# Generación de mallas biomédicas

Nicolas Laverde *, Melissa Robles †, Johan Rodríguez ‡, Leandro Rodríguez §
Universidad de los Andes. Bogotá, Colombia
Email: *n.laverdem@uniandes.edu.co, †mv.robles@uniandes.edu.co, ‡jd.rodriguezp1234@uniandes.edu.co y §l.rodriguez1123@uniandes.edu.co

## I. Introducción

La nueva tendencia en salud son los tratamientos personalizados, dejando atrás los enfoques universales y los tratamientos de "un solo diseño para todos". Según el Servicio de Salud Nacional de Inglaterra, los tratamientos universales presentan una ineficiencia del 70 % en los pacientes, lo que marca la necesidad de tratamientos personalizados [1]. Sin embargo, aplicar un enfoque de tratamientos personalizados resulta inviable con los procesos de fabricación tradicionales, ya que conlleva tareas intensivas en tiempo y esfuerzo, mientras que los productos en 3D han sido bien recibidos por los pacientes [2]. Esta tendencia genera la necesidad de que la industria transite hacia nuevas tecnologías que permitan la personalización.
La impresión 3D fue introducida hace más de tres décadas. Desde entonces, esta tecnología ha transformado radicalmente la industria de la manufactura. El propósito original de esta tecnología era la creación de prototipos de ingeniería, como partes de autos, accesorios de moda e incluso modelos de casas, que eran producidos mediante esta tecnología. De hecho, hoy en día, aún se utiliza para estos fines [3, 4, 5]. Sin embargo, las aplicaciones no se limitan a eso. La impresión 3D ha revolucionado sin duda el sistema de salud, ya que puede producir objetos a medida, lo que la hace perfecta para la personalización de prótesis, implantes, sistemas de entrega de medicamentos, tejidos y dispositivos médicos [6]. En la Figura 1 se pueden visualizar las cinco grandes categorías en las que se emplea la impresión en 3D.

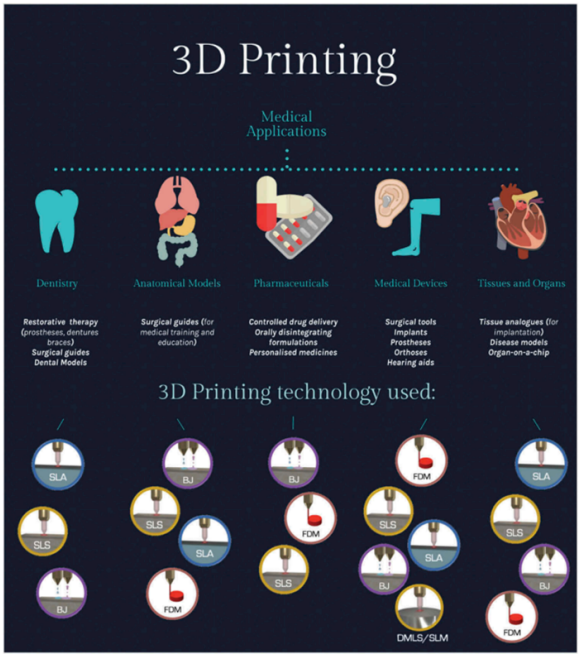


Figura 1: Aplicaciones actuales de la impresión 3D en medicina y atención médica.

La impresión 3D aborda una de las principales problemáticas en el ámbito de la salud: la escasez de órganos para trasplantes. Según las estadísticas sobre donación de órganos, aproximadamente 20 pacientes fallecen diariamente mientras esperan la oportunidad de recibir un trasplante [7]. Además de la limitada disponibilidad de donantes de órganos, otro de los desafíos más significativos es garantizar la biocompatibilidad del órgano donado [8]. La impresión en 3D tiene la capacidad de generar tejidos con dimensiones y materiales específicos para cada paciente, lo que puede contribuir significativamente a los trasplantes de órganos [9].
Los avances en la impresión en 3D para tratamientos personalizados han sido significativos. Las estructuras pueden generarse a partir de un archivo digital en 3D utilizando software de *diseño asistido por computadora* (CAD) y técnicas de visión por computadora, que incluyen resonancias magnéticas o tomografías en 3D [10]. Aunque los resultados son muy positivos, es importante destacar que la técnica de visión por computadora tiene un límite de personalización determinado por las imágenes obtenidas.
El objetivo de este proyecto es desarrollar un asistente dedicado a la creación de modelos anatómicos y órganos, condicionado por texto o imagen. Esto se hace con el fin de facilitar la creación de prototipos de investigación de tejidos y explorar la viabilidad de su utilización en trasplantes. Adicionalmente, se contempla su aplicación como una herramienta educativa, beneficiando a estudiantes en el aprendizaje y comprensión de la anatomía y la medicina. Siguiendo con la idea de personalización en los tratamientos, buscamos incorporar información específica del paciente para el diseño de estas representaciones en 3D. Lograremos esto mediante un *fine-tuning* de modelos generativos basados en modelos de difusión con mallas biomédicas. Aunque se presentarán estrategias que generan mallas en 3D condicionadas más adelante, es relevante destacar que ninguna de ellas se enfoca específicamente en el dominio biomédico.

