PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE UNA APLICACIÓN DE VISIÓN ARTIFICIAL PARA ESCANEAR OBJETOS 3D CON CÁMARAS DE FOTOS

Tesis para obtener el título profesional de Ingeniero Informático

AUTOR:

Ernie Ludwick Sumoso Vicuña

ASESOR:

Dr. Iván Anselmo Sipiran Mendoza

Lima, Setiembre, 2021

Resumen

Los métodos tradicionales de escaneo 3D requieren de un contacto físico directo con los objetos a escanear. En algunos casos demandan incluso la rotación y movimiento constante de estos, lo cual representa un riesgo para objetos frágiles como las piezas arqueológicas. Si ignoramos este factor de riesgo podemos causar daños irreparables y frustrar el proceso de documentación de estas. Por otro lado, existen técnicas de escaneo 3D sin contacto - pasivas que no requieren de una interacción directa con los objetos. Es por ello que se propone desarrollar un software que permita escanear piezas arqueológicas usando las técnicas de reconstrucción tridimensional mediante conceptos de visión artificial, aprendizaje de máquina, *data augmentation* y mallas poligonales.

Para lograr nuestro cometido se parte de un set inicial de 962 huacos peruanos pre escaneados proporcionados por el grupo de Inteligencia Artificial PUCP. Con este conjunto de datos se genera una extenso volumen de imágenes los cuales son procesados y utilizados para el entrenamiento de un modelo de aprendizaje de máquina. Segundo, al obtener unos primeros resultados se propone llevar a cabo la técnica de *data augmentation* para extender nuestra data disponible, normalizarla, segmentarla y con ello entrenar múltiples modelos bajo 2 experimentos definidos. Todo ello nos permite mejorar los resultados de reconstrucción de objetos 3D considerando la alta variabilidad de huacos peruanos. Finalmente se implementa una interfaz gráfica la cual permite al usuario interactuar con el proyecto desarrollado.

En conclusión, se logra desarrollar una herramienta de software que nos permite cargar videograbaciones reales de piezas arqueológicas (bajo ciertos parámetros establecidos), procesar los archivos, visualizar y descargar los resultados obtenidos como mallas poligonales (reconstrucciones 3D almacenados en el computador).

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1. Generalidades

1.1 Problemática

1.1.1 Árbol de problemas

Tabla 1. Árbol de problemas

Problemas consecuencia	Se mantienen los métodos tradicionales de escaneo 3D (de contacto) los cuales son más precisos, pero en el caso de piezas arqueológicas representan un riesgo constante de preservación física	Los resultados obtenidos de reconstrucciones 3D a partir de métodos de escaneo sin contacto – pasiva no son suficientemente precisos o son totalmente incorrectos en comparación a los métodos tradicionales	Se generan daños irreparables a las piezas arqueológicas escaneadas en 3D mediante métodos tradicionales (de contacto)
Problema central	El proceso tradicional de escaneo 3D de las piezas arqueológicas pone en riesgo la preservación física de estas.		
Problemas causa	No se enfoca algún método de escaneo 3D sin contacto – pasivo en la reconstrucción de huacos peruanos específicamente	Los métodos de escaneo sin contacto – pasivos no consideran la gran variedad de huacos peruanos y poca disponibilidad de estos	Se requiere de un contacto físico directo y manipulación de las piezas arqueológicas para llevar a cabo el proceso de escaneo 3D.

1.1.2 Descripción

Las tecnologías de escaneo 3D se pueden dividir en dos grupos: de contacto y sin contacto. El primer grupo se caracteriza por la necesidad de reposar el escáner sobre el objeto (Sreenivasa K. 2003). Dicha actividad representa un riesgo para los objetos frágiles ya que no deben ser manipulados. Por otro lado, el segundo grupo no requiere de un contacto físico

directo, pero a su vez se divide en dos subgrupos: técnicas de escaneo activas y pasivas (Pears N. 2012). Las técnicas activas se basan en la medición del tiempo de envío y retorno de una señal emitida hacia el objeto con el fin de estimar o capturar la superficie del mismo. Dentro de esta categoría los métodos más usados son: "tiempo de vuelo" (comúnmente llamada "tecnología láser") y "luz estructurada". Por otro lado, el grupo de técnicas pasivas solo aprovechan la luz ambiental haciendo uso de fotos o videograbaciones para predecir la forma de los objetos capturados. Algunas técnicas representativas de este grupo son: estereoscopía y reconstrucción 3D en base a siluetas (Pears N. 2012). En la siguiente figura encontramos el árbol de tecnologías de escaneo 3D mencionadas.

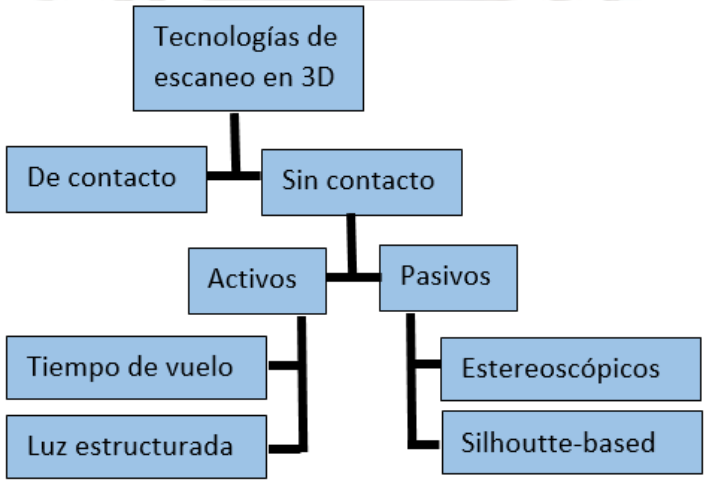


Figura 1. Árbol de tecnologías de escaneo en 3D

En el Perú se realizan los primeros esfuerzos por aplicar estas nuevas tecnologías trabajando con múltiples colecciones arqueológicas y fondos documentales con el fin de preservar y documentar la historia de la cultura prehispánica. Algunos de los museos que aplican o han aplicado estas tecnologías son el Museo Pachacamac y el Museo de Arqueología Josefina Ramos de Cox (Museo Josefina Ramos de Cox, n.d.). Dentro de las colecciones arqueológicas que se manejan encontramos una gran variedad de huacos peruanos los cuales son escaneados en 3D como parte de un proceso de documentación. Muchos de estos procesos requieren de un contacto físico directo con las piezas arqueológicas, lo que pone en riesgo su

preservación. También existen casos en donde los huacos son muy grandes y no caben en el entorno de los escáneres u otros casos en donde son descartados debido al riesgo que implica manipularlos (IA PUCP, n.d.). Todos estos factores representan un riesgo constante y retrasan el proceso de documentación de las piezas arqueológicas, a veces incluso no llevándose a cabo.

En conclusión, dada la fragilidad y variedad de los huacos peruanos, es necesario implementar una herramienta que facilite el proceso de escaneo 3D de dichos objetos. Para ello se propone desarrollar una solución que emplee una tecnología de escaneo sin contacto - pasiva haciendo uso de modelos de aprendizaje de máquina para reconstruir piezas arqueológicas en 3D en el computador a partir de videograbaciones o secuencias de imágenes RGB.

1.1.3 Problema seleccionado

El problema central radica en que el proceso de escaneo 3D tradicional de piezas arqueológicas pone en riesgo la preservación física de estas debido al contacto directo necesario, la manipulación y rotación que estas deben soportar durante el proceso; por ello, se necesita desarrollar una herramienta que aplique alguna técnica de escaneo sin contacto – pasiva y permita la reconstrucción 3D de estos objetos en el computador.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un software para el escaneo en 3D de piezas arqueológicas usando las técnicas de reconstrucción de objetos mediante visión artificial, aprendizaje de máquina, *data augmentation* y mallas poligonales.

1.2.2 Objetivos específicos

OE1. Preprocesar el conjunto de datos de piezas arqueológicas para el entrenamiento y prueba de un modelo de aprendizaje de máquina

OE2. Adaptar el proyecto y mejorar la reconstrucción de objetos 3D considerando la alta variabilidad de las piezas arqueológicas

OE3. Implementar un medio de interacción entre el usuario y la herramienta desarrollada para el escaneo 3D de piezas arqueológicas

1.2.3 Resultados esperados

OE1. R1. Conjunto de datos generado: renderizado de secuencias

R2. Datos segmentados en 2 subconjuntos: entrenamiento y prueba

R3. Modelo entrenado con los datos preprocesados

OE2. R4. Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos

R5. Videos construidos que simulan grabaciones reales de piezas arqueológicas R6. *Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada

OE3. R7. Interfaz gráfica integrada con la herramienta desarrollada y validada mediante pruebas funcionales

1.2.4 Mapeo de objetivos, resultados y verificación

Tabla 2. Mapeo O1

Objetivo 1: Preprocesar el conjunto de datos de piezas arqueológicas para el entrenamiento y prueba de un modelo de aprendizaje de máquina		
Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R1. Conjunto de datos generado: renderizado de secuencias	Carpeta que contiene las imágenes generadas a partir de las piezas arqueológicas	100% de piezas arqueológicas procesadas exitosamente
R2. Datos segmentados en 2 subconjuntos:	• Archivos LIST del proyecto base PMO creados y modificados	100% de objetos segmentados y

entrenamiento y prueba	• *Script* desarrollado para la automatización de la tarea	organizados en 2 archivos LIST independientes.
R3. Modelo entrenado con los datos preprocesados	• *Checkpoints* del modelo guardados durante el entrenamiento • Reconstrucciones 3D de huacos peruanos empleando el modelo entrenado	• Al menos 10 *checkpoints* generados durante el entrenamiento • Al menos 4 reconstrucciones de prueba

Tabla 3. Mapeo O2

Objetivo 2: Adaptar el proyecto y mejorar la reconstrucción de objetos 3D considerando la alta variabilidad de las piezas arqueológicas		
Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R4. Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos	Tabla con los resultados cuantitativos obtenidos del modelo inicial y los experimentos realizados.	Los indicadores de los últimos resultados (experimentos) deben ser mejores que los del modelo inicial.
R5. Videos construidos que simulan grabaciones reales de piezas arqueológicas	Los archivos de video generados	21 archivos MP4 generados cuyo contenido son videograbaciones a piezas arqueológicas
R6. *Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada	Flujo y *script* desarrollados que definen el *pipeline* de conversión de videos a archivos NPY	• 1 flujo desarrollado • 1 *script* desarrollado

Tabla 4. Mapeo O3

Objetivo 3: Implementar un medio de interacción entre el usuario y la herramienta desarrollada para el escaneo 3D de piezas arqueológicas		
Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R7. Interfaz gráfica integrada con la herramienta desarrollada	Interfaz gráfica funcional	100% de pruebas funcionales concluidas con éxito

y validada mediante pruebas funcionales

1.3 Métodos y Procedimientos

Tabla 5. Tabla de herramientas y métodos

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos
R1. Conjunto de datos generado: renderizado de secuencias	Generación de una vasta cantidad de secuencias de imágenes a partir de los huacos peruanos pre escaneados, siguiendo un patrón de movimiento definido y modificando el código de la herramienta a utilizar.	• Stanford Shapenet Renderer • Python • PuTTY
R2. Datos segmentados en 2 subconjuntos: entrenamiento y prueba	Inicialización del proyecto base PMO y desarrollo de un *script* en Python para la automatización de la segmentación de un conjunto de datos, modificando/creando los archivos LIST correspondientes y definiendo una distribución de 80% - 20%.	• PMO • Python • Sublime Text 3 • PuTTY
R3. Modelo entrenado con los datos preprocesados	Generación de las nubes de puntos o archivos PLY y NPY a partir de los huacos peruanos. Desarrollo de *script*s para la normalización, procesamiento e integración de la data al proyecto base PMO. Entrenamiento de un nuevo modelo de aprendizaje de máquina con la obtención de unos primeros resultados cualitativos.	• PMO • Python • Sublime Text 3 • PuTTY
R4. Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos	Hallar las escalas individuales de cada objeto, realizar una segmentación manual interna para definir subcategorías dentro del set inicial de 962 piezas arqueológicas, desarrollar *script*s para la normalización de la data y generación de imágenes de las subcategorías aplicando la técnica de *data augmentation*. Finalmente, entrenar nuevos modelos de aprendizaje de máquina bajo 2 experimentos definidos, obteniendo resultados cualitativos y cuantitativos.	• PMO • Python • Sublime Text 3 • PuTTY • Blender • *Data augmentation*
R5. Videos construidos que simulan grabaciones reales de piezas arqueológicas	Edición de videos para la construcción de grabaciones simuladas a piezas arqueológicas, definiendo los rangos de resolución y duración de los videos.	• Editor de videos de la aplicación “Fotos” de Windows 10

R6. *Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada	Definición de un proceso o *pipeline* que permita la conversión del archivo de video *input* a un archivo NPY procesable por los modelos entrenados. Establecer los parámetros permitidos para los videos como la resolución, duración, formatos y selección de fotogramas clave. Finalmente, plasmar este *pipeline* en código.	• Python • Sublime Text 3 • FFmpeg
R7. Interfaz gráfica integrada con la herramienta desarrollada y validada mediante pruebas funcionales	Implementación de una interfaz gráfica y con los servicios web correspondientes alojados en el servidor de IA-PUCP. Integrar la interfaz con el proyecto desarrollado y probar las funcionalidades mediante pruebas funcionales unitarias.	• PMO • Python • Sublime Text 3 • PuTTY • Visual Studio Code • VueJS • Flask

A continuación, se describen las herramientas y/o métodos mencionados:

- Stanford-Shapenet-Renderer: herramienta que permite generar una serie de imágenes a partir de archivos en formato OBJ, por defecto genera 30 imágenes uniformemente espaciadas alrededor de los 360 grados sexagesimales del objeto. (ShapeNet-Renderer, 2018).
- Python: lenguaje de programación de alto nivel y multiparadigma, se desarrolló a fines de la década de 1980 y en la actualidad es utilizado en incontables proyectos de ciencias de la computación (python org, s.f.).
- PuTTY: emulador de terminal con múltiples funcionalidades, entre ellas está realizar conexiones SSH con equipos remotos. En este proyecto servirá para establecer una conexión entre mi equipo personal y el servidor IA-PUCP donde se alojará el proyecto.
- PMO: es el diminutivo del paper "*Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction*" (Lin et al., s. f.) el cual fue presentado en la CVPR 2019 (Congreso sobre el Reconocimiento de Patrones y Visión Artificial). Dicho proyecto propone la optimización fotométrica de mallas poligonales para mejorar la

reconstrucción de objetos 3D y nos crea un ámbito ideal para el entrenamiento de modelos de aprendizaje de máquina e integración de nuestra data de huacos peruanos.

- Sublime Text 3: editor de textos que cuenta con múltiples funciones y ayudas gráficas para facilitar la edición de código fuente. En este caso, es utilizado para la creación y edición de todos los *script*s desarrollados.
- Blender: software aplicado a la animación, modelado, renderizado y grabación de objetos 3D en el computador, permite exportar e importar archivos de mallas poligonales universalmente conocidos como PLY, OBJ, STL, etc. (Blender, n.d.).
- *Data augmentation*: técnica que consiste en aumentar la cantidad de datos disponibles de un conjunto de datos mediante la modificación de la data original, de tal forma que se creen copias ligeramente modificadas con el propósito de aumentar el universo de datos disponibles
- Aplicación "Fotos" de Windows 10: aplicación de Windows 10 la cual tiene una funcionalidad de crear y editar videos mediante fotos. En este caso se emplea para la creación de videograbaciones simuladas a piezas arqueológicas.
- FFmpeg: software libre que permite trabajar con archivos de audio y video, entre sus funcionalidades está la conversión de archivos en múltiples formatos, escalar la resolución, grabación, etc.
- Visual Studio Code: software editor de código fuente desarrollado por Microsoft, facilita la depuración, refactorización y compilación del código; en este caso es utilizado para el desarrollo de la interfaz gráfica.
- Vue.js: *framework* open *source* de Java*Script* empleado para el desarrollo web principalmente enfocado en el *front-end* o interfaz (Vue, n.d.).
- Flask: *framework* de Python empleado para el desarrollo de aplicaciones web e interfaces; en este caso es utilizado para los servicios web alojados en el servidor.

1.4 Alcance y limitaciones

1.4.1 Alcance

Para lograr nuestro cometido se partirá de un set inicial de datos de 962 piezas arqueológicas pre escaneadas las cuales pasarán por un proceso de normalización, generación de imágenes y nubes de puntos. Además, se aplicarán conceptos de visión artificial, modelos de aprendizaje de máquina, técnicas como "*data augmentation*" y se empleará una representación de mallas poligonales de los objetos 3D reconstruidos. También se implementará una interfaz gráfica la cual permitirá al usuario interactuar directamente con la herramienta desarrollada ingresando sus videograbaciones de piezas arqueológicas, esperando a que se concluya el procesamiento y obteniendo las reconstrucciones 3D correspondientes (con posibilidad de observarlas inmediatamente y descargar los archivos). El resultado final reduce considerablemente el factor de riesgo de preservación física de los objetos, ya que el usuario solo deberá grabar videos (bajo ciertos estándares) de las piezas e ingresarlas a la interfaz para obtener resultados También es importante resaltar que para el entrenamiento de nuestros modelos y ejecución de las reconstrucciones se hace uso del proyecto base PMO o *Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction*, presentado en la CVPR 2019, el cual provee el ámbito adecuado para estas tareas. Este proyecto base se modifica y se adapta convenientemente durante el desarrollo de esta tesis para lograr nuestro objetivo final.

1.4.2 Limitaciones

- El tiempo de reconstrucción de las piezas arqueológicas tarda aproximadamente 5 minutos por cada objeto a reconstruir.
- Las extensiones de archivo permitidas para las videograbaciones *input* solo podrán ser archivos MP4 o AVI.

- Las mallas poligonales obtenidas como resultado a partir de las reconstrucciones de las piezas arqueológicas serán archivos en formato PLY, los cuales permiten guardar objetos 3D en el computador.
- Las pruebas y videograbaciones construidas solo se realizan bajo el rango de resoluciones 540p hasta 1080p y duraciones de 10 hasta 40 segundos. Videos fuera de estos rangos pueden aún funcionar, pero no se han realizado las pruebas correspondientes en este proyecto de investigación.
- Para los videos, la cantidad mínima de fotogramas permitidos son desde 72 hasta 10 000 fotogramas, los cuales a 30 fotogramas por segundo implican duraciones de 2.4 hasta 333 segundos.
- Las videograbaciones realizadas deben seguir cierto patrón de movimiento de la cámara (detallado en el primer resultado esperado) y mantener el objeto lo más alineado posible al centro del fotograma en todo momento.

Capítulo 2. Marco Conceptual

2.1 Introducción

En el presente marco conceptual se detallan algunos conceptos importantes que el lector debe tener presente para comprender la investigación y resultados obtenidos. Estos términos nos ayudan a contextualizar la problemática y representan una base para lo que se quiere desarrollar como objetivo final de este proyecto de investigación.

2.2 Desarrollo del marco

2.2.1 Visión Artificial

El campo de la visión artificial (también llamada "*computer vision*") se encarga del procesamiento y análisis de imágenes o videos para su entendimiento a alto nivel. También se define como la automatización del sistema visual del ser humano. En general, consiste en

extraer información no trivial de imágenes para la comprensión de estos en el computador al mismo nivel que puede realizarlo un ser humano (Forsyth & Ponce, 2002). Por lo tanto, este campo se encarga de replicar las capacidades visuales del ser humano mediante algoritmos, técnicas, aprendizaje de máquina, etc. (Forsyth & Ponce, 2002). Algunas de sus aplicaciones son: el modelado de objetos en base a imágenes, el reconocimiento y reconstrucción de objetos 3D.

2.2.2 Fotometría

La fotometría es el campo de la ciencia que estudia las medidas de la luz y la sensibilidad del ojo humano frente a las diferentes longitudes de ondas electromagnéticas. El ojo humano puede percibir un cierto rango de colores dependiendo de la longitud de las ondas electromagnéticas (380-760 nm), fuera de este rango nuestro órgano visual no es capaz de percibir color alguno. La fotometría juega el papel de medir qué tan sensible es el ojo humano con respecto a la luz con ayuda de las funciones fotópica y escotópica, las cuales modelan la visión del ojo en condiciones de alta y baja luminosidad respectivamente (Stimson, 1974). Además, este campo se asocia con la visión artificial en el sentido de que se simulan las capacidades visuales del ser humano bajo distintas condiciones de luz (Shashua, 1992).

2.2.3 Métodos de reconstrucción 3D pasivos

Los métodos de reconstrucción 3D pasivos se encargan de reconstruir digitalmente un objeto o una escena 3D a partir de imágenes aprovechando únicamente la iluminación ambiental. Para lograr el cometido se aplican conceptos de geometría epipolar, la cual se encarga de estudiar relaciones geométricas entre puntos en el espacio y su proyección sobre planos en 2D (Pears N. 2012). Estos planos se relacionan estrechamente con las imágenes tomadas por una cámara de video. Para aplicar los métodos pasivos no es necesario interactuar

con la escena a reconstruir, ya que las distancias son calculadas desde los puntos de vista de las cámaras. (D. Nister, 2004).

2.2.4 Redes Neuronales

Las redes neuronales son modelos de aprendizaje de máquina que permiten simular (bajo ciertas limitaciones) el aprendizaje de un ser humano en el computador. Dichos modelos son análogos al sistema de neuronas que posee un cerebro humano (Zeiler & Fergus, 2014). Cada neurona recibe entradas, las procesa y emite salidas. Estas salidas a su vez son entradas de otras neuronas las cuales vuelven a procesar la data y emiten otras salidas; el proceso se repite múltiples veces. Una red neuronal consiste en varias neuronas conectadas entre sí de forma ordenada y organizadas por capas, donde la primera capa recibe las entradas generales de la red, luego están las capas intermedias que realizan el proceso mencionado y finalmente la última capa que emite los resultados finales (Zeiler & Fergus, 2014). Generalmente a la simulación del aprendizaje de un modelo se le conoce como la etapa de "entrenamiento". Para ello es necesario una extensa cantidad de datos los cuales son procesados por la red neuronal.

2.2.5 *Data augmentation*

La técnica de *data augmentation* consiste en crear copias ligeramente modificadas a partir de un conjunto de datos ya existente. Esta técnica nos permite aumentar nuestro universo de data manteniendo la unicidad de cada elemento y partiendo de un grupo reducido. El entrenamiento de las redes neuronales dependen de la existencia de un gran volumen de datos que puedan ser procesados. Esta técnica nos permite expandir nuestro set inicial de data mejorando el desempeño de los modelos de aprendizaje de máquina y aprovechando mejor la variedad de datos.

2.2.6 Mallas poligonales

Existen múltiples formas de representar objetos 3D en el computador. Uno de ellos son las mallas poligonales, estas están compuestas por un conjunto de vértices y caras conectadas entre sí de tal manera que componen una superficie. Generalmente las caras son triángulos, aunque algunas veces también se trata de cuadriláteros o polígonos de "n" lados en general. La cantidad de caras existentes en una malla es directamente proporcional al detalle del objeto reconstruido y también al poder de procesamiento requerido para renderizar y/o manipular la superficie. En gráficos de computación existen múltiples algoritmos para simplificar, suavizar, detectar colisiones entre 2 o más mallas, etc.

Capítulo 3. Estado del Arte

3.1 Introducción

La reconstrucción de superficies 3D empleando técnicas de escaneo es un tema que se ha venido desarrollando a profundidad en la última década. El campo encargado de su estudio es la visión artificial dentro de las ciencias de la computación. Actualmente, en este campo existen múltiples técnicas para la generación de superficies a partir imágenes. Muchos de estos métodos han sido retomados recientemente a pesar de que los primeros trabajos datan de la década de 1990 (Blanz & Vetter, 1999). Con el reciente desarrollo de la tecnología se han impulsado este tipo de proyectos y consecuentemente se han descubierto nuevos métodos y tecnologías de reconstrucción de superficies. Los trabajos de investigación en este campo pueden clasificarse dependiendo del propósito (o enfoque) de la reconstrucción. Existen métodos que facilitan el reconocimiento de un solo objeto en una imagen, otros que permiten detectar múltiples objetos a partir de una imagen panorámica, etc. Veremos que estos métodos serán empleados en casos específicos y para suplir necesidades específicas. También pueden clasificarse dependiendo de la técnica empleada para la representación del objeto 3D en el

computador, ya que existen múltiples formas de adquirir, procesar y guardar la data. Para entender mejor estas tecnologías emergentes (así como la interpretación de resultados) ahondamos en el estado del arte empleando una revisión sistemática de la literatura

3.2 Objetivos de revisión

El objetivo principal de esta revisión sistemática es identificar las diferentes técnicas de representación de superficies 3D en el computador, así como reconocer los métodos de reconstrucción de superficies que se vienen desarrollando recientemente. Esto nos brindará un contexto teórico adecuado y actualizado a las tecnologías emergentes en el campo de la visión artificial. Se utilizarán los criterios PICOC para estructurar los elementos de los objetivos de la revisión sistemática.

Tabla 6. Criterios PICOC

Population	Escaneo de superficies 3D empleando cualquier técnica de representación de las superficies Modelos entrenados para una visión artificial pero empleados para objetos de cualquier tipo
Intervention	Aplicación de aprendizaje de máquina para el procesamiento de imágenes que permitan obtener una superficie altamente variable
Comparison	Entre modelos de reconstrucción de superficies 3D a partir de una secuencia de imágenes o videos
Outcome	Identificación de las recientes técnicas empleadas para representar superficies en el computador y el uso de modelos de *machine learning* para el escaneo 3D de objetos reales
Context	Múltiples trabajos de investigación pertenecientes al área de ciencias de la computación que emplean *deep learning* para el escaneo 3D de objetos en el mundo real

3.3 Preguntas de revisión

Con el fin de poder entender las diferentes técnicas y métodos existentes en el estado del arte sobre las reconstrucciones 3D de objetos necesitamos clasificar los enfoques, así como las técnicas utilizadas para la representación de objetos 3D en el computador. Frente a estos métodos podemos plantear las siguientes preguntas de investigación:

- P1. ¿Qué enfoques se buscan con la generación de superficies u objetos reales a partir de imágenes?
- P2. ¿Cuáles son las técnicas que se utilizan para representar un objeto 3D en el computador?

3.4 Estrategia de búsqueda

Para llevar a cabo el proceso de búsqueda se usarán como herramientas algunos motores de búsqueda reconocidos y respaldados por instituciones académicas a nivel mundial. Estos motores son:

- *IEEE Xplore Digital Library*: base de datos para investigaciones, como su nombre lo indica es una librería digital que contiene millones de artículos, revistas científicas, *papers*, etc. relacionados a temas de ciencias e ingeniería (IEEE 2020). Este motor de búsqueda está respaldado por ambas instituciones: IEEE (*Institute of Electrical and Electronics Engineers*) e IET (*Institution of Engineering and Technology*).
- ACM DL: por sus siglas en inglés *Association for Computing Machinery Digital Library* es una librería digital perteneciente a la asociación estadounidense ACM fundada en 1946 con fines académicos como una sociedad científica dedicada específicamente a la informática y ciencias de la computación, actualmente tiene presencia en más de 100 países.
- Springer: editorial alemana con publicaciones multinacionales cuyo motor de búsqueda contiene múltiples disciplinas relacionadas a la ciencia incluyendo ciencias de la computación e ingeniería.
- MIT Libraries: versión oficial y digital de la librería del conocido MIT (*Massachusetts Institute of Technology*) cuyo contenido multimedia se encuentra distribuido entre las más de 400 bases de datos de la institución.

A continuación, se muestran los resultados de las búsquedas.

Tabla 7. Resultados de Búsqueda

Motor de búsqueda	**Cadena de búsqueda**	**Cantidad de resultados**	**Preguntas de revisión relacionadas**
IEEE Xplore	"*machine learning*" AND "*computer vision*" AND ("*3D object reconstruction*" OR "*3D shape recognition*")	24	1
IEEE Xplore	("*object classification*" OR "*object detection*" OR "*Object segmentation*") AND "*machine learning*"	41	1
ACM	("*computer stereo vision*" OR "*computer vision*") AND "*3D shape recognition*"	3	1
ACM	"*neural networks*" AND ("*generating shape*" OR "*volumetric shape*")	2	2
Springer	"*photometric mesh optimization*"	1	2
MIT Libraries	"convolutional *neural networks*" AND ("*3D object classification*" OR "*3D shape recognition*")	2	1
MIT *libraries*	"*3D pose estimation*" OR "*3D geometry reconstruction*"	2	1

3.5 Criterios de inclusión/exclusión

Se incluirán los estudios que cumplan con los siguientes criterios:

- Los resultados del estudio son replicables.
- El idioma del estudio es el inglés o español.
- El modelo involucrado en el estudio posee un score de precisión superior al 60%.

Se excluirán los estudios que cumplan con los siguientes criterios:

- El estudio no es de libre acceso.
- El estudio fue desarrollado hace más de 10 años, debido a que los modelos de aprendizaje de máquina enfocados en el reconocimiento de superficies presentan mejores métricas en la última década.
- El estudio no realiza pruebas con videos (grabados del mundo real), puesto que el propósito es también investigar casos reales aplicables.

3.6 Formulario de extracción de datos

El siguiente formulario de extracción de datos es aplicable para ambas preguntas de revisión planteadas previamente. De manera general, los datos extraídos de cada artículo son: autores, título, año de publicación, tipo de bibliografía, fecha de extracción, técnicas y enfoques presentados, imágenes y referencias.

Tabla 8. Formulación de extracción de datos

Campo	**Descripción**	**Ejemplo**	**Pregunta**
ID	-	P01	General
Autor/es	-	Pears N.	General
Título	-	Multi-view Convolutional *Neural networks* for 3D Shape Recognition	General
Año de publicación	-	2015	General
Tipo de bibliografía	-	Paper, revista de investigación, libro, etc.	General
Fecha de extracción	-	09/2019	General
Motor de búsqueda	-	IEEE Xplore	General
Técnica de representación	Técnica empleada para la representación del objeto/forma en el computador	*Multi-view Images* (Imágenes de vista múltiple)	P2

Enfoque del reconocimiento	Propósito del desarrollo del modelo de reconocimiento de superficies en 3D	Reconocimiento y clasificación de múltiples objetos dentro de una propia escena/imagen panorámica. Reconstrucción de la forma de un único objeto a partir de un conjunto de imágenes.	P1
Problemática abordada	Problemática a partir del cual parte el trabajo de investigación.	Análisis de la viabilidad de la clasificación de múltiples objetos en una misma imagen.	P1, P2

3.7 Resultados de la revisión

Luego de aplicar los criterios de inclusión y exclusión nos quedamos con los siguientes estudios primarios obtenidos de los distintos motores de búsqueda. Los resultados completos se encuentran al final del documento en el Anexo A.

Tabla 9. Lista de estudios primarios utilizados

ID	Título	Autor	Año	Motor	Tipo
R01	*Category-Specific Object Reconstruction from a Single Image*	A. Kar, S. Tulsiani, J. Carreira, & J. Malik	2015	IEEE Xplore	CVPR
R02	*Volumetric and Multi-View CNNs for Object Classification on 3D Data*	Qi, C. R., Su, H., Niessner, M., Dai, A., Yan, M., & Guibas, L. J.	2016	IEEE Xplore	CVPR
R03	*Object Detection in 3D Scenes Using CNNs in Multi-view Images*	Qi Charles	2016	IEEE Xplore	Paper
R04	*ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes*	Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T.	2017	IEEE Xplore	CVPR
R05	*Learning Category-Specific Mesh Reconstruction from Image Collections*	Kanazawa, A., Tulsiani, S., Efros, A. A., & Malik, J.	2018	ACM	ECCV

R06	*Multi-view Convolutional Neural networks for 3D Shape Recognition*	Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E.	2015	ACM	ICCV
R07	*3D ShapeNets: A Deep Representation for Volumetric Shapes*	Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, & J. Xiao	2015	ACM	CVPR
R08	*VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*	Y. Zhou, & O. Tuzel.	2018	MIT Libraries	CVPR
R09	*Variational Autoencoders for Deforming 3D Mesh Models*	Q. Tan, L. Gao, Y. Lai, & S. Xia	2018	MIT Libraries	CVPR
R10	*3D-PRNN: Generating shape Primitives with Recurrent Neural networks*	Z. Chuhang, E. Yumer, J. Yang, D. Ceylan & D. Hoiem	2017	IEEE Xplore	Paper
R11	*Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction*	Lin, C.-H., Wang, O., Russell, B. C., Shechtman, E., Kim, V. G., Fisher, M., & Lucey, S	2019	Springer	CVPR

3.8 Discusión

En esta sección se discutirán los resultados de la revisión sistemática enfocado a responder las preguntas de investigación planteadas en las secciones previas. Para ello, cada pregunta de investigación representará una subsección bajo el cual se mencionarán y discutirán los trabajos de investigación relacionados.

3.8.1 ¿Qué enfoques se buscan con la generación de superficies u objetos reales a partir de imágenes?

En el estado del arte se han identificado múltiples técnicas de generación de superficies a partir de imágenes. Entre los enfoques hallados encontramos: la estimación de la posición 3D de un objeto a partir de una única imagen, la clasificación de un objeto, la detección de

múltiples objetos en una escena, la segmentación de una escena por regiones y finalmente la reconstrucción y síntesis geométrica 3D a partir de una colección de imágenes.

3.8.1.1 Estimación de posición 3D: "*Category-Specific Object Reconstruction from a Single Image* (2014)"

El ser humano es capaz de reconstruir un objeto en su mente a partir de una sola imagen. Esto lo podemos lograr gracias a los años de experiencia que tenemos viendo otros objetos de la misma clase. Con clases nos referimos a tipos de objetos (ej.: carros, aviones, motocicletas, televisores, etc.). Pueden existir muchas variaciones dentro de cada clase, lo que llamamos subclases o subcategorías.

Para lograr una reconstrucción digital se necesita construir un modelo que reciba como entrada pixeles y genere una superficie 3D de la clase y subcategoría correspondiente (A. Kar et al. 2015). Para ello se modeló un proceso donde a partir de un conjunto de imágenes pertenecientes a una misma subcategoría se estima el punto de vista por cada imagen. El *framework* de NRSfM o *Non-rigid Shape from Model* propuesto por Bregler C. et al. se emplea para las estimaciones de los puntos de vistas esparcidos alrededor del objeto (C. Bregler, A. Hertzmann, & H. Biermann, 2000). El siguiente paso combina las siluetas detectadas en cada imagen con los puntos de vista calculados para generar distintas formas medias en 3D (A. Kar et al. 2015). Estas formas son deformables y por ende capturan las variaciones que se presentan con las diferentes imágenes. Estas variaciones con respecto a la forma aprendida media del objeto se denominan variaciones intraclase (variación de superficie entre objetos de una misma subclase). En el presente trabajo de tesis encontramos que dentro de la categoría: "piezas arqueológicas" y subcategoría: "huacos" los objetos presentan múltiples diferencias, lo que complica el entrenamiento del modelo.

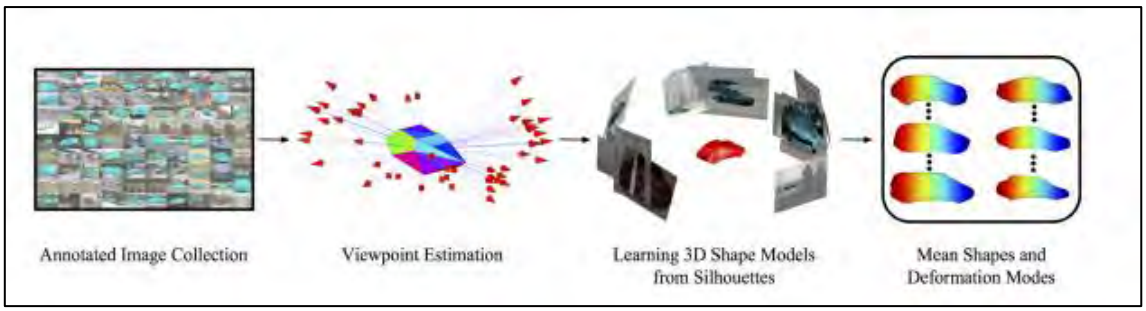


Figura 2. Flujo para la reconstrucción de un objeto a partir de una colección de imágenes (A. Kar et al., 2015)

3.8.1.2 Clasificación de un único objeto: "*Volumetric and Multi-View CNNs for Object Classification on 3D Data* (2016)"

Las redes neuronales convolucionales (CNNs) son utilizados en trabajos de investigación para resolver problemas de clasificación de imágenes. Debido a que la captura de información 3D de objetos es desarrollada constantemente, las CNN's juegan un papel importante en la clasificación de dichos objetos (Qi et al., 2016). Existen dos tipos de CNN's principales en este campo: *volumetric CNN's* y *multi-view CNN's*, las cuales son utilizadas cuando las representaciones de los objetos en el computador son volumétricas o multi-vistas respectivamente. Se puede afirmar que ambas herramientas son el soporte del entendimiento del espacio para el proceso de clasificación (Qi et al., 2016).

Las CNN's son principalmente ventajosas en "aprender" características sobre objetos en imágenes o vídeos RGB. Esto permite la identificación de las características principales. El uso de esta herramienta en el campo 3D empieza a partir de un conjunto de datos (representaciones en 3D de objetos) donde se detectan las características principales de los grupos de objetos, a esto se le llama fase de entrenamiento (Qi et al., 2016). Generalmente 14 se utiliza data de diferentes bases de datos para entrenar el modelo y datos del mundo real para las pruebas. Luego, con el modelo entrenado se puede procesar data de casos reales a través de las diferentes capas (*convolutional layer*, *pooling layer* y *fully connected layer* o *FC layer*) para la clasificación.

El trabajo presentado por Qi Charles, Su Hao et al. en 2016 pretende explorar las características de las CNN's volumétricas y multi-vistas. Ellos buscan analizar y mejorar el uso de dichas herramientas. La representación multi-vista constituye en una forma 3D que se renderiza en múltiples imágenes mediante una o varias cámaras. Estas vistas del objeto deben captar el 100% de la superficie de ser posible (Qi et al., 2016). Luego, los atributos principales de las imágenes se extraen por cada vista generada con ayuda de la CNN. Estos atributos son procesados entre las diferentes vistas en la capa de *pooling* para finalmente pasar a la capa FC (*fully connected layer*). Por otro lado, las CNN's volumétricas codifican la representación de la forma en 3D en un tensor de valores binarios o reales (Qi et al., 2016).

Figura 3. Resultados de los experimentos realizados a las CNN's (Qi et al., 2016)

Las estadísticas iniciales entre ambas redes nos indican que un CNN volumétrico es 7.3% peor que un CNN multi-vistas. Luego de una serie de experimentos sobre el desempeño de ambas redes, desde un punto de vista arquitectónico y de resoluciones 3D, los autores proponen nuevas arquitecturas que superan a las anteriores bajo las mismas condiciones y tamaño de *input*.

3.8.1.3 Detección de múltiples objetos: "*Object Detection in 3D Scenes Using CNNs in Multi-view Images* (2016)"

En el trabajo propuesto por Qi C. R. se busca detectar múltiples objetos en una escena reconstruida en 3D. Para lograr ello se combinaron 2 componentes básicos que ya existían en trabajos de investigación previos: la detección de cuerpos en imágenes 2D y el cálculo de profundidades en imágenes. Para lograr lo propuesto se realizó un proceso (*pipeline*) que recibe

un video RGB y genera un mapa de calor. Este mapa de calor se divide en varios rangos espaciales pequeños donde cada uno muestra la probabilidad de que cierto objeto 15 se encuentre en dicho espacio. De esta forma, se logra representar la detección de múltiples objetos en una escena completa (Qi C. R. 2016).

La escena, en este caso, debe estar representada por una serie de imágenes de la misma. Primero se clasifican dichas imágenes obteniendo como resultado un cuadrilátero sobre la imagen que encierra el objeto clasificado. Esto se logra haciendo uso de las CNN's entrenados previamente con los objetos a clasificar. Además, el algoritmo también indica el score de precisión para usarlo posteriormente en el cálculo de las probabilidades.

Una vez se cuenta con todos los cuadros clasificados se recuperan las posiciones de las cámaras (donde fueron tomadas las imágenes) y las profundidades de los objetos en las imágenes (Qi C.R. 2016). Esto se logra haciendo uso de propuestas en trabajos de investigación previos. Al contar con los cuadriláteros y las profundidades se procede a proyectar las imágenes 2D a una cuadrícula de voxeles, donde cada profundidad de la imagen corresponde únicamente a un voxel (*volumetric pixel* / pixel volumétrico). Finalmente, se realizan los cálculos estadísticos: los pixeles al interior de un cuadrilátero son los píxeles pertenecientes a un objeto (teóricamente), por lo tanto, el voxel correspondiente es asignado con una probabilidad (basado en el score) que mide la presencia del objeto (Qi C. R. 2016). En este trabajo se prescindió de la colisión entre cuerpos.

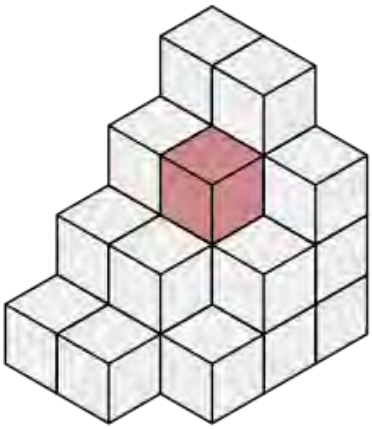


Figura 4. Conjunto de voxeles agrupados (M. W. Toews. para Wikipedia)

3.8.1.4 Segmentación de una escena por regiones: "*ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes* (2017)"

En el campo de segmentación de una escena, Dai A. et al. propusieron el uso de un *dataset* llamado ScanNet, que contiene cerca de 2.5 millones de vistas de 1513 escenas en videos RGB-D. Como mencionan los autores, existen muy pocos *datasets* que puedan ser usados en este campo para estimar posiciones 3D de cámaras, reconstrucción de superficies y 16 segmentaciones semánticas. ScanNet permitió (en el trabajo de investigación mencionado) la construcción de un sistema capaz de segmentar escenas con sus respectivas etiquetas (Dai et al., 2017).

En este trabajo de investigación se propone el flujo de un sistema dedicado a usuarios con nulo conocimiento sobre el tema. La utilidad principal del sistema es que el usuario pueda grabar una escena (*indoor scene*) y reciba como respuesta una construcción 3D de la escena grabada, pero con el valor agregado de la segmentación (refiriéndose a la clasificación de regiones pertenecientes a objetos y sus respectivas etiquetas). Por ejemplo, si en una escena de un cuarto encontramos una cama, una mesa y una silla, el *output* de dicho sistema debería ser la reconstrucción 3D del cuarto y con regiones indicando los objetos que pertenecen a dichas regiones, así como una región que pertenezca al espacio desocupado.

El modelo CNN que se emplea es entrenado con los 2.5 millones de imágenes pertenecientes a ScanNet y formas parciales de ShapeNet. Por cada voxel de la escena se guarda información sobre la clase de objeto al que pertenece (incluyendo espacio vacío). Luego se divide el espacio en volúmenes de 31 x 31 x 62 voxeles y son alineados con el plano del piso. Para ello, se debe verificar que la muestra entregada (escena) posee el 2% de su espacio lleno, de lo contrario es descartada. Luego se emplea el CNN entrenado con una posibilidad de clasificar entre 20 objetos más el espacio vacío. Para la clasificación se analiza una columna de voxeles y se compara con sus columnas más cercanas. El análisis previo a la clasificación es netamente geométrico (Dai et al., 2017). El resultado final es una clasificación de todas las superficies visibles de una escena en 3D bajo 20 posibles etiquetas de objetos, una herramienta que sin duda ayuda al entendimiento digital del espacio y puede ser usado para otras investigaciones.

3.8.1.5 Reconstrucción y síntesis geométrica en 3D: "*Learning Category-Specific Mesh Reconstruction from Image Collections* (2018)"

La reconstrucción geométrica en 3D se lleva a cabo mediante 3 tipos de métodos que se basan en diferentes conceptos. Estos métodos son:

- Métodos basados en modelos paramétricos cambiantes (*Parametric morphable Model-based*)
- Métodos basados en el aprendizaje de plantillas (*Part-based Template learning*)
- Métodos de aprendizaje profundo (*Deep learning methods*)

En este trabajo se expone nuevamente la reconstrucción 3D de un objeto, pero además se considera el agregado de texturas. El modelo puede funcionar con múltiples formas si se cuenta con la data suficiente para el entrenamiento. La idea que proponen los autores es 17 que a través de un modelo entrenado (empleando *deep learning*) sobre una colección de imágenes

(pertenecientes a un objeto que se desea reproducir) se parametriza una forma media aprendida. Dicha forma es representada por una malla deformable en 3D. Cuando se experimenta con una nueva imagen/instancia no perteneciente al conjunto de entrenamiento la forma 3D se genera a partir de la forma media aprendida y las deformaciones predichas por la instancia actual (Kanazawa, Tulsiani, Efros, & Malik, 2018). Dos problemas principales que presenta este enfoque son:

- Por cada objeto que se desea reproducir debe aprenderse una vasta colección de imágenes, por la tanto, si no se cuenta con la data será imposible realizar la reconstrucción. Las imágenes deben tener información adjunta sobre la expansión de puntos claves (*keypoints*) y las máscaras de segmentación, esto los hace más difíciles de obtener.
- Existe una malla deformable 3D por objeto aprendido. Una instancia muy distinta al resto generará predicciones con mayor cantidad de deformaciones, lo que implica mayor error cuadrático medio en la forma 3D generada.

Como fue mencionado, los autores también proponen la recuperación de las texturas y ligarlos a la forma reconstruida. La siguiente imagen muestra la perspectiva general del modelo construido, el cual se divide en 3 módulos que calculan la posición de la cámara relativa al objeto, la deformación con respecto a la malla media aprendida y las texturas.

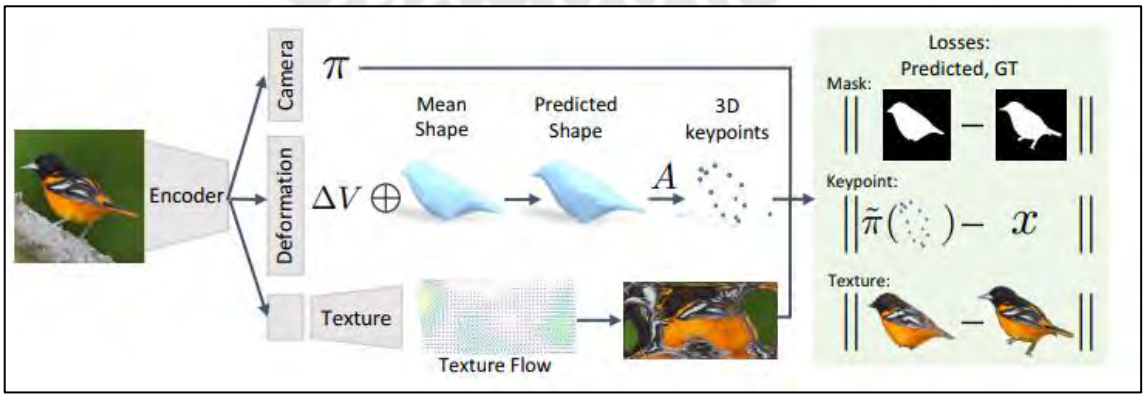


Figura 5. Representación de los módulos y flujo (Kanazawa et al., 2018)

3.8.2 ¿Cuáles son las técnicas que se utilizan para representar un objeto 3D en el computador?

Entre los artículos revisados encontramos las siguientes técnicas de representación: imágenes de vista múltiple, representación volumétrica, nube de puntos y red de polígonos.

3.8.2.1 Imágenes de vista múltiple: "*Multi-view Convolutional Neural networks for 3D Shape Recognition* (2015)"

En este trabajo se propone el reconocimiento de una superficie a partir de 12 imágenes que capten las vistas 360° alrededor del objeto. Las imágenes se procesan mediante una CNN para extraer las principales características del objeto. Se simplifican las vistas mediante la operación de "*pooling*" para que sean comparadas entre ellas y procesadas mediante un segundo CNN. El *output* final del segundo CNN será la predicción de la clasificación del objeto (Su, Maji, Kalogerakis, & Learned-Miller, 2015). Este trabajo en particular emplea la representación de un objeto en 3D mediante el uso de múltiples vistas en diferentes imágenes. Estas imágenes se compactan en un único contenedor llamado de*scriptor* de forma (*shape descriptor*). De esta forma, la información sobre las múltiples vistas de un mismo objeto permiten su clasificación.

