

El proceso para hallar la matriz de rotación y el cuaternión equivalente está implementado dentro de una función de Unity (*Quaternion.LookRotation*). Está función recibe dos argumentos, un vector que indica la dirección "*Forward*" y otro que indica la dirección "*Up*"; y regresa una rotación en forma de cuaternión [125]. Al no tener acceso al código fuente de Unity, se presume que el proceso como se realizan los cálculos es bastante aproximado a lo descrito en esta sección; no obstante, pueden presentarse variaciones debido al producto cruz de Unity ("*left hand rule*") [126], o al orden de la matriz de rotación, entre otros.

4. Resultados

Con el propósito de enfocar la aplicación como un dispositivo de neuronavegación, se obtienen dos resultados principales que complementan el trabajo desarrollado anteriormente, particularmente en el modo de control “Dibujo Libre”. Uno es manipular cámaras virtuales desde diferentes ángulos y el otro es controlar la orientación del efector final del robot. De este modo, es posible crear trayectorias con gran precisión al interior de representaciones 3D de partes de la anatomía humana, como un cráneo, en este caso específico.

En primer lugar, se dispuso de 9 cámaras que apuntan en dirección al cráneo virtual, creando vistas desde perspectivas y ángulos diferentes, utilizando una *suite* de herramientas (*Cinemachine*) que ofrece cámaras dinámicas e inteligentes. Cada cámara virtual tiene una tarea independiente; algunas son fijas, otras tienen *zoom*. Estas últimas pueden ser manipuladas por periféricos como el *scroll* del mouse. Además, propiedades como el *near* y el *far* de algunas cámaras fueron modificadas, permitiendo realizar cortes de tipo sagital, transversal o coronal del cráneo, al tener control de la distancia a renderizar (ver anexo A.1.3).

Por otro lado, el control de movimiento del robot en “Dibujo Libre” está determinado por las posiciones de dos marcadores en forma de esferas, que el usuario puede manipular con el mouse para definir la trayectoria del robot de forma visual. Desde un enfoque neuroquirúrgico, estos marcadores pueden concebirse como un punto de acceso y un punto de llegada del efector final o como un marcador de entrada y un marcador objetivo. Con las nuevas modificaciones, el efector final no solo se desplaza desde una posición a otra, sino que también se orienta respecto al vector que se forma entre estos dos marcadores. De esta forma, diferentes ubicaciones del marcador de entrada o del marcador objetivo generan diferentes orientaciones del efector final. Además, esta opción cuenta con dos modos automáticos: el modo 1 que consiste en un movimiento lineal entrada-objetivo, más un movimiento adicional de seguridad en sentido objetivo-entrada; y el modo 2 que consiste en un movimiento angular tomando como punto de pivote el marcador de entrada (ver anexo A.1.4 y A.1.6).

Adicionalmente, se logró implementar una herramienta analóga (endoscopio), con la que se pueden recorrer trayectorias al interior de corredores estrechos como los orificios nasales del *phantom*.

4.1. Trayectorias endonasales

Gracias a las características mencionadas y al proceso de registro del *phantom* con el cráneo virtual, se realizaron cortes del cráneo virtual para ubicar de forma precisa el marcador objetivo al interior de este, con el fin de ejecutar diferentes trayectorias (Figura 4.1). Se recrearon 10 trayectorias transesfenoidales transnasales; 6 pasando por el orificio nasal derecho y 4 pasando por el orificio nasal izquierdo, que luego se convirtieron a archivos .json para guardar las 50 posiciones, desde el marcador inicial hasta el marcador final, de cada trayectoria.

Video: https://youtu.be/2GFgaUTg0pg

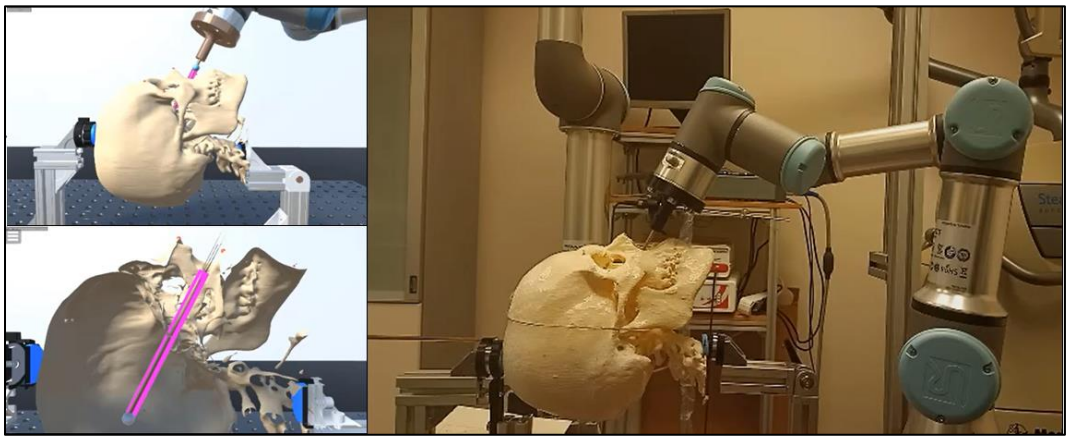


Figura 4.1: UR3e y gemelo digital ejecutando una trayectoria endonasal.

Las trayectorias se lograron gradualmente, intercambiando vistas en el plano sagital y coronal principalmente. Primero se definió el punto de acceso a la fosa nasal correspondiente, para después manipular el marcador objetivo recorriendo pequeñas distancias hacia el interior del cráneo, asegurándose de no comprometer las estructuras circundantes del *phantom*. El límite del movimiento rotacional de las trayectorias son los vértices del área del orificio hecho en la pared anterior del seno esfenoidal; mientras que el límite del movimiento lineal es la pared interior del hueso parietal. Una vez concluido el

recorrido completo de la trayectoria, se guardaba en un archivo y se procedía con la siguiente.

El siguiente paso consiste en ingresar a Matlab los archivos donde están consignadas las posiciones de las trayectorias, con el fin de representarlas en un plot tridimensional. Esto permite observarlas en conjunto de manera más clara y verificar conjeturas al respecto. Para ello primero se deben invocar los archivos de las trayectorias, para cargarlas al *workspace* de Matlab. Luego, y esto es importante, se debe realizar una conversión del sistema de coordenadas de Unity al sistema de coordenadas de Matlab, con las siguientes convensiones:

Tabla 4.*1*: Conversión sistema de referencia Matlab-Unity

MATLAB	*UNITY*
EJE X	EJE X
EJE Y	EJE Z
EJE Z	EJE Y

Y finalmente, se dibujan las trayectorias en un *plot* 3D con las respectivas leyendas. El resultado es el siguiente para ambas fosas nasales:

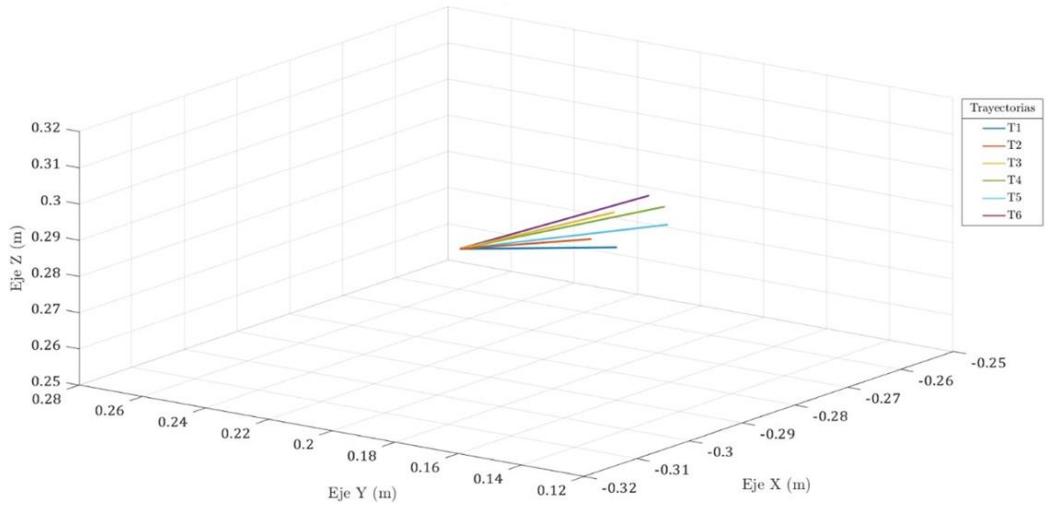


Figura 4.2: Trayectorias endonasales de fosa nasal derecha.

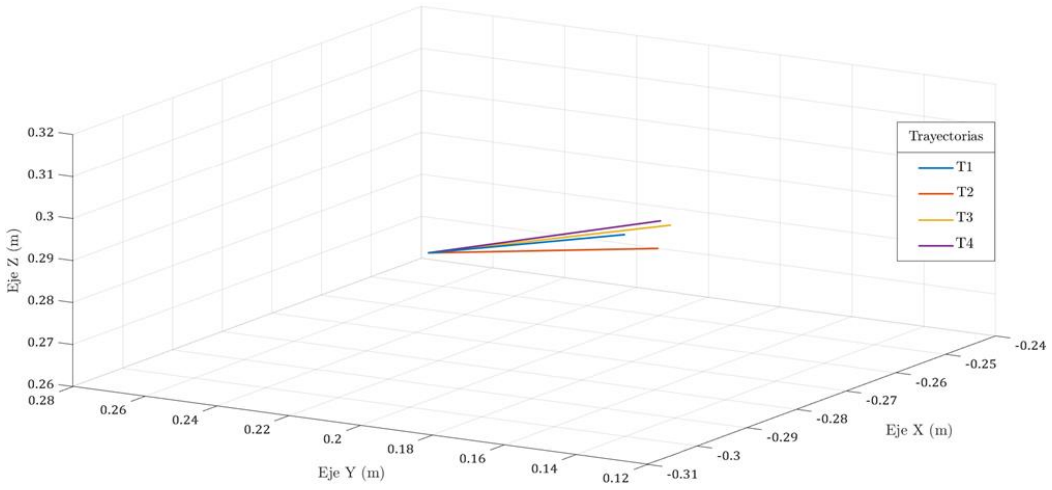


Figura 4.3: Trayectorias endonasales de fosa nasal izquierda.

En la figura 4.2 y 4.3, es evidente el punto de acceso (marcador de entrada) a la fosa nasal derecha e izquierda, donde pivotean todas las trayectorias. Este movimiento angular es posible al manipular únicamente el marcador objetivo al interior de la cavidad craneal, después de completar cada trayectoria. Se puede advertir que estos movimientos no son muy amplios, lo que indicaría un espacio de trabajo bastante reducido. El volumen que generan estas trayectorias es una pirámide hexagonal irregular en la fosa nasal derecha y una pirámide rectangular irregular en la fosa nasal izquierda.

Una vez se han cargado las trayectorias de manera exitosa, se procede a cargar el objeto 3D del cráneo al *workspace* (Figura 4.4). Las coordenadas de posición del cráneo también deben ser convertidas al sistema de coordenadas de Matlab, de manera que las trayectorias coincidan exactamente con las posiciones correspondientes al interior de este. La razón por la que el cráneo 3D aparece inclinado en el *plot*, se debe al sistema de coordenadas original (Unity). Por otro lado, Matlab sólo acepta archivos en formato de impresión 3D (.stl), motivo por el que anteriormente se debió realizar una conversión de formato en Blender.

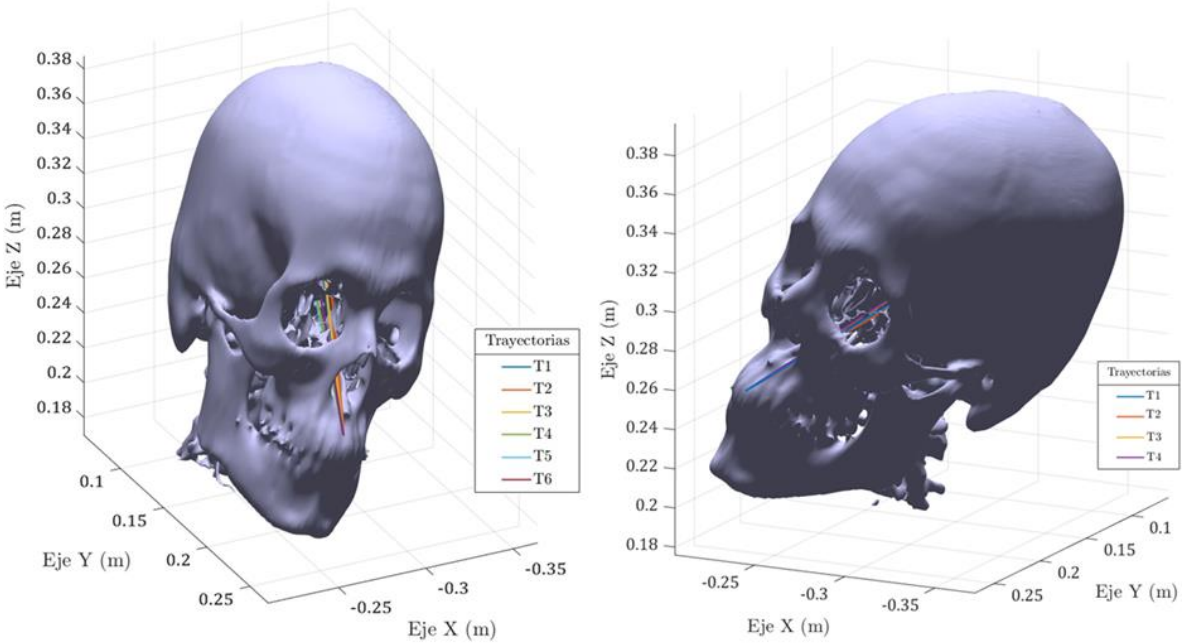


Figura 4.4: Trayectorias endonasales de fosa nasal derecha e izquierda en el cráneo.

Como se puede observar, las trayectorias tienen un nombre y un correspondiente color para ser fácilmente identificables. Gracias a las propiedades del *plot* de Matlab, es posible analizar las trayectorias en conjunto o individualmente.

A continuación, se realizó un corte coronal del cráneo en Matlab, por medio de la modificación del ángulo de visualización del *plot* (Figura 4.5). El corte coronal y el corte sagital serán representados en dos dimensiones, haciendo un eje igual a cero.

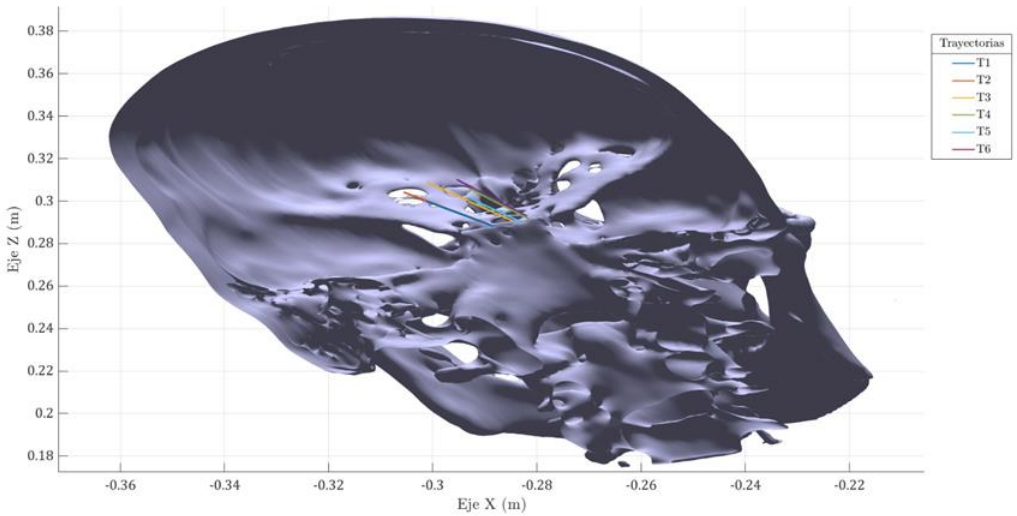


Figura 4.5: Plano coronal trayectorias endonasales de fosa nasal derecha en el cráneo.

Del plano coronal se destaca el área del seno esfenoidal que debió perforarse en el *phantom* para poder ingresar el endoscopio. Si se desea eliminar esta área en el cráneo virtual, se debe realizar de nuevo el proceso de segmentación en Slicer 3D; sin embargo, el problema realmente sería lograr que ambos orificios coincidan en tamaño y forma.

Finalmente, se realizó un corte sagital del cráneo virtual, con el fin de observar el alcance de las trayectorias a las diferentes áreas del encéfalo. Para ello, se utilizó una imagen 2D de un corte sagital del encéfalo y con un programa de edición de imágenes, se montó encima el corte hecho en Matlab. Se trató, en la medida de lo posible, hacer coincidir las estructuras anatómicas de la imagen con las del cráneo virtual; aunque lo más adecuado en este caso, es generar un modelo 3D del encéfalo del mismo paciente a partir de un MRI, para tener un modelo virtual completo de toda la cabeza. De esta forma, se podrían planificar las trayectorias en la etapa preoperatoria y observar en tiempo real la ejecución de las mismas en la etapa operatoria.

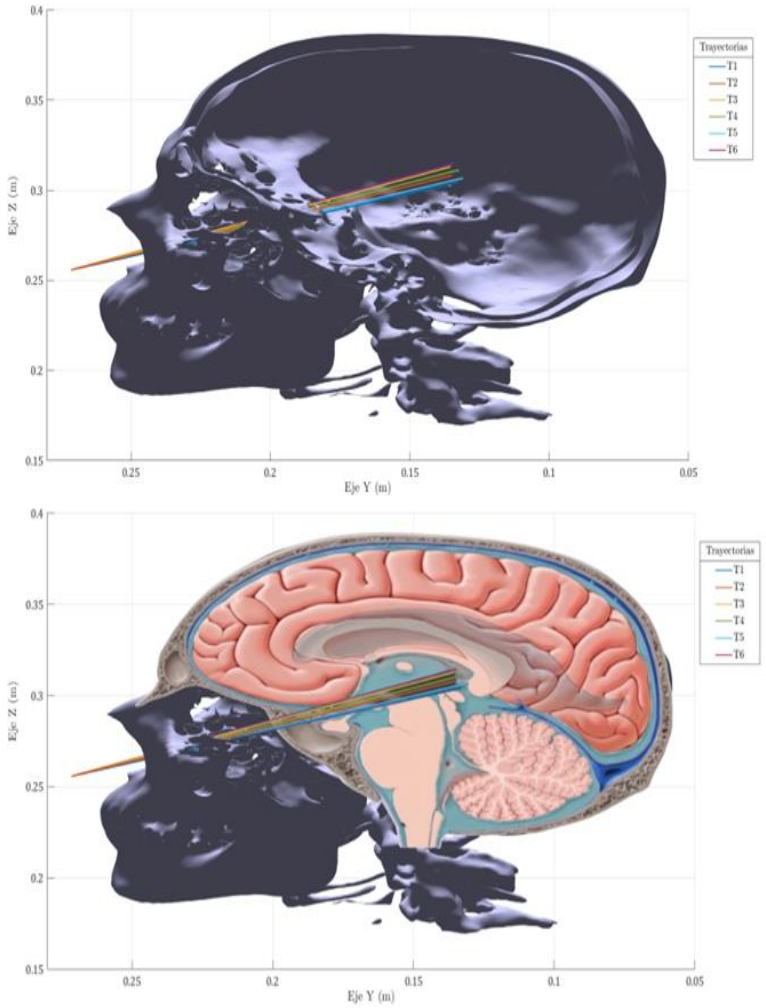


Figura 4.6: Plano sagital trayectorias endonasales de fosa nasal derecha con encéfalo.

En la figura 4.6, es apreciable que las trayectorias del orificio nasal derecho tienen un alcance hacia la región antero-medial del encéfalo; sin embargo, ninguna de las 6 trayectorias entra en contacto con la hipófisis, razón por la que deben ser descartadas en caso de que el objetivo haya sido intervenir esta glándula. Por otro lado, si el objetivo era intervenir el tronco encefálico en su parte más superior (mesencéfalo), las trayectorias T1, T5 y T6 son las más adecuadas.

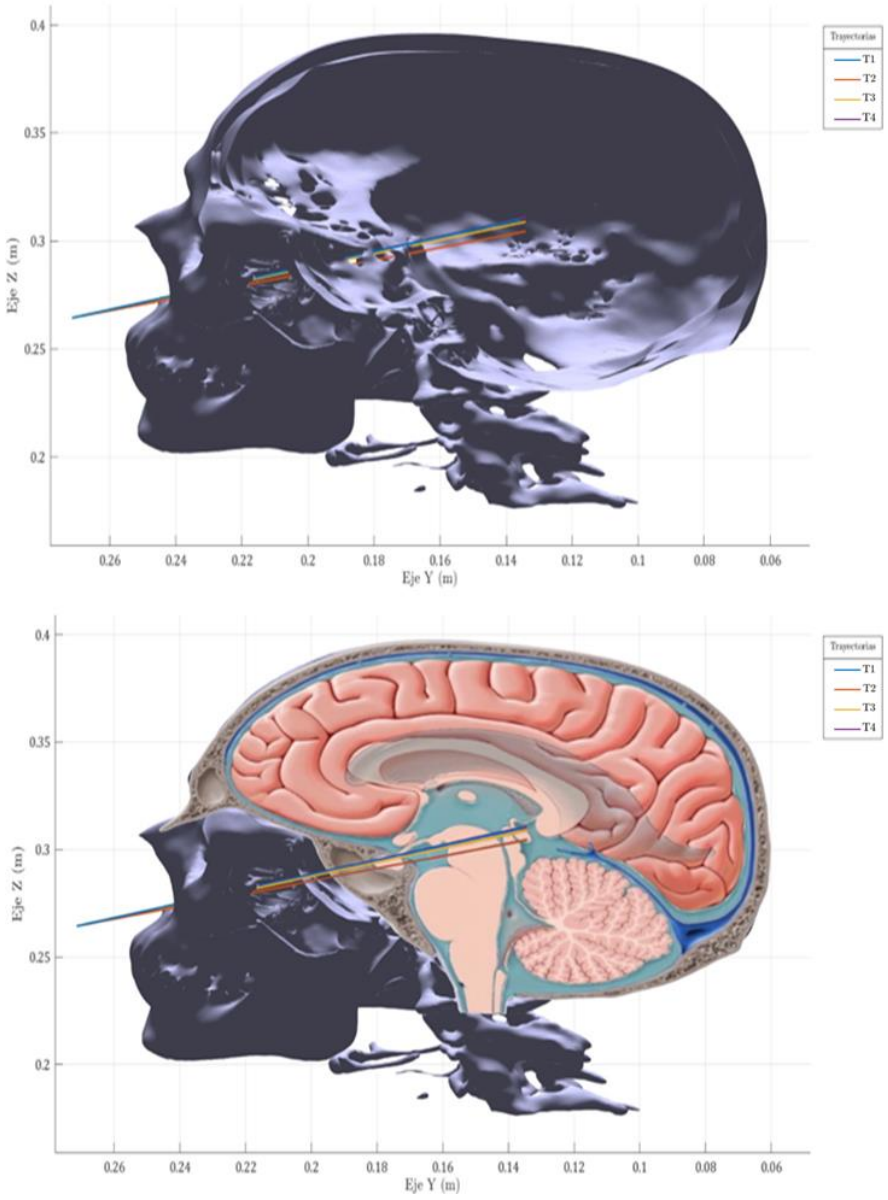


Figura 4.7: Plano sagital trayectorias endonasales de fosa nasal izquierda con encéfalo.

Por otro lado, en la figura 4.7 es apreciable que las trayectorias del orificio nasal izquierdo también tienen un alcance hacia la región antero-medial del encéfalo; sin embargo, en este caso las 4 trayectorias entran en contacto con la hipófisis, razón por la que son las más indicadas para intervenir esta glándula. Además, si el objetivo era intervenir el tronco encefálico en su parte más superior (mesencéfalo), estas trayectorias también son adecuadas.

