UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTACIÓN DE HERRAMIENTA PARA ANOTACIÓN Y VALIDACIÓN DE SIMETRÍAS EN OBJETOS 3D

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

VALERIA CONSTANZA NAHUELPÁN MUÑOZ

PROFESOR GUÍA:
IVÁN SIPIRAN MENDOZA

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
PABLO GONZÁLEZ JURE

SANTIAGO DE CHILE
2023

Resumen

La simetría es una cualidad ampliamente presente en nuestro mundo, se manifiesta en objetos vivos, como animales y plantas, así como en objetos fabricados por el ser humano. Esta cualidad simétrica ha dejado una huella indeleble en una variedad de disciplinas, desde el ámbito artístico hasta la ingeniería. Dada su importancia, resulta fundamental entender este concepto desde diversas perspectivas para abordar problemas de manera efectiva. Asimismo, la simetría es un concepto que simplifica la complejidad, lo cual lo hace especialmente atractivo, ya que nos permite representar una entidad con menos información.

El enfoque del presente trabajo de título es proveer las herramientas necesarias para llevar a cabo uno de los objetivos esenciales requeridos para completar el proyecto de investigación dirigido por el profesor Iván Sipiran, que busca responder cómo la información de simetría de una figura puede ayudar a comprender los resultados de un algoritmo automático en una tarea de reconstrucción de objetos 3D. Este objetivo consiste en la creación de un benchmark[1] para la evaluación de algoritmos de detección de simetrías en objetos tridimensionales. Para lograrlo, es crucial disponer de un conjunto de datos que cuente con la anotación detallada de las simetrías presentes en cada objeto, con el propósito de hacer comparaciones entre resultados obtenidos de los algoritmos y los esperados o etiquetas. Hasta el momento, la ausencia de dicho conjunto de datos ha dificultado la realización de evaluaciones estandarizadas de los métodos de detección.

Se propone la creación de una aplicación de escritorio con el propósito de obtener etiquetas de simetría en objetos tridimensionales. Esta aplicación deberá habilitar la realización de anotaciones de manera semi-automática a través de la inserción de puntos en la superficie de los objetos. Estos puntos, introducidos por los usuarios, darán lugar a la generación de planos reflectivos o ejes rotacionales. La totalidad de los datos relativos a las anotaciones se registrarán en un archivo JSON para facilitar su gestión y utilización posterior.

La herramienta fue desarrollada y sometida a pruebas por parte de usuarios, quienes indicaron que logra cumplir con el objetivo general. De igual manera, se señaló que existen oportunidades de mejora en lo que respecta a la interfaz de usuario.

[1]Prueba de rendimiento donde se comparan datos etiquetados o esperados con los resultados de los algoritmos a evaluar.

Dedicado a mi familia por acompañarme en este proceso

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto

La simetría desempeña un papel fundamental en computación debido a su capacidad para simplificar y optimizar los procesos de análisis y síntesis de información. Por ejemplo, en el caso de una imagen simétrica, no es necesario almacenar la totalidad de los datos, sino únicamente la parte no repetida. De esta manera, es posible reconstruir la entidad original utilizando el conocimiento de las simetrías. Lo mismo se aplica a un objeto en 3D. Sin embargo, las representaciones computacionales no brindan información sobre las posibles simetrías que pueden contener, por lo que se vuelve necesario analizarlas y buscar simetrías basadas en la información disponible. Desde la perspectiva computacional, esta tarea de detección de simetrías es desafiante y ha recibido mucha atención recientemente por parte de comunidades de diferentes áreas de investigación, como la visión artificial y el procesamiento geométrico.

Por otro lado, el problema de completar objetos tridimensionales ha ganado relevancia debido a la necesidad de procesar datos 3D incompletos o ruidosos de dispositivos como LIDAR o Kinect. En el proyecto de investigación dirigido por el profesor Iván Sipiran se busca responder cómo la información de simetría de una figura puede ayudar a comprender los resultados de un algoritmo que reconstruye objetos 3D. La principal hipótesis radica en que al desvelar datos sobre la simetría estructural de un objeto tridimensional, es posible emplear dicha información para sintetizar o generar la geometría faltante dadas las observaciones parciales. La propuesta de investigación consta de tres ejes principales:

1. La evaluación efectiva de algoritmos automáticos para detección de simetría.
2. La detección robusta de simetrías rígidas y regiones de soporte basadas en redes neuronales.
3. La adopción de un enfoque consciente de la simetría para completar objetos.

El objetivo general para llevar a cabo el proyecto de investigación es desarrollar nuevos algoritmos para la detección de simetría y reconstrucción de objetos basados en enfoques de aprendizaje. Para lograr este propósito, se han establecido tres objetivos específicos:

1. Construir un benchmark para la evaluación de algoritmos de detección de simetrías y evaluar métodos existentes.
2. Diseñar e implementar un método basado en redes neuronales que detecte simultáneamente las simetrías en objetos tridimensionales y la geometría que sostiene la simetría.
3. Diseñar e implementar un método basado en redes neuronales que complete una entrada con geometría faltante con la adopción de un enfoque consciente de la simetría.

1.2. Problema

El presente trabajo de título se encuentra netamente relacionado con el primer objetivo de la investigación señalada anteriormente, que se refiere a la búsqueda de una evaluación efectiva de los algoritmos automáticos utilizados para detectar simetrías mediante la creación de un benchmark. Hasta la fecha, no se ha llevado a cabo una evaluación estándar en este ámbito. Con el fin de abordar esta necesidad, es necesario desarrollar dicho benchmark, el cual servirá como punto de comparación para contrastar los resultados obtenidos por los algoritmos con los resultados esperados. Como consecuencia de esta premisa, emerge la necesidad de una herramienta de anotación que facilite la tarea de etiquetar las simetrías presentes en un conjunto de objetos potenciales, generando así los resultados esperados indispensables para la comparación.

En el Capítulo 3, se llevará a cabo una exploración más exhaustiva de la problemática en cuestión.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo principal de este trabajo de título es desarrollar una herramienta que permita a usuarios realizar la anotación o etiquetamiento de simetrías más fina posible en objetos tridimensionales de manera semiautomática.

1.3.2. Objetivos específicos

1. Diseñar e implementar un visualizador interactivo capaz de cargar y manipular objetos tridimensionales permitiendo su rotación con el cursor del mouse.

2. Incorporar la participación del usuario, en la cual se le solicitará que elija dos correspondencias simétricas en la superficie del objeto para simetrías reflectantes. Posteriormente, se procederá a la generación y representación gráfica de un plano en función de estas selecciones.

3. Integrar la participación del usuario, donde la aplicación de simetrías rotacionales podría requerir únicamente la selección de puntos clave a lo largo de la órbita del eje simétrico. A continuación, se procedería a generar y visualizar de manera automática el eje de simetría correspondiente.

4. Facilitar la visualización de un listado de cada simetría ingresada que permita diferenciarlas entre sí.

5. Optimizar la simetría proporcionada por el usuario a través de la integración de métodos de refinamiento.

6. Enriquecer las capacidades al introducir funcionalidades adicionales para la eliminación y preservación de ejes o planos.

7. Crear un archivo JSON destinado a almacenar la información generada a través de la herramienta.

1.4. Solución propuesta

En un enfoque general para alcanzar el objetivo planteado, se propone desarrollar una herramienta en forma de programa utilizando la librería OpenGL en el entorno de Python, para su implementación como una aplicación de escritorio. Este programa tendrá la capacidad de cargar y representar objetos tridimensionales en una ventana de la librería GLFW (acrónimo de "Graphics Library Framework"). Junto a esto, se incorporará una interfaz proporcionada por la biblioteca IMGUI (acrónimo de "Immediate Mode Graphical User Interface"), la cual contendrá elementos interactivos que permitirán al usuario añadir simetrías reflectivas y rotacionales a través de puntos de referencia.

Para posicionar estos puntos de referencia en el objeto, se empleará la técnica de trazado de rayos utilizando librerías disponibles en Python, lo que garantizará una ubicación precisa. Es relevante destacar que se establece la restricción de que un objeto podrá contar con múltiples simetrías reflectivas o planares, sin embargo, solo se permitirá una única simetría rotacional.

El proceso completo para abordar esta problemática se describirá en detalle en la sección dedicada a la solución en el Capítulo 4.

1.5. Resumen de resultados

Se logró completar el desarrollo de la herramienta, la cual cuenta con las funcionalidades básicas para llevar a cabo el etiquetamiento de simetrías en objetos tridimensionales. Además, se sometió a pruebas realizadas por terceros. Posteriormente, se aplicó una encuesta diseñada para evaluar la usabilidad de la herramienta, solicitando a los participantes sus comentarios y sugerencias para su mejora continua. Este proceso de retroalimentación proporcionó valiosas perspectivas que han contribuido significativamente a perfeccionar la herramienta y asegurar su eficacia en la tarea de etiquetar simetrías en objetos tridimensionales.

Capítulo 2

Estado del Arte

2.1. Aplicaciones de simetría en tres dimensiones

La simetría juega un papel de suma relevancia en el ámbito de la informática, especialmente en la concepción y modelado de objetos tridimensionales. Sin embargo, su utilidad trasciende ampliamente este dominio, ya que el concepto de simetría en objetos tridimensionales ha encontrado innumerables aplicaciones en diversas áreas, permitiendo abordar y resolver una amplia gama de problemas. A continuación, se exponen algunas de estas aplicaciones significativas:

- Diseño Asistido por Computadora (CAD) [20] [3] [11] [19]: Estas aplicaciones buscan reducir la complejidad del almacenamiento de piezas diseñadas por computadora.
- Modelamiento computacional [21]: Esta aplicación utiliza simetría facial 3D para mejorar las características de los rostros modelados en 3D.
- Arqueología [40] [39]: Estas aplicaciones utilizan simetrías para reconstruir o mejorar la representación de objetos del patrimonio cultural en 3D.
- Medicina [33] [47]: Estas aplicaciones buscan aprovechar las simetrías para segmentar la forma de las próstatas y así mejorar el diagnóstico de enfermedades.
- Química [38]: Esta aplicación demuestra que explotar las simetrías del proceso de diseño molecular 3D mejora la calidad de las moléculas generadas.
- Modelado en gráficos [17] [18]: Estas aplicaciones utilizan el concepto de simetría para imponer buenas propiedades de modelado en objetos 3D.
- Robótica [8]: En esta aplicación se utiliza reconstrucción de objetos 3D basada en simetría para localizar frutas mediante robots recolectores.

