Reconstrucción de superficies mediante Procesos Gaussianos

Manuel Guillermo López Buenfil[1]

[1]Instituto de Ingeniería y Tecnología
Universidad Autónoma de Ciudad Juárez

Mayo 15, 2014

Definición del problema

Entrada Un conjunto finito de puntos P en $\mathbb{R}^3$ muestreados de una superficie S

Salida Una función implícita $F : \mathbb{R}^3 \to \mathbb{R}$ tal que la isosuperficie de nivel cero, $F^{-1}(0)$, aproxime a S.

Definición del problema

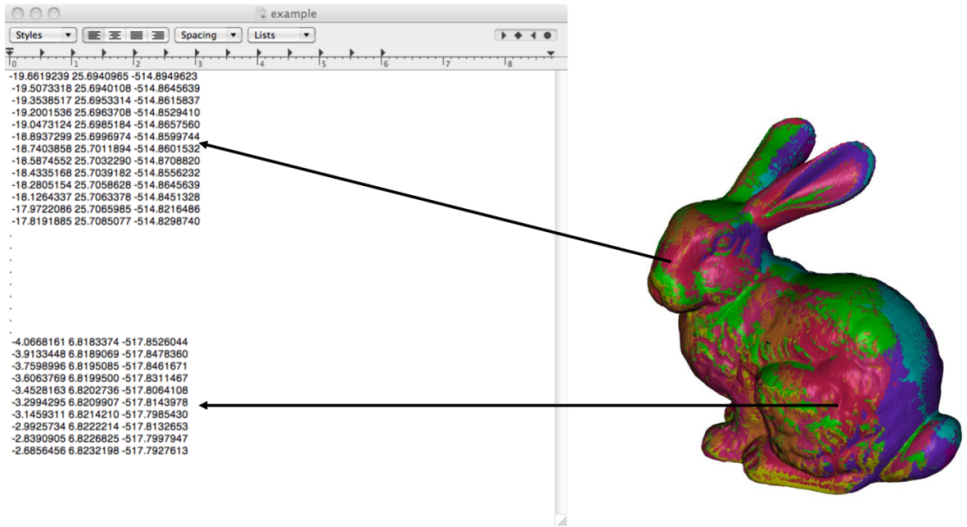
Entrada Un conjunto finito de puntos P en $\mathbb{R}^3$ muestreados de una superficie S

Salida Una función implícita $F : \mathbb{R}^3 \to \mathbb{R}$ tal que la isosuperficie de nivel cero, $F^{-1}(0)$, aproxime a S.

Definición del problema

Entrada Un conjunto finito de puntos P en $\mathbb{R}^3$ muestreados de una superficie S

Salida Una función implícita $F : \mathbb{R}^3 \to \mathbb{R}$ tal que la isosuperficie de nivel cero, $F^{-1}(0)$, aproxime a S.



Solución propuesta

Combinar las técnicas de:

- *Multilevel Partition of Unity* (MPU)
- Regresión usando Procesos Gaussianos

Partición de la Unidad

El algoritmo de partición de la unidad tiene los siguientes pasos fundamentales:

- Particionar el dominio.
- Obtener una aproximación local para cada parte.
- Combinación de las aproximaciones locales para obtener una aproximación global.

Partición de la Unidad

Aproximación global:

$$f(x,y,z) \approx \sum_{i=1}^{n} \phi_i(x,y,z) f_i(x,y,z)$$

Partición de la Unidad

Aproximación global:

$$f(x,y,z) \approx \sum_{i=1}^{\overset{n \text{ particiones}}{\downarrow\; n}} \phi_i(x,y,z) f_i(x,y,z)$$

Partición de la Unidad

Aproximación global:

$$f(x,y,z) \approx \sum_{i=1}^{n} \phi_i(x,y,z) f_i(x,y,z)$$

Pesos

Partición de la Unidad

Aproximación global:

Aproximaciones locales

$$f(x,y,z) \approx \sum_{i=1}^{n} \phi_i(x,y,z) f_i(x,y,z)$$

Partición de la Unidad

Aproximación global:

$$f(x,y,z) \approx \sum_{i=1}^{n} \phi_i(x,y,z) f_i(x,y,z)$$

- $\{\phi_i\}_{i=1\ldots n}$ son funciones no negativas y suaves con soporte compacto en el dominio de los datos tal que $\sum_{i=1}^{n} \phi_i(x,y,z) = 1$
- f_i es la función que mejor aproxima los puntos contenidos en $supp(\phi_i)$

Partición Multinivel de la Unidad (MPU)

El método MPU se divide en las siguientes etapas:

- Dividir jerárquicamente el dominio en partes. Se usa un octree.
- Regresión en cada uno de las hojas del octree para obtener una superficie local implícita.
- Unión de las distintas superficies locales para conformar una sola superficie global implícita.
- Triangulación de la superficie implícita. Se utiliza el método Marching Cubes