En todo caso, las trayectorias serán escogidas por los especialistas (neurocirujano y otorrinolaringólogo), para crear el corredor más seguro que permita la intervención del área afectada al interior del paciente.

4.2. Trayectorias endonasales con inclinación de *phantom*

En esta última prueba se quería evaluar la utilidad de la aplicación para realizar trayectorias endonasales cambiando la orientación del *phantom* 30°. Para ello, se realizó el registro del *phantom* con el cráneo virtual en esta nueva posición. De la misma forma como se explicó en la Sección 3.7.2, el registro se debe hacer de forma manual ubicando el endoscopio real en la fiducia del *phantom* y, con los *gizmos* de posición y orientación del cráneo virtual, manipularlo de tal forma que la fiducia virtual coincida con la punta del endoscopio virtual.

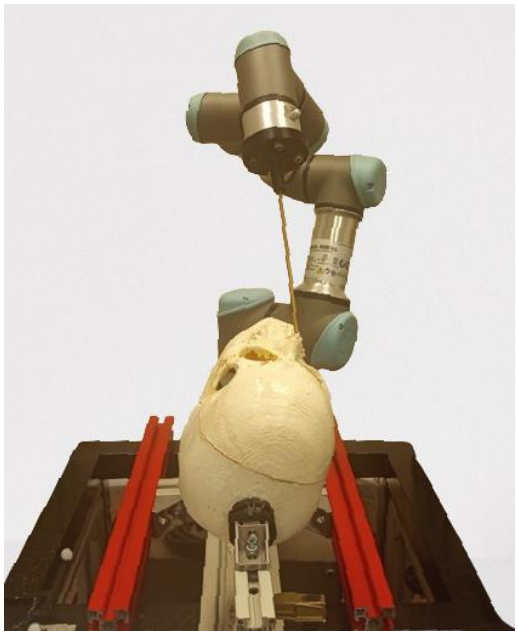


Figura 4.8: Registro de fiducias del *phantom* inclinado.

Una vez completado el proceso de registro, se confirmó que las cámaras se reubican de forma automática cada vez que se modifica la posición o la orientación del cráneo virtual. De este modo, los cortes sagitales continúan alineados con los planos anatómicos de este.

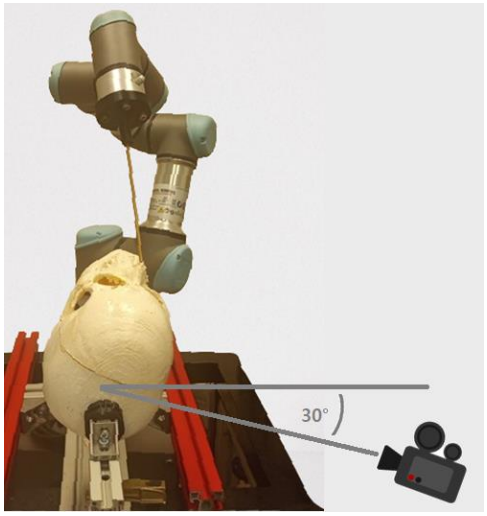


Figura 4.9: Reajuste de posición y orientación de cámaras en Unity.

En este caso no se vio la necesidad de hallar el error de registro, ya que no se tendría otro método con el cual compararlo (no es posible conocer la orientación del *phantom* con el cálculo del *bounding box*).

Acto seguido, se realizaron diferentes pruebas con la aplicación para verificar que, sin importar la orientación del cráneo, el robot puede ejecutar trayectorias endonasales sin problema. De la misma forma que en los casos anteriores, los límites de la cavidad nasal del *phantom* deben ser tenidos en cuenta durante la planeación de las trayectorias.

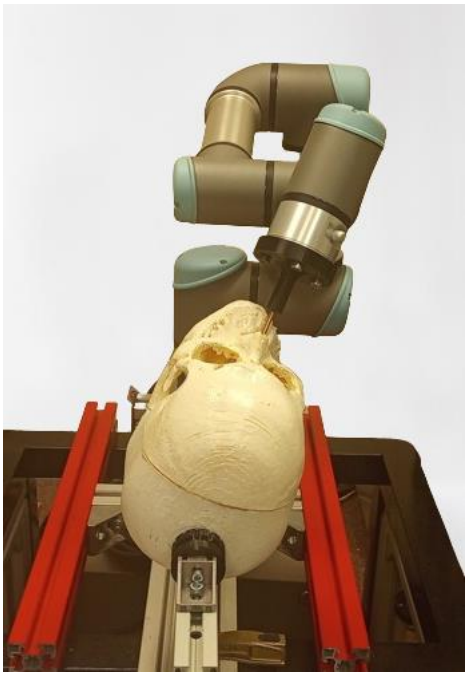


Figura 4.10: Trayectoria endonasal con cráneo inclinado.

Al final se obtuvieron 5 trayectorias endonasales en el orificio nasal derecho, con el *phantom* rotado 30°.

Video: https://www.youtube.com/watch?v=0aMxlJRIzvY

4.3. Error de trayectorias

Para evaluar el error de las trayectorias planificadas en la plataforma en comparación con las trayectorias obtenidas por el robot real, se utiliza la medición euclidiana entre lo que se envía desde Unity, y los datos que se captan del efector final del robot utilizando el tópico */tf* de ROS, el cual publica constantemente las posiciones y orientaciones del órgano terminal del robot. Luego, para determinar el error máximo de cada trayectoria se hace uso de un algoritmo de Matlab.

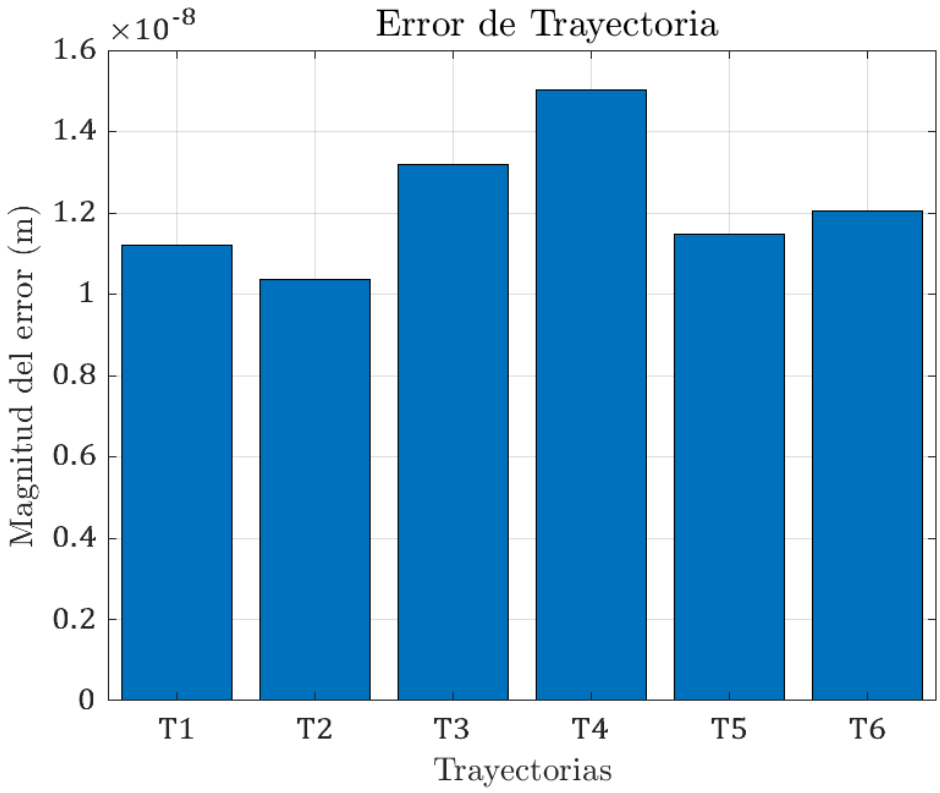


Figura 4.11: Error de trayectoria fosa nasal derecha.

En la figura 4.11 se puede evidenciar que el error máximo de cada trayectoria oscila entre $1.03\ X\ 10^{-8}m$ a $1.5\ X\ 10^{-8}m$, aunque en algunos casos pueda no ser muy precisa, da una idea de cómo es el del error en las trayectorias que se ejecutan en la plataforma en

comparación con las que realiza el robot real. Aproximadamente el máximo error es de 0,15 micrómetros, que se evidencia en la trayectoria endonasal T4.

4.4. Repositorio del proyecto

Los resultados de este trabajo de investigación se encuentran en el repositorio del proyecto anterior como la versión final de este. Esta contribución consiste en una aplicación software (proyecto editable más ejecutable), el manual de usuario, las guías de usuario en video y una carpeta con las trayectorias endonasales (videos de referencia más archivos JSON). Los archivos correspondientes al modelo 3D del cráneo y del endoscopio se encuentran dentro de la carpeta *assets* del proyecto editable (Figura 4.12). https://github.com/alvira13/PlataformaUR3_V8

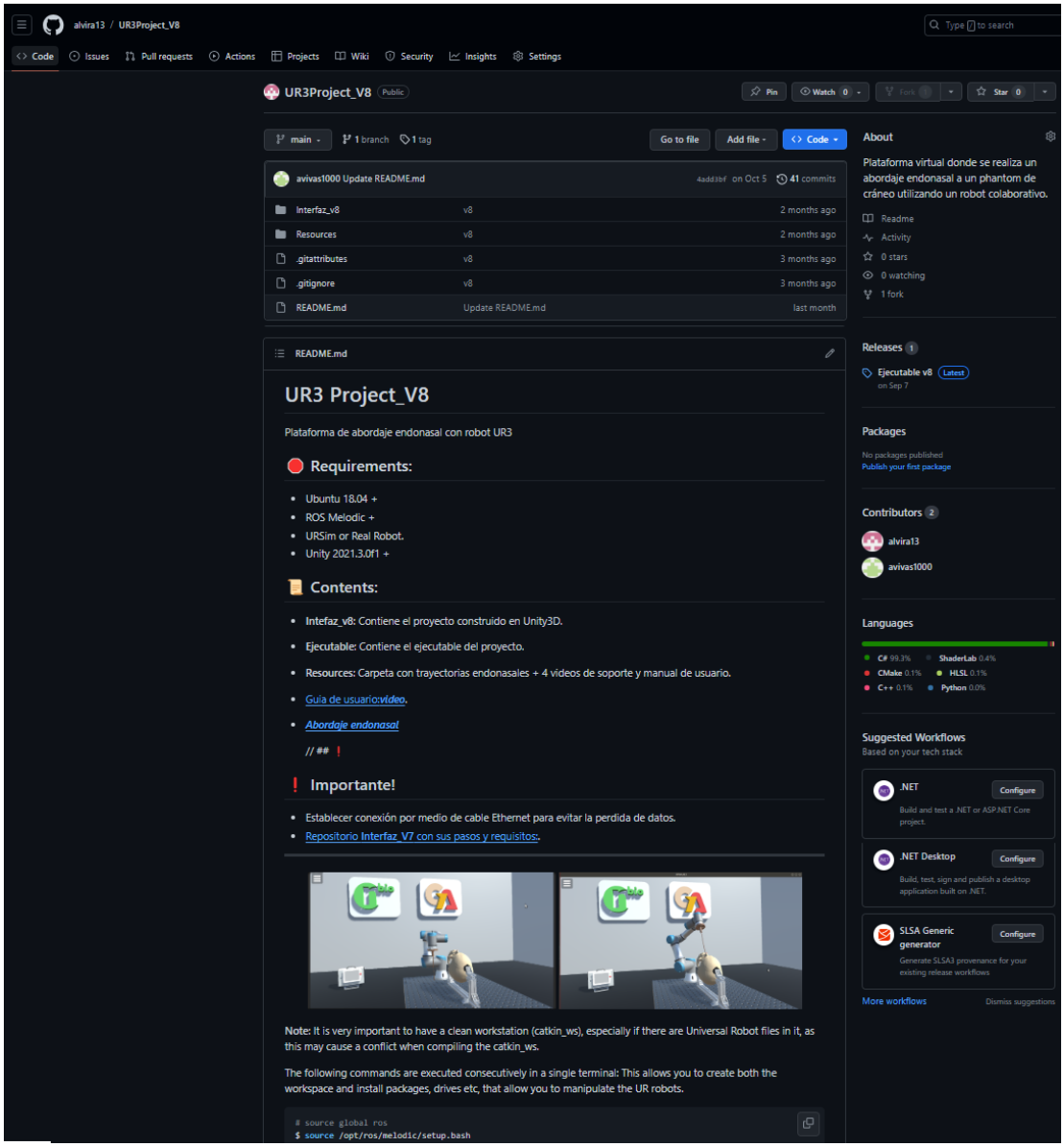


Figura 4.12: Repositorio en GitHub.

5. Conclusiones y trabajos futuros

5.1. Conclusiones

Gracias al convenio interinstitucional establecido entre la Universidad del Cauca (Popayán, Colombia), y la Universidad Miguel Hernández (Elche, España), se realizó una estancia de investigación de tres meses en los laboratorios del grupo nBio. Durante este periodo se llevaron a cabo actividades que permitieron utilizar una plataforma de simulación y control de un robot UR3e, como un dispositivo de navegación guiado por imágenes médicas, cambiando el foco de la plataforma de la ingeniería hacia la bioingeniería. Dichas modificaciones consisten en la manipulación visual de la orientación del efector final del robot y la creación de tomas dinámicas por parte de un sistema de cámaras. Además, se diseñó una herramienta con características similares a un endoscopio endonasal, que se acopla tanto al robot real como a su gemelo digital. Para evaluar los resultados se realizaron pruebas con un *phantom* de cráneo, donde se utilizó la plataforma para crear trayectorias neuroquirúrgicas con acceso endonasal transesfenoidal.

En el modo de control “Dibujo Libre”, se empleó un sistema de cámaras que generan una vista desde diferentes ángulos del espacio de trabajo, posibilitando alejarse y acercarse al cráneo 3D, e incluso realizar cortes al interior de este. Por ende, se facilita el posicionamiento de las esferas de control (marcadores de entrada y salida) en las cavidades del cráneo 3D, confirmando desde cada plano y en tiempo real la correcta ubicación del punto de acceso y del punto de llegada del efector final del robot. Además, es posible explorar el cráneo virtual con la cámara endoscópica, con el fin de sortear obstáculos al interior de este. Las trayectorias pueden ser almacenadas en archivos locales para su posterior uso y modificación.

A partir de las pruebas realizadas, se confirmó la capacidad de la plataforma para seguir trayectorias de forma tan precisa que es posible rotar el efector final del robot dentro de la cavidad nasal del *phantom*. En cuanto a los riesgos que supone trabajar con trayectorias al interior de objetos tan delicados, se implementó un modo automático que permite una ejecución segura frente a errores humanos.

Finalmente, este trabajo evidenció la versatilidad y el potencial de la plataforma en diferentes aplicaciones que involucren un robot UR3e. El objetivo de este trabajo ha sido contribuir al desarrollo de la robótica en áreas como la neurocirugía, donde tales avances no se han explorado en su totalidad por los sectores académico o comercial. Con un posible refinamiento de la plataforma investigadores, desarrolladores e incluso profesionales del área de la salud podrían ver un gran potencial en esta dirección.

5.2. **Desarrollos futuros**

En todo el proceso de investigación y desarrollo del proyecto resultaron ciertas bifurcaciones interesantes que no fueron exploradas, pues no hacían parte del propósito fundamental del trabajo, pero sin duda son ejes temáticos que pueden profundizarse. A continuación, se plantean algunas actividades que pueden ser implementadas en proyectos futuros, como posibles mejoras dentro de la aplicación:

- Implementar un algoritmo automático o semiautomático de registro, haciendo uso de un programa auxiliar como 3Dslicer, conectado a Unity por medio del puente ROS-IGTL (nodo ROS que permite el intercambio de transformaciones, imágenes y modelos 3D gracias a *OpenIGTLink*).
- Incorporar un cráneo virtual con toda la anatomía cerebral: arterias, venas, nervios, etc.
- Evaluar la precisión de las trayectorias endonasales con métodos externos como visión de máquina, sensores magnéticos, Doppler, etc.
- Implementar mejoras en la interfaz como agregar botones que activen las nuevas funcionalidades de “Dibujo Libre” (cámaras, modos de trayectoria, posición inicial); con el fin de facilitar el manejo de la herramienta al usuario y hacerla más intuitiva.
- Incluir físicas en el cráneo 3D, de manera que sea posible visualizar colisiones y fuerzas de contacto en el mundo virtual.
- Incorporar una rotación de 360° en la cámara del endoscopio, para ajustar el ángulo de visión a voluntad.
- Incorporar un panel que permita simular el monitoreo de variables fisiológicas como la presión intracraneal, la función neurológica, o el flujo sanguíneo cerebral entre otros.

- Implementar las ecuaciones de punto de fulcro que permitan un movimiento angular preciso sobre un eje de rotación, para controlar el acceso de la herramienta quirúrgica y restringir su campo de operación.

A. Anexos

A.1. Manual de usuario

UR3e Project v8

🔴 *Requirements*:

- Ubuntu 18.04 +
- *ROS Melodic* + • *URSim or Real Robot*.
- Unity 2021.3.0f1 +

📜 *Contents*:

- **Intefaz_v8**: Contiene el proyecto construido en Unity3D.
- **Ejecutable**: Ejecutable del proyecto.
- ***Resources:*** Carpeta con trayectorias endonasales + 4 videos de soporte.

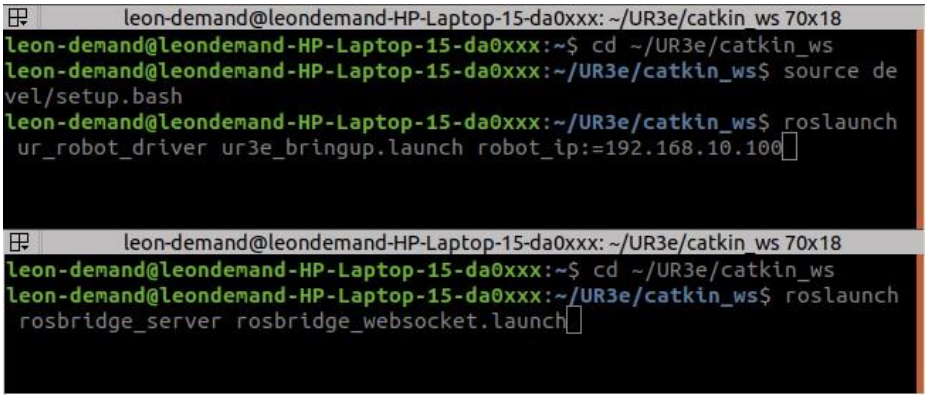
A.1.1. URSim

Se debe inicializar el simulador URSim o el *Teach Pendant* del robot, de la misma forma que en las versiones anteriores del tutorial, salvo que se debe ajustar el tamaño de la herramienta acoplada al robot, modificando el valor del TCP en z del robot:

Anexo A.1*: Teach Pendant.*

A.1.2. *Rosbridge* y UR *Drive*

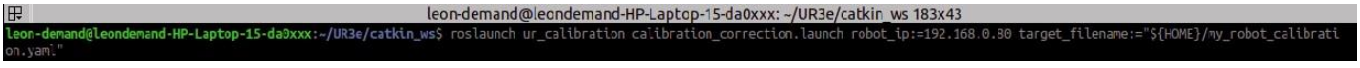
Se ejecuta *Rosbridge* y el controlador de la misma forma que en las versiones anteriores del tutorial (para URSim se coloca la dirección IP del PC desde donde se trabaja):



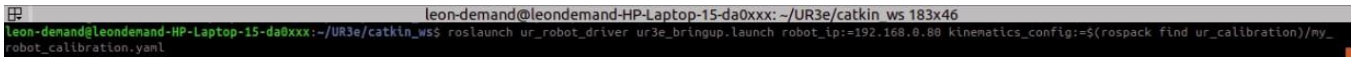
Anexo A.2: Ejecución *Rosbridge* y el controlador.

¡Importante!

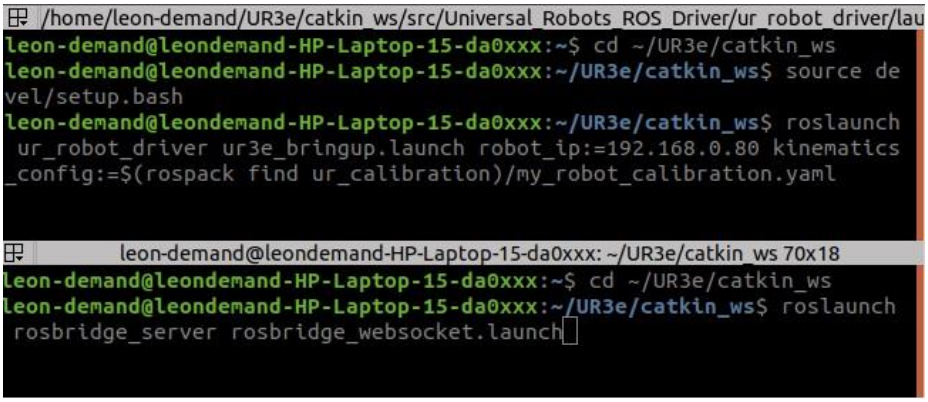
- Si se va a utilizar el robot real a veces puede ocurrir un error de calibración. Para solucionarlo efectuar los siguientes pasos:
- Se extrae el archivo de calibración de fábrica del robot, con la dirección IP de este:



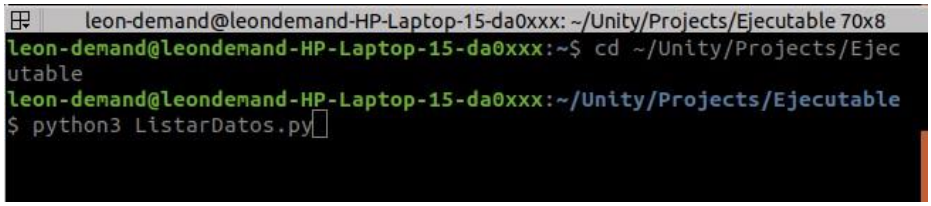
- Una vez se tiene el archivo se debe ejecutar junto con el controlador *URCap*:



En conjunto se ejecuta *Rosbridge* y el controlador de esta forma:

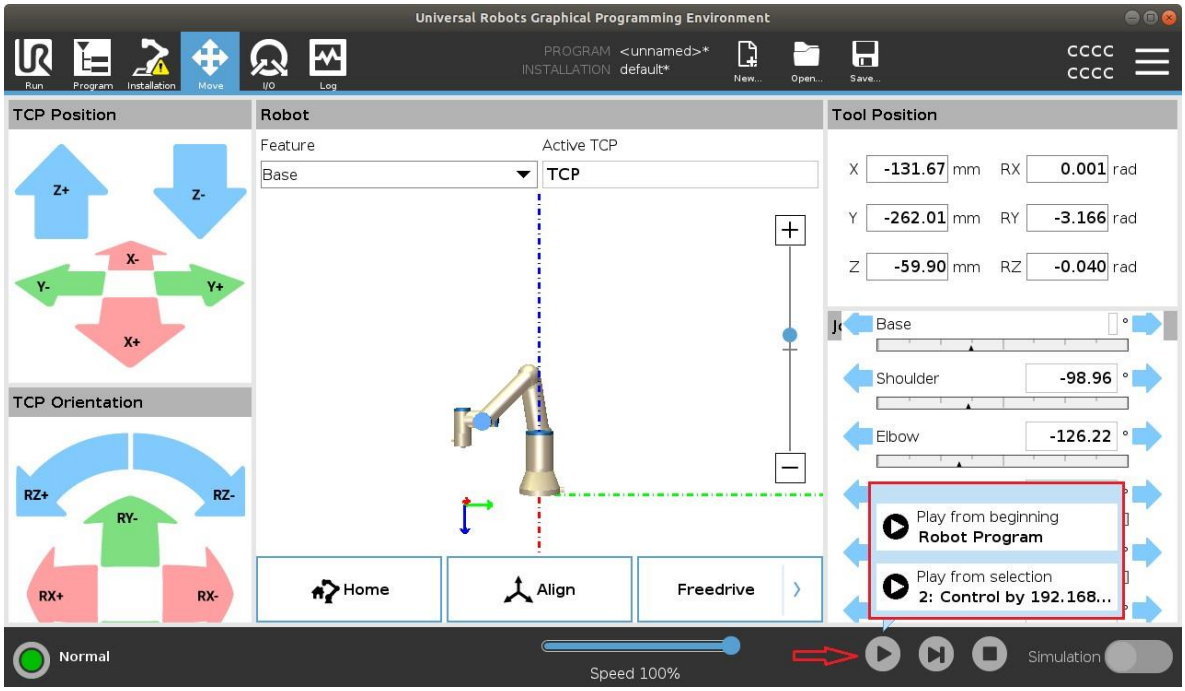


Finalmente, en ambos casos se debe correr el archivo “ListarDatos.py”:



Anexo A.3: Ejecuta ListarDatos.py

Nota: A continuación, se procede a correr el proyecto en Unity o directamente en el ejecutable. En este caso particular, el manual se aborda desde Unity, debido a que entrega una vista adicional de la escena (mitad derecha de la pantalla) y permite conocer ciertas características de los objetos que lo componen. Cabe resaltar que dichas características sólo pueden ser modificadas desde Unity (*Inspector*) y no en el ejecutable, como se verá más adelante. No olvidar una vez ingresada la dirección IP en la pestaña “conexión” de la plataforma, activar la opción “*Play from selection*” al correr el simulador URSim.