## II. Antecedentes

Determinar la metodología de representación de objetos en tres dimensiones constituye un desafío por la complejidad inherente a las formas, las variaciones en la iluminación y las dimensiones espaciales. Diversas estrategias son empleadas para abordar distintos aspectos de la representación tridimensional, tales como el uso de mallas poligonales, nubes de puntos y aproximaciones basadas en funciones implícitas. A modo de ejemplos, se encuentran los formatos STL, XYZ y NeRF, los cuales se corresponden, respectivamente, con cada una de las técnicas anteriormente expuestas. En esta sección se describen las distintas representaciones necesarias para entender los artículos a analizar.

### *II-A. STL*

La representación tridimensional de un objeto mediante un archivo STL implica una descripción geométrica mediante mallas

poligonales. Este formato de archivo almacena la superficie del objeto mediante una red de triángulos, donde cada vértice define un punto en el espacio tridimensional. Una representación STL, por lo tanto, es una matriz de dimensiones $(T, 3, 3)$, en donde $T$ representa el número de triángulos representados y cada triángulo se representa por una matriz de dimensiones $3 \times 3$ de la forma

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}.$$

También se encuentran los componentes del vector normal unitario al triángulo, asegurándose que apunte hacia afuera de acuerdo con el modelo. Esta representación permite la transferencia de modelos 3D entre diferentes programas y sistemas, facilitando su utilización en aplicaciones de diseño, ingeniería y fabricación.

### *II-B. NeRF*

La representación 3D de NeRF se propuso en 2022 [11] como una alternativa a las representaciones clásicas utilizadas hasta el momento. Esta se basa en la construcción de una función implícita que mapea tuplas $(\mathbf{x}, \mathbf{d})$ de coordenadas y direcciones a tuplas $(\mathbf{c}, \sigma)$ de colores y densidades. Las coordenadas de entrada, denotadas por $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$, se representan como puntos, mientras que las direcciones se expresan mediante dos ángulos, $(\theta, \phi) \in \mathbb{R}^2$, que representan el punto de vista desde el cual se observa el objeto. Por otro lado, los colores siguen la representación RGB $(c_R, c_G, c_B)$, y las densidades son números reales que simbolizan la opacidad o la *cantidad de materia* en una determinada ubicación dentro del volumen tridimensional de la escena. En resumen, la función se puede caracterizar como

$$\begin{aligned} F_\Theta : \mathbb{R}^5 &\to \mathbb{R}^4 \\ (\mathbf{x}, \mathbf{d}) &\mapsto (\mathbf{c}, \sigma). \end{aligned}$$

La forma en la que se parametriza la función es por medio de una red neuronal clásica Multi Layer Perceptron (MLP) con pesos $\Theta$, lo que le da el nombre de *función implícita* a $F_\Theta$. A partir de esta, dada una dirección, es posible construir una imagen asignando a cada uno de los píxeles el color obtenido por la red neuronal.

### *II-C. STF*

La representación STF (Signed Distance Functions and Texture Fields) se refiere a la formulación de una función implícita como método de representación tridimensional, caracterizada por generar como salidas distancias con signo o colores con textura. Las representaciones SDF (Signed Distance Functions) se definen a partir de una función

$$\begin{aligned} G_\Theta : \mathbb{R}^3 &\to \mathbb{R} \\ (x, y, z) &\mapsto d. \end{aligned}$$

donde $|d|$ representa la distancia del punto $(x, y, z)$ al punto más cercano de la superficie del objeto, y el signo $\text{sign}(d)$ es negativo si el punto está dentro del objeto y positivo si se encuentra fuera del mismo. Esta función es considerada *implícita*, dado que la superficie del objeto está definido como las coordenadas $(x, y, z)$ que satisfacen $G_\Theta(x, y, z) = 0$. Esta representación SDF ha sido utilizada en modelos generativos como en DMTet [12]. Otras aproximaciones sugieren el uso de información adicional a la distancia con signo, agregando información de textura (Texture Fields) y color en formato RGB.

## III. Trabajo Relacionado

### *III-A. Point-E*

Al momento de publicar Point-E [13] en el año 2022, todos los modelos del estado del arte requerían un alto tiempo de inferencia en múltiples GPU para producir solo un ejemplo, lo cual hacía imposible democratizar la generación de contenido 3D a partir de texto al público. Por ello, en este artículo se plantea una alternativa que genera una nube de puntos 3D a partir de texto. Para ello, se define una nube de puntos como un arreglo de tuplas, donde cada tupla representa un punto que tiene 3 entradas para las coordenadas tridimensionales y 3 otras tres para los canales de color RGB. Por ejemplo, una nube de puntos se puede visualizar en la figura 2.