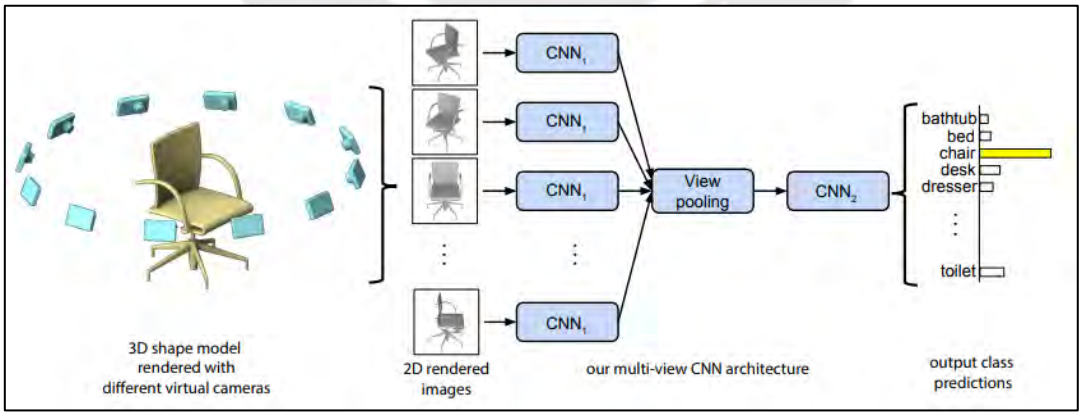


Figura 6. *Pipeline* para el uso de CNN's y representación multi-vistas de un objeto (Su et al., 2015)

3.8.2.2 Volumétrico: "*3D ShapeNets: A Deep Representation for Volumetric Shapes* (2015)"

En este caso se aprovechan los mapas de profundidad (*depth maps*) generados por sensores de profundidad 2.5D para la representación de objetos del mundo real en el computador. Para ello, se construye la representación como una distribución de un conjunto de variables binarias. Lo que se conoce como representación volumétrica. Estas variables se almacenan en una cuadrícula de voxeles como se muestra en el 3° paso de la imagen inferior y almacenan la probabilidad de que cierto objeto se encuentre en dicho punto en el espacio.

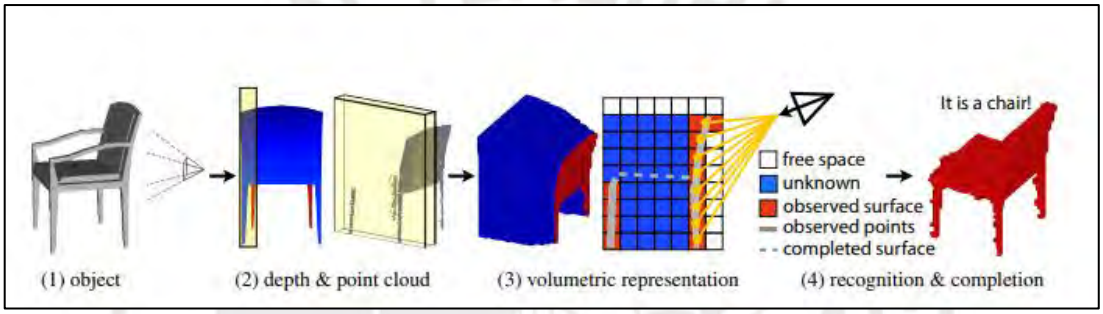


Figura 7. Flujo de un objeto en el mundo real hasta su representación volumétrica (Zhirong Wu et al., 2015)

Los autores introducen 3D-ShapeNets, un modelo CNN que aprende la distribución del espacio que ocupa la forma y también soporta una sugerencia para el autocompletado de mapas de profundidad incompletos (Zhirong Wu et al., 2015). Lo resaltante de este trabajo es la forma en cómo se representan las formas 3D digitalmente. Como podemos apreciar en la figura, se consigue una cuadrícula consistente de voxeles con las probabilidades asociadas para regiones visibles de la superficie, así como las regiones desconocidas.

3.8.2.3 Nube de puntos: "*VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection* (2017)"

En este proyecto se propone el modelo VoxelNet, una red de detección 3D basado en un modelo de *deep learning* capaz de predecir/clasificar objetos, recuperar las características

principales y unirlas con la generación de un recuadro alrededor de la imagen en la predicción. Además, se demuestra que VoxelNet tiene un mejor desempeño que algunos sistemas LiDAR propuestos anteriormente (Y. Zhou & O. Tuzel, 2018). La representación 3D se da mediante una nube de puntos. Por cada conjunto de puntos perteneciente al rango de un voxel se detecta una característica principal (conocido como *feature*). De esta forma la nube de puntos se convierte en una representación volumétrica de*scripti*va (Y. Zhou & O. Tuzel, 2018).

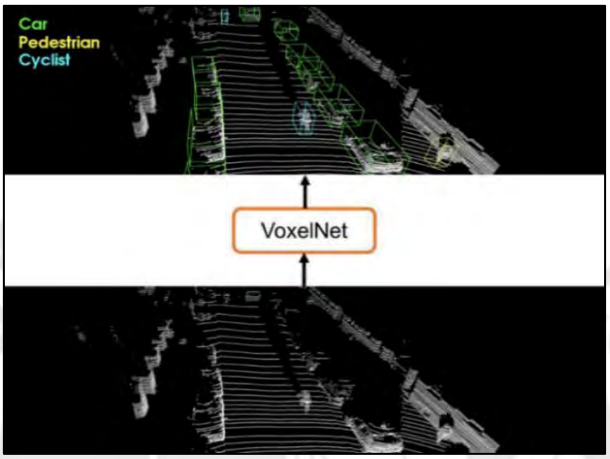


Figura 8. *Input* y *Output* de VoxelNet. Nube de puntos sin procesar (Y. Zhou & O. Tuzel, 2018)

Recuperar la superficie a partir de una nube de puntos requiere de aprendizaje de máquina debido a que debemos asociar dichos puntos a una superficie o forma. Los puntos o conjuntos de coordenadas X-Y-Z en realidad carecen de sentido alguno si no se procesan o explotan (Y. Zhou & O. Tuzel, 2018). Sin embargo, visualizar una nube de puntos podría darnos una idea rudimentaria de la densidad del objeto y su complejidad de reconstrucción. La captura de los puntos también se realiza mediante un proceso de escaneo sobre el objeto físico (Y. Zhou & O. Tuzel, 2018).

3.8.2.4 Malla poligonal: "*Variational Autoencoders for Deforming 3D Mesh Models* (CVPR 2018)"

En este trabajo los autores resaltan la importancia de una representación de modelos 3D como mallas deformables. Este tipo de representación permite la modificación del objeto (en forma) luego de ser reconstruido. Esta variabilidad apoya algunas aplicaciones en medicina, así como animaciones y entretenimiento (Q. Tan, L. Gao, Y. Lai, & S. Xia, 2018).

Una malla como representación 3D se compone de polígonos adheridos que imitan la superficie del objeto. Estos polígonos generalmente son triángulos. La ventaja es que gracias a esta representación de polígonos todas las regiones de la superficie son convexas, por lo tanto, no existen regiones no visibles desde las múltiples vistas.

En el trabajo de investigación se propone un *framework* que permite el modelamiento de probabilidades de la presencia de cuerpos en un ambiente 3D representado por mallas deformables. Los autores lo denominaron modelo de malla VAE (*Variational AutoEncoder*) (Q. Tan et al., 2018). Se trata de una red fácil de entrenar ya que requiere de mínima data como *input* de formas deformables. Sin embargo, la limitación es que solo se pueden procesar mallas homogéneas.

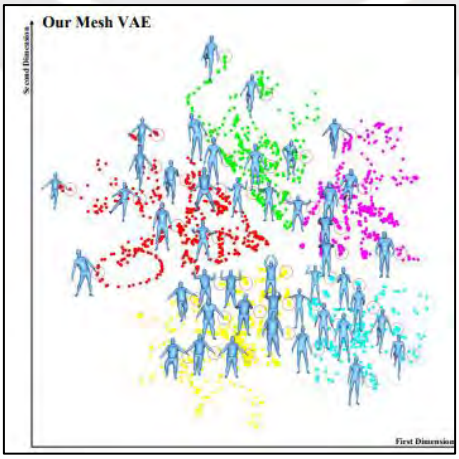


Figura 9. Clasificación de poses humanas en Dyna *dataset* (Q. Tan et al., 2018)

3.9 Conclusiones

En conclusión, existen varias formas de representar objetos 3D en el computador. Cada representación implica un procesamiento diferente y dependen fuertemente de las herramientas empleadas durante el proceso de escaneo 3D de los objetos. Además, encontramos que algunos tipos de representaciones como las mallas de polígonos y nube de puntos pueden generarse a partir de redes neuronales entrenadas en una extensa colección de imágenes sobre los propios objetos. Estos conjuntos de datos iniciales y su preprocesamiento son cruciales para la obtención de resultados. Debido a que existe una dependencia importante con los datos de entrenamiento para la obtención de resultados, los autores buscan integrar la mayor cantidad de datos posibles de distintas fuentes o *datasets* con el fin de mejorar la calidad de sus reconstrucciones.

También existen proyectos que emplean técnicas de reconstrucción con múltiples enfoques o propósitos distintos como la estimación de la posición 3D de un objeto a partir de una única imagen, la clasificación de uno o varios objetos en una escena, la segmentación de una escena por regiones o la reconstrucción y síntesis geométrica 3D partiendo de una colección de imágenes. Sin embargo, son pocos trabajos los que se dedican específicamente al escaneo de superficies altamente variables debido a la dificultad que implica el reconocimiento de tales superficies. En el estado del arte explorado encontramos métodos de reconstrucción 3D pasivos que permiten capturar las superficies de los objetos sin interactuar físicamente con los mismos. Esto se logra mediante cámaras fotográficas que generan una gran colección de imágenes las cuales son procesadas por redes neuronales.

Capítulo 4. Preprocesamiento de datos y entrenamiento de un modelo de aprendizaje de máquina

4.1 Introducción

En el presente capítulo se describe el proceso desarrollado asociado al objetivo específico número uno: “Preprocesar el conjunto de datos de piezas arqueológicas para el entrenamiento y prueba de un modelo de aprendizaje de máquina”. Para lograr nuestro cometido partiremos de un set inicial de datos compuesto por 962 piezas arqueológicas escaneadas en 3D mediante algún método convencional. Estos objetos 3D fueron proporcionados por el grupo IA-PUCP o Inteligencia Artificial PUCP y se encuentran representados en el computador como archivos OBJ.

Para preprocesar este conjunto de datos procederemos primero a generar grandes cantidades de imágenes a partir de los huacos peruanos. Estas imágenes deben seguir cierto patrón y estándares los cuales se definen en la siguiente sección. Para ello se utiliza la herramienta Stanford ShapeNet Renderer la cual se modifica a nivel de código con el fin de generar las imágenes de acuerdo con nuestras necesidades. Estas imágenes posteriormente serán procesadas por un modelo de aprendizaje de máquina, es por ello la necesidad de generarlas bajo ciertos parámetros y un orden.

En el segundo resultado se procede a automatizar la tarea de segmentación de datos, en este caso, de las imágenes creadas. Para ello primero introducimos el proyecto PMO o *Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction*. Dicho proyecto nos permite reconstruir objetos 3D en el computador a partir de videograbaciones mediante modelos de aprendizaje de máquina. Sin embargo, las categorías predefinidas de dicho proyecto son: sillones, carros, aviones, etc. Es ahí donde tenemos que agregar nuestra categoría

de piezas arqueológicas, tarea la cual requiere de múltiples modificaciones en los archivos y carpetas del proyecto. Esta tarea se automatizó mediante un *script* desarrollado en Python.

Tercero, se genera la data faltante para entrenar un nuevo modelo de aprendizaje de máquina. Esta data requerida se trata de nubes de puntos (archivos PLY) y arreglos NumPy (archivos NPY). Además, es necesario todo un proceso de normalización de los datos mejor detallado en el resultado esperado n°3. Este proceso modifica los puntos y reescribe los archivos PLY múltiples veces para que puedan ser procesados posteriormente. Al tener toda la data lista se procede con el entrenamiento en huacos peruanos de un modelo de aprendizaje de máquina. Finalmente se prueban unos primeros resultados de reconstrucciones 3D de piezas arqueológicas utilizando el modelo entrenado y secuencias de imágenes compuestas por piezas arqueológicas y fondos panorámicos aleatorios. Durante todo el desarrollo de este objetivo se utilizó como recurso el servidor de IA-PUCP.

4.2 Resultados Alcanzados

4.2.1 Conjunto de datos generado: renderizado de secuencias

4.2.1.1 Descripción

En esta sección se describirá el desarrollo del resultado esperado n°1: “Conjunto de datos generado: renderizado de secuencias”. Este resultado consiste en generar una extensa cantidad de imágenes a partir de un conjunto inicial de huacos peruanos en formato OBJ (formato para la representación de objetos 3D) proporcionados por el grupo IA-PUCP (Inteligencia Artificial de la Pontificia Universidad Católica del Perú). Estas imágenes deben estar organizadas en secuencias las cuales se generarán 1 por cada objeto. Al proceso de generación de imágenes se le conoce como “renderización”, término al cual haré referencia posteriormente. Para generar las imágenes se empleó la herramienta Stanford Shapenet Renderer (Chang et al., 2015) la cual se modificó a nivel de código para lograr nuestro cometido. Además, existen múltiples

parámetros modificables como el número de vistas/imágenes a generar por cada secuencia, la escala, la resolución, etc. los cuales se han ido manipulando en el proceso.

4.2.1.2 Métodos y procedimientos empleados

Tabla 10. Herramientas para el RE1 (tomado del Capítulo 1)

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos
R1. Conjunto de datos generado: renderizado de secuencias	Generación de una vasta cantidad de secuencias de imágenes a partir de los huacos peruanos pre escaneados, siguiendo un patrón de movimiento definido y modificando el código de la herramienta a utilizar.	• Stanford Shapenet Renderer • Python • PuTTY

El conjunto inicial de datos (archivos OBJ) está conformado por 962 huacos peruanos los cuales se encuentran almacenados en el servidor del grupo IA-PUCP al cual se accede remotamente mediante la herramienta PuTTY. Dicho servidor utiliza el sistema operativo Ubuntu 18.04.4 LTS por lo cual todos los comandos empleados posteriormente seguirán dicho contexto. Cada huaco corresponde con un único archivo OBJ, MTL y JPG (generados al momento del escaneo con algún método tradicional). A continuación, en la siguiente tabla se detallan las extensiones involucradas y en la siguiente imagen se muestra el contenido (recortado) del repositorio de 962 huacos peruanos proporcionado por IA-PUCP .

Tabla 11. Extensiones de archivo del conjunto inicial

Extensión del archivo	Nombre completo de la extensión	Descripción
OBJ	*Wavefront 3D Object File*	La extensión OBJ es un estándar para representar objetos tridimensionales en el computador. Desarrollado por la compañía *WaveFront Technologies*, el archivo contiene información explícita sobre los elementos que componen un objeto 3D: puntos, caras, texturas, etc.
JPG	*Joint Photographic Experts Group*	Estas imágenes contienen las texturas de las piezas arqueológicas, es decir, los colores y patrones asociados a la superficie del objeto.

MTL	*Material Template Library*	Contienen información sobre cómo deben aplicarse las texturas al objeto. Dentro de este archivo encontramos una referencia la archivo JPG y las coordenadas 3D de dónde aplicar las texturas.

```
esumoso@enterprise:/data/esumoso$ ls
HuacosFinal  models
esumoso@enterprise:/data/esumoso$ cd HuacosFinal
esumoso@enterprise:/data/esumoso/HuacosFinal$ ls
map.json  models  thumbnails
esumoso@enterprise:/data/esumoso/HuacosFinal$ cd models
esumoso@enterprise:/data/esumoso/HuacosFinal/models$ ls
0001.obj      0284.obj      0527.obj      0747.obj
0001.obj.jpg  0284.obj.jpg  0527.obj.jpg  0747.obj.jpg
0001.obj.mtl  0284.obj.mtl  0527.obj.mtl  0747.obj.mtl
0002.obj      0285.obj      0528.obj      0748.obj
0002.obj.jpg  0285.obj.jpg  0528.obj_1.png  0748.obj_1.png
0002.obj.mtl  0285.obj.mtl  0528.obj.jpg  0748.obj.jpg
0003.obj      0286.obj      0528.obj.mtl  0748.obj.mtl
0003.obj.jpg  0286.obj.jpg  0529.obj      0749.obj
0003.obj.mtl  0286.obj.mtl  0529.obj_1.png  0749.obj_1.png
0004.obj      0287.obj      0529.obj_2.png  0749.obj.jpg
0004.obj.jpg  0287.obj.jpg  0529.obj.jpg  0749.obj.mtl
0004.obj.mtl  0287.obj.mtl  0529.obj.mtl  0750.obj
0005.obj      0288.obj      0530.obj      0750.obj_1.png
```

Figura 10. Conjunto inicial de huacos peruanos (recortado)

Para renderizar las imágenes se emplea la herramienta *open source* Stanford Shapenet Renderer la cual recibe como *input* un archivo OBJ y genera múltiples imágenes que enfocan y rodean al objeto siguiendo una trayectoria circular. Dicho componente recibe los siguientes parámetros (observar la siguiente tabla). Además, se detallan los valores asignados a cada parámetro por medio de múltiples pruebas ensayo-error.

Tabla 12. Valores de parámetros de renderización

Parámetro	**Tipo de dato**	**Descripción**	**Valor asignado**
views	*Integer*	Imágenes o vistas a generar por archivo OBJ	72
scale	*Float*	Escala del modelo	0.006 083
remove_doubles	*Bool*	Remover vértices dobles para mejorar la calidad de la malla de polígonos	True

edge_split	*Bool*	Agregar un filtro de división de bordes	True
depth_scale	*Float*	Escala de profundidad	1.4
color_depth	*String*	Cantidad de bits por canal para los colores (los únicos valores permitidos son “8” y “16”)	“8”

A continuación, veamos un ejemplo del *output* de la herramienta utilizada. En este caso se han generado 30 vistas que rodean a una silla. Es decir, el parámetro *views* tiene un valor de 30, por ende, la rotación por imagen es de 12 grados sexagesimales. Para entender cómo calcular la rotación de la cámara aplicamos la siguiente fórmula simple.

Rotación por imagen = 360° / parámetro “views”

En este ejemplo tenemos:

Rotación por imagen = 360° / 30 = 12°

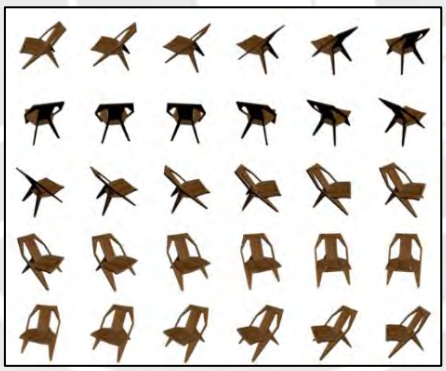


Figura 11. *Output* ejemplo del componente Stanford Shapenet Renderer (Chang et al., 2015)

Sin embargo, nuestras imágenes de huacos peruanos no pueden seguir dicho patrón exactamente. Necesitamos cambiar el patrón de rotación por motivos que descubriremos más adelante cuando introduzcamos el proyecto base sobre el cual se está trabajando, denominado *Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction* (Lin et al., s. f.) o simplemente PMO. De momento basta resaltar que estas modificaciones son parte de los estándares requeridos para la integración de nuevos conjuntos de datos (en este caso huacos

peruanos) al proyecto mencionado. Esta modificación consiste en establecer la generación de 72 imágenes por archivo OBJ, establecer la resolución de la cámara en 224 x 224 pixeles y realizar 1 trayectoria circular alrededor del objeto por cada 24 imágenes (es decir, 3 circunferencias en total) cambiando el valor del eje z con cada círculo. La siguiente ilustración nos muestra la escena propuesta, donde el objeto 3D se encuentra posicionado en el punto (0,0,0) y la trayectoria de la cámara son las 3 circunferencias paralelas. Las imágenes son capturadas enfocando al objeto desde las circunferencias con un total de 24 imágenes uniformemente espaciadas a lo largo de cada círculo.

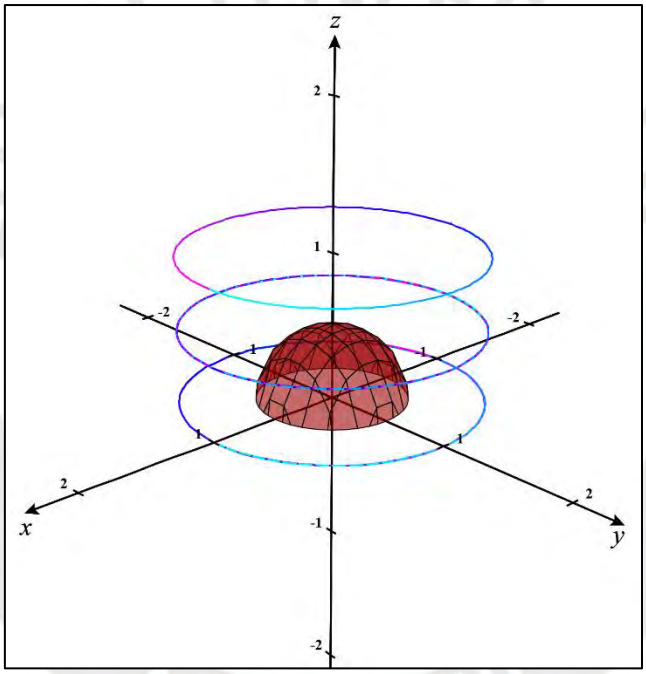


Figura 12. Trayectoria de la cámara (elaboración propia)

Otro detalle para considerar es el tamaño del objeto el cual debemos escalar para que se ajuste a la trayectoria establecida y se mantenga al interior de los aros. Cada huaco tiene un tamaño diferente, es por ello que mediante pruebas con diferentes escalas encontramos un valor promedio que mantenga a los 962 objetos al interior de los aros. Otros *outputs* generados por la herramienta son las diferentes versiones de cada imagen las cuales se describen a continuación.

- *Albedo*: misma imagen, pero sin considerar los valores de iluminación, es decir, la totalidad de la superficie se ve iluminada al 100%.
- *Depth*: En cada punto del objeto se guarda también la distancia de la cámara al objeto (profundidad).
- *Normal*: En cada punto del objeto se guarda también el vector normal a la superficie.

Sin embargo, en este caso no se requieren de las versiones *depth* ni *normal*. En cuanto a la versión *albedo*, inicialmente se incluyeron en la generación de imágenes y se probaron en las siguientes etapas del presente trabajo de investigación obteniendo resultados no satisfactorios. Es por ello que deben ser descartadas en esta etapa. A continuación, se listan todos los cambios realizados a nivel de código en la herramienta Stanford Shapenet Renderer:

- Eliminar la generación de imágenes en versión *albedo*, *depth* y *normal* (quedarse solo con la versión original RGB).
- Modificar los parámetros de resolución (224 x 224) de las imágenes y vistas por secuencia (72).
- Modificar la trayectoria de la cámara siguiendo las reglas mencionadas anteriormente.

Para un análisis más a profundidad podemos encontrar el código modificado de la herramienta *open source* en el Anexo B. Con los parámetros definidos y modificaciones se aplica el siguiente comando en el servidor IA-PUCP el cual busca todos los archivos con extensión OBJ dentro de la carpeta correspondiente y ejecuta el componente "*render_blender.py*" por cada una de ellas. Luego de 1 hora y 12 minutos de procesamiento el programa generó las 72 imágenes por cada una de las 962 piezas arqueológicas. Es decir, finalmente obtuvimos 69 264 imágenes en formato PNG los cuales serán procesados posteriormente por un modelo de aprendizaje de máquina.

find /data/esumoso/HuacosFinal/models -name '.obj' -print0 | xargs -0 -n1 -P3 -I {} blender --background --python render_blender.py -- --output_folder /data/esumoso/renderings {} --scale=0.006083*

4.2.1.3 Medio de verificación e IOV

Tabla 13. Medio de verificación e IOV del RE1 (tomado del Capítulo 1)

Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R1. Conjunto de datos generado: renderizado de secuencias	Carpeta que contiene las imágenes generadas a partir de las piezas arqueológicas	100% de piezas arqueológicas procesadas exitosamente

Para demostrar la consecución del resultado esperado nos ubicamos en la carpeta donde se generaron las 69 264 imágenes y ejecutamos los siguientes comandos:

ls

ls | wc –l

El primero nos lista todos los archivos dentro de la carpeta lo cual nos muestra una extensa lista de imágenes en formato PNG (véase la siguiente figura). Mientras que el segundo comando nos calcula la cantidad de archivos dentro de la carpeta actual. Dado el procesamiento exitoso de todas las piezas arqueológicas y la verificación de la existencia de una carpeta que contiene toda la data generada podemos afirmar que se cumplió con el IOV planificado.

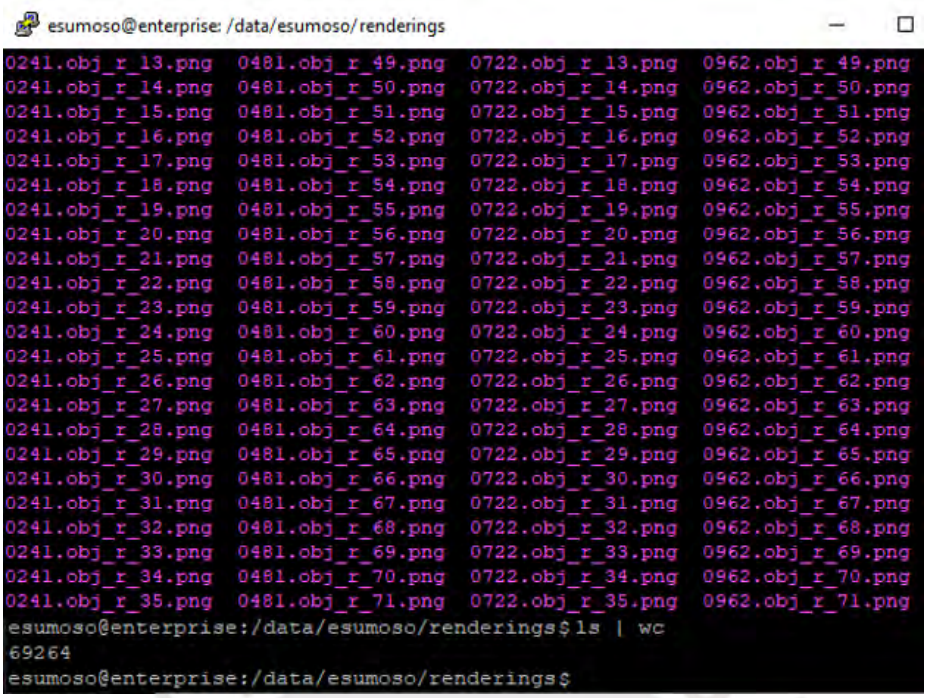


Figura 13. Imágenes generadas (contenido recortado)

4.2.2 Datos segmentados en 2 subconjuntos: entrenamiento y prueba

4.2.2.1 Descripción

En esta sección se describirá el resultado esperado número 2: “Datos segmentados en 2 subconjuntos: entrenamiento y prueba”. Habiendo obtenido el extenso conjunto de imágenes en el resultado previo, ahora se pretende segmentar la data definiendo así los subconjuntos de entrenamiento y de prueba. Esta segmentación es necesaria para proceder con el entrenamiento del modelo de aprendizaje de máquina el cual trabajaremos en el siguiente resultado esperado. En esta sección se introduce también el proyecto base denominado *Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction* (Lin et al., s. f.) al cual llamaremos PMO en adelante. Dicho proyecto se enfoca en la reconstrucción de objetos 3D a partir de videograbaciones sobre algunos conjuntos de datos predefinidos. En esta tesis se pretende integrar un nuevo conjunto de datos (piezas arqueológicas) al proyecto PMO para lo cual serán necesarios múltiples procesos de normalización de nuestra data, incluyendo la segmentación.

En esta sección automatizamos el proceso de segmentación mediante un *script* en Python el cual a su vez modifica múltiples archivos del PMO como parte de la integración del nuevo conjunto de datos al proyecto.

4.2.2.2 Métodos y procedimientos empleados

Tabla 14. Herramientas para el RE2 (tomado del Capítulo 1)

Resultado Esperado	**Actividad para la obtención del resultado**	**Herramientas y/o métodos**
R2. Datos segmentados en 2 subconjuntos: entrenamiento y prueba	Inicialización del proyecto base PMO y desarrollo de un *script* en Python para la automatización de la segmentación de un conjunto de datos, modificando/creando los archivos LIST correspondientes y definiendo una distribución de 80% - 20%.	• PMO • Python • Sublime Text 3 • PuTTY

Antes de segmentar la data debemos entender cómo el proyecto base PMO organiza sus conjuntos de datos predefinidos y cómo las segmenta. Para ello clonamos el repositorio del proyecto y nos ubicamos en la carpeta "*data*" (véase la figura). Dentro de esta carpeta encontramos 3 *script*s que nos permiten descargar: las imágenes de fondos panorámicos, las imágenes de los objetos 3D y las secuencias (composición de ambos), sumando un total de 189 GB. Luego de varias horas de descarga finalmente podemos descomprimir y ubicar la data en las carpetas "*background*", "*rendering*" y "*sequences*" respectivamente. Dicho contenido pertenece a los conjuntos de datos predefinidos del PMO, por ejemplo: aviones, carros, sillas, etc. cuyo contenido será detallado a profundidad posteriormente.

Figura 14. Clonación del repositorio y descarga de data

En esta sección nos compete revisar el archivo "*categoriest.txt*" cuyo contenido se muestra en la siguiente imagen. Como podemos observar, a cada categoría de objetos se le asigna un código único y un nombre. Por ejemplo, para la categoría de aviones el código encontrado es 02 691 156 y el nombre es "*plane*". De manera análoga nuestro *script* de segmentación de datos deberá añadir una nueva categoría al archivo cuyo código y nombre serán 70 309 649 y "*archaeology*" respectivamente.

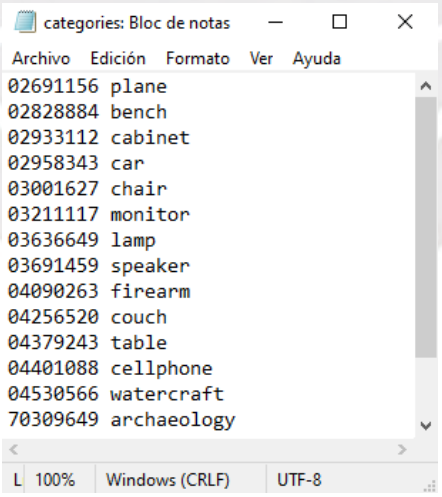


Figura 15. Archivo "*categories.txt*" donde se listan las categorías del PMO

Para entender cómo se organizan las imágenes de cada categoría nos ubicaremos en la carpeta "*rendering*" donde encontraremos varias subcarpetas comprimidas. Los nombres de

dichas carpetas corresponden a los códigos asignados a las categorías. Por ejemplo, la carpeta comprimida “02691156.tar.gz” corresponde a la categoría de aviones y contiene todas las imágenes de estos objetos. Luego de descomprimir las carpetas nos ubicamos en cualquiera de ellas (“02691156” por ejemplo) y observamos que contiene múltiples subcarpetas con nomenclatura de largas cadenas de caracteres alfanuméricos. Dentro de cada subcarpeta encontraremos 72 imágenes PNG nominadas del “0” al “71”. Se recomienda ver la siguiente imagen para entender la jerarquía. Podemos afirmar entonces que cada objeto de la categoría corresponde a una única secuencia de imágenes, por lo tanto, la cadena de caracteres alfanuméricos podría interpretarse como el identificador único del objeto 3D.

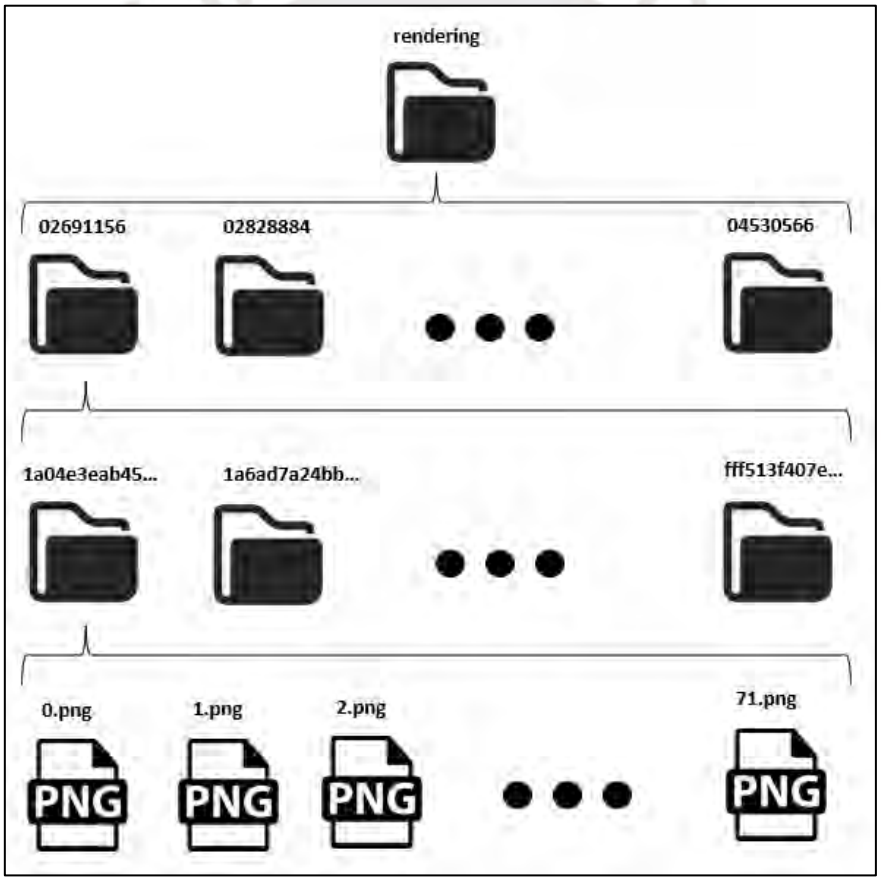


Figura 16. Jerarquía de los conjuntos predefinidos en el PMO (elaboración propia)

Todos los objetos se encuentran registrados en archivos de listas (extensión LIST) dentro de la carpeta “*lists*” ubicada en “*data*”. En estas listas se lleva a cabo la segmentación de cada

categoría en 2 subconjuntos: entrenamiento y prueba. Es decir, la segmentación de datos se realiza empleando listas, mientras que los datos reales permanecen intactos dentro de la jerarquía que se explicó en párrafos anteriores. A cada categoría le corresponde 2 archivos de listas, por ejemplo, para la categoría de aviones las listas son los archivos: "*02691156_train.list*" y "*02691156_test.list*". El primero contiene los nombres (cadenas únicas de caracteres alfanuméricos) de todos los objetos pertenecientes al subconjunto de entrenamiento, mientras que el segundo archivo contiene todos los nombres de los objetos pertenecientes al subconjunto de prueba. Esto quiere decir que para integrar y segmentar una nueva categoría al PMO tenemos que crear 2 archivos LIST correspondientes.

Finalmente también encontramos 3 listas independientes: "*indoor_test.list*", "*indoor_train.list*" y "*all_test.list*". Las 2 primeras almacenan los nombres de los fondos panorámicos y sus identificadores únicos (Ej.: "*bathroom abmyjxdzsvhsiz*"), la diferencia está en que la primera lista contiene los fondos usados para subconjuntos de prueba y la segunda los fondos usados para subconjuntos de entrenamiento. Por otro lado, la tercera lista en mención agrupa todos los subconjuntos de prueba de todas las categorías asociando cada objeto con su código de categoría. (Ej.: "*02691156 d172705764e25e20884a857d19f7439f*"). La siguiente figura muestra algunos elementos de las listas.

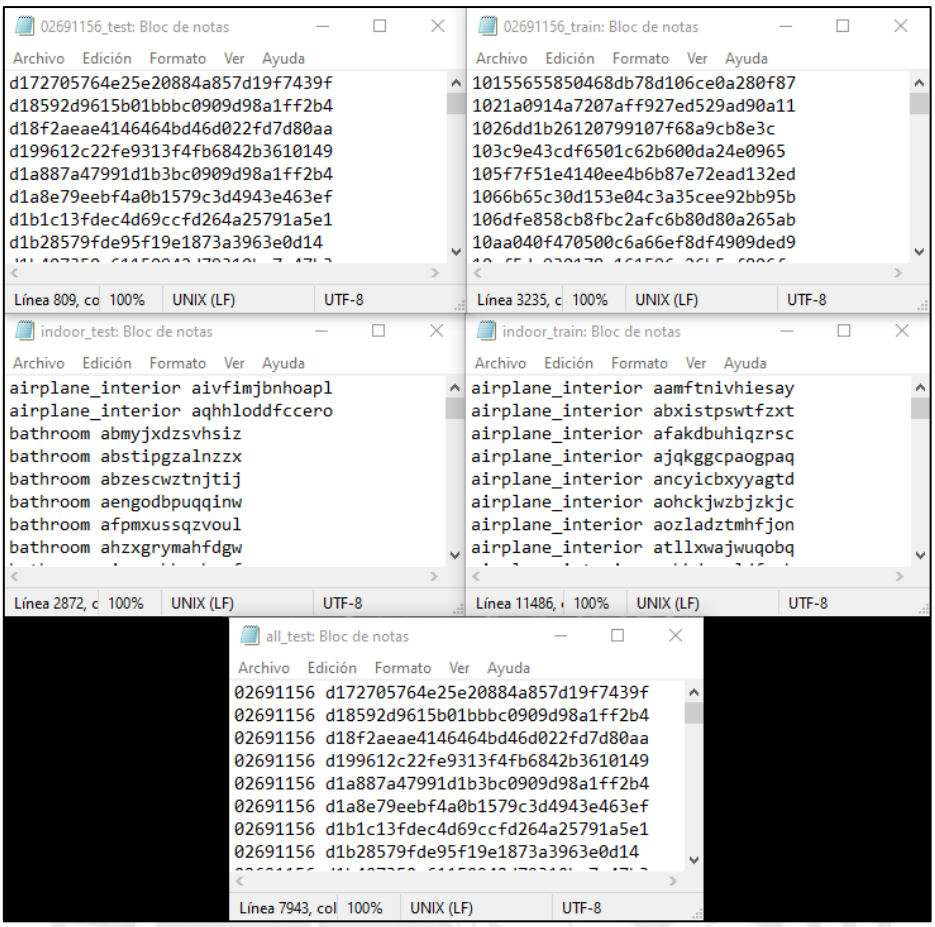


Figura 17. Contenido de archivos LIST

Habiendo entendido todo ello, se desarrolló un *script* en Python mediante el editor de textos Sublime Text 3. La edición y creación de los próximos *script*s también se llevarán a cabo en el editor mencionado. Este nuevo *script* recibe como parámetros: el nombre de la categoría que se quiere crear, el código único asignado a dicha categoría, la cantidad de objetos en total de la categoría, la ruta al archivo "*categories.txt*" del PMO y la ruta a la carpeta "*lists*" también del PMO. Con estos 5 parámetros seremos capaces de realizar los siguientes cambios en el proyecto base: modificar el archivo "*categories.txt*" añadiendo la nueva categoría (en este caso "*70309649 archaeology*") y crear 2 nuevas listas: "*70309649_train.list*" y "*70309649_test.list*" con los subconjuntos correspondientes además de modificar la lista existente "*all_test.list*".

Para la distribución de subconjuntos se siguió un estándar del 80% - 20% (entrenamiento - prueba) análogo al resto de categorías ya existentes. El *script* desarrollado calcula las

cantidades exactas correspondientes a cada subconjunto. En este caso contamos con 962 piezas arqueológicas:

- Conjunto de entrenamiento: 962 * 80% = 769.6 → 770 piezas
- Conjunto de prueba: 962 * 20% = 192.4 → 192 piezas

Para nombrar nuestros objetos simplificamos la cadena de caracteres alfanuméricos por una secuencia simple del "0001" al "0962". Acto seguido creamos la lista "*70309649_test.list*" y le añadimos aleatoriamente 192 objetos. El resto de objetos se anexan a la lista "*70309649_train.list*" y se guardan ambos archivos en la ruta ingresada como parámetro. Luego se modifica la lista general "*all_test.list*" para añadir el conjunto de prueba que acabamos de crear y finalmente se imprime en consola la distribución de objetos. Para un mayor detalle del código, el *script* desarrollado "*write_seq.py*" se encuentra en el Anexo C.

4.2.2.3 Medio de verificación e IOV

Tabla 15. Medio de verificación e IOV del RE2 (tomado del Capítulo 1)

Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R2. Datos segmentados en 2 subconjuntos: entrenamiento y prueba	• Archivos LIST del proyecto base PMO creados y modificados • *Script* desarrollado para la automatización de la tarea	100% de objetos segmentados y organizados en 2 archivos LIST independientes.

Para verificar el éxito de la segmentación de datos nos ubicamos en la carpeta de listas del proyecto base PMO y ejecutamos los siguientes comandos:

ls

wc -l 70309649_test.list

wc -l 70309649_train.list

El primer comando lista todos los archivos ubicados en la carpeta actual donde podemos identificar 2 nuevas listas pertenecientes a la categoría de piezas arqueológicas. Con los siguientes 2 comandos calculamos la cantidad de elementos de dichas listas nuevas. Los *outputs* de los comandos se pueden apreciar en la siguiente figura.

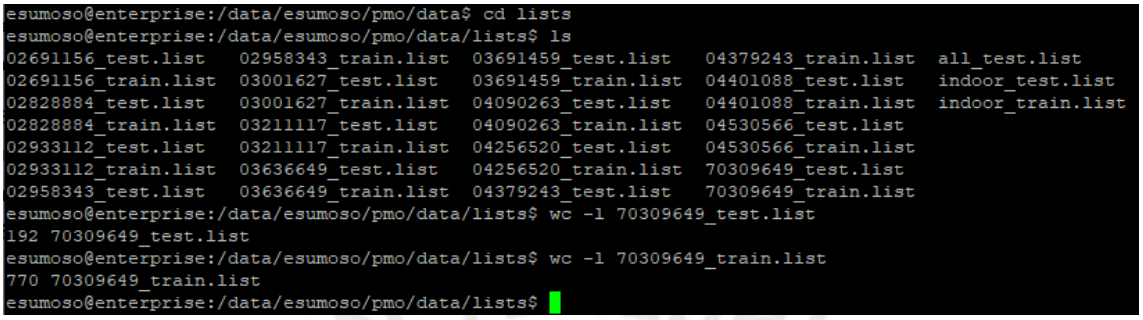


Figura 18. Listas creadas para la segmentación de datos

Esto evidencia que la lista del conjunto de prueba contiene 192 elementos mientras que la lista de entrenamiento contiene 770. Por lo tanto, se concluye que se han segmentado exitosamente los 962 objetos (100%) y ahora estos están organizados en 2 archivos LIST independientes siguiendo una distribución de 80% - 20%. Asimismo, encontramos el *script* desarrollado para la automatización de la tarea en el Anexo C.

4.2.3 Modelo entrenado con los datos pre-procesados

4.2.3.1 Descripción

En esta sección se describe el resultado esperado número 3: “Modelo entrenado con los datos pre-procesados”. Este resultado consiste en entrenar un modelo de aprendizaje de máquina utilizando el proyecto base PMO. Para ello es necesario integrar al proyecto las imágenes producidas en el primer resultado esperado y generar un nuevo set de datos conformado por nubes de puntos (archivos PLY y NPY). Estos archivos o datos deben ser normalizados mediante algunos procesos plasmados en los *script*s desarrollados durante la presente sección. Luego de integrar las imágenes y los archivos PLY y NPY al proyecto base,

sumado a la segmentación de datos realizada en el segundo resultado esperado, podemos ejecutar el comando de entrenamiento de un nuevo modelo de aprendizaje de máquina.

4.2.3.2 Métodos y procedimientos empleados

Tabla 16. Herramientas para el RE3 (tomado del Capítulo 1)

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos
R3. Modelo entrenado con los datos preprocesados	Generación de las nubes de puntos o archivos PLY y NPY a partir de los huacos peruanos. Desarrollo de *script*s para la normalización, procesamiento e integración de la data al proyecto base PMO. Entrenamiento de un nuevo modelo de aprendizaje de máquina con la obtención de unos primeros resultados cualitativos.	• PMO • Python • Sublime Text 3 • PuTTY

Para poder entrenar un modelo con un nuevo set de datos necesitamos cumplir 3 requisitos:

- Contar con 1 extenso set de imágenes correspondientes a los objetos 3D
- Contar con los archivos PLY y NPY (cada par de archivos representa 1 objeto 3D)
- Crear/modificar los archivos LIST para la adición de una nueva categoría y su segmentación

Este último requisito ya lo tenemos cubierto en el resultado anterior. El primer requisito lo tenemos parcialmente cubierto puesto que hemos generado las imágenes, pero aún nos falta organizarlas de acuerdo con la jerarquía de secuencias detallada en la sección anterior e ilustrada en la figura 16. Revisando nuevamente las imágenes generadas encontramos que la nomenclatura de estas es la siguiente "*[#####].obj_r_[##].png*" (Ejemplo: "*0152.obj_r_71*"), donde el primer número es una cadena de caracteres numéricos con valores del "0001" al "0962", mientras que el segundo número es una cadena similar pero con el rango de "00" al "71". Podemos encontrar más ejemplos en la figura 13. Entendemos pues que el primer número es el identificador único de la pieza arqueológica, mientras que el segundo es el número de

imagen, siendo un total de 72 imágenes por pieza. Para organizar este set a la jerarquía establecida del PMO se desarrolló el siguiente *script* simple.

```
import glob, os
strings = []
total_objects = 962 # <=9999

# crear arreglo de strings: "0001"-"0962"
for i in range(total_objects):
    strings.append(str(i + 1).zfill(4))

# crear 1 carpeta por cada secuencia de imágenes
# y mover las imágenes correspondientes a cada carpeta
for i in strings:
    os.system("mkdir " + i)
    os.system("mv *" + i + "* " + i + " 2>/dev/null")

# entrar a cada carpeta y cambiar los nombres de las imágenes: "0"-"71"
for i in strings:
    os.chdir(i)
    file_names = glob.glob("*.png")
    file_names.sort()
    counter = 0
    for j in file_names:
        os.system("mv " + j + " " + str(counter) + ".png 2>/dev/null")
        counter += 1
    os.chdir("..")

```

Figura 19. *Script* para organizar el set de imágenes (elaboración propia)

Lo que realiza el *script* es crear 962 carpetas independientes con los nombres de los objetos y mover las 72 imágenes que corresponden a cada carpeta. Acto seguido el programa entra a cada una de ellas y cambia los nombres de las imágenes a “0.png” – “71.png” respetando el orden secuencial de la trayectoria de la cámara. Luego de ejecutar el *script* y e integrar la carpeta raíz al PMO tenemos cubierto el primer requisito, solo estaría faltando la generación de archivos PLY y NPY los cuales almacenan nubes de puntos. Para ello debemos entender el contenido de este tipo de archivos (véase la siguiente tabla y figura).

Tabla 17. Extensión de archivos involucrados

Extensión del archivo	**Nombre completo de la extensión**	**Descripción**
PLY	*Polygon File Format*	La extensión PLY permite la representación de objetos tridimensionales en el computador mediante la

		definición de puntos y polígonos. El archivo guarda las coordenadas de cada punto y cómo estos se asocian para formar una malla de polígonos.
NPY	NumPy	Es el formato de archivo binario estándar para guardar arreglos de la librería NumPy, donde cada archivo contiene los valores de los elementos del arreglo, el tipo de dato, la forma y otras propiedades (numpy org. s.f.).