2.2. Herramienta de anotación de simetrías 2D

Dentro del área de detección de simetrías en imágenes, es importante destacar la investigación presentada en el artículo del año 2016 "Symmetry reCAPTCHA" [6]. Este estudio evidenció el bajo desempeño de los algoritmos de detección de simetría al ser aplicados a imágenes del mundo real, una evaluación que se remonta al año 2011 en el marco de la Conferencia de Visión por Computadora y Reconocimiento de Patrones (CVPR). Se realizó un riguroso análisis sistemático que reveló una diferencia significativa entre las simetrías etiquetadas por humanos (reflectivas y rotacionales) en fotografías y la salida de algoritmos de visión por computadora en el mismo conjunto de fotos. Se aprovechó esta brecha en la percepción de simetría entre humanos y máquinas para proponer una novedosa prueba de Turing basada en la simetría. Se diseño e implementó una interfaz gráfica para que evaluadores humanos ingresen su elección de simetrías percibidas en el mundo real en una imagen. La interfaz de usuario guía a los evaluadores para etiquetar una rotación o una simetría de reflexión. Una vez que un tipo de simetría es seleccionado, pueden identificar un centro de rotación con un clic o un eje de reflexión haciendo clic en dos puntos finales de un segmento de línea.

Gracias a dicha herramienta se recopilaron más de 78.000 etiquetas de simetría de 400 evaluadores de Amazon Mechanical Turk en 1.200 fotografías del conjunto de datos COCO de Microsoft. Utilizando este conjunto de simetrías de referencia generadas automáticamente a partir de las etiquetas humanas, la efectividad de este trabajo se evidencia en una prueba en la que se logra una tasa de éxito de más del 96 %. Al demostrar resultados estadísticamente significativos propusieron utilizar la percepción de simetría como un potente reCAPTCHA alternativo basado en imágenes, sugiriendo así una aplicación innovadora de la percepción humana de la simetría en el ámbito de la seguridad en línea y la autenticación.

Si bien lo mencionado anteriormente parece ser interesante y similar en nuestro contexto sobre detección de simetrías 3D, la herramienta desarrollada solo funciona para etiquetar imágenes. Actualmente, no existe un software especializado que permita llevar a cabo anotaciones sobre las simetrías de objetos tridimensionales. De igual manera, es fundamental destacar la presencia de aplicaciones diseñadas para la anotación de diversos aspectos de objetos en entornos tridimensionales. Un ejemplo significativo de esto es la capacidad de las herramientas para llevar a cabo la clasificación de objetos. Entre las opciones disponibles en esta categoría se incluyen CVAT, Amazon SageMaker Ground Truth, Diffgram, Voxel51, entre otras destacadas alternativas.

Obtener las anotaciones de simetrías resulta esencial para la evaluación de algoritmos de detección. En los métodos recientes basados en el aprendizaje, la evaluación no se ha realizado correctamente ya que existen inconvenientes tanto en la selección de objetos para la evaluación como en la construcción del conjunto de datos con objetos modelados y con alineación conocida.

2.3. Métodos de detección de simetrías 3D

Con el propósito de resaltar la importancia de la creación de un benchmark para evaluar los algoritmos de detección de simetrías 3D, se procederá a realizar una clasificación de estos métodos en base a los distintos enfoques empleados para afrontar este desafío. Esto adquiere relevancia debido a la diversidad de métodos disponibles en el campo y la necesidad de contar con una base de referencia sólida para evaluar su desempeño de manera objetiva. Los benchmarks desempeñan un papel fundamental al proporcionar un marco estandarizado que facilita la comparación y evaluación objetiva de la eficacia de los algoritmos, lo que contribuye al avance y desarrollo continuo de la investigación y desarrollo de métodos algorítmicos en múltiples disciplinas.

2.3.1. Métodos basados en transformación

Estos métodos se basan en aplicar transformaciones geométricas a los datos y analizar cómo cambian bajo estas transformaciones para determinar si existen simetrías.La idea es calcular una representación intermedia para un objeto 3D de modo que las simetrías sean más fáciles de encontrar. La Tabla 2.1 muestra algunos métodos basados en este enfoque.

Referencia	Nombre
Martinet et al. 2006 [25]	Generalized Moments
Grushko et al. 2012 [9]	Intrinsic Local Symmetries
Kakarala et al. 2013 [14]	Bilateral Symmetry in Phase Domain
Ovsjanikov et al. 2013 [31]	Quotient Spaces
Zhang et al. 2013 [48]	Symmetry Robust Descriptor
Jiang et al. 2014 [13]	Fourier-theoretic Approach
Wang et al. 2014 [43]	Global Intrinsic Symmetry Invariant Functions
Li et al. 2015 [16]	View-based Symmetry Detection
Liu et al. 2015 [24]	Orthonormal Functional Maps

Tabla 2.1: Métodos con el enfoque basado en la transformación.

2.3.2. Métodos basados en correspondencia

En este enfoque, la detección de simetría se basa en la determinación de correspondencias candidatas simétricas (puntos o regiones) en los objetos 3D. Estos métodos se basan en la idea de que si un objeto o una imagen es simétrico, entonces debe haber puntos o características en un lado del objeto o imagen que correspondan de manera cercana a puntos o características en el otro lado. La Tabla 2.2 muestra métodos relacionados a este enfoque.

Referencia	Nombre
Mitra y Bronstein 2010 [27]	Intrinsic Regularity Detection
Raviv et al. 2010 [35]	Full and Partial Symmetries in Non-rigid Shapes
Raviv et al. 2010 [34]	Diffusion Symmetries
Berner et al. 2011 [1]	Subspace Symmetries
Wang et al. 2011 [44]	Symmetry Hierarchy
Liu et al. 2012 [23]	Symmetry Axis Curves
Shehu et al. 2014 [36]	Characterization of Partial Intrinsic Symmetries
Tevs et al. 2014 [42]	Geometric Symmetries and Regularities
Yoshiyasu et al.2014 [46]	Symmetry-aware Non-rigid Matching

Tabla 2.2: Métodos con el enfoque basado en correspondencia.

2.3.3. Métodos basados en votaciones

Los métodos de este enfoque se basan en la acumulación de evidencia (votos) para las simetrías. En general estos métodos codifican transformaciones generadas a partir de puntos con características geométricas comunes. Después estas transformaciones deben verificarse para encontrar el conjunto final de simetrías. En la Tabla 2.3 se muestran algunos métodos de esta categoría.

Referencia	Nombre
Mitra et al. 2006 [28]	Partial and Approximate Symmetries
Pauly et al. 2008 [32]	Structural Regularity Detection
Lipman et al. 2010 [22]	Symmetry Factored Embedding
Zheng et al. 2010 [49]	Non-local Scan Consolidation
Xu et al. 2012 [45]	Multi-scale Partial Intrinsic Symmetry
Jiang et al. 2013 [12]	Skeleton-based Intrinsic Symmetry
Sipiran et al. 2014 [39]	Approximate Symmetry in Partial 3D Meshes

Tabla 2.3: Métodos con el enfoque basado en votos.

2.3.4. Métodos basados en optimización

En este enfoque, el problema de detección de simetría es un problema de optimización donde la simetría es la mejor transformación que contiene las restricciones de optimización. Estos métodos buscan ajustar parámetros o transformaciones para que los datos sean más simétricos o cumplan ciertas propiedades de simetría. En la Tabla 2.4 se muestran métodos basados en este enfoque.

Referencia	Nombre
Korman et al. 2015[15]	Probably Approximately Symmetries
Mavridis et al. 2015 [26]	K-sparse Optimization
Speciale et al. 2016 [41]	Convex Variational
Ecins et al. 2017 [5]	Symmetrical Fitting
Cicconet et al. 2017 [4]	Registration and Pairwise Alignment of Curves
Nagar et al. 2019 [29]	Symmetry by Manifold Optimization
Nagar et al. 2020 [30]	3DSymm: Reflection Symmetry Detection

Tabla 2.4: Métodos con el enfoque basado en optimización.

2.3.5. Métodos basados en el aprendizaje

En este enfoque, los métodos aprovechan el progreso reciente en enfoques de deep learning. En general, estos métodos se basan en el entrenamiento de una red neuronal que recibe una forma 3D y entrega una representación de simetría.

Referencia	Nombre
Bu et al. 2015 [2]	Local Deep Feature Learning
Ji et al. 2019 [10]	3D reflectional symmetry detector with neural network
Gao et al. 2020 [7]	PRS-Net
Shi et al. 2020 [37]	SymmetryNet

Tabla 2.5: Métodos con el enfoque basado en el aprendizaje.

Capítulo 3

Descripción del problema

El problema central que se busca abordar en esta memoria se relaciona con el proyecto de investigación dirigido por el profesor guía, cuyo objetivo consiste en desarrollar nuevos algoritmos para detección de simetrías en objetos tridimensionales y finalización de formas utilizando enfoques de aprendizaje que extraigan patrones estructurales relevantes. Actualmente, la evaluación de la precisión de los métodos de detección de simetría se ha realizado de forma no estándar. En ocasiones, los métodos sólo se prueban en un conjunto limitado de casos. En los métodos recientes basados en el aprendizaje, la evaluación no se ha realizado adecuadamente ya que tanto la selección de objetos para la evaluación como la construcción del conjunto de datos de objetos modelados con simetría conocida presentan desafíos al ser un proceso manual y no existir herramientas que lo faciliten.

Los métodos actuales que detectan simetrías se basan en el conocimiento previo de conjuntos de datos existentes. Por ejemplo, a menudo se asume que las formas están correctamente alineadas con respecto a un cierto eje en el espacio tridimensional. Esta suposición facilita la rápida adopción de estos conjuntos de datos para evaluar algoritmos de detección de simetría. Sin embargo, hay algunas desventajas al dar por sentadas estas suposiciones. En primer lugar, se sabe que los conjuntos de datos a gran escala no siempre mantienen esta alineación. Por lo tanto, algunos modelos pueden estar desalineados, lo que causa problemas en los métodos de aprendizaje al introducir sesgos durante el entrenamiento. En segundo lugar, se asume que la simetría es global y no se conocen las simetrías locales. Esto hace difícil saber si un algoritmo de detección puede manejar simetrías parciales. En tercer lugar, los conjuntos de datos populares provienen principalmente de objetos generados por software de modelado, lo que significa que no reflejan las imperfecciones presentes en formas obtenidas de aplicaciones reales mediante procesos de reconstrucción. En este proyecto, se busca superar estas desventajas y proporcionar un conjunto de datos que evalúe de manera efectiva la capacidad de los métodos de detección de simetría en simetrías globales y simetrías parciales. También se quiere evaluar la resistencia de los métodos frente a imperfecciones geométricas, que son comunes en objetos escaneados o reconstruidos.

Con el propósito de cumplir con el objetivo fundamental de la investigación señalada, que consiste en desarrollar algoritmos y evaluarlos de manera precisa, es imperativo contar con un conjunto de datos debidamente etiquetado que proporcione información detallada acerca

de las simetrías en objetos tridimensionales, ya sean de tipo reflectivas o rotacionales. Este paso crítico sienta las bases para la creación de un benchmark, el cual desempeñará un papel esencial en la evaluación estandarizada de los algoritmos. En primer lugar, permitirá comparar la precisión entre métodos. En segundo lugar, evaluará la solidez de los métodos frente a varias condiciones, como simetrías parciales, imperfecciones de la superficie y datos faltantes. Además se evaluará proponer el benchmark como parte de las competencias SHREC[1](Shape Retrieval Contest) que goza de un reconocimiento amplio en la comunidad de análisis de objetos. La participación en estas competencias proporciona una oportunidad estratégica para promover la adopción del benchmark en investigaciones futuras, consolidando su impacto en el ámbito de la detección de simetría en objetos tridimensionales.