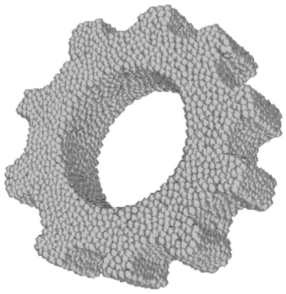

Figura 2: Visualización de nube de puntos para un engranaje extraída de [13]

Para lograr generar nubes de puntos, como la presentada anteriormente, se tiene una arquitectura basada en transformers [14] y modelos de difusión Gaussiana [15], los cuales parten de un proceso de adición de ruido por pasos definido como

$$q(x_t|x_{t-1}) : \mathcal{N}(x_t; \sqrt{1-\beta}x_{t-1}, \beta_t\mathbf{I})$$

Donde $x_{t-1}$ y $x_t$ son dos pasos consecutivos de adición de ruido a un ejemplo $x$, que en este caso es una nube de puntos, y el proceso generativo está modelado con una distribución normal condicionada al ejemplo con adición de ruido en una etapa anterior $x_{t-1}$, y una variable de programación de ruido $\beta$ que se encarga de ajustar el ruido gradualmente en cada paso, limitando la varianza de este para evitar que haya saltos muy grandes de ruido entre pasos. Este proceso de adición de ruido se aproxima con una arquitectura transformer [14], que se encarga de modelar la probabilidad condicional inversa $p\theta(x_{t-1}|x_t)$, por medio de la predicción del ruido $\epsilon$.

Si bien, este proceso es ideal para generar nubes de puntos con significado a partir de nubes de puntos creadas por ruido distribuido normalmente, es necesario hacer modificaciones para poder generarlas con base a un texto dado. Por lo tanto, generalmente se tiene un proceso de condicionamiento, en que se concatena la nube de puntos creada por ruido y una representación textual como entrada de la arquitectura transformer para poder utilizar texto como entrada en tiempo de inferencia [16]. No obstante, en la experimentación para crear Point-E [13] se encontró de forma empírica que se obtenía un mejor resultado condicionando sobre imágenes en vez de texto, por lo que se tiene un arquitectura de difusión basada en transformers que parte de una representación de imagen dada por un modelo ViT-L/14 CLIP [17] y una nube de puntos ruidosa para predecir una nube de puntos con sentido que represente lo que está en la imagen, como se puede ver en la figura 3.

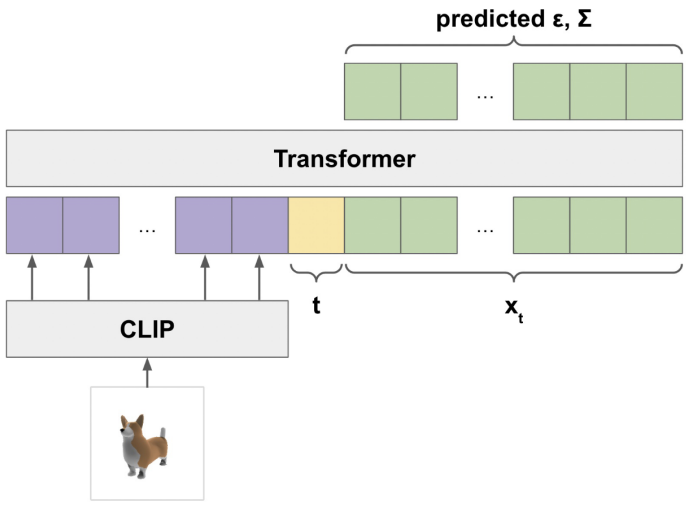


Figura 3: Arquitectura de difusión basada en transformers para generar una nube de puntos con sentido a partir de una imagen y una nube de puntos creada con ruido distribuido normalmente [13]

Como se observa, esta arquitectura solamente permite generar nubes de puntos a partir de imágenes, por lo que es necesario introducir un componente que genere imágenes a partir de texto. Este componente es un modelo GLIDE [18], que consiste en un modelo de difusión condicionado a texto, que a diferencia de CLIP [17], solo necesita texto sin etiqueta de clasificación para generar imágenes. Para poder usar GLIDE de manera óptima se hizo fine-tuning con un dataset propio compuesto de millones de imágenes que se crearon a partir de renders 3D con Blender [19], creando 20 imágenes por modelo, cada una dada por un ángulo de cámara aleatorio. Ya teniendo un componente de texto a imagen, se presenta un arquitectura completa de texto a nube de puntos, como la presente en la figura 4.

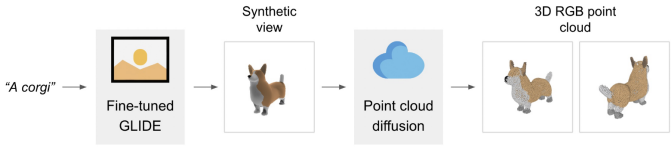


Figura 4: Arquitectura de texto a nube de puntos de Point-E

Esta arquitectura se divide en tres componentes generativos consecutivos:

1. **Generador de texto a imagen**: Se usa una arquitectura GLIDE para convertir de texto a imagen.
2. **Generador de nube de puntos de baja resolución**: Se usa una arquitectura transformer como la presente en la figura 3 para crear una nube de puntos de baja resolución de 1.000 puntos a partir de la imagen generada en el paso anterior.
3. **Generador de nube de puntos de alta resolución**: Al igual que en el paso anterior, se usa una arquitectura transformer como la de la figura 3, que está condicionada a la imagen generada en el paso 1. No obstante, esta arquitectura también se condiciona a la nube de puntos generada en el paso 2, por lo que parte de los puntos de la nube de puntos usados en la entrada se reemplazan por la nube de puntos creada en el paso 2 para generar una nube de puntos de alta resolución de 4.000 puntos.