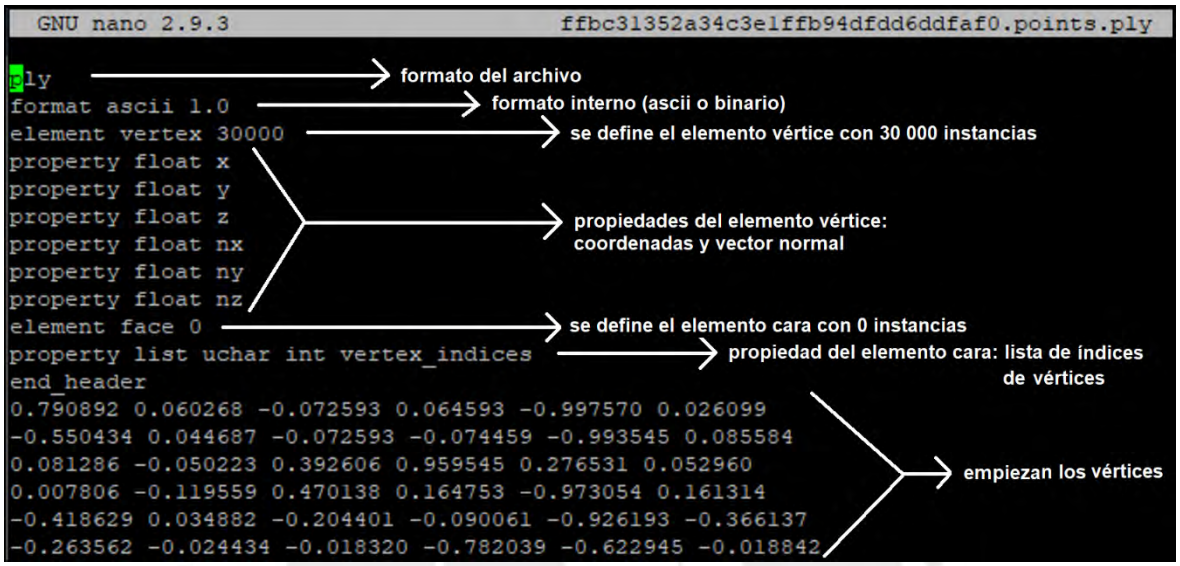


Figura 20. Contenido de un archivo PLY en formato ASCII (Groueix et al., 2018)

Como podemos observar, el archivo PLY contiene una cabecera donde se definen los elementos vértice y cara, siendo un vértice el equivalente a un arreglo de 6 números de punto flotante (3 coordenadas x-y-z y 3 componentes x-y-z del vector normal en dicho punto de la superficie) y una cara el equivalente a una lista de índices de los vértices (números enteros), en otras palabras, una cara es una relación entre los vértices. Con toda esta información podríamos reconstruir perfectamente un objeto 3D en un sistema de coordenadas, formando así una malla de polígonos. Sin embargo, si prescindimos de las caras y nos quedamos únicamente con los puntos se produce lo que llamamos nube de puntos. Es este tipo de contenido el que necesitamos para cada uno de los huacos peruanos.

Posteriormente también debemos guardar las nubes de puntos como arreglos de la librería NumPy en archivos NPY. Todo ello con el fin de que el modelo pueda procesar la data y

asociarla a las imágenes generadas. Para obtener los archivos PLY leemos cada archivo OBJ empleando la librería Open3D y lo convertimos a PLY escribiendo un archivo nuevo por cada objeto. El siguiente *script* realiza dicha tarea y además guarda algunas propiedades como el centro geométrico y la escala en un archivo de texto aparte por cada objeto (información que estaremos utilizando posteriormente).

```
import open3d as o3d
import os

total_objects = 962 # <= 9999
strs = [] # "0001" - "0962"
scale = 0.006083

# crear arreglo de strings: "0001"-"0962"
for i in range(total_objects):
    strs.append(str(i+1).zfill(4))

# convertir los archivos OBJ a PLY
# y guardar algunas propiedades de los objetos en archivos txt
os.chdir('genPlys')
for count,st in enumerate(strs):
    mesh = o3d.io.read_triangle_mesh('../'+st+'.obj')
    mesh = mesh.compute_vertex_normals()
    mesh = mesh.normalize_normals()
    o3d.io.write_triangle_mesh(st+'.points.ply', mesh,
        write_ascii=True, write_triangle_uvs=False, write_vertex_colors=False)
    ply2file = open(st+".points.ply2.txt","x")
    center = mesh.get_center()
    ply2file.write(str(round(center[0],9))+" "+str(round(center[1],9))+
        " "+str(round(center[2],9))+"\n")
    ply2file.write(str(round(scale,6))+"\n")
    ply2file.close()
    print(st)

```

Figura 21. *Script* para generar archivos PLY (elaboración propia)

Este *script* nos genera los archivos PLY, sin embargo, existen algunos inconvenientes: los objetos no se han escalado aún y tampoco están centrados en el origen de coordenadas. Por otro lado, las caras aún existen y no han sido eliminadas. Todo este proceso de normalización (escalamiento, traslación al origen de coordenadas y eliminación de caras) se lleva a cabo en un nuevo *script* desarrollado, el cual recibe como parámetros: el código de la categoría y el total objetos (962). Las acciones que realiza este segundo *script* por cada objeto son:

- Recuperar el centro geométrico y la escala leyendo el archivo de texto

- Crear un nuevo archivo PLY en el que se escribe la cabecera correspondiente con la misma cantidad de vértices, pero omitiendo las caras
- Escribir cada vértice restándole el centro geométrico y multiplicándolo por la escala, manteniendo el vector normal intacto. En este caso no necesitamos modificar los vectores normales puesto que ya son de módulo 1. También se redondea cada número de punto flotante a 6 decimales. Esta aproximación es totalmente necesaria ya que en el proyecto base PMO existe una función llamada "*sample_points_from_ply*" la cual recupera puntos aleatorios de los archivos PLY, pero los lee a nivel de caracteres numéricos y toma como muestra una cierta cantidad de caracteres. En otras palabras, si no redondeamos estos números a 6 decimales es muy probable que la función "*sample_points_from_ply*" levante una excepción y detenga por completo el entrenamiento del modelo.
- Guardar el nuevo archivo PLY

Habiendo procesado todas las piezas arqueológicas el *script* realiza las siguientes acciones generales:

- Por cada archivo PLY se leen los vértices y se guardan en un arreglo de la librería NumPy, acto seguido guarda el arreglo como un archivo NPY. Es decir, se generan los 962 archivos NPY.
- Se organizan todos los archivos siguiendo la jerarquía establecida en el PMO (véase la siguiente figura).

Para un mayor detalle de este segundo *script* con el nombre de "*act_plys3.py*" podemos revisar el Anexo D.

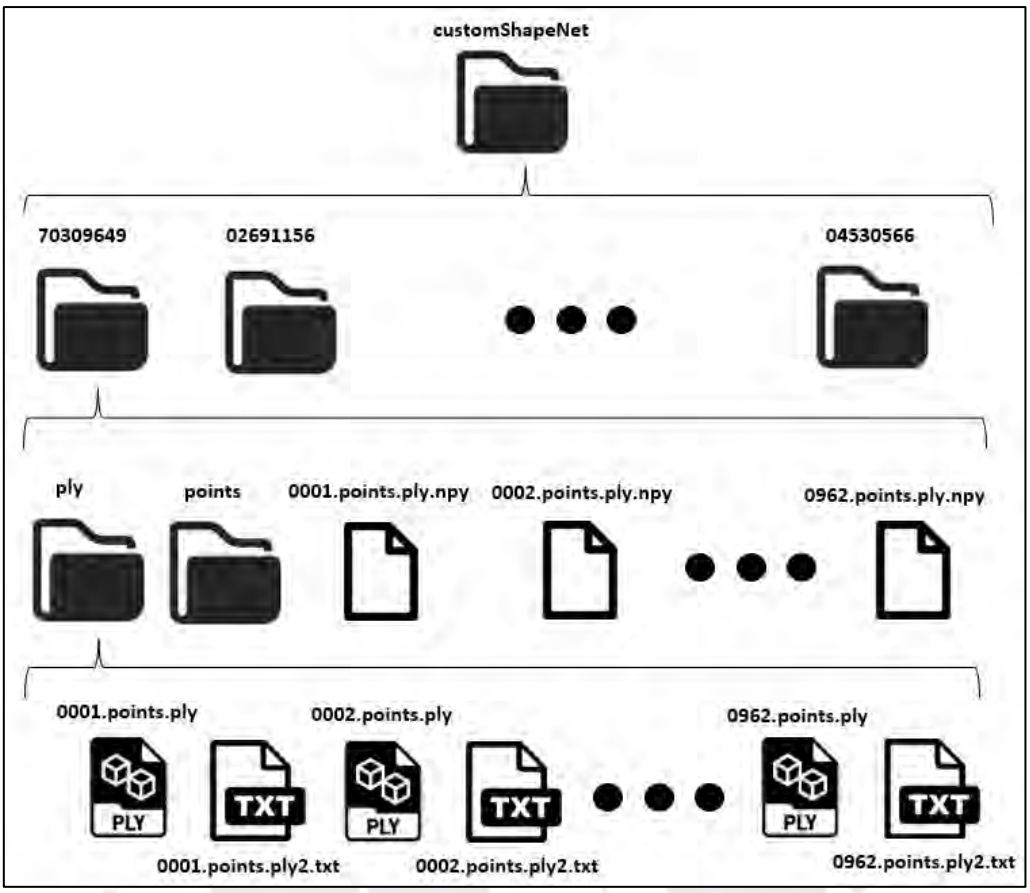


Figura 22. Nubes de puntos generadas, organizadas e integradas al PMO (elaboración propia)

Finalmente, con los datos organizados, podemos integrar estos a la carpeta "*customShapeNet*" del PMO, lugar en donde se almacenan todas las nubes de puntos de todas las categorías. Con esta última acción estamos cumpliendo los 3 requisitos necesarios para entrenar un modelo de aprendizaje de máquina. Por lo tanto, procedemos a ejecutar los siguientes comandos.

cat=70309649

python3 main_pretrain.py --category=${cat} --name=${cat}_pretrain --imagenet-enc --pretrained-dec=pretrained/ae_atlasnet_25.pth

En el primero estamos asignando el código de la categoría de huacos a la variable "*cat*", mientras que en el segundo ejecutamos el componente "*main_pretrain.py*" del PMO pasándole los parámetros necesarios para entrenar el modelo. Como veremos a continuación, por cada

época del entrenamiento se imprimen: el ratio de aprendizaje o "*lr*", la pérdida que se busca minimizar ("*loss*") y el tiempo transcurrido ("*time*"). Además, cada 50 épocas se guarda un "*checkpoint*" que es el propio modelo entrenado hasta ese instante.

Figura 23. Entrenamiento del modelo

Un detalle muy importante sobre el entrenamiento es que se utilizan las imágenes generadas en la primera sección para combinarlas con las imágenes de fondos panorámicos (descargadas en la figura 14 bajo la carpeta "*background*"). Estos fondos panorámicos constan de 72 imágenes de resolución 224 x 224 pixeles cada uno, donde la cámara se mantiene en una posición fija, pero va rotando sobre su propio eje en cada imagen y aumenta un ángulo de depresión cada 24 imágenes (véase la siguiente figura como ejemplo). Las imágenes de huacos peruanos se sobreponen sobre los fondos y así se simulan fotos reales. Este es el motivo por el cual nuestras imágenes de piezas arqueológicas debían seguir cierto patrón en cuanto a la trayectoria de la cámara y tener los mismos valores específicos de cantidad (72) y resolución (224 x 224).

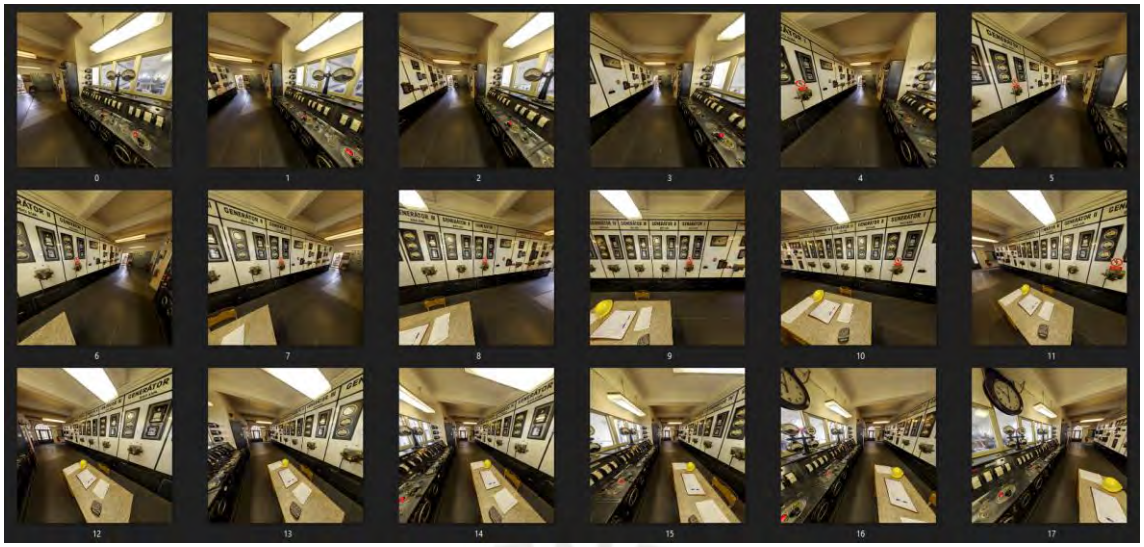


Figura 24. Secuencia de un fondo panorámico (contenido recortado)

Ahora que contamos con un modelo entrenado podemos probar la reconstrucción de algunos huacos obteniendo así unos primeros resultados. Para probar el modelo necesitamos una secuencia de 72 imágenes (almacenadas como un único archivo NPY) por cada huaco que queramos reconstruir. El modelo procesará estos arreglos Numpy, los cuales deben tener la forma (72, 224, 224, 4), donde el primer componente es la cantidad de imágenes, el segundo y tercer componente son la resolución de la cámara y el cuarto componente los valores RGB de los pixeles. Es decir, se guardan todos los valores *Red Green* y *Blue* de cada píxel y de cada imagen en un solo arreglo. Para estos primeros resultados se van a elegir aleatoriamente 4 huacos del conjunto inicial y mediante un *script* desarrollado se construyen las secuencias de imágenes, se guardan en arreglos NumPy y se guardan los arreglos como archivos NPY. Este tercer *script* de la sección actual recibe como parámetros:

- La ruta al archivo LIST del PMO donde se guardan los nombres de los fondos panorámicos usados durante entrenamiento
- La ruta a la carpeta "*background*" del PMO donde se guardan las imágenes de fondos panorámicos
- La ruta a "*rendering*" detallado en la figura 16

- La ruta a "*sequences*" dentro del PMO donde añadiremos los archivos NPY a procesar por el modelo
- La ruta a "*sequences*" fuera del PMO donde guardaremos las imágenes utilizadas para crear los archivos NPY. No es imprescindible guardar estas imágenes, pero nos ayudan a visualizar las secuencias creadas, ya que una vez convertidas a NPY no podremos visualizarlas.
- El código de la categoría (70 309 649)
- El arreglo de nombres de los objetos que se reconstruirán utilizando el modelo. Ejemplo: ["0013", "0253", "0463", "0529"]

Lo que realiza el *script* por cada objeto es:

- Recuperar un fondo panorámico aleatorio de la carpeta "*background*" del PMO
- Emplear la librería ImageIO para leer las 72 imágenes del fondo panorámico y guardarlas como 1 arreglo
- Emplear la librería ImageIO para leer las 72 imágenes del objeto 3D y guardarlas como 1 arreglo
- Componer ambas secuencias sobreponiendo los huacos sobre los fondos. Esto se realiza a nivel de arreglos.
- Emplear la librería ImageIO para guardar las imágenes compuestas
- Procesar los valores RGB del arreglo, convirtiéndolos (de valores reales que varían del 0 al 1) a valores enteros que varíen del 0 al 255
- Guardar el arreglo final como un archivo NPY

Para un mayor detalle de este tercer *script* (denominado "*create_seq.py*") podemos revisar el Anexo E. Luego de ejecutar dicho componente obtenemos los archivos NPY listos para ser procesados por el modelo. Sin embargo, actualmente el *output* del modelo son

únicamente arreglos de vértices y caras. Debemos convertir estos arreglos en archivos PLY para poder visualizar los resultados obtenidos. Para ello modificamos la función "*save_mesh*" ubicada en el componente "*util.py*" del PMO y adicionamos la creación de un archivo PLY con los arreglos de vértices y caras obtenidos de la reconstrucción del objeto. Este archivo se guardará en una ruta específica y tendrá el nombre de "*[#########]_[####].ply*" donde el primer número es el código de la categoría y el segundo es el identificador o nombre del objeto. Se muestra la versión final de esta función en la siguiente figura.

```
50  def save_mesh(opt,model,vertices):
51      os.makedirs("optimized_mesh/{0}/{1}".format(opt.group,opt.name),exist_ok=True)
52      with torch.cuda.device(opt.gpu):
53          torch.save({
54              "code": model.code,
55              "sim3": model.sim3,
56              "faces": model.faces,
57              "vertices": vertices,
58          },"optimized_mesh/{0}/{1}/{2}_{3}.npz".format(opt.group,opt.name,model.c,model.m))
59          mesh_file = open("/data/esumoso/meshes/{0}_{1}.ply".format(model.c,model.m),"x")
60          count_vertices = list(vertices.size())[0] #number of vertices
61          count_faces = list(model.faces.size())[0] #number of faces
62          print(str(count_vertices)+" "+str(count_faces))
63          mesh_file.write("ply\nformat ascii 1.0\nelement vertex {0}\n".format(count_vertices))
64          mesh_file.write("property float x\n")
65          mesh_file.write("property float y\n")
66          mesh_file.write("property float z\n")
67          mesh_file.write("element face {0}\n".format(count_faces))
68          mesh_file.write("property list uchar int vertex_indices\nend_header\n")
69          for i in range(count_vertices):
70              mesh_file.write("{0} {1} {2}\n".format(round(float(vertices[i][0]),6),
71                  round(float(vertices[i][1]),6),round(float(vertices[i][2]),6)))
72          for i in range(count_faces):
73              mesh_file.write("3 {0} {1} {2}\n".format(model.faces[i][0],model.faces[i][1],
74                  model.faces[i][2]))
75      print(green("optimized mesh saved: ({0}) {1}".format(opt.group,opt.name)))
```

Figura 25. Versión final de función "*save_mesh*"

Un último detalle antes de ejecutar los comandos de reconstrucción es el desarrollo de un pequeño cuarto *script* el cual nos automatiza la ejecución de los comandos a ejecutar. Este pequeño *script* toma como parámetros la ruta del modelo (en este caso el *checkpoint* guardado en la época 500), el código de la categoría (70 309 649), la ruta al PMO (carpeta raíz del proyecto) y el arreglo de objetos (Ejemplo: ["0013", "0253", "0463", "0529"]). También se añaden 2 parámetros generales al PMO "*filecategory*" y "*filename*" los cuales guardan la categoría y el nombre del objeto que se quiere reconstruir (Ejemplo: *filecategory* guarda el

valor “70309649” y *filename,* “0013”). Al ejecutar este último *script* (véase la figura) denominado “*exe_seqs.py*” estamos reconstruyendo 4 huacos peruanos empleando el modelo entrenado.

```
import os
model = "checkpoint/1/70309649_pretrain_seed0/ep500.npz"
category = "70309649"
path = "/data/esumoso/pmo"
objects = ['0013','0253','0463','0529']
os.chdir(path)
for obj in objects:
    os.system("model="+model+"; python3 main.py --load=${model}" +
        " --code=5e-2 --scale=2e-2 --lr-pmo=3e-3 --noise=0.1" +
        " --filecategory "+category+" --filename "+obj+" --gpu 1;")

```

Figura 26. *Script* “*exe_seqs.py*” para automatizar la ejecución de los comandos (elaboración propia)

4.2.3.3 Medio de verificación e IOV

Tabla 18. Medio de verificación e IOV del RE3 (tomado del Capítulo 1)

Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R3. Modelo entrenado con los datos preprocesados	• *Checkpoints* del modelo guardados durante el entrenamiento • Reconstrucciones 3D de huacos peruanos empleando el modelo entrenado	• Al menos 10 *checkpoints* generados durante el entrenamiento • Al menos 4 reconstrucciones de prueba

Para corroborar el resultado esperado de esta sección nos ubicamos en la carpeta del PMO donde se guardan los modelos entrenados. Como se aprecia en la siguiente figura, en la categoría 70 309 649 encontraremos los 10 *checkpoints* guardados durante el entrenamiento. Estos *checkpoints* se guardaron cada 50 épocas, teniendo un entrenamiento de 500 épocas.

```
esumoso@enterprise:/data/esumoso/pmo$
esumoso@enterprise:/data/esumoso/pmo$ cd checkpoint/0
esumoso@enterprise:/data/esumoso/pmo/checkpoint/0$ ls
70309649_pretrain_seed0
esumoso@enterprise:/data/esumoso/pmo/checkpoint/0$ cd 70309649_pretrain_seed0
esumoso@enterprise:/data/esumoso/pmo/checkpoint/0/70309649_pretrain_seed0$ ls
ep100.npz  ep200.npz  ep300.npz  ep400.npz  ep500.npz
ep150.npz  ep250.npz  ep350.npz  ep450.npz  ep50.npz
esumoso@enterprise:/data/esumoso/pmo/checkpoint/0/70309649_pretrain_seed0$
```

Figura 27. *Checkpoints* guardados durante el entrenamiento

Por otro lado, tenemos los 4 archivos PLY generados precisamente de las reconstrucciones 3D de los huacos escogidos aleatoriamente como prueba. En la siguiente figura vemos los resultados cualitativos, donde comparamos una imagen del objeto con la reconstrucción obtenida. Para visualizar los objetos 3D se utilizó el software Blender. Los objetos escogidos aleatoriamente para las pruebas son (en orden superior a inferior): “0001”, “0227”, “0402” y “0616” del universo de 962 huacos.

Figura 28. Primeros resultados cualitativos de las reconstrucciones 3D

4.3 Discusión

El primer problema causa definido en el árbol de problemas es: “No se enfoca algún método de escaneo 3D sin contacto – pasivo en la reconstrucción de huacos peruanos específicamente”. Entendiendo que nuestro método de escaneo 3D sin contacto – pasivo en este caso será el uso de un modelo de aprendizaje de máquina para la reconstrucción 3D de piezas arqueológicas, se ha dividido la solución de este problema causa en 3 resultados consecutivos. Cada uno de ellos forma parte del proceso que debemos seguir para finalmente obtener un modelo entrenado en huacos peruanos que pueda generar reconstrucciones 3D de dichos objetos.

En el primer resultado hemos generado un extenso set de imágenes bajo ciertos estándares y patrones de cámara utilizando una herramienta la cual modificamos a nivel de código. Es importante generar estos datos puesto que serán procesados por el modelo posteriormente. En el segundo implementamos un *script* que automatiza la segmentación de datos, introducimos el proyecto PMO, modificamos y organizamos los archivos correspondientes para poder integrar la data al proyecto, teniendo en cuenta la segmentación planteada de 80%-20%. Como tercer y último resultado generamos las nubes de puntos (archivos PLY y NPY), los normalizamos y los integramos al PMO.

Finalmente habremos integrado una nueva categoría al proyecto la cual utilizamos para entrenar un modelo de aprendizaje de máquina y obtener unos primeros resultados de reconstrucciones. De momento la limitante presente es que hemos trabajado en base únicamente a los 962 piezas arqueológicas proporcionadas por el grupo IA-PUCP para el presente trabajo de investigación. Sin embargo, se ha resuelto satisfactoriamente el primer problema causa ya que ahora contamos con una herramienta que hace uso del método de escaneo 3D sin contacto – pasivo enfocado únicamente a huacos peruanos.

Capítulo 5. Mejora en la reconstrucción de objetos 3D considerando la alta variabilidad de piezas arqueológicas

5.1 Introducción

En el presente capítulo se describe el proceso desarrollado asociado al objetivo específico número 2: "Adaptar el proyecto y mejorar la reconstrucción de objetos 3D considerando la alta variabilidad de las piezas arqueológicas". En este segundo objetivo se busca primero mejorar la reconstrucción 3D de las piezas arqueológicas, puesto que los primeros resultados obtenidos no fueron del todo favorables. Para mejorar las reconstrucciones se emplea la técnica de "*data augmentation*" y se combina con una segmentación interna del conjunto de datos en subconjuntos que compartan características similares. Se crean *script*s para reproducir la segmentación y normalización de datos de un subconjunto. Con todas las nuevas categorías creadas se realizan 2 experimentos. El primero consiste en entrenar un modelo de aprendizaje de máquina por cada categoría identificada dentro del set de 962 piezas arqueológicas (con un previo descarte de categorías con data insuficiente). Mientras que el segundo experimento consiste en entrenar un solo modelo con todas las categorías identificadas (excluyendo las descartadas).

Al terminar el cuarto resultado obtendremos 8 modelos entrenados cuyas reconstrucciones se comparan con las iniciales y notamos una mejora en los resultados. Como siguiente paso o quinto resultado esperado se proceden a construir 21 videos que simulan grabaciones reales a piezas arqueológicas. Estos videos están construidos en 3 diferentes resoluciones y 7 duraciones en segundos. La idea es contar con la mayor variedad de resolución y duración para poder posteriormente realizar pruebas de videos *input* a los modelos entrenados.

Finalmente, como sexto resultado esperado se define un *pipeline* el cual nos permite procesar archivos de video en los modelos entrenados. Existe todo un procesamiento previo que deben sufrir los videos para que puedan ser procesados, en general, se escala el video a una resolución específica, se generan los fotogramas como imágenes independientes, se seleccionan ciertos fotogramas de toda la secuencia, se convierten las imágenes en un arreglo NumPy, se procesan los valores RGB de los pixeles y finalmente se guarda el arreglo como un archivo NPY. Además, se definen cuestiones como los formatos de video permitidos y otras restricciones.

5.2 Resultados Alcanzados

5.2.1 Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos

5.2.1.1 Descripción

En esta sección se describirá el resultado esperado número 4: "Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos". Habiendo obtenido unos primeros resultados en la sección anterior, necesitamos mejorar las reconstrucciones 3D adaptando el proyecto mediante una serie de métodos los cuales se irán detallando a lo largo de la sección. En general, primero hallamos la escala individual de cada objeto (ya no usamos una sola escala general para todo el set de datos), segundo segmentamos el conjunto inicial de 962 piezas arqueológicas en varios subconjuntos que compartan características similares. Tercero, se aplica el método de "*data augmentation*" para cada uno de los subconjuntos que superen una cierta cantidad de elementos. Cuarto, se desarrollan los *script*s para automatizar las tareas de generación de nubes de puntos e imágenes de cada subconjunto, considerando el "*data augmentation*". Quinto, con los datos generados se llevan a cabo 2 experimentos: entrenar 1 modelo por cada subconjunto y entrenar 1 modelo que incluya a todos los subconjuntos. Finalmente se comparan los resultados cualitativos y cuantitativos obtenidos de cada uno de

los modelos entrenados. Para los resultados cualitativos se comparan las imágenes de los objetos con sus reconstrucciones 3D generadas. Por otro lado, para los resultados cuantitativos se comparan los promedios de la función de pérdida o “loss" para ambos experimentos y el primer modelo entrenado en el capítulo anterior.

5.2.1.2 Métodos y procedimientos empleados

Tabla 19. Herramientas para el RE4 (tomado del Capítulo 1)

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos
R4. Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos	Hallar las escalas individuales de cada objeto, realizar una segmentación manual interna para definir subcategorías dentro del set inicial de 962 piezas arqueológicas, desarrollar *script*s para la normalización de la data y generación de imágenes de las subcategorías aplicando la técnica de *data augmentation*. Finalmente, entrenar nuevos modelos de aprendizaje de máquina bajo 2 experimentos definidos, obteniendo resultados cualitativos y cuantitativos.	• PMO • Python • Sublime Text 3 • PuTTY • Blender • *Data augmentation*

Lo primero que hay que mejorar es la escala que hemos definido para todo el conjunto. En el primer resultado se utilizó la escala 0.006 053 como valor general para todo el set de datos debido a que dicho valor encajaba a los objetos dentro del lente de la cámara. Esto causaba que los objetos más grandes encajen perfectamente en las fotos, sin embargo, los más pequeños se volvían difíciles de distinguir. Por ende, la primera mejora que haremos es encontrar una escala adecuada para cada una de las 962 piezas del set de datos. Para agilizar convenientemente esta tarea se realizó el siguiente artificio:

- Ejecutamos 8 veces el siguiente comando recuperado de la primera sección, pero cambiando la escala a múltiplos de 0.0005 empezando del 0.004 y terminando en 0.0075. Por supuesto también se cambió la ruta de salida en cada ejecución

find /data/esumoso/HuacosFinal/models -name '.obj' -print0 | xargs -0 -n1 -P3 -I {}*

blender --background --python render_blender.py -- --output_folder

/data/esumoso/renderings {} --scale=0.006083

- Por cada objeto comparamos las imágenes generadas en las diferentes escalas y nos quedamos con la mejor opción, es decir, donde el objeto encaje perfectamente en la imagen. Aproximadamente la escala del 80% de las piezas se encontraba dentro del rango ejecutado (0.004 – 0.0075). El resto de las piezas tuvieron que ser ejecutadas individualmente ya que sus escalas estaban fuera del rango.

Luego de realizar esta tarea se guardaron las 962 escalas en un archivo de texto denominado "*scales.txt*", donde cada línea contiene 1 número de punto flotante. Habiendo determinado estos valores procedemos a resolver otros 2 problemas totalmente distintos que afectan en gran medida los resultados obtenidos. El primer problema es la alta variabilidad del set de datos, nuestras piezas arqueológicas son muy variables y tienen múltiples formas, muchas de ellas totalmente distintas al resto. El segundo problema es la poca cantidad de datos con los que contamos. A diferencia de otros conjuntos predefinidos del PMO que contienen aproximadamente 4000 objetos, nuestro set de datos apenas contiene 962.

Para atacar el primer problema se propone segmentar el set de datos en subconjuntos los cuales iremos definiendo. Esta clasificación se hace manualmente ya que necesitamos analizar cada pieza y decidir a qué subconjunto pertenece. El resultado final de la segmentación se resume en el siguiente cuadro.

N°	Nombre	Total objetos
1	animal-bottle	26
2	animal-head	9
3	**bowl**	132
4	**cone-vase**	157
5	cuenco	31
6	flat-canteen	16
7	**jar**	65
8	**lebrillo**	83
9	**olla**	112
10	**plate**	84
11	statue	40
12	vase	27
13	**vessel**	132
	Total general	**914**

Figura 29. Segmentación del set de datos

Como podemos apreciar, se definieron 13 subconjuntos con los nombres traducidos: botellas en forma de animales, cabezas de animales, tazones, jarrones en forma de cono, cuencos, cantimploras planas, jarras, lebrillos, ollas, platos, estatuas, jarrones y vasijas. Se detalla el total de objetos asignados por cada subconjunto y el total general (914), las piezas faltantes (48) fueron descartadas ya que no encajaban en ningún subconjunto. Si escogemos únicamente los grupos que superan los 60 elementos (filas resaltadas en verde) nos quedamos con 7 subconjuntos (*bowl*, *cone-vase*, *jar*, lebrillo, olla, *plate* y *vessel*), los demás contienen muy pocos elementos (menor o igual a 40), por lo tanto, no es viable procesarlos con los métodos siguientes.

Habiendo resuelto el primer problema (variabilidad), se procede a resolver el segundo contratiempo (falta de datos). Como se mencionó previamente, a diferencia de los conjuntos predefinidos del PMO (4000 objetos) nuestro set de datos apenas contiene 962. Peor aún, con la segmentación realizada nos quedamos con 914 de los cuales solamente 765 piezas pertenecen a los subconjuntos escogidos (color verde). Para afrontar este problema se propone aplicar la técnica de *data augmentation*. Este método consiste en crear copias ligeramente

modificadas del conjunto inicial de datos de tal forma que podamos ampliar nuestro universo de objetos y proveer así suficiente data para entrenar modelos de aprendizaje de máquina.

En este caso, se aplicará una rotación en cada copia de los objetos. Esta rotación debe aplicarse tanto a las imágenes como a las nubes de puntos. Además, necesitamos al menos 4000 objetos por subconjunto. La siguiente tabla detalla cómo aplicaremos esta técnica, donde la rotación se calcula en grados sexagesimales.

SUBCONJUNTO	OBJETOS	COPIAS POR OBJETO	TOTAL PSEUDO OBJETOS	ROTACIÓN	NOMBRE EN PMO	CÓDIGO EN PMO
CONE-VASE	157	26	4082	13	subhuacos	70309650
BOWL	132	31	4092	11	bowl2	70309651
JAR	65	62	4030	5	jar2	70309652
LEBRILLO	83	49	4067	7	lebrillo2	70309653
OLLA	112	36	4032	10	olla2	70309654
PLATE	84	48	4032	7	plate2	70309655
VESSEL	132	31	4092	11	vessel2	70309656
TOTAL	765	6	4590	60	exp2	70309660

Figura 30. Detalles de los subconjuntos

De izquierda a derecha encontramos lo siguiente: el nombre del subconjunto, la cantidad de objetos asignados, las copias a generar por objeto, el total de pseudo objetos (objetos multiplicado por las copias), la rotación aplicada en cada copia (en grados sexagesimales) y el nombre y código asignados al subconjunto en el PMO. Nótese que el total es considerado otro subconjunto de 765 objetos donde se realizan 6 copias por objeto con una rotación de 60° para obtener así 4590 pseudo objetos.

Para llevar a cabo la técnica de *data augmentation* se han desarrollado 2 *script*s denominados "*subrendering.py*" y "*subset.py*" los cuales podemos analizar a detalle en el Anexo F. El primero se encarga de generar las imágenes con la herramienta Stanford Shapenet Renderer y la trayectoria de la cámara definida en el primer resultado de esta tesis, aunque aplicándolo únicamente a un subconjunto de las 962 piezas arqueológicas iniciales. Este *script*

emplea la técnica de *data augmentation* creando copias, aplicando una rotación y considerando las escalas individuales. Los parámetros que recibe son:

- El arreglo de objetos a procesar. Por ejemplo, para el subconjunto *cone-vase* tendremos un arreglo de 157 cadenas de caracteres de la siguiente forma: [“0013”, “0052”, “0063”, …, “0912”]
- El arreglo de escalas individuales. Este contiene las escalas correspondientes a los objetos del arreglo anterior. Ejemplo: [“0,006”, “0.0055”, “0.007”, …, “0.005”]
- La cantidad de copias por objeto (número entero)
- El ángulo de rotación en grados sexagesimales (número de punto flotante)
- La ruta al componente Stanford Shapenet Renderer, el cual fue modificado en el primer resultado y nos permite generar las imágenes
- La ruta a la carpeta que contiene los archivos OBJ del conjunto inicial
- Nombre de la carpeta que contiene el conjunto inicial
- La ruta de salida, donde se guardarán todas las imágenes
- Código de la categoría en el PMO (última columna del cuadro anterior)

Lo que realiza el *script* con todos estos parámetros es primero crear una carpeta con el código de la categoría. Acto seguido, por cada objeto del arreglo y por cada copia se realiza lo siguiente:

- Se aumenta en 1 el contador general (este varía del 1 al total de pseudo objetos, es decir, aproximadamente 4000)
- Se generan las imágenes del objeto en cuestión haciendo una llamada a la herramienta Stanford Shapenet Renderer y pasándole como parámetros: la ruta de salida, la escala correspondiente al objeto y el ángulo de rotación. Para este último parámetro se tuvo

que modificar ligeramente el código de la herramienta mencionada, rotando la posición inicial de la cámara.

- Con las imágenes generadas se crea una carpeta con el nombre del objeto (utilizando el contador general), se mueven las imágenes dentro de dicha carpeta y se cambian los nombres de estas a “0” - “71” respetando el orden secuencial de la cámara.

Finalmente, con el *script* ejecutado tendremos una carpeta general (nominada con el código de la categoría) que contiene carpetas (nominadas del “0001” al “4000” aproximadamente) y 72 imágenes dentro de cada una de estas, similar a la figura 16. Ahora pasaremos al segundo *script* “*subset.py*” que se encarga de generar las nubes de puntos (archivos PLY y NPY) de nuestros subconjuntos. Este *script* recibe como parámetros:

- El arreglo de objetos a procesar. Por ejemplo, para el subconjunto *cone-vase* tendremos un arreglo de 157 cadenas de caracteres de la siguiente forma: [“0013”, “0052”, “0063”, …, “0912”]
- La ruta donde se guardarán los archivos PLY
- La ruta donde se guardarán los archivos NPY
- La ruta donde encontramos a los 962 archivos PLY obtenidos en el tercer resultado esperado
- La rotación en grados sexagesimales (número de punto flotante)
- La cantidad de copias por objeto (número entero)

Acto seguido se calcula la matriz de rotación (en sentido antihorario) la cual se aplicará en cada copia, para ello se utiliza el parámetro de grados sexagesimales. A continuación, por cada objeto se realiza lo siguiente:

- Se emplea la librería Open3D para leer el archivo PLY (generado en el tercer resultado) del objeto en cuestión

- Por cada copia que se tenga que crear del objeto se realiza lo siguiente:
    - Se aumenta en 1 el contador general (este varía del 1 al total de pseudo objetos, es decir, aproximadamente 4000)
    - Rotamos el archivo PLY utilizando la matriz de rotación y el centro del objeto
    - Guardamos el nuevo archivo PLY rotado empleando la librería Open3D. Este nuevo archivo es temporal ya que contiene los estándares de la librería y debemos procesarlo nuevamente para que siga los estándares del PMO.
    - Creamos un nuevo archivo PLY (versión final) donde escribimos la cabecera correspondiente, definiendo los vértices y cero caras.
    - Escribimos todos los vértices del objeto leyéndolos del archivo PLY generado por la librería y escribiéndolos en nuestra versión final. Se redondean los valores a 6 decimales.
- Generar el archivo NPY. Recordemos que para ello se lee el archivo PLY (versión final) utilizando la librería Open3D, se guardan los vértices en un arreglo NumPy y se almacena este arreglo como un archivo NPY.
- Copiamos el archivo de texto que contiene el centro del objeto y la escala individual a la ruta de salida de los PLY. Estos archivos TXT se generaron durante el tercer resultado.

Finalmente se borran los archivos PLY temporales, los cuales equivalen a aproximadamente 4 GB. Para un mayor detalle de ambos *script*s revisar el Anexo F. Ambos *script*s se ejecutan 8 veces (7 veces de los subconjuntos y 1 del total mostrado en la figura 30) cambiando los parámetros respectivos. Cabe resaltar que cada ejecución de "*subrendering.py*" tomó aproximadamente 30 horas, mientras que cada ejecución de "*subset.py*" menos de 5 minutos. Todo este procesamiento finalizó luego de 10 días. Con los nuevos subconjuntos definidos se proponen 2 experimentos para mejorar las reconstrucciones 3D de los huacos:

- Entrenar 1 nuevo modelo por cada uno de los 7 subconjuntos seleccionados
- Entrenar 1 modelo con la data total de los 7 subconjuntos

Para entrenar un nuevo modelo de aprendizaje recordemos que existían 3 requisitos:

- Contar con 1 extenso set de imágenes correspondientes a los objetos 3D
- Contar con los archivos PLY y NPY (cada par de archivos representa 1 nube de puntos)
- Crear/modificar los archivos LIST para la adición de una nueva categoría y su segmentación

Ya contamos con los 2 primeros, únicamente estaría faltando la modificación de los archivos LIST en el PMO. Para ello ejecutamos el *script* "*write_seq.py*" desarrollado en la sección anterior. Solo debemos cambiar los parámetros: código, nombre y total de objetos de la categoría, los cuales encontramos en la figura 30 por cada subconjunto. Luego de ejecutar 8 veces dicho *script*, cambiando los parámetros, habremos concluido con el tercer y último requisito. Ahora podemos empezar con el entrenamiento de los modelos.

cat=70309650

python3 main_pretrain.py --category=${cat} --name=${cat}_pretrain --imagenet-enc --pretrained-dec=pretrained/ae_atlasnet_25.pth

Ejecutamos ambos comandos 8 veces cambiando la variable "*cat*" en cada ejecución y obtenemos 8 nuevos modelos entrenados. Cada entrenamiento duró aproximadamente 5 horas con 40 minutos, es decir, un total de 45 horas con 20 minutos. Habiendo entrenado satisfactoriamente todos nuestros modelos procedemos a realizar las pruebas para comprobar si se han mejorado las reconstrucciones 3D de huacos peruanos. Para ello, se propone ejecutar 4 pruebas por cada modelo del 1° experimento, (es decir, un total de 28 reconstrucciones) y las mismas pruebas empleando el modelo del 2° experimento (otras 28 reconstrucciones). Con

todo ello finalmente tendremos un total de 56 piezas reconstruidas con los que podremos determinar cualitativa y cuantitativamente si las reconstrucciones 3D han prosperado.

A continuación, se muestran los objetos de prueba escogidos para los subconjuntos "*plate*" y "*vessel*". Estos mismos objetos ("0014", "0093", "0289", "0416", "0029", "0031", "0045" y "0046") también pertenecen a las pruebas del 2° experimento. El Anexo G es un archivo en Excel donde por cada subcategoría se detalla: la lista de objetos asignados y los 4 objetos de prueba (resaltados de amarillo). En el 2° experimento los objetos de prueba ascienden a 28 en total, se recomienda revisar el Anexo G.

PLATE			VESSEL				2° EXP		
INDEX	OBJ	PSEUDO	INDEX	OBJ	PSEUDO		INDEX	OBJ	PSEUDO
1	0014.obj	1	1	0029.obj	1		1	0001.obj	1
2	0093.obj	49	2	0031.obj	32		2	0002.obj	7
3	0094.obj	97	3	0045.obj	63		3	0003.obj	13
4	0235.obj	145	4	0046.obj	94		4	0004.obj	19
5	0289.obj	193	5	0050.obj	125		5	0005.obj	25
6	0292.obj	241	6	0072.obj	156		6	0006.obj	31
7	0293.obj	289	7	0082.obj	187		7	0007.obj	37
8	0294.obj	337	8	0092.obj	218		8	0008.obj	43
9	0297.obj	385	9	0120.obj	249		9	0009.obj	49
10	0298.obj	433	10	0126.obj	280		10	0010.obj	55
11	0303.obj	481	11	0133.obj	311		11	0011.obj	61
12	0306.obj	529	12	0162.obj	342		12	0012.obj	67
13	0307.obj	577	13	0163.obj	373		13	0013.obj	73
14	0308.obj	625	14	0176.obj	404		14	0014.obj	79
15	0309.obj	673	15	0186.obj	435		15	0015.obj	85
16	0311.obj	721	16	0187.obj	466		16	0016.obj	91
17	0315.obj	769	17	0189.obj	497		17	0017.obj	97
18	0316.obj	817	18	0202.obj	528		18	0018.obj	103
19	0317.obj	865	19	0217.obj	559		19	0019.obj	109
20	0332.obj	913	20	0218.obj	590		20	0020.obj	115
21	0333.obj	961	21	0219.obj	621		21	0021.obj	121
22	0334.obj	1009	22	0236.obj	652		22	0022.obj	127
23	0335.obj	1057	23	0243.obj	683		23	0023.obj	133
24	0336.obj	1105	24	0250.obj	714		24	0024.obj	139
25	0337.obj	1153	25	0259.obj	745		25	0025.obj	145
26	0338.obj	1201	26	0264.obj	776		26	0026.obj	151
27	0398.obj	1249	27	0267.obj	807		27	0028.obj	157
28	0402.obj	1297	28	0286.obj	838		28	0029.obj	163
29	0415.obj	1345	29	0296.obj	869		29	0030.obj	169
30	0416.obj	1393	30	0300.obj	900		30	0031.obj	175

Figura 31. Objetos escogidos para pruebas (contenido recortado)

Para empezar con las pruebas utilizaremos los *scripts* "*create_seq.py*" (Anexo E) y "*exe_seqs.py*" (Figura 26) desarrollados en la sección anterior. El primero nos permite crear

los archivos NPY que serán procesados por los modelos y el segundo ejecuta los comandos de reconstrucción (se recomienda revisar la sección anterior para entender a detalle lo realizado por cada *script*). Recordemos que cada modelo ejecutará 4 reconstrucciones, a excepción del 2° experimento cuyo caso ejecuta 28.

Habiendo ejecutado ambos *script*s 8 veces (cambiando los parámetros) obtenemos 56 archivos PLY los cuales podemos visualizar mediante la herramienta Blender. Estos archivos son precisamente las reconstrucciones 3D obtenidas de los huacos escogidos para las pruebas. Ahora evidenciaremos los resultados cualitativos comparando los objetos 3D con una imagen del objeto original.

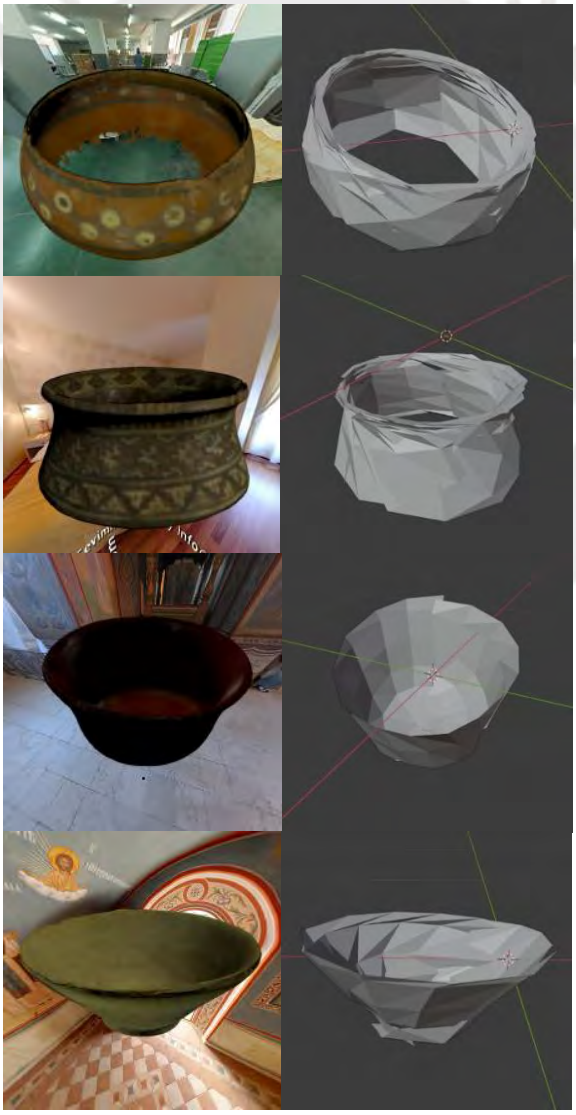


Figura 32. Resultados cualitativos de los nuevos modelos entrenados (contenido recortado)

Debido a la extensa cantidad de resultados estos se encuentran en un documento aparte como el Anexo H. Como hemos podido notar estos han mejorado en comparación a los resultados obtenidos en una primera instancia. Sin embargo, también debemos cuantificar los resultados para llegar a una conclusión correctamente sustentada. Para ello, calcularemos los indicadores de pérdida o "*loss*" que se imprime en consola cada vez que ejecutamos una reconstrucción.

Durante la ejecución de cada reconstrucción 3D existe una función de pérdida la cual se busca minimizar en cada época del procesamiento. Para las reconstrucciones se han establecido 100 épocas por defecto del proyecto PMO. En cada una de ellas se imprime en consola el valor de la pérdida o "*loss*", el cual idealmente debe reducirse en cada época. El valor de pérdida final (época 100) es la resultante del menor valor posible logrado durante todo el procesamiento. Es este valor el que consideraremos para el cálculo de nuestros indicadores.