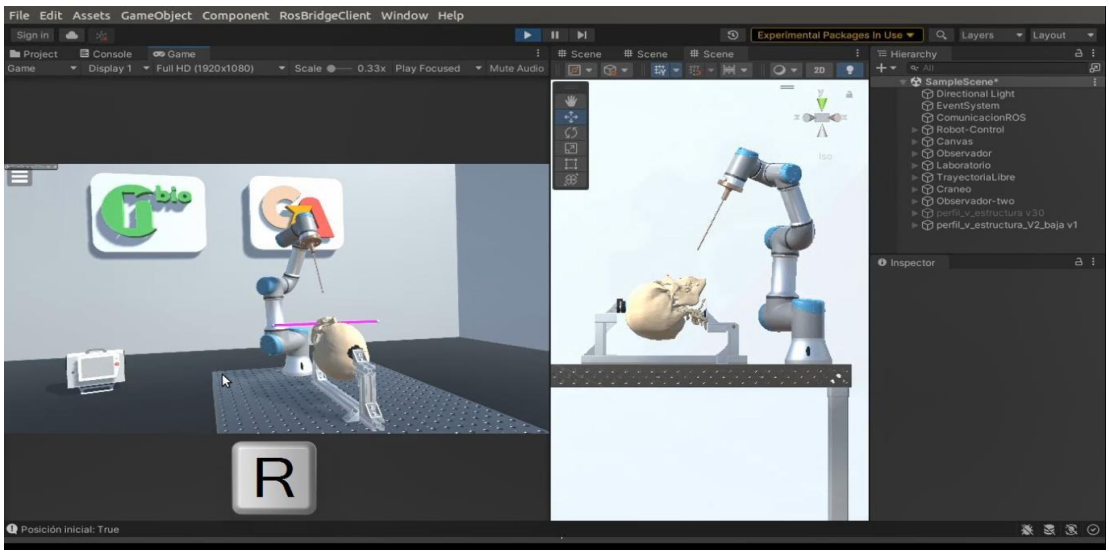


Anexo A.4: *Play from selection* en URSim

A.1.3. Unity

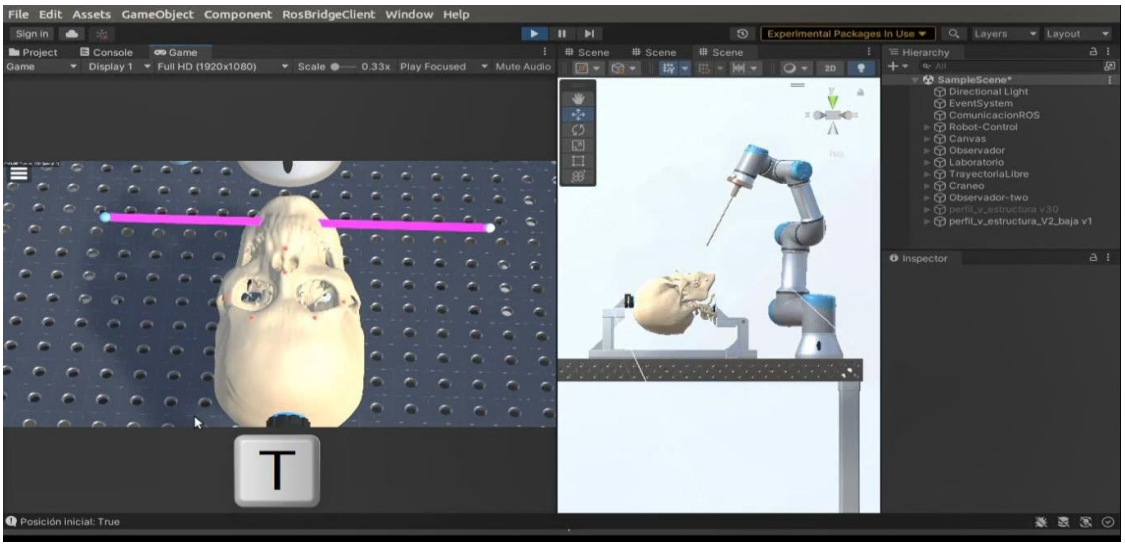
Cámaras:

- **Cámara principal**: Es la vista por defecto; se puede volver a ella presionando la tecla **R**. Esta cámara está ubicada a una distancia fija del robot.



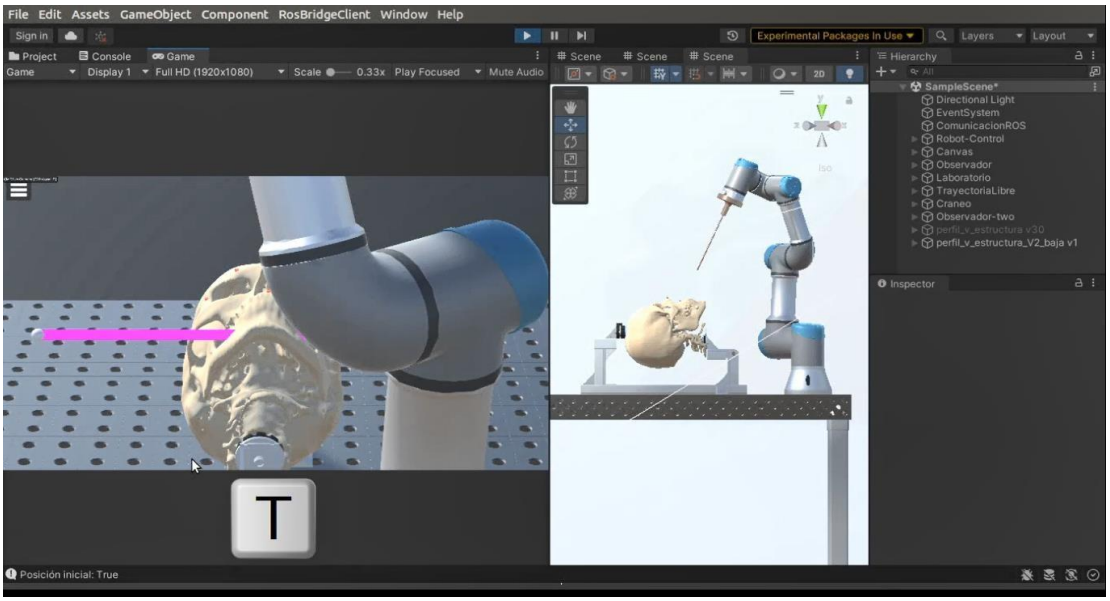
Anexo A.5: Vista cámara principal.

- **Cámara superior**: Esta vista se ubica por encima del cráneo; se puede acceder a ella presionando la tecla **T**. Esta cámara está ubicada a una distancia fija del robot.



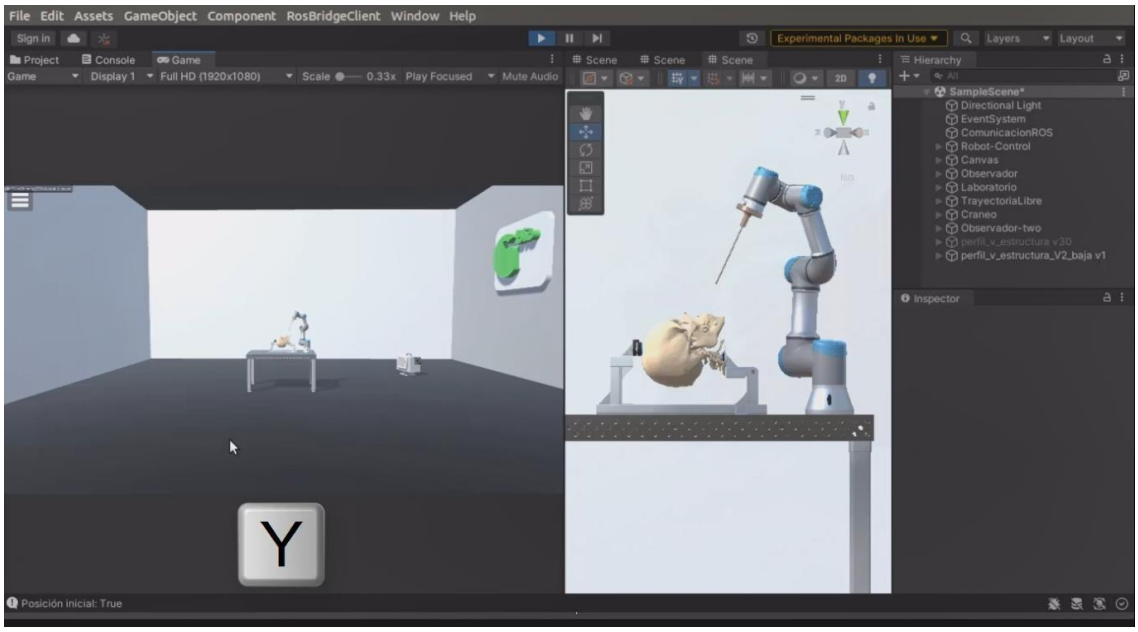
Anexo A.6: Vista cámara superior.

- **Cámara inferior:** Esta vista se ubica por debajo del cráneo; se accede a ella presionando la tecla **T** de nuevo. Esta cámara está a una distancia fija del robot.



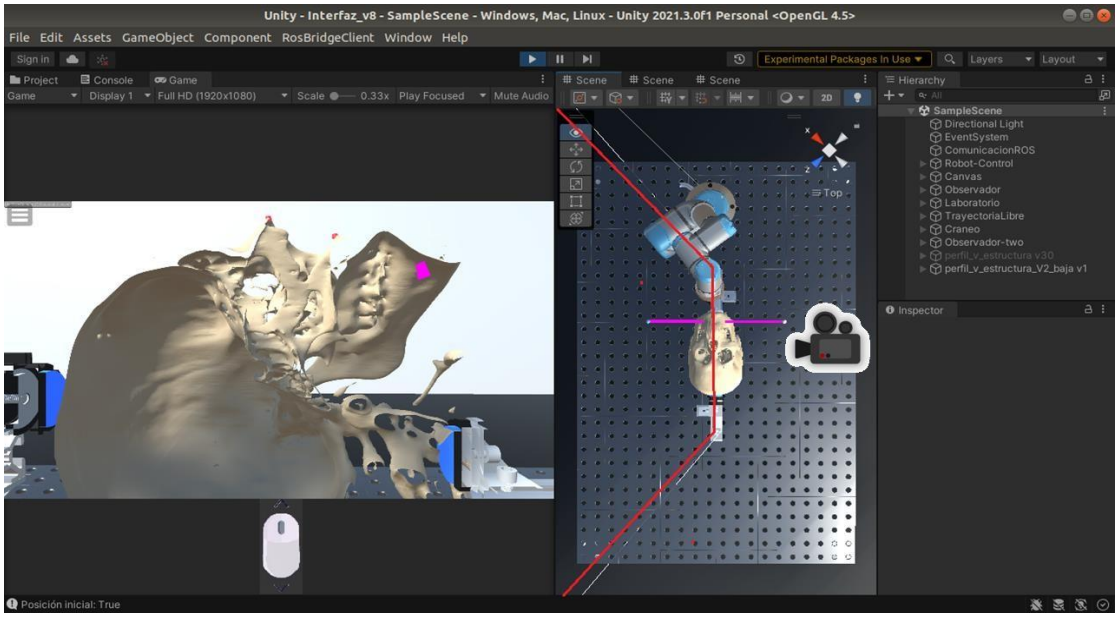
Anexo A.7: Vista cámara inferior.

- **Cámara lateral derecha:** Esta vista se ubica al lado derecho del cráneo; se puede acceder a ella presionando la tecla **Y**. Esta cámara tiene *zoom*, que permite acercarse o alejarse del cráneo con el *scroll* del mouse.



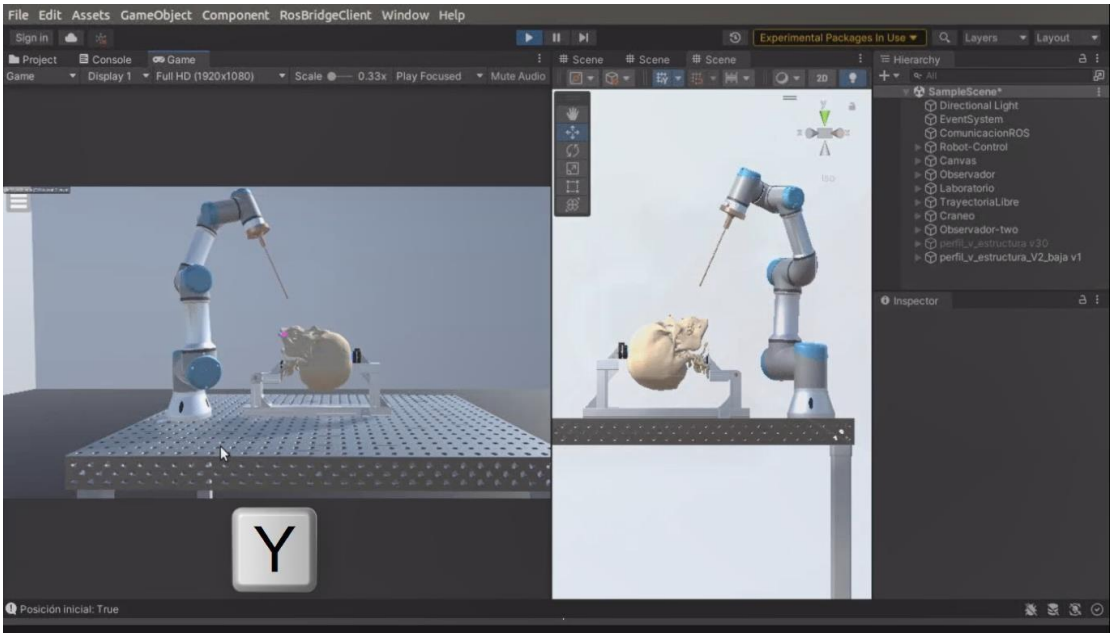
Anexo A.8: Vista cámara lateral derecha.

Al acercarse al cráneo es posible realizar cortes sagitales que permiten visualizar su interior hasta la mitad aproximadamente.



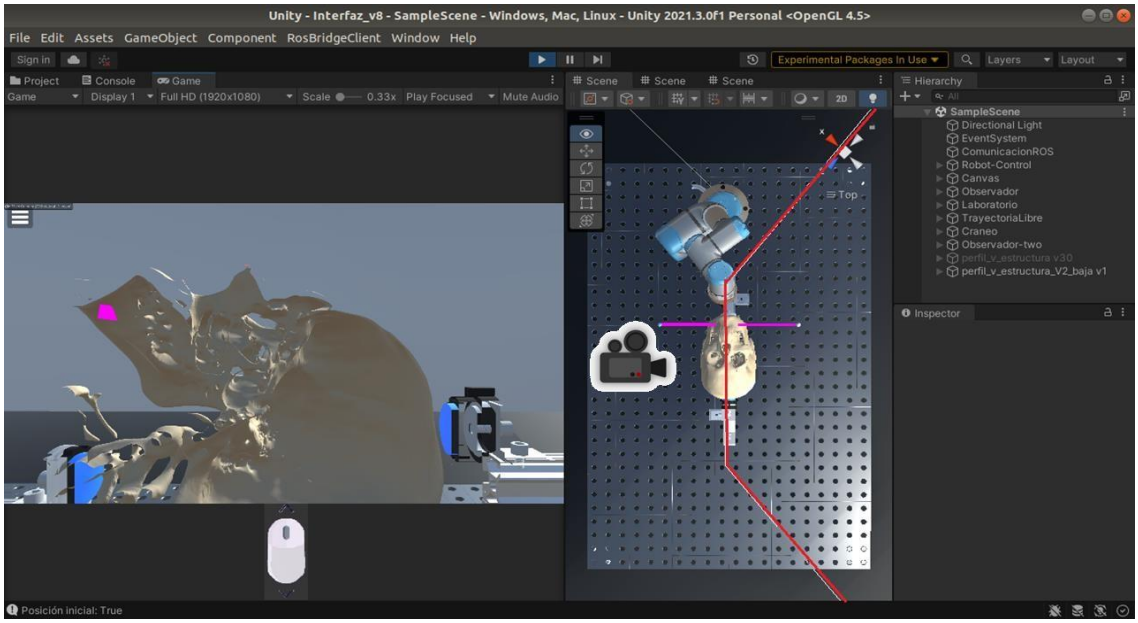
Anexo A.9: Vista lateral derecha corte sagital.

Cámara lateral izquierda: Esta vista se ubica al lado izquierdo del cráneo; se accede a ella presionando la tecla **Y** de nuevo. Esta cámara tiene *zoom*, que permite acercarse o alejarse del cráneo con el *scroll* del mouse.



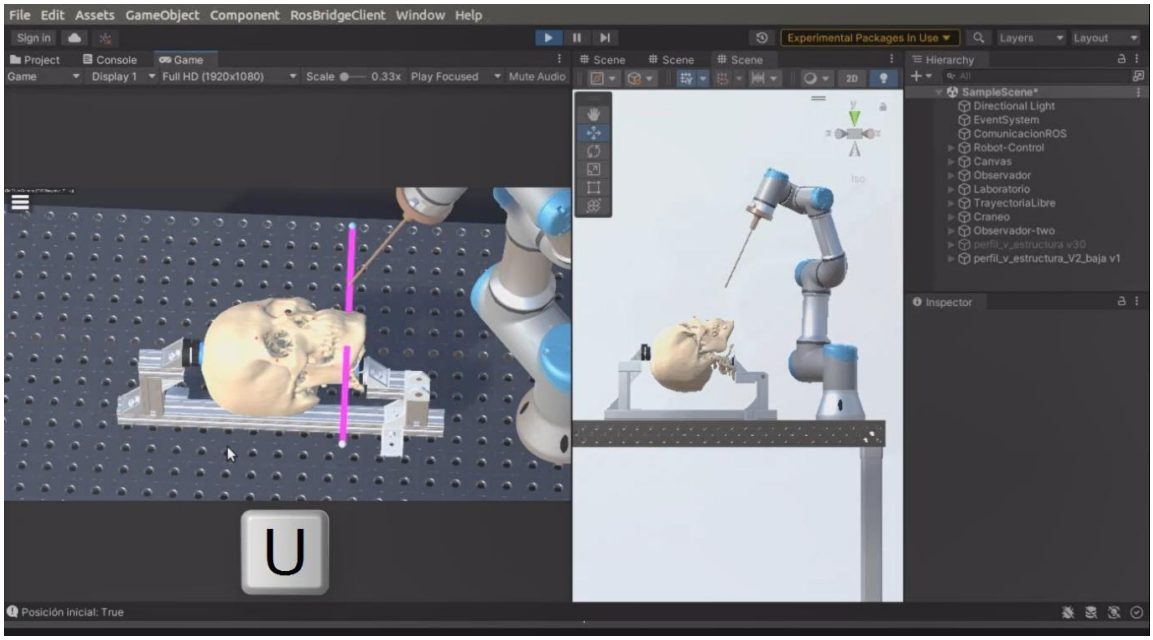
Anexo A.10: Vista lateral izquierda.

Al acercarse al cráneo es posible realizar cortes sagitales que permiten visualizar su interior hasta la mitad aproximadamente.



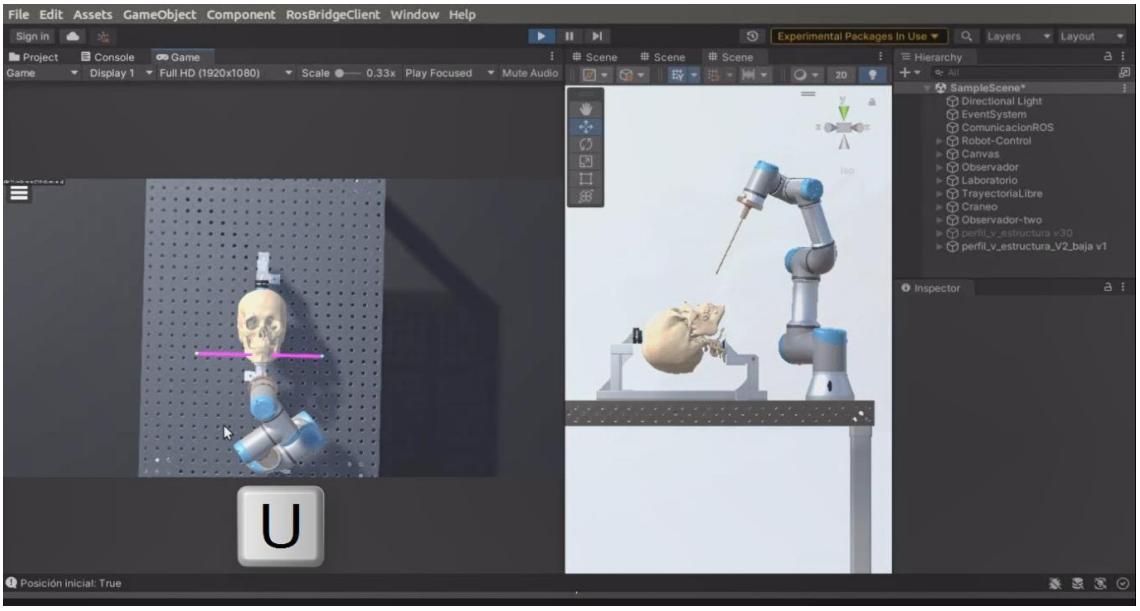
Anexo A.11: Vista lateral izquierda corte sagital.

Cámara diagonal derecha: Esta vista se ubica al lado diagonal derecho del cráneo en un ángulo aproximado de 45°; se puede acceder a ella presionando la tecla **U**. Esta cámara tiene *zoom*, que permite acercarse o alejarse del cráneo con el *scroll* del mouse.