En la arquitectura descrita anteriormente, se tiene el detalle de entrenamiento del primer paso por medio de tuplas de texto y vistas sintéticas creadas a partir de un dataset de objetos 3D. Para completar el entrenamiento del flujo completo, se crean nubes de puntos de muy alta resolución a partir de las 20 vistas que se tienen para cada objeto y se hace un muestreo para obtener nubes de 4.000 puntos y 1.000 puntos respectivamente. Las primeras nubes se usan para entrenar el componente del paso 3 y las segundas para entrenar el componente del paso 2, luego de aplicar el proceso de adición de ruido descrito en [15].

Para evaluar el modelo se usa la métrica CLIP R-Precision [20], que mide la calidad de generación de texto a imagen u objeto 3D por medio de un modelo CLIP (ViT-B/32, ViT-L/14), y se encuentra que el rendimiento es inferior al de los modelos del estado del arte, que tienen un valor por encima de 67, ya que el valor de CLIP R-Precision máximo obtenido con Point-E es de 46. **Sin embargo, la latencia llega a ser entre 10 y 100 veces menor a la de los modelos del estado del arte, con un valor de 1.5 minutos**.

### *III-B. Shap-E*

Shap-E [21] se configura como un modelo generativo condicional que, a partir de textos o imágenes, genera una representación tridimensional. Este desarrollo, concebido por OpenAI, se distingue de Point-E al generar directamente los parámetros de una función implícita, que puede emplearse como representación de un objeto tridimensional, como se requiere en las representaciones NeRF (II-B) y STF (II-C). La estructura del modelo consta de dos componentes principales: un *encoder* que produce los parámetros de una función implícita a partir de una representación de un objeto tridimensional y un modelo de difusión condicional ya sea a imágenes o descripciones de texto.

La arquitectura del encoder se refleja en la Figura 5. Esta parte del modelo toma como entrada dos representaciones distintas de un objeto tridimensional: una representación en forma de nube compuesta por 16,000 puntos y una serie de vistas en formato RGBA del objeto desde diversos ángulos. Estas entradas son sometidas a capas de atención cruzada, seguidas por una arquitectura de transformer que genera una secuencia de representaciones de los inputs. Estas representaciones son utilizadas como parámetros en las funciones implícitas que finalmente representan el objeto tridimensional. Específicamente, la salida del encoder puede concebirse como los pesos de una red neuronal MLP como la utilizada en NeRF, en la cual, a partir de una entrada $(x, y, z)$, se generan tres salidas coherentes con las representaciones de funciones implícitas:

- $\sigma$: Representa la densidad en la representación NeRF.
- $RGB$: Representa el color.
- $SDF$: Representa la distancia con signo, característica de la representación STF.

Inicialmente, el encoder se entrena únicamente para obtener representaciones NeRF. Para esto, se utiliza una mezcla de dos funciones de pérdida, utilizando una función conjunta

$$\mathcal{L}_{NeRF} = \mathcal{L}_{RGB} + L_T.$$

La función de pérdida $\mathcal{L}_{RGB}$ compara los colores de los objetos 3D reales y los colores obtenidos a partir de la función implícita generada por el encoder. Más formalmente, se obtienen aleatoriamente un conjunto $R$ de 4.096 rayos (direcciones) del objeto 3D y se comparan los colores estimados de cada rayo con la función implícita y el color real utilizando la norma $L_1$ de la

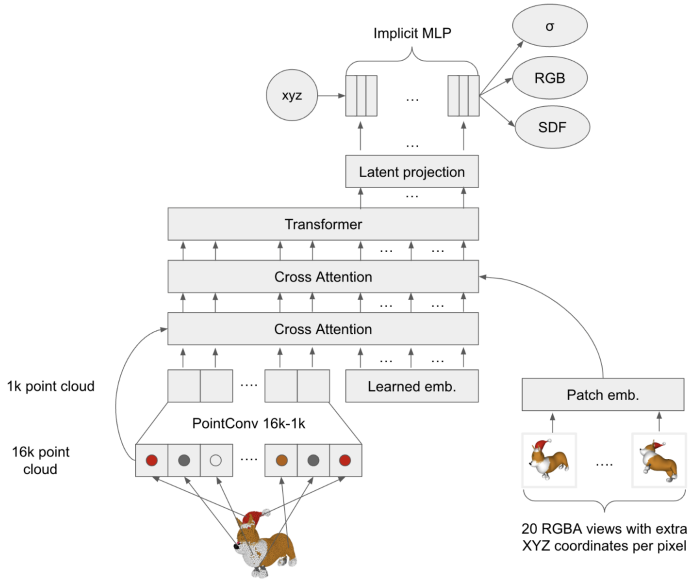


Figura 5: Representación del encoder de Shap-E. Obtenido en [21].

diferencia. Por otro lado, la función de pérdida $L_T$ compara la "transmitancia" [1] real y la compara con la generada a partir de la función implícita. Al igual que para $\mathcal{L}_{RGB}$, se compara a partir de la norma $L_1$ de la diferencia entre los valores reales y los estimados en el mismo conjunto de rayos.