Para abordar esta problemática, surge la idea de desarrollar una herramienta de anotación que facilite la generación del conjunto de objetos 3D con sus simetrías reflectivas y rotacionales debidamente etiquetadas. Esta herramienta consiste en una aplicación de escritorio la cual permite que usuarios etiqueten los objetos 3D, es decir que a los objetos ingresados le añadan simetrias relfectivas y rotacionales que corresponden a planos y ejes respectivamente. El resultado de la utilización de esta herramienta se traduciría en un archivo JSON que contendrá toda la información necesaria para el análisis y evaluación subsiguiente.

[1]https://www.shrec.net/

Capítulo 4

Solución

En virtud de la problemática previamente planteada en relación a la ausencia de un conjunto de objetos debidamente etiquetados, se ha concebido una solución consistente en el desarrollo de una herramienta semiautomática, la cual habilita la posibilidad de efectuar anotaciones específicas sobre objetos tridimensionales. Con el propósito de otorgar una comprensión exhaustiva del proyecto en cuestión, las secciones subsiguientes se enfocarán en un análisis detallado que abarcará el diseño, funcionamiento y proceso de desarrollo de dicha herramienta.

4.1. Diseño de interfaz

La herramienta de anotaciones cuenta con una interfaz que se compone de una ventana principal. En la parte superior de dicha ventana, se encuentra una barra de menú que permite cargar objetos mediante archivos con extensión .off. Una vez cargado el objeto, este se renderiza y se visualiza en la ventana de forma apropiada.

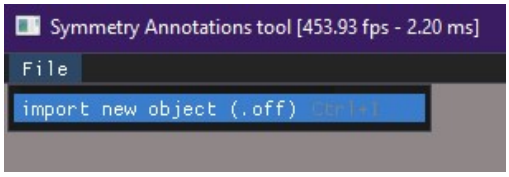

Figura 4.1: Importar nuevo objeto

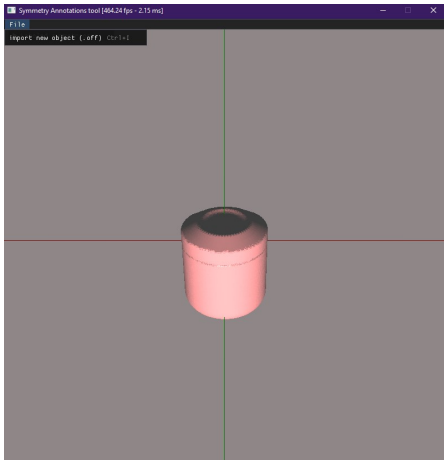

Figura 4.2: Objeto cargado

Para facilitar la manipulación del objeto, se puede utilizar el mouse para rotarlo según se desee. Además, en el costado izquierdo de la ventana, se sitúa una pequeña sección que consta de dos listados distintos: simetrías reflectivas, las cuales corresponden a planos de simetría, y una simetría rotacional única, que corresponde a un eje de simetría y es aplicado a solidos de revolución[1]. En cada sección, se encuentra un botón específico que permite añadir la simetría correspondiente, ya sea reflectiva o rotacional como se muestra en la Figura 4.3.

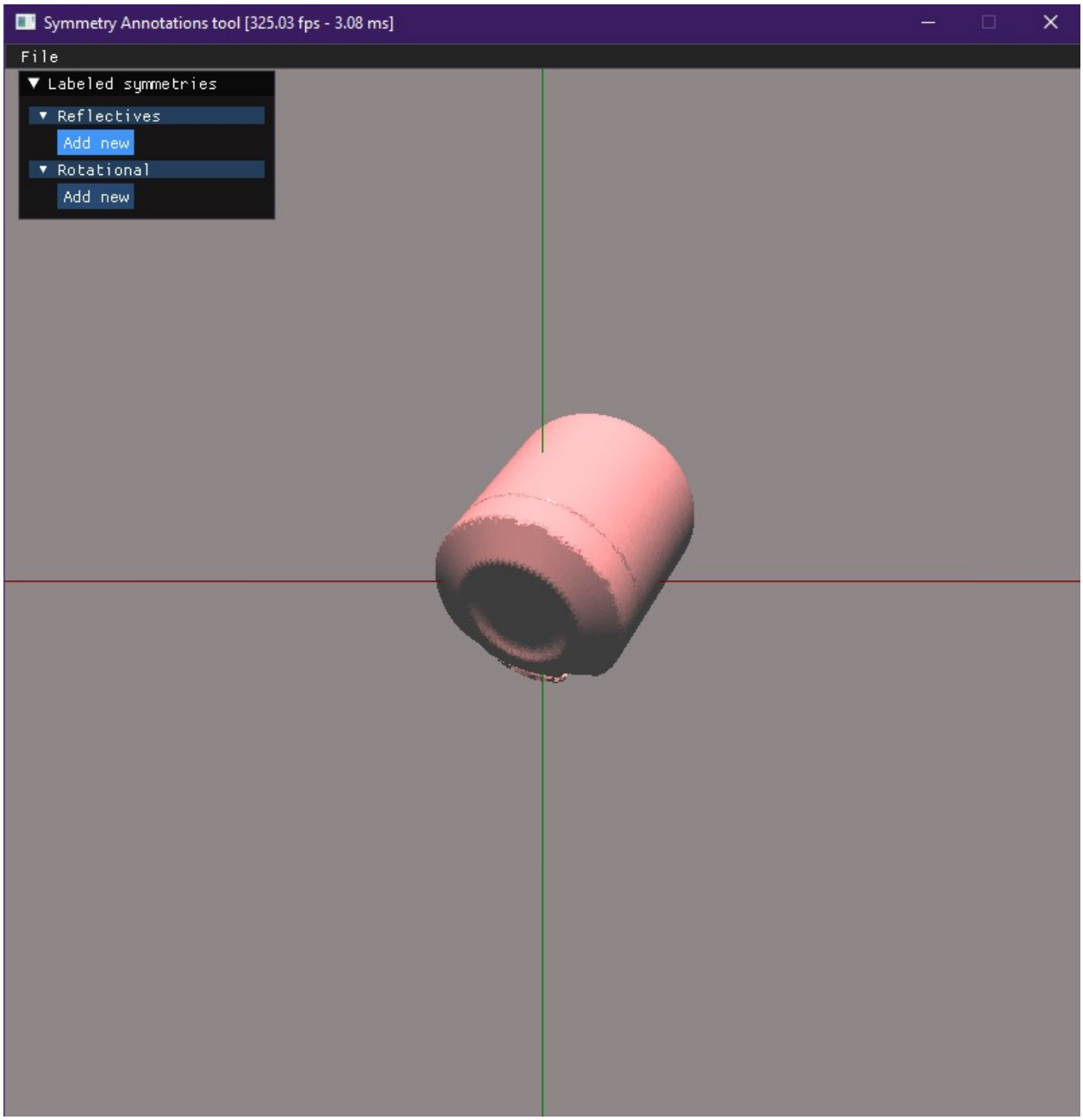

Figura 4.3: Secciones y botones para añadir distintos tipos de simetrías

[1]Objeto tridimensional que se obtiene al girar una figura bidimensional alrededor de un eje determinado en el espacio.

Al presionar el botón “Add new”de la sección de simetrías reflectivas, el usuario debe emplear el clic derecho para situar dos puntos sobre el objeto. Una vez estos dos puntos hayan sido establecidos, se generará y se dibujará en pantalla de manera automática un plano el cual es perpendicular al vector que conecta ambos puntos. Este plano estará ubicado en el punto medio de este vector. Al llevar a cabo este proceso de forma reiterada, surgirá un nuevo plano que se distingue mediante un color diferente, además de contar con un identificador único. Esto se puede apreciar en la Figura 4.4. La estrategia de asignar colores diferentes con un grado de transparencia tiene como finalidad mejorar la claridad en la visualización tanto de los planos generados como del objeto en la pantalla.

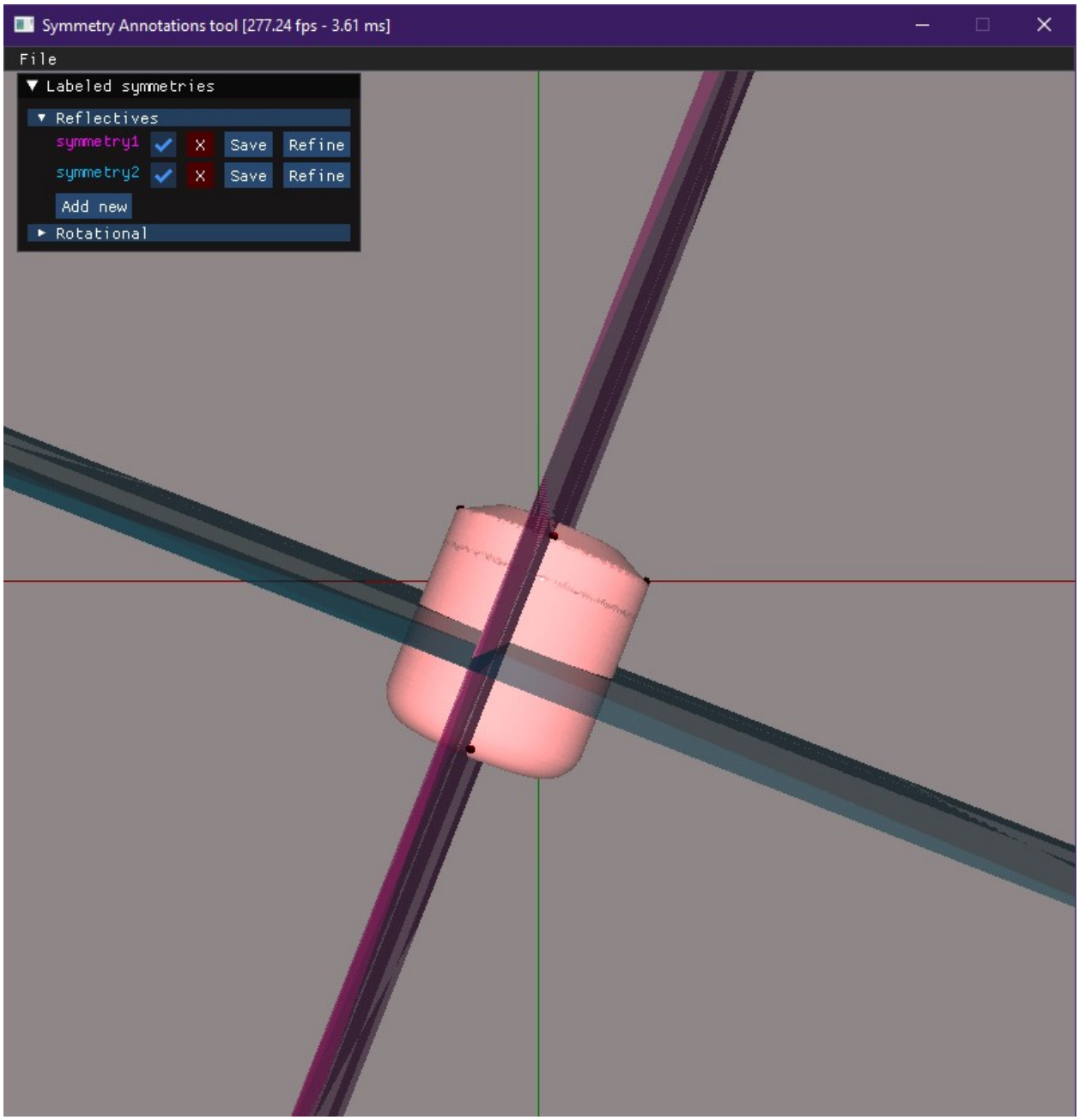

Figura 4.4: Simetrías reflectivas

Por otro lado, al hacer clic en el botón “Add new” de la sección de simetrías rotacionales, el usuario deberá colocar como mínimo tres puntos sobre el borde superior del objeto. Se recomienda añadir la mayor cantidad de puntos posible, ya que esto mejorará la precisión en el cálculo del eje rotacional. Una vez que el usuario considere haber colocado los puntos necesarios, deberá presionar el botón “Draw rotation axis” y el eje de rotación se mostrará en el objeto, proporcionando una referencia visual para la simetría rotacional aplicada.