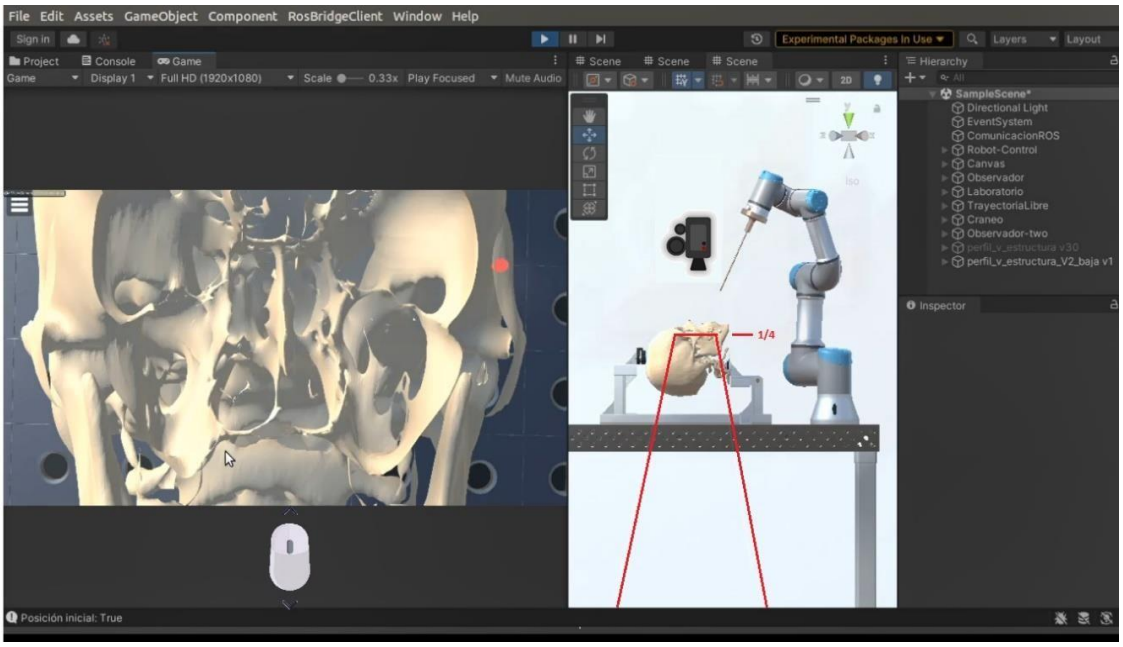
Anexo A.12: Vista cámara diagonal derecha

Cámara anterior: Esta vista se ubica al frente del cráneo; se accede a ella presionando la tecla **U** de nuevo. Esta cámara tiene *zoom*, que permite acercarse o alejarse del cráneo con el *scroll* del mouse.



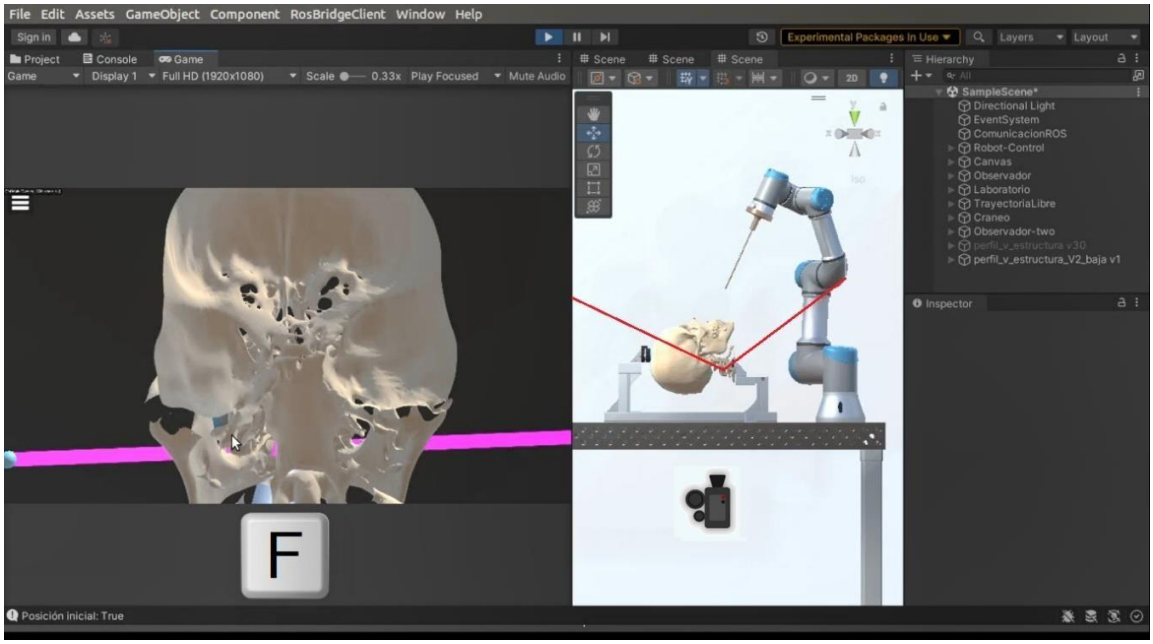
Anexo A.13: Vista cámara anterior.

Al acercarse al cráneo es posible realizar cortes coronales que permiten visualizar su interior hasta ¼ del mismo aproximadamente.



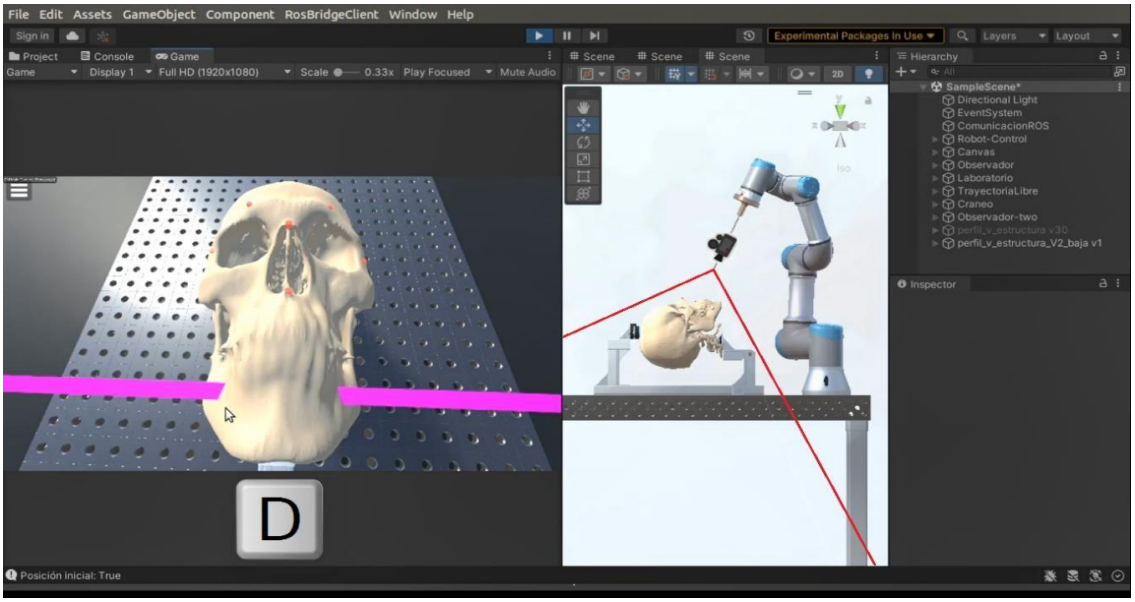
Anexo A.14: Vista cámara anterior corte coronal.

Cámara posterior interna: Esta vista se ubica en el interior desde atrás del cráneo; se puede acceder a ella presionando la tecla **F**. Esta cámara tiene un plano coronal fijo que permite visualizar su interior hasta la mitad aproximadamente.



Anexo A.15: Vista cámara posterior interna.

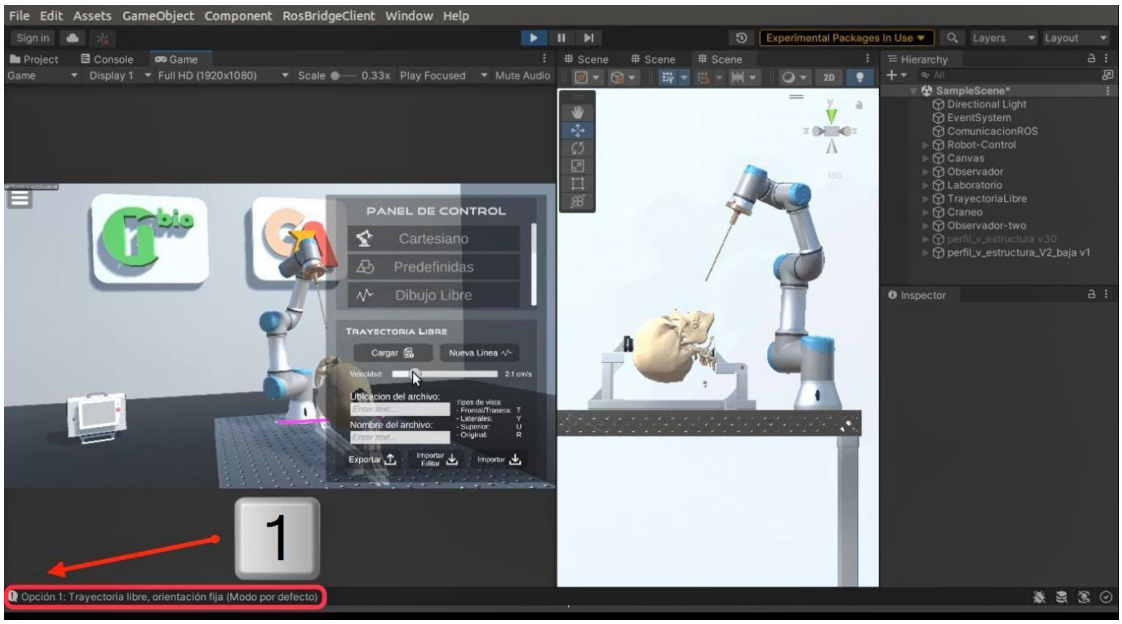
Cámara endoscopio: Se puede acceder a ella presionando la tecla **D**. Esta vista se ubica en la punta de la herramienta acoplada al robot como un objeto hijo; por lo tanto, estará atada a sus movimientos con el propósito de observar los recorridos que se realicen al interior del cráneo. Esta cámara tiene una distancia focal pequeña, con el fin de tener un ángulo de visión amplio que permita captar la mayor cantidad de detalles a distancias muy cortas.



Anexo A.16: Vista cámara tipo endoscopio.

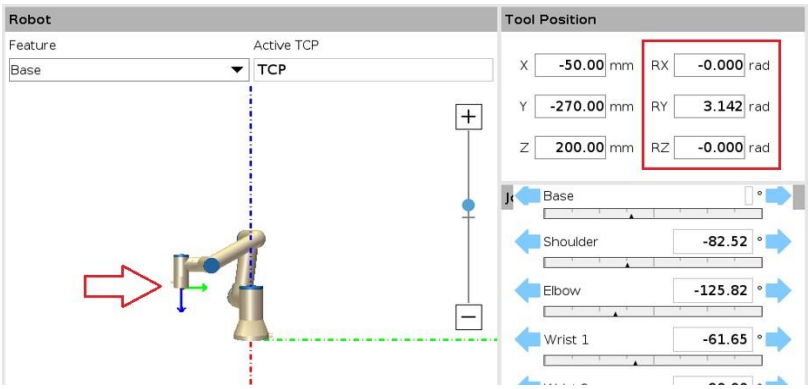
A.1.4. Tipos de trayectoria

- **Opción 1: Trayectoria libre – orientación fija:**



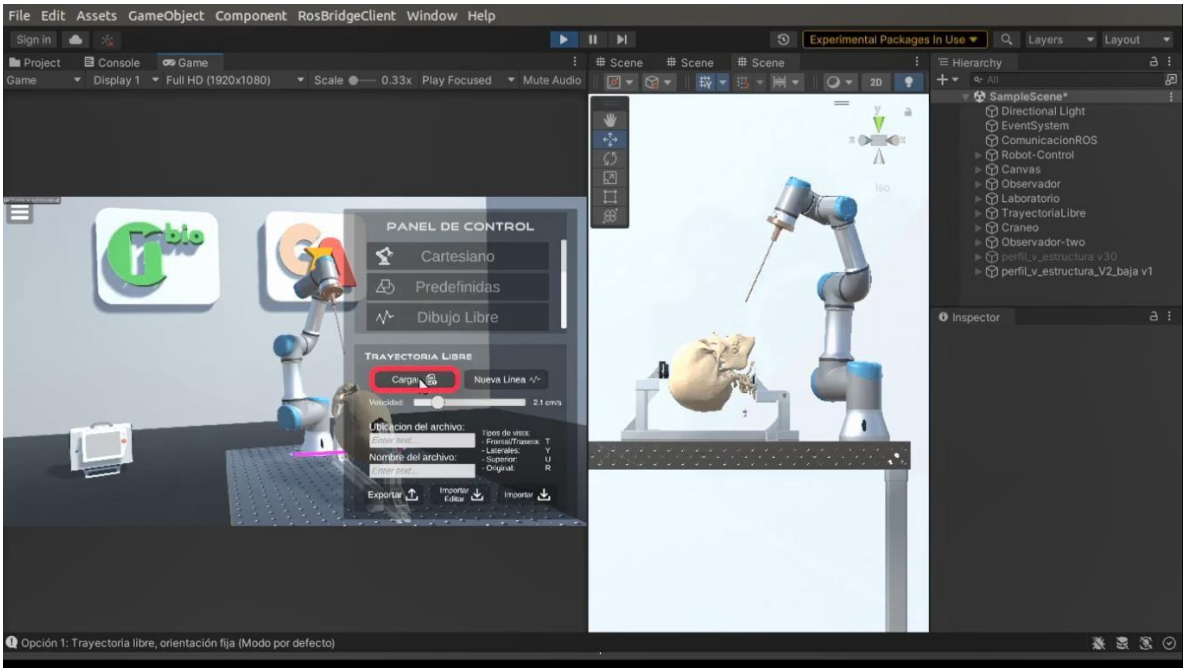
Anexo A.17: Trayectoria libre - orientación fija

Desde el “Panel de Control” al seleccionar la pestaña “Dibujo Libre”, se accede a la **opción 1** u opción por defecto (tecla 1 del teclado alfanumérico). Esta se caracteriza porque el movimiento que realiza el robot está determinado por las posiciones de dos marcadores en forma de esferas conectados por una línea rosa llamada “*Line Renderer*”. Estos marcadores son manipulables con el mouse y pueden situarse en cualquier punto tridimensional del espacio del juego. Su función es determinar el recorrido que debe realizar el efector final del robot desde una posición inicial hasta una posición final. El controlador del robot realizará el cálculo de la cinemática inversa que se debe llevar a cabo para moverse en dicha trayectoria. En la **opción 1** es posible modificar el “*line renderer*”, de modo que se creen nuevos marcadores que modifican la trayectoria en línea recta y permitan generar diferentes curvas. Sin embargo, la orientación del efector final siempre va a ser la misma; es decir, siempre será (0, π, 0) en ángulos de Euler o (0,1,0,0) en cuaterniones.



Anexo A.18: Orientación del TCP en URSim.

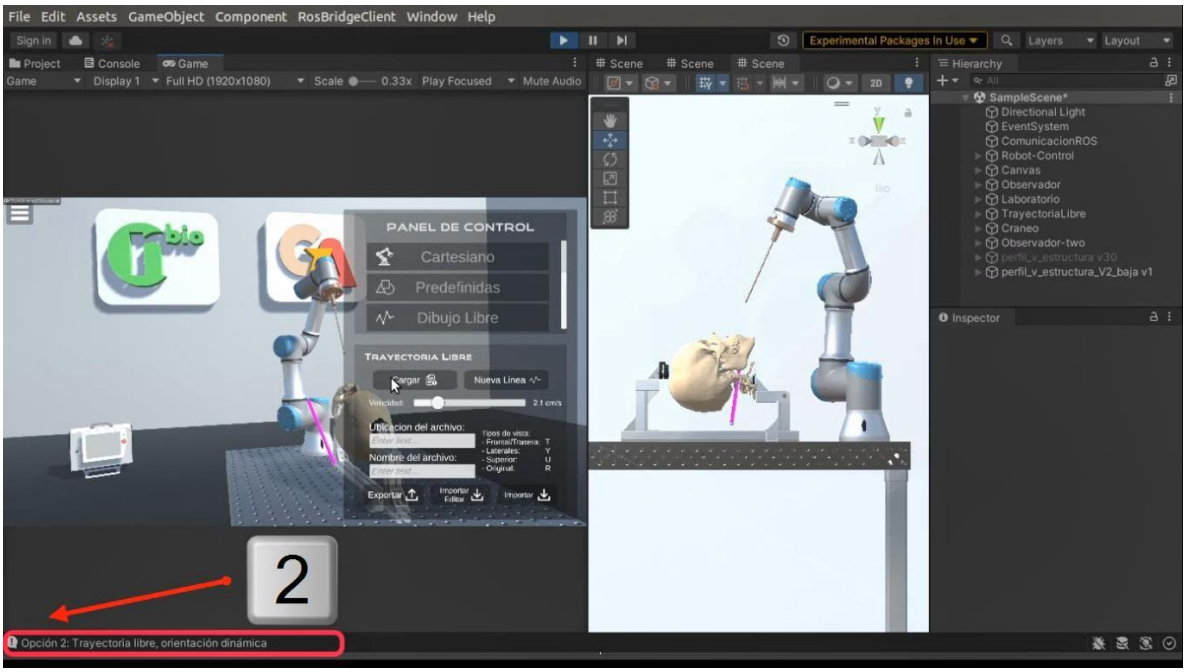
Cabe resaltar que las esferas deben ser manipuladas bajo ciertas restricciones, como el espacio de trabajo del robot o la orientación del efector final; en caso contrario se va a generar un error en el controlador al presentarse una singularidad en el cálculo de la cinemática inversa.



Anexo A.19: Cargar trayectoria libre - orientación fija.

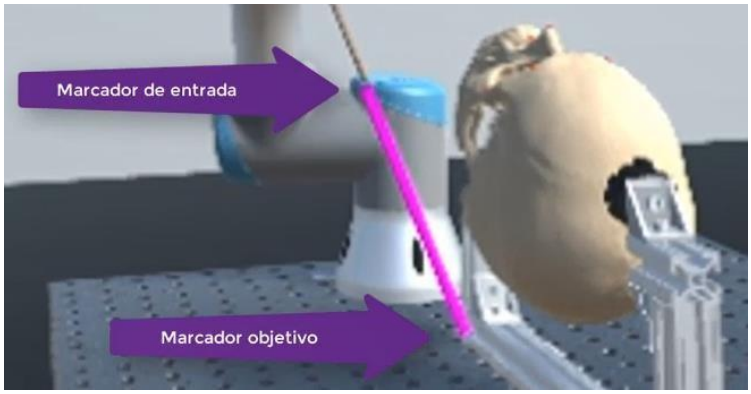
Una vez se haya definido el recorrido se da clic en cargar. El efector final del robot se moverá con una orientación fija a través de esta línea o curva. Se puede aumentar o disminuir la velocidad del robot con el *slider* ubicado debajo del botón cargar.

- **Opción 2: Trayectoria libre – orientación dinámica:**



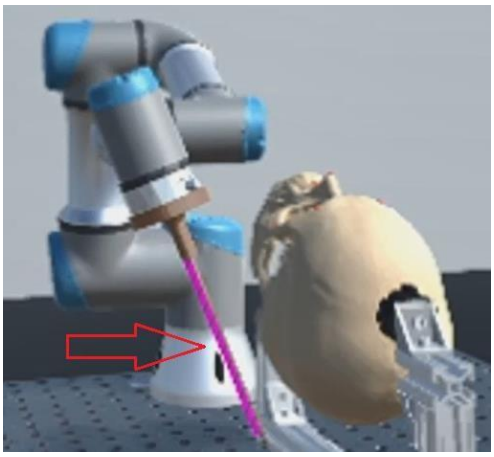
Anexo A.20: Trayectoria libre - orientación dinámica.

Para acceder a la **opción 2** de la pestaña “Dibujo Libre” se debe presionar la tecla 2 del teclado alfanumérico. A diferencia de la opción anterior, en la opción 2 no importa si se crean nuevos marcadores dentro del “*line renderer*”, ya que las únicas posiciones que va a tener en cuenta el controlador para ejecutar una trayectoria es la del marcador inicial y la del marcador final; por lo tanto, sólo es posible moverse en línea recta. Estos marcadores serán conocidos de ahora en adelante como marcador de entrada y marcador objetivo, debido a que para efectos prácticos representan un punto de acceso al cráneo y un punto de llegada del efector final.



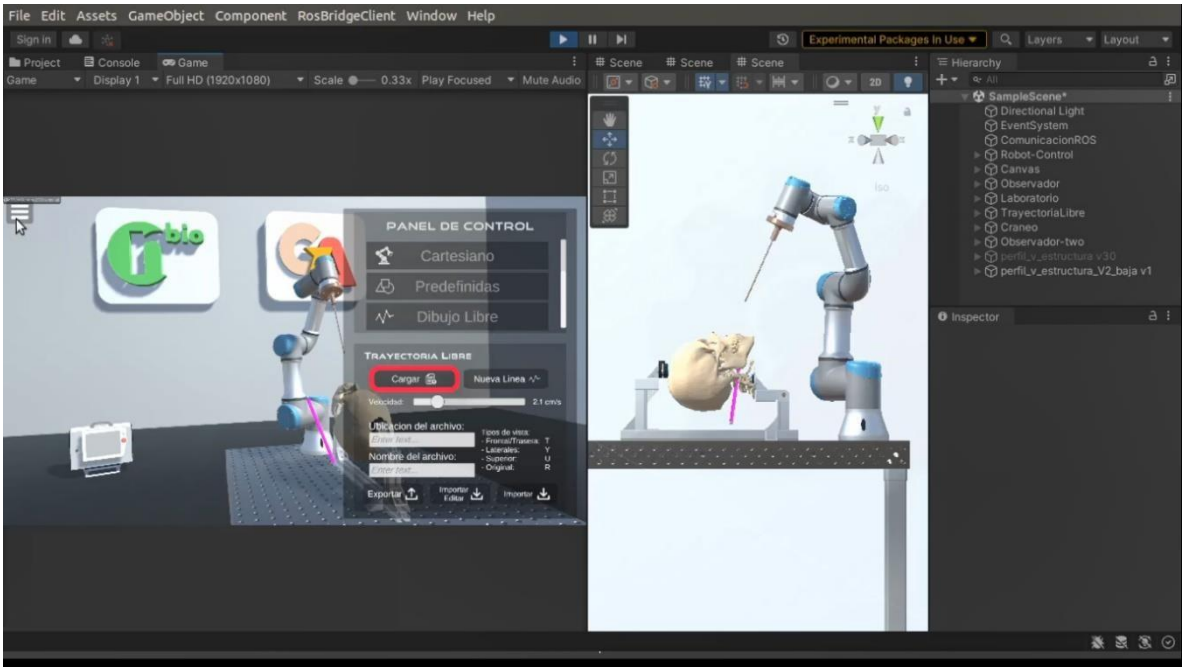
Anexo A.21: Marcador de entrada y objetivo.

Ahora, la orientación no será fija sino cambiante, ya que la orientación que se envía al controlador del robot es calculada dentro de la plataforma y estará determinada por los ángulos formados por los dos marcadores mencionados. De esta forma, diferentes ubicaciones del marcador de entrada o del marcador objetivo, generarán diferentes orientaciones del efector final, o, en otras palabras, el "*line renderer*" y la herramienta estarán superpuestos uno sobre el otro indicando visualmente su coincidencia.



Anexo A.22: Orientación del efector según el *line renderer*.