Para el modelo inicial calcularemos el promedio o media de los 4 valores de "*loss*" recuperados de las 4 reconstrucciones realizadas en el capítulo anterior. De la misma manera se calcula el promedio del "*loss*" de las 4 reconstrucciones ejecutadas por cada uno de los modelos entrenados del 1° experimento. Para el 2° experimento, el indicador es calculado a partir de las 28 reconstrucciones efectuadas con el último modelo entrenado. La siguiente tabla nos muestra los valores calculados.

	SUBCONJUNTO	CÓDIGO DE LA CATEGORÍA	CANTIDAD DE OBJETOS	CANTIDAD DE RECONSTRUCCIONES DE PRUEBA REALIZADAS	VALORES DE PÉRDIDA REGISTRADOS (ÉPOCA 100)				INDICADOR	DIFERENCIA PORCENTUAL CON MODELO INICIAL
					1° VALOR DE PÉRDIDA O "LOSS"	2° VALOR DE PÉRDIDA O "LOSS"	3° VALOR DE PÉRDIDA O "LOSS"	4° VALOR DE PÉRDIDA O "LOSS"	PROMEDIO DE LOS VALORES DE PÉRDIDA O "LOSS"	
MODELO INICIAL	-	70309649	962	4	6.2804	3.8412	0.2179	2.0276	3.0918	0%
1° EXPERIMENTO	CONE-VASE	70309650	4082	4	3.1716	1.9519	3.1259	2.7298	2.7448	-13%
	BOWL	70309651	4092	4	4.8243	0.4948	0.8785	1.8274	2.0063	-54%
	JAR	70309652	4030	4	1.2613	3.6641	3.8413	1.3534	2.5300	-22%
	LEBRILLO	70309653	4067	4	7.3706	4.4283	1.0476	8.8732	5.4299	43%
	OLLA	70309654	4032	4	5.1552	2.2650	1.3185	0.9529	2.4229	-28%
	PLATE	70309655	4032	4	1.1398	0.8489	0.9470	0.8700	0.9514	-225%
	VESSEL	70309656	4092	4	3.6752	1.9381	4.5064	2.9965	3.2791	6%
2° EXPERIMENTO	-	70309660	4590	28	4.7400	6.9895	2.1916	6.1897	2.8409	-9%
					2.8065	5.9074	3.8930	3.1472		
					0.9920	0.4786	0.9470	1.6827		
					2.6985	3.6774	3.6730	2.5516		
					2.1674	1.2215	4.4942	3.7880		
					1.8862	1.4530	0.9851	0.8689		
					1.4858	6.6860	0.9825	0.9622		

Figura 33. Resultados cuantitativos de todos los modelos entrenados

Como podemos apreciar, ambos experimentos muestran una mejora (el valor de pérdida es menor) de ambos indicadores en comparación al modelo inicial. También se han calculado las diferencias porcentuales. Por lo tanto, podemos concluir que estos nuevos modelos se adaptan mejor a la reconstrucción de superficies 3D de huacos peruanos, a excepción de los modelos entrenados para las categorías "Lebrillo" y "*Vessel*", cuyos indicadores aumentan.

5.2.1.3 Medio de verificación e IOV

Tabla 20. Medio de verificación e IOV del RE4 (tomado del Capítulo 1)

Resultado Esperado	Medio de verificación	Indicador objetivamente verificable
R4. Modelos adaptados a la reconstrucción de superficies 3D de huacos peruanos	Tabla con los resultados cuantitativos obtenidos del modelo inicial y los experimentos realizados.	Los indicadores de los últimos resultados (experimentos) deben ser mejores que los del modelo inicial.

En la figura 33 encontramos que el indicador calculado tiene mejores valores (el promedio de la pérdida de las reconstrucciones es menor) que los obtenidos en una primera instancia por el modelo inicial entrenado en el Capítulo 3. El indicador general del primer experimento sigue siendo favorable a pesar de los 2 modelos ("Lebrillo" y "*Vessel*") que presentan indicadores mayores al del modelo inicial. En conclusión, se cumple con el IOV

planificado, ya que nuestros resultados cuantitativos sustentan los mejores resultados obtenidos con estos últimos modelos.

5.2.2 Videos construidos que simulan grabaciones reales de piezas arqueológicas

5.2.2.1 Descripción

En esta sección se describirá el resultado esperado número 5: "Videos construidos que simulan grabaciones reales de piezas arqueológicas". En este resultado se construyen los videos de prueba que simulan videograbaciones reales a piezas arqueológicas. Estos videos servirán posteriormente para llevar a cabo pruebas con la herramienta desarrollada en los resultados anteriores. En total, se realizaron 21 videos de 3 diferentes resoluciones y 7 duraciones distintas (tiempo) con el propósito de simular aleatoriedad (de resolución y duración de los videos). Para llevar a cabo nuestro cometido se empleó el editor de videos de la aplicación "Fotos" de Windows 10 y las imágenes generadas en el resultado anterior para probar los modelos de aprendizaje de máquina. Cada video consta de una pieza distinta.

5.2.2.2 Métodos y procedimientos empleados

Tabla 21. Herramientas para el RE5 (tomado del Capítulo 1)

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos
R5. Videos construidos que simulan grabaciones reales de piezas arqueológicas	Edición de videos para la construcción de grabaciones simuladas a piezas arqueológicas, definiendo los rangos de resolución y duración de los videos.	• Editor de videos de la aplicación "Fotos" de Windows 10

Para empezar, escogemos los rangos de resolución y duración de los videos prueba que vamos a construir. Para ello observamos la siguiente figura. Como podemos apreciar los rangos definidos (resaltados en negro) son:

- Para la duración de los videos: 10, 15, 20, 25, 30, 35 y 40 segundos
- Para la resolución de los videos: 540p, 720p y 1080p

	Resolución			Duración por
Duración (s)	540p	720p	1080p	imagen (s)
10	"0013"	"0253"	"0433"	0.13
15	"0031"	"0259"	"0463"	0.20
20	"0079"	"0337"	"0499"	0.26
25	"0139"	"0343"	"0529"	0.34
30	"0169"	"0355"	"1105"	0.40
35	"0175"	"0409"	"1543"	0.47
40	"0193"	"0421"	"2143"	0.54

Figura 34. Tabla de videos prueba

En el centro de la tabla tenemos los nombres de los objetos a utilizar para construir cada video. Por ejemplo, el objeto “0013” será utilizado para construir un video de 10 segundos y de resolución 540p. De esta forma contamos con 21 videos cada uno con diferente combinación de resolución y duración. Los nombres de los objetos son los pertenecientes a la categoría 70 309 660 desarrollada en el resultado previo (2° experimento). Se escogió esta categoría simplemente por el hecho de que ya contiene a todos los objetos de los subconjuntos segmentados. Para proceder con la creación de los videos abrimos la aplicación “Fotos” de Windows y nos ubicamos en el editor de videos (véase la figura). Acto seguido, por cada video realizamos lo siguiente:

- Importamos las 72 imágenes correspondientes al objeto definido
- Movemos las 72 imágenes a la línea temporal del video (estas ya se encuentran en orden gracias a su nomenclatura “0” - “71”.
- Seleccionamos a todas las imágenes de la línea temporal
- Cambiamos la duración de todas las imágenes seleccionadas según la figura 34 (última columna de la derecha).
- Quitamos las barras negras verticales que se añaden por defecto y establecemos la vista en horizontal (4:3).
- Exportamos el video en la resolución correspondiente

Luego de realizar estas actividades por cada video habremos creado finalmente los 21 videos con sus características correspondientes. Cabe resaltar que se ajusta la duración de cada imagen dependiendo de la duración total del video que se quiere lograr. Por ejemplo, para crear videos de 25 segundos cada una de las 72 imágenes debe mostrarse durante 0.34 segundos en el video, de tal manera que todas las imágenes estén uniformemente distribuidas a lo largo del video. La duración final del video no es exacta dada las limitaciones de la herramienta, pero se aproxima a la duración que se quiere lograr.

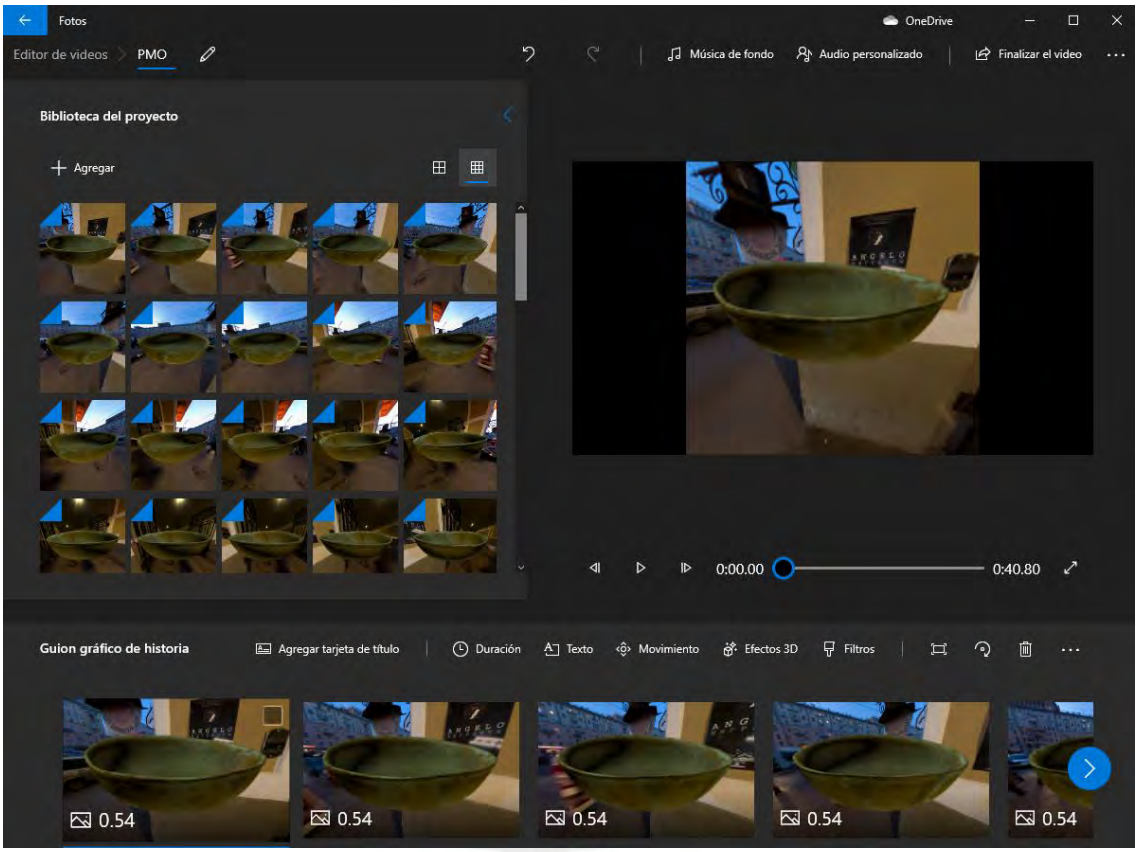


Figura 35. Editor de videos de la aplicación Fotos de Windows 10

5.2.2.3 Medio de verificación e IOV

Tabla 22. Medio de verificación e IOV del RE5 (tomado del Capítulo 1)

Resultado Esperado	Medio de verificación	Indicador objetivamente verificable

R5. Videos construidos que simulan grabaciones reales de piezas arqueológicas	Los archivos de video generados	21 archivos MP4 generados cuyo contenido son videograbaciones a piezas arqueológicas

A continuación, verificamos la existencia de estos 21 videos cuyo fin es simular videograbaciones reales a piezas arqueológicas. En la siguiente figura podemos observar dichos archivos MP4 almacenados en el computador. El total de videos tiene una duración de 8 minutos con 44 segundos y un tamaño de 73.6 MB. En conclusión, es posible afirmar que se cumplió con el IOV planificado.

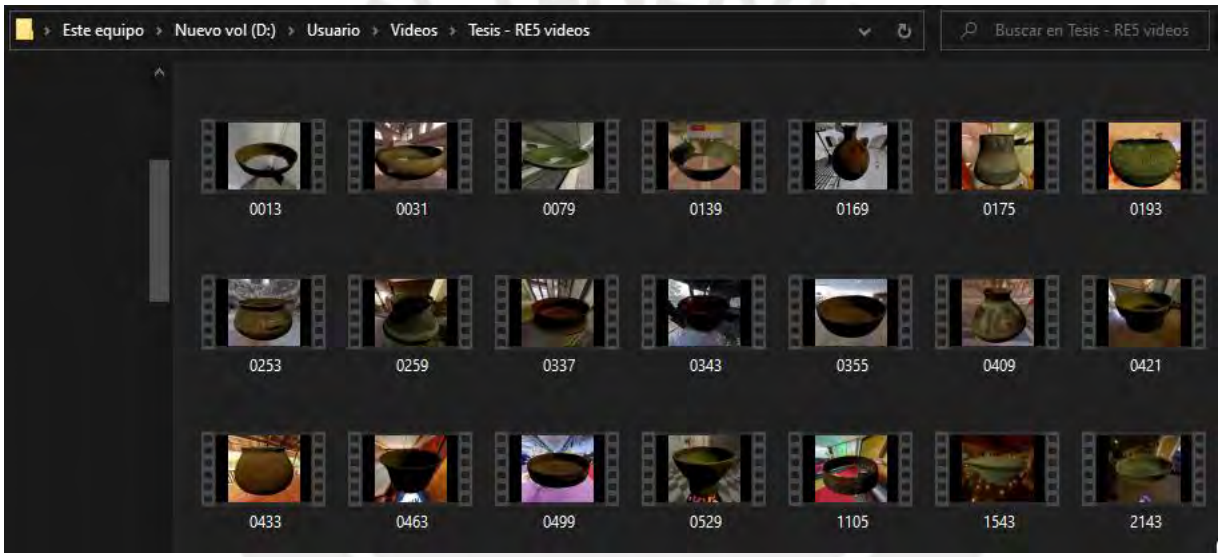


Figura 36. Videograbaciones creadas

5.2.3 *Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada

5.2.3.1 Descripción

En esta sección se describirá el resultado esperado número 6: "*Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada". Hasta ahora hemos probado los modelos de aprendizaje de máquina con múltiples secuencias de imágenes convertidas en archivos NPY. Sin embargo, parte del objetivo final de este

proyecto es que la herramienta pueda ser utilizada sobre videograbaciones reales de piezas arqueológicas. Para ello, es indispensable desarrollar y definir una serie de actividades o "*pipeline*" que nos permita convertir videos en archivos NPY que puedan ser procesados por la herramienta. En este *pipeline* se definen cuestiones como la restricción de formatos permitidos, las resoluciones y duraciones recomendadas, la extracción de los fotogramas del video como archivos PNG independientes, el proceso de selección de fotogramas, y la conversión final de todas estas imágenes en un solo archivo NPY.

5.2.3.2 Métodos y procedimientos empleados

Tabla 23. Herramientas para el RE6 (tomado del Capítulo 1)

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos
R6. *Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada	Definición de un proceso o *pipeline* que permita la conversión del archivo de video *input* a un archivo NPY procesable por los modelos entrenados. Establecer los parámetros permitidos para los videos como la resolución, duración, formatos y selección de fotogramas clave. Finalmente, plasmar este *pipeline* en código.	• Python • Sublime Text 3 • FFmpeg

En esta sección introducimos la herramienta FFmpeg, la cual nos permite trabajar con archivos de audio y video, crearlos, modificarlos y convertirlos mediante la ejecución de comandos. Lo primero que haremos será definir los formatos (extensiones de archivos) permitidos para los videos *input* que serán procesados por la herramienta. Para ello, se eligieron los formatos MP4 y AVI debido a que son 2 formatos muy comunes entre la mayoría de dispositivos electrónicos o cámaras fotográficas. Por lo tanto, el primer paso de nuestro *pipeline* será verificar que los videos ingresados pertenecen a las extensiones permitidas (MP4 o AVI).

Nuestra segunda tarea será escalar el video a una resolución de 224 x 224 pixeles. Los fotogramas que extraeremos del video posteriormente deben tener dicha resolución debido a los estándares del PMO y lo establecido en el tercer resultado esperado. Para lograr esta tarea

debemos ejecutar el siguiente comando, donde "*input*" es el nombre del archivo (incluyendo la extensión) del video, mientras que *output* es el nombre del video a generarse con la resolución especificada.

ffmpeg -i [input] -vf scale=224:224 [output]

Luego de ejecutar dicho comando obtenemos una copia del video, pero escalada a una resolución de 224 x 224 pixeles. Por lo tanto, ya tenemos definidas las resoluciones recomendadas para los videos *input* del proyecto:

- Un mínimo de 224 x 224 pixeles (lo cual volvería el escalamiento innecesario)
- Un máximo de 1080p, es decir, 1920 x 1080 pixeles. Es probable que el *pipeline* siga funcionando con resoluciones superiores, sin embargo, no se recomienda debido a que no se han planificado pruebas con resoluciones superiores. Como vimos en la sección anterior, los videos construidos tienen una máxima resolución de 1080p.

Ahora debemos proceder con el tercer paso el cual es extraer todos los fotogramas del video escalado. Para ello ejecutamos el siguiente comando, donde "*input*" en este caso es el nombre del archivo "*output*" del comando anterior.

ffmpeg -i [input] frame-%04d.png

Al ejecutar dicho comando obtendremos un set de imágenes (archivos PNG) las cuales corresponden a cada uno de los fotogramas del video. En el caso de los videos construidos en la sección anterior, estos fueron generados a una velocidad de 30 fotogramas por segundo. Es decir, para los videos de 10 segundos se generarán cerca de 300 imágenes, mientras que, para los videos de 40 segundos, más de 1200 imágenes. Además, nótese la limitación del comando al nombrar las imágenes con 4 dígitos del "0001" al "9999", es decir, cada video podrá tener una duración máxima de 9999 fotogramas, lo cual (a 30 fotogramas por segundo) equivale a

333.3 segundos, o 5 minutos con 33.3 segundos: duración máxima que podrá tener un video *input* de la herramienta desarrollada.

El cuarto paso es elegir los fotogramas que vamos a procesar de todo el set de imágenes. Recordemos que los modelos de aprendizaje de máquina pueden procesar arreglos de la forma (72, 224, 224, 4) guardados como archivos NPY, donde el primer componente hace referencia a la cantidad de imágenes, los siguientes 2 componentes a la resolución y el último componente a la información RGB en cada píxel. En otras palabras, debemos elegir los 72 fotogramas que mejor representen al video, recuperándolos del set de imágenes que acabamos de generar. Para lograrlo vamos a elegir aquellas 72 imágenes que estén igualmente distribuidas a lo largo de la duración del video. Para entender el proceso de selección de fotogramas véase la siguiente figura.

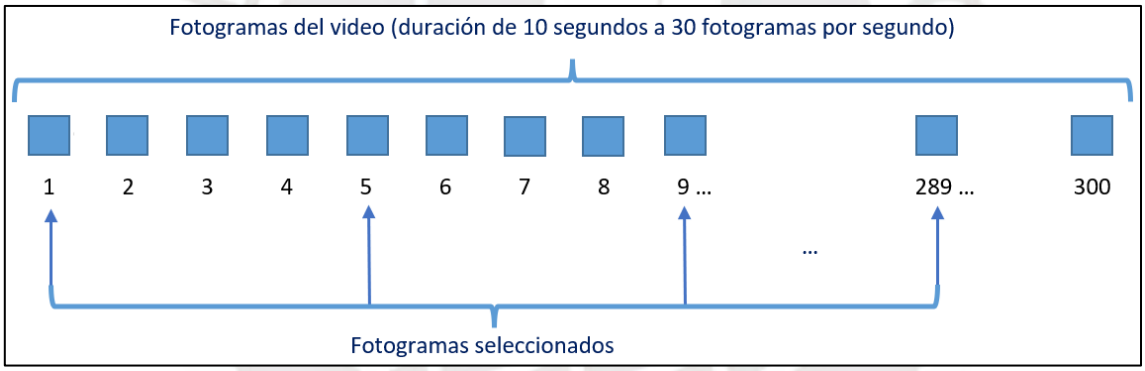


Figura 37. Ejemplo de selección de fotogramas (elaboración propia)

Observamos que elegimos el primer fotograma, luego aumentamos en 4 y elegimos ese fotograma. El patrón se repite hasta un máximo de 72 fotogramas seleccionados, siendo el último el fotograma número 289 (72 * 4 + 1 = 289). El incremento se halla aplicando la siguiente fórmula.

$$\text{Incremento} = \text{Función piso}\ (\text{Total de fotogramas} / 72)$$

En este caso, el video cuenta con 300 fotogramas, por lo tanto:

Incremento = Función piso (300 / 72) = Función piso (4.16) = 4

En otro ejemplo, poniendo como caso un video de 40 segundos de duración a 30 fotogramas por segundo, tendríamos un total de 1200 fotogramas.

Incremento = Función piso (1200 / 72) = Función piso (16.6) = 16

Nótese que si en vez de usar la función piso redondeáramos normalmente los valores tendríamos un incremento de 17. Esto en la práctica nos llevaría a seleccionar el último fotograma 1225 (72 fotogramas * 17 incrementos + 1 fotograma inicial), lo cual no es posible debido a que el video solo tiene un total de 1200 fotogramas. Es por ello que usamos siempre la función piso para redondear el incremento.

También debemos notar que necesitamos al menos 72 fotogramas para poder llevar a cabo este proceso de selección. Es decir, los videos *input* deben tener una duración mínima de 2.4 segundos (a 30 fotogramas por segundo). Con ello hemos hallado el rango de duración permitido de los videos *input*:

- Un mínimo de 2.4 segundos (72 fotogramas a 30 fotogramas por segundo)
- Un máximo de 5 minutos con 33.3 segundos (9999 fotogramas a 30 fotogramas por segundo)

Con las 72 imágenes seleccionadas debemos repetir parte del proceso definido en el *script* "*create_seq.py*" desarrollado en el tercer resultado esperado.

- Guardar las imágenes en un arreglo de la librería NumPy de la forma (72, 224, 224, 4)
- Procesar los valores RGB del arreglo, convirtiéndolos (de valores reales que varían del 0 al 1) a valores enteros que varíen del 0 al 255
- Guardar el arreglo final como un archivo NPY

Finalmente, con todas la serie de actividades definidas tenemos un *pipeline* de conversión de videograbaciones a archivos NPY. Este *pipeline* se ha planteado visualmente en el siguiente gráfico, donde finalmente obtenemos un archivo NPY (por cada video *input*) que puede ser procesado por algún modelo de aprendizaje de máquina para la reconstrucción de la pieza arqueológica. Además, se plasman dichas actividades en un *script* denominado "*pipeline.py*" el cual no podemos ejecutar todavía, pero cuyo código usaremos en el siguiente resultado. Debido a la extensión del *script* este se encuentra como el Anexo I.

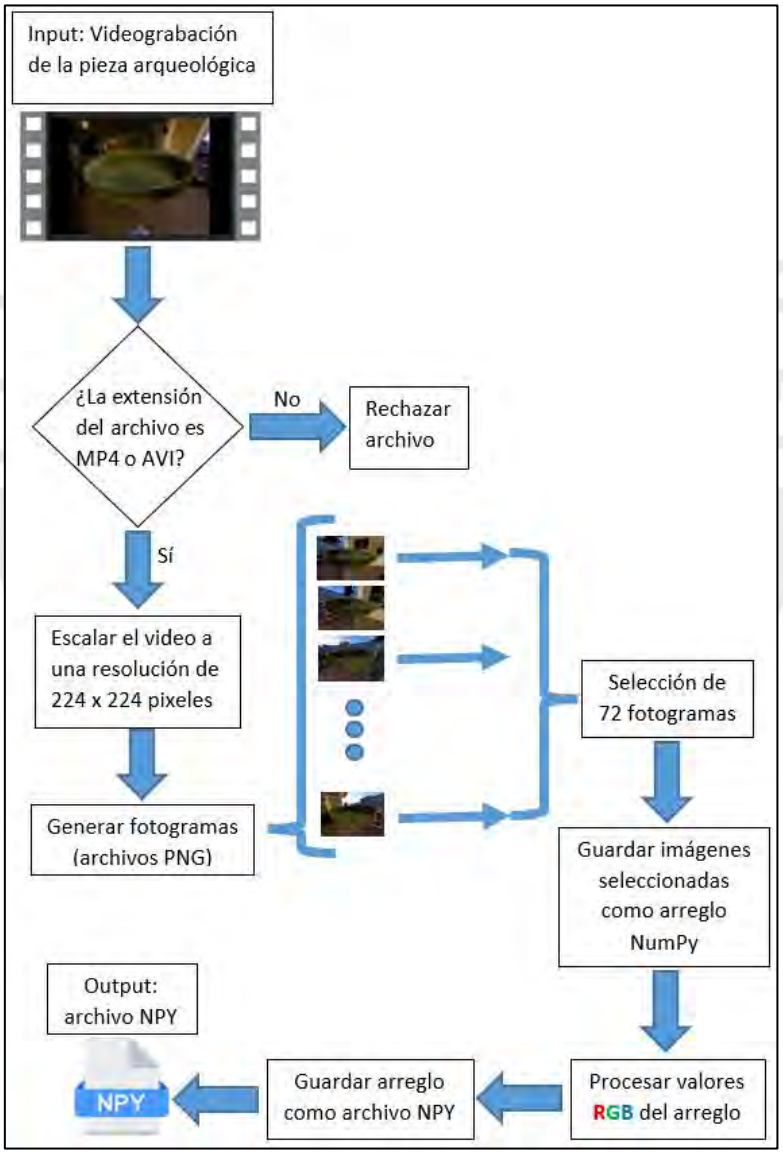


Figura 38. *Pipeline* para la conversión de videos a archivos NPY (elaboración propia)

5.2.3.3 Medio de verificación e IOV

Tabla 24. Medio de verificación e IOV del RE6 (tomado del Capítulo 1)

Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R6. *Pipeline* para la conversión de videos a secuencias que puedan ser procesadas por la herramienta desarrollada	Flujo y *script* desarrollados que definen el *pipeline* de conversión de videos a archivos NPY	• 1 flujo desarrollado • 1 *script* desarrollado

Para concluir con esta sección evidenciamos la existencia del flujo definido del *pipeline*, habiendo entendido la teoría detrás de los pasos a seguir y las herramientas que usaremos se presenta dicho flujo como un gráfico en la figura 38. Asimismo, también se presenta el *pipeline* a nivel de código como un procedimiento que recibe como parámetros:

- El nombre del archivo del video a ser procesado (se verifica extensiones permitidas)
- La categoría a la que pertenece la pieza arqueológica que está siendo grabada, entendiendo por categoría a los subconjuntos definidos en el cuarto resultado esperado ("*cone-vase*", "*bowl*", "*jar*", "lebrillo", "olla", "*plate*", "*vessel*" o "general", siendo el último la unión de todos los anteriores o el 2° experimento desarrollado)

Dicho procedimiento retorna un mensaje ya sea de éxito o de error con motivo por el cual no se pudo llevar a cabo el procesamiento del video. Se finaliza el código haciendo un llamado al proyecto PMO desarrollado con el archivo NPY recién generado y empleando el modelo de aprendizaje de máquina correspondiente a la categoría ingresada como parámetro. No es necesario volver a mencionar lo que realiza el código a detalle puesto que ello ya lo menciona el gráfico 38. En conclusión, dado el *pipeline* presentado como un flujo y un *script* anexado se puede afirmar que se cumplió exitosamente con el IOV planificado.

5.3 Discusión

Recordando el segundo problema causa definido en el árbol de problemas encontramos: “Los métodos de escaneo sin contacto – pasivos no consideran la gran variedad de huacos peruanos y poca disponibilidad de estos”. Como pudimos apreciar en el capítulo anterior, los primeros resultados obtenidos no fueron precisos. Esto se debió principalmente a la gran variedad de huacos peruanos que existen incluso dentro del set proporcionado por IA-PUCP de 962 piezas arqueológicas. Otro problema era la falta de volumen de datos, ya que otras categorías del PMO cuentan con al menos 4000 objetos, mientras que la nuestra es de 962. Para resolver este problema causa se han desarrollado 3 resultados (4°, 5° y 6°).

En el cuarto resultado atacamos directamente los problemas de variabilidad y cantidad aplicando el método de *data augmentation* y realizando 2 experimentos que nos permitan ampliar las posibilidades de reconstrucción de objetos 3D. Además, existió una segmentación interna y manual bajo las 962 piezas arqueológicas la cual nos ayudó a identificar las características principales que comparten ciertos subconjuntos de huacos peruanos. Con todo ello finalmente contamos con múltiples modelos de aprendizajes de máquina (una por subconjunto y otra que engloba a todos).

En el quinto resultado construimos videos que simulan grabaciones reales a piezas arqueológicas. Si bien es cierto ello no ataca directamente el problema causa, fue una actividad necesaria para probar los modelos desarrollados en situaciones reales y considerando ciertos rangos de resolución y duración de videos. En el sexto y último resultado definimos un *pipeline* que nos permite convertir las videograbaciones en archivos NPY procesables por los modelos entrenados en el cuarto resultado. Este *pipeline* plasmado en código inclusive nos permite identificar los pasos a seguir para probar nuestra herramienta de escaneo con videograbaciones reales.

Una limitante son las restricciones establecidas para los videos *input* de la herramienta (formatos de archivo, duración y resolución). Estas restricciones son reglas que el usuario debe seguir para obtener resultados. Sin embargo, en general se logró resolver el segundo problema causa pues ahora contamos con un método de escaneo 3D sin contacto – pasivo desarrollado que a su vez considera la gran variedad de huacos peruanos presentes en el set inicial de datos y la poca disponibilidad o cantidad de estos.

Capítulo 6. Desarrollo de una interfaz para la interacción entre el usuario y la herramienta de escaneo 3D de piezas arqueológicas

6.1 Introducción

En el presente capítulo se describe el proceso desarrollado asociado al objetivo específico número 3: "Implementar un medio de interacción entre el usuario y la herramienta desarrollada para el escaneo 3D de piezas arqueológicas". Este último objetivo cuenta únicamente con un resultado esperado en el cual se desarrolla una interfaz gráfica que permita al usuario interactuar amigablemente con la herramienta de reconstrucción 3D de piezas arqueológicas. En general, para desarrollar la interfaz se requiere también de servicios web los cuales son alojados en el servidor de IA-PUCP. Estos servicios recepcionan el archivo de video, ejecutan el *pipeline* definido en el resultado esperado n°6 y emplean el proyecto PMO (modelos entrenados) para llevar a cabo las reconstrucciones. Finalmente, la interfaz permite visualizar y descargar las mallas poligonales resultantes, además de otras funcionalidades notables.

6.2 Resultados Alcanzados

6.2.1 Interfaz gráfica integrada con la herramienta desarrollada y validada mediante pruebas funcionales

6.2.1.1 Descripción

En esta sección se describirá el resultado esperado número 7: “Interfaz gráfica integrada con la herramienta desarrollada y validada mediante pruebas funcionales”. En este último resultado nos corresponde desarrollar una interfaz gráfica la cual facilitará al usuario la interacción con la herramienta desarrollada a lo largo del proyecto. Para poder llevar a cabo esta tarea necesitamos construir un *front-end* (interfaz) y un *back-end* (servicios web). El primero se encargará principalmente de recepcionar las videograbaciones del usuario y poner a disposición los resultados (mallas poligonales), más adelante se detallarán a profundidad todas las funcionalidades desarrolladas. Segundo, el *back-end* se encargará de procesar los videos y emplear los modelos de aprendizaje de máquina para generar las mallas poligonales.

Para la implementación del *front-end* se utilizó Vue JS como *framework* de Java*Script* debido a que cuenta con librerías como “vue-3d-model” que permiten la visualización de objetos 3D en el navegador web. Mientras que para el desarrollo del *back-end* se empleó Flask como *framework* de Python, ya que nos permite crear un ámbito para los servicios web que son consumidos desde la interfaz. Además, se utiliza el servidor IA-PUCP para alojar dicho *back-end*, mientras que el *front-end* se despliega localmente en el computador del usuario. Finalmente, para corroborar el correcto funcionamiento de la herramienta se llevan a cabo pruebas funcionales unitarias.

6.2.1.2 Métodos y procedimientos empleados

Tabla 25. Herramientas para el RE7 (tomado del Capítulo 1)

Resultado Esperado	Actividad para la obtención del resultado	Herramientas y/o métodos

R7. Interfaz gráfica integrada con la herramienta desarrollada y validada mediante pruebas funcionales	Implementación de una interfaz gráfica y con los servicios web correspondientes alojados en el servidor de IA-PUCP. Integrar la interfaz con el proyecto desarrollado y probar las funcionalidades mediante pruebas funcionales unitarias.	• PMO • Python • Sublime Text 3 • PuTTY • Visual Studio Code • VueJS • Flask

Para empezar, se explicarán las 2 vistas desarrolladas y todas las funcionalidades incluidas en la implementación de la interfaz. La primera vista denominada vista principal se muestra en la siguiente figura. Como podemos observar en la sección izquierda encontramos 3 componentes. A continuación, explicaremos cada uno de ellos empezando de arriba hacia abajo.

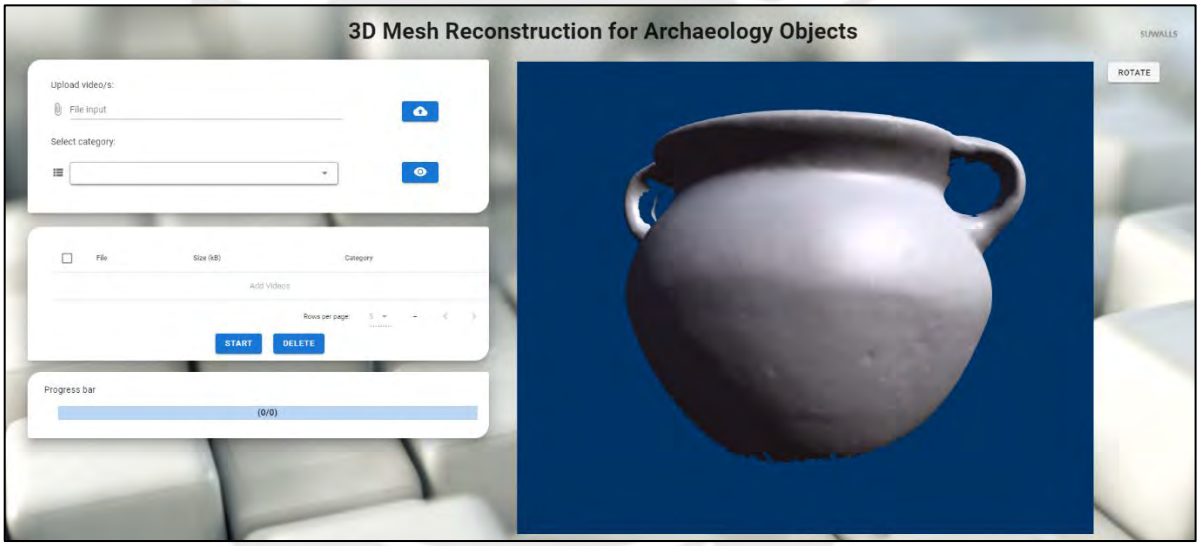


Figura 39. Vista principal de la interfaz desarrollada

Figura 40. Primer componente de la vista principal

El primer componente (figura previa) contiene 4 elementos descritos a continuación:

- *File input*: permite al usuario seleccionar 1 o más archivos, en este caso, los videos
- Botón azul superior: sube los archivos seleccionados al servidor IA-PUCP luego de realizar algunas validaciones internas (archivos seleccionados existentes, categoría seleccionada y extensiones de archivo permitidas)
- *Combo box*: permite al usuario seleccionar la categoría correspondiente a su video o conjunto de videos
- Botón azul inferior: permite observar un ejemplo (malla poligonal) de la categoría seleccionada en el visualizador de objetos 3D ubicado en la sección derecha de la vista

Por lo tanto, con este primer componente el usuario será capaz de subir sus videograbaciones al servidor IA-PUCP asignándoles una categoría y pudiendo visualizar ejemplos de estas categorías utilizando el visualizador de objetos 3D ubicado en la sección derecha de la vista.

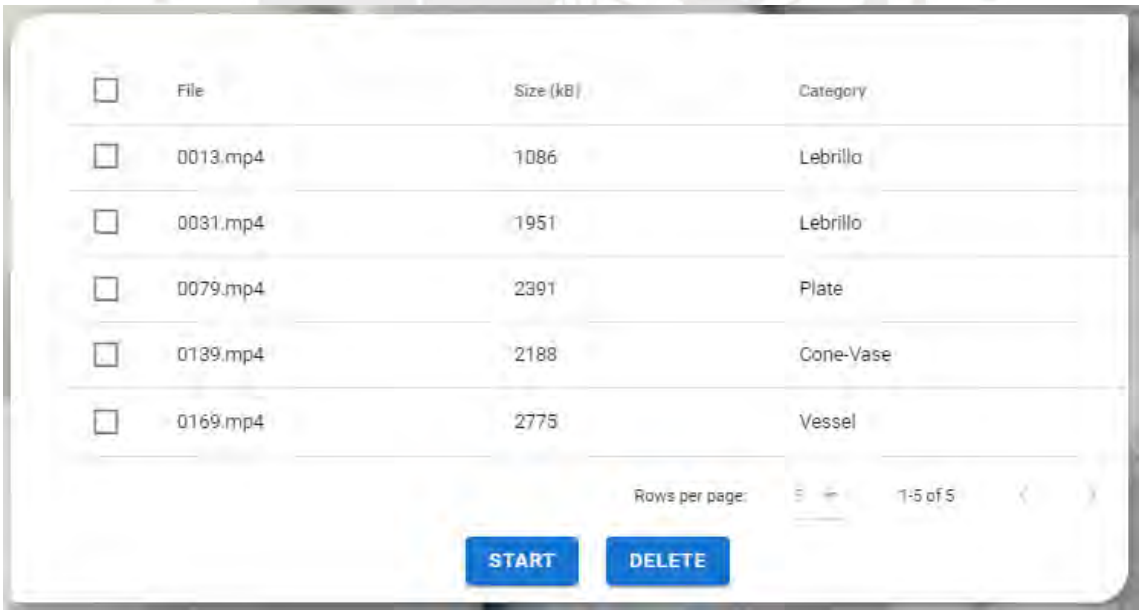


Figura 41. Segundo componente de la vista principal

En este segundo componente encontramos 3 elementos:

- Tabla de datos: en esta tabla encontramos todos los videos agregados hasta el momento con los detalles del nombre del archivo, el tamaño en kB y la categoría asignada
- Botón azul izquierdo: al hacer *click* se inicia el procesamiento (reconstrucción de objetos 3D) de todos los videos que se encuentren en la tabla
- Botón azul derecho: elimina los videos seleccionados de la tabla de datos para descartarlos del procesamiento.

Por lo tanto, con este segundo componente tenemos la opción de controlar todos los videos agregados hasta el momento en una tabla de datos y eliminar a aquellos que queramos descartar. Además, podemos empezar el procesamiento de reconstrucción.

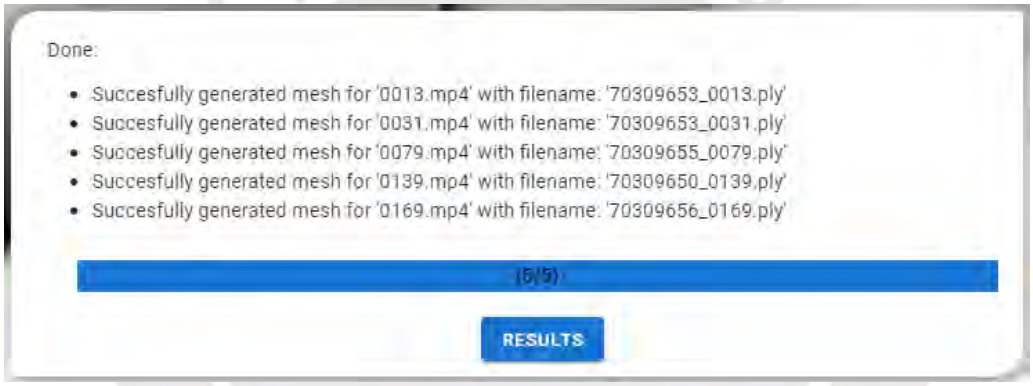


Figura 42. Tercer componente de la vista principal

En este tercer componente encontramos 3 elementos:

- Barra de progreso: esta barra nos indica el progreso general del procesamiento de todos los videos, usualmente el proceso tarda 5 minutos por video
- Mensajes/logs: cada vez que se termina de procesar un video se muestra un mensaje el cual nos indica el estado final del procesamiento del video, al finalizar todo se tendrá 1 mensaje por video
- Botón azul central: al finalizar el procesamiento de todos los archivos se revelará un botón azul central, el cual (al hacer *click*) nos conduce a la vista de resultados

En la siguiente figura observamos la vista de resultados. De manera análoga, se explicarán los 2 componentes presentes en la vista.

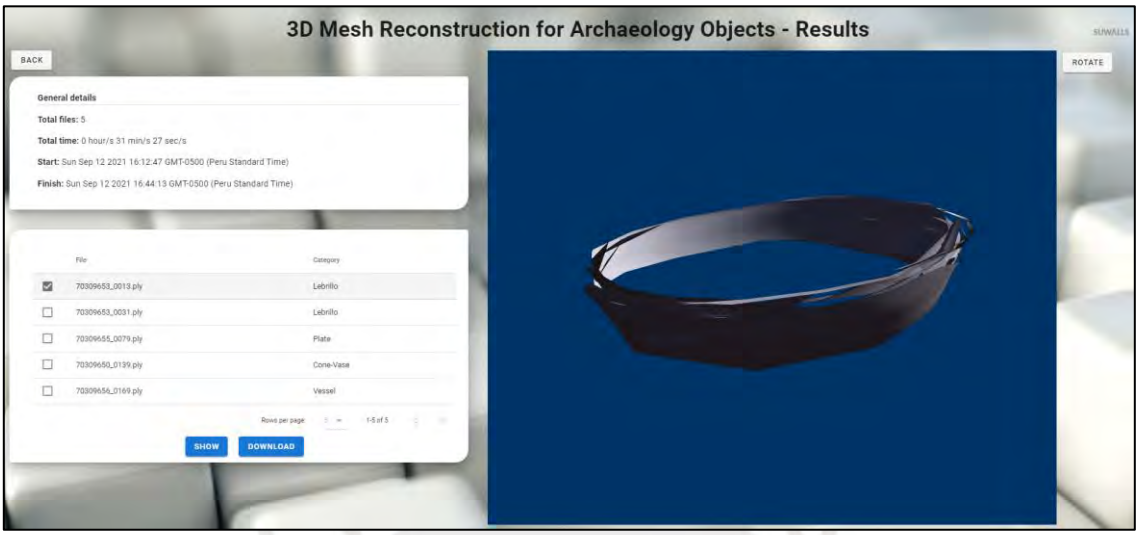


Figura 43. Vista de resultados de la interfaz desarrollada

El primero componente no requiere de interacción por parte del usuario. Este simplemente muestra los detalles generales del procesamiento con datos como: el total de archivos procesados, tiempo total transcurrido, momento (fecha y hora) del inicio y fin de la tarea. Véase la siguiente figura para un mayor detalle.

Figura 44. Primer componente de la vista de resultados

En el segundo componente (siguiente figura) encontramos 3 elementos:

- Tabla de datos: en esta tabla encontramos listados todos los archivos de mallas poligonales (formato PLY) que se han generado a partir de los videos, asimismo, la categoría asignada a cada una de ellas
- Botón azul izquierdo: al seleccionar una de las mallas poligonales de la tabla y hacer *click* en este botón observaremos el resultado en el visualizador de objetos 3D de la sección derecha de la vista
- Botón azul derecho: al hacer *click* descargaremos todas las mallas poligonales en un solo archivo comprimido

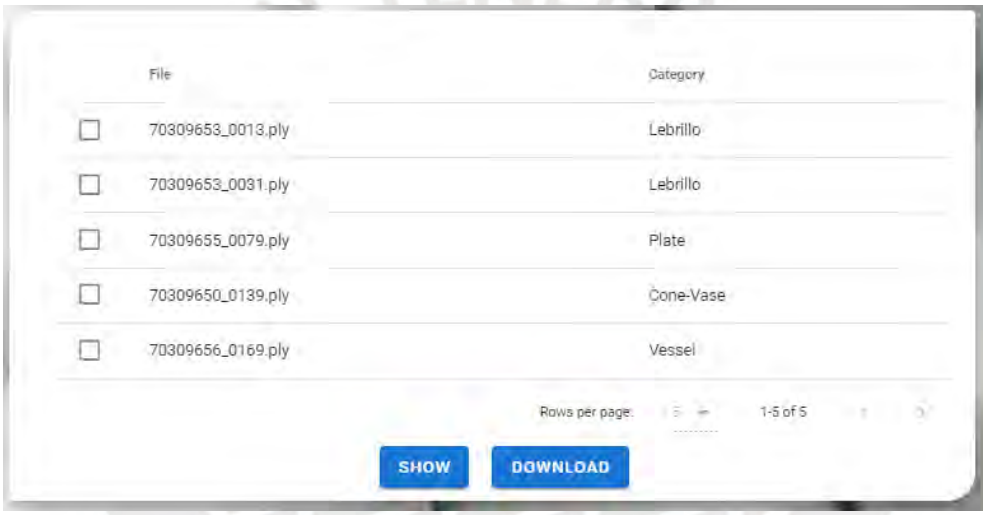


Figura 45. Segundo componente de la vista de resultados

Finalmente, también encontramos otros elementos externos a los componentes. El botón blanco izquierdo nos permite regresar a la vista principal, mientras que el botón blanco derecho nos permite rotar o dejar de rotar el objeto 3D. Además, como habremos podido notar en ambas vistas encontramos un visualizador básico de objetos 3D en la sección derecha (véase la siguiente figura). Este visualizador no siempre representa precisamente a la malla poligonal obtenida, por lo tanto, se recomienda utilizarlo únicamente como una primera impresión del resultado obtenido.

Figura 46. Visualizador de objetos 3D de la interfaz desarrollada

Hasta este punto ya se exhibieron todas las funcionalidades de la interfaz y se procederá a explicar todos los servicios web desarrollados. Estos servicios están alojados en el servidor IA-PUCP donde también se encuentra nuestro proyecto de reconstrucción de objetos elaborado previamente. La interfaz enviará los archivos de video al servidor IA-PUCP el cual seguirá el *pipeline* definido en el resultado específico 6 y los resultados serán reenviados a la interfaz nuevamente para que el usuario pueda observarlos y descargarlos.

El primer servicio web desarrollado se denomina "*upload_video*". Este realiza las siguientes tareas:

- Recepcionar el conjunto de videos cargado por el usuario en la interfaz
- Por cada uno de los videos se realiza lo siguiente:
    - Asegurarse de que el formato del video sea permitido (AVI o MP4)
    - Asegurarse de que el nombre del archivo no representa ningún daño potencial al servidor
    - Guardar el video en una carpeta específica en el servidor IA-PUCP
    - Añadir el nombre del archivo a un arreglo de nombres
- Unir todos los nombres de los archivos en una sola cadena de caracteres separada por espacios

- Retornar dicha cadena de caracteres a la interfaz. Esta cadena nos servirá para determinar qué archivos fueron subidos al servidor correctamente.

El segundo servicio se denomina "*process_mesh*" y realiza lo siguiente:

- Recibe como parámetros el nombre del archivo de video a procesar y la categoría asginada.
- A continuación, ejecuta todo el *pipeline* desarrollado en el resultado esperado anterior. Para un mayor detalle observar la figura 38. En general se ejecuta lo siguiente:
    - Definir las rutas a los diferentes archivos y carpetas que serán afectadas
    - Asegurarse nuevamente de que los formatos estén permitidos
    - Asegurarse de que el rango de fotogramas del video sea permitido (recordemos que mínimo se necesitan 72 fotogramas y máximo 9999)
    - Escalar el video a 224 x 224 pixeles
    - Generar los fotogramas como imágenes independientes
    - Proceso de selección de los 72 fotogramas
    - Guardar las imágenes seleccionadas como un arreglo NumPy
    - Procesar los valores RGB del arreglo NumPy
    - Guardar el arreglo como un archivo NPY
    - Editar el archivo "*all_test.list*" del proyecto PMO para añadir la secuencia creada
    - Ejecutar el comando de reconstrucción 3D de objetos empleando los modelos de aprendizaje de máquina entrenados anteriormente y el archivo NPY que acabamos de generar a partir del video.