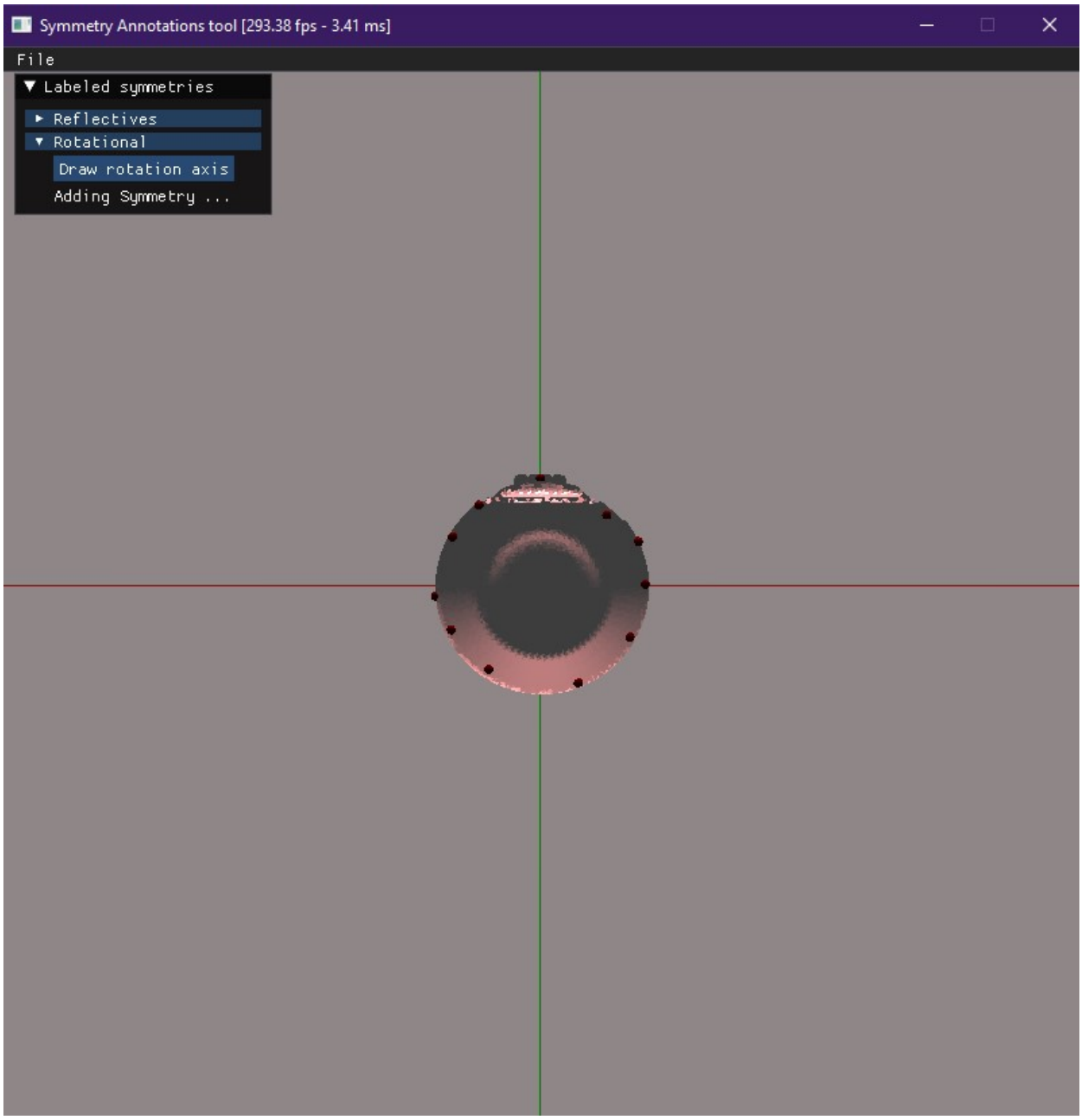

Figura 4.5: Puntos para añadir simetría rotacional

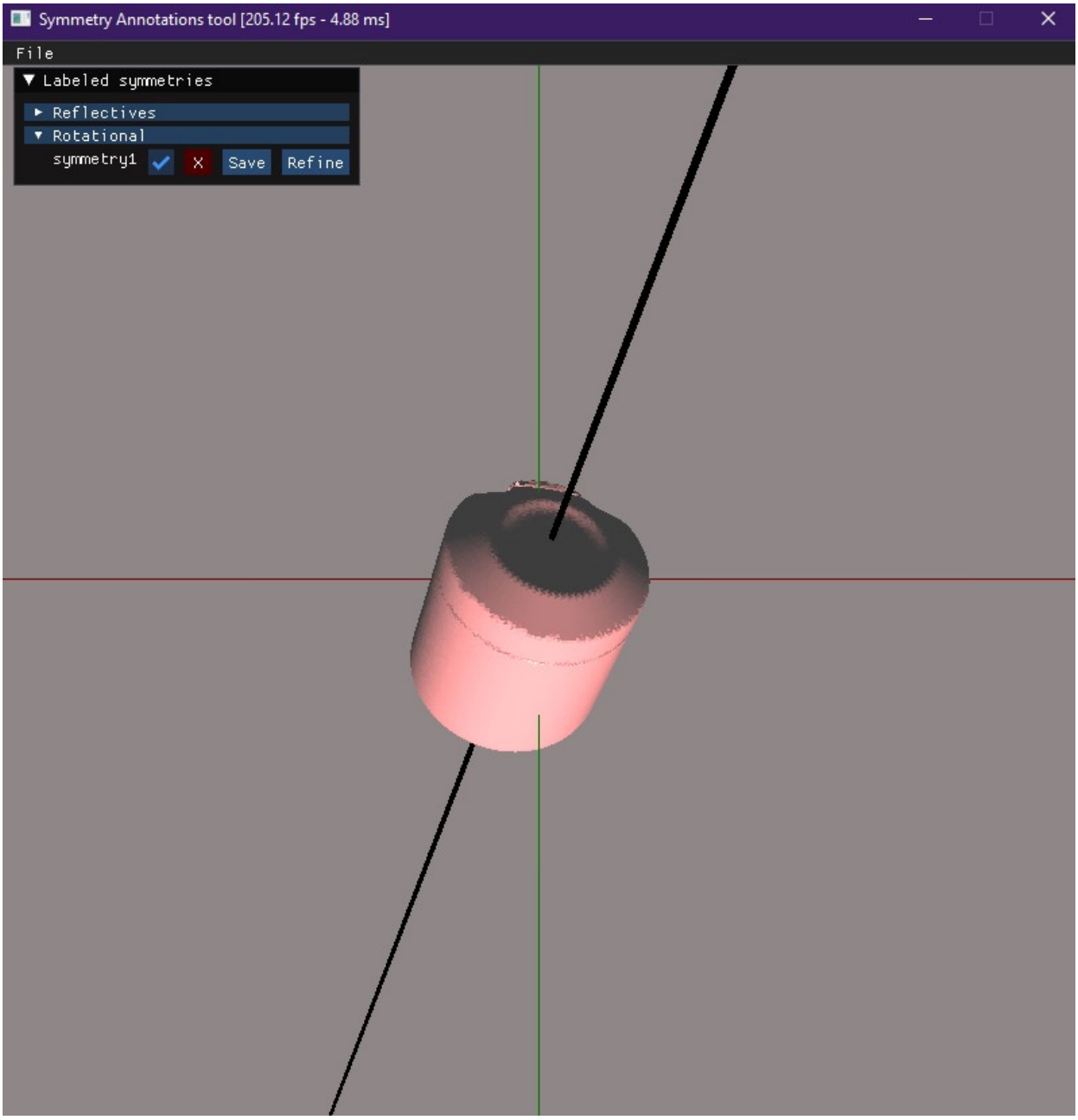

Figura 4.6: Simetría rotacional

Como se aprecia en las imágenes anteriores, al finalizar el proceso de etiquetar una nueva simetría (un plano o un eje), se despliegan opciones adicionales junto a su respectiva identificación que amplían la funcionalidad. Estas opciones brindan la posibilidad de mostrar u ocultar cada simetría, refinarlas, confirmar su permanencia o eliminar. Es esencial enfatizar que al utilizar la función de guardar, las simetrías etiquetadas se registran en un archivo JSON, representándose a través de un punto específico junto con un vector normal. Esta información es fundamental para la creación del benchmark que permitirá evaluar el rendimiento de los algoritmos de detección de simetría. A continuación, la Figura 4.7 ofrece un ejemplo concreto de la estructura de un archivo JSON donde se ilustra el etiquetamiento de simetrías en dos objetos diferentes:

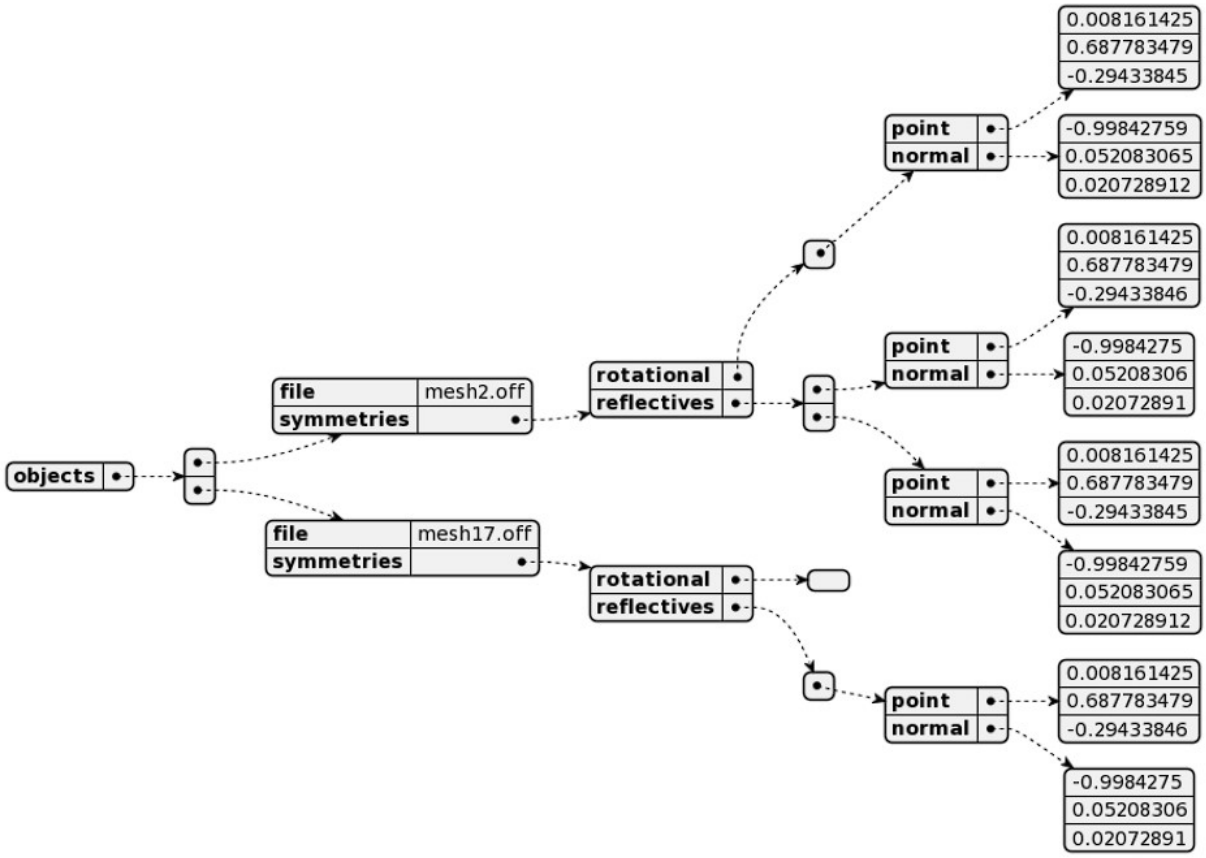

Figura 4.7: Ejemplo estructura de archivo JSON

En caso de que el usuario vuelva a abrir un objeto al cual se le haya añadido simetrías y estas se encuentren guardadas en el archivo JSON, toda la información de simetrías se carga nuevamente y se muestra en la interfaz de la herramienta. Esto asegura que al abrir nuevamente un objeto anotado, todas las simetrías previamente aplicadas sean recuperadas y visibles para su posterior edición o análisis. De esta manera, se garantiza la preservación y visualización de las anotaciones de simetría realizadas en el objeto.

En lo que respecta al proceso de refinamiento, una vez que el usuario ha seleccionado la opción "Refine", se procede a mejorar la precisión del plano o eje correspondiente. Posteriormente, el usuario tiene la capacidad de elegir entre preservar la simetría original o conservar la versión refinada a través del botón de guardado. Al efectuar la operación de guardar, una de las dos opciones se retiene mientras que la otra se elimina de manera automática, en consonancia con la preferencia del usuario. En la Figura 4.8 se puede observar un ejemplo de refinamiento de una simetría rotacional, donde el eje negro corresponde a la simetría original y el rosado es el refinado. Ambas se pueden observar en el listado junto a las funcionalidades de mostrar, ocultar, eliminar o guardar.

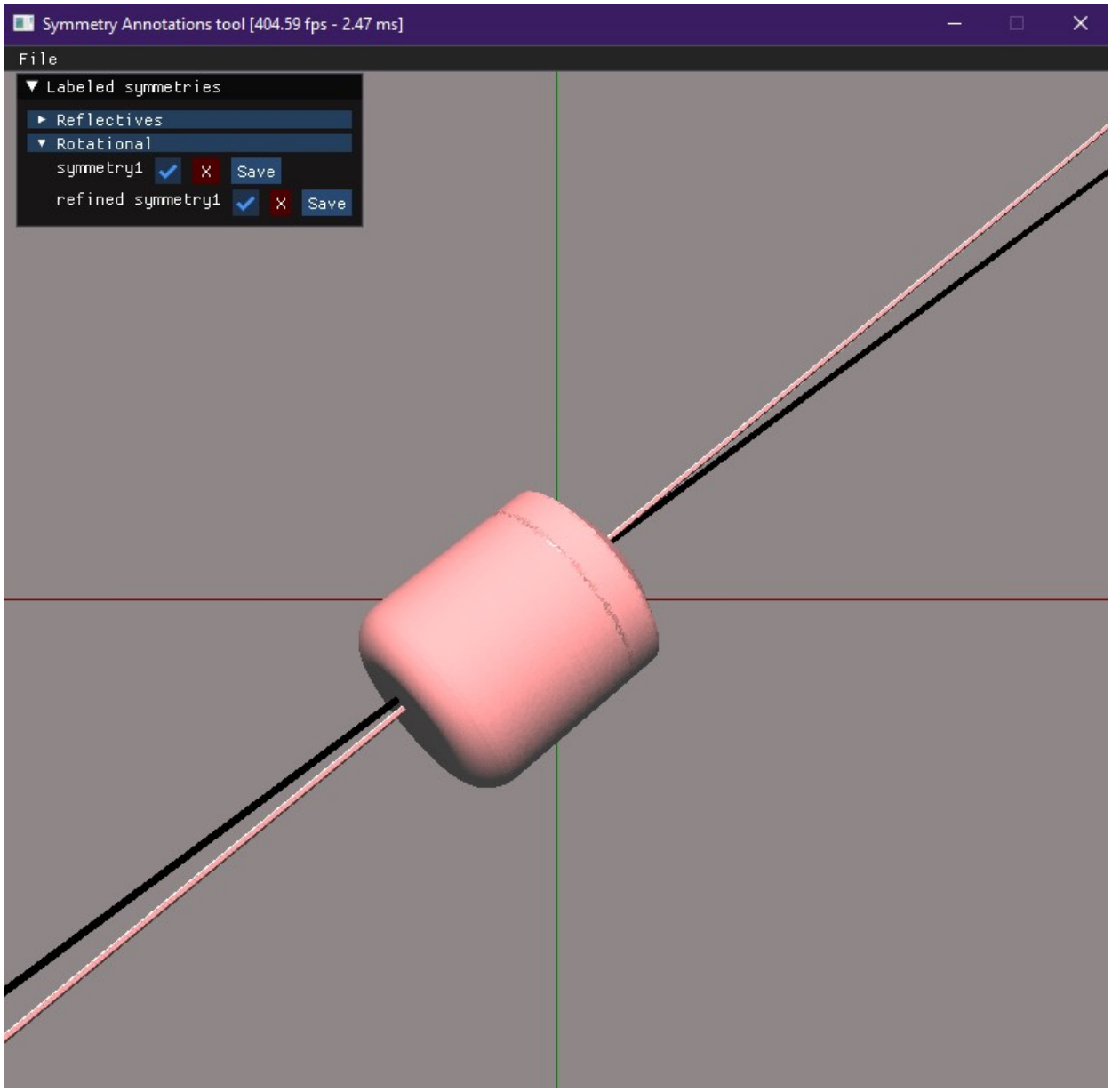

Figura 4.8: Refinamiento de simetría rotacional.

En el contexto de una simetría de reflexión, se observa una situación análoga, aunque con la distinción de que emerge un nuevo plano tanto en la representación de la escena tridimensional como en la enumeración de las simetrías en el listado.

4.2. Desarrollo y tecnologías utilizadas

4.2.1. Interfaz de usuario

Durante el desarrollo de la aplicación se ha utilizado Python como lenguaje principal y PyOpenGL como biblioteca para aprovechar las capacidades de la API gráfica OpenGL. Esta elección se fundamenta en la necesidad de poder visualizar en pantalla el objeto tridimensional cargado en la herramienta, el cual posee una extensión de archivo ".off".

En el proceso de creación y gestión de la ventana principal de la herramienta, se ha hecho uso de la biblioteca GLFW. La elección de esta biblioteca se ha basado en diversos aspectos que aportan significativas ventajas al desarrollo del proyecto. En primer lugar, se destaca su notable facilidad de uso, lo cual simplifica la implementación de la ventana principal y reduce la complejidad del código. Asimismo, se resalta su amplia compatibilidad con múltiples plataformas, lo que garantiza que la aplicación pueda ser ejecutada de manera consistente en distintos sistemas operativos. Además, ofrece funcionalidades adicionales que resultan sumamente útiles en este contexto. Entre estas características se incluye el manejo eficiente de las entradas de teclado y mouse mediante eventos, lo cual permite una interacción fluida y precisa con la herramienta. Es gracias a esta capacidad que se ha logrado implementar el movimiento del objeto con el mouse y agregar puntos sobre este mediante clics.

La ventana que aloja el listado de simetrías y los botones correspondientes a las acciones de agregar, eliminar, guardar y abrir un archivo se ha llevado a cabo haciendo uso de la biblioteca ImGui (Immediate mode GUI). Es una biblioteca altamente integrable y destaca por su naturaleza ligera y eficiente, permitiendo una rápida incorporación a la estructura del proyecto, permitiendo crear una interfaz simple y clara.

4.2.2. Implementación de puntos sobre el objeto

La implementación del proceso de colocación de puntos sobre el objeto se llevó a cabo utilizando la biblioteca Trimesh, esta librería ofrece una amplia gama de herramientas y funcionalidades para trabajar con mallas triangulares. Entre las operaciones que se pueden realizar se encuentran la carga y el guardado de mallas en diversos formatos, así como operaciones geométricas como traslaciones y rotaciones. La biblioteca trimesh también admite trazado de rayos en objetos 3D. Se pueden lanzar rayos desde un origen en una dirección específica y verificar si esos rayos intersectan con la malla tridimensional.

Durante el desarrollo de la aplicación, se procedió a crear una réplica del objeto visualizado en la ventana utilizando una malla de Trimesh, manteniendo las mismas transformaciones correspondientes al objeto original. A continuación, se llevó a cabo el trazado de rayos mediante el empleo del método *intersects_location(ray_origins, ray_directions)* provisto por Trimesh. En dicho proceso, se especificó un punto de origen que coincidía con la posición de la cámara, mientras que la dirección del rayo fue calculada como el vector que va desde la posición de la cámara hacia el punto en el espacio 3D correspondiente a la posición del cursor. Trimesh efectuó los cálculos necesarios para determinar si se producían intersecciones entre los rayos

trazados y los triángulos que conformaban la malla. En caso de detectarse una intersección, la función devolvió información detallada sobre dicho evento, incluyendo la posición de la intersección y el triángulo correspondiente con el cual se produjo la intersección. Con base en la posición de la intersección obtenida, se procedió a la creación y representación de una pequeña esfera en esa ubicación.