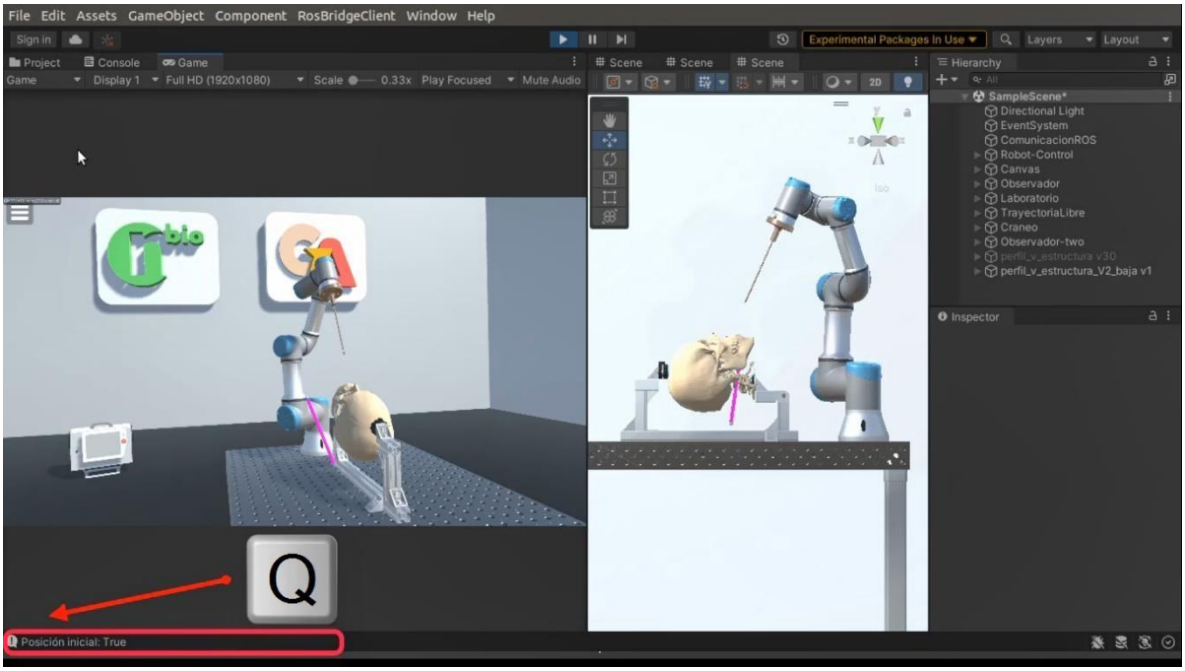
Al igual que en el caso anterior, las esferas deben ser manipuladas bajo ciertas restricciones, como el espacio de trabajo del robot o la orientación del efector final.



Anexo A.23: Cargar trayectoria libre - orientación dinámica.

Una vez se haya definido el recorrido se da clic en cargar. El efector final del robot se moverá con una orientación dinámica a través de esta línea. Se puede aumentar o disminuir la velocidad del robot con el *slider* ubicado debajo del botón cargar.

Nota: La **opción 2** tiene una característica adicional que consiste en una posición inicial, accesible al presionar la tecla **Q** seguida del botón cargar. Desde esta posición es fácil acceder al área frontal del cráneo.



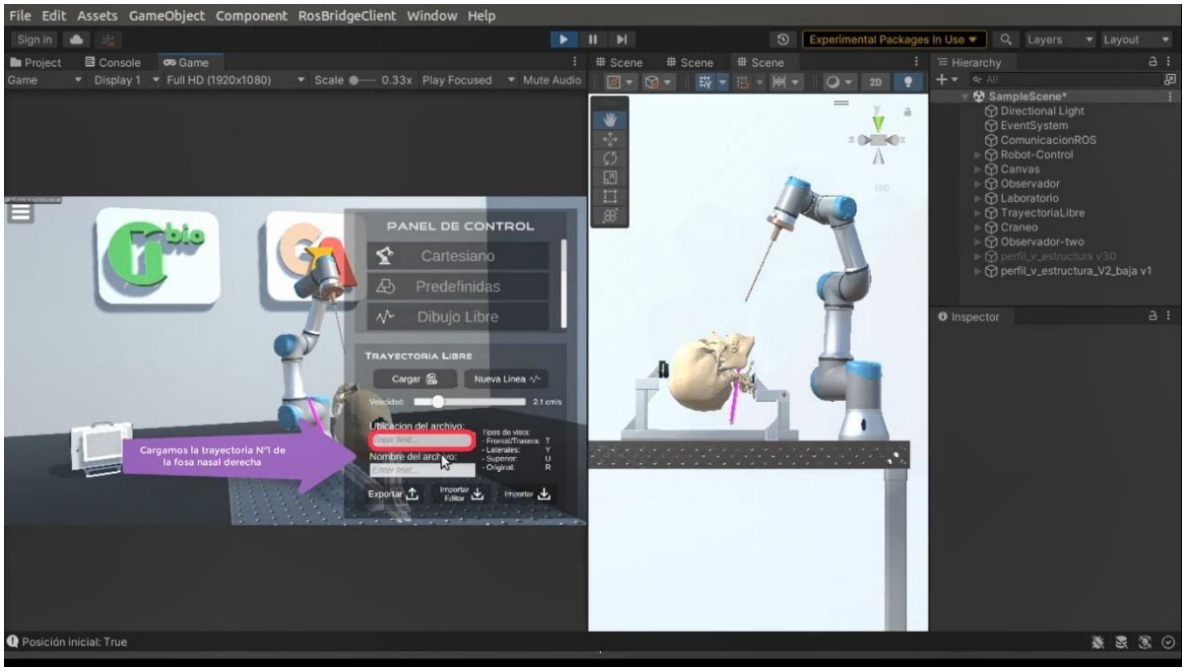
Anexo A.24: Posición inicial del robot.

¡Importante!

Es relevante señalar, que los mensajes en consola que aparecen al presionar las teclas 1, 2 o Q, sólo son visibles al ejecutar el proyecto desde Unity y no así, en el ejecutable. Se debe tener especial cuidado al cambiar entre las opciones 1 y 2 para no generar errores en el controlador del robot; ya que existen trayectorias que se pueden ejecutar desde una opción, pero no desde la otra. También se debe tener en cuenta que al ingresar en las cajas de texto nombres o direcciones que contengan el número 1 o 2, se están activando indirectamente estas opciones (las teclas 1 o 2 del teclado numérico no activan estas opciones).

A.1.5. Trayectorias endonsales

- **Guardar y cargar un archivo:**



Anexo A.25: Guardar y cargar un archivo.

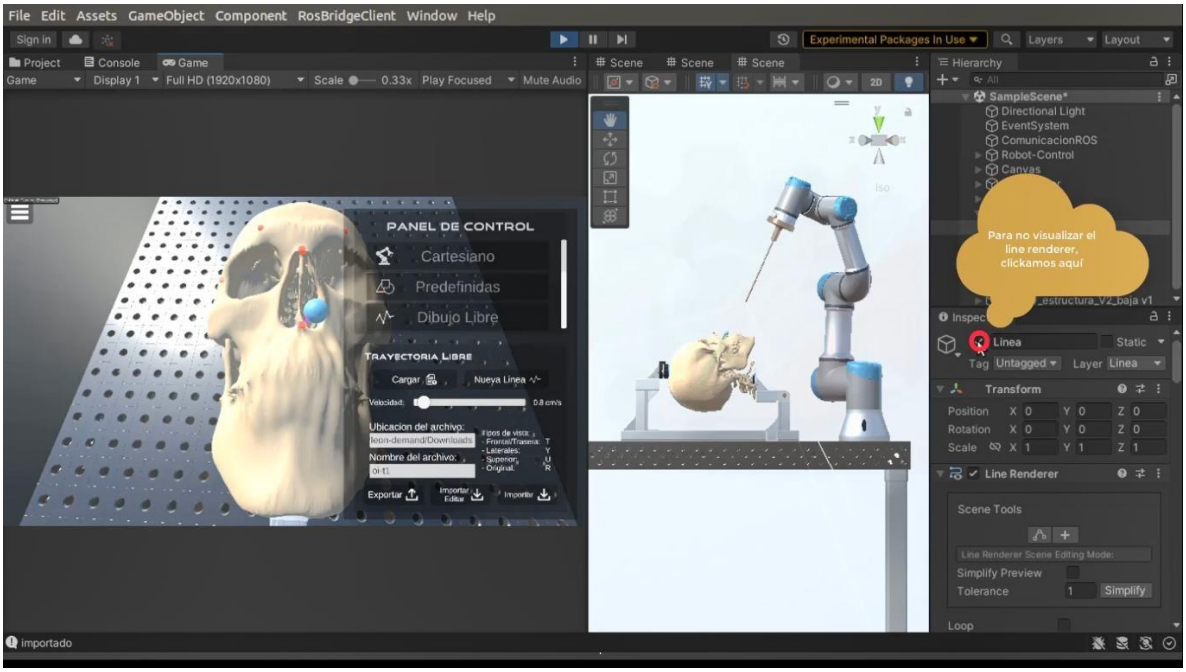
La pestaña “Dibujo Libre” ofrece la característica de guardar las trayectorias como un archivo *JSON*, con las posiciones (X, Y, Z) del marcador inicial, el marcador final y el “*Line Renderer”,* dando un total de 50 puntos. Para ello, se debe definir un nombre y una dirección dentro del “*file system*” de Ubuntu. Además, es posible importar y editar dichas trayectorias, al cargar el archivo correspondiente a dicha trayectoria. Se debe señalar que, al ingresar el nombre y la dirección del archivo por teclado, se están activando de forma indirecta las diferentes cámaras del juego.

¡Importante!

En la carpeta *“Resources*” se encuentran 10 muestras de trayectorias endonasales; 6 trayectorias correspondientes al orificio nasal derecho y 4 trayectorias correspondientes al

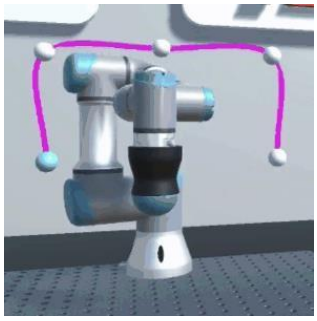
orificio nasal izquierdo. Estas trayectorias fueron probadas en el *phantom* del modelo 3D del cráneo, con un robot UR3e real.

- **Visualización del "*Line Renderer*":**



Anexo A.26: Visualización "*Line Renderer".*

El "*Line Renderer*" o la línea rosa que conecta el marcador inicial y el marcador final, tiene una utilidad sobre todo en la opción 1 de la pestaña "Dibujo Libre", ya que permite manipular la trayectoria del efector final del robot, creando diferentes tipos de curvas al generar nuevos marcadores.



Anexo A.27: Dibujo libre opción por defecto.

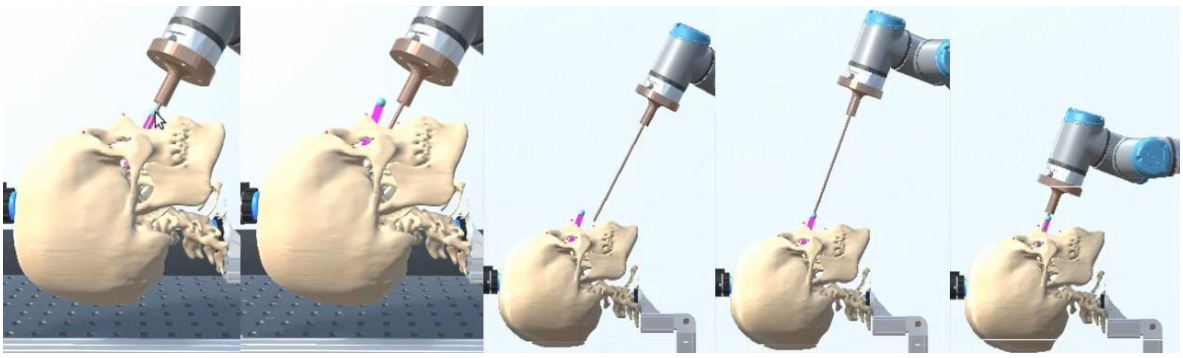
Sin embargo, en la opción 2 el "*Line Renderer*" pierde su utilidad al solo considerarse las posiciones del marcador inicial y el marcador final; y así como se mencionó anteriormente, no importa si se crean nuevos marcadores, el efector final solo va a seguir una trayectoria

en línea recta desde el marcador inicial hasta el marcador final, eso sí, con la orientación calculada entre estos dos puntos. Ahora, si solo son necesarios el marcador de entrada y el marcador objetivo y, considerando que en la cámara endoscopio la línea rosa es particularmente invasiva, se pone a discreción del usuario si activa o desactiva la visualización de esta característica de la plataforma. Para ello, se debe dirigir al *Inspector* en Unity y desactivar la palomita del objeto “Línea”. Cabe mencionar que esta acción no se puede realizar en el ejecutable, sino solo en el proyecto.

A.1.6.Modos de ejecución de las trayectorias

En la opción 2 de la pestaña “Dibujo Libre”, al cargar una trayectoria se pueden activar 2 modos. Estos modos se ejecutan de forma automática dependiendo de cuál de los dos marcadores es manipulado.

- **Modo 1: *Entry - Target*:**

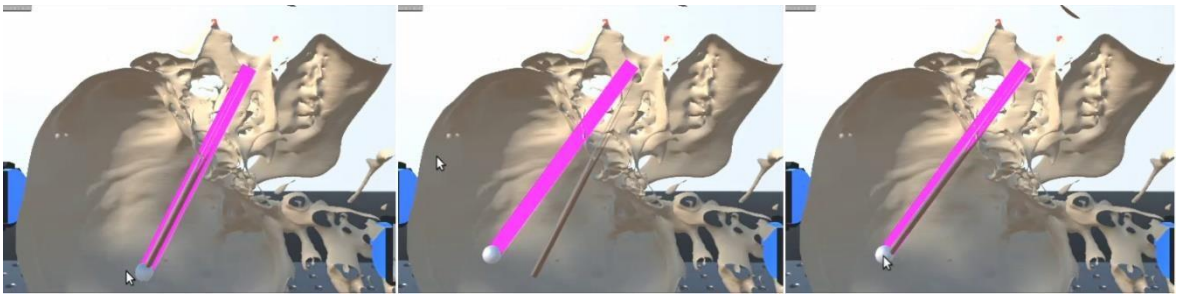


Anexo A.28: Modo 1 (*Entry-Target*).

El **modo 1** se puede activar siempre y cuando el marcador de entrada sea ubicado en una nueva posición; no importa si el marcador objetivo se ha movido, mientras el primer marcador se haya modificado aunque sea una sola vez, el modo 1 se activará. Este modo consiste en un movimiento entrada-objetivo tal como su nombre lo sugiere, salvo que cada vez que se modifica la trayectoria, el efector final realiza el movimiento inverso en sentido objetivo-entrada; es decir, se devuelve por donde llegó antes de ubicarse en el punto de entrada de la nueva trayectoria. Este movimiento adicional podría considerarse como un seguro a movimientos indeseados del efector final en busca de realizar una

nueva trayectoria; ya que, por ejemplo, en el caso del *phantom* del cráneo para ir de la fosa nasal derecha a la izquierda, si no se tuviera este seguro, el robot simplemente atravesaría el efector final por medio de las membranas que separan estas dos fosas. También, en caso de que se quiera ir a la posición inicial, el robot primero sale de la fosa nasal y luego se dirige hacia dicha posición, protegiendo así la integridad del *phantom* frente a descuidos humanos en el manejo de la plataforma.

- **Modo 2: Punto de pivote:**

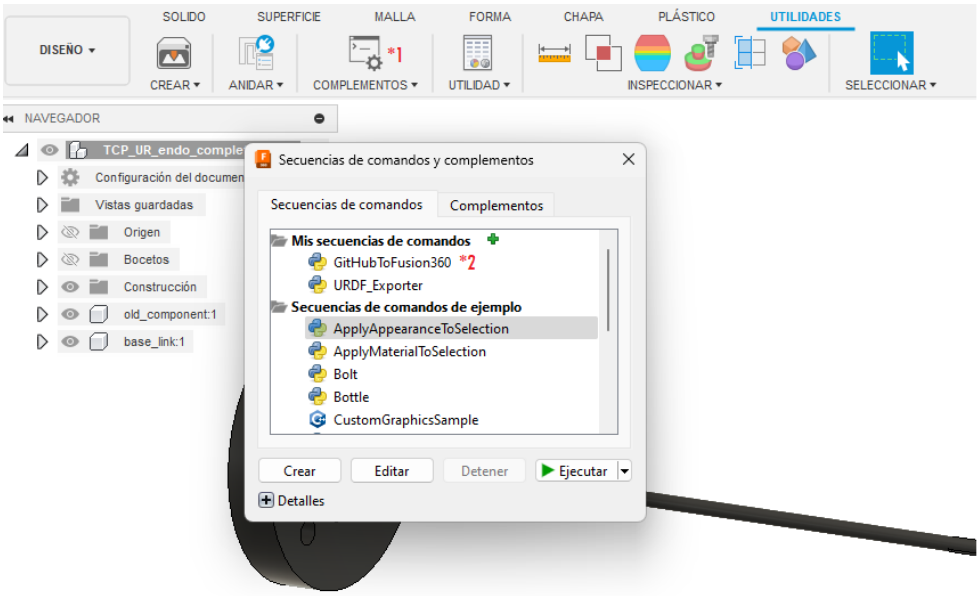


Anexo A.29: Modo 2 (punto de pivote).

El **modo 2** se puede activar siempre y cuando sólo se haya modificado la posición del marcador objetivo; si se ha movido el marcador de entrada aunque sea una sola vez, este modo no se activará. Como su nombre lo indica, al mover el marcador objetivo el robot realiza un movimiento pendular tomando como punto de pivote el marcador de entrada. Entonces en este caso el recorrido es angular y no lineal como en el modo anterior. Así pues, este modo permite alcanzar puntos al interior del cráneo, sin tener que salir y volver a entrar al mismo.

A.2. Exportar modelo 3D a URDF

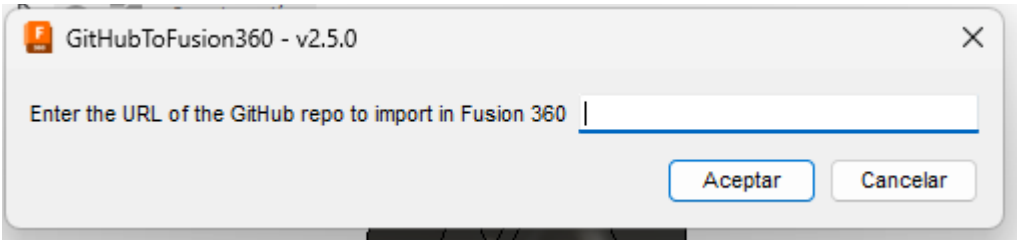
Esta sección se centra en el proceso de conversión y exportación de la pieza diseñada en *Fusión 360* a un formato URDF, utilizando *Autodesk Fusión 360* en el sistema operativo Windows 11. El objetivo de este procedimiento es incorporar la pieza diseñada al robot que se encuentra en Unity.



Anexo A.30: Instalación GithubToFusion360.

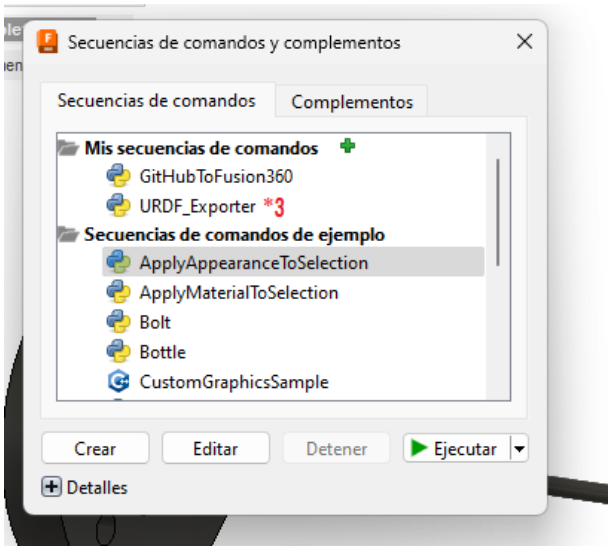
Ya teniendo la pieza diseñada, nos dirigimos a la barra de menús de *Fusión 360*, haciendo clic en la pestaña Utilidades, se desplegará un menú. Nos dirigimos a Complementos, donde se hace clic en la opción *app store Fusion 360*, la cual va a redirigir el navegador a la tienda de aplicaciones de *Autodesk Fusión 360* donde se buscará "Instalar scripts o complementos desde GitHub". Al descargarse se instalarán los complementos de *Fusion 360*, para utilizarlos se sigue la secuencias de números (Figura A.2.1).

Al dar clic en *2 (ver Anexo A.29), se despliega la siguiente barra de tareas donde se pega el repositorio de git https://github.com/syuntoku14/fusion2urdf y se siguen los pasos que salen en la ventana de instalación.



Anexo A.31: Instalación URDF_Exporter

Luego de hacer la instalación debe ya aparecer el complemento en Python *URDF_Exporter* (Anexo A.31).



Anexo A.32: Librería Python URDF_Exporter

Con el complemento instalado ya se pueden exportar las piezas diseñadas en *Fusion 360* al formato URDF, se selecciona el complemento y dando clic en ejecutar se genera automáticamente el archivo URDF. Se deben seguir las instrucciones adicionales que pueden surgir en la ventana para configurar las opciones de exportar.

NOTA:

Es importante aclarar que los pasos que se siguieron anteriormente son para obtener las piezas en un formato XACRO que es un tipo de URDF, que tiene como propósito simplificar documentos XML extensos.

A.3. Tutorial *Optitrack*

El sistema *Optitrack* consiste en 3 cámaras infrarrojas que capturan el movimiento de pequeños marcadores retrorreflectores adheridos a un objeto, con el fin de entregar datos en tiempo real ($\pm 0.2\ mm$ *accuracy*) sobre la posición y orientación de este [127].

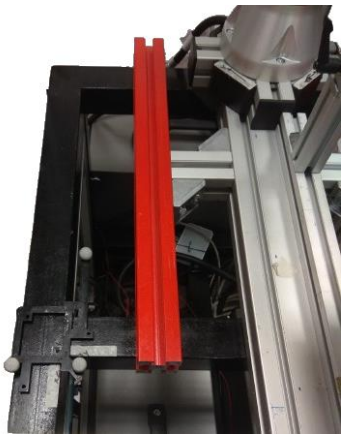
Anexo A.33: Panel de cámaras Optitrack.

Para hacer uso de este dispositivo se necesitan dos elementos muy importantes: una escuadra de calibración y un cuerpo rígido (*Rigid Body*). La escuadra de calibración definirá el origen del sistema de coordenadas del *Optitrack*, mientras que el cuerpo rígido permitirá ubicar en el espacio los puntos que se quieren medir. En primer lugar, se diseñó en el laboratorio un modelo sencillo de escuadra (Figura A.34(a)), para utilizar de forma provisional, ya que no se tenía a disposición una escuadra distribuida por la misma compañía (Figura A.34(b)). Cabe destacar que el proceso de impresión de la escuadra (deformación del material al enfriarse) y la utilización de marcadores retrorreflectores blandos en esta (circularidad y reflectividad no uniformes), son factores que a la larga introducirán un error en la medición. Por otro lado, como cuerpo rígido se utilizó un apuntador de un dispositivo de navegación Medtronic (Anexo A.34).

(a) Impresión 3D. (b) Escuadra comercial [128].

Anexo A.34: Escuadra de calibración.

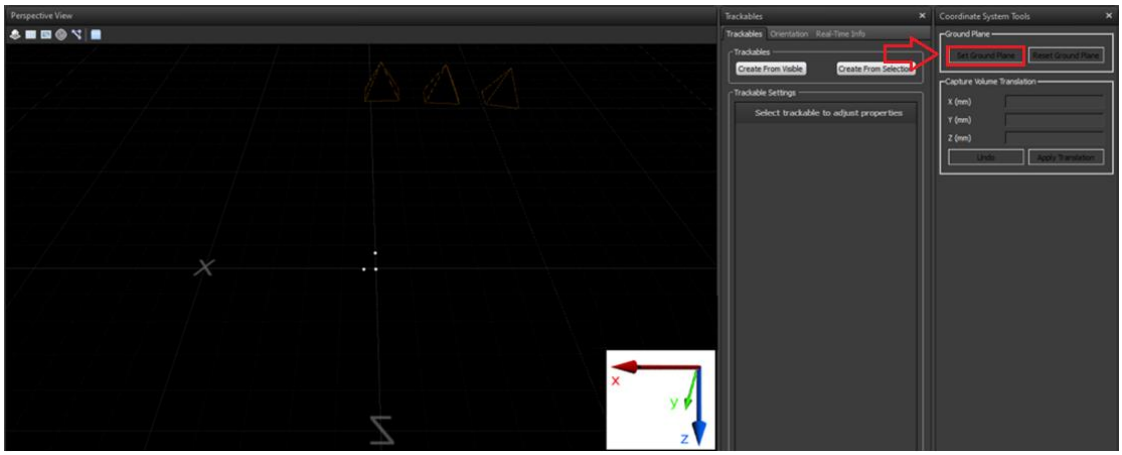
Anexo A.35: Pointer de Medtronic.



Anexo A.36: Ubicación de la escuadra de calibración respecto al robot.