Posteriormente, se emplea la red preentrenada para llevar a cabo el Fine-Tuning considerando la representación STF. Con este propósito, se define una función de pérdida conjunta dada por

$$\mathcal{L}_{FT} = \mathcal{L}_{NeRF} + \mathcal{L}_{STF},$$

donde $\mathcal{L}_{STF}$ compara la malla real con la malla reconstruida. Para la construcción de la malla a partir de las salidas del encoder, se define una grilla de dimensiones $128 \times 128 \times 128$, y se calculan los valores asociados a la función en dicha grilla. Con esta información, se generan múltiples imágenes mediante el proceso de renderizado, las cuales se comparan utilizando la norma $L_2$ con las reales para obtener el valor final de la función de pérdida $\mathcal{L}_{STF}$.

La segunda componente del modelo es un modelo de difusión condicional basado en transformers, al igual que en Point-E (III-A). A diferencia de este último, la entrada no es una nube de puntos, sino una secuencia de vectores latentes que representan los pesos de una función implícita. Esta secuencia tiene dimensiones $1024 \times 1024$, lo que la integra en la arquitectura como una secuencia de 1024 tokens. Para la parte condicional del modelo de difusión, se adopta la misma aproximación que en Point-E (III-A), utilizando embeddings de CLIP tanto para textos como para imágenes.

Los resultados del artículo se presentan mediante métricas que evalúan la eficiencia del encoder, tales como las métricas PSNR y la R-Precision de CLIP, acompañadas de un análisis cualitativo de las muestras generadas. El modelo exhibe mejoras significativas en comparación con Point-E en la tarea condicional basada en textos, mientras que en la tarea condicional basada en imágenes muestra resultados comparables. Esto se puede evidenciar en los ejemplos proporcionados en el artículo, comparando las salidas de ambos modelos (Figura 6). Por último, Shap-E demuestra una inferencia más rápida que Point-E al no requerir un modelo adicional de difusión para el upsampling, lo que representa una ventaja en comparación con el modelo previamente analizado.

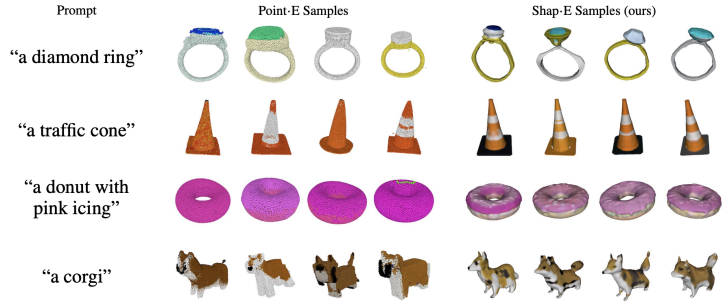


Figura 6: Comparación de resultados Shap-E vs Point-E. Imagen obtenida de [21].

### *III-C. Modelo de Reconstrucción de Gran Escala (LRM)*

El Modelo de Reconstrucción de Gran Escala [22] (LRM, por sus siglas en inglés) introduce un enfoque basado en datos para la reconstrucción 3D a partir de una única imagen, utilizando una arquitectura de codificador-decodificador basada en Transformers [14] de gran envergadura (ver figura 7). Este modelo se compone de tres componentes principales:

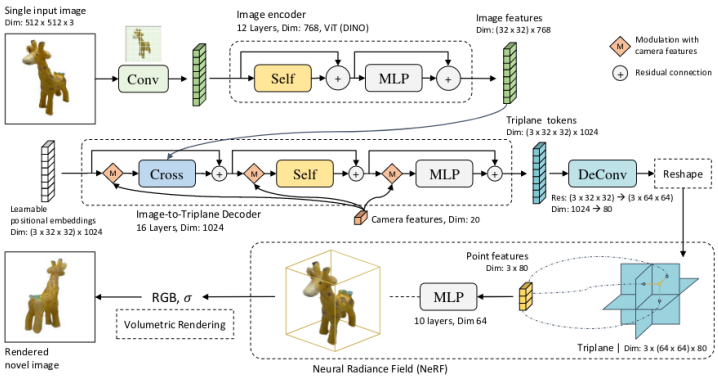


Figura 7: Arquitectura LRM

- **Codificador de Imagen:** LRM utiliza inicialmente un Vision Transformer (ViT) [23], específicamente DINO [24], para codificar la imagen en tokens de características por parche. DINO es escogido por su capacidad para aprender de manera autodidacta sobre la estructura y textura del contenido relevante en las imágenes, lo cual es crucial para que LRM reconstruya la geometría y el color en el espacio 3D.
- **Decodificador de Imagen a Triplano:** Se implementa un decodificador para proyectar características de la imagen y de la cámara en embeddings espaciales-posicionales aprendibles, traduciéndolos a representaciones triplanas. Este decodificador actúa como una red previa que proporciona información geométrica y de apariencia necesaria, compensando las ambigüedades inherentes a la reconstrucción a partir de una única imagen.
- **Representación 3D:** Se adopta la representación triplana como una característica compacta y expresiva. Cada uno de los planos, $(64 \times 64) \times d_T$, se usa para proyectar cualquier punto

[1] Transmitancia: propiedad que describe cómo la luz viaja a través de un medio u objeto. Representa la fracción de luz que atraviesa un punto a lo largo de un rayo sin ser absorbida ni dispersada. Se calcula a partir de las densidades $\sigma$.

