**Universidad Peruana de Ciencias Aplicadas**

**UPC**



**CURSO**: Machine Learning - CC57

**CARRERA:** Ciencias de la Computación

**CICLO:** 7

**SECCIÓN:** CC72

**Trabajo**

Profesor: Luis Martin Canaval Sanchez

Integrantes:

* Gonzales de la Cruz, Grober Ericson (U201920609)
* Maguiña Bernuy, Richard José (U202021097)
* Mauricio Tecco, Cristian Alexander (U201922705)

Septiembre 2022

**Índice**

[**Introducción**](#_y7qlzij2fiwb) **3**

[**Formatos utilizados**](#_h7wyygo0je5h) **3**

[Formato OFF](#_42twd2agaf0z) 3

[Formato OBJ](#_1uwxmzdn58i) 4

[**Formato STL**](#_fiwzy5s252r) **5**

[**Conversión de formatos**](#_1pbpzz8g47nf) **6**

[**Funciones para transformación de formatos**](#_jt4h4jx62qlm) **6**

[**Bibliografía**](#_apm7byiob9me) **7**

# Introducción

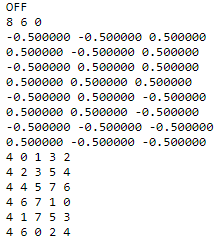
El presente trabajo tiene como finalidad reconocer los formatos de objetos 3D utilizados para entrenar un modelo de machine learning que busca reconstruir objetos con pedazos faltantes en este. El proyecto fue presentado en el paper *3D Reconstruction of Incomplete Archaeological Objects Using a Generative Adversarial Network* por Renato Hermoza e Ivan Sipiran (2018). Asimismo, se busca conocer la estructura de los formatos *STL*, el cual será utilizado para realizar la impresión de los objetos reconstruidos haciendo uso de impresoras 3D.

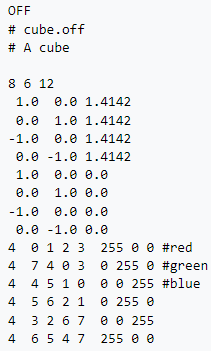
# Formatos utilizados

De acuerdo a Hermoza y Sipiran (2018), para armar su propio dataset de entrenamiento se hizo uso de otros datasets, entre los que encontramos al *ModelNet10* y el *3D Pottery* dataset. En ambos, hemos podido reconocer 2 formatos distintos, el *Object File Format* (OFF) y el *OBJ File Format* (OBJ), respectivamente. A continuación se describirán estos formatos para conocer su estructura interna y entender cómo estos podrían ser utilizados en nuestro proyecto.

## Formato OFF

Es un archivo basado en texto ASCII que describe objetos geométricos en 2D y 3D. Este tipo de formato define las superficies geométricas de los objetos mediante vértices y polígonos. Su contenido es de fácil comprensión, la primera línea siempre contiene la palabra OFF, para luego continuar con una línea que especifica el número de vértices, número de caras y número de bordes totales del objeto. A continuación se nos presentarán *n* líneas con las coordenadas *X*, *Y*, *Z* de cada vértice; *m* líneas para especificar las caras del objeto, las cuales comienzan con el número de vértices que conforman esta cara, seguido de los índices de cada vértices (comenzando desde 0); por último, *p* líneas que representan los bordes, estas comienzan con el número de caras, seguido de los índices de las caras que conforman un borde (comenzando desde 0). Definir bordes en este modelo suele ser opcional, así como agregar los valores RGB para cada una de las caras al lado de su respectiva línea. Los comentarios se redactan haciendo uso del caracter *#*.

Ejemplo de un archivo OFF que representa un cubo 3D:

Ejemplo de un archivo OFF que especifica colores RGB y hace uso de comentarios:

## 

## 

## Formato OBJ

El segundo formato de los modelos es el formato OBJ conocido como Wavefront 3D Object file y desarrollada por Wavefront Technologies. Este archivo permite el almacenamiento de datos importantes de un objeto como la posición de cada vértice, posición UV para texturas, el vector normal de los vértices y las caras del objeto definido con una lista de vértices y texturas.

Aunque los formatos OBJ pueden llegar a ser algo complejos, realizar lectura de un formato simple nos ayudará a comprender su estructura con facilidad. Principalmente, al inicio de cada línea encontraremos algunos caracteres: *v*, *vt*, *vn*, *f*. El caracter *v* representa la definición de la posición de un vértice, *vt* representa una coordenada UV y *vn* declara un vector normal. Con el comando *f* se comienzan a crear las caras del objeto 3D, su formato puede representarse de la siguiente manera: *f v1[/vt1][/vn1] v2[/vt2][/vn2] v3[/vt3][/vn3]*. Cada polígono tiene que ser creado por un mínimo de 3 vértices, a cada uno de estos vértices se le puede especificar el índice de las coordenadas UV y un respectivo vector normal; sin embargo, estos son opcionales. Al igual que el formato OFF, el caracter *#* representa un comentario dentro del archivo.