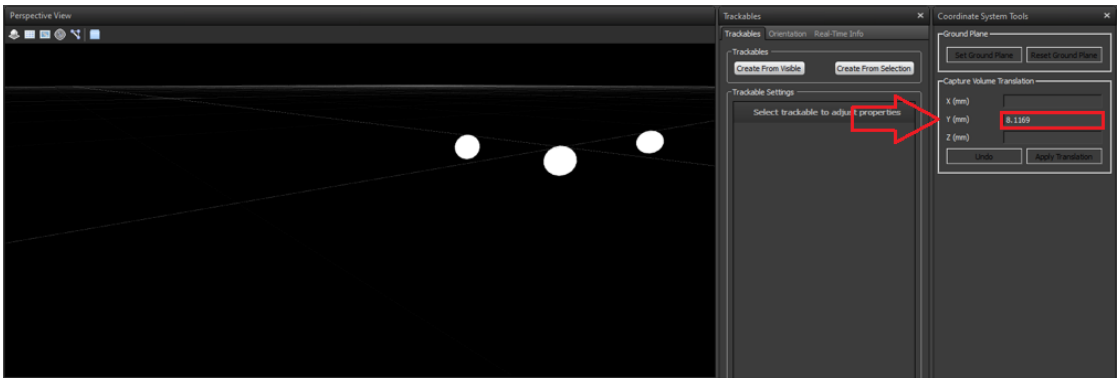
Una vez definido el sitio de ubicación de la escuadra de calibración, se debe configurar el programa *Tracking Tools* para que reconozca la escuadra como el origen de coordenadas, dando *click* en la opción "*Set Ground Plane*" del panel "*Coordinate System Tools*". En la siguiente imagen se puede observar la representación gráfica del espacio 3D

generado por el programa; en este, el brazo largo de la escuadra determina el eje Z, mientras el brazo corto determina el eje X.



Anexo A.37: Sistema de coordenadas del software *Tracking Tools*.

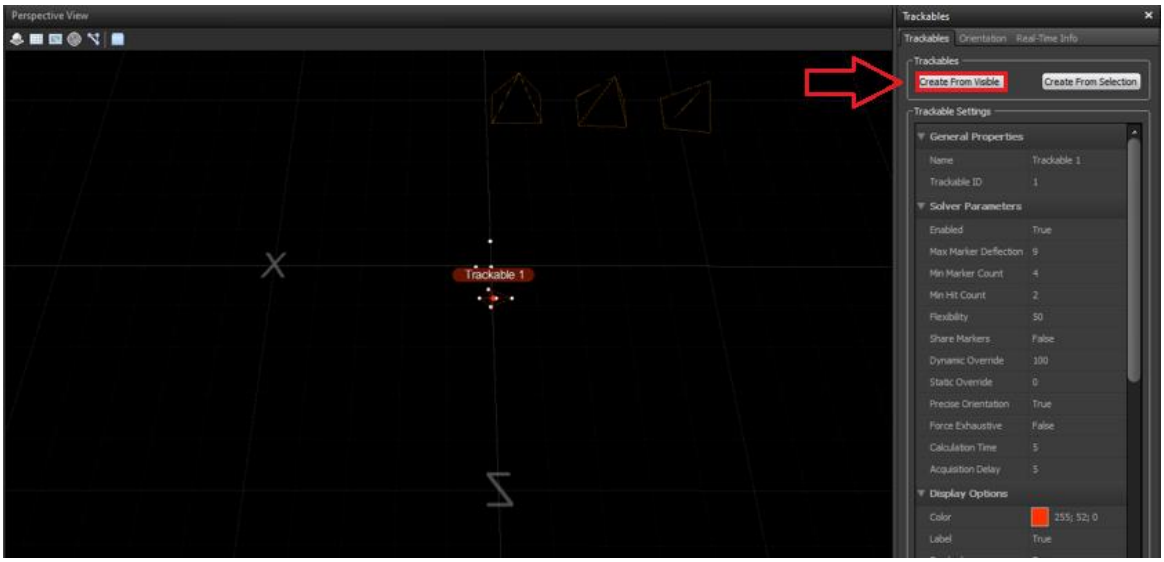
Debido a que el sistema de coordenadas se origina en el centro de la esfera central, es necesario ajustar un *vertical offset,* para que el origen del sistema se desplace hacia arriba y se ubique tangencialmente al punto más superior de la esfera. Este valor corresponde al radio de la esfera (8.1169 mm) y se ubica en la casilla Y del panel "*Capture Volume Translation*" (Anexo A.38). Este paso es significativo porque más adelante se va a medir la distancia del origen del sistema de coordenadas del *Optitrack* al origen del sistema de coordenadas del robot, tocando con el efector final este punto de la esfera central.



Anexo A.38: Vertical *offset* del sistema de coordenadas.

A continuación, se debe crear el "*Trackable*"; este es el objeto virtual que va a representar el cuerpo rígido dentro del programa. Para ello, una vez el puntero se encuentre dentro

del campo visual de las cámaras, se da *click* en "*Create From Visible*" del panel "*Trackables*".

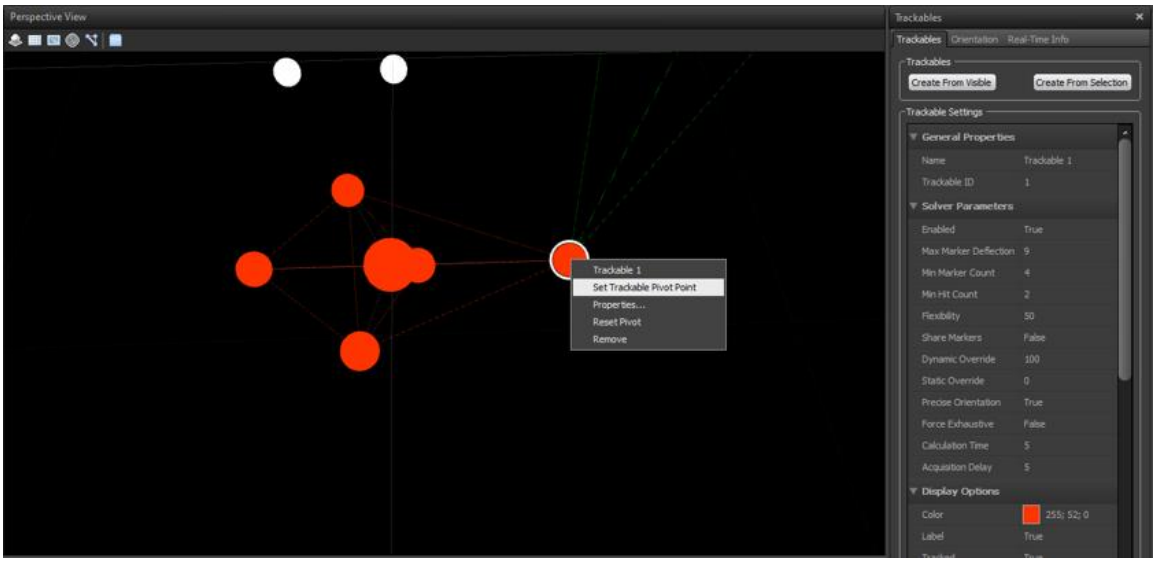


Anexo A.39: Creación del *Trackable.*

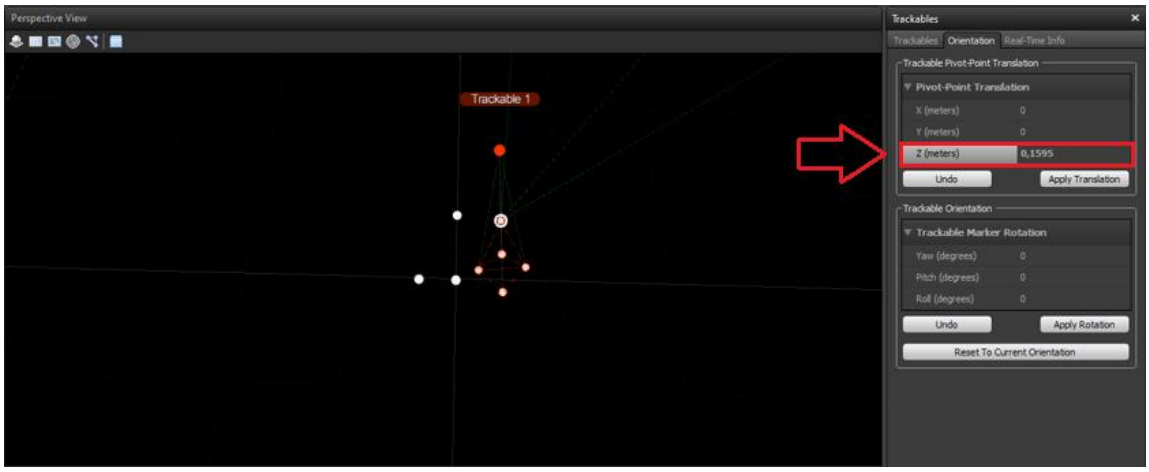
Una de las propiedades más importantes del *Trackable*, es el *Pivot Point*. Este corresponde al punto que el *Optitrack* va a medir (posición y orientación) en todo momento, siempre que el *Trackable* esté dentro del campo de visión de las cámaras. Como se puede observar, el programa lo ubica por defecto en el centroide de la figura geométrica que forman las esferas retrorreflectoras del cuerpo rígido. Sin embargo, es necesario que el *Pivot Point* coincida con la punta del *pointer* para medir la posición de las fiducias. Para ello, se ubica el cuerpo rígido en una superficie que asegure que la línea media del objeto esté paralela al eje Z de la escuadra de calibración. Esto se debe a que la traslación en cualquiera de los ejes XYZ se realiza respecto a las coordenadas globales y no respecto a las coordenadas del objeto.

Anexo A.40: Definición del *Pivot Point*.

Luego, habiendo medido de antemano la distancia de la punta del *pointer* hasta la esfera retrorreflectora más cercana a esta (debe coincidir con la línea media del objeto), se define el nuevo *Pivot Point* en esta esfera, para en la pestaña "*Orientation*" del panel "*Trackables*" ingresar el valor de la distancia medida en la casilla Z. Esta es la distancia en Z (0,1595 m) que se debe desplazar el nuevo *Pivot Point* desde el *Pivot Point* de la esfera retrorreflectora.

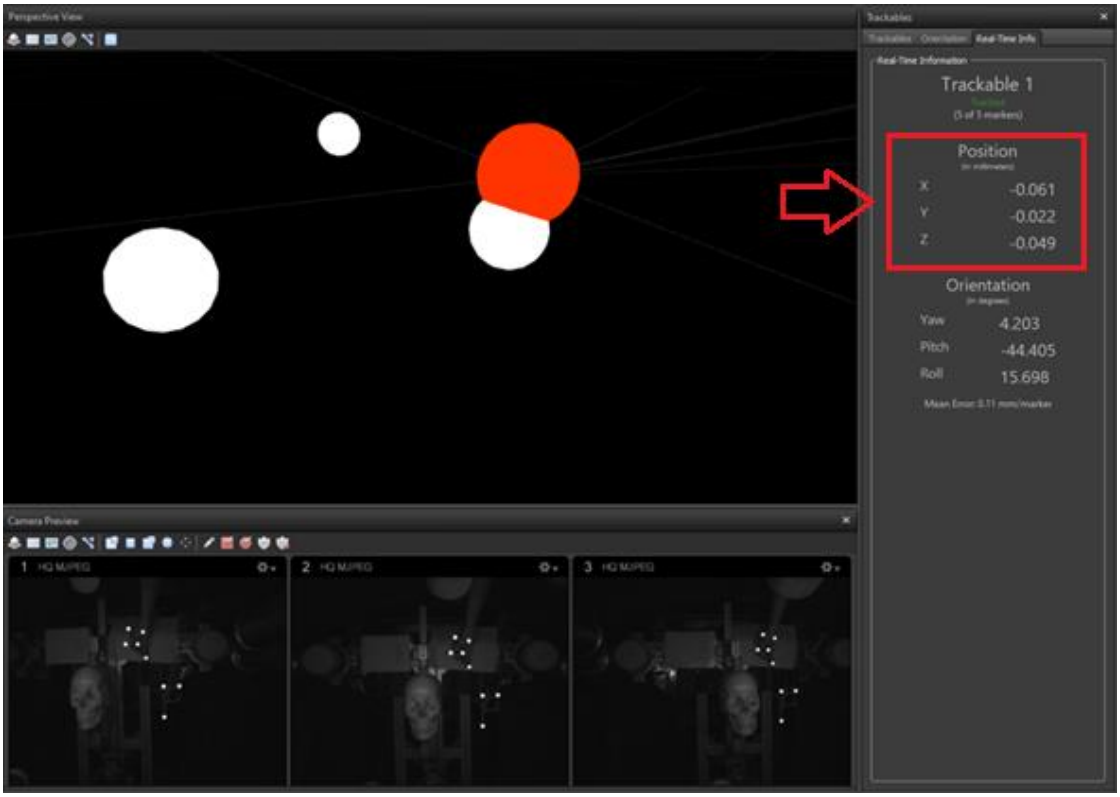


Anexo A.41: Imagen *Optitrack* definición del *Pivot Point*.



Anexo A.42: Desplazamiento del *Pivot Point* sobre el eje Z.

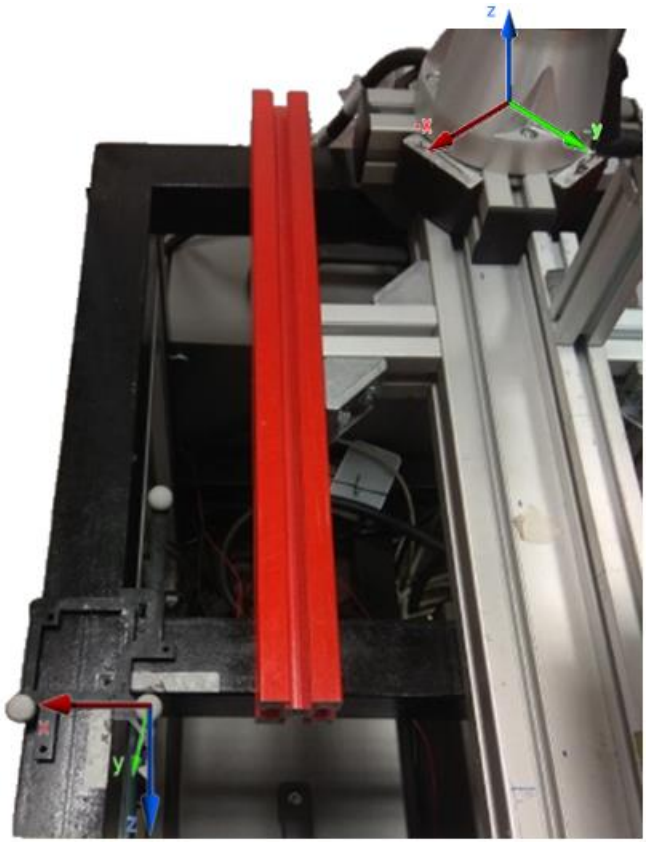
Después de realizar este proceso, se puede verificar que el *Trackable* está bien configurado si al ubicar el *pointer* sobre la esfera central de la escuadra, las posiciones medidas en tiempo real (pestaña "*Real-Time Info*") son 0 o próximas a 0. *El Pivot Point* es representado por una esfera roja en el programa.



Anexo A.43: *Trackable* se ubica en el origen del sistema de coordenadas.

Nótese como al inferior de la figura (Anexo A.43), se observan 3 paneles que representan el campo visual capturado por cada cámara infrarroja. Es fundamental que el cráneo esté dentro de dicho campo visual al momento de medir las posiciones de las fiducias.

El siguiente paso consiste en calcular la matriz de transformación que convertirá el sistema de referencia del *Optitrack* al sistema de referencia del robot; esto con el fin de transformar las posiciones de las fiducias medidas con el *pointer*. En primer lugar se deben calcular las matrices de rotación y, en segundo lugar, se deben medir las distancias del origen de un sistema al origen del otro, para calcular la matriz de traslación.



Anexo A.44: Sistemas de referencia *Optitrack* y UR3e.

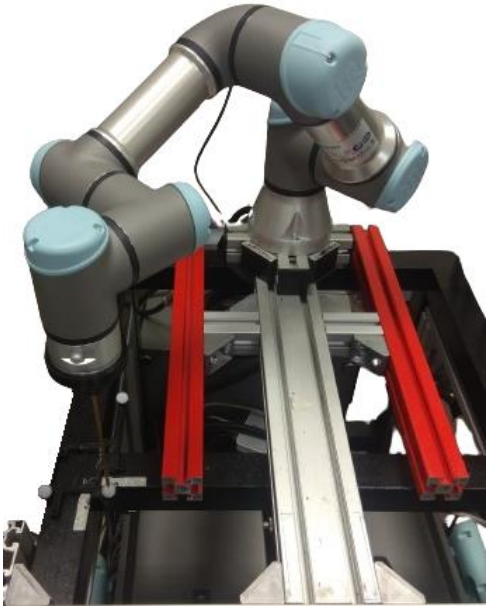
En la figura (Anexo A.44), se puede observar que se necesita realizar un giro sobre el eje X y luego un giro sobre el eje Z, para orientar el sistema de coordenadas del *Optitrack* al del robot. Cabe mencionar que el sistema de referencia del *Optitrack* es "*Right handed*", al igual que el del robot.

Las matrices de rotación son las siguientes:

$$T(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\,\alpha & -sin\,\alpha & 0 \\ 0 & sin\,\alpha & cos\,\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, donde\ \alpha = 270° \qquad (A.\ 1)$$

$$T(z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, donde\ \theta = 135°$$

A continuación, se hallan las distancias de un sistema al otro, haciendo uso del efector final del robot y el *Teach Pendant*.



Anexo A.45: Efector final y la esfera retrorreflectora en contacto.

Anexo A.46: Posiciones XYZ de la esfera retrorreflectora.

Después de orientar el sistema de coordenadas del *Optitrack* y coincidir los ejes con el sistema de coordenadas del robot, solo hace falta trasladarlo con las posiciones encontradas.

Matriz de traslación:

$$T = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}, donde \begin{matrix} X = -458.06\, mm \\ Y = -115.48\, mm \\ Z = -70.100\, mm \end{matrix}$$

Finalmente, se calcula la matriz de transformación:

$$M_T = T * R_Z * R_X \tag{A. 2}$$

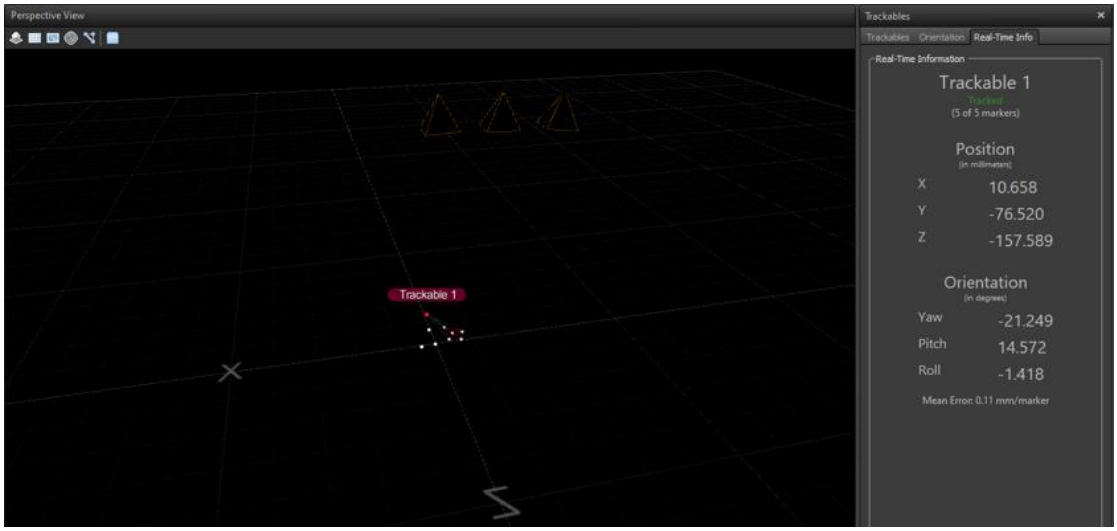
$$M_T = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_T = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & X \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & Y \\ 0 & \sin\alpha & \cos\alpha & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De esta manera, cada punto medido con el *Optitrack* debe ser multiplicado por esta matriz para ser convertido al sistema de coordenadas del robot. Como se mencionó anteriormente, la fabricación de la escuadra de calibración y la forma irregular de las esferas retrorreflectoras usadas en este, generan un error que debe ser corregido. Esto se traduce en una sintonización de los dos ángulos y las distancias utilizados en la matriz de transformación. Para este propósito se utilizó una posición del efector final que fuera fácil de verificar. Esta posición es 0 en Y, 0 en Z y -350 en X.

Anexo A.47: Posición de verificación.

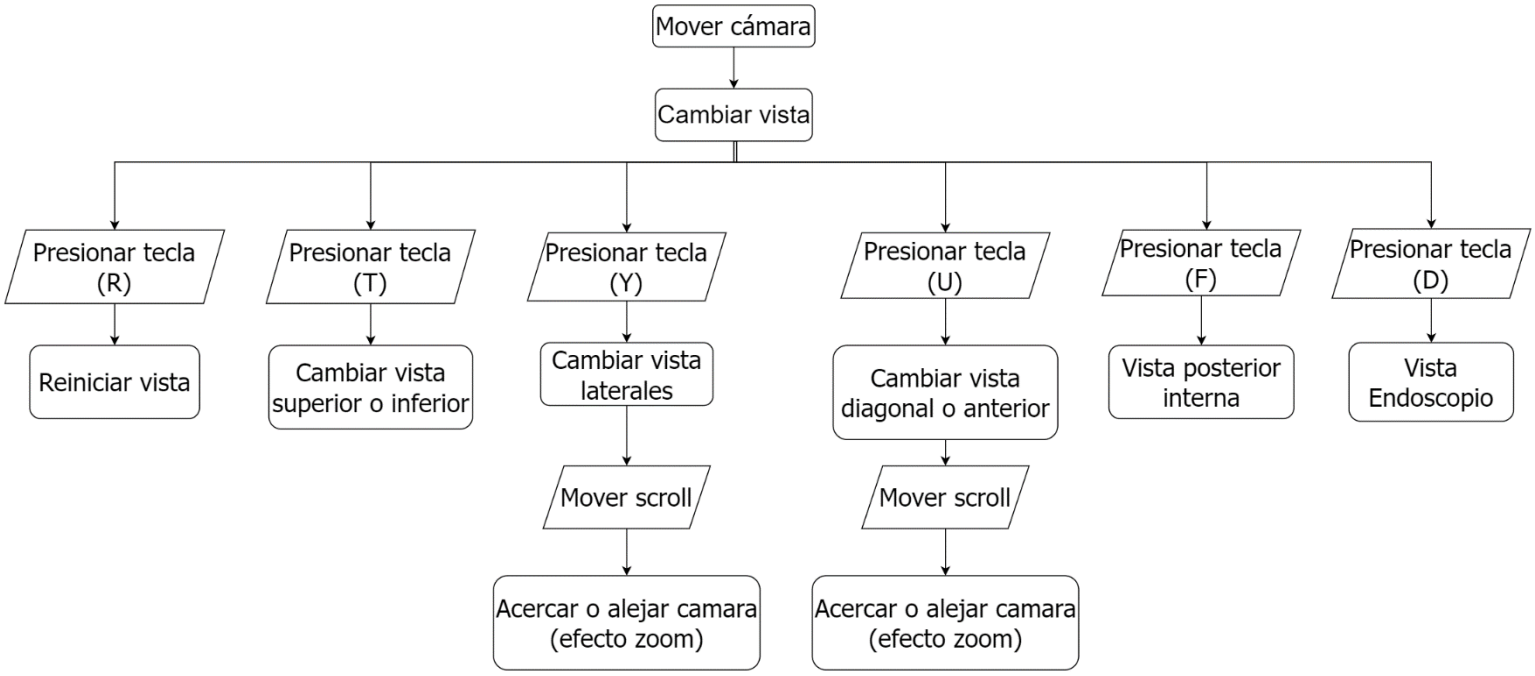
Acto seguido, se ubica el *pointer* en la punta del efector final del robot y se anotan las posiciones.