Este enfoque permitió lograr una interacción intuitiva y precisa, donde el usuario puede colocar puntos de manera directa y visualmente coherente sobre el objeto tridimensional. La utilización de Trimesh como biblioteca facilitó las operaciones geométricas necesarias y garantizó un procesamiento eficiente de la malla triangular del objeto. En consecuencia, se logró implementar exitosamente la funcionalidad de colocación de puntos sobre el objeto con precisión y fidelidad visual.

4.2.3. Visualización de simetrías

Reflectiva

Con el fin de incorporar una simetría reflectiva, se realiza el cálculo del vector que conecta los dos puntos inicialmente posicionados por el usuario sobre el objeto, lo cual permite generar y representar un plano perpendicular a dicho vector y que pasa por su punto medio. Con este propósito, se ha diseñado una clase denominada *ReflectiveSymmetry*. Esta clase se enfoca en crear y preservar simetrías reflectivas para cada objeto presente en el sistema.

El constructor de la clase, que no requiere argumentos, inicializa dos atributos importantes: *pointsPairs* e *info*. *pointsPairs* es una lista que almacena en tuplas los pares de puntos ingresados para cada plano de simetría. Por otro lado, *info* es una lista de diccionarios que contiene información relevante sobre cada simetría, incluyendo el punto y el vector normal asociados.

La clase *ReflectiveSymmetry* también proporciona dos métodos. El primero se llama *create_point(position)*, el cual recibe la posición de un punto y genera los puntos necesarios para la simetría reflectiva. Estos puntos son almacenados en la lista de tuplas *pointsPairs*. El segundo método, denominado *save_symmetry(point, normal)*, se utiliza para guardar la información de cada simetría, es decir, el punto y el vector normal.

Rotacional

Para el cálculo del eje de simetría en objetos con simetría rotacional se requiere ubicar puntos en la órbita rotacional del objeto. Un mínimo de tres puntos es necesario para este cálculo. Supongamos el escenario más simple, donde se disponen de tres puntos. Estos puntos son utilizados para formar un plano, y el eje de simetría se representa mediante el vector normal que se encuentra en el punto medio de estos tres puntos.

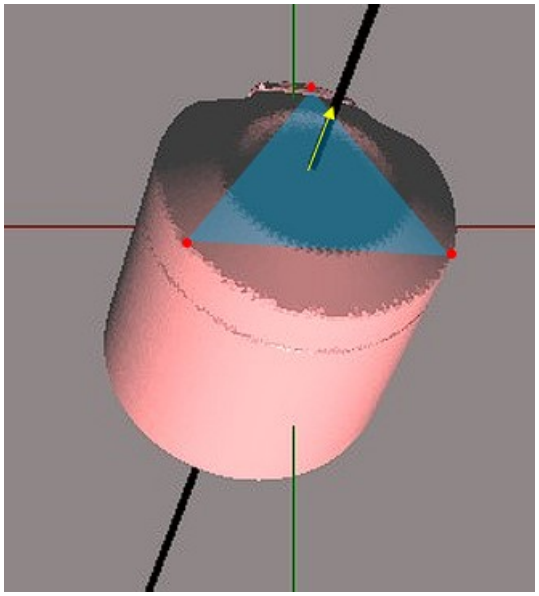

Figura 4.9: Escenario trivial para el cálculo del eje. Rojo: Puntos, Azul: Plano, Amarillo: Vector normal

Si se ingresan más de tres puntos, se generan todas las combinaciones de conjuntos de tres a partir de la lista contiene la totalidad de estos. Se hace un proceso iterativo donde cada una de estas combinaciones se utiliza para formar un plano, y luego se calcula el error de los demás puntos con respecto a ese plano específico. El error se define como la suma de las distancias de todos los demás puntos al plano en consideración.

Una vez determinado el plano con el menor error, se obtiene el eje de simetría representado por la normal de dicho plano que se ubica en el punto medio de todos los puntos agregados inicialmente.

Para llevar a cabo todo el proceso descrito anteriormente, se ha implementado una clase denominada *RotationalSymmetry*. Esta clase cuenta con los siguientes atributos: *points*, que es una lista que almacena los puntos ingresados, y al igual que *ReflectiveSymmetry*, tiene un atributo llamado *info* que almacena la información de la simetría generada, es decir, un punto y una normal que representan el eje de rotación.

La clase *RotationalSymmetry* también proporciona dos métodos. El primero es *create_point(position)*, el cual agrega un punto dado una determinada posición del objeto. Este método permite incorporar nuevos puntos al conjunto de puntos existentes. El segundo método es *create_plane()*, el cual realiza todo el proceso iterativo antes mencionado, para encontrar el mejor plano que posteriormente genera el eje de simetría correspondiente para finalmente dibujarlo en la escena.

4.2.4. Refinamiento de simetrías

Se integraron dos métodos fundamentales para el refinamiento de las simetrías previamente ingresadas por el usuario. Estos valiosos enfoques fueron proporcionados y orientados por

el profesor guía. Los métodos en cuestión representan herramientas esenciales que permiten mejorar la precisión y calidad de las simetrías definidas inicialmente.

- refineRotationTransform(point, normal, points): Esta función recibe el punto y la normal que definen una simetría rotacional y el conjunto de puntos del objeto, o nube de puntos que se refiere a la representación digital de un objeto tridimensional en el espacio. Comienza calculando la distancia de Chamfer entre la nube de puntos original y la nube de puntos transformada por la rotación. Luego, realiza iteraciones para encontrar un ángulo de rotación que minimice esta distancia. Retorna una nueva normal y un nuevo centro como simetría refinada.
- refineReflectionTransform(point, normal, points): Esta función recibe el punto y la normal que definen una simetría rotacional y el conjunto de puntos del objeto 3D. Comienza calculando la distancia de Chamfer entre la nube de puntos original y la nube de puntos transformada por la simetría. Luego, realiza iteraciones para encontrar una normal de reflexión que minimice esta distancia. Retorna una nueva normal como simetría refinada.

La fórmula de la distancia de Chamfer entre X e Y es la siguiente:

$$\text{Chamfer}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 + \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2 \tag{4.1}$$

Para calcular la distancia de Chamfer entre dos nubes de puntos 3D, representadas como matrices NumPy se utilizan las librerías scikit-learn (NearestNeighbors) y cálculos matriciales. El algoritmo funciona calculando la distancia mínima promedio entre los puntos de una nube y los puntos más cercanos de la otra nube. Puede calcular esta distancia en ambas direcciones ("Y a X" o "X a Y") o en ambas direcciones bidireccionalmente. La distancia de Chamfer se utiliza para evaluar cuán similares son dos nubes de puntos en términos de su distribución espacial, siendo menor el valor cuando las nubes son más similares.

Por otro lado, debido a la duración necesaria para completar las funciones de refinamiento, surgieron inconvenientes con la interfaz. Inicialmente, al activar el botón de refinamiento, se desencadenaba la ejecución de la función correspondiente, lo que, lamentablemente, resultaba en limitaciones significativas para agregar simetrías adicionales o interactuar con la herramienta. Este problema se debía a que la ventana de la interfaz quedaba inmovilizada y no respondía a las acciones del usuario durante la ejecución de las operaciones de refinamiento. En términos técnicos, esto sucedía dado que todas las operaciones se ejecutaban en una sola secuencia, lo que resultaba en una utilización subóptima de los recursos disponibles.

Para abordar este desafío y garantizar una experiencia de usuario fluida e ininterrumpida, se optó por implementar hilos de ejecución utilizando el módulo de threading en Python. Este módulo es esencial en programación concurrente y permite crear múltiples hilos de ejecución dentro de un solo proceso. Cada hilo opera de manera independiente y puede ejecutar tareas en paralelo, lo que aprovecha al máximo los núcleos de la CPU y los recursos disponibles. El objetivo principal de esta implementación es permitir que la interfaz siga

siendo altamente receptiva mientras los procesos de refinamiento se llevan a cabo en segundo plano de manera concurrente. Esto significa que los usuarios pueden continuar interactuando con la herramienta, agregar simetrías adicionales o realizar otras acciones como presionar botones del listado sin verse afectados por las operaciones de refinamiento en curso.

4.3. Potenciales Mejoras a Futuro

En virtud de limitaciones de tiempo, algunas características quedaron sin implementarse. Además, ciertos aspectos con oportunidades de mejora no fueron abordados en su totalidad. Aunque estas características y mejoras no resultaban esenciales desde la perspectiva funcional, su inclusión habría aportado valor a la experiencia general del usuario. La decisión de omitirlas se fundamentó en la necesidad de priorizar la finalización puntual del proyecto dentro de los plazos establecidos. Sin embargo, se reconoce plenamente la posibilidad y el valor de incorporarlas en iteraciones futuras. Este enfoque estaría alineado con el compromiso continuo de mejorar y refinar la aplicación en base a la retroalimentación y la evolución de las necesidades del usuario, garantizando una experiencia cada vez más satisfactoria y completa.

En primer lugar, dado que los objetos seleccionados en formato OFF presentan variaciones en cuanto a sus dimensiones, se observa una disparidad entre aquellos que presentan un tamaño considerablemente mayor en comparación con otros de proporciones notoriamente menores. Con el objetivo de abordar esta disparidad y mejorar la experiencia visual, se considera pertinente implementar una funcionalidad en el visualizador. Esta nueva característica facultaría al usuario la capacidad de acercar o alejar los objetos mediante la simple interacción con la rueda de desplazamiento del ratón. Esta funcionalidad se llevaría a cabo mediante la adaptación de la posición de la cámara en el espacio tridimensional, otorgando así una mayor flexibilidad y habilidad para ajustar la visualización de los objetos en diversas escalas según la preferencia del usuario. Este enfoque, además de mitigar las discrepancias dimensionales, proporcionaría una herramienta poderosa para explorar los objetos desde perspectivas más detalladas o generales, enriqueciendo así la experiencia visual y la comprensión de las características intrínsecas de cada objeto en el entorno tridimensional.

En segundo lugar, se identifica una oportunidad importante relacionada con la gestión de la iluminación en el entorno tridimensional. Actualmente, se ha observado que la fuente de luz de un objeto permanece fija en un punto específico cuando se rota, lo que puede resultar en una pérdida de realismo y en la generación de sombras no deseadas. Para abordar este desafío, se plantea la necesidad de una investigación más profunda sobre las capacidades de OpenGL y las técnicas de manipulación de luces en la aplicación. La mejora potencial consistiría en mantener el foco de luz en el lugar que se está visualizando en la cámara, de manera que, al rotar el objeto, la iluminación se ajuste de manera coherente con la perspectiva, evitando así la aparición de sombras no naturales. Esta optimización contribuiría significativamente a la calidad visual y realismo de la experiencia del usuario al interactuar con objetos en el entorno tridimensional. Esta área de desarrollo representa un desafío técnico valioso.