Ejemplo de un archivo OBJ que representa un cubo 3D:

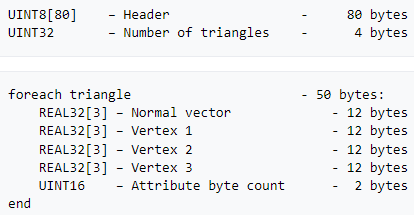
# 

# Formato STL

STL es el formato de archivo comúnmente utilizado para la impresión en 3D. Además tiene varias ventajas sobre los otros formatos existentes, una de ellas es su reconocimiento a nivel mundial, lo que significa su gran desarrollo y distribución en páginas de modelos 3D. Otra de las ventajas es que los archivos STL son pequeños y simples, lo que hace que su procesamiento sea realmente eficiente.

STL (StereoLithography) fue creado por la compañía 3D Systems como formato de un archivo informático CAD (Computer aided design), el cual define la geometría de archivos en 3D, pero excluye la información como el color, la textura o propiedades físicas. Este formato es bastante particular, pues existen versiones en ASCII y definidas en binario, siendo este segundo mucho más simplificado. Sin embargo, requiere de un editor de mallas para poder trabajar con dichos formatos.

El formato STL es fácil de entender, primero encontraremos 80 bytes que representan la cabecera del archivo. Seguido de esto, 4 bytes almacenan un número entero que representa el número de triángulos totales del objeto 3D. A continuación, para cada triángulo se separan 50 bytes distribuidos de la siguiente manera: 12 bytes para el vector normal (3 números flotantes), 36 bytes para los 3 vértices del triángulo (3 números flotantes por cada vértices que representan las coordenadas X/Y/Z) y los 2 últimos bytes que sirve como atributo contador.

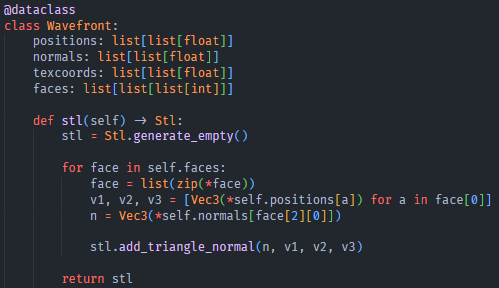
Representación de un formato STL:

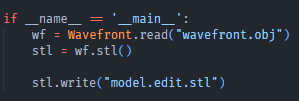
También existen un par de variaciones que incluyen información de los colores RGB del modelo 3D. Sin embargo, estos no son tan reconocidos debido a la poca necesidad de usar color para estos modelos.

# Conversión de formatos

Para decidir qué formato era necesario escoger para realizar su conversión a un archivo STL, nos enfocamos en los dos datasets mencionados en el apartado de formatos utilizados en el paper citado. El primero es el *ModelNet10* dataset, que contiene objetos como camas, sillas, escritorios, mesas, entre otros. Por otro lado, el *3D Pottery* dataset contiene únicamente cerámicas. Ya que el objetivo de este trabajo es entrenar un modelo de machine learning que obtenga los resultados más efectivos, sería conveniente enfocarlo en un solo tipo de objeto. Es por ello que se optó por escoger el formato OBJ (presente en el segundo dataset) como enfoque para la creación de las funciones de transformación de formatos a STL.

# Funciones para transformación de formatos

Las funciones para la transformación de formatos se pueden visualizar accediendo al repositorio del trabajo mediante el siguiente enlace: <https://github.com/ggonzalesd/tf_ml_g3>. El archivo “model\_reader.py” contiene toda la implementación.



# Bibliografía

3D Systems (s.f.). ¿Qué es un archivo .STL? Recuperado de: <https://es.3dsystems.com/quickparts/learning-center/what-is-stl-file> [Consulta: 25 de septiembre de 2022].

FileFormat (s.f.). STL File format. Recuperado de: <https://docs.fileformat.com/cad/stl/> [Consulta: 25 de septiembre de 2022].

FILExt (s.f.). Todo sobre los archivos OFF. Recuperado de: <https://filext.com/es/extension-de-archivo/OFF> [Consulta: 25 de septiembre de 2022].

Hermoza, R. & Sipiran, I. (2018). 3D reconstruction of incomplete archaeological objects using a generative adversarial network. Recuperado de: <https://dl.acm.org/doi/abs/10.1145/3208159.3208173> [Consulta: 24 de septiembre de 2022].

Resycam (2019). Diferencias de los formatos de impresión 3D. STL vs OBJ vs PLY. Recuperado de: <https://www.resycam.com/diferencias-de-los-formatos-de-impresion-3d-stl-vs-obj-vs-ply/> [Consulta: 25 de septiembre de 2022].