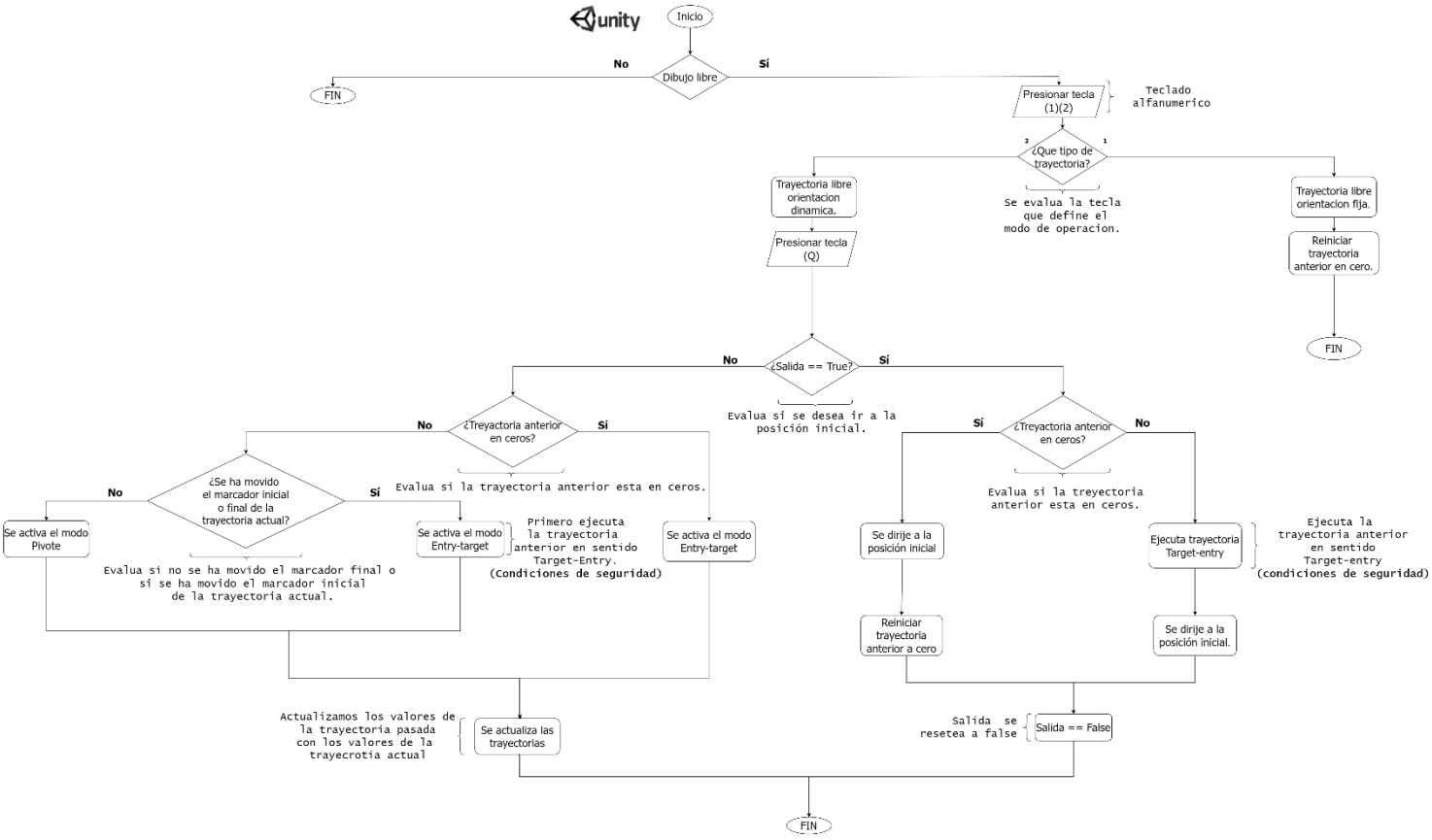
Anexo A.48: Posiciones del efector final.

Estas posiciones se multiplican por la matriz de transformación y se ajustan los ángulos y las distancias de esta, hasta que se obtenga el mismo resultado que aparece en el *Teach Pendant*. De esta forma, se asegura que la matriz de transformación sea más exacta.

A.4. Diagramas de flujo



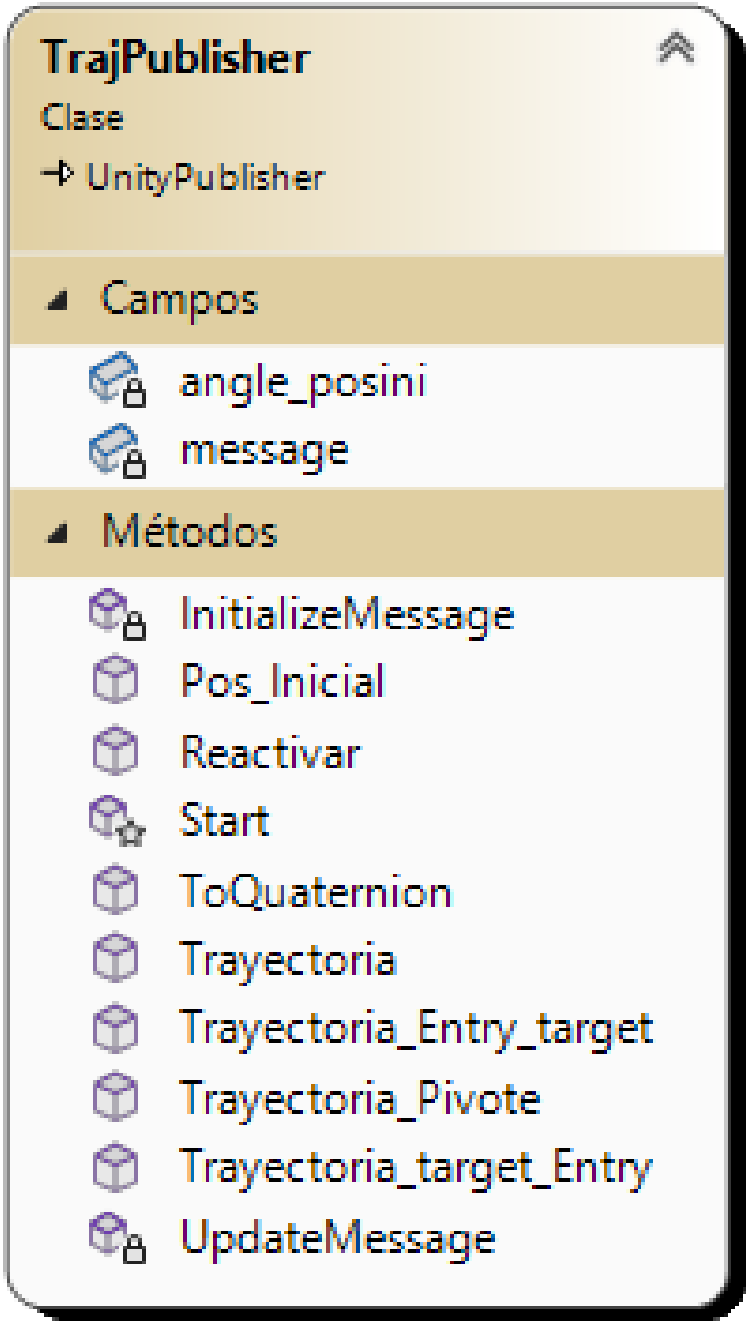
Anexo A.49: Diagrama de flujo observador II.



Anexo A.50: Diagrama de flujo “Dibujo Libre”.

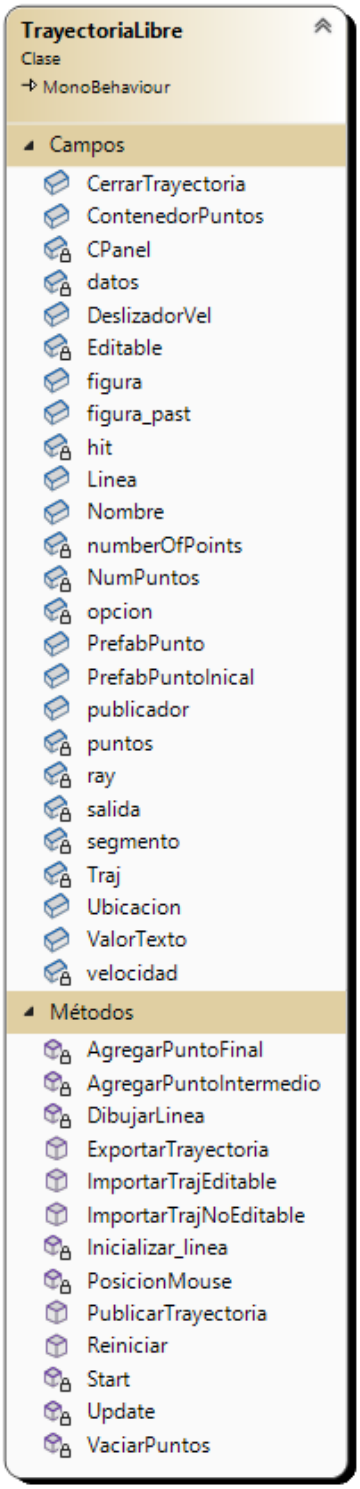
A.5. Diagramas de clases

A.5.1. Publicador ROS modificado.

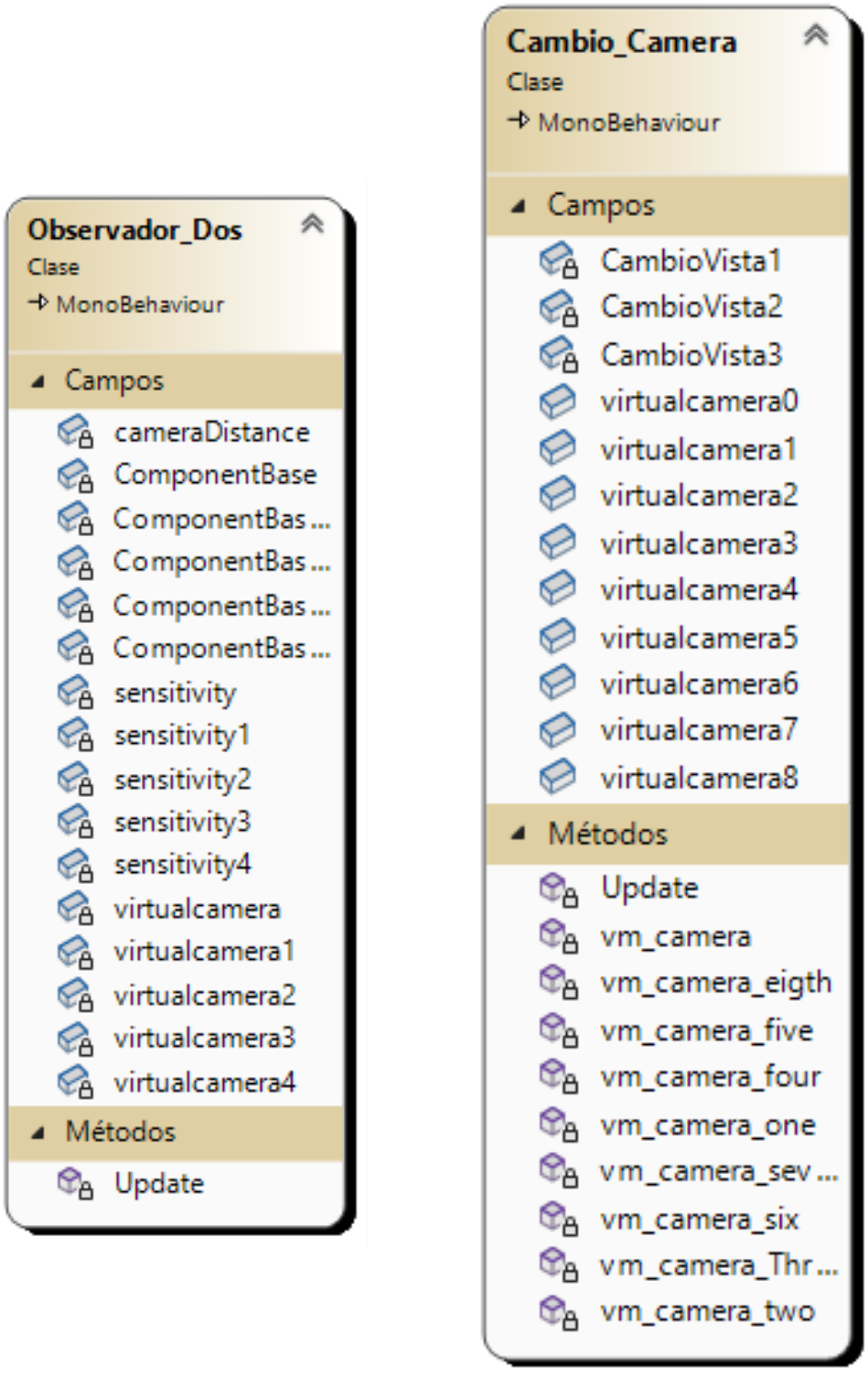


Anexo A.51: *TrajPublisher* modificado.

A.5.2. Control de la interfaz modificado.



Anexo A.52: Control interfaz modificado.



Anexo A.53: Observador dos y cambio de cámara.

Bibliografía

[1] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, and B. Siciliano, "Portable dVRK: an augmented V-REP simulator of the da Vinci Research Kit," *Acta Polytechnica Hungarica*, vol. 16, no. 8, 2019, Accessed: Nov. 28, 2022. [Online]. Available: https://github.com/jhu-dvrk

[2] J. J. Kanitra *et al.*, "Transference of skills in robotic vs. laparoscopic simulation: a randomized controlled trial," *BMC Surg*, vol. 21, no. 1, pp. 1–8, Dec. 2021, doi: 10.1186/S12893-021-01385-Y/FIGURES/3.

[3] C. Kwan *et al.*, "Co-registration of Imaging Modalities (MRI, CT and PET) to Perform Frameless Stereotaxic Robotic Injections in the Common Marmoset," *Neuroscience*, vol. 480, pp. 143–154, Jan. 2022, doi: 10.1016/J.NEUROSCIENCE.2021.11.009.

[4] S. Chumnanvej, B. M. Pillai, S. Chalongwongse, and J. Suthakorn, "Endonasal endoscopic transsphenoidal approach robot prototype: A cadaveric trial," *Asian J Surg*, vol. 44, no. 1, pp. 345–351, Jan. 2021, doi: 10.1016/J.ASJSUR.2020.08.011.

[5] B. Rodríguez-Vila *et al.*, "A low-cost pedagogical environment for training on technologies for image-guided robotic surgery," *IFMBE Proc*, vol. 68, no. 2, pp. 821–824, 2018, doi: 10.1007/978-981-10-9038-7_151/TABLES/1.

[6] C. Simmonds, M. Brentnall, and J. Lenihan, "Evaluation of a novel universal robotic surgery virtual reality simulation proficiency index that will allow comparisons of users across any virtual reality simulation curriculum," *Surg Endosc*, vol. 35, no. 10, pp. 5867–5875, Oct. 2021, doi: 10.1007/S00464-021-08609-5/FIGURES/2.

[7] C. Fernández-Riomalo, *et al.*, H. G.-M.-R. F., and undefined 2016, "Diseño y modelado del robot PA-10 virtual para aplicaciones quirúrgicas," *scielo.org.co*, Accessed: Nov. 28, 2022. [Online]. Available: http://www.scielo.org.co/pdf/rfing/v25n42/v25n42a03.pdf

[8] A. Vivas and J. M. Sabater, "UR5 Robot Manipulation using Matlab/Simulink and ROS," *2021 IEEE International Conference on Mechatronics and Automation, ICMA 2021*, pp. 338–343, Aug. 2021, doi: 10.1109/ICMA52036.2021.9512650.

[9] D. A. Gilley and C. P. Sundaram, "Impact of Virtual Reality Simulators in Training of Robotic Surgery," *Robotics in Genitourinary Surgery*, pp. 163–168, 2011, doi: 10.1007/978-1-84882-114-9_14.

[10] S. B. Ahmad, M. J. Rice, C. Chang, A. H. Zureikat, H. J. Zeh, and M. E. Hogg, "dV-Trainer vs. da Vinci Simulator: Comparison of Virtual Reality Platforms for Robotic Surgery," *Journal of Surgical Research*, vol. 267, pp. 695–704, Nov. 2021, doi: 10.1016/J.JSS.2021.06.036.

[11] L. Sessa *et al.*, "Face and content validity of Xperience™ Team Trainer: bed-side assistant training simulator for robotic surgery," *Updates Surg*, vol. 70, no. 1, pp. 113–119, Mar. 2018, doi: 10.1007/S13304-017-0509-X/TABLES/2.

[12] M. İ. C. Dede, T. Büyüköztekin, Ş. Hanalioğlu, İ. Işıkay, and M. Berker, "Enabling personalization of a robotic surgery procedure via a surgery training simulator," *Robotica*, pp. 1–16, Jul. 2022, doi: 10.1017/S0263574722001023.

[13] A. I. Moral *et al.*, "3D endoscopic approach for endonasal sinus surgery," *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, pp. 4683–4686, 2007, doi: 10.1109/IEMBS.2007.4353385.

[14] S. A. H. Perez, K. Harada, and M. Mitsuishi, "Haptic Virtual Fixtures to Assist Endonasal Micro Robotic Surgery through Virtual Reality Simulation," *MHS 2018 - 2018 29th International Symposium on Micro-NanoMechatronics and Human Science*, Dec. 2018, doi: 10.1109/MHS.2018.8886945.

[15] A. Djuric, J. Rickli, J. Sefcovic, D. Hutchison, and M. M. Goldin, "Integrating Collaborative Robots in Engineering and Engineering Technology Programs," *ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE)*, vol. 5, Jan. 2019, doi: 10.1115/IMECE2018-88147.

[16] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada, "The evolution of robotics research," *IEEE Robot Autom Mag*, vol. 14, no. 1, pp. 90–103, Mar. 2007, doi: 10.1109/MRA.2007.339608.

[17] D. Mukherjee, H. K. Tripathy, and S. Mishra, "Scope of Medical Bots in Clinical Domain," *Studies in Computational Intelligence*, vol. 936, pp. 339–363, 2021, doi: 10.1007/978-981-33-4698-7_17/COVER.

[18] S. M. Mavadati, H. Feng, M. Salvador, S. Silver, A. Gutierrez, and M. H. Mahoor, "Robot-based therapeutic protocol for training children with Autism," *25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016*, pp. 855–860, Nov. 2016, doi: 10.1109/ROMAN.2016.7745219.

[19] A. Nagendran, A. Steed, B. Kelly, and Y. Pan, "Symmetric telepresence using robotic humanoid surrogates," *Comput Animat Virtual Worlds*, vol. 26, no. 3–4, pp. 271–280, May 2015, doi: 10.1002/CAV.1638.

[20] I. P. Ktistakis and N. Bourbakis, "A multimodal human-machine interaction scheme for an intelligent robotic nurse," *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, vol. 2018-November, pp. 749–756, Dec. 2018, doi: 10.1109/ICTAI.2018.00118.

[21] S. Balasubramanian, J. Klein, and E. Burdet, "Robot-assisted rehabilitation of hand function," *Curr Opin Neurol*, vol. 23, no. 6, pp. 661–670, Dec. 2010, doi: 10.1097/WCO.0B013E32833E99A4.

[22] H. I. Krebs and B. T. Volpe, "Rehabilitation Robotics," *Handb Clin Neurol*, vol. 110, pp. 283–294, Jan. 2013, doi: 10.1016/B978-0-444-52901-5.00023-X.

[23] V. Kaul, S. Enslin, and S. A. Gross, "History of artificial intelligence in medicine," *Gastrointest Endosc*, vol. 92, no. 4, pp. 807–812, Oct. 2020, doi: 10.1016/J.GIE.2020.06.040.

[24] Patil Darshan Rajkumar, Aishna Gupta, and Kunal Kulkarni, "The Scope and Application of Nanorobots in Biomedicine." Accessed: Nov. 27, 2022. [Online]. Available: https://www.researchgate.net/profile/Kunal_Kulkarni3/publication/354203796_THE_SCOPE_AND_APPLICATION_OF_NANOROBOTS_IN_BIOMEDICINE/links/612bd4310360302a0066f168/THE-SCOPE-AND-APPLICATION-OF-NANOROBOTS-IN-BIOMEDICINE.pdf#page=177

[25] W. Tao and N. A. Peppas, "Robotic pills for gastrointestinal-tract-targeted oral mRNA delivery," *Matter*, vol. 5, no. 3, pp. 775–777, Mar. 2022, doi: 10.1016/J.MATT.2022.02.008.

[26] W. R. Chitwood *et al.*, "Robotic Surgical Training in an Academic Institution," *Ann Surg*, vol. 234, no. 4, p. 475, 2001, doi: 10.1097/00000658-200110000-00007.

[27] D. J. Pangal, G. Kugener, S. Shahrestani, F. Attenello, G. Zada, and D. A. Donoho, "A Guide to Annotation of Neurosurgical Intraoperative Video for Machine Learning Analysis and Computer Vision," *World Neurosurg*, vol. 150, pp. 26–30, Jun. 2021, doi: 10.1016/J.WNEU.2021.03.022.

[28] A. Khatavkar, N. Kharade, G. Navale, and T. Khadtare, "COVID-19 pandemic deep learning implementations of prediction of disease with data analysis and real-time face-mask detection with camera," *Proceedings - 2021 1st IEEE International Conference on Artificial Intelligence and Machine Vision, AIMV 2021*, 2021, doi: 10.1109/AIMV53313.2021.9670960.

[29] Y. Chen, S. Zhang, Z. Wu, B. Yang, Q. Luo, and K. Xu, "Review of surgical robotic systems for keyhole and endoscopic procedures: state of the art and perspectives," *Frontiers of Medicine 2020 14:4*, vol. 14, no. 4, pp. 382–403, Jul. 2020, doi: 10.1007/S11684-020-0781-X.

[30] N. Feizi, M. Tavakoli, R. V. Patel, and S. F. Atashzar, "Robotics and AI for Teleoperation, Tele-Assessment, and Tele-Training for Surgery in the Era of COVID-19: Existing Challenges, and Future Vision," *Front Robot AI*, vol. 8, p. 16, Apr. 2021, doi: 10.3389/FROBT.2021.610677/BIBTEX.

[31] J. C. Fraile, J. Pérez-Turiel, JLGonzález-Sánchez, J. López-Cruzado, and J. L. Rodríguez, "Experiencias en el desarrollo de una aplicación robótica con control de fuerza para taladrado de huesos," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 5, no. 2, pp. 93–106, Apr. 2008, doi: 10.1016/S1697-7912(08)70148-1.

[32] V. Seifert, "Intraoperative MRI in neurosurgery: Technical overkill or the future of brain surgery?," *Neurol India*, vol. 51, no. 3, p. 329, Sep. 2003, Accessed: Nov. 28, 2022. [Online]. Available: https://www.neurologyindia.com/article.asp?issn=0028-3886;year=2003;volume=51;issue=3;spage=329;epage=332;aulast=Seifert

[33] K. Correa-Arana, O. A. Vivas-Albán, and J. M. Sabater-Navarro, "Neurosurgery and brain shift: review of the state of the art and main contributions of robotics Neurocirugía y desplazamientos cerebrales: una revisión del estado del arte y principales contribuciones desde la robótica," vol. 20, no. 40, pp. 125–138, 2017.

[34] G. Haskins, U. Kruger, and P. Yan, "Deep learning in medical image registration: a survey," *Mach Vis Appl*, vol. 31, no. 1, pp. 1–18, Jan. 2020, doi: 10.1007/S00138-020-01060-X/FIGURES/10.

[35] M. A. Viergever, J. B. A. Maintz, S. Klein, K. Murphy, M. Staring, and J. P. W. Pluim, "A survey of medical image registration – under review," *Med Image Anal*, vol. 33, pp. 140–144, Oct. 2016, doi: 10.1016/J.MEDIA.2016.06.030.