3D dentro del cuadro delimitador del objeto NeRF y consultar características de puntos mediante interpolación bilineal, que luego se decodifican en color y densidad usando una MLP.

A continuación del proceso de codificación y decodificación inicial, el LRM integra de manera innovadora las características de la cámara de la imagen de entrada. Para esto, se genera una característica de cámara $c \in R^{20}$, la cual se obtiene aplanando la matriz de extrínsecos de la cámara y concatenándola con la longitud focal y el punto principal. Esta información se normaliza y se transforma en un embedding de alta dimensión mediante un MLP, ajustándose precisamente a las necesidades del modelo para una reconstrucción 3D fidedigna.

En cuanto a la arquitectura de red, LRM se beneficia del modelo ViT-B/16 de DINO preentrenado como codificador de imágenes, junto con un decodificador de imagen a triplano y un $MLP^{nerf}$ dedicado a la representación NeRF. Esta configuración está especialmente diseñada para manejar las especificidades de las dimensiones de entrada y salida, promoviendo un entrenamiento y una inferencia altamente eficientes.

El entrenamiento del LRM es una fase crítica, donde el objetivo principal es generar la forma 3D a partir de la imagen inicial, valiéndose de vistas adicionales como referencia. Para ello, se emplean objetivos de reconstrucción de imagen simples, optimizando una combinación de las pérdidas MSE y LPIPS [25], lo que facilita la supervisión entre las vistas renderizadas y las vistas reales, asegurando una reconstrucción precisa y de alta calidad.

$$\mathcal{L}_{recon}(x) = \frac{1}{V}\sum_{v=1}^{V}(\mathcal{L}_{MSE}(\hat{x_v}, x_v^{GT}) + \lambda\mathcal{L}_{LPIPS}(\hat{x_v}, x_v^{GT}))$$

Durante la fase de inferencia, LRM procesa cualquier imagen de entrada bajo la premisa de parámetros de cámara estandarizados empleados durante el entrenamiento. Utilizando la representación triplana-NeRF reconstruida, el modelo consulta un alto número de puntos para luego extraer la malla mediante el algoritmo de Marching Cubes [26]. Este enfoque no solo demuestra la flexibilidad del modelo para manejar diversas entradas sino también su eficiencia al generar reconstrucciones 3D detalladas en cuestión de segundos.

El entrenamiento del modelo, realizado sobre un conjunto de datos extenso y diverso, incluyendo contenido tanto sintético de Objaverse [27] como capturas reales de MVImgNet [28], junto con una serie de ajustes en los hiperparámetros y técnicas de optimización, asegura que LRM establezca nuevos estándares en la reconstrucción 3D precisa. Este meticuloso proceso desde la codificación inicial con DINO, pasando por la innovadora incorporación de características de cámara, hasta el detallado entrenamiento y la eficiente fase de inferencia, subraya el enfoque holístico y avanzado del LRM hacia la reconstrucción 3D a partir de imágenes 2D.

Finalmente, para abordar el problema Text-to-3D se tendría que utilizar un modelo previo al LRM para generar la imagen de entrada, como por ejemplo Stable Diffusion [29].

## IV. Propuesta

En la sección III se mencionaron tres arquitecturas candidatas a ser usadas en el presente problema biomédico. Estas tres arquitecturas se basan en codificación de texto/imagen a una representación 3D y poseen repositorios de código abierto en GitHub, lo cual permite explorar su funcionamiento al detalle. Teniendo esto en cuenta, se propone un proceso experimental compuesto de los siguientes pasos:

1. **Selección de modelo fine-tuneable**: Se exploran los repositorios en búsqueda de entender la capacidad de fine-tuning de los modelos, así como la facilidad de transformar el dataset actual de mallas (mesh .stl) al formato recibido por estos.
2. **Pipeline de transformación de datos**: Una vez se elija el modelo que permita hacer lo descrito anteriormente, se programa un pipeline que permita transformar de formato .stl al formato requerido por el modelo. Por ejemplo, para Point-E es necesario convertir de .stl a .RARGB para poder utilizar los scripts de generación de vistas con GLIDE.
3. **Generación de baseline**: Se produce inferencia de los datos transformados con el Pipeline realizado para establecer métricas iniciales de desempeño, como CLIP R-Precision, con el modelo elegido.
4. **Fine-tuning de modelo**: Se hace fine-tuning del modelo elegido para que aprenda sobre los datos biomédicos presentes, los cuales deben estar transformados con el Pipeline de transformación de datos.
5. **Búsqueda de hiperparámetros (opcional)**: Si se observa que los resultados no fueron los deseados, y hay un tiempo aceptable, se hace la búsqueda de hiperpárametros del modelo elegido para obtener resultados de mayor calidad.