Capítulo 5

Validación

Luego de realizar varias iteraciones de pruebas por parte de la estudiante, se determinó el conjunto de objetos candidatos que serían sometidos a prueba por otros usuarios. Se procedió a compartir la implementación de la herramienta alojada en un repositorio de GitHub[1], acompañado por un README[2] exhaustivo. Este archivo README incluye el contexto de la herramienta, detalladas instrucciones que abarcan desde la instalación de las dependencias necesarias hasta la guía completa de uso, cubriendo aspectos como la introducción de simetrías, su almacenamiento, eliminación y refinamiento. Para una comprensión más efectiva, se enriqueció el documento con GIFs ilustrativos que demuestran el funcionamiento de la herramienta.

Adicionalmente, se llevó a cabo una sesión de explicación a fondo del contexto del proyecto, conceptos claves importantes y las instrucciones presentes en el README para aquellas personas que se ofrecieron voluntariamente para realizar la validación de la aplicación. La diversidad de los usuarios participantes fue un aspecto clave de la estrategia de validación, incluyendo personas con diversos niveles de experiencia en el campo de la simetría, así como usuarios de diferentes antecedentes profesionales. Esto aseguró una evaluación integral de la aplicabilidad y usabilidad de la herramienta en diferentes contextos y para un público medianamente diverso, con un total de ocho participantes.

Simultáneamente, se realizó una encuesta dirigida a los usuarios. Esta encuesta se llevó a cabo empleando la conocida "Escala de Usabilidad del Sistema" (SUS, por sus siglas en inglés) mediante un formulario alojado en la plataforma de Google. Los resultados de esta encuesta proporcionaron valiosas métricas cuantitativas relacionadas con la usabilidad de la herramienta. Se evaluaron aspectos como la facilidad de aprendizaje y la satisfacción del usuario. Además de las puntuaciones, se incorporó un espacio dedicado a la recopilación de comentarios y sugerencias por parte de los usuarios. Estos comentarios y sugerencias fueron un componente esencial para enriquecer aún más la retroalimentación obtenida durante el proceso de validación, lo que contribuyó significativamente a la mejora continua de la aplicación.

[1] https://github.com/ValeriaNahuelpan/SymmetryAnnotationTool

[2] Archivo que proporciona información importante y relevante sobre el contenido o el funcionamiento del proyecto o directorio en el que se encuentra.

El resultado obtenido en la encuesta SUS arrojó un porcentaje de satisfacción del usuario del 78.5 % y es una métrica significativa que puede interpretarse desde diversas perspectivas. En primer lugar, este resultado indica que la mayoría de los usuarios que participaron en la validación perciben la aplicación como razonablemente usable y fácil de aprender. Por otro lado, es esencial reconocer que el 21.5 % restante representa áreas donde los usuarios pueden experimentar desafíos o insatisfacciones. Estas áreas de mejora identificadas en la evaluación proporcionan una base valiosa para futuras iteraciones de desarrollo. Además, el resultado también sugiere que, aunque la herramienta ha tenido un buen desempeño, existe un espacio potencial para elevar aún más la satisfacción del usuario y la eficacia de la aplicación mediante ajustes basados en los comentarios recopilados.

En relación a los comentarios recibidos por parte de los usuarios, estos fueron mayoritariamente favorables y aportaron valiosas sugerencias constructivas. En primer lugar, se destacó la consideración del trabajo como innovador, ya que los usuarios no habían encontrado previamente una herramienta de este tipo.

Por otro lado, en cuanto a las sugerencias constructivas, se hizo hincapié en la importancia de incorporar la función de zoom en el objeto, una mejora que ya estaba contemplada en las futuras implementaciones y que se consideraría altamente beneficiosa si se lleva a cabo. Asimismo, hubo observaciones relacionadas con el comportamiento del objeto dentro de la aplicación, con varios usuarios sugiriendo que se podría mejorar significativamente si en lugar de rotar el objeto en la escena, se optara por mover la cámara. También, se planteó una sugerencia adicional con respecto a los planos, se propuso que podrían ser reducidos en tamaño para permitir una visualización más óptima del objeto. Además, se señaló la conveniencia de eliminar los puntos después de aplicar una simetría reflectiva, siguiendo el mismo procedimiento que se utiliza con la simetría rotacional. Esta modificación se considera beneficiosa, ya que la retención de puntos tras una simetría reflectiva podría resultar en confusión al intentar añadir otro plano. Finalmente, se recibió una recomendación importante relacionada con la visibilidad del etiquetamiento, sugiriendo la posibilidad de permitir la exportación directa del archivo JSON desde la aplicación.

También se presentaron otras sugerencias vinculadas al archivo README. Entre ellas, se destacó la necesidad de especificar de manera más clara los nombres de los objetos que pueden ser considerados sólidos de revolución. Asimismo, se sugirió proporcionar explicaciones más detalladas sobre las funcionalidades de cada botón en el listado de simetrías.

A pesar de estas valiosas sugerencias de mejora, en general, los usuarios coincidieron en que esta herramienta representa un sólido y prometedor comienzo en su desarrollo.

Capítulo 6

Conclusión

6.1. Recapitulación

A lo largo de este proyecto, se ha emprendido un viaje significativo en el dominio de la simetría, particularmente en el contexto de objetos tridimensionales. Desde la fase inicial, donde se delinea el contexto y la problemática central, hasta la fase final, donde se presenta una solución robusta, se ha recorrido un camino largo y fructífero.

El documento comenzó con una introducción detallada que estableció el contexto y delineó el problema central que se busca abordar. A través de una revisión exhaustiva del estado del arte, se exploraron varias aplicaciones de simetría en tres dimensiones, una herramienta de anotación de simetrías 2D, y métodos de detección de simetrías 3D, incluyendo métodos basados en transformación, correspondencia, votaciones, optimización y aprendizaje. Esto último para justificar la importancia de la creación de un benchmark.

En la fase central del estudio, nos embarcamos en el desarrollo de una herramienta innovadora que facilita el etiquetado de simetrías en objetos tridimensionales. Esta herramienta, que ahora cuenta con las funcionalidades básicas necesarias para llevar a cabo el etiquetado de simetrías, representa un paso significativo en este campo de estudio.

La herramienta fue sometida a pruebas rigurosas por parte de estudiantes y otros usuarios, proporcionando una plataforma para evaluar su usabilidad y eficacia. A través de este proceso, se han recopilado comentarios y sugerencias valiosas que han contribuido significativamente a perfeccionar la herramienta, asegurando su eficacia en la tarea de etiquetar simetrías en objetos tridimensionales.

Una parte crucial de este estudio fue la evaluación de la usabilidad de la herramienta desarrollada. A través de una encuesta SUS, se logró medir el nivel de satisfacción del usuario, que se situó en un 78.5 %. Este resultado indica que la mayoría de los usuarios que participaron en la validación perciben la aplicación como razonablemente usable y fácil de aprender.

Sin embargo, también se reconoce que existe un margen para mejoras adicionales. El 21.5 % restante de los usuarios señaló áreas donde la herramienta podría enfrentar desafíos

o insatisfacciones. Estas áreas de mejora, identificadas durante la evaluación, proporcionan una base valiosa para futuras iteraciones de desarrollo.

6.2. Reflexiones finales y futuras direcciones

En conclusión, este proyecto no solo ha logrado su objetivo de desarrollar una herramienta funcional para el etiquetado de simetrías en objetos tridimensionales, sino que también ha sentado una sólida base para futuras investigaciones y desarrollos en este campo. Todos los objetivos iniciales se han cumplido: se diseñó un visualizador interactivo para manipular objetos 3D, se incorporó la participación del usuario para simetrías reflectantes y rotacionales, se facilitó la visualización y optimización de simetrías, y se creó un archivo JSON para almacenar la información del etiquetamiento.

Mirando hacia el futuro, existe un compromiso en utilizar la retroalimentación valiosa recibida para refinar aún más la herramienta. Se ve un potencial considerable para elevar aún más la satisfacción del usuario y la eficacia de la aplicación mediante ajustes basados en los comentarios recopilados.

En lo personal, el desarrollo de una herramienta de estas características ha representado un desafío sumamente enriquecedor. Mi experiencia académica previa se había centrado principalmente en proyectos relacionados con el desarrollo web y otros campos más comunes en el ámbito universitario. Ingresar al mundo de la computación gráfica fue un giro inusual en mi trayecto educativo.

Es relevante mencionar que, hasta este momento, había tenido la oportunidad de tomar solo un curso de computación gráfica. A pesar de su duración limitada, este curso resultó ser un pilar fundamental en la construcción de las bases necesarias para abordar este proyecto de manera efectiva. Los recursos, conocimientos y técnicas adquiridos durante ese período fueron absolutamente invaluables, ya que proporcionaron la sólida cimentación sobre la cual se construyó esta herramienta.

Explorar las complejidades de la computación gráfica y su aplicación en la visualización de objetos tridimensionales fue un camino desafiante pero gratificante. A lo largo de este proceso, tuve la oportunidad de aplicar y expandir mi comprensión de los conceptos aprendidos en el curso, así como de adquirir nuevos conocimientos a medida que enfrentaba desafíos específicos de este proyecto. La experiencia me permitió profundizar en áreas como la visualización de modelos 3D, trazado de rayos, la gestión de eventos con el mouse, entre otros aspectos cruciales en el desarrollo de aplicaciones de esta naturaleza.

En retrospectiva, esta travesía no solo me brindó un valioso conjunto de habilidades en el campo de la computación gráfica, sino que también me dejó con la satisfacción de haber superado un reto importante en mi formación académica a pesar de las limitaciones de tiempo y recursos.

Bibliografía

[1] Alexander Berner, Michael Wand, Niloy J Mitra, Daniel Mewes, and Hans-Peter Seidel. Shape analysis with subspace symmetries. In *Computer Graphics Forum*, volume 30, pages 277–286. Wiley Online Library, 2011.

[2] Shuhui Bu, Pengcheng Han, Zhenbao Liu, Junwei Han, and Hongwei Lin. Local deep feature learning framework for 3d shape. *Computers & Graphics*, 46:117–129, 2015.

[3] Minho Chang and Sang C Park. Reverse engineering of a symmetric object. *Computers & Industrial Engineering*, 55(2):311–320, 2008.

[4] Marcelo Cicconet, David GC Hildebrand, and Hunter Elliott. Finding mirror symmetry via registration and optimal symmetric pairwise assignment of curves. In *ICCV Workshops*, pages 1749–1758, 2017.

[5] Aleksandrs Ecins, Cornelia Fermuller, and Yiannis Aloimonos. Detecting reflectional symmetries in 3d data through symmetrical fitting. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1779–1783, 2017.

[6] Chris Funk and Yanxi Liu. Symmetry recaptcha. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5165–5174, 2016.

[7] Lin Gao, Ling-Xiao Zhang, Hsien-Yu Meng, Yi-Hui Ren, Yu-Kun Lai, and Leif Kobbelt. Prs-net: Planar reflective symmetry detection net for 3d models. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3007–3018, 2020.

[8] Yuanyue Ge, Ya Xiong, and Pål J From. Symmetry-based 3d shape completion for fruit localisation for harvesting robots. *biosystems engineering*, 197:188–202, 2020.