Cabe resaltar que los resultados del procesamiento de cada video se guardan como archivos PLY o malla de polígonos en una carpeta específica donde posteriormente serán recuperadas. El tercer servicio desarrollado se denomina "*get_mesh*" y realiza:

- Recibe como único parámetro el nombre del archivo a recuperar
- Retorna el archivo encontrado (en este caso OBJ o PLY, que son objetos 3D representados en el computador) a la interfaz.

El cuarto y último servicio desarrollado se denomina "*get_zip*" y realiza lo siguiente:

- Recibir como parámetros una cadena de caracteres que contiene todos los nombres de los archivos o mallas poligonales que se quieren descargar, separados por un carácter especial
- Borrar el archivo ZIP previo de alguna descarga anterior
- Crear una nueva carpeta
- Copiar todos los archivos a descargar dentro de la carpeta
- Generar un archivo ZIP al comprimir la carpeta creada
- Borrar la carpeta creada
- Retornar el archivo ZIP que contiene todas las mallas poligonales

Con todos estos servicios incluidos en un *script* denominado "app.py" en el servidor de IA-PUCP podemos levantar la instancia de Flask ejecutando el siguiente comando.

flask run --host=0.0.0.0

Ahora desplegamos localmente la interfaz haciendo uso del *framework* Vue JS y finalmente accedemos al mismo mediante nuestro navegador web. Con estos pasos finales tendremos la herramienta a disposición del usuario. Para evaluar las funcionalidades de la interfaz se llevaron a cabo una serie de pruebas funcionales que se listan a continuación.

1. Añadir un video o un conjunto de videos
2. Seleccionar y observar los ejemplos de las categorías en el visualizador de objetos 3D
3. Eliminar un video de la tabla de datos de videos añadidos y empezar el procesamiento de los videos restantes

4. Descargar los resultados obtenidos a partir del procesamiento de los videos

Para cada una de las pruebas se definió el objetivo, precondición y se documentaron los resultados obtenidos. Cabe resaltar que se emplearon los 21 videos construidos en el resultado esperado número 5 para llevar a cabo todas las pruebas. Los resultados se encuentran documentados en el Anexo J.

6.2.1.3 Medio de verificación e IOV

Tabla 26. Medio de verificación e IOV del RE7 (tomado del Capítulo 1)

Resultado Esperado	**Medio de verificación**	**Indicador objetivamente verificable**
R7. Interfaz gráfica integrada con la herramienta desarrollada y validada mediante pruebas funcionales	Interfaz gráfica funcional	100% de pruebas funcionales concluidas con éxito

Si exploramos el anexo de las pruebas funcionales observaremos que todas estas se han concluido exitosamente, evidenciando así el correcto funcionamiento de todos los elementos previstos para la interfaz. En conclusión, sí se cumple con el IOV planificado.

6.3 Discusión

Recordando el tercer problema causa definido en el árbol de problemas encontramos: “Se requiere de un contacto físico directo y manipulación de las piezas arqueológicas para llevar a cabo el proceso de escaneo 3D”. Para resolver este problema se debe desarrollar una herramienta que permita el escaneo 3D de piezas arqueológicas sin la necesidad de un contacto físico directo. Tal es el caso del producto final obtenido en este capítulo, puesto que las actividades requeridas por el usuario son: grabar a los objetos bajo ciertos parámetros y estándares, cargar los videos a la interfaz gráfica, empezar el procesamiento y descargar los resultados. Las limitantes presentes son: el tiempo de procesamiento el cual asciende a aproximadamente 5 minutos por objeto y la primera impresión básica del visualizador 3D de la interfaz. A pesar de todo ello, el usuario puede obtener mallas poligonales como resultados

a partir de sus videograbaciones. Esto nos indica que hemos sido capaces de resolver este tercer problema causa, ya que con este proyecto se prescinde de la manipulación física durante el proceso de escaneo.

Capítulo 7. Conclusiones

7.1 Conclusiones

El problema principal identificado al inicio de este proyecto de investigación es que el proceso tradicional de escaneo 3D de piezas arqueológicas pone en riesgo la preservación física de estas. Recordemos que para llevar a cabo un proceso de escaneo 3D de cualquier objeto es necesario moverlo, manipularlo y rotarlo múltiples veces, lo cual no es recomendable en objetos frágiles como las piezas arqueológicas. Para resolver esta problemática se desarrolló una herramienta de software que nos permite escanear en 3D huacos peruanos usando las técnicas de reconstrucción de objetos mediante visión artificial, aprendizaje de máquina, *data augmentation* y mallas poligonales. Ahora que contamos con tal herramienta, el usuario podrá ingresar sus videograbaciones (bajo ciertos estándares predefinidos) a la interfaz gráfica, esperar el tiempo de procesamiento y obtener resultados los cuales podrá visualizar en la propia interfaz y descargar los archivos de mallas poligonales.

Mediante la herramienta desarrollada logramos resolver el problema central, ya que se vuelve posible la obtención de reconstrucciones 3D de las piezas arqueológicas sin necesidad de interactuar físicamente con ellas. Esto quiere decir que se cumple con el objetivo general del proyecto. Sin embargo, la limitación más importante presente es la precisión de superficies reconstruidas cuando se trata de objetos más complejos, como el caso de las vasijas con mango, lebrillos con doble fondo, botellas en forma de animales o cabezas, etc. Estas formas son muy particulares y por ende no se cuenta con la data suficiente como para entrenar un nuevo modelo de aprendizaje de máquina con dicha data. En estos casos, es mejor aproximar la forma

utilizando la categoría general compuesta por todo el set inicial de datos de 962 piezas arqueológicas posterior a la segmentación manual realizada.

En conclusión, se logró cumplir exitosamente con el objetivo general del proyecto y con cada uno de los 7 resultados esperados. Esto pudo lograrse primero gracias a la disposición de un set inicial de 962 piezas arqueológicas pre-escaneadas proporcionado por el grupo de Inteligencia Artificial PUCP. Segundo, se lleva a cabo la generación, organización y segmentación de imágenes mediante *script*s y modificación a nivel de código de algunas herramientas externas. Tercero, se entrena un modelo de aprendizaje de máquina con la data generada para la obtención de unos primeros resultados y empleando el proyecto base PMO. Cuarto, se aplica la técnica de *data augmentation* para extender nuestra data y se desarrollan múltiples *script*s para la normalización y procesamiento de esta. Quinto, se construyen videos que simulan situaciones reales de videograbaciones para probar nuestros modelos entrenados. Sexto, se define un *pipeline* que convierta estos videos en archivos procesables por los modelos y el PMO, ello implica una selección de fotogramas claves de cada video. Finalmente, se desarrolla una interfaz gráfica cuyas funcionalidades se comprueban mediante pruebas funcionales y servicios web alojados en el servidor de IA-PUCP. Todo ello nos permite afirmar que se cumple con el objetivo general y que ahora contamos con la herramienta propuesta.

7.2 Trabajos futuros

Como primer trabajo futuro se propone ampliar el set inicial compuesto por 962 huacos peruanos pre-escaneados a un mínimo de 4000 piezas arqueológicas distintas. Aumentar este conjunto nos permitirá complementar las categorías definidas con la segmentación manual de los datos, definir nuevas categorías existentes, permitir que los modelos de aprendizaje de máquina reconozcan mejor las características compartidas entre los objetos y por ende aumentar la precisión las reconstrucciones. Este trabajo futuro representa una gran ventaja para la mejora de este proyecto.

Un segundo trabajo futuro es disminuir las restricciones de las videograbaciones *input* de los usuarios. Actualmente los únicos formatos permitidos por la interfaz son AVI y MP4. A pesar de que estos constituyen un gran volumen de videos generados por los dispositivos de grabación, el hecho de aumentar el rango de extensiones permitidas representa una gran ventaja para los usuarios que trabajan con otras extensiones como MOV, MKV, WMV, FLV, etc., ya que se prescinde de un proceso previo de conversión de archivos a MP4 o AVI.

Finalmente, un tercer trabajo futuro sería generar las mallas poligonales resultantes en diferentes formatos de archivo. En el producto final desarrollado las reconstrucciones de huacos peruanos se obtienen como archivos PLY. Sin embargo, si modificamos el código de generación de las mallas podemos incluir nuevos formatos de archivos conocidos como OBJ y STL, los cuales también sirven para almacenar objetos 3D en el computador. Además, en la interfaz gráfica podría añadirse una funcionalidad para que el usuario decida en qué formato desea que se generen los resultados. Esto representaría una ventaja para los usuarios que van a post-procesar los resultados y necesitan formatos específicos de archivos 3D, evitando así un proceso de conversión posterior.

Referencias

A. Kar, S. Tulsiani, J. Carreira, & J. Malik. (2015). Category-specific object reconstruction from a single image. *2015 IEEE Conference on Computer vision and Pattern Recognition (CVPR)*, 1966-1974. https://doi.org/10.1109/CVPR.2015.7298807

C. Bregler, A. Hertzmann, & H. Biermann. (2000). Recovering non-rigid 3D shape from image streams. *Proceedings IEEE Conference on Computer vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, *2*, 690-696 vol.2. https://doi.org/10.1109/CVPR.2000.854941

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nie\s sner, M. (2017). ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *Proc. Computer vision and Pattern Recognition (CVPR), IEEE*.

Kanazawa, A., Tulsiani, S., Efros, A. A., & Malik, J. (2018). Learning Category-Specific Mesh Reconstruction from Image Collections. *ECCV*.

Q. Tan, L. Gao, Y. Lai, & S. Xia. (2018). Variational Autoencoders for Deforming 3D Mesh Models. *2018 IEEE/CVF Conference on Computer vision and Pattern Recognition*, 5841-5850. https://doi.org/10.1109/CVPR.2018.00612

Qi, C. R., Su, H., Niessner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and Multi-View CNNs for Object Classification on 3D Data. *The IEEE Conference on Computer vision and Pattern Recognition (CVPR)*.

Qi C. R. (2016). Object Detection in 3D Scenes Using CNNs in Multi-view Images. Stanford University.

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. G. (2015). Multi-view convolutional *neural networks* for 3d shape recognition. *Proc. ICCV*.

Y. Zhou, & O. Tuzel. (2018). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *2018 IEEE/CVF Conference on Computer vision and Pattern Recognition*, 4490-4499. https://doi.org/10.1109/CVPR.2018.00472

Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, & J. Xiao. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer vision and Pattern Recognition (CVPR)*, 1912-1920. https://doi.org/10.1109/CVPR.2015.7298801

Blanz, V., & Vetter, T. (1999). A Morphable Model for the Synthesis of 3D Faces. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 187–194. https://doi.org/10.1145/311535.311556

C. Bregler, A. Hertzmann, & H. Biermann. (2000). Recovering non-rigid 3D shape from image streams. *Proceedings IEEE Conference on Computer vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, *2*, 690-696 vol.2. https://doi.org/10.1109/CVPR.2000.854941

D. Nister. (2004). Automatic passive recovery of 3D from images and video. *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, 438–445. https://doi.org/10.1109/TDPVT.2004.1335271

Forsyth, D. A., & Ponce, J. (2002). *Computer vision: A Modern Approach*. Prentice Hall Professional Technical Reference.

Hadjitheophanous, S., Ttofis, C., Georghiades, A. S., & Theocharides, T. (2010). Towards Hardware Stereoscopic 3D Reconstruction: A Real-time FPGA Computation of the Disparity Map. *Proceedings of the Conference on Design, Automation and Test in Europe*, 1743–1748. Recuperado de http://dl.acm.org/citation.cfm?id=1870926.1871347

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. En D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 346–361). Cham: Springer International Publishing.

Lin, C.-H., Wang, O., Russell, B. C., Shechtman, E., Kim, V. G., Fisher, M., & Lucey, S. (2019). *Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction*. *The IEEE Conference on Computer vision and Pattern Recognition (CVPR)*.

Menzies, R. (2014). Lidar Systems. En E. G. Njoku (Ed.), *Encyclopedia of Remote Sensing* (pp. 334–339). https://doi.org/10.1007/978-0-387-36699-9_85

Peter Seitz. (2007, junio 18). *Photon-noise limited distance resolution of optical metrology methods*. *6616*. Recuperado de https://doi.org/10.1117/12.732040

Scharstein, D., & Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. *Proceedings of the IEEE Computer Society Conference on Computer vision and Pattern Recognition*, *1*, I/195-I/202. Recuperado de https://www.scopus.com/inward/record.uri?eid=2-s2.0-0041939772&partnerID=40&md5=556c4efb08415aba1a4d2e9fb7171ad7

Shashua, A. (1992). *Geometry and Photometry in 3D Visual Recognition*. Cambridge, MA, USA: Massachusetts Institute of Technology.

Sreenivasa Kumar Mada, Melvyn L. Smith, Lyndon N. Smith, & Prema Sagar Midha. (2003, marzo 19). *Overview of passive and active vision techniques for hand-held 3D data acquisition*. *4877*. Recuperado de https://doi.org/10.1117/12.463773

Stimson, A. (1974). *Photometry and radiometry for engineers*.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. En D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 818–833). Cham: Springer International Publishing.

Boehler W., Heinz G. & Marbs A. (2001). *The potential of non-contact close range laser scanners for cultural heritage recording.* Mainz

Pears, Nick, Yonghuai LIU y Peter BUNTING

2012 *3D Imaging, Analysis and Applications.* York: Springer

Python. (s.f). *Applications for Python*. Consulta: 3 de noviembre de 2019.

https://www.python.org/about/apps/

NumPy. (s.f). *NumPy*. Consulta: 3 de noviembre de 2019.

https://numpy.org/

Vue (n.d.) Vue.js. Disponible en: https://vuejs.org/. Consulta: 7 de Setiembre del 2020

ShapeNet-Renderer (10 nov 2018). Stanford-Shapenet-Renderer. Disponible en: https://github.com/panmari/stanford-shapenet-renderer.

Museo Josefina Ramos de Cox (n.d.). Museo de Arqueología Josefina Ramos de Cox. Disponible en: https://ira.pucp.edu.pe/museo-arqueologico-josefina-ramos-cox/.

IA PUCP (n.d.). Grupo de Inteligencia Artificial de la Pontificia Universidad Católica del Perú. Disponible en: https://investigacion.pucp.edu.pe/grupo-investigacion/grupo-de-inteligencia-artificial-pucp/.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., & Yu, F. (2015). *ShapeNet: An Information-Rich 3D Model Repository* (arXiv:1512.03012 [cs.GR]). Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Groueix, T., Fisher, M., Kim, V. G., Russell, B., & Aubry, M. (2018). AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. *Proceedings IEEE Conf. on Computer vision and Pattern Recognition (CVPR)*.

Lin, C.-H., Wang, O., Russell, B. C., Shechtman, E., Kim, V. G., Fisher, M., & Lucey, S. (s. f.). *Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction*

Anexo A: Resultados de la revisión sistemática

Tabla A1. Resultados de la revisión sistemática

ID	Autor/es	Título	Año	Tipo	Fecha de extracción	Motor	Técnica	Enfoque	Problemática abordada	Métrica
R01	A. Kar, S. Tulsiani, J. Carreira, & J. Malik	Category-Specific Object Reconstruction from a Single Image	2015	CVPR	1/09/2019	IEEE Xplore	-	3D Pose Estimation	Estimación de la posición de un objeto con respecto a la cámara a partir de una serie de imágenes capturadas convenientemente siguiendo un patrón.	Precisión del modelo
R02	Qi, C. R., Su, H., Niessner, M., Dai, A., Yan, M., & Guibas, L. J.	Volumetric and Multi-View CNNs for Object Classification on 3D Data	2016	CVPR	1/10/2019	IEEE Xplore	-	Single Object Classification	Clasificación de un único objeto a partir de una secuencia de imágenes capturadas convenientemente alrededor del objeto.	Precisión del modelo
R03	Qi Charles	Object Detection in 3D Scenes Using	2016	Paper	1/09/2019	IEEE Xplore	-	Multiple Objects Detection	Detección y clasificación de múltiples objetos a partir de una única imagen panorámica que contiene una escena corriente.	Precisión del modelo

		CNNs in Multi-view Images								
R04	Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T.	ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes	2017	CVPR	1/09/2019	IEEE Xplore	-	Scene/Object Semantic Segmentation	Análisis y reconstrucción de una escena "indoor" mediante una red neuronal denominada ScanNet.	Porcentaje de reconstrucción
R05	Kanazawa, A., Tulsiani, S., Efros, A. A., & Malik, J.	Learning Category-Specific Mesh Reconstruction from Image Collections	2018	ECCV	1/10/2019	ACM	-	3D Geometry Synthesis / Reconstruction	Aprender a categorizar reconstrucciones de objetos en 3D a partir de una colección de imágenes.	Porcentaje de reconstrucción
R06	Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E.	Multi-view Convolutional Neural Networks for 3D Shape Recognition	2015	ICCV	1/10/2019	ACM	Multi-view Images	-	Reconocimiento de la profundidad de un objeto a partir de múltiples imágenes empleando el uso de una CNN.	Precisión del modelo

R07	Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, & J. Xiao	3D ShapeNets: A Deep Representation for Volumetric Shapes	2015	CVPR	1/10/2019	ACM	Volumetric	-	Desarrollo de la representación de objetos 3D mediante formas volumétricas y deep learning.	Precisión del modelo
R08	Y. Zhou, & O. Tuzel.	VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection	2018	CVPR	1/11/2019	MIT Libraries	Point Cloud	-	Desarrollo de un modelo para la reconstrucción de superficies 3D pero empleando la representación de nube de puntos.	Precisión del modelo
R09	Q. Tan, L. Gao, Y. Lai, & S. Xia	Variational Autoencoders for Deforming 3D Mesh Models	2018	CVPR	1/11/2019	MIT Libraries	Polygonal Mesh	-	Análisis de desempeño entre los diferentes enfoques de reconstrucción de superficies 3D presentados en los últimos años.	Precisiones de los diferentes modelos

R10	Z. Chuhang, E. Yumer, J. Yang, D. Ceylan & D. Hoiem	3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks	2017	Paper	1/11/2019	IEEE Xplore	Primitive-based	-	Implementación de una red neuronal que permita la generación de formas empleando una representación primitive-based.	Precisión del modelo
R11	Lin, C.-H., Wang, O., Russell, B. C., Shechtman, E., Kim, V. G., Fisher, M., & Lucey, S	Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction	2019	CVPR	1/08/2019	Springer	Polygonal Mesh	3D Geometry Synthesis/Reconstruction	Reconstrucción de objetos 3D mediante el uso de un modelo de aprendizaje de máquina, empleando una representación de mayas poligonales y videos RGB del mundo real como input.	Cobertura de forma y precisión de forma

Anexo B: Código modificado de Stanford ShapeNet Renderer

Para acceder al código modificado de la herramienta ShapeNet Renderer (ShapeNet-Renderer, 2018) se provee el siguiente enlace de Github. Este nos muestra un *script* llamado "*render_blender.py*" que contiene todas las modificaciones y parámetros que fueron añadidos para la ejecución del primer resultado esperado. Al ejecutar este *script* con los parámetros definidos se generan las imágenes empleando los 962 archivos OBJ de piezas arqueológicas escaneadas previamente.

https://github.com/ErnieLud/tesis/blob/main/render_blender.py

Anexo C: *Script* desarrollado "*write_seq.py*"

Para acceder al código desarrollado del *script* "write_seq.py" se provee el siguiente enlace de GitHub. Este *script* modifica las listas del PMO (archivos LIST) y crea 2 nuevas listas de la nueva categoría añadida con el fin de segmentar sus objetos. La distribución realizada es del 80% - 20% para los subconjuntos de entrenamiento y prueba respectivamente.

https://github.com/ErnieLud/tesis/blob/main/write_seq.py

Anexo D: *Script* desarrollado "*act_plys3.py*"

Para acceder al código desarrollado del *script* "*act_plys3.py*" se provee el siguiente enlace de GitHub. Este archivo modifica las nubes de puntos de las piezas arqueológicas y las normaliza, centra los objetos y elimina las caras existentes, creando así nuevos archivos PLY. Además, también se guardan los puntos como arreglos NumPy o archivos NPY que servirán para el entrenamiento del modelo de aprendizaje de máquina.

https://github.com/ErnieLud/tesis/blob/main/act_plys3.py

Anexo E: *Script* desarrollado "*create_seq.py*"

Para acceder al código desarrollado del *script* "*create_seq.py*" se provee el siguiente enlace de GitHub. Este *script* crea las secuencias de imágenes compuestas por piezas arqueológicas y fondos panorámicos. Posteriormente se procesan los valores RGB de la secuencia y esta se guarda como un archivo NPY que puede ser procesado por un modelo de aprendizaje de máquina para la reconstrucción del objeto 3D.

https://github.com/ErnieLud/tesis/blob/main/create_seq.py

Anexo F: *Script*s desarrollados "*subrendering.py*" y "*subset.py*"

Para acceder al código desarrollado de los *script*s "*subrendering.py*" y "*subset.py*" se proveen los siguientes enlaces de GitHub. El primer *script* crea las secuencias de imágenes compuestas para un subconjunto del universo inicial de 962 huacos peruanos aplicando la técnica de *data augmentation*. El segundo *script* realiza una tarea similar, pero para las nubes de puntos (archivos PLY) del conjunto inicial de 962 huacos.

https://github.com/ErnieLud/tesis/blob/main/subrendering.py

https://github.com/ErnieLud/tesis/blob/main/subset.py

Anexo G: Listas de objetos asignados y de prueba

Tabla G1. Escalas calculadas de los objetos

Índice	Objeto	Escala	Índice	Objeto	Escala	Índice	Objeto	Escala
1	0001.obj	0.0055 83	322	0322.obj	0.0050 83	643	0643.obj	0.0050 83
2	0002.obj	0.0055 83	323	0323.obj	0.0115 83	644	0644.obj	0.0055 83
3	0003.obj	0.0045 83	324	0324.obj	0.0050 83	645	0645.obj	0.0055 83
4	0004.obj	0.0043 83	325	0325.obj	0.0050 83	646	0646.obj	0.0060 83
5	0005.obj	0.0043 83	326	0326.obj	0.0050 83	647	0647.obj	0.0065 83
6	0006.obj	0.0050 83	327	0327.obj	0.0040 83	648	0648.obj	0.0075 83
7	0007.obj	0.0045 83	328	0328.obj	0.0050 83	649	0649.obj	0.0055 83
8	0008.obj	0.0050 83	329	0329.obj	0.0040 83	650	0650.obj	0.0075 83
9	0009.obj	0.0050 83	330	0330.obj	0.0055 83	651	0651.obj	0.0045 83
10	0010.obj	0.0047 83	331	0331.obj	0.0065 83	652	0652.obj	0.0040 83
11	0011.obj	0.0047 83	332	0332.obj	0.0055 83	653	0653.obj	0.0045 83
12	0012.obj	0.0060 83	333	0333.obj	0.0050 83	654	0654.obj	0.0045 83
13	0013.obj	0.0047 83	334	0334.obj	0.0055 83	655	0655.obj	0.0040 83
14	0014.obj	0.0043 83	335	0335.obj	0.0060 83	656	0656.obj	0.0065 83
15	0015.obj	0.0047 83	336	0336.obj	0.0055 83	657	0657.obj	0.0070 83
16	0016.obj	0.0041 83	337	0337.obj	0.0050 83	658	0658.obj	0.0075 83
17	0017.obj	0.0047 83	338	0338.obj	0.0055 83	659	0659.obj	0.0060 83
18	0018.obj	0.0040 83	339	0339.obj	0.0060 83	660	0660.obj	0.0055 83
19	0019.obj	0.0047 83	340	0340.obj	0.0040 83	661	0661.obj	0.0075 83
20	0020.obj	0.0057 83	341	0341.obj	0.0035 83	662	0662.obj	0.0055 83
21	0021.obj	0.0057 83	342	0342.obj	0.0045 83	663	0663.obj	0.0055 83
22	0022.obj	0.0057 83	343	0343.obj	0.0040 83	664	0664.obj	0.0050 83
23	0023.obj	0.0053 83	344	0344.obj	0.0055 83	665	0665.obj	0.0055 83
24	0024.obj	0.0050 83	345	0345.obj	0.0040 83	666	0666.obj	0.0050 83
25	0025.obj	0.0049 83	346	0346.obj	0.0045 83	667	0667.obj	0.0055 83
26	0026.obj	0.0060 83	347	0347.obj	0.0065 83	668	0668.obj	0.0050 83
27	0027.obj	0.0070 83	348	0348.obj	0.0045 83	669	0669.obj	0.0050 83
28	0028.obj	0.0065 83	349	0349.obj	0.0045 83	670	0670.obj	0.0050 83
29	0029.obj	0.0055 83	350	0350.obj	0.0055 83	671	0671.obj	0.0060 83

30	0030.obj	0.0055 83	351	0351.obj	0.0055 83	672	0672.obj	0.0050 83
31	0031.obj	0.0050 83	352	0352.obj	0.0065 83	673	0673.obj	0.0050 83
32	0032.obj	0.0050 83	353	0353.obj	0.0055 83	674	0674.obj	0.0050 83
33	0033.obj	0.0055 83	354	0354.obj	0.0045 83	675	0675.obj	0.0045 83
34	0034.obj	0.0060 83	355	0355.obj	0.0050 83	676	0676.obj	0.0045 83
35	0035.obj	0.0060 83	356	0356.obj	0.0055 83	677	0677.obj	0.0045 83
36	0036.obj	0.0048 83	357	0357.obj	0.0050 83	678	0678.obj	0.0040 83
37	0037.obj	0.0053 83	358	0358.obj	0.0050 83	679	0679.obj	0.0040 83
38	0038.obj	0.0053 83	359	0359.obj	0.0045 83	680	0680.obj	0.0045 83
39	0039.obj	0.0060 83	360	0360.obj	0.0055 83	681	0681.obj	0.0040 83
40	0040.obj	0.0053 83	361	0361.obj	0.0055 83	682	0682.obj	0.0060 83
41	0041.obj	0.0060 83	362	0362.obj	0.0055 83	683	0683.obj	0.0045 83
42	0042.obj	0.0055 83	363	0363.obj	0.0070 83	684	0684.obj	0.0045 83
43	0043.obj	0.0060 83	364	0364.obj	0.0060 83	685	0685.obj	0.0040 83
44	0044.obj	0.0055 83	365	0365.obj	0.0045 83	686	0686.obj	0.0040 83
45	0045.obj	0.0060 83	366	0366.obj	0.0055 83	687	0687.obj	0.0055 83
46	0046.obj	0.0050 83	367	0367.obj	0.0050 83	688	0688.obj	0.0040 83
47	0047.obj	0.0052 83	368	0368.obj	0.0050 83	689	0689.obj	0.0065 83
48	0048.obj	0.0060 83	369	0369.obj	0.0050 83	690	0690.obj	0.0045 83
49	0049.obj	0.0060 83	370	0370.obj	0.0045 83	691	0691.obj	0.0055 83
50	0050.obj	0.0065 83	371	0371.obj	0.0040 83	692	0692.obj	0.0050 83
51	0051.obj	0.0055 83	372	0372.obj	0.0045 83	693	0693.obj	0.0050 83
52	0052.obj	0.0052 83	373	0373.obj	0.0045 83	694	0694.obj	0.0040 83
53	0053.obj	0.0060 83	374	0374.obj	0.0070 83	695	0695.obj	0.0045 83
54	0054.obj	0.0045 83	375	0375.obj	0.0045 83	696	0696.obj	0.0040 83
55	0055.obj	0.0045 83	376	0376.obj	0.0070 83	697	0697.obj	0.0070 83
56	0056.obj	0.0055 83	377	0377.obj	0.0070 83	698	0698.obj	0.0065 83
57	0057.obj	0.0060 83	378	0378.obj	0.0060 83	699	0699.obj	0.0055 83
58	0058.obj	0.0060 83	379	0379.obj	0.0060 83	700	0700.obj	0.0045 83
59	0059.obj	0.0060 83	380	0380.obj	0.0055 83	701	0701.obj	0.0050 83
60	0060.obj	0.0060 83	381	0381.obj	0.0055 83	702	0702.obj	0.0050 83
61	0061.obj	0.0050 83	382	0382.obj	0.0065 83	703	0703.obj	0.0040 83

62	0062.obj	0.0043 83	383	0383.obj	0.0050 83	704	0704.obj	0.0045 83
63	0063.obj	0.0065 83	384	0384.obj	0.0050 83	705	0705.obj	0.0060 83
64	0064.obj	0.0065 83	385	0385.obj	0.0050 83	706	0706.obj	0.0055 83
65	0065.obj	0.0053 83	386	0386.obj	0.0050 83	707	0707.obj	0.0055 83
66	0066.obj	0.0045 83	387	0387.obj	0.0045 83	708	0708.obj	0.0065 83
67	0067.obj	0.0048 83	388	0388.obj	0.0055 83	709	0709.obj	0.0055 83
68	0068.obj	0.0050 83	389	0389.obj	0.0065 83	710	0710.obj	0.0050 83
69	0069.obj	0.0045 83	390	0390.obj	0.0065 83	711	0711.obj	0.0070 83
70	0070.obj	0.0055 83	391	0391.obj	0.0070 83	712	0712.obj	0.0060 83
71	0071.obj	0.0050 83	392	0392.obj	0.0065 83	713	0713.obj	0.0060 83
72	0072.obj	0.0060 83	393	0393.obj	0.0050 83	714	0714.obj	0.0055 83
73	0073.obj	0.0060 83	394	0394.obj	0.0045 83	715	0715.obj	0.0050 83
74	0074.obj	0.0055 83	395	0395.obj	0.0055 83	716	0716.obj	0.0055 83
75	0075.obj	0.0055 83	396	0396.obj	0.0055 83	717	0717.obj	0.0055 83
76	0076.obj	0.0050 83	397	0397.obj	0.0050 83	718	0718.obj	0.0045 83
77	0077.obj	0.0055 83	398	0398.obj	0.0045 83	719	0719.obj	0.0060 83
78	0078.obj	0.0045 83	399	0399.obj	0.0055 83	720	0720.obj	0.0050 83
79	0079.obj	0.0045 83	400	0400.obj	0.0065 83	721	0721.obj	0.0050 83
80	0080.obj	0.0060 83	401	0401.obj	0.0050 83	722	0722.obj	0.0040 83
81	0081.obj	0.0055 83	402	0402.obj	0.0045 83	723	0723.obj	0.0045 83
82	0082.obj	0.0060 83	403	0403.obj	0.0045 83	724	0724.obj	0.0055 83
83	0083.obj	0.0045 83	404	0404.obj	0.0040 83	725	0725.obj	0.0060 83
84	0084.obj	0.0050 83	405	0405.obj	0.0045 83	726	0726.obj	0.0060 83
85	0085.obj	0.0045 83	406	0406.obj	0.0045 83	727	0727.obj	0.0045 83
86	0086.obj	0.0045 83	407	0407.obj	0.0045 83	728	0728.obj	0.0040 83
87	0087.obj	0.0045 83	408	0408.obj	0.0045 83	729	0729.obj	0.0060 83
88	0088.obj	0.0045 83	409	0409.obj	0.0050 83	730	0730.obj	0.0060 83
89	0089.obj	0.0050 83	410	0410.obj	0.0050 83	731	0731.obj	0.0055 83
90	0090.obj	0.0050 83	411	0411.obj	0.0050 83	732	0732.obj	0.0045 83
91	0091.obj	0.0050 83	412	0412.obj	0.0065 83	733	0733.obj	0.0050 83
92	0092.obj	0.0072 83	413	0413.obj	0.0065 83	734	0734.obj	0.0060 83
93	0093.obj	0.0052 83	414	0414.obj	0.0035 83	735	0735.obj	0.0065 83

94	0094.obj	0.0045 83	415	0415.obj	0.0040 83	736	0736.obj	0.0045 83
95	0095.obj	0.0070 83	416	0416.obj	0.0040 83	737	0737.obj	0.0045 83
96	0096.obj	0.0052 83	417	0417.obj	0.0055 83	738	0738.obj	0.0060 83
97	0097.obj	0.0055 83	418	0418.obj	0.0045 83	739	0739.obj	0.0040 83
98	0098.obj	0.0050 83	419	0419.obj	0.0040 83	740	0740.obj	0.0040 83
99	0099.obj	0.0052 83	420	0420.obj	0.0055 83	741	0741.obj	0.0045 83
100	0100.obj	0.0065 83	421	0421.obj	0.0040 83	742	0742.obj	0.0045 83
101	0101.obj	0.0060 83	422	0422.obj	0.0045 83	743	0743.obj	0.0040 83
102	0102.obj	0.0060 83	423	0423.obj	0.0035 83	744	0744.obj	0.0035 83
103	0103.obj	0.0060 83	424	0424.obj	0.0040 83	745	0745.obj	0.0040 83
104	0104.obj	0.0060 83	425	0425.obj	0.0040 83	746	0746.obj	0.0040 83
105	0105.obj	0.0060 83	426	0426.obj	0.0040 83	747	0747.obj	0.0035 83
106	0106.obj	0.0055 83	427	0427.obj	0.0035 83	748	0748.obj	0.0055 83
107	0107.obj	0.0050 83	428	0428.obj	0.0065 83	749	0749.obj	0.0070 83
108	0108.obj	0.0060 83	429	0429.obj	0.0060 83	750	0750.obj	0.0040 83
109	0109.obj	0.0055 83	430	0430.obj	0.0040 83	751	0751.obj	0.0045 83
110	0110.obj	0.0055 83	431	0431.obj	0.0050 83	752	0752.obj	0.0045 83
111	0111.obj	0.0055 83	432	0432.obj	0.0055 83	753	0753.obj	0.0050 83
112	0112.obj	0.0055 83	433	0433.obj	0.0065 83	754	0754.obj	0.0050 83
113	0113.obj	0.0072 83	434	0434.obj	0.0055 83	755	0755.obj	0.0040 83
114	0114.obj	0.0050 83	435	0435.obj	0.0065 83	756	0756.obj	0.0050 83
115	0115.obj	0.0050 83	436	0436.obj	0.0060 83	757	0757.obj	0.0040 83
116	0116.obj	0.0050 83	437	0437.obj	0.0060 83	758	0758.obj	0.0045 83
117	0117.obj	0.0050 83	438	0438.obj	0.0060 83	759	0759.obj	0.0050 83
118	0118.obj	0.0050 83	439	0439.obj	0.0060 83	760	0760.obj	0.0035 83
119	0119.obj	0.0075 83	440	0440.obj	0.0060 83	761	0761.obj	0.0040 83
120	0120.obj	0.0070 83	441	0441.obj	0.0050 83	762	0762.obj	0.0065 83
121	0121.obj	0.0060 83	442	0442.obj	0.0050 83	763	0763.obj	0.0050 83
122	0122.obj	0.0060 83	443	0443.obj	0.0050 83	764	0764.obj	0.0060 83
123	0123.obj	0.0060 83	444	0444.obj	0.0050 83	765	0765.obj	0.0155 83
124	0124.obj	0.0060 83	445	0445.obj	0.0050 83	766	0766.obj	0.0135 83
125	0125.obj	0.0060 83	446	0446.obj	0.0045 83	767	0767.obj	0.0115 83

126	0126.obj	0.0070 83	447	0447.obj	0.0045 83	768	0768.obj	0.0125 83
127	0127.obj	0.0080 83	448	0448.obj	0.0055 83	769	0769.obj	0.0150 83
128	0128.obj	0.0060 83	449	0449.obj	0.0055 83	770	0770.obj	0.0110 83
129	0129.obj	0.0060 83	450	0450.obj	0.0040 83	771	0771.obj	0.0110 83
130	0130.obj	0.0060 83	451	0451.obj	0.0045 83	772	0772.obj	0.0125 83
131	0131.obj	0.0075 83	452	0452.obj	0.0105 83	773	0773.obj	0.0140 83
132	0132.obj	0.0055 83	453	0453.obj	0.0060 83	774	0774.obj	0.0110 83
133	0133.obj	0.0060 83	454	0454.obj	0.0100 83	775	0775.obj	0.0140 83
134	0134.obj	0.0060 83	455	0455.obj	0.0105 83	776	0776.obj	0.0120 83
135	0135.obj	0.0060 83	456	0456.obj	0.0095 83	777	0777.obj	0.0180 83
136	0136.obj	0.0055 83	457	0457.obj	0.0170 83	778	0778.obj	0.0150 83
137	0137.obj	0.0055 83	458	0458.obj	0.0145 83	779	0779.obj	0.0180 83
138	0138.obj	0.0055 83	459	0459.obj	0.0150 83	780	0780.obj	0.0140 83
139	0139.obj	0.0045 83	460	0460.obj	0.0115 83	781	0781.obj	0.0180 83
140	0140.obj	0.0050 83	461	0461.obj	0.0145 83	782	0782.obj	0.0170 83
141	0141.obj	0.0050 83	462	0462.obj	0.0095 83	783	0783.obj	0.0240 83
142	0142.obj	0.0055 83	463	0463.obj	0.0095 83	784	0784.obj	0.0360 83
143	0143.obj	0.0050 83	464	0464.obj	0.0075 83	785	0785.obj	0.0360 83
144	0144.obj	0.0050 83	465	0465.obj	0.0075 83	786	0786.obj	0.0360 83
145	0145.obj	0.0045 83	466	0466.obj	0.0065 83	787	0787.obj	0.0360 83
146	0146.obj	0.0045 83	467	0467.obj	0.0075 83	788	0788.obj	0.0330 83
147	0147.obj	0.0050 83	468	0468.obj	0.0075 83	789	0789.obj	0.0360 83
148	0148.obj	0.0045 83	469	0469.obj	0.0055 83	790	0790.obj	0.0360 83
149	0149.obj	0.0045 83	470	0470.obj	0.0075 83	791	0791.obj	0.0330 83
150	0150.obj	0.0045 83	471	0471.obj	0.0075 83	792	0792.obj	0.0330 83
151	0151.obj	0.0050 83	472	0472.obj	0.0110 83	793	0793.obj	0.0105 83
152	0152.obj	0.0070 83	473	0473.obj	0.0105 83	794	0794.obj	0.0105 83
153	0153.obj	0.0075 83	474	0474.obj	0.0075 83	795	0795.obj	0.0105 83
154	0154.obj	0.0070 83	475	0475.obj	0.0075 83	796	0796.obj	0.0115 83
155	0155.obj	0.0060 83	476	0476.obj	0.0060 83	797	0797.obj	0.0160 83
156	0156.obj	0.0050 83	477	0477.obj	0.0065 83	798	0798.obj	0.0330 83
157	0157.obj	0.0050 83	478	0478.obj	0.0065 83	799	0799.obj	0.0330 83

158	0158.obj	0.0050 83	479	0479.obj	0.0075 83	800	0800.obj	0.0180 83
159	0159.obj	0.0050 83	480	0480.obj	0.0050 83	801	0801.obj	0.0180 83
160	0160.obj	0.0055 83	481	0481.obj	0.0075 83	802	0802.obj	0.0140 83
161	0161.obj	0.0050 83	482	0482.obj	0.0075 83	803	0803.obj	0.0070 83
162	0162.obj	0.0050 83	483	0483.obj	0.0100 83	804	0804.obj	0.0095 83
163	0163.obj	0.0060 83	484	0484.obj	0.0070 83	805	0805.obj	0.0095 83
164	0164.obj	0.0055 83	485	0485.obj	0.0075 83	806	0806.obj	0.0075 83
165	0165.obj	0.0055 83	486	0486.obj	0.0105 83	807	0807.obj	0.0135 83
166	0166.obj	0.0045 83	487	0487.obj	0.0075 83	808	0808.obj	0.0070 83
167	0167.obj	0.0050 83	488	0488.obj	0.0095 83	809	0809.obj	0.0080 83
168	0168.obj	0.0055 83	489	0489.obj	0.0095 83	810	0810.obj	0.0075 83
169	0169.obj	0.0060 83	490	0490.obj	0.0095 83	811	0811.obj	0.0080 83
170	0170.obj	0.0055 83	491	0491.obj	0.0095 83	812	0812.obj	0.0110 83
171	0171.obj	0.0050 83	492	0492.obj	0.0075 83	813	0813.obj	0.0110 83
172	0172.obj	0.0050 83	493	0493.obj	0.0040 83	814	0814.obj	0.0105 83
173	0173.obj	0.0050 83	494	0494.obj	0.0050 83	815	0815.obj	0.0145 83
174	0174.obj	0.0055 83	495	0495.obj	0.0115 83	816	0816.obj	0.0145 83
175	0175.obj	0.0060 83	496	0496.obj	0.0055 83	817	0817.obj	0.0145 83
176	0176.obj	0.0060 83	497	0497.obj	0.0055 83	818	0818.obj	0.0130 83
177	0177.obj	0.0060 83	498	0498.obj	0.0035 83	819	0819.obj	0.0130 83
178	0178.obj	0.0060 83	499	0499.obj	0.0060 83	820	0820.obj	0.0170 83
179	0179.obj	0.0060 83	500	0500.obj	0.0060 83	821	0821.obj	0.0090 83
180	0180.obj	0.0060 83	501	0501.obj	0.0050 83	822	0822.obj	0.0150 83
181	0181.obj	0.0050 83	502	0502.obj	0.0050 83	823	0823.obj	0.0130 83
182	0182.obj	0.0060 83	503	0503.obj	0.0060 83	824	0824.obj	0.0085 83
183	0183.obj	0.0060 83	504	0504.obj	0.0075 83	825	0825.obj	0.0105 83
184	0184.obj	0.0055 83	505	0505.obj	0.0075 83	826	0826.obj	0.0065 83
185	0185.obj	0.0055 83	506	0506.obj	0.0065 83	827	0827.obj	0.0095 83
186	0186.obj	0.0050 83	507	0507.obj	0.0115 83	828	0828.obj	0.0055 83
187	0187.obj	0.0055 83	508	0508.obj	0.0055 83	829	0829.obj	0.0155 83
188	0188.obj	0.0045 83	509	0509.obj	0.0060 83	830	0830.obj	0.0070 83
189	0189.obj	0.0075 83	510	0510.obj	0.0070 83	831	0831.obj	0.0075 83

190	0190.obj	0.0040 83	511	0511.obj	0.0115 83	832	0832.obj	0.0075 83
191	0191.obj	0.0055 83	512	0512.obj	0.0125 83	833	0833.obj	0.0105 83
192	0192.obj	0.0050 83	513	0513.obj	0.0045 83	834	0834.obj	0.0070 83
193	0193.obj	0.0060 83	514	0514.obj	0.0045 83	835	0835.obj	0.0065 83
194	0194.obj	0.0060 83	515	0515.obj	0.0040 83	836	0836.obj	0.0095 83
195	0195.obj	0.0060 83	516	0516.obj	0.0045 83	837	0837.obj	0.0105 83
196	0196.obj	0.0055 83	517	0517.obj	0.0055 83	838	0838.obj	0.0105 83
197	0197.obj	0.0045 83	518	0518.obj	0.0045 83	839	0839.obj	0.0130 83
198	0198.obj	0.0045 83	519	0519.obj	0.0055 83	840	0840.obj	0.0095 83
199	0199.obj	0.0050 83	520	0520.obj	0.0055 83	841	0841.obj	0.0095 83
200	0200.obj	0.0055 83	521	0521.obj	0.0055 83	842	0842.obj	0.0130 83
201	0201.obj	0.0045 83	522	0522.obj	0.0050 83	843	0843.obj	0.0085 83
202	0202.obj	0.0050 83	523	0523.obj	0.0050 83	844	0844.obj	0.0150 83
203	0203.obj	0.0065 83	524	0524.obj	0.0050 83	845	0845.obj	0.0060 83
204	0204.obj	0.0045 83	525	0525.obj	0.0050 83	846	0846.obj	0.0085 83
205	0205.obj	0.0055 83	526	0526.obj	0.0040 83	847	0847.obj	0.0095 83
206	0206.obj	0.0055 83	527	0527.obj	0.0045 83	848	0848.obj	0.0085 83
207	0207.obj	0.0050 83	528	0528.obj	0.0045 83	849	0849.obj	0.0060 83
208	0208.obj	0.0055 83	529	0529.obj	0.0050 83	850	0850.obj	0.0105 83
209	0209.obj	0.0055 83	530	0530.obj	0.0045 83	851	0851.obj	0.0115 83
210	0210.obj	0.0045 83	531	0531.obj	0.0040 83	852	0852.obj	0.0135 83
211	0211.obj	0.0055 83	532	0532.obj	0.0040 83	853	0853.obj	0.0135 83
212	0212.obj	0.0055 83	533	0533.obj	0.0040 83	854	0854.obj	0.0075 83
213	0213.obj	0.0045 83	534	0534.obj	0.0045 83	855	0855.obj	0.0050 83
214	0214.obj	0.0060 83	535	0535.obj	0.0040 83	856	0856.obj	0.0055 83
215	0215.obj	0.0050 83	536	0536.obj	0.0040 83	857	0857.obj	0.0050 83
216	0216.obj	0.0055 83	537	0537.obj	0.0045 83	858	0858.obj	0.0055 83
217	0217.obj	0.0055 83	538	0538.obj	0.0055 83	859	0859.obj	0.0050 83
218	0218.obj	0.0060 83	539	0539.obj	0.0050 83	860	0860.obj	0.0065 83
219	0219.obj	0.0050 83	540	0540.obj	0.0040 83	861	0861.obj	0.0050 83
220	0220.obj	0.0060 83	541	0541.obj	0.0050 83	862	0862.obj	0.0055 83
221	0221.obj	0.0060 83	542	0542.obj	0.0050 83	863	0863.obj	0.0045 83

222	0222.obj	0.0060 83	543	0543.obj	0.0050 83	864	0864.obj	0.0060 83
223	0223.obj	0.0055 83	544	0544.obj	0.0045 83	865	0865.obj	0.0060 83
224	0224.obj	0.0055 83	545	0545.obj	0.0045 83	866	0866.obj	0.0045 83
225	0225.obj	0.0050 83	546	0546.obj	0.0045 83	867	0867.obj	0.0055 83
226	0226.obj	0.0065 83	547	0547.obj	0.0040 83	868	0868.obj	0.0055 83
227	0227.obj	0.0065 83	548	0548.obj	0.0040 83	869	0869.obj	0.0055 83
228	0228.obj	0.0050 83	549	0549.obj	0.0045 83	870	0870.obj	0.0050 83
229	0229.obj	0.0055 83	550	0550.obj	0.0050 83	871	0871.obj	0.0040 83
230	0230.obj	0.0055 83	551	0551.obj	0.0050 83	872	0872.obj	0.0040 83
231	0231.obj	0.0055 83	552	0552.obj	0.0045 83	873	0873.obj	0.0040 83
232	0232.obj	0.0065 83	553	0553.obj	0.0040 83	874	0874.obj	0.0055 83
233	0233.obj	0.0060 83	554	0554.obj	0.0040 83	875	0875.obj	0.0040 83
234	0234.obj	0.0055 83	555	0555.obj	0.0065 83	876	0876.obj	0.0045 83
235	0235.obj	0.0055 83	556	0556.obj	0.0060 83	877	0877.obj	0.0045 83
236	0236.obj	0.0065 83	557	0557.obj	0.0055 83	878	0878.obj	0.0040 83
237	0237.obj	0.0060 83	558	0558.obj	0.0045 83	879	0879.obj	0.0045 83
238	0238.obj	0.0065 83	559	0559.obj	0.0040 83	880	0880.obj	0.0070 83
239	0239.obj	0.0060 83	560	0560.obj	0.0060 83	881	0881.obj	0.0050 83
240	0240.obj	0.0060 83	561	0561.obj	0.0065 83	882	0882.obj	0.0055 83
241	0241.obj	0.0060 83	562	0562.obj	0.0105 83	883	0883.obj	0.0050 83
242	0242.obj	0.0055 83	563	0563.obj	0.0125 83	884	0884.obj	0.0050 83
243	0243.obj	0.0055 83	564	0564.obj	0.0160 83	885	0885.obj	0.0060 83
244	0244.obj	0.0055 83	565	0565.obj	0.0160 83	886	0886.obj	0.0060 83
245	0245.obj	0.0060 83	566	0566.obj	0.0170 83	887	0887.obj	0.0065 83
246	0246.obj	0.0075 83	567	0567.obj	0.0190 83	888	0888.obj	0.0045 83
247	0247.obj	0.0050 83	568	0568.obj	0.0190 83	889	0889.obj	0.0055 83
248	0248.obj	0.0045 83	569	0569.obj	0.0165 83	890	0890.obj	0.0065 83
249	0249.obj	0.0060 83	570	0570.obj	0.0100 83	891	0891.obj	0.0065 83
250	0250.obj	0.0065 83	571	0571.obj	0.0145 83	892	0892.obj	0.0055 83
251	0251.obj	0.0050 83	572	0572.obj	0.0075 83	893	0893.obj	0.0055 83
252	0252.obj	0.0055 83	573	0573.obj	0.0170 83	894	0894.obj	0.0060 83
253	0253.obj	0.0065 83	574	0574.obj	0.0185 83	895	0895.obj	0.0075 83