[36] F. Alam, S. U. Rahman, S. Ullah, and K. Gulati, "Medical image registration in image guided surgery: Issues, challenges and research opportunities," *Biocybern Biomed Eng*, vol. 38, no. 1, pp. 71–89, Jan. 2018, doi: 10.1016/J.BBE.2017.10.001.

[37] I. Leizea, A. Mendizabal, H. Alvarez, I. Aguinaga, D. Borro, and E. Sanchez, "Real-Time Visual Tracking of Deformable Objects in Robot-Assisted Surgery," *IEEE Comput Graph Appl*, vol. 37, no. 1, pp. 56–68, Jan. 2017, doi: 10.1109/MCG.2015.96.

[38] N. D. Serej, A. Ahmadian, S. Mohagheghi, and S. M. Sadrehosseini, "A projected landmark method for reduction of registration error in image-guided surgery systems," *Int J Comput Assist Radiol Surg*, vol. 10, no. 5, pp. 541–554, May 2015, doi: 10.1007/S11548-014-1075-Z/TABLES/3.

[39] R. Panda, S. Agrawal, M. Sahoo, and R. Nayak, "A novel evolutionary rigid body docking algorithm for medical image registration," *Swarm Evol Comput*, vol. 33, pp. 108–118, Apr. 2017, doi: 10.1016/J.SWEVO.2016.11.002.

[40] Z. Faraji-Dana, A. L. D. Mariampillai, B. A. Standish, V. X. D. Yang, and M. K. K. Leung, "Machine-Vision Image-Guided Surgery for Spinal and Cranial Procedures," *Handbook of Robotic and Image-Guided Surgery*, pp. 551–574, Jan. 2020, doi: 10.1016/B978-0-12-814245-5.00032-3.

[41] D. Zikic, B. Glocker, and A. Criminisi, "Encoding atlases by randomized classification forests for efficient multi-atlas label propagation," *Med Image Anal*, vol. 18, no. 8, pp. 1262–1273, Dec. 2014, doi: 10.1016/J.MEDIA.2014.06.010.

[42] M. Regodic´, R. Regodic´, Z. Bardosi, and W. Freysinger, "Automated fiducial marker detection and localization in volumetric computed tomography images: a three-step hybrid approach with deep learning," *https://doi.org/10.1117/1.JMI.8.2.025002*, vol. 8, no. 2, p. 025002, Apr. 2021, doi: 10.1117/1.JMI.8.2.025002.

[43] I. Reducindo, E. R. Arce-Santana, D.U. Campos-Delgado, and F. Vigueras-Gómez, "Registro elástico de imágenes médicas multimodales basado en medidas de variabilidad local." Accessed: Nov. 26, 2022. [Online]. Available: https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-95322013000100001

[44] N. Zagzoog, G. Zadeh, V. Lin, and V. X. D. Yang, "Perspective review on applications of optics in skull base surgery," *Clin Neurol Neurosurg*, vol. 212, p. 107085, Jan. 2022, doi: 10.1016/J.CLINEURO.2021.107085.

[45] É. Léger, J. Reyes, S. Drouin, D. L. Collins, T. Popa, and M. Kersten-Oertel, "Gesture-based registration correction using a mobile augmented reality image-guided neurosurgery system," *Healthc Technol Lett*, vol. 5, no. 5, pp. 137–142, Oct. 2018, doi: 10.1049/HTL.2018.5063.

[46] J. Shen *et al.*, "Intraoperative Ultrasonography-based Augmented Reality For Application In Image Guided Robotic Surgery," *Int J Comput Assist Radiol Surg*, vol. 13, no. S1, pp. S45–S46, Jun. 2018, doi: 10.1007/S11548-018-1766-Y.

[47] H. H. Glas, J. Kraeima, P. M. A. van Ooijen, F. K. L. Spijkervet, L. Yu, and M. J. H. Witjes, "Augmented Reality Visualization for Image-Guided Surgery: A Validation Study Using a Three-Dimensional Printed Phantom," *Journal of Oral and Maxillofacial Surgery*, vol. 79, no. 9, pp. 1943.e1-1943.e10, Sep. 2021, doi: 10.1016/J.JOMS.2021.04.001.

[48] K. Madhavan, J. P. G. Kolcun, L. O. Chieng, and M. Y. Wang, "Augmented-reality integrated robotics in neurosurgery: are we there yet?," *Neurosurg Focus*, vol. 42, no. 5, p. E3, May 2017, doi: 10.3171/2017.2.FOCUS177.

[49] C. Faria, W. Erlhagen, M. Rito, E. De Momi, G. Ferrigno, and E. Bicho, "Review of robotic technology for stereotactic neurosurgery," *IEEE Rev Biomed Eng*, vol. 8, pp. 125–137, 2015, doi: 10.1109/RBME.2015.2428305.

[50] E. Dimitrakakis *et al.*, "Robotic Handle Prototypes for Endoscopic Endonasal Skull Base Surgery: Pre-clinical Randomised Controlled Trial of Performance and Ergonomics," *Ann Biomed Eng*, vol. 50, no. 5, pp. 549–563, May 2022, doi: 10.1007/S10439-022-02942-Z/TABLES/1.

[51] E. Hempel *et al.*, "An MRI-Compatible Surgical Robot for Precise Radiological Interventions," *https://mc.manuscriptcentral.com/tcas*, vol. 8, no. 4, pp. 180–191, 2010, doi: 10.3109/10929080309146052.

[52] C. Neudorfer, S. Hunsche, M. Hellmich, F. El Majdoub, and M. Maarouf, "Comparative Study of Robot-Assisted versus Conventional Frame-Based Deep Brain Stimulation Stereotactic Neurosurgery," *Stereotact Funct Neurosurg*, vol. 96, no. 5, pp. 327–334, Dec. 2018, doi: 10.1159/000494736.

[53] Q. H. Li, L. Zamorano, A. Pandya, R. Perez, J. Gong, and F. Diaz, "The Application Accuracy of the NeuroMate Robot—A Quantitative Comparison with Frameless and Frame-Based Surgical Localization Systems," *https://mc.manuscriptcentral.com/tcas*, vol. 7, no. 2, pp. 90–98, Jan. 2010, doi: 10.3109/10929080209146020.

[54] N. J. Brandmeir, S. Savaliya, P. Rohatgi, and M. Sather, "The comparative accuracy of the ROSA stereotactic robot across a wide range of clinical applications and registration techniques," *J Robot Surg*, vol. 12, no. 1, pp. 157–163, Mar. 2018, doi: 10.1007/S11701-017-0712-2/TABLES/4.

[55] G. Widmann, "Image-guided surgery and medical robotics in the cranial area," *Biomed Imaging Interv J*, vol. 3, no. 1, 2007, doi: 10.2349/BIIJ.3.1.E11.

[56] E. Dimitrakakis *et al.*, "A Spherical Joint Robotic End-Effector for the Expanded Endoscopic Endonasal Approach," *https://doi.org/10.1142/S2424905X21500021*, vol. 5, no. 3–4, Feb. 2021, doi: 10.1142/S2424905X21500021.

[57] R. A. Bly *et al.*, "Multiportal robotic access to the anterior cranial fossa: A surgical and engineering feasibility study," *Otolaryngology - Head and Neck Surgery (United States)*, vol. 149, no. 6, pp. 940–946, Dec. 2013, doi: 10.1177/0194599813509587/ASSET/IMAGES/LARGE/10.1177_0194599813509587-FIG3.JPEG.

[58] T. Ogiwara, T. Goto, A. Nagm, and K. Hongo, "Endoscopic endonasal transsphenoidal surgery using the iArmS operation support robot: initial experience in 43 patients," *Neurosurg Focus*, vol. 42, no. 5, p. E10, May 2017, doi: 10.3171/2017.3.FOCUS16498.

[59] R. Ramakrishna, L. J. Kim, R. A. Bly, K. Moe, and M. Ferreira, "Transorbital neuroendoscopic surgery for the treatment of skull base lesions," *Journal of Clinical Neuroscience*, vol. 24, pp. 99–104, Feb. 2016, doi: 10.1016/J.JOCN.2015.07.021.

[60] J. Meulemans *et al.*, "Transoral robotic surgery (TORS) using the da Vinci Xi: prospective analysis of feasibility, safety, and outcomes," *Head Neck*, vol. 44, no. 1, pp. 143–157, Jan. 2022, doi: 10.1002/HED.26902.

[61] D. J. Pangal *et al.*, "Robotic and robot-assisted skull base neurosurgery: systematic review of current applications and future directions," *Neurosurg Focus*, vol. 52, no. 1, p. E15, Jan. 2022, doi: 10.3171/2021.10.FOCUS21505.

[62] G. Larca T., J. Arzola R., and O. Pereira B., "Segmentación de Imágenes Médicas Digitales mediante Técnicas de Clustering," *APORTE SANTIAGUINO*, vol. 3, no. 1, p. 108, Jul. 2010, doi: 10.32911/as.2010.v3.n1.428.

[63] Peña Acevedo J., "Segmentación de cerebros en imágenes de resonancia magnética usando superficies deformables," Centro de Investigación en Matemáticas, Guanajuato, 2004.

[64] N. Sharma *et al.*, "Automated medical image segmentation techniques," *J Med Phys*, vol. 35, no. 1, p. 3, 2010, doi: 10.4103/0971-6203.58777.

[65] K. Dorsa Sadat, "volumetric analysis of plexiform neurofibroma," Université de Montréal, Montréal, 2022.

[66] A. Fedorov *et al.*, "3D Slicer as an image computing platform for the Quantitative Imaging Network," *Magn Reson Imaging*, vol. 30, no. 9, pp. 1323–1341, Nov. 2012, doi: 10.1016/j.mri.2012.05.001.

[67] 3D Slicer, "3D Slicer image computing platform." Accessed: Jun. 19, 2023. [Online]. Available: https://www.slicer.org/

[68] T. Frank, A. Krieger, S. Leonard, N. A. Patel, and J. Tokuda, "ROS-IGTL-Bridge: an open network interface for image-guided therapy using the ROS environment.," *Int J*

Comput Assist Radiol Surg, vol. 12, no. 8, pp. 1451–1460, Aug. 2017, doi: 10.1007/s11548-017-1618-1.

[69] L. Ma, C. Zhang, W. Wu, J. Chang, and J. Cui, "Secondary development based on 3D Slicer extension modules," *Journal of Complexity in Health Sciences*, vol. 3, no. 1, pp. 73–80, Jun. 2020, doi: 10.21595/chs.2020.21267.

[70] W. D. Bidgood, S. C. Horii, F. W. Prior, and D. E. Van Syckle, "Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging," *Journal of the American Medical Informatics Association*, vol. 4, no. 3, pp. 199–212, May 1997, doi: 10.1136/jamia.1997.0040199.

[71] M. Mantri, S. Taran, and G. Sunder, "DICOM Integration Libraries for Medical Image Interoperability: A Technical Review," *IEEE Rev Biomed Eng*, vol. 15, pp. 247–259, 2022, doi: 10.1109/RBME.2020.3042642.

[72] B. K. Sahu and R. Verma, "DICOM search in medical image archive solution e-Sushrut Chhavi," in *2011 3rd International Conference on Electronics Computer Technology*, IEEE, Apr. 2011, pp. 256–260. doi: 10.1109/ICECTECH.2011.5942093.

[73] I. A. Cruz Matías, "Rotaciones multidimensionales generales," Escuela de Ingeniería y Ciencias, Universidad de las Américas Puebla., 2007.

[74] HOLODICOM, "Imagen médica de diagnóstico en realidad aumentada." Accessed: Jun. 27, 2023. [Online]. Available: http://www.holodicom.com/

[75] M. Zhang, W. Liu, N. Weibel, and J. P. Schulze, "A DirectX-Based DICOM Viewer for Multi-user Surgical Planning in Augmented Reality," 2022, pp. 217–228. doi: 10.1007/978-3-031-20716-7_17.

[76] "Introducción Blender manual." Accessed: Jun. 13, 2023. [Online]. Available: https://docs.blender.org/manual/es/latest/getting_started/about/

[77] "Historia de Blender 3d." Accessed: Jun. 13, 2023. [Online]. Available: https://www.artedinamico.com/portal/sitio/articulos_mo_comentarios.php?it=2042&categoria=3,

[78] B. R. Kent, "3DScientificVisualization with Blender" in *3D Scientific Visualization with Blender*®, Morgan & Claypool Publishers, 2015, pp. 1-1-1–6. doi: 10.1088/978-1-6270-5612-0ch1.

[79] B. R. Kent, "Visualizing Astronomical Data with Blender," *Publications of the Astronomical Society of the Pacific*, vol. 125, no. 928, pp. 731–748, Jun. 2013, doi: 10.1086/671412.

[80] S. Rooppakhun, P. Surasith, N. Vatanapatimakul, Y. Kaewprom, and K. Sitthiseripratip, "Craniometric study of Thai skull based on three-dimensional computed tomography (CT) data.," *J Med Assoc Thai*, vol. 93, no. 1, pp. 90–8, Jan. 2010.

[81] "Huesos de la cara: tipos, características y ubicación. (s.f.)." Accessed: Jun. 19, 2023. [Online]. Available: https://psicologiaymente.com/salud/huesos-cara

[82] Alex M. Germann and Varun Kashyap, “Anatomy, Head and Neck, Occipital Bone, Artery, Vein, and Nerve - StatPearls - NCBI Bookshelf.” Accessed: Nov. 26, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK541093/

[83] “Diccionario de cáncer del NCI. Instituto Nacional del Cáncer.” Accessed: Jun. 19, 2023. [Online]. Available: https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/hueso-cigomatico

[84] “Diccionario de cáncer del NCI. (s.f.). Instituto Nacional del Cáncer.” Accessed: Jun. 20, 2023. [Online]. Available: https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/Mandibula

[85] Serrano Cinthia, “Hueso parietal.” Accessed: Jun. 20, 2023. [Online]. Available: https://www.kenhub.com/es/library/anatomia-es/hueso-parietal

[86] blender.org, “Introduction — Blender Manual. (s.f.). Blender Documentation -.” Accessed: Jun. 21, 2023. [Online]. Available: https://docs.blender.org/manual/en/latest/sculpt_paint/sculpting/introduction.html

[87] Y. L. Yap *et al.*, “3D printed bio-models for medical applications,” *Rapid Prototyp J*, vol. 23, no. 2, pp. 227–235, Mar. 2017, doi: 10.1108/RPJ-08-2015-0102.

[88] “Términos direccionales y planos anatómicos.” Accessed: Aug. 13, 2023. [Online]. Available: https://www.kenhub.com/es/library/anatomia-es/terminos-direccionales-y-planos-anatomicos

[89] “La Cirugía endoscópica de tumores de la base del cráneo permite una resección más completa del tumor.” Accessed: Jul. 26, 2023. [Online]. Available: https://www.quironsalud.com/es/comunicacion/actualidad-quironsalud/cirugia-endoscopica-de-tumores-de-la-base-del-craneo

[90] “Cirugía transesfenoidal transnasal endoscópica.” Accessed: Jul. 26, 2023. [Online]. Available: https://www.mayoclinic.org/es/diseases-conditions/acromegaly/multimedia/endoscopic-transnasal-transsphenoidal-surgery/img-20006940

[91] A. Cohen-Gadol, “Patient Positioning,” in *Neurosurgical Atlas*, Neurosurgical Atlas, Inc., 2017. doi: 10.18791/nsatlas.v2.08.

[92] “Posiciones del paciente” Accessed: Aug. 28, 2023. [Online]. Available: https://enfermeriablog.com/posicionesdelpaciente/

[93] “Posiciones en neurocirugía,” *Revista Chilena de Anestesia*, vol. 50, no. 2, 2021, doi: 10.25237/revchilanestv50n02-08.

[94] P. C. De Groen, “History of the Endoscope [Scanning Our Past],” *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1987–1995, Oct. 2017, doi: 10.1109/JPROC.2017.2742858.

[95] A. B. Slatkin, J. Burdick, and W. Grundfest, “The development of a robotic endoscope,” in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent*

Robots and Systems. Human Robot Interaction and Cooperative Robots, IEEE Comput. Soc. Press, pp. 162–171. doi: 10.1109/IROS.1995.526155.

[96] "Endoscopio Rígido - Optomic." Accessed: Jul. 17, 2023. [Online]. Available: https://www.optomic.com/productos/otorrinolaringologia-orl/endoscopio-rigido/endoscopio-rigido/

[97] J. A. Ospina, J. Castellanos, A. Moreno, and N. Gil, "El papel de la cirugía endoscópica endonasal en el manejo Tumores Malignos de Senos Paranasales y Base de Cráneo: Indicaciones, Ventajas y Limitaciones.," *ACTA DE OTORRINOLARINGOLOGÍA & CIRUGÍA DE CABEZA Y CUELLO*, vol. 46, no. 3, pp. 231–239, Apr. 2019, doi: 10.37076/acorl.v46i3.355.

[98] "Qué endoscopio médico elegir - Guías de compra MedicalExpo." Accessed: Jun. 06, 2023. [Online]. Available: https://guide.medicalexpo.com/es/que-endoscopio-medico-elegir/

[99] M. Cuadrado Robles, L. A. Buitrago Orjuela, and A. F. Suarez, "La neuroendoscopia: una técnica revolucionaria en el campo de la neurocirugía," *Revista Repertorio de Medicina y Cirugía*, vol. 31, no. 1, Dec. 2021, doi: 10.31260/RepertMedCir.01217372.1062.

[100] G. Iddan, G. Meron, A. Glukhovsky, and P. Swain, "Wireless capsule endoscopy," *Nature*, vol. 405, no. 6785, pp. 417–417, May 2000, doi: 10.1038/35013140.

[101] B. González-Suárez, S. Galter, and J. Balanzó, "Cápsula endoscópica: fundamentos y utilidad clínica," *Cir Esp*, vol. 81, no. 6, pp. 299–306, Jun. 2007, doi: 10.1016/S0009-739X(07)71328-3.

[102] "Instrumentos para la cirugía endonasal de la base del cráneo." Accessed: Jul. 26, 2023. [Online]. Available: https://www.tecnomed.com.mx/productos/craneo-cappabianca-storz.pdf

[103] "'urdf - ROS Wiki'. Documentation - ROS Wiki." Accessed: Jul. 10, 2023. [Online]. Available: http://wiki.ros.org/urdf

[104] M. Feder, A. Giusti, and R. Vidoni, "An approach for automatic generation of the URDF file of modular robots from modules designed using SolidWorks," *Procedia Comput Sci*, vol. 200, pp. 858–864, 2022, doi: 10.1016/j.procs.2022.01.283.

[105] "Robotics in UR Robots." Accessed: Sep. 02, 2023. [Online]. Available: https://cdck-file-uploads-global.s3.dualstack.us-west-2.amazonaws.com/business7/uploads/universal_robots/original/2X/0/0d9757b3000c8bf414a33a9fd76b40d9481d2019.pdf

[106] "Unity - Manual: Cámara. (s.f.). Unity - Manual: Unity User Manual 2022.3 (LTS)." Accessed: Jun. 12, 2023. [Online]. Available: https://docs.unity3d.com/es/530/Manual/class-Camera.html

[107] "Cámaras - Unity Manual. (s.f.). Unity - Manual: Unity User Manual 2022.3 (LTS)." Accessed: Jul. 13, 2023. [Online]. Available: https://docs.unity3d.com/es/2018.4/Manual/CamerasOverview.html

[108] "Using Cinemachine | Cinemachine | 2.6.17. (s.f.). Unity - Manual: Unity User Manual 2022.3 (LTS)." Accessed: Jul. 15, 2023. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.cinemachine@2.6/manual/CinemachineUsing.html

[109] "Entendiendo el View Frustum - Unity Manual. (s.f.). Unity - Manual: Unity User Manual." Accessed: Apr. 16, 2023. [Online]. Available: https://docs.unity3d.com/es/2019.4/Manual/UnderstandingFrustum.html

[110] J. S. Montenegro Bravo, J. D. Ruiz Flórez, J. D. Romero Ante, J. Manrrique Córdoba, O. A. Vivas Albán, and J. M. Sabater Navarro, "Generador 3D de trayectorias adaptativas libres de colisiones para un manipulador UR3e con pinza blanda," *Revista Iberoamericana de Automática e Informática industrial*, Jul. 2023, doi: 10.4995/riai.2023.19332.

[111] "Bounding Box." Accessed: Aug. 13, 2023. [Online]. Available: https://encord.com/glossary/bounding-box-definition/

[112] "Bounds." Accessed: Aug. 13, 2023. [Online]. Available: https://docs.unity3d.com/ScriptReference/Bounds.html

[113] "Marcador fiduciario." Accessed: Aug. 13, 2023. [Online]. Available: https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-cancer/def/marcador-fiduciario

[114] C. A. Lores, "Cirugía asistida por computador en ortopedia oncológica," *Revista Colombiana de Ortopedia y Traumatología*, vol. 30, no. 4, pp. 159–166, Dec. 2016, doi: 10.1016/j.rccot.2016.11.003.

[115] F. Suligoj, M. Svaco, B. Jerbic, B. Sekoranja, and J. Vidakovic, "Automated Marker Localization in the Planning Phase of Robotic Neurosurgery," *IEEE Access*, vol. 5, pp. 12265–12274, 2017, doi: 10.1109/ACCESS.2017.2718621.

[116] E. B. Strong, A. Rafii, B. Holhweg-Majert, S. C. Fuller, and M. C. Metzger, "Comparison of 3 Optical Navigation Systems for Computer-Aided Maxillofacial Surgery," *Arch Otolaryngol Head Neck Surg*, vol. 134, no. 10, p. 1080, Oct. 2008, doi: 10.1001/archotol.134.10.1080.

[117] "An introduction of registration errors." Accessed: Sep. 19, 2023. [Online]. Available: https://help.supermap.com/iDesktop/en/tutorial/DataProcessing/Registration/RegistrationError

[118] "Quaternion." Accessed: Sep. 02, 2023. [Online]. Available: https://docs.unity3d.com/ScriptReference/Quaternion.html

[119] "REP 103: Standard Units of Measure and Coordinate Conventions." Accessed: Sep. 02, 2023. [Online]. Available: https://www.ros.org/reps/rep-0103.html

[120] "Explanation on robot orientation." Accessed: Sep. 02, 2023. [Online]. Available: https://www.universal-robots.com/articles/ur/application-installation/explanation-on-robot-orientation/

[121] "Other Explicit Representation for the Orientation in Robotics: Roll-Pitch-Yaw Angles." Accessed: Sep. 02, 2023. [Online]. Available: https://mecharithm.com/learning/lesson/explicit-representations-orientation-robotics-roll-pitch-yaw-angles-15

[122] "Transform.eulerAngles." Accessed: Sep. 02, 2023. [Online]. Available: https://docs.unity3d.com/ScriptReference/Transform-eulerAngles.html#:~:text=Euler%20angles%20can%20represent%20a,Y%20axis%2C%20in%20that%20order

[123] "Entendiendo la Aritmética de Vectores." Accessed: Sep. 02, 2023. [Online]. Available: https://docs.unity3d.com/es/530/Manual/UnderstandingVectorArithmetic.html

[124] "Maths – Conversion Matrix to Quaternion." Accessed: Aug. 13, 2023. [Online]. Available: https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/

[125] "Quaternion.LookRotation." Accessed: Sep. 02, 2023. [Online]. Available: https://docs.unity3d.com/ScriptReference/Quaternion.LookRotation.html

[126] "Vector3.Cross." Accessed: Sep. 02, 2023. [Online]. Available: https://docs.unity3d.com/es/530/ScriptReference/Vector3.Cross.html

[127] "Motive Optical motion capture software." Accessed: Sep. 19, 2023. [Online]. Available: https://optitrack.com/software/motive/

[128] "Calibration Squares." Accessed: Sep. 19, 2023. [Online]. Available: https://docs.optitrack.com/v/v2.3/motive/calibration/calibration-squares

[129] "UR3e robot", Jun. 15, 2023. [Online]. Available: https://www.universal-robots.com/es/productos/robot-ur3/