## V. Dataset

Para llevar a cabo la fase de fine-tuning del modelo, utilizaremos el dataset *MedShapeNet* como fuente de mallas biomédicas [30]. Este dataset en realidad agrupa 23 datasets distintos, que suman más de 100,000 mallas con sus respectivas anotaciones (ver Figura 8). A diferencia de datasets existentes, como ShapeNet, que se componen de modelos tridimensionales de diseño asistido por computadora (CAD) de objetos del mundo real (como aviones, automóviles, sillas y escritorios), MedShapeNet proporciona formas tridimensionales extraídas de datos de imágenes de pacientes reales, abarcando tanto a sujetos saludables como patológicos. Los autores de MedShapeNet crearon dos modos de interacción para utilizar este dataset: Una interfaz basada en internet que brinda acceso a los datos originales de formas en alta resolución, y una API de Python que permite a los usuarios interactuar con los datos de formas a través de Python.

Las figuras en 3D se almacenan en los formatos estándar para estructuras de datos geométricas, es decir, NIfTI (.nii) para cuadrículas de vóxeles, estereolitografía (.stl) para mallas y formato de archivo poligonal (.ply) para nubes de puntos, lo que facilita una vista previa rápida de las formas a través de softwares existentes.

## Referencias

[1] NHS. “IMPROVING OUTCOMES THROUGH PERSONALISED MEDICINE”. En: (ago. de 2016), págs. 1-18. URL: https://www.england.nhs.uk/wp-content/uploads/2016/09/improving-outcomes-personalised-medicine.pdf.

[2] Alvaro Goyanes et al. “Patient acceptability of 3D printed medicines”. En: *International Journal of Pharmaceutics* 530.1 (2017), págs. 71-78. ISSN: 0378-5173. DOI: https://doi.org/10.1016/j.ijpharm.2017.07.064. URL: https://www.sciencedirect.com/science/article/pii/S0378517317306725.

[3] C. Barnatt. *3D Printing: The Next Industrial Revolution*. ExplainingTheFuture.com, 2013. ISBN: 9781484181768. URL: https://books.google.com.co/books?id=Tvb6nAEACAAJ.

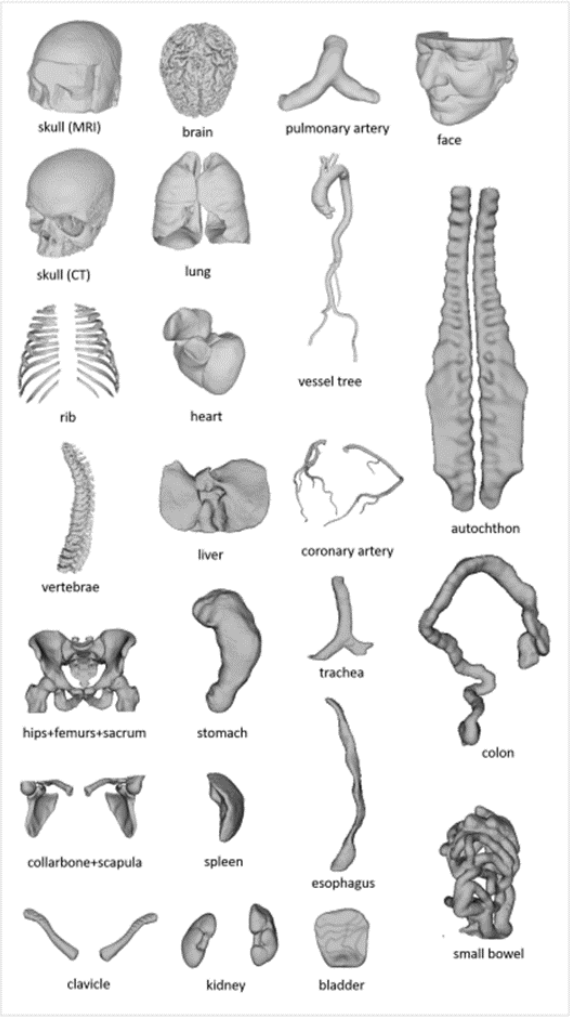


Figura 8: Mallas de ejemplo en MedShapeNet, incluidos varios huesos (cráneos, costillas y vértebras), órganos (cerebro, pulmón, corazón, hígado), vasos (árbol de vasos aórticos y arteria pulmonar) y músculos.


[4] A. Chowdhry. *What can 3D printing do? Here are 6 creative examples*. Accessed: 2024-03-08. 2013. URL: https://www.forbes.com/sites/amitchowdhry/2013/10/08/what-can-3d-printing-do-here-are-6-creative-examples/?sh=6901d5e15491.

[5] C. Lee Ventola. "Medical Applications for 3D Printing: Current and Projected Uses". En: *P & T : a peer-reviewed journal for formulary management* 39.10 (oct. de 2014), págs. 704-11. ISSN: 1052-1372. DOI: 10.4069/pj.2014.4910704. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4189697/.

[6] Cecilia Sahlgren et al. "Tailored Approaches in Drug Development and Diagnostics: From Molecular Design to Biological Model Systems". En: *Adv Healthc Mater* 6.21 (nov. de 2017), 10.1002/adhm.201700258. ISSN: 2192-2659 (Electronic), 2192-2640 (Linking). DOI: 10.1002/adhm.201700258. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/adhm.201700258.