254	0254.obj	0.0060 83	575	0575.obj	0.0075 83	896	0896.obj	0.0060 83
255	0255.obj	0.0060 83	576	0576.obj	0.0105 83	897	0897.obj	0.0050 83
256	0256.obj	0.0060 83	577	0577.obj	0.0075 83	898	0898.obj	0.0060 83
257	0257.obj	0.0055 83	578	0578.obj	0.0105 83	899	0899.obj	0.0040 83
258	0258.obj	0.0065 83	579	0579.obj	0.0115 83	900	0900.obj	0.0055 83
259	0259.obj	0.0060 83	580	0580.obj	0.0100 83	901	0901.obj	0.0055 83
260	0260.obj	0.0065 83	581	0581.obj	0.0100 83	902	0902.obj	0.0060 83
261	0261.obj	0.0055 83	582	0582.obj	0.0060 83	903	0903.obj	0.0075 83
262	0262.obj	0.0060 83	583	0583.obj	0.0055 83	904	0904.obj	0.0060 83
263	0263.obj	0.0050 83	584	0584.obj	0.0055 83	905	0905.obj	0.0050 83
264	0264.obj	0.0065 83	585	0585.obj	0.0055 83	906	0906.obj	0.0060 83
265	0265.obj	0.0050 83	586	0586.obj	0.0055 83	907	0907.obj	0.0060 83
266	0266.obj	0.0065 83	587	0587.obj	0.0050 83	908	0908.obj	0.0045 83
267	0267.obj	0.0050 83	588	0588.obj	0.0045 83	909	0909.obj	0.0040 83
268	0268.obj	0.0040 83	589	0589.obj	0.0055 83	910	0910.obj	0.0060 83
269	0269.obj	0.0060 83	590	0590.obj	0.0050 83	911	0911.obj	0.0060 83
270	0270.obj	0.0065 83	591	0591.obj	0.0050 83	912	0912.obj	0.0050 83
271	0271.obj	0.0060 83	592	0592.obj	0.0055 83	913	0913.obj	0.0065 83
272	0272.obj	0.0065 83	593	0593.obj	0.0050 83	914	0914.obj	0.0050 83
273	0273.obj	0.0055 83	594	0594.obj	0.0055 83	915	0915.obj	0.0070 83
274	0274.obj	0.0060 83	595	0595.obj	0.0110 83	916	0916.obj	0.0060 83
275	0275.obj	0.0065 83	596	0596.obj	0.0075 83	917	0917.obj	0.0045 83
276	0276.obj	0.0055 83	597	0597.obj	0.0075 83	918	0918.obj	0.0045 83
277	0277.obj	0.0055 83	598	0598.obj	0.0065 83	919	0919.obj	0.0045 83
278	0278.obj	0.0060 83	599	0599.obj	0.0050 83	920	0920.obj	0.0065 83
279	0279.obj	0.0050 83	600	0600.obj	0.0050 83	921	0921.obj	0.0050 83
280	0280.obj	0.0050 83	601	0601.obj	0.0065 83	922	0922.obj	0.0050 83
281	0281.obj	0.0065 83	602	0602.obj	0.0050 83	923	0923.obj	0.0050 83
282	0282.obj	0.0050 83	603	0603.obj	0.0070 83	924	0924.obj	0.0050 83
283	0283.obj	0.0075 83	604	0604.obj	0.0060 83	925	0925.obj	0.0040 83
284	0284.obj	0.0075 83	605	0605.obj	0.0060 83	926	0926.obj	0.0055 83
285	0285.obj	0.0055 83	606	0606.obj	0.0055 83	927	0927.obj	0.0045 83

286	0286.obj	0.0070 83	607	0607.obj	0.0060 83	928	0928.obj	0.0045 83
287	0287.obj	0.0075 83	608	0608.obj	0.0045 83	929	0929.obj	0.0045 83
288	0288.obj	0.0055 83	609	0609.obj	0.0030 83	930	0930.obj	0.0050 83
289	0289.obj	0.0055 83	610	0610.obj	0.0040 83	931	0931.obj	0.0050 83
290	0290.obj	0.0045 83	611	0611.obj	0.0045 83	932	0932.obj	0.0055 83
291	0291.obj	0.0060 83	612	0612.obj	0.0050 83	933	0933.obj	0.0070 83
292	0292.obj	0.0050 83	613	0613.obj	0.0060 83	934	0934.obj	0.0055 83
293	0293.obj	0.0070 83	614	0614.obj	0.0050 83	935	0935.obj	0.0055 83
294	0294.obj	0.0050 83	615	0615.obj	0.0050 83	936	0936.obj	0.0060 83
295	0295.obj	0.0045 83	616	0616.obj	0.0055 83	937	0937.obj	0.0045 83
296	0296.obj	0.0065 83	617	0617.obj	0.0050 83	938	0938.obj	0.0060 83
297	0297.obj	0.0050 83	618	0618.obj	0.0050 83	939	0939.obj	0.0055 83
298	0298.obj	0.0050 83	619	0619.obj	0.0030 83	940	0940.obj	0.0055 83
299	0299.obj	0.0055 83	620	0620.obj	0.0040 83	941	0941.obj	0.0045 83
300	0300.obj	0.0045 83	621	0621.obj	0.0035 83	942	0942.obj	0.0055 83
301	0301.obj	0.0070 83	622	0622.obj	0.0045 83	943	0943.obj	0.0045 83
302	0302.obj	0.0050 83	623	0623.obj	0.0060 83	944	0944.obj	0.0045 83
303	0303.obj	0.0050 83	624	0624.obj	0.0040 83	945	0945.obj	0.0040 83
304	0304.obj	0.0060 83	625	0625.obj	0.0070 83	946	0946.obj	0.0050 83
305	0305.obj	0.0060 83	626	0626.obj	0.0050 83	947	0947.obj	0.0040 83
306	0306.obj	0.0050 83	627	0627.obj	0.0055 83	948	0948.obj	0.0040 83
307	0307.obj	0.0050 83	628	0628.obj	0.0045 83	949	0949.obj	0.0040 83
308	0308.obj	0.0050 83	629	0629.obj	0.0050 83	950	0950.obj	0.0045 83
309	0309.obj	0.0060 83	630	0630.obj	0.0050 83	951	0951.obj	0.0040 83
310	0310.obj	0.0050 83	631	0631.obj	0.0070 83	952	0952.obj	0.0035 83
311	0311.obj	0.0060 83	632	0632.obj	0.0060 83	953	0953.obj	0.0045 83
312	0312.obj	0.0045 83	633	0633.obj	0.0065 83	954	0954.obj	0.0055 83
313	0313.obj	0.0050 83	634	0634.obj	0.0060 83	955	0955.obj	0.0055 83
314	0314.obj	0.0055 83	635	0635.obj	0.0065 83	956	0956.obj	0.0045 83
315	0315.obj	0.0045 83	636	0636.obj	0.0050 83	957	0957.obj	0.0045 83
316	0316.obj	0.0050 83	637	0637.obj	0.0050 83	958	0958.obj	0.0040 83
317	0317.obj	0.0050 83	638	0638.obj	0.0050 83	959	0959.obj	0.0035 83

318	0318.obj	0.0050 83	639	0639.obj	0.0065 83	960	0960.obj	0.0040 83
319	0319.obj	0.0060 83	640	0640.obj	0.0050 83	961	0961.obj	0.0040 83
320	0320.obj	0.0065 83	641	0641.obj	0.0040 83	962	0962.obj	0.0040 83
321	0321.obj	0.0045 83	642	0642.obj	0.0075 83	-	-	-

Tabla G2. Objetos Asignados de la categoría *Cone-Vase*

Índice	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**
1	0001.obj	1	54	0167.obj	1379	107	0433.obj	2757
2	0002.obj	27	55	0168.obj	1405	108	0438.obj	2783
3	0007.obj	53	56	0169.obj	1431	109	0481.obj	2809
4	0008.obj	79	57	0170.obj	1457	110	0497.obj	2835
5	0009.obj	105	58	0171.obj	1483	111	0499.obj	2861
6	0010.obj	131	59	0173.obj	1509	112	0500.obj	2887
7	0011.obj	157	60	0174.obj	1535	113	0505.obj	2913
8	0012.obj	183	61	0175.obj	1561	114	0507.obj	2939
9	0013.obj	209	62	0179.obj	1587	115	0512.obj	2965
10	0017.obj	235	63	0180.obj	1613	116	0530.obj	2991
11	0018.obj	261	64	0181.obj	1639	117	0538.obj	3017
12	0020.obj	287	65	0182.obj	1665	118	0539.obj	3043
13	0032.obj	313	66	0183.obj	1691	119	0544.obj	3069
14	0034.obj	339	67	0184.obj	1717	120	0550.obj	3095
15	0036.obj	365	68	0185.obj	1743	121	0571.obj	3121
16	0037.obj	391	69	0192.obj	1769	122	0578.obj	3147
17	0038.obj	417	70	0196.obj	1795	123	0579.obj	3173
18	0039.obj	443	71	0197.obj	1821	124	0596.obj	3199
19	0040.obj	469	72	0199.obj	1847	125	0611.obj	3225
20	0041.obj	495	73	0203.obj	1873	126	0643.obj	3251
21	0042.obj	521	74	0205.obj	1899	127	0645.obj	3277
22	0043.obj	547	75	0226.obj	1925	128	0646.obj	3303
23	0048.obj	573	76	0231.obj	1951	129	0647.obj	3329
24	0049.obj	599	77	0241.obj	1977	130	0650.obj	3355

25	0061.obj	625	78	0244.obj	2003	131	0657.obj	3381
26	0075.obj	651	79	0253.obj	2029	132	0677.obj	3407
27	0113.obj	677	80	0258.obj	2055	133	0683.obj	3433
28	0119.obj	703	81	0263.obj	2081	134	0689.obj	3459
29	0121.obj	729	82	0265.obj	2107	135	0690.obj	3485
30	0125.obj	755	83	0266.obj	2133	136	0691.obj	3511
31	0127.obj	781	84	0268.obj	2159	137	0692.obj	3537
32	0129.obj	807	85	0277.obj	2185	138	0693.obj	3563
33	0131.obj	833	86	0312.obj	2211	139	0700.obj	3589
34	0134.obj	859	87	0352.obj	2237	140	0703.obj	3615
35	0135.obj	885	88	0353.obj	2263	141	0721.obj	3641
36	0136.obj	911	89	0355.obj	2289	142	0734.obj	3667
37	0137.obj	937	90	0356.obj	2315	143	0736.obj	3693
38	0138.obj	963	91	0360.obj	2341	144	0751.obj	3719
39	0139.obj	989	92	0361.obj	2367	145	0759.obj	3745
40	0143.obj	1015	93	0362.obj	2393	146	0781.obj	3771
41	0144.obj	1041	94	0367.obj	2419	147	0814.obj	3797
42	0146.obj	1067	95	0368.obj	2445	148	0817.obj	3823
43	0147.obj	1093	96	0370.obj	2471	149	0820.obj	3849
44	0149.obj	1119	97	0371.obj	2497	150	0893.obj	3875
45	0152.obj	1145	98	0376.obj	2523	151	0912.obj	3901
46	0154.obj	1171	99	0379.obj	2549	152	0915.obj	3927
47	0156.obj	1197	100	0381.obj	2575	153	0919.obj	3953
48	0157.obj	1223	101	0382.obj	2601	154	0930.obj	3979
49	0158.obj	1249	102	0385.obj	2627	155	0932.obj	4005
50	0159.obj	1275	103	0387.obj	2653	156	0946.obj	4031
51	0161.obj	1301	104	0392.obj	2679	157	0957.obj	4057
52	0164.obj	1327	105	0395.obj	2705	-	-	-
53	0165.obj	1353	106	0428.obj	2731	-	-	-

Tabla G3. Objetos Asignados de la categoría *Bowl*

Índice	Objeto	Pseudo-Objeto	Índice	Objeto	Pseudo-Objeto	Índice	Objeto	Pseudo-Objeto
1	0004.obj	1	45	0191.obj	1365	89	0532.obj	2729
2	0015.obj	32	46	0204.obj	1396	90	0533.obj	2760
3	0022.obj	63	47	0206.obj	1427	91	0535.obj	2791
4	0026.obj	94	48	0207.obj	1458	92	0541.obj	2822
5	0057.obj	125	49	0208.obj	1489	93	0542.obj	2853
6	0063.obj	156	50	0210.obj	1520	94	0543.obj	2884
7	0064.obj	187	51	0211.obj	1551	95	0545.obj	2915
8	0065.obj	218	52	0212.obj	1582	96	0546.obj	2946
9	0066.obj	249	53	0213.obj	1613	97	0547.obj	2977
10	0068.obj	280	54	0215.obj	1644	98	0548.obj	3008
11	0070.obj	311	55	0220.obj	1675	99	0549.obj	3039
12	0071.obj	342	56	0222.obj	1706	100	0551.obj	3070
13	0076.obj	373	57	0225.obj	1737	101	0552.obj	3101
14	0079.obj	404	58	0228.obj	1768	102	0553.obj	3132
15	0080.obj	435	59	0230.obj	1799	103	0554.obj	3163
16	0084.obj	466	60	0234.obj	1830	104	0556.obj	3194
17	0086.obj	497	61	0242.obj	1861	105	0558.obj	3225
18	0088.obj	528	62	0249.obj	1892	106	0572.obj	3256
19	0089.obj	559	63	0251.obj	1923	107	0620.obj	3287
20	0090.obj	590	64	0271.obj	1954	108	0621.obj	3318
21	0091.obj	621	65	0272.obj	1985	109	0627.obj	3349
22	0096.obj	652	66	0273.obj	2016	110	0644.obj	3380
23	0097.obj	683	67	0274.obj	2047	111	0652.obj	3411
24	0098.obj	714	68	0279.obj	2078	112	0654.obj	3442
25	0101.obj	745	69	0285.obj	2109	113	0671.obj	3473
26	0106.obj	776	70	0288.obj	2140	114	0674.obj	3504
27	0107.obj	807	71	0314.obj	2171	115	0684.obj	3535
28	0109.obj	838	72	0340.obj	2202	116	0737.obj	3566
29	0111.obj	869	73	0359.obj	2233	117	0739.obj	3597

30	0112.obj	900	74	0364.obj	2264	118	0744.obj	3628
31	0114.obj	931	75	0365.obj	2295	119	0752.obj	3659
32	0115.obj	962	76	0378.obj	2326	120	0753.obj	3690
33	0117.obj	993	77	0383.obj	2357	121	0761.obj	3721
34	0118.obj	1024	78	0386.obj	2388	122	0804.obj	3752
35	0123.obj	1055	79	0390.obj	2419	123	0805.obj	3783
36	0128.obj	1086	80	0391.obj	2450	124	0810.obj	3814
37	0130.obj	1117	81	0394.obj	2481	125	0811.obj	3845
38	0132.obj	1148	82	0396.obj	2512	126	0812.obj	3876
39	0141.obj	1179	83	0448.obj	2543	127	0816.obj	3907
40	0148.obj	1210	84	0452.obj	2574	128	0822.obj	3938
41	0153.obj	1241	85	0477.obj	2605	129	0886.obj	3969
42	0178.obj	1272	86	0478.obj	2636	130	0909.obj	4000
43	0188.obj	1303	87	0484.obj	2667	131	0956.obj	4031
44	0190.obj	1334	88	0496.obj	2698	132	0961.obj	4062

Tabla G4. Objetos Asignados de la categoría *Jar*

Índice	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**
1	0030.obj	1	23	0342.obj	1365	45	0642.obj	2729
2	0054.obj	63	24	0374.obj	1427	46	0701.obj	2791
3	0055.obj	125	25	0397.obj	1489	47	0720.obj	2853
4	0074.obj	187	26	0405.obj	1551	48	0738.obj	2915
5	0077.obj	249	27	0407.obj	1613	49	0769.obj	2977
6	0099.obj	311	28	0431.obj	1675	50	0772.obj	3039
7	0103.obj	373	29	0432.obj	1737	51	0788.obj	3101
8	0122.obj	435	30	0439.obj	1799	52	0807.obj	3163
9	0142.obj	497	31	0453.obj	1861	53	0818.obj	3225
10	0151.obj	559	32	0454.obj	1923	54	0828.obj	3287
11	0214.obj	621	33	0485.obj	1985	55	0829.obj	3349
12	0216.obj	683	34	0503.obj	2047	56	0830.obj	3411
13	0227.obj	745	35	0513.obj	2109	57	0831.obj	3473

14	0260.obj	807	36	0561.obj	2171	58	0832.obj	3535
15	0269.obj	869	37	0574.obj	2233	59	0853.obj	3597
16	0276.obj	931	38	0581.obj	2295	60	0902.obj	3659
17	0287.obj	993	39	0586.obj	2357	61	0903.obj	3721
18	0290.obj	1055	40	0618.obj	2419	62	0904.obj	3783
19	0313.obj	1117	41	0622.obj	2481	63	0905.obj	3845
20	0324.obj	1179	42	0623.obj	2543	64	0906.obj	3907
21	0339.obj	1241	43	0626.obj	2605	65	0929.obj	3969
22	0341.obj	1303	44	0640.obj	2667	-	-	-

Tabla G5. Objetos Asignados de la categoría Lebrillo

Índice	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**
1	0003.obj	1	29	0656.obj	1373	57	0740.obj	2745
2	0005.obj	50	30	0658.obj	1422	58	0741.obj	2794
3	0016.obj	99	31	0659.obj	1471	59	0742.obj	2843
4	0019.obj	148	32	0660.obj	1520	60	0743.obj	2892
5	0021.obj	197	33	0662.obj	1569	61	0745.obj	2941
6	0023.obj	246	34	0663.obj	1618	62	0746.obj	2990
7	0024.obj	295	35	0664.obj	1667	63	0747.obj	3039
8	0025.obj	344	36	0665.obj	1716	64	0749.obj	3088
9	0058.obj	393	37	0666.obj	1765	65	0754.obj	3137
10	0060.obj	442	38	0667.obj	1814	66	0755.obj	3186
11	0062.obj	491	39	0668.obj	1863	67	0756.obj	3235
12	0078.obj	540	40	0669.obj	1912	68	0757.obj	3284
13	0083.obj	589	41	0670.obj	1961	69	0758.obj	3333
14	0150.obj	638	42	0672.obj	2010	70	0760.obj	3382
15	0166.obj	687	43	0673.obj	2059	71	0806.obj	3431
16	0198.obj	736	44	0675.obj	2108	72	0855.obj	3480
17	0201.obj	785	45	0676.obj	2157	73	0877.obj	3529
18	0280.obj	834	46	0679.obj	2206	74	0888.obj	3578
19	0282.obj	883	47	0680.obj	2255	75	0899.obj	3627

20	0354.obj	932	48	0681.obj	2304	76	0924.obj	3676
21	0372.obj	981	49	0685.obj	2353	77	0953.obj	3725
22	0373.obj	1030	50	0686.obj	2402	78	0954.obj	3774
23	0375.obj	1079	51	0687.obj	2451	79	0955.obj	3823
24	0576.obj	1128	52	0730.obj	2500	80	0958.obj	3872
25	0582.obj	1177	53	0731.obj	2549	81	0959.obj	3921
26	0624.obj	1226	54	0732.obj	2598	82	0960.obj	3970
27	0653.obj	1275	55	0733.obj	2647	83	0962.obj	4019
28	0655.obj	1324	56	0735.obj	2696	-	-	-

Tabla G6. Objetos Asignados de la categoría Olla

Índice	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**
1	0006.obj	1	39	0305.obj	1369	77	0705.obj	2737
2	0028.obj	37	40	0320.obj	1405	78	0707.obj	2773
3	0035.obj	73	41	0321.obj	1441	79	0709.obj	2809
4	0044.obj	109	42	0322.obj	1477	80	0711.obj	2845
5	0051.obj	145	43	0357.obj	1513	81	0712.obj	2881
6	0052.obj	181	44	0358.obj	1549	82	0713.obj	2917
7	0053.obj	217	45	0366.obj	1585	83	0714.obj	2953
8	0081.obj	253	46	0369.obj	1621	84	0715.obj	2989
9	0087.obj	289	47	0377.obj	1657	85	0716.obj	3025
10	0110.obj	325	48	0380.obj	1693	86	0717.obj	3061
11	0116.obj	361	49	0393.obj	1729	87	0718.obj	3097
12	0124.obj	397	50	0420.obj	1765	88	0719.obj	3133
13	0140.obj	433	51	0434.obj	1801	89	0723.obj	3169
14	0145.obj	469	52	0437.obj	1837	90	0726.obj	3205
15	0155.obj	505	53	0442.obj	1873	91	0727.obj	3241
16	0160.obj	541	54	0446.obj	1909	92	0728.obj	3277
17	0172.obj	577	55	0447.obj	1945	93	0748.obj	3313
18	0177.obj	613	56	0457.obj	1981	94	0763.obj	3349
19	0193.obj	649	57	0464.obj	2017	95	0764.obj	3385

20	0194.obj	685	58	0467.obj	2053	96	0770.obj	3421
21	0195.obj	721	59	0468.obj	2089	97	0777.obj	3457
22	0209.obj	757	60	0482.obj	2125	98	0778.obj	3493
23	0221.obj	793	61	0483.obj	2161	99	0785.obj	3529
24	0237.obj	829	62	0487.obj	2197	100	0914.obj	3565
25	0238.obj	865	63	0518.obj	2233	101	0916.obj	3601
26	0239.obj	901	64	0519.obj	2269	102	0918.obj	3637
27	0240.obj	937	65	0534.obj	2305	103	0920.obj	3673
28	0245.obj	973	66	0577.obj	2341	104	0923.obj	3709
29	0246.obj	1009	67	0593.obj	2377	105	0933.obj	3745
30	0252.obj	1045	68	0595.obj	2413	106	0934.obj	3781
31	0261.obj	1081	69	0597.obj	2449	107	0935.obj	3817
32	0262.obj	1117	70	0608.obj	2485	108	0936.obj	3853
33	0275.obj	1153	71	0612.obj	2521	109	0937.obj	3889
34	0283.obj	1189	72	0651.obj	2557	110	0941.obj	3925
35	0291.obj	1225	73	0661.obj	2593	111	0943.obj	3961
36	0295.obj	1261	74	0682.obj	2629	112	0944.obj	3997
37	0299.obj	1297	75	0697.obj	2665	-	-	-
38	0304.obj	1333	76	0704.obj	2701	-	-	-

Tabla G7. Objetos Asignados de la categoría *Plate*

Índice	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**
1	0014.obj	1	29	0415.obj	1345	57	0870.obj	2689
2	0093.obj	49	30	0416.obj	1393	58	0871.obj	2737
3	0094.obj	97	31	0430.obj	1441	59	0872.obj	2785
4	0235.obj	145	32	0450.obj	1489	60	0873.obj	2833
5	0289.obj	193	33	0619.obj	1537	61	0874.obj	2881
6	0292.obj	241	34	0628.obj	1585	62	0876.obj	2929
7	0293.obj	289	35	0750.obj	1633	63	0878.obj	2977
8	0294.obj	337	36	0808.obj	1681	64	0879.obj	3025
9	0297.obj	385	37	0809.obj	1729	65	0880.obj	3073

10	0298.obj	433	38	0813.obj	1777	66	0881.obj	3121
11	0303.obj	481	39	0815.obj	1825	67	0882.obj	3169
12	0306.obj	529	40	0819.obj	1873	68	0883.obj	3217
13	0307.obj	577	41	0823.obj	1921	69	0884.obj	3265
14	0308.obj	625	42	0825.obj	1969	70	0885.obj	3313
15	0309.obj	673	43	0856.obj	2017	71	0887.obj	3361
16	0311.obj	721	44	0857.obj	2065	72	0889.obj	3409
17	0315.obj	769	45	0858.obj	2113	73	0890.obj	3457
18	0316.obj	817	46	0859.obj	2161	74	0891.obj	3505
19	0317.obj	865	47	0860.obj	2209	75	0894.obj	3553
20	0332.obj	913	48	0861.obj	2257	76	0895.obj	3601
21	0333.obj	961	49	0862.obj	2305	77	0896.obj	3649
22	0334.obj	1009	50	0863.obj	2353	78	0897.obj	3697
23	0335.obj	1057	51	0864.obj	2401	79	0898.obj	3745
24	0336.obj	1105	52	0865.obj	2449	80	0900.obj	3793
25	0337.obj	1153	53	0866.obj	2497	81	0901.obj	3841
26	0338.obj	1201	54	0867.obj	2545	82	0908.obj	3889
27	0398.obj	1249	55	0868.obj	2593	83	0910.obj	3937
28	0402.obj	1297	56	0869.obj	2641	84	0911.obj	3985

Tabla G8. Objetos Asignados de la categoría *Vessel*

Índice	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**	**Índice**	**Objeto**	**Pseudo-Objeto**
1	0029.obj	1	45	0435.obj	1365	89	0615.obj	2729
2	0031.obj	32	46	0440.obj	1396	90	0616.obj	2760
3	0045.obj	63	47	0458.obj	1427	91	0617.obj	2791
4	0046.obj	94	48	0465.obj	1458	92	0625.obj	2822
5	0050.obj	125	49	0470.obj	1489	93	0629.obj	2853
6	0072.obj	156	50	0474.obj	1520	94	0631.obj	2884
7	0082.obj	187	51	0486.obj	1551	95	0632.obj	2915
8	0092.obj	218	52	0488.obj	1582	96	0636.obj	2946
9	0120.obj	249	53	0489.obj	1613	97	0637.obj	2977

10	0126.obj	280	54	0490.obj	1644	98	0638.obj	3008
11	0133.obj	311	55	0491.obj	1675	99	0639.obj	3039
12	0162.obj	342	56	0495.obj	1706	100	0648.obj	3070
13	0163.obj	373	57	0501.obj	1737	101	0649.obj	3101
14	0176.obj	404	58	0514.obj	1768	102	0699.obj	3132
15	0186.obj	435	59	0515.obj	1799	103	0702.obj	3163
16	0187.obj	466	60	0516.obj	1830	104	0708.obj	3194
17	0189.obj	497	61	0517.obj	1861	105	0710.obj	3225
18	0202.obj	528	62	0520.obj	1892	106	0722.obj	3256
19	0217.obj	559	63	0521.obj	1923	107	0729.obj	3287
20	0218.obj	590	64	0523.obj	1954	108	0773.obj	3318
21	0219.obj	621	65	0526.obj	1985	109	0774.obj	3349
22	0236.obj	652	66	0527.obj	2016	110	0776.obj	3380
23	0243.obj	683	67	0528.obj	2047	111	0784.obj	3411
24	0250.obj	714	68	0536.obj	2078	112	0786.obj	3442
25	0259.obj	745	69	0557.obj	2109	113	0790.obj	3473
26	0264.obj	776	70	0575.obj	2140	114	0791.obj	3504
27	0267.obj	807	71	0583.obj	2171	115	0792.obj	3535
28	0286.obj	838	72	0584.obj	2202	116	0796.obj	3566
29	0296.obj	869	73	0585.obj	2233	117	0798.obj	3597
30	0300.obj	900	74	0587.obj	2264	118	0799.obj	3628
31	0350.obj	931	75	0588.obj	2295	119	0826.obj	3659
32	0363.obj	962	76	0590.obj	2326	120	0827.obj	3690
33	0388.obj	993	77	0591.obj	2357	121	0833.obj	3721
34	0389.obj	1024	78	0598.obj	2388	122	0842.obj	3752
35	0403.obj	1055	79	0599.obj	2419	123	0907.obj	3783
36	0406.obj	1086	80	0600.obj	2450	124	0913.obj	3814
37	0409.obj	1117	81	0601.obj	2481	125	0917.obj	3845
38	0410.obj	1148	82	0602.obj	2512	126	0926.obj	3876
39	0411.obj	1179	83	0603.obj	2543	127	0928.obj	3907
40	0412.obj	1210	84	0604.obj	2574	128	0931.obj	3938

41	0413.obj	1241	85	0605.obj	2605	129	0938.obj	3969
42	0414.obj	1272	86	0606.obj	2636	130	0939.obj	4000
43	0417.obj	1303	87	0607.obj	2667	131	0940.obj	4031
44	0418.obj	1334	88	0614.obj	2698	132	0942.obj	4062

Tabla G9. Objetos Prueba de la categoría *Cone-Vase*

Índice	**Objeto**	**Pseudo-Objeto**
1	0001.obj	1
10	0017.obj	235
20	0041.obj	495
25	0061.obj	625

Tabla G10. Objetos Prueba de la categoría *Bowl*

Índice	**Objeto**	**Pseudo-Objeto**
1	0004.obj	1
6	0063.obj	156
8	0065.obj	218
14	0079.obj	404

Tabla G11. Objetos Prueba de la categoría *Jar*

Índice	**Objeto**	**Pseudo-Objeto**
1	0030.obj	1
4	0074.obj	187
5	0077.obj	249
6	0099.obj	311

Tabla G12. Objetos Prueba de la categoría Lebrillo

Índice	**Objeto**	**Pseudo-Objeto**
1	0003.obj	1
7	0024.obj	295

11	0062.obj	491
17	0201.obj	785

Tabla G13. Objetos Prueba de la categoría Olla

Índice	**Objeto**	**Pseudo-Objeto**
1	0006.obj	1
3	0035.obj	73
8	0081.obj	253
9	0087.obj	289

Tabla G14. Objetos Prueba de la categoría *Plate*

Índice	**Objeto**	**Pseudo-Objeto**
1	0014.obj	1
2	0093.obj	49
5	0289.obj	193
30	0416.obj	1393

Tabla G15. Objetos Prueba de la categoría *Vessel*

Índice	**Objeto**	**Pseudo-Objeto**
1	0029.obj	1
2	0031.obj	32
3	0045.obj	63
4	0046.obj	94

Tabla G16. Objetos Prueba del 2° experimento

Índice	**Objeto**	**Pseudo-Objeto**
1	0001.obj	1
3	0003.obj	13
4	0004.obj	19
6	0006.obj	31

14	0014.obj	79
17	0017.obj	97
24	0024.obj	139
28	0029.obj	163
29	0030.obj	169
30	0031.obj	175
33	0035.obj	193
39	0041.obj	229
43	0045.obj	253
44	0046.obj	259
56	0061.obj	331
57	0062.obj	337
58	0063.obj	343
60	0065.obj	355
66	0074.obj	391
69	0077.obj	409
71	0079.obj	421
73	0081.obj	433
78	0087.obj	463
84	0093.obj	499
89	0099.obj	529
185	0201.obj	1105
258	0289.obj	1543
358	0416.obj	2143

Anexo H: Resultados cualitativos

Para entender los resultados cualitativos se deben tener en cuenta las siguientes consideraciones:

- Cada resultado cualitativo está compuesto por 2 imágenes adjuntas
- La primera imagen (izquierda) es el objeto original ingresado al modelo. Recordemos que el modelo recibe como entrada un archivo NPY el cual contiene un arreglo compuesto por los valores RGB de la secuencia de imágenes del objeto original. La imagen mostrada es 1 de estas imágenes pertenecientes a la secuencia de 72 imágenes.
- La segunda imagen (derecha) es el objeto reconstruido (malla poligonal) por el modelo y visualizado mediante el software Blender.
- Se realizaron 4 pruebas por cada categoría del primer experimento
- Para el segundo experimento se procesaron los mismos objetos, sin embargo, se utilizó el modelo entrenado en todas las categorías.

Resultados del 1° experimento

Resultados del modelo de la categoría "*Cone-Vase*"

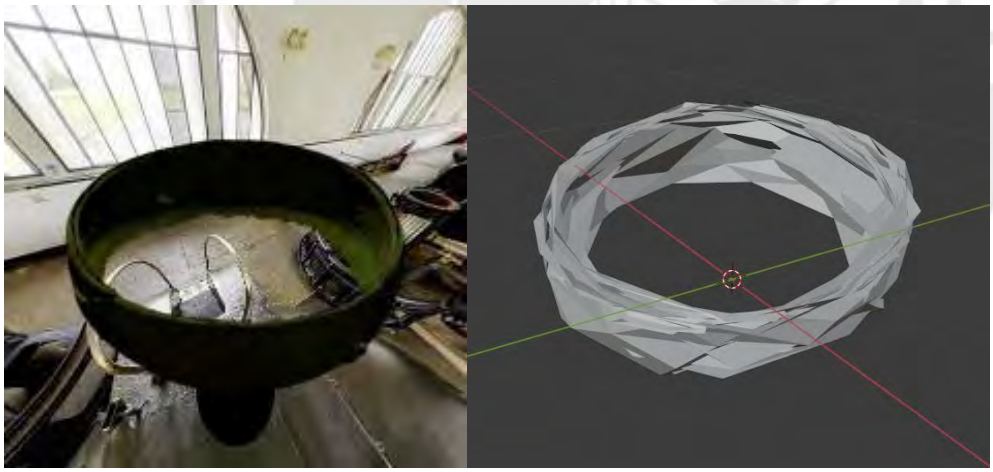


Figura H1. Objeto "0001"

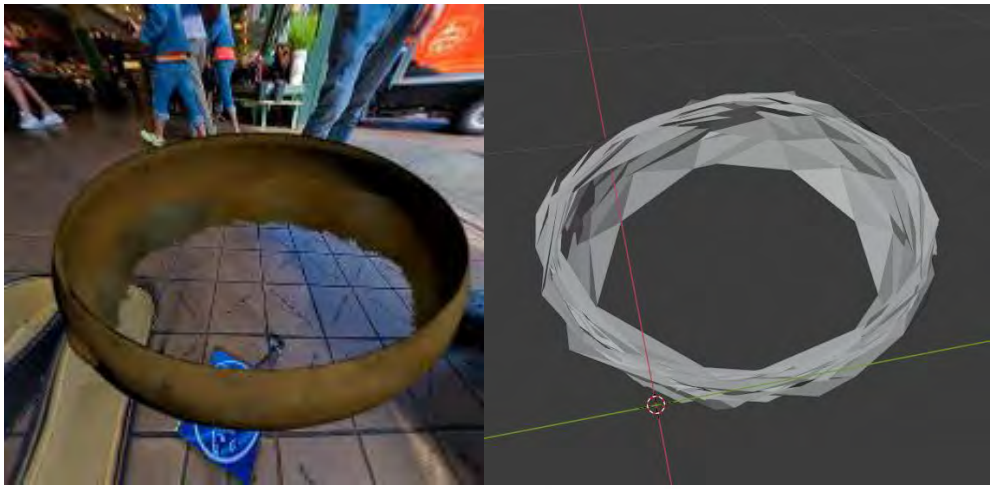


Figura H2. Objeto “0017”

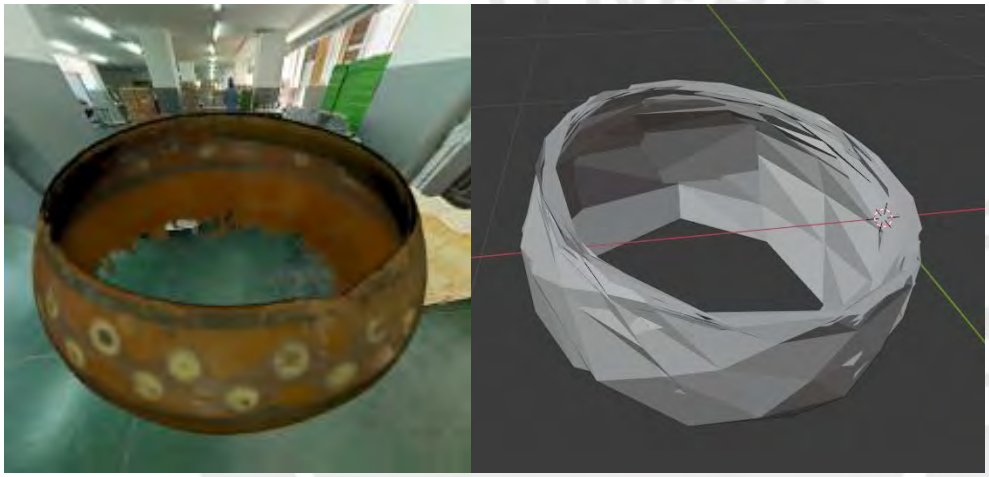


Figura H3. Objeto “0041”

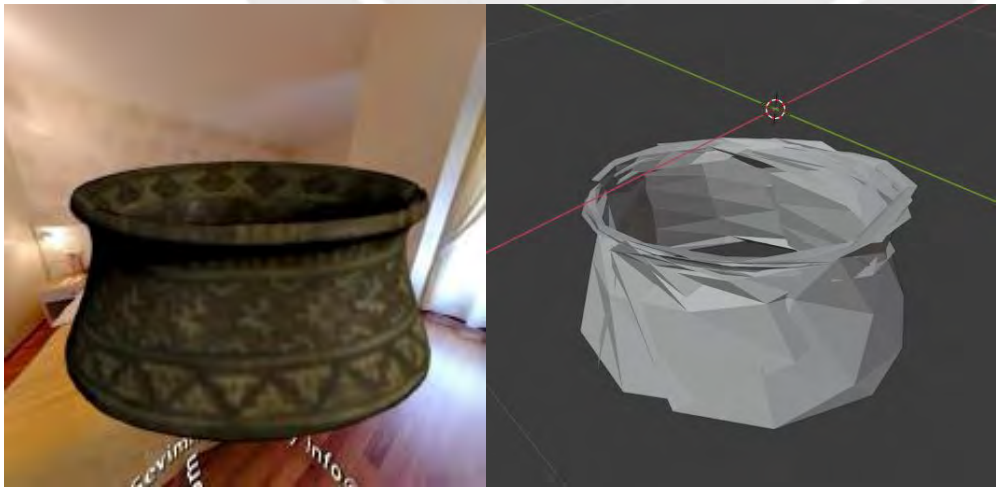


Figura H4. Objeto “0061”

Resultados del modelo de la categoría “*Bowl*”

Figura H5. Objeto “0004”

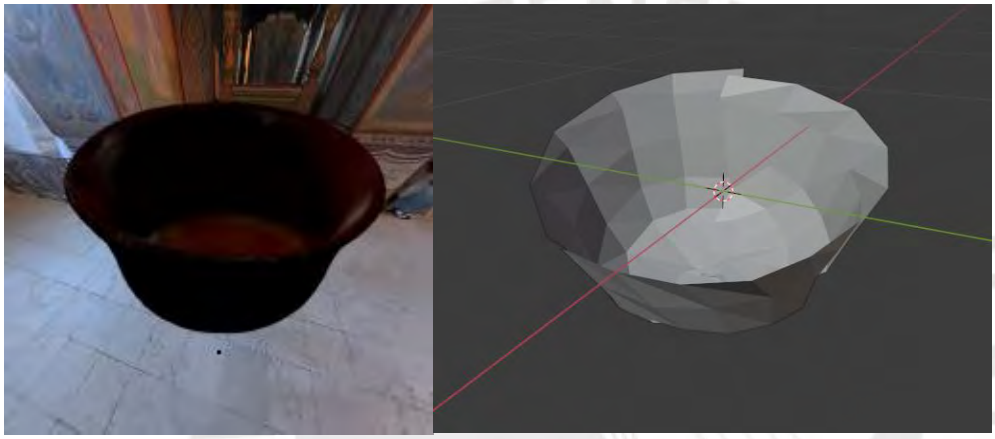


Figura H6. Objeto “0063”

Figura H7. Objeto “0065”

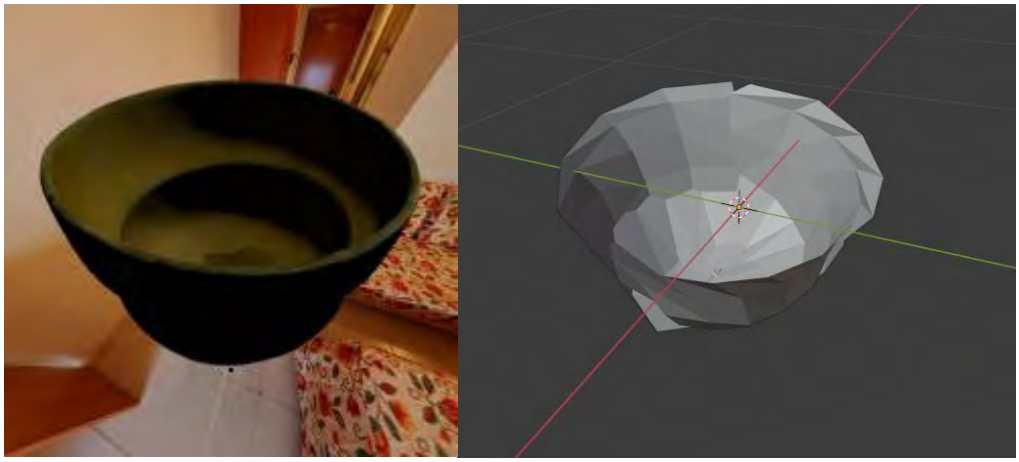


Figura H8. Objeto “0079”

Resultados del modelo de la categoría “*Jar*”

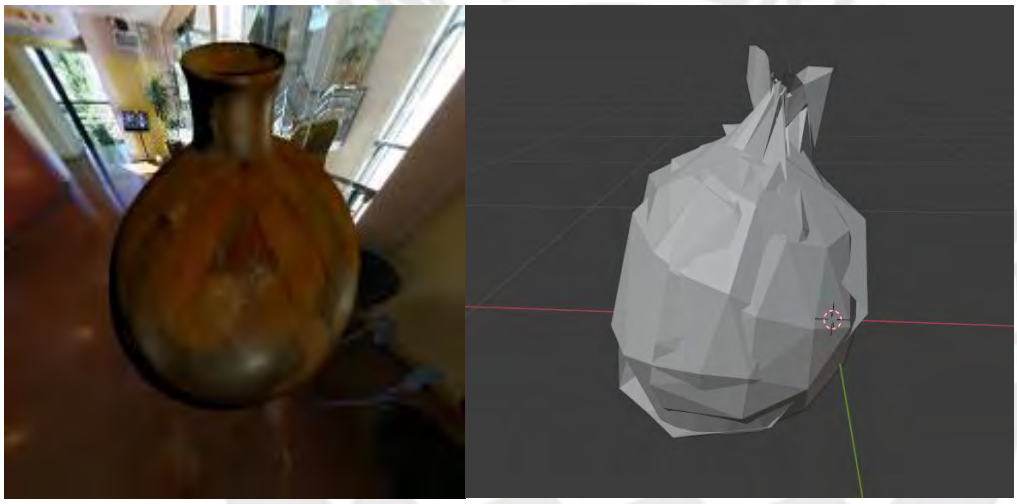


Figura H9. Objeto “0030”

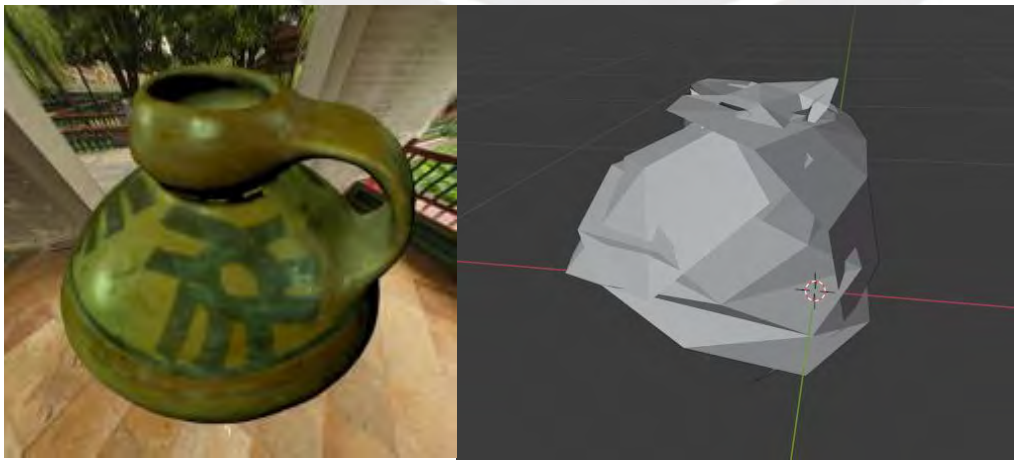


Figura H10. Objeto “0074”

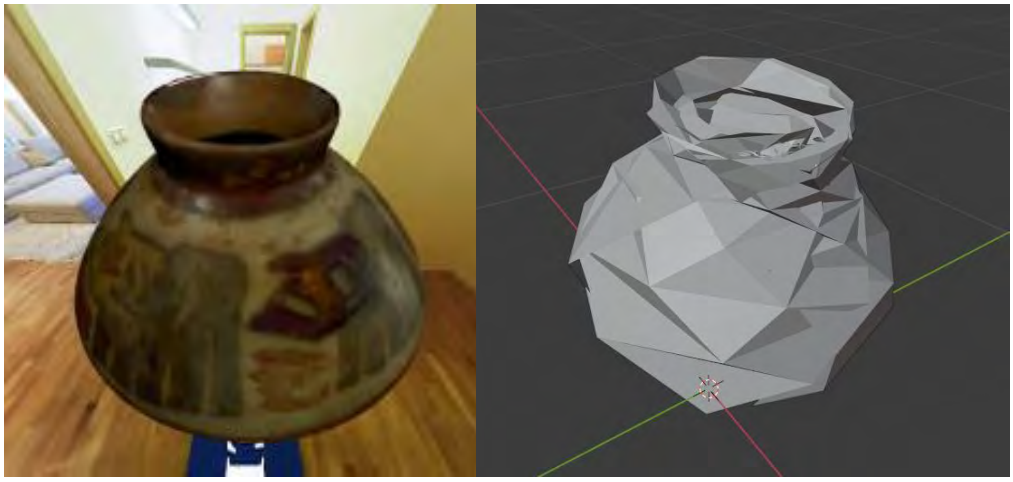


Figura H11. Objeto “0077”

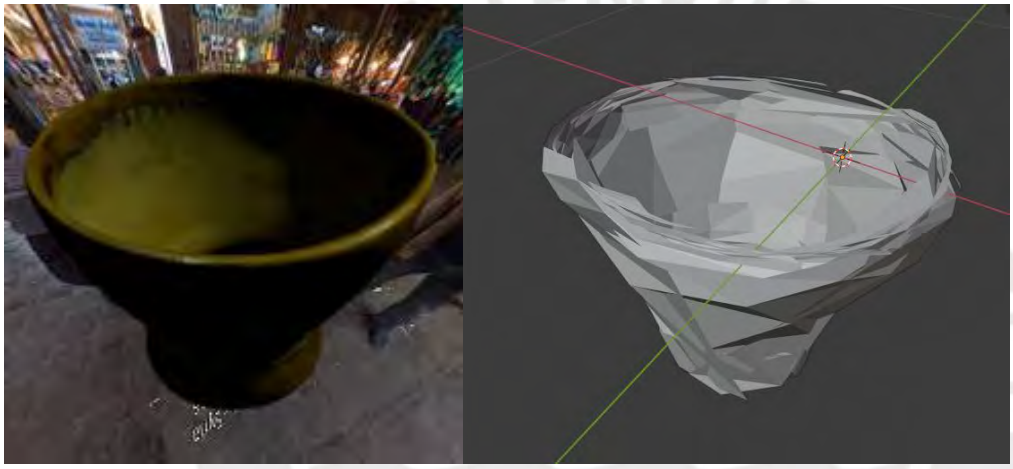


Figura H12. Objeto “0099”