[9] Carmi Grushko, Dan Raviv, and Ron Kimmel. Intrinsic local symmetries: A computational framework. In *Proceedings of the 5th Eurographics conference on 3D Object Retrieval*, pages 33–38, 2012.

[10] Penglei Ji and Xinguo Liu. A fast and efficient 3d reflection symmetry detector based on neural networks. *Multimedia Tools and Applications*, 78(24):35471–35492, 2019.

[11] Junfeng Jiang, Zhengming Chen, and Kunjin He. A feature-based method of rapidly detecting global exact symmetries in cad models. *Computer-Aided Design*, 45(8-9):1081–1094, 2013.

[12] Wei Jiang, Kai Xu, Zhi-Quan Cheng, and Hao Zhang. Skeleton-based intrinsic symmetry detection on point clouds. *Graphical Models*, 75(4):177–188, 2013.

[13] Xiaoye Jiang, Jian Sun, and Leonidas Guibas. A fourier-theoretic approach for inferring symmetries. *Computational Geometry*, 47(2):164–174, 2014.

[14] Ramakrishna Kakarala, Prabhu Kaliamoorthi, and Vittal Premachandran. Three-dimensional bilateral symmetry plane estimation in the phase domain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–256, 2013.

[15] Simon Korman, Roee Litman, Shai Avidan, and Alex Bronstein. Probably approximately symmetric: Fast rigid symmetry detection with global guarantees. In *Computer Graphics Forum*, volume 34, pages 2–13. Wiley Online Library, 2015.

[16] Bo Li, Henry Johan, Yuxiang Ye, and Yijuan Lu. Efficient view-based 3d reflection symmetry detection. In *SIGGRAPH Asia 2014 Creative Shape Modeling and Design*, pages 1–8. 2014.

[17] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11362–11369, 2020.

[18] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[19] Ke Li, Gilles Foucault, J-C Léon, and Moreno Trlin. Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components. *Computer-Aided Design*, 53:70–89, 2014.

[20] Ming Li, Frank C Langbein, and Ralph R Martin. Detecting approximate symmetries of discrete point subsets. *Computer-Aided Design*, 40(1):76–93, 2008.

[21] Qiqi Liao, Xiaogang Jin, and Wenting Zeng. Enhancing the symmetry and proportion of 3d face geometry. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1704–1716, 2012.

[22] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. In *ACM SIGGRAPH 2010 papers*, pages 1–12. 2010.

[23] Tianqiang Liu, Vladimir G Kim, and Thomas Funkhouser. Finding surface correspondences using symmetry axis curves. In *Computer Graphics Forum*, volume 31, pages 1607–1616. Wiley Online Library, 2012.

[24] Xiuping Liu, Shuhua Li, Risheng Liu, Jun Wang, Hui Wang, and Junjie Cao. Properly constrained orthonormal functional maps for intrinsic symmetries. *Computers & Graphics*, 46:198–208, 2015.

[25] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, and François X Sillion. Accurate detection of symmetries in 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(2):439–464, 2006.

[26] Pavlos Mavridis, Ivan Sipiran, Anthousis Andreadis, and Georgios Papaioannou. Object completion using k-sparse optimization. In *Computer Graphics Forum*, volume 34, pages 13–21. Wiley Online Library, 2015.

[27] Niloy J Mitra, Alex Bronstein, and Michael Bronstein. Intrinsic regularity detection in 3d geometry. In *European Conference on Computer Vision*, pages 398–410. Springer, 2010.

[28] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.

[29] Rajendra Nagar and Shanmuganathan Raman. Detecting approximate reflection symmetry in a point set using optimization on manifold. *IEEE Transactions on Signal Processing*, 67(6):1582–1595, 2019.

[30] Rajendra Nagar and Shanmuganathan Raman. 3dsymm: robust and accurate 3d reflection symmetry detection. *Pattern Recognition*, 107:107483, 2020.

[31] Maks Ovsjanikov, Quentin Mérigot, Viorica Pătrăucean, and Leonidas Guibas. Shape matching via quotient spaces. In *Computer Graphics Forum*, volume 32, pages 1–11. Wiley Online Library, 2013.

[32] Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering structural regularity in 3d geometry. In *ACM SIGGRAPH 2008 papers*, pages 1–11. 2008.

[33] Wu Qiu, Jing Yuan, Eranga Ukwatta, Yue Sun, Martin Rajchl, and Aaron Fenster. Prostate segmentation: an efficient convex optimization approach with axial symmetry using 3-d trus and mr images. *IEEE transactions on medical imaging*, 33(4):947–960, 2014.

[34] Dan Raviv, Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Symmetries of non-rigid shapes. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007.

[35] Dan Raviv, Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Full and partial symmetries of non-rigid shapes. *International journal of computer vision*, 89(1):18–39, 2010.

[36] Aurela Shehu, Alan Brunton, Stefanie Wuhrer, and Michael Wand. Characterization of partial intrinsic symmetries. In *European Conference on Computer Vision*, pages 267–282. Springer, 2014.

[37] Yifei Shi, Junwen Huang, Hongjia Zhang, Xin Xu, Szymon Rusinkiewicz, and Kai Xu. Symmetrynet: learning to predict reflectional and rotational symmetries of 3d shapes from single-view rgb-d images. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[38] Gregor N. C. Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-aware actor-critic for 3d molecular design, 2020.

[39] Ivan Sipiran, Robert Gregor, and Tobias Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library, 2014.

[40] Kilho Son, Eduardo B. Almeida, and David B. Cooper. Axially symmetric 3d pots configuration system using axis of symmetry and break curve. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[41] Pablo Speciale, Martin R Oswald, Andrea Cohen, and Marc Pollefeys. A symmetry prior for convex variational 3d reconstruction. In *European Conference on Computer Vision*, pages 313–328. Springer, 2016.

[42] Art Tevs, Qixing Huang, Michael Wand, Hans-Peter Seidel, and Leonidas Guibas. Relating shapes via geometric symmetries and regularities. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[43] Hui Wang, Patricio Simari, Zhixun Su, and Hao Zhang. Spectral global intrinsic symmetry invariant functions. In *Graphics Interface 2014*, pages 209–215. AK Peters/CRC Press, 2020.

[44] Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhiquan Cheng, and Yueshan Xiong. Symmetry hierarchy of man-made objects. In *Computer graphics forum*, volume 30, pages 287–296. Wiley Online Library, 2011.

[45] Kai Xu, Hao Zhang, Wei Jiang, Ramsay Dyer, Zhiquan Cheng, Ligang Liu, and Baoquan Chen. Multi-scale partial intrinsic symmetry detection. *ACM Transactions On Graphics (TOG)*, 31(6):1–11, 2012.

[46] Yusuke Yoshiyasu, Eiichi Yoshida, Kazuhito Yokoi, and Ryusuke Sagawa. Symmetry-aware nonrigid matching of incomplete 3d surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4193–4200, 2014.

[47] Jing Yuan, Wu Qiu, Eranga Ukwatta, Martin Rajchl, Xue-Cheng Tai, and Aaron Fenster. Efficient 3d endfiring trus prostate segmentation with globally optimized rotational symmetry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[48] Zhiyuan Zhang, KangKang Yin, and Kelvin WC Foong. Symmetry robust descriptor for non-rigid surface matching. In *Computer Graphics Forum*, volume 32, pages 355–362. Wiley Online Library, 2013.

[49] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. Graph.*, 29(4):94–1, 2010.

Anexo

Escala de usabilidad del sistema

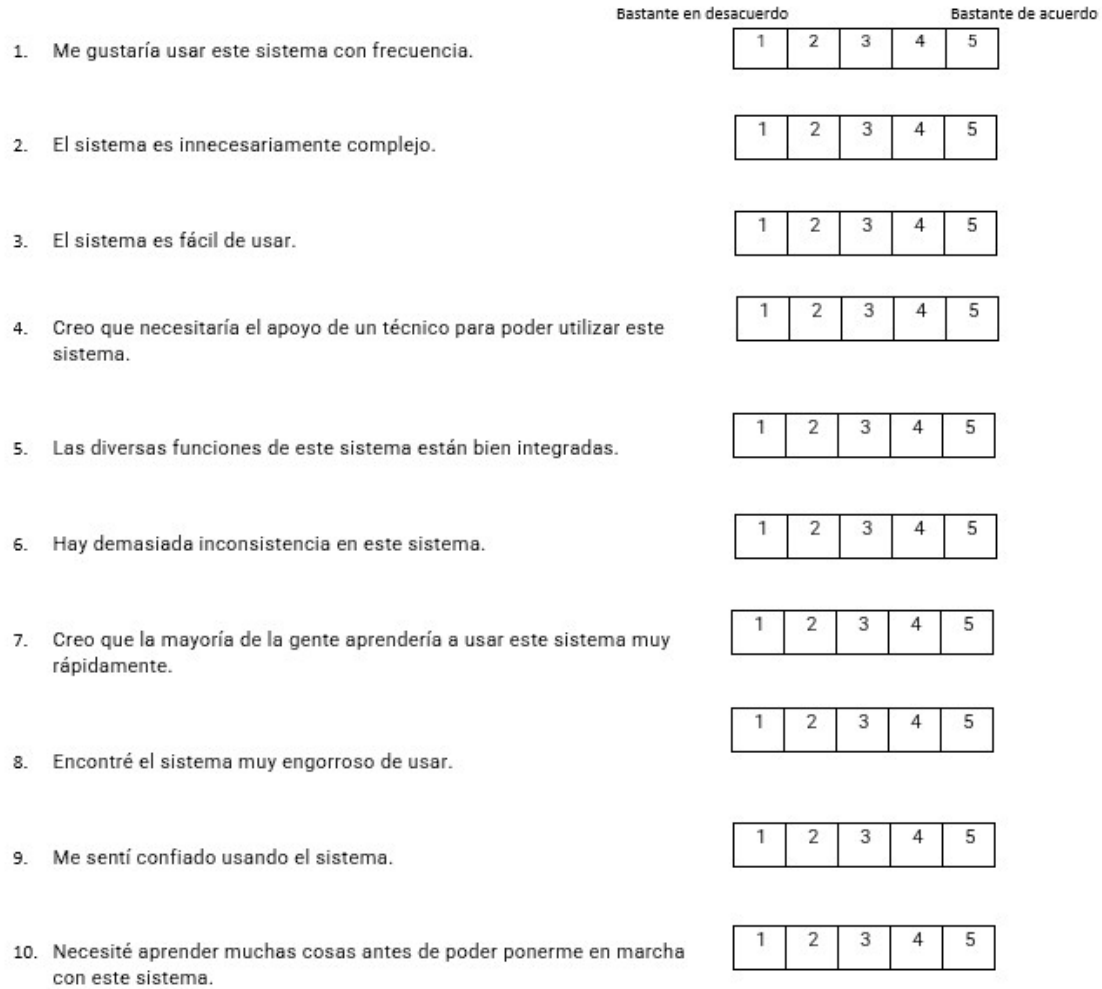

Figura 6.1: Escala de usabilidad del sistema(SUS)