[7] S. V. Shinde et al., eds. *Disruptive Developments in Biomedical Applications*. 1st. CRC Press, 2022. DOI: 10.1201/9781003272694. URL: https://doi.org/10.1201/9781003272694.

[8] Karla Jiménez Oliver. "Overview of Organ Donation". En: *Mexican Journal of Medical Research ICSA* 11.21 (ene. de 2023), págs. 55-63. DOI: 10.29057/mjmr.v11i21.10007. URL: https://repository.uaeh.edu.mx/revistas/index.php/MJMR/article/view/10007.

[9] XuanQi Zheng et al. "3D Bioprinting in Orthopedics Translational Research". En: *Journal of Biomaterials Science, Polymer Edition* 30.13 (2019). PMID: 31124402, págs. 1172-1187. DOI: 10.1080/09205063.2019.1623989. eprint: https://doi.org/10.1080/09205063.2019.1623989. URL: https://doi.org/10.1080/09205063.2019.1623989.

[10] Sarah J Trenfield et al. "3D Printing Pharmaceuticals: Drug Development to Frontline Care". En: *Trends in Pharmacological Sciences* 39.5 (mayo de 2018), págs. 440-451. ISSN: 1873-3735 (Electronic), 0165-6147 (Linking). DOI: 10.1016/j.tips.2018.02.006. URL: https://www.sciencedirect.com/science/article/pii/S0165614718300440.

[11] Ben Mildenhall et al. "NeRF". En: *Communications of the ACM* 65 (1 ene. de 2022), págs. 99-106. ISSN: 15577317. DOI: 10.1145/3503250.

[12] Tianchang Shen et al. *Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis*. URL: https://nv-tlabs.github.io/DMTet/..

[13] Alex Nichol et al. "Point-E: A System for Generating 3D Point Clouds from Complex Prompts". En: (dic. de 2022). URL: http://arxiv.org/abs/2212.08751.

[14] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Ago. de 2023. DOI: 10.48550/arXiv.1706.03762. URL: http://arxiv.org/abs/1706.03762 (visitado 09-03-2024).

[15] Jonathan Ho, Ajay Jain y Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[16] Aaron van den Oord, Oriol Vinyals y Koray Kavukcuoglu. *Neural Discrete Representation Learning*. 2018. arXiv: 1711.00937 [cs.LG].

[17] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[18] Alex Nichol et al. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*. 2022. arXiv: 2112.10741 [cs.CV].

[19] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[20] Dong Huk Park et al. "Benchmark for Compositional Text-to-Image Synthesis". En: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 2021. URL: https://openreview.net/forum?id=bKBhQhPeKaF.

[21] Heewoo Jun y Alex Nichol. "Shap-E: Generating Conditional 3D Implicit Functions". En: (mayo de 2023). URL: http://arxiv.org/abs/2305.02463.

[22] Yicong Hong et al. *LRM: Large Reconstruction Model for Single Image to 3D*. arXiv:2311.04400 [cs]. Nov. de 2023. DOI: 10.48550/arXiv.2311.04400. URL: http://arxiv.org/abs/2311.04400 (visitado 08-03-2024).

[23] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929 [cs] version: 2. Jun. de 2021. DOI: 10.48550/arXiv.2010.11929. URL: http://arxiv.org/abs/2010.11929 (visitado 09-03-2024).

[24] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers*. arXiv:2104.14294 [cs]. Mayo de 2021. DOI: 10.48550/arXiv.2104.14294. URL: http://arxiv.org/abs/2104.14294 (visitado 10-03-2024).

[25] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. arXiv:1801.03924 [cs]. Abr. de 2018. DOI: 10.48550/arXiv.1801.03924. URL: http://arxiv.org/abs/1801.03924 (visitado 10-03-2024).

[26] William E. Lorensen y Harvey E. Cline. "Marching cubes: a high resolution 3D surface construction algorithm". En: *Seminal graphics: pioneering efforts that shaped the field, Volume 1*. Vol. Volume 1. New York, NY, USA: Association for Computing Machinery, jul. de 1998, págs. 347-353. ISBN: 978-1-58113-052-2. URL: https://doi.org/10.1145/280811.281026 (visitado 10-03-2024).

[27] Matt Deitke et al. *Objaverse-XL: A Universe of 10M+ 3D Objects*. arXiv:2307.05663 [cs]. Jul. de 2023. DOI: 10.48550/arXiv.2307.05663. URL: http://arxiv.org/abs/2307.05663 (visitado 10-03-2024).

[28] Xianggang Yu et al. *MVImgNet: A Large-scale Dataset of Multi-view Images*. arXiv:2303.06042 [cs]. Mar. de 2023. DOI: 10.48550/arXiv.2303.06042. URL: http://arxiv.org/abs/2303.06042 (visitado 10-03-2024).

[29] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. arXiv:2112.10752 [cs]. Abr. de 2022. DOI: 10.48550/arXiv.2112.10752. URL: http://arxiv.org/abs/2112.10752 (visitado 10-03-2024).

[30] Jianning Li et al. *MedShapeNet – A Large-Scale Dataset of 3D Medical Shapes for Computer Vision*. 2023. arXiv: 2308.16139 [cs.CV].