Resultados del modelo de la categoría “Lebrillo”

Figura H13. Objeto “0003”

Figura H14. Objeto “0024”

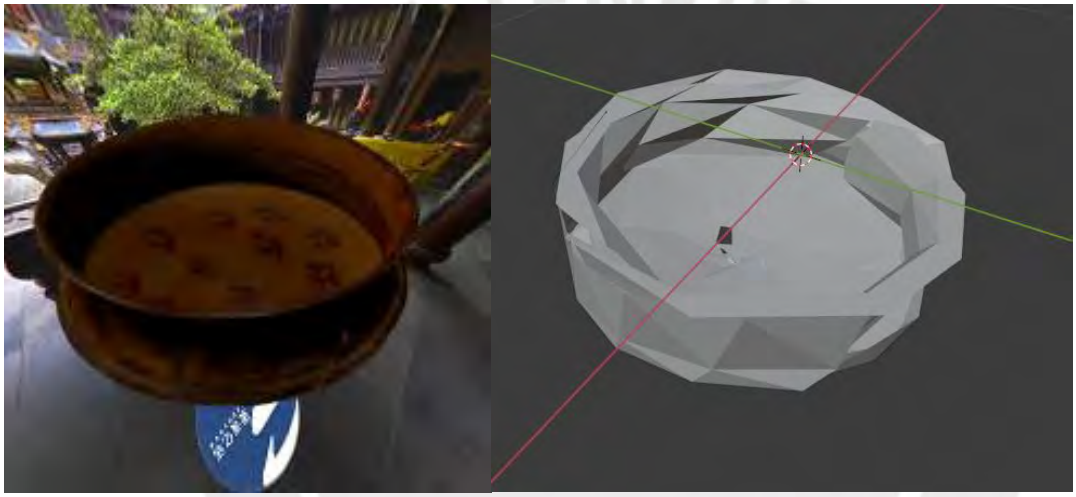


Figura H15. Objeto “0062”

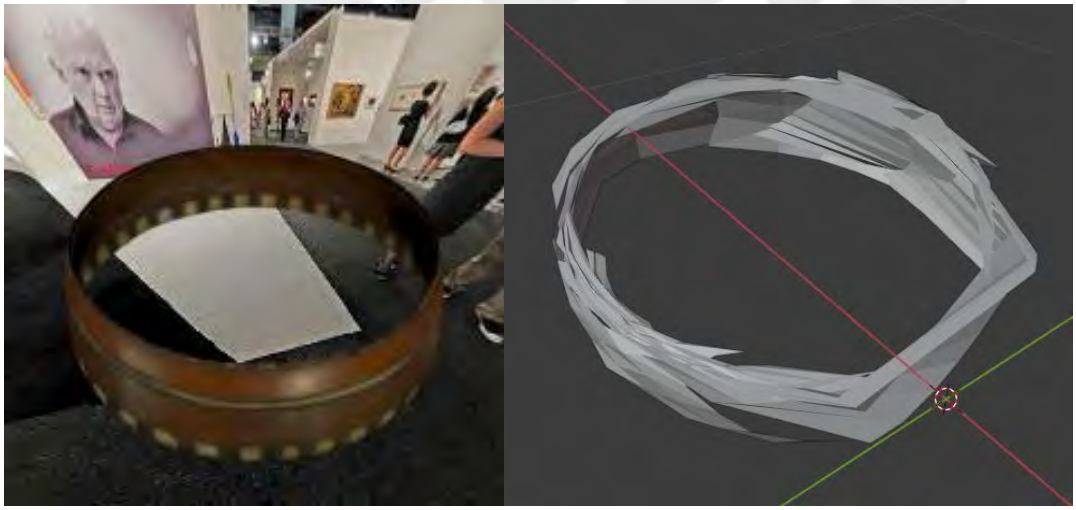


Figura H16. Objeto “0201”

Resultados del modelo de la categoría “Olla”

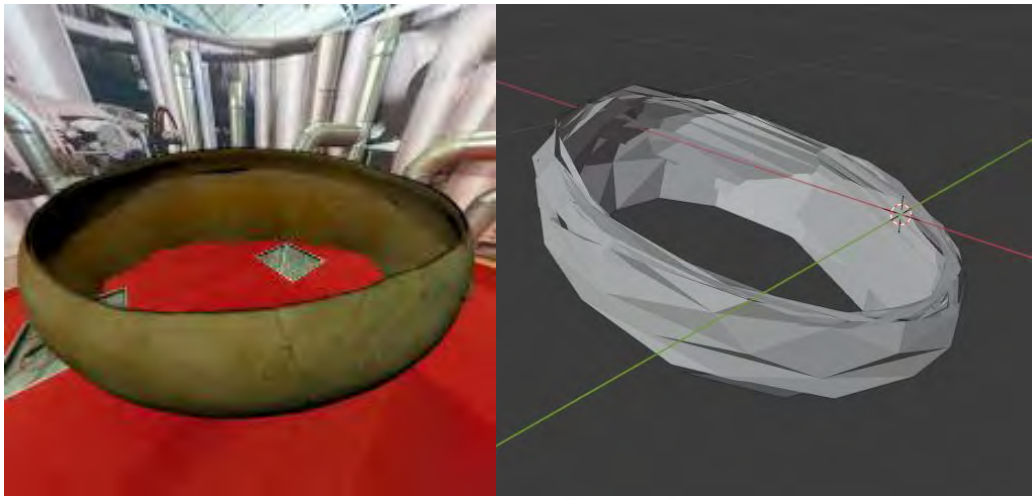


Figura H17. Objeto “0006”

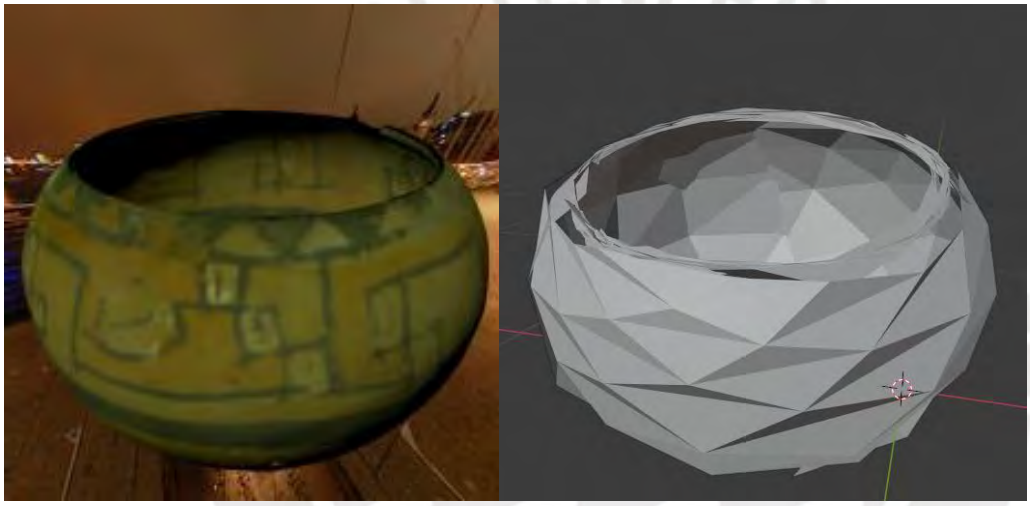


Figura H18. Objeto “0035”

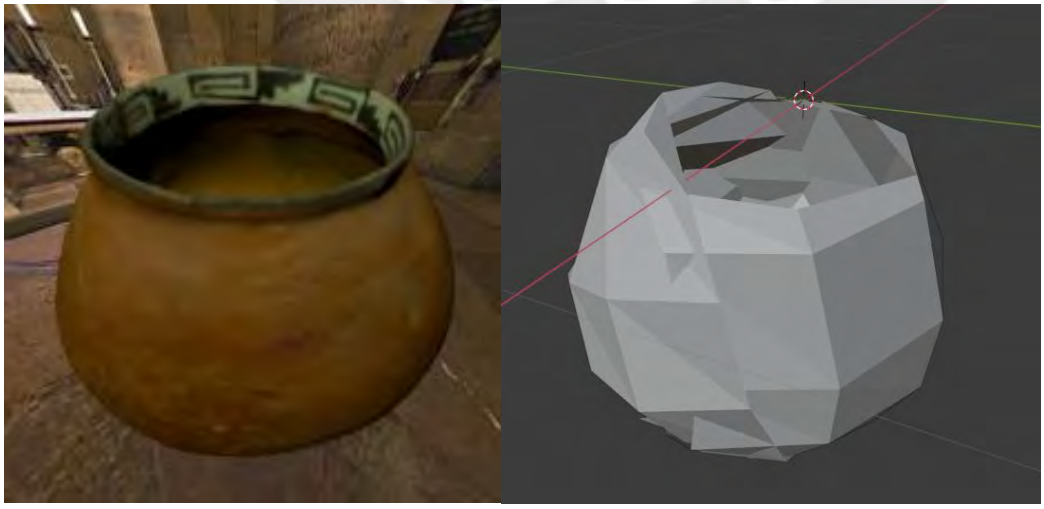


Figura H19. Objeto “0081”

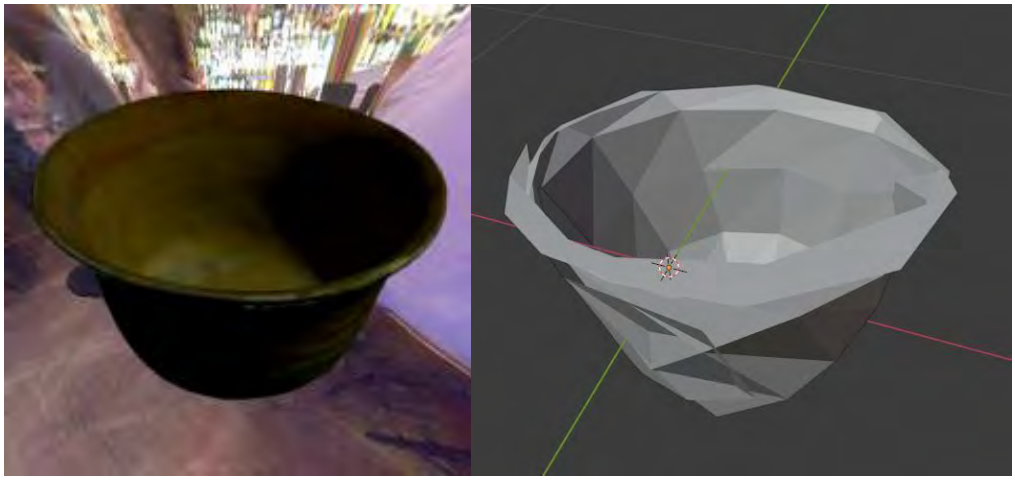


Figura H20. Objeto “0087”

Resultados del modelo de la categoría “*Plate*”

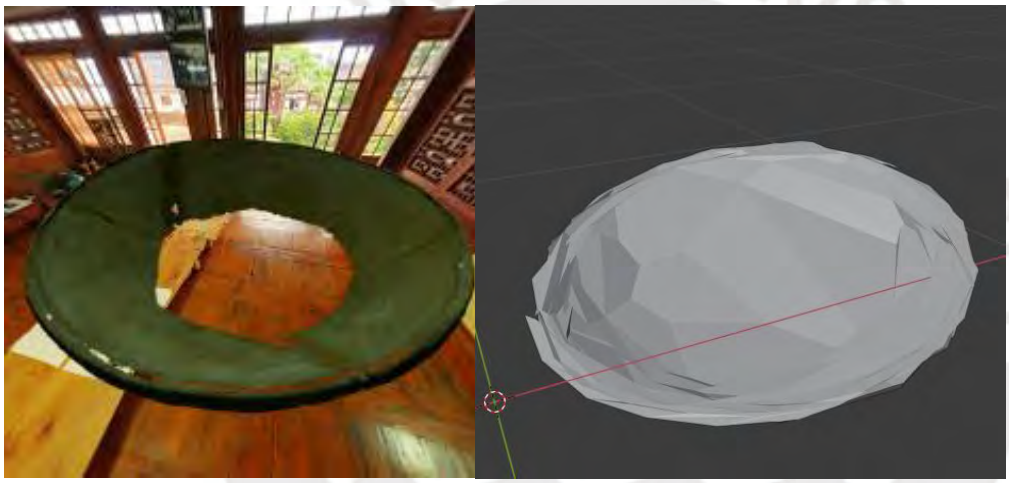


Figura H21. Objeto “0014”

Figura H22. Objeto “0093”

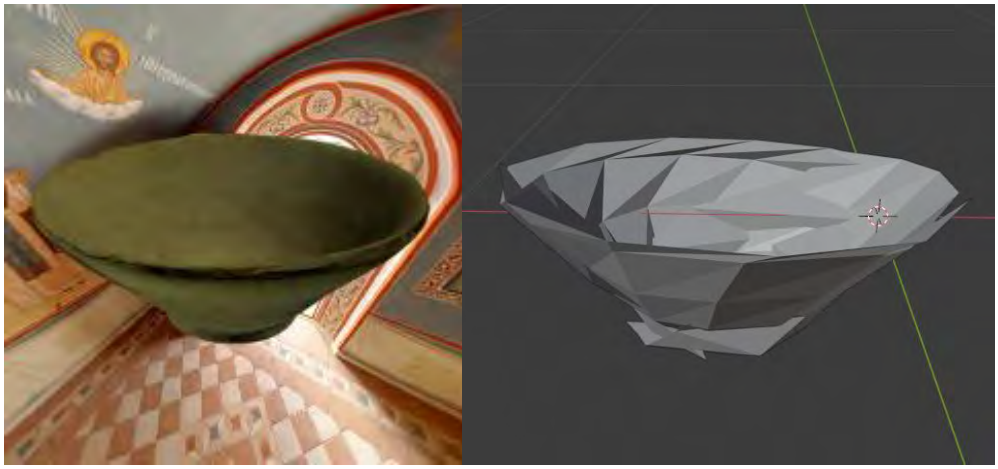


Figura H23. Objeto “0289”

Figura H24. Objeto “0416”

Resultados del modelo de la categoría “*Vessel*”

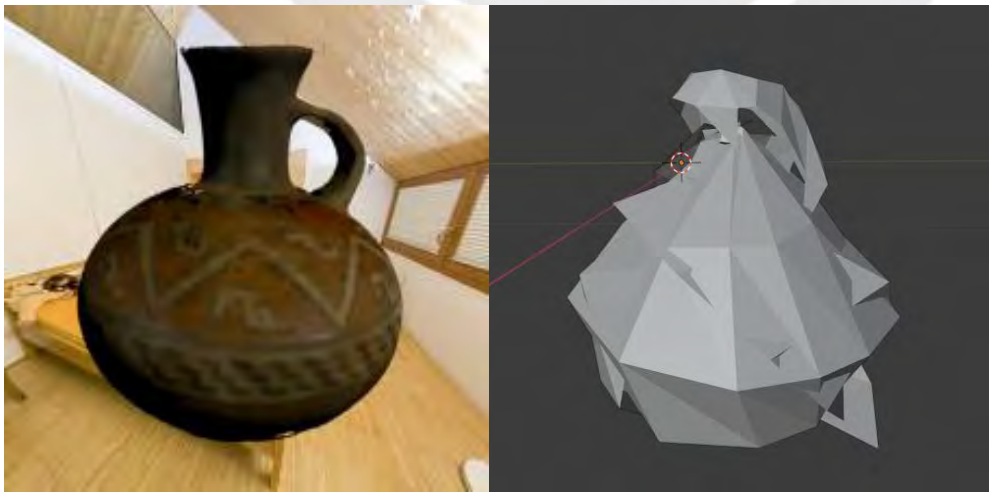


Figura H25. Objeto “0029”

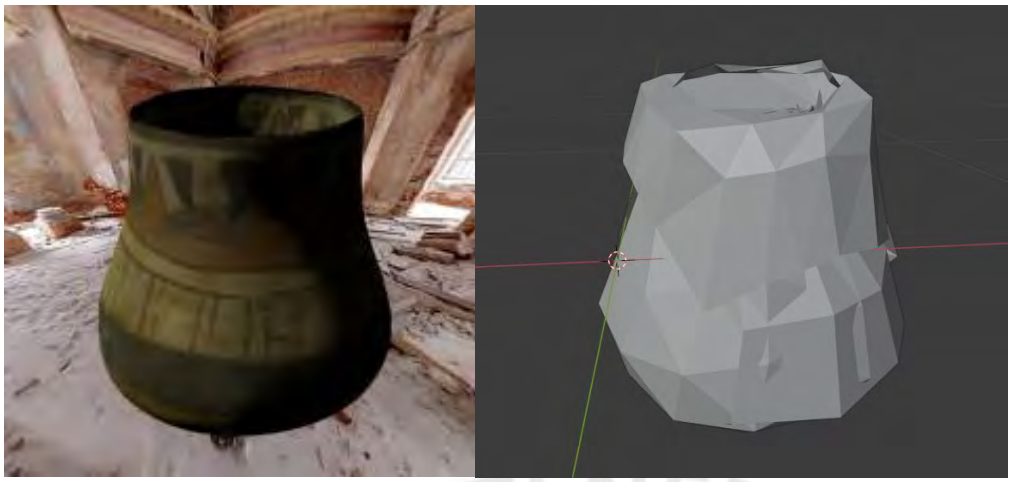


Figura H26. Objeto “0031”

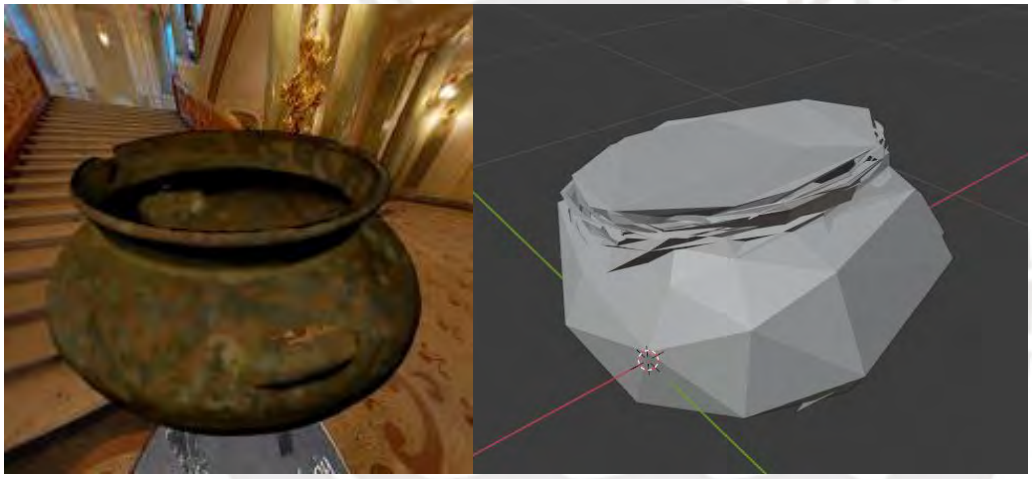


Figura H27. Objeto “0045”

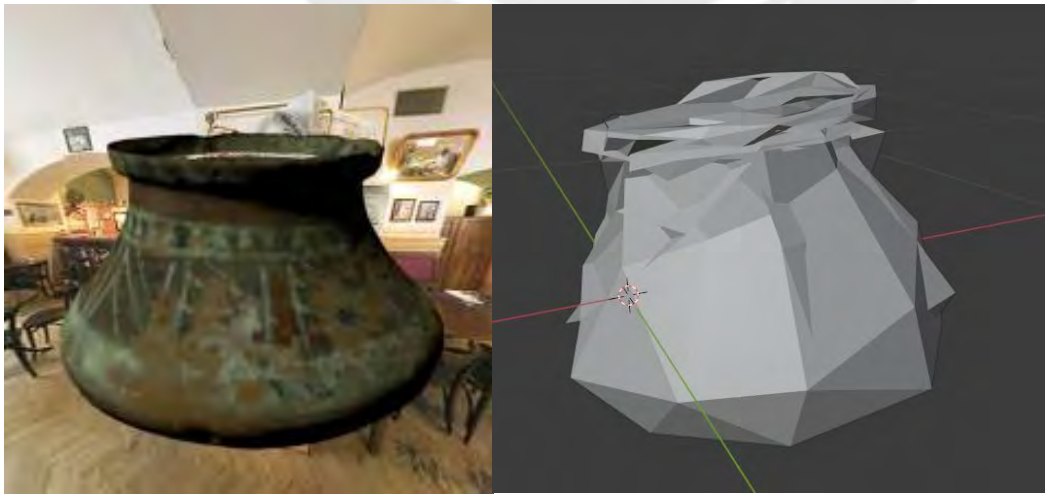


Figura H28. Objeto “0046”

Resultados del 2° experimento

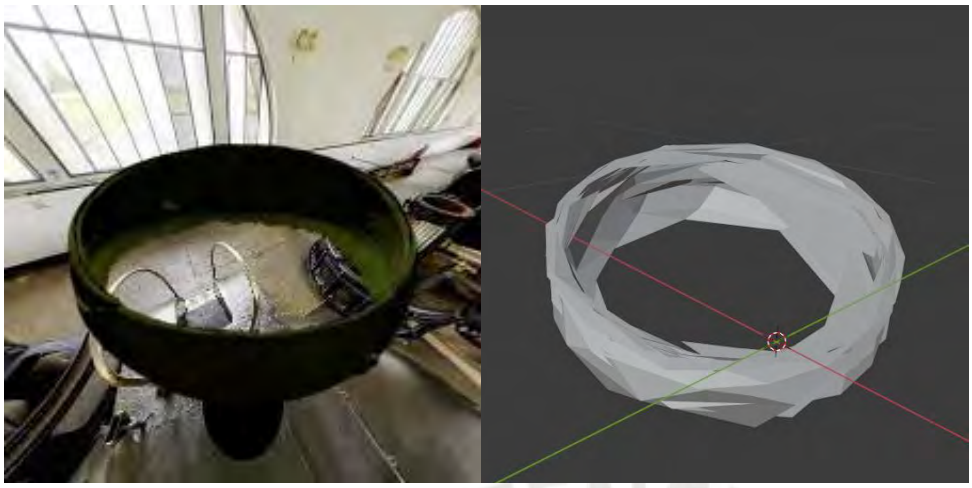


Figura H29. Objeto “0001”

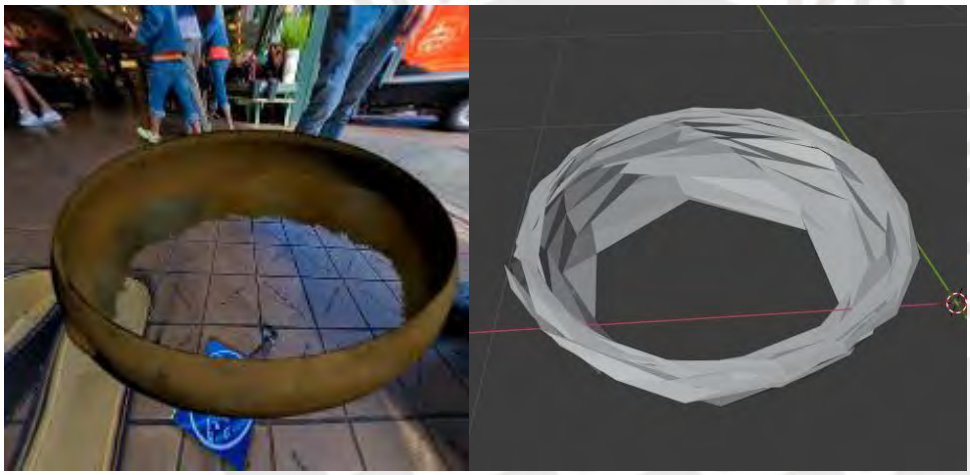


Figura H30. Objeto “0017”

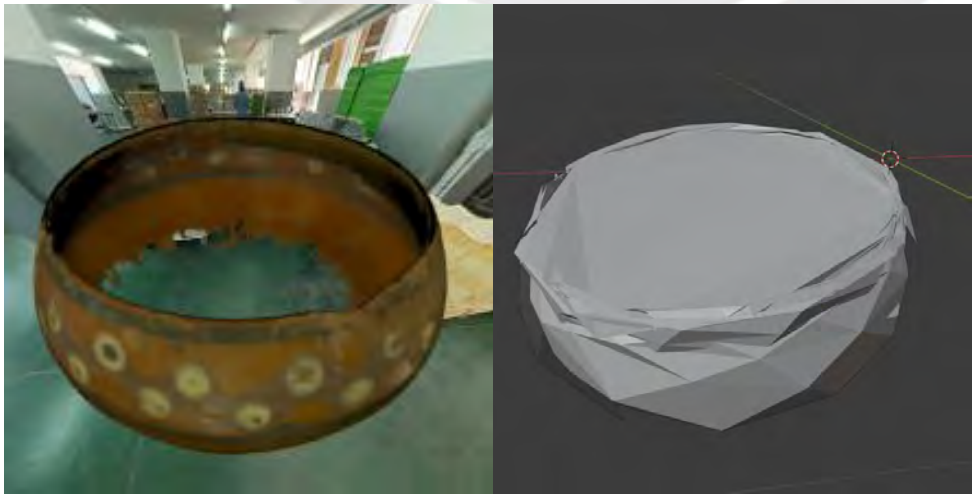


Figura H31. Objeto “0041”

Figura H32. Objeto “0061”

Figura H33. Objeto “0004”

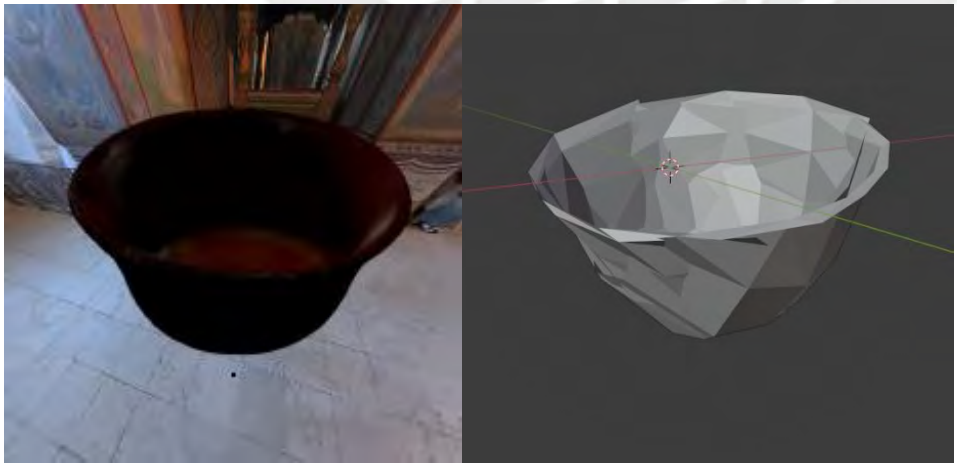


Figura H34. Objeto “0063”

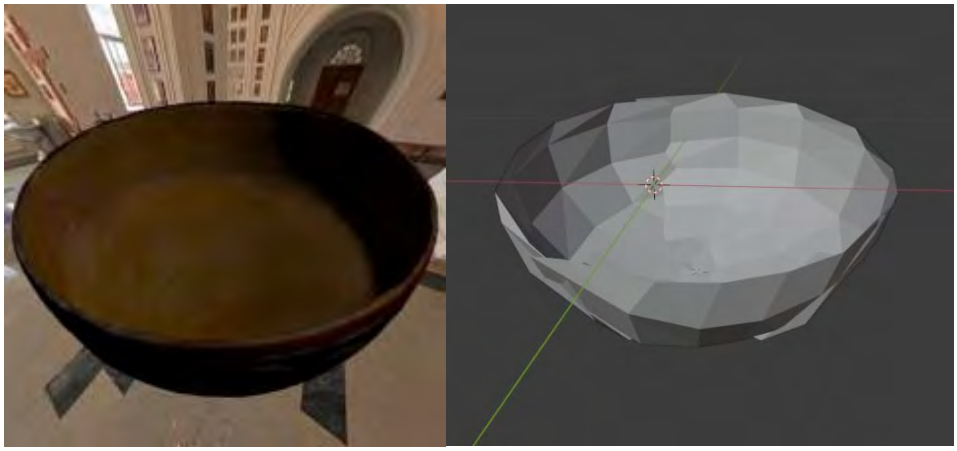


Figura H35. Objeto “0065”

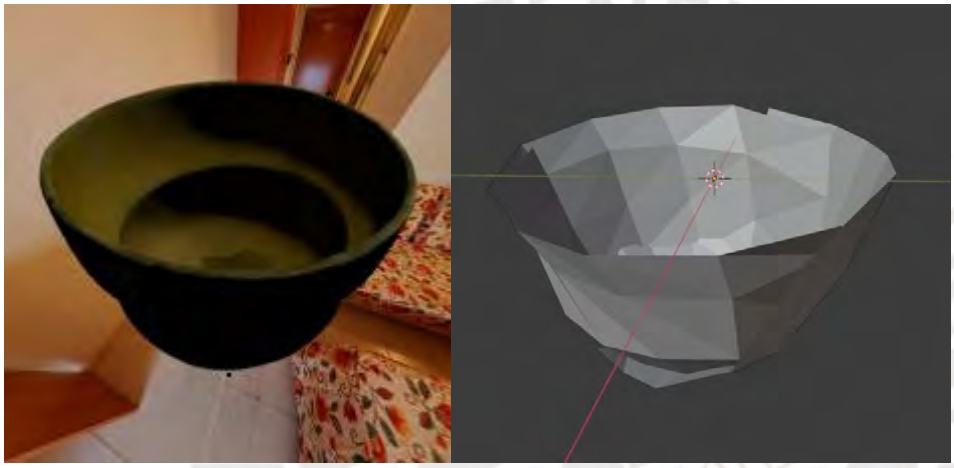


Figura H36. Objeto “0079”

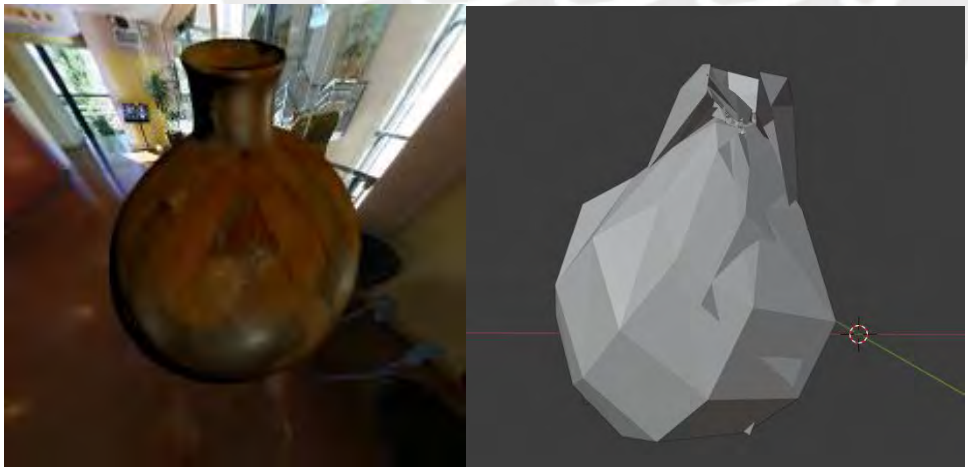


Figura H37. Objeto “0030”

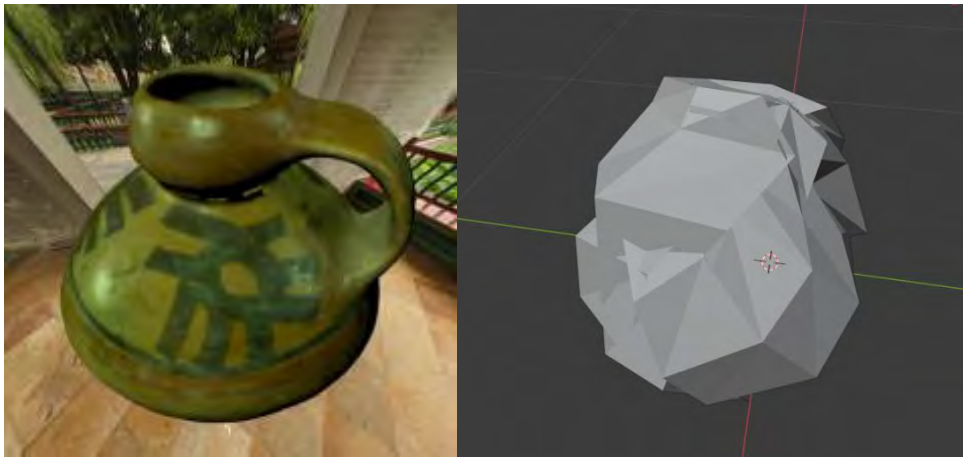


Figura H38. Objeto “0074”

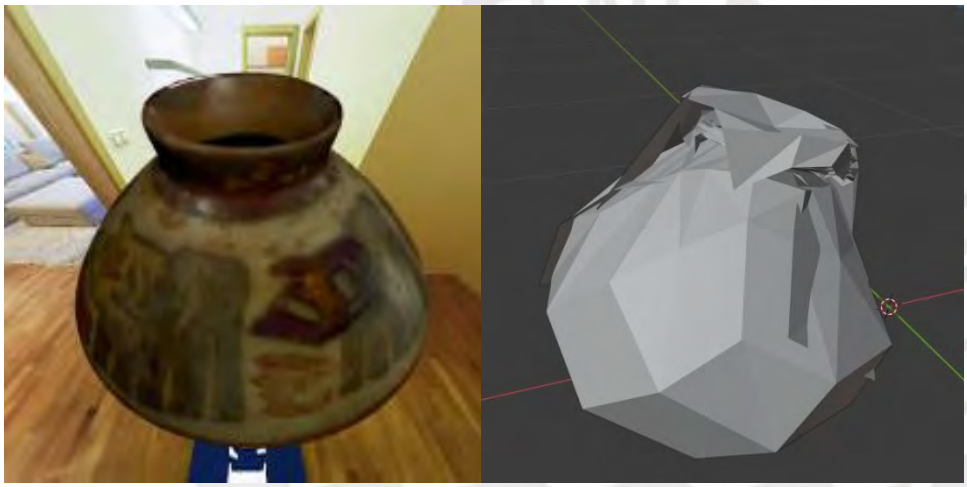


Figura H39. Objeto “0077”

Figura H40. Objeto “0099”

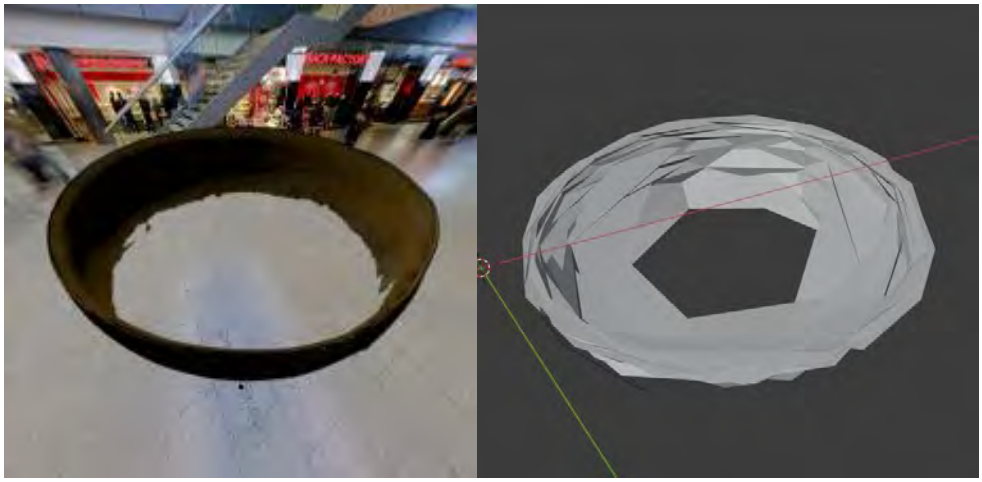


Figura H41. Objeto “0003”

Figura H42. Objeto “0024”

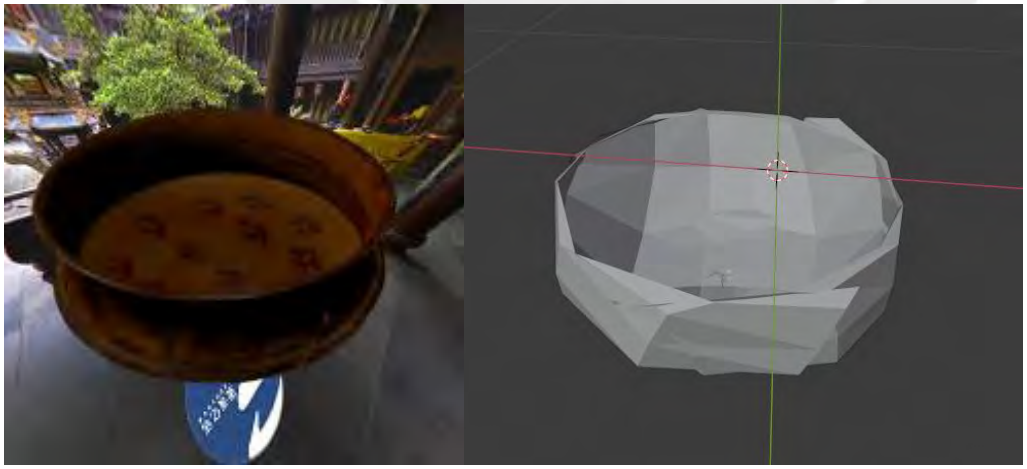


Figura H43. Objeto “0062”

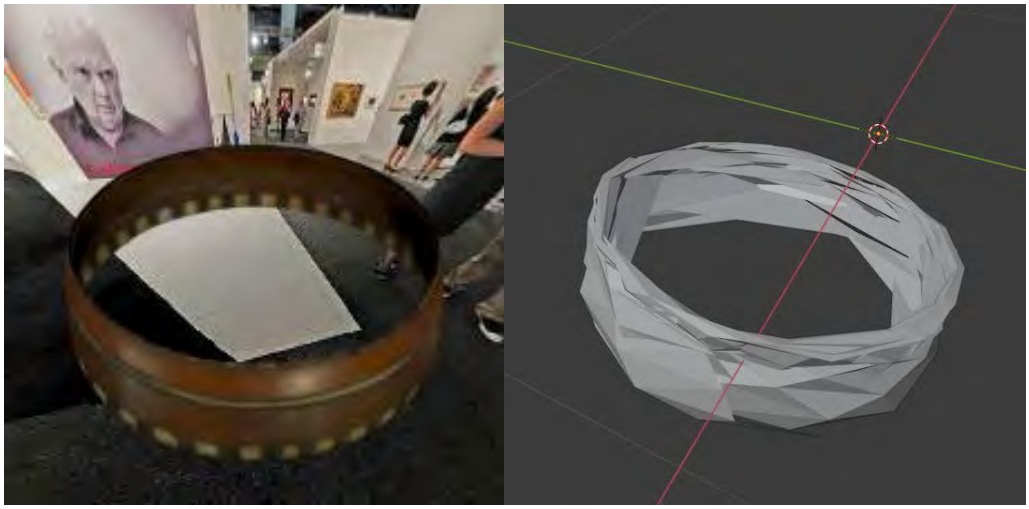


Figura H44. Objeto “0201”

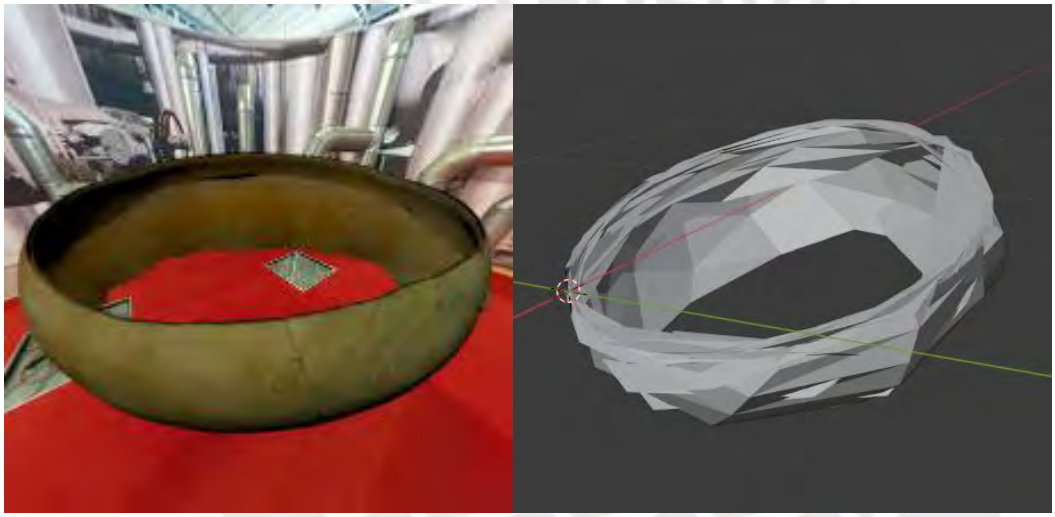


Figura H45. Objeto “0006”

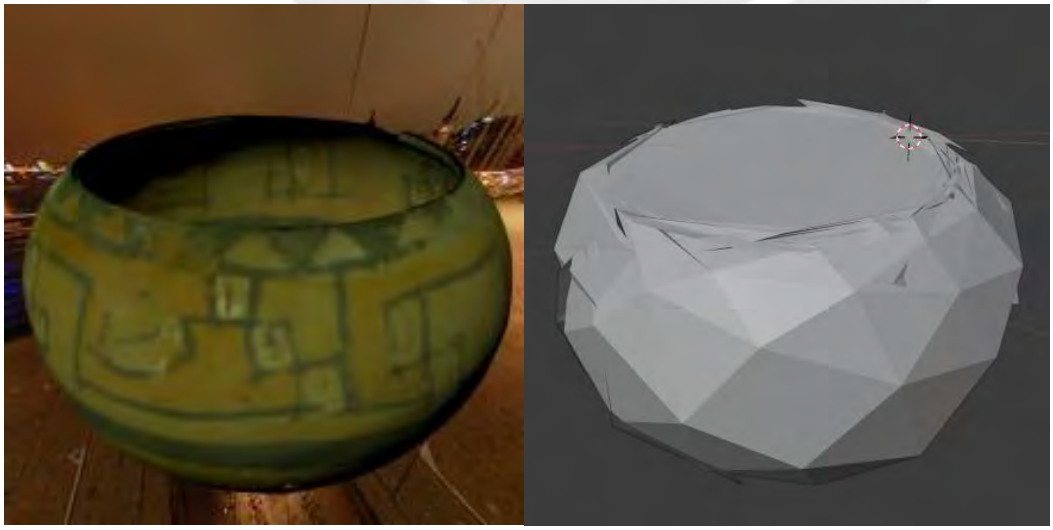


Figura H46. Objeto “0035”

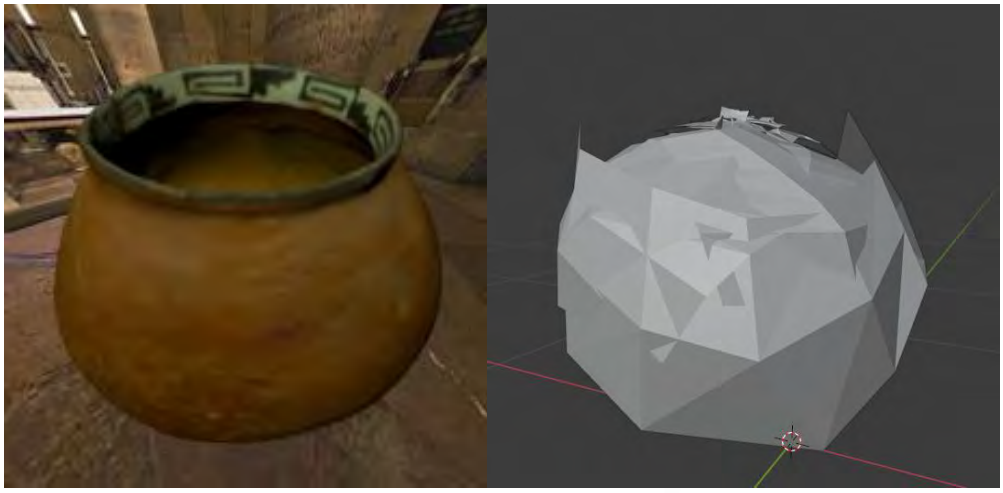


Figura H47. Objeto “0081”

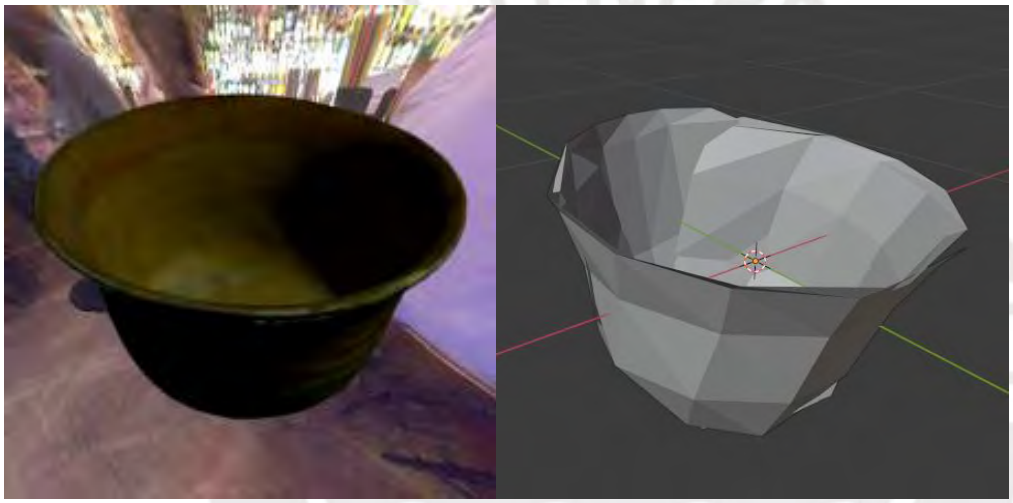


Figura H48. Objeto “0087”

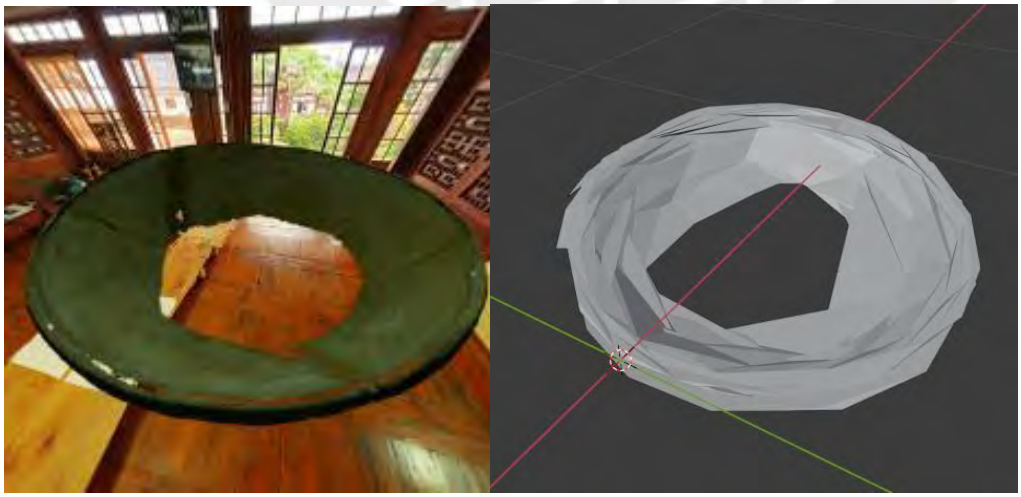


Figura H49. Objeto “0014”

Figura H50. Objeto “0093”

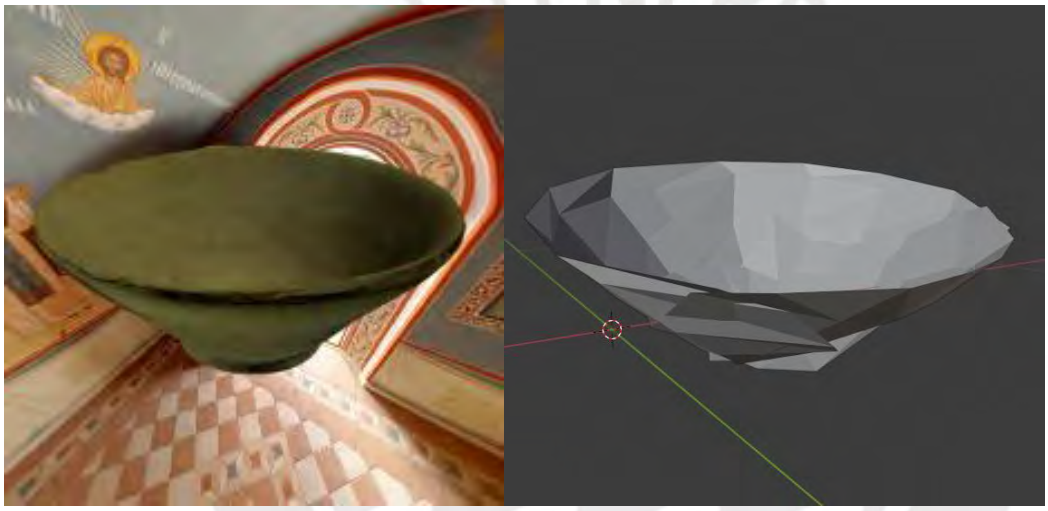


Figura H51. Objeto “0289”

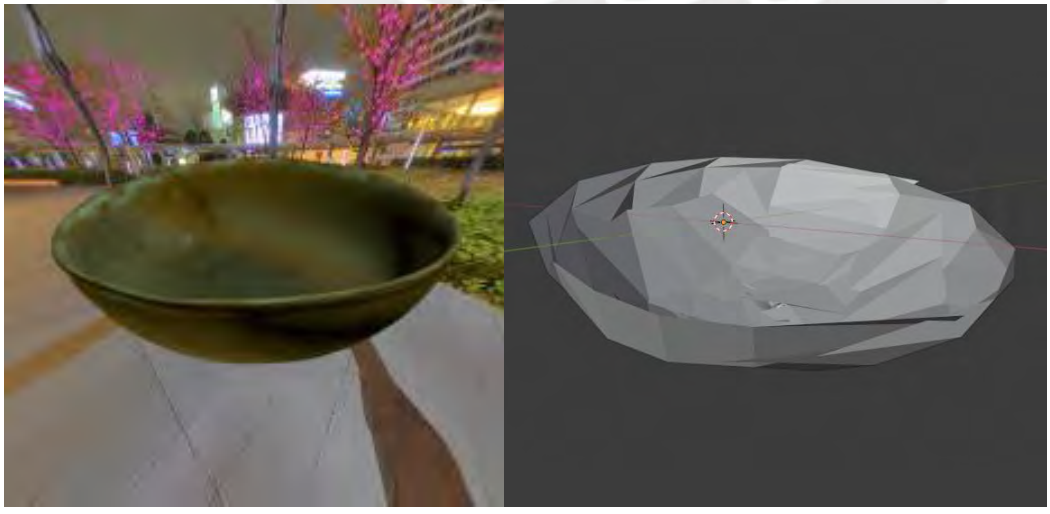


Figura H52. Objeto “0416”

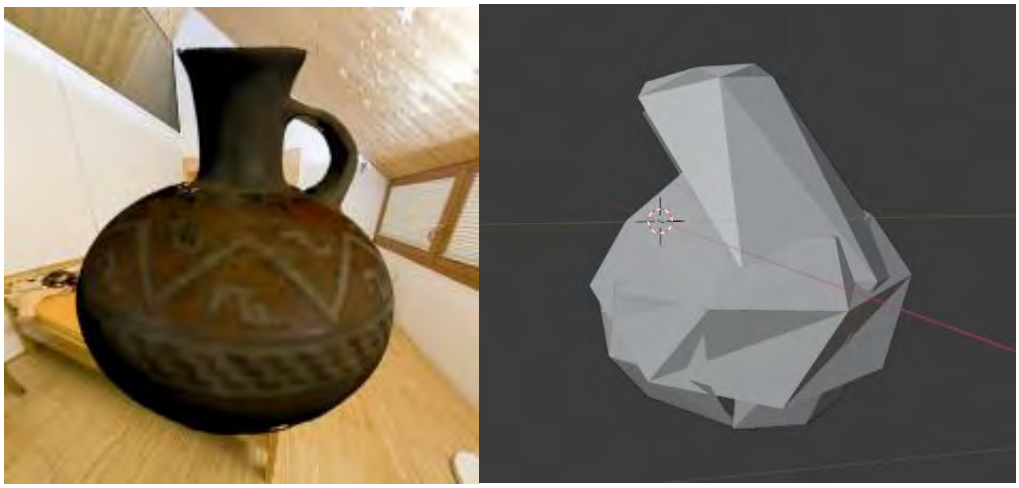


Figura H53. Objeto “0029”

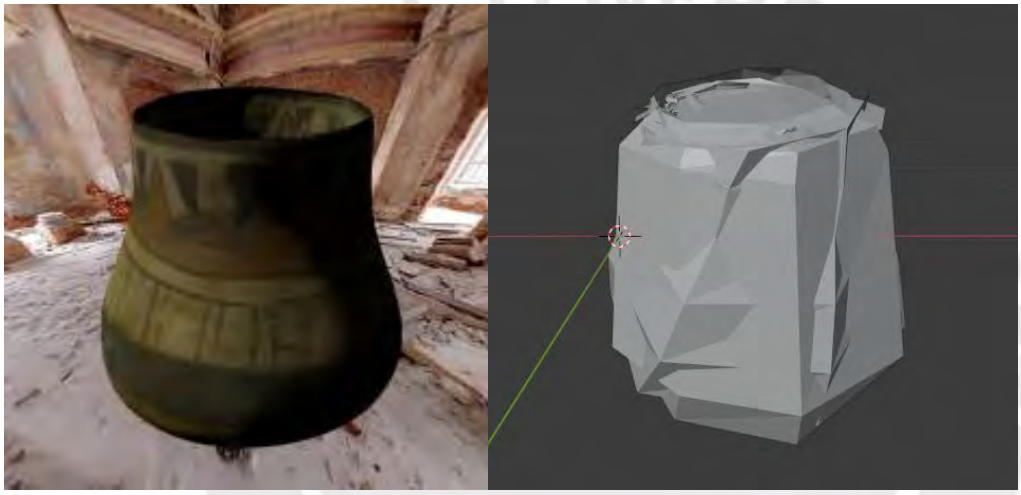


Figura H54. Objeto “0031”

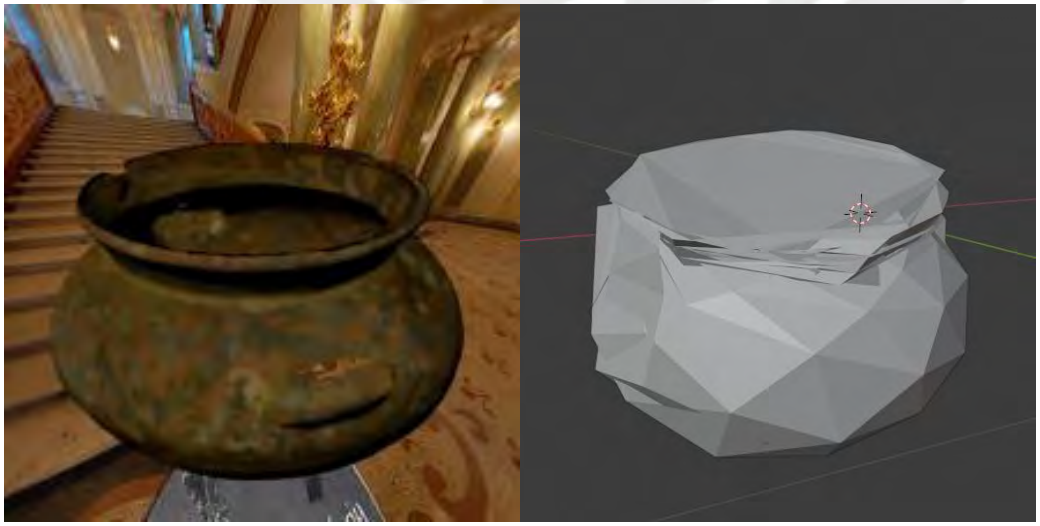


Figura H55. Objeto “0045”

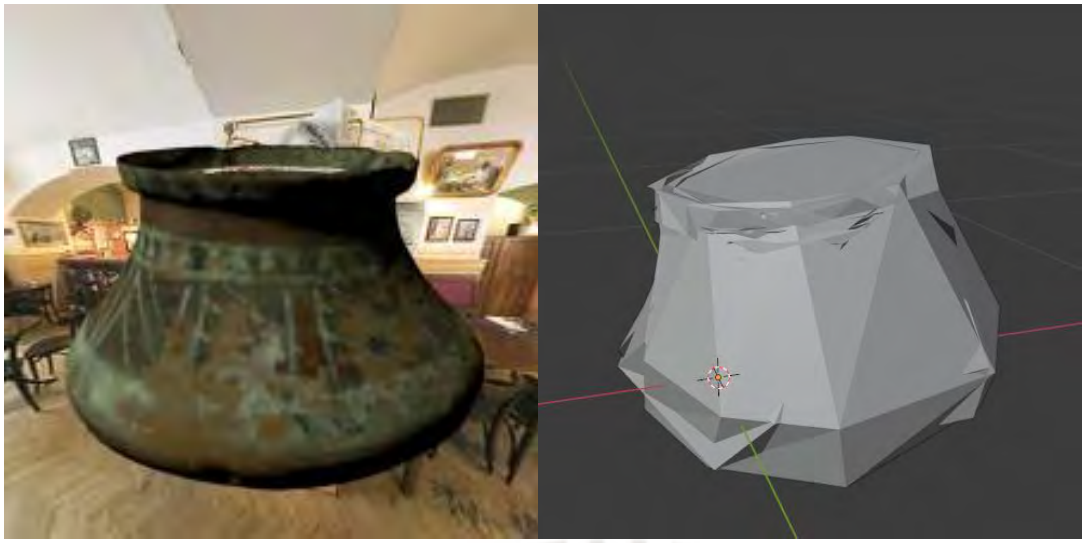


Figura H56. Objeto “0046”

Anexo I: *Script* desarrollado "*pipeline*.py"

Para acceder al código desarrollado del *script* "*pipeline*.py" se provee el siguiente enlace de GitHub. Este código se presenta como un procedimiento el cual recibe como parámetros el nombre del archivo (videograbación) y la categoría asignada. Todo el proceso ejecutado consta de la conversión del video (archivo MP4 o AVI) en un archivo NPY, lo que incluye el escalamiento del video, generación de fotogramas, procesamiento de valores RGB de las imágenes, selección de fotogramas y creación del archivo NPY procesable por un modelo.

https://github.com/ErnieLud/tesis/blob/main/pipeline.py

Anexo J: Resultados de pruebas funcionales

Para entender los resultados de las pruebas funcionales unitarias se deben tener en cuenta las siguientes consideraciones. Por cada una de las pruebas se adjunta:

- Objetivo: descripción del objetivo final y/o resultado esperado de la prueba
- Precondición: el conjunto de acciones a realizar, estados previos o contexto que debe cumplirse para poder llevar a cabo la prueba funcional en mención
- Descripción: en caso sea necesario modificar algunos campos previo a la ejecución estos serán especificados en esta sección, en este caso, con los campos nos referimos a los elementos de la interfaz desarrollada
- Resultados obtenidos: contiene las capturas de pantalla y breve descripción de todo lo ocurrido una vez ejecutada la prueba, es en esta sección donde podemos llegar a una conclusión sobre el éxito de la prueba funcional

A continuación, se muestran la lista de videos construidos durante el resultado esperado número 5 del proyecto de tesis. Estos videos serán utilizados para la ejecución de las pruebas. En la primera figura observamos las duraciones y resoluciones de los videos, donde el nombre de cada archivo es una cadena de 4 dígitos (Ejemplo: “0337”), el cual pertenece a la pieza arqueológica utilizada para la construcción del video. En la segunda figura observamos la categoría a la que pertenece dicho objeto/video.

	Resolución		
Duración (s)	**540p**	**720p**	**1080p**
10	"0013"	"0253"	"0433"
15	"0031"	"0259"	"0463"
20	"0079"	"0337"	"0499"
25	"0139"	"0343"	"0529"
30	"0169"	"0355"	"1105"
35	"0175"	"0409"	"1543"
40	"0193"	"0421"	"2143"

Figura J1. Tabla de videos de prueba (elaboración propia)

Objeto	Categoría
"0013"	LEBRILLO
"0031"	OLLA
"0079"	PLATE
"0139"	LEBRILLO
"0169"	JAR
"0175"	VESSEL
"0193"	OLLA
"0253"	VESSEL
"0259"	VESSEL
"0337"	LEBRILLO
"0343"	BOWL
"0355"	BOWL
"0409"	JAR
"0421"	BOWL
"0433"	OLLA
"0463"	OLLA
"0499"	PLATE
"0529"	JAR
"1105"	LEBRILLO
"1543"	PLATE
"2143"	PLATE

Figura J2. Tabla de categorías de los videos de prueba (elaboración propia)

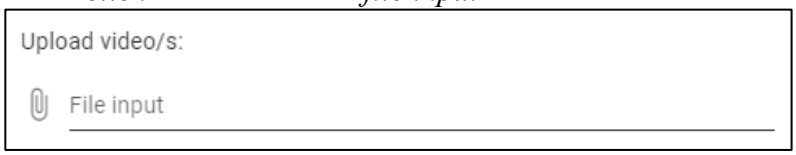
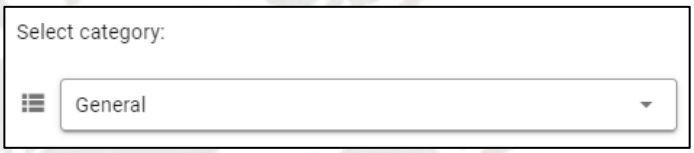
Lista de pruebas funcionales realizadas

1. Añadir un video o un conjunto de videos
2. Seleccionar y observar los ejemplos de las categorías en el visualizador de objetos 3D
3. Eliminar un video de la tabla de datos de videos añadidos y empezar el procesamiento de los videos restantes
4. Descargar los resultados obtenidos a partir del procesamiento de los videos

Resultados de pruebas funcionales

Tabla J1. Resultados de la prueba funcional unitaria 1

Prueba 1. Añadir un video o un conjunto de videos	
Objetivo	Demostrar que la interfaz desarrollada permite al usuario cargar un conjunto de videos al servidor y añadirlos a la lista de videos cargados, los cuales se encuentran listos para ser procesados. Además, para esta prueba se usarán los 21 videos construidos en el resultado esperado 5 del proyecto de tesis.

Precondición	• Ubicarse en la vista "*Home*" y en el primer componente • Hacer *click* en el elemento "*file input*"  Figura J3. Elemento "*file input*" del primer componente • Repetir el siguiente proceso hasta haber seleccionado los 21 videos de prueba desarrollados o Seleccionar 1 o más videos dentro de los formatos permitidos (MP4 y AVI) que cumplan con las características adecuadas para ser procesadas por la herramienta (videograbaciones a piezas arqueológicas bajo los estándares establecidos) o Hacer *click* en el elemento "*combo box*"  Figura J4. Elemento "*combo box*" del primer componente o Seleccionar 1 de las categorías (la que corresponda según la lista de la figura J2) o Hacer *click* en el primer botón azul, el cual carga los videos al servidor y les asigna la categoría seleccionada  Figura J5. Primer botón azul del primer componente
Descripción (campos)	Campo "*Upload video/s*": 1 o más archivos de video Campo "*Select category*": Cualquiera de las 8 opciones ("*Bowl*", "*Cone-Vase*", "*Jar*", "Lebrillo", "Olla", "*Plate*", "*Vessel*", "General")
Resultados obtenidos	Al ejecutar todas las tareas listadas en la precondición obtenemos el siguiente estado en nuestra interfaz gráfica (véase las figuras). En conclusión, sí se logró el objetivo puesto que encontramos a los 21 videos correctamente agregados, paginados de 5 en 5 en la tabla, con sus respectivas categorías asignadas, subidos al servidor y listos para ser procesados.

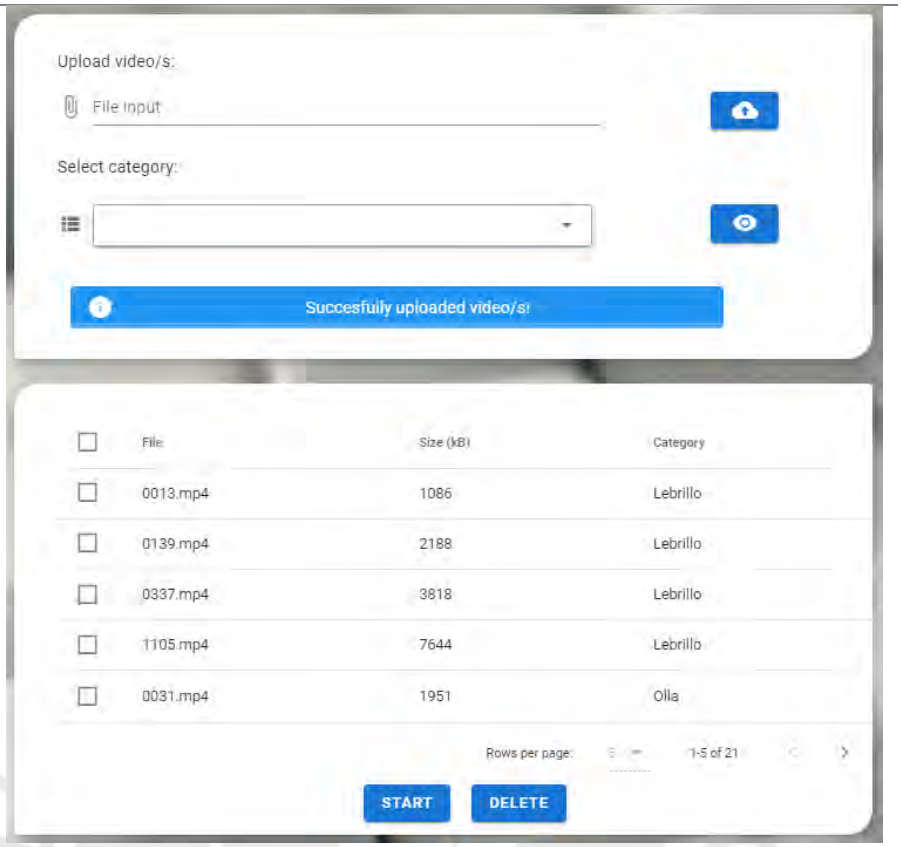


Figura J6. Resultados obtenidos (1) de la primera prueba funcional

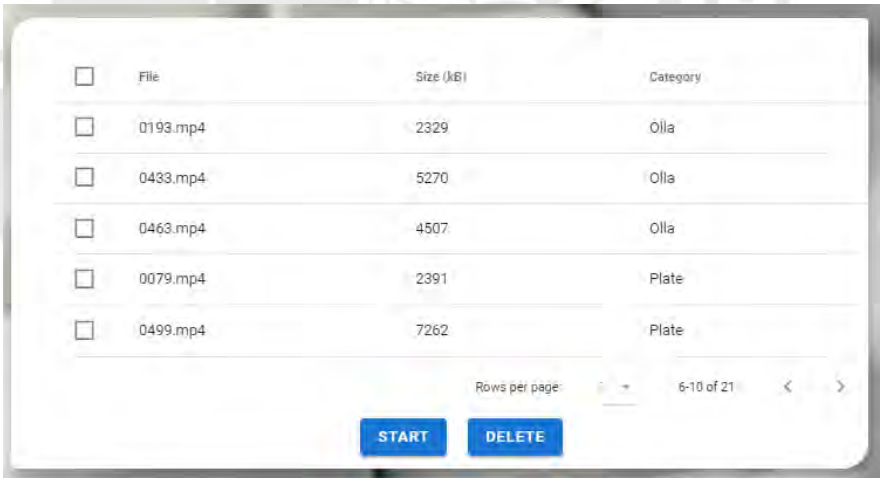


Figura J7. Resultados obtenidos (2) de la primera prueba funcional

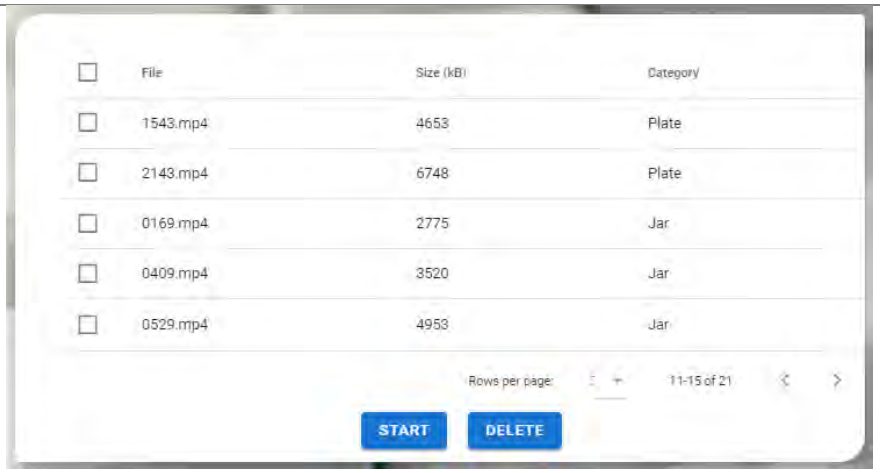


Figura J8. Resultados obtenidos (3) de la primera prueba funcional

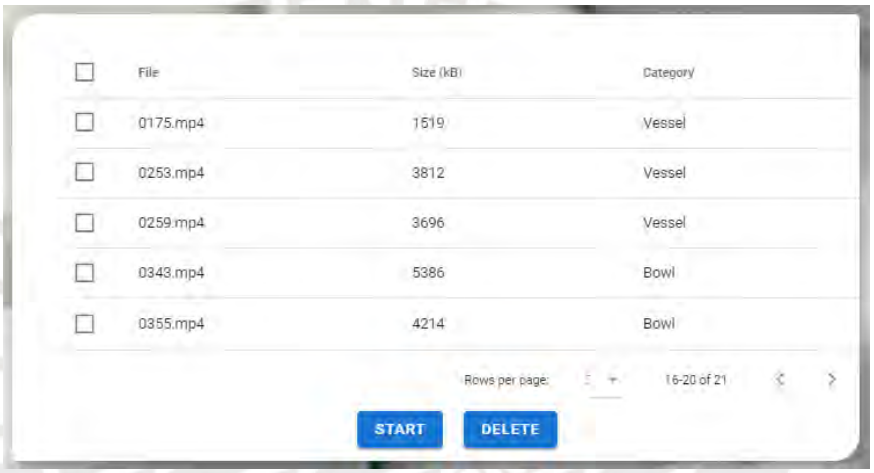


Figura J9. Resultados obtenidos (4) de la primera prueba funcional

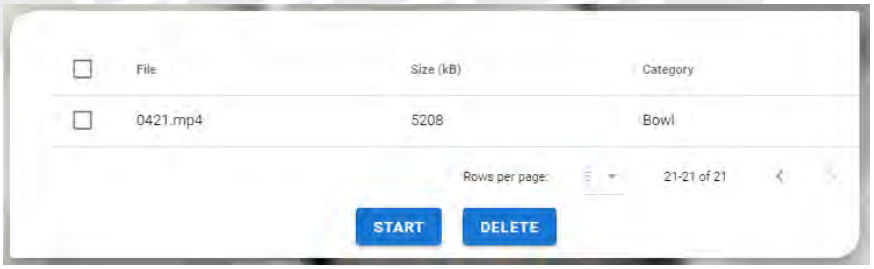
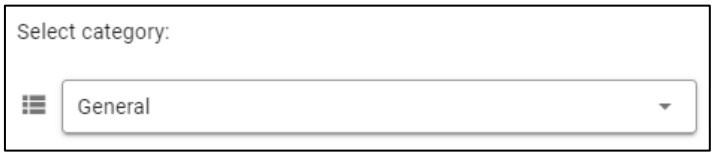
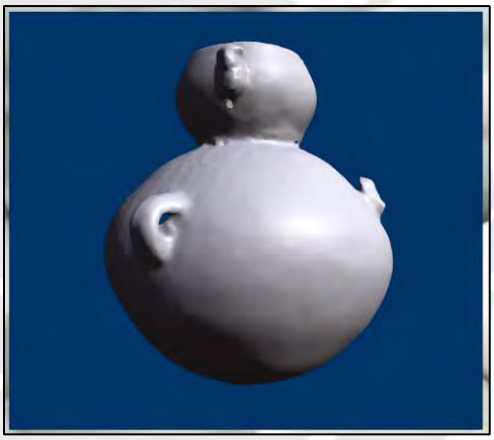


Figura J10. Resultados obtenidos (5) de la primera prueba funcional

Tabla J2. Resultados de la prueba funcional unitaria 2

Prueba 2. Seleccionar y observar los ejemplos de las categorías en el visualizador de objetos 3D	
Objetivo	Demostrar el funcionamiento del visualizador 3D implementado que nos debe permitir mover la cámara alrededor del objeto y rotarlo automáticamente. Además, cada categoría debe mostrar una pieza arqueológica como ejemplo.

Precondición	• Ubicarse en la vista "*Home*" y en el primer componente • Hacer *click* en el elemento "*combo box*"  Figura J11. Elemento "*combo box*" del primer componente • Repetir el siguiente proceso hasta que se hayan seleccionado todas las categorías posibles o Seleccionar 1 de las categorías listadas en el "*combo box*" o Hacer *click* en el segundo botón azul, el cual muestra el ejemplo de pieza arqueológica en el visualizador 3D implementado  Figura J12. Segundo botón azul del primer componente o Mover la cámara del visualizador 3D  Figura J13. Visualizador 3D de la primera vista o Activar/desactivar la rotación automática haciendo *click* en el botón blanco  Figura J14. Botón de rotación para el visualizador 3D
Descripción (campos)	Campo "*Select category*": Cualquiera de las 8 opciones ("*Bowl*", "*Cone-Vase*", "*Jar*", "Lebrillo", "Olla", "*Plate*", "*Vessel*", "General")
Resultados obtenidos	Al realizar los pasos mencionados, por cada categoría se muestra 1 ejemplo único de pieza arqueológica perteneciente a dicha categoría en el visualizador 3D. Este último nos permite mover la cámara y activar/cancelar la rotación automática para un mejor análisis del objeto. Véanse las siguientes figuras de todos los ejemplos mostrados.

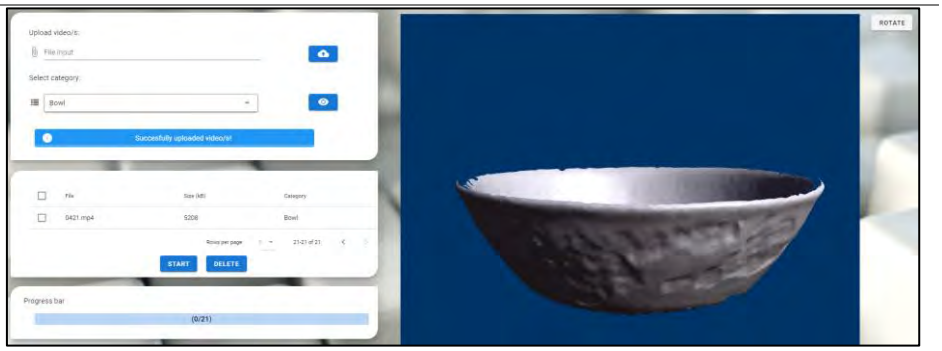


Figura J15. Ejemplo de la categoría "Bowl"

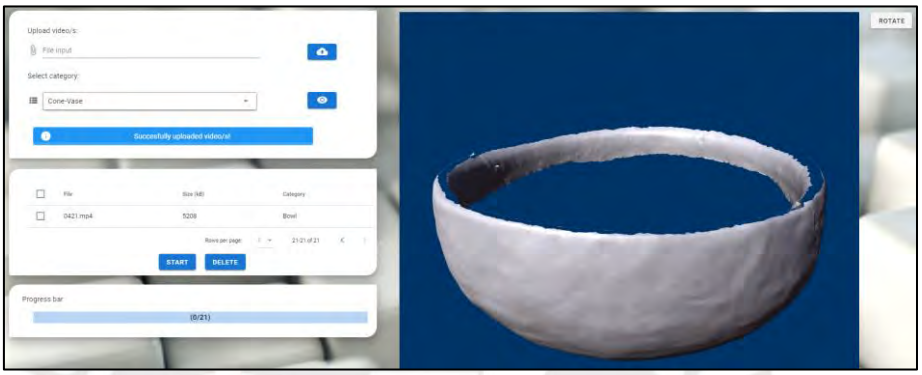


Figura J16. Ejemplo de la categoría "*Cone-Vase*"

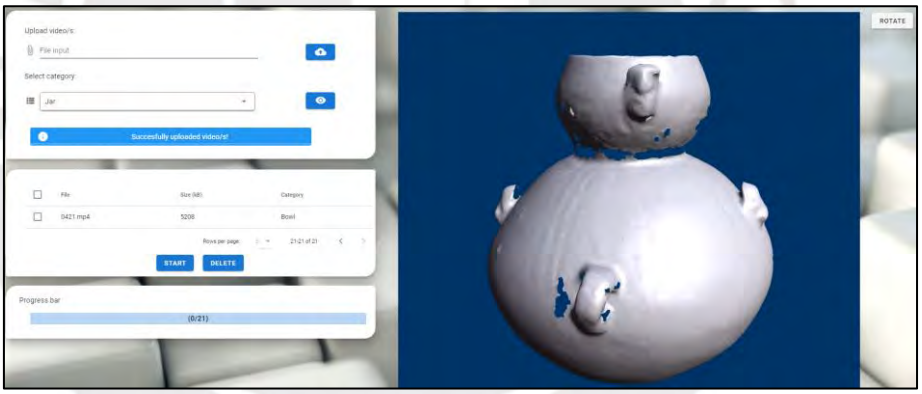


Figura J17. Ejemplo de la categoría "*Jar*"

Figura J18. Ejemplo de la categoría "Lebrillo"

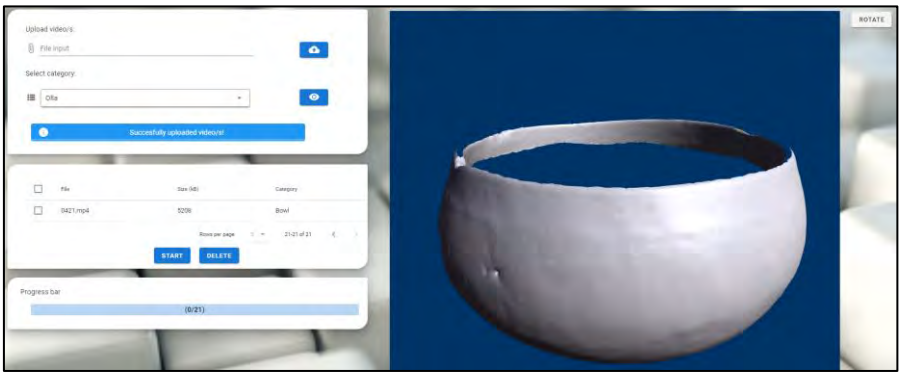


Figura J19. Ejemplo de la categoría "Olla"

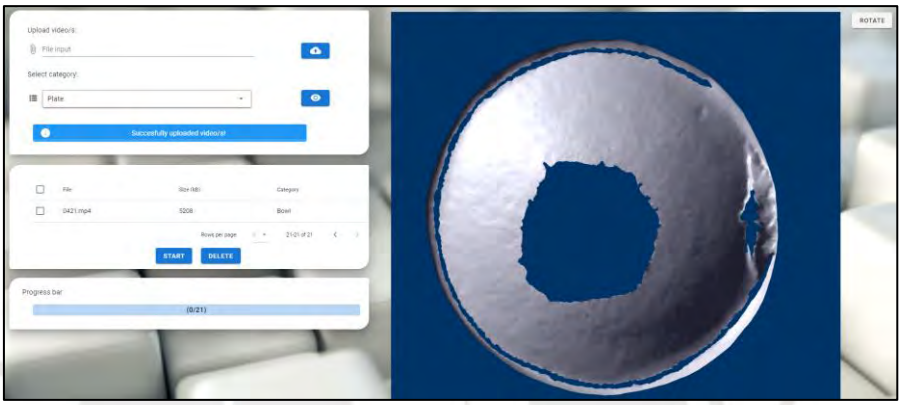


Figura J20. Ejemplo de la categoría "*Plate*"

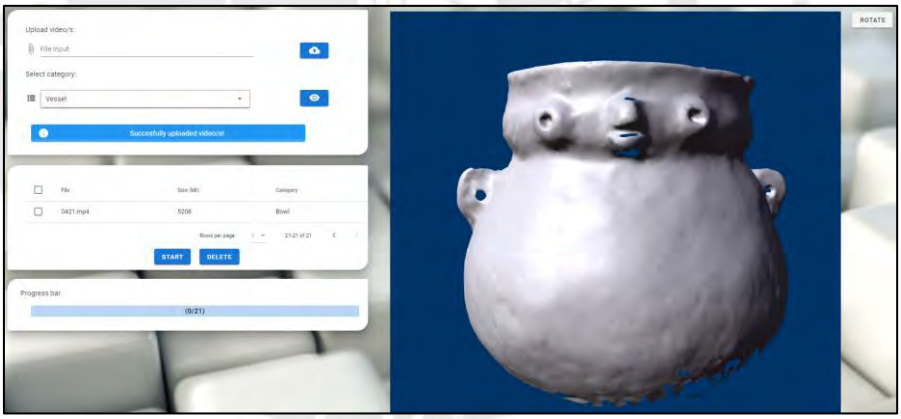


Figura J21. Ejemplo de la categoría "*Vessel*"

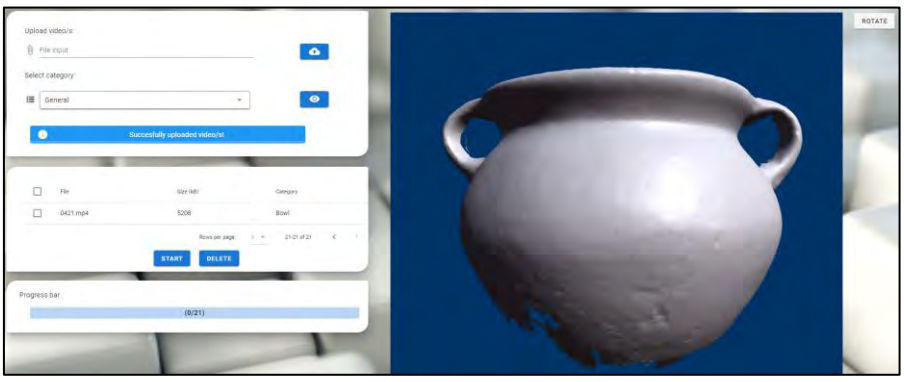


Figura J22. Ejemplo de la categoría “General”

Tabla J3. Resultados de la prueba funcional unitaria 3

Prueba 3. Eliminar un video de la tabla de datos de videos añadidos y empezar el procesamiento de los videos restantes	
Objetivo	Permitir al usuario eliminar o descartar videos que ya hayan sido agregados a la tabla general del segundo componente.
Precondición	• Ubicarse en la vista “*Home*” • Repetir el proceso de la prueba funcional 1 (en caso no se haya ejecutado aún) para contar con múltiples videos ya agregados a la tabla general. • Ubicarse en el segundo componente

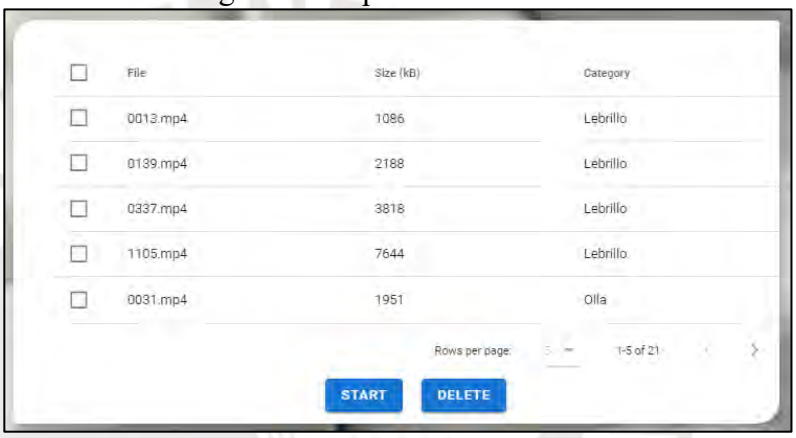


Figura J23. Segundo componente de la primera vista

- Hacer *click* en los “*check box*” de los videos de la tabla que se quieren eliminar

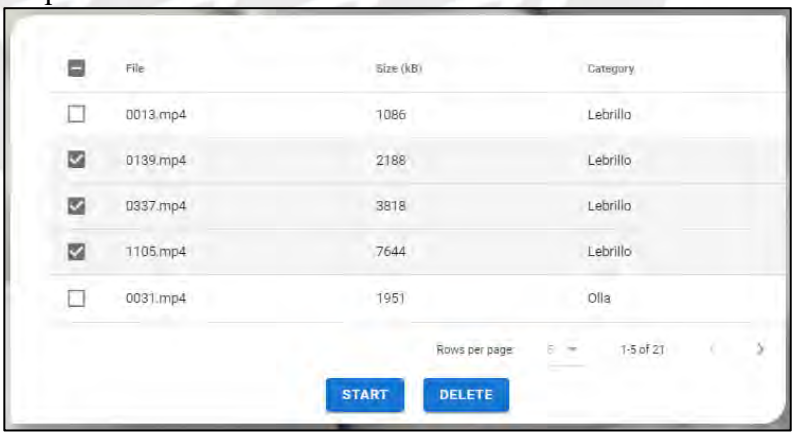


Figura J24. Selección de videos en el segundo componente

- Hacer *click* en el segundo botón azul para eliminar los videos seleccionados.

Figura J25. Segundo botón azul del segundo componente

Descripción (campos)	No es necesario llenar ningún campo.
Resultados obtenidos	Al ejecutar la prueba, los 3 elementos seleccionados desaparecen de la tabla general. Esto quiere decir que ya no serán procesados posteriormente. En la parte inferior derecha de las últimas 2 capturas de pantallas podemos notar el siguiente mensaje "1-5 *of* 21", esto quiere decir que se están mostrando los primeros 5 videos agregados de los 21 existentes. Sin embargo, en la siguiente figura veremos que ahora el mensaje ha cambiado a "1-5 *of* 18" corroborando así la eliminación de los 3 videos seleccionados.

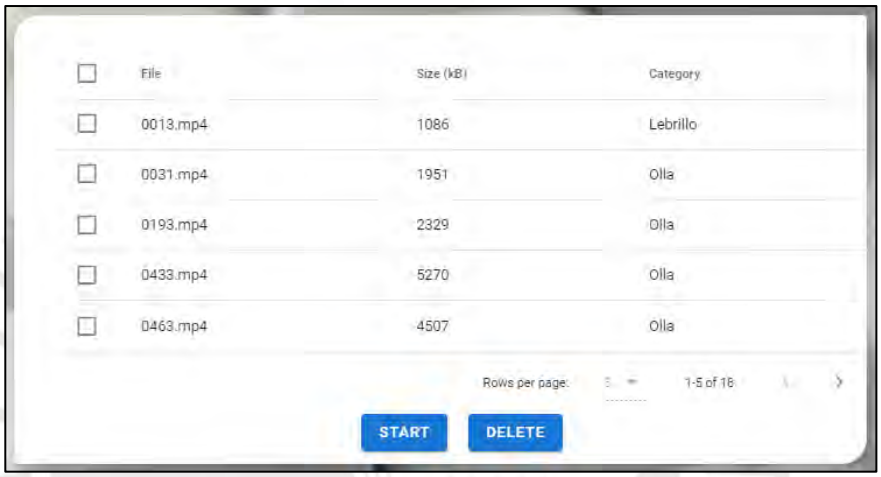


Figura J26. Resultados obtenidos de la tercera prueba funcional

Tabla J4. Resultados de la prueba funcional unitaria 4. Elaboración propia

Prueba 4. Descargar los resultados obtenidos a partir del procesamiento de los videos	
Objetivo	Permitir al usuario descargar los resultados (mallas poligonales en formato PLY) obtenidos a partir de las videograbaciones ingresadas a la interfaz.
Precondición	• Ubicarse en la vista "*Home*" • Repetir el proceso de la prueba funcional 1 (en caso no se haya ejecutado aún) para contar con los 21 videos ya agregados a la tabla general. • Ubicarse en el segundo componente

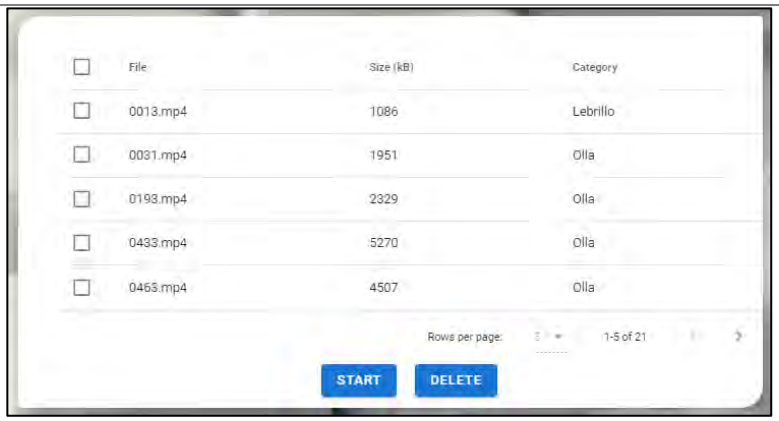


Figura J27. Segundo componente de la primera vista

- Hacer *click* en el primer botón azul para iniciar el procesamiento

Figura J28. Primer botón azul del segundo componente

- Esperar un aproximado de 5 minutos por video, es decir, 1 hora con 45 minutos aproximadamente. Mientras tanto, el tercer componente nos informará del progreso realizado hasta el momento: cuántos videos han sido procesados y un mensaje resultante por cada video procesado. Este mensaje puede informarnos del éxito de la reconstrucción 3D de la pieza arqueológica, o en su defecto, de algún error surgido durante el mismo.

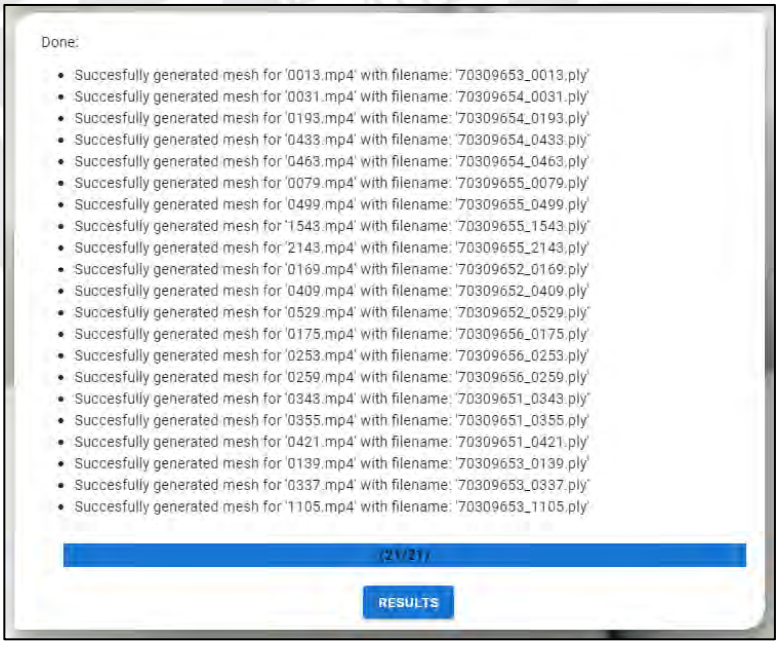


Figura J29. Tercer componente de la primera vista

- Hacer *click* en el único botón azul del tercer componente. Este solo estará disponible una vez finalizado de procesar todos los videos. Este botón nos ubicará en la vista "*Results*".

Figura J30. Botón azul del tercer componente

- Como paso opcional podemos observar los detalles generales del procesamiento en el primer componente de la vista actual.

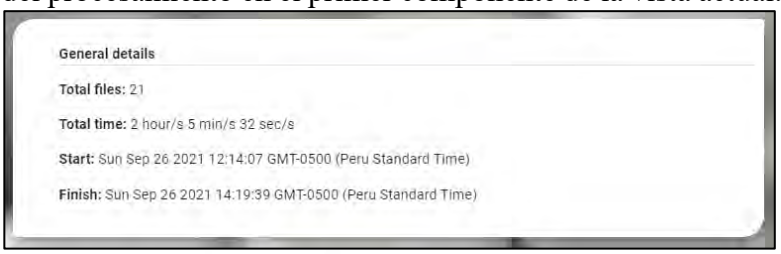


Figura J31. Detalles generales del procesamiento

- En el segundo componente (tabla general de resultados) opcionalmente también podemos seleccionar una malla poligonal y hacer *click* en el primer botón azul para observar el objeto 3D en el visualizador implementado. Esta primera impresión del objeto no es 100% precisa debido a las limitaciones del visualizador, para apreciar correctamente el objeto podemos usar otro software como Blender, empleando los archivos PLY a descargar en el siguiente paso.

Figura J32. Vista de resultados de la interfaz gráfica

- Hacer *click* en el segundo botón azul del segundo componente para descargar todos los resultados.

Figura J33. Botón de descarga de resultados

Descripción (campos)	No es necesario llenar ningún campo.
Resultados obtenidos	Al ejecutar la prueba inmediatamente empieza la descarga de un único archivo ZIP comprimido el cual contiene los 21 archivos PLY de mallas poligonales. En conclusión, luego del procesamiento de

todos los videos es posible descargar los resultados (1 archivo PLY por cada video procesado).

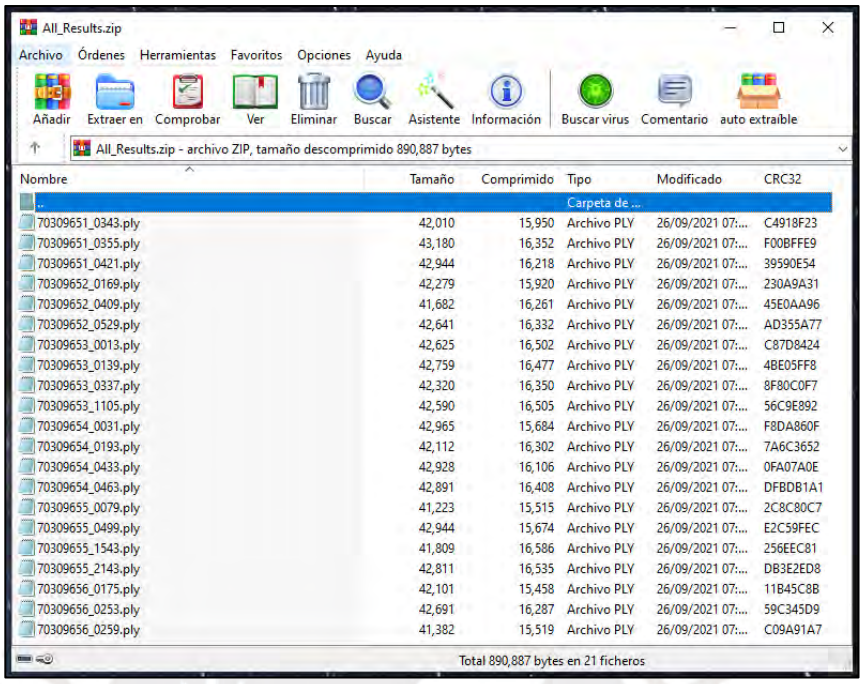


Figura J34. Archivo ZIP que contiene todas las mallas poligonales