Partición Multinivel de la Unidad (MPU)

El método MPU se divide en las siguientes etapas:

- Dividir jerárquicamente el dominio en partes. Se usa un octree.
- Regresión en cada uno de las hojas del octree para obtener una superficie local implícita.
- Unión de las distintas superficies locales para conformar una sola superficie global implícita.
- Triangulación de la superficie implícita. Se utiliza el método Marching Cubes

Partición Multinivel de la Unidad (MPU)

El método MPU se divide en las siguientes etapas:

- Dividir jerárquicamente el dominio en partes. Se usa un octree.
- Regresión en cada uno de las hojas del octree para obtener una superficie local implícita.
- Unión de las distintas superficies locales para conformar una sola superficie global implícita.
- Triangulación de la superficie implícita. Se utiliza el método Marching Cubes

Partición Multinivel de la Unidad (MPU)

El método MPU se divide en las siguientes etapas:

- Dividir jerárquicamente el dominio en partes. Se usa un octree.
- Regresión en cada uno de las hojas del octree para obtener una superficie local implícita.
- Unión de las distintas superficies locales para conformar una sola superficie global implícita.
- Triangulación de la superficie implícita. Se utiliza el método Marching Cubes

Proceso Gaussiano

Definición

Una sucesión $\{f(x_i)\}_{i \in I}$ se dice que es un proceso Gaussiano si dado cualquier conjunto $x_1, ..., x_m$, el vector:

$$\mathbf{f} = \{f(x_1), ..., f(x_m)\} \sim \mathcal{N}(m_f, K)$$

$$m_f = (E(f(x_1), ..., E(f(x_m)),$$

$$K = (cov(f(x_i), f(x_j)))$$

Regresión

Sean $x_i \in \mathbb{R}^m$ las entradas, $y_i \in \mathbb{R}^d$ las salidas:

$$y = f(x) + \epsilon,$$

donde $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
Asumiendo que la función $f(x)$ proviene de un *proceso Gaussiano*:

$$f_x | D \sim \mathcal{N}(m_{post}(x), \sigma_{post}(x)^2),$$

donde f_x es el valor de la función f en un nuevo dato x, y D es el conjunto de entrenamiento.

$$f_x|D \sim \mathcal{N}(m_{post}(x), \sigma_{post}(x)^2),$$

Se obtiene lo siguiente:

$$m_{post}(x) = \boldsymbol{k}^T(x)(\boldsymbol{K} + \sigma^2 I)^{-1}\boldsymbol{y},$$

$$\sigma^2_{post}(x) = k(x,x) - \boldsymbol{k}^T(x)(\boldsymbol{K} + \sigma^2 I)^{-1}\boldsymbol{k}(x).$$

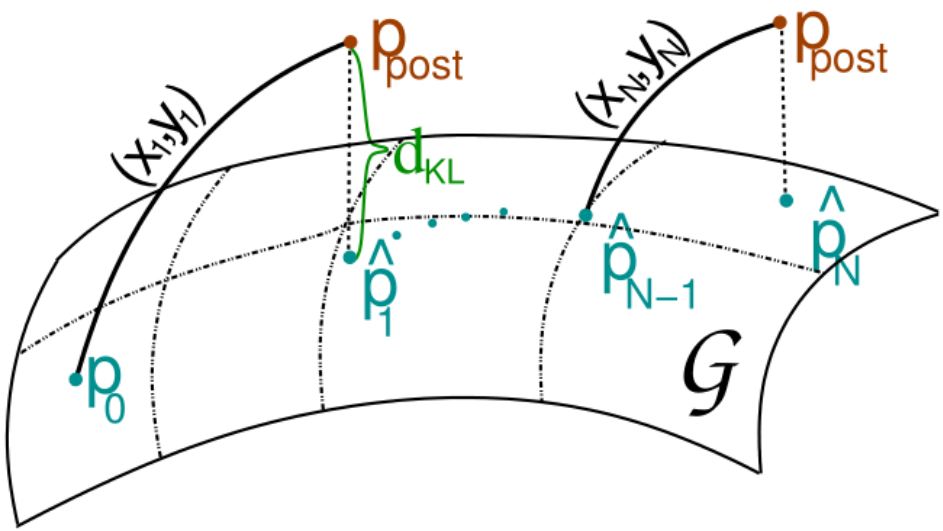
Donde:

- $\boldsymbol{k}(x) = (k(x,x_1), ..., k(x,x_n))^T$
- $\boldsymbol{K}$ es la matriz de covarianzas del proceso Gaussiano evaluada en D
- σ^2 es la varianza de ϵ

Regresión Online

La posterior p_{t+1} en el paso $t+1$ es calculada como:

$$p_{t+1}(\boldsymbol{f}) = \frac{P(y_{t+1} \mid \boldsymbol{f})\hat{p}_t(\boldsymbol{f})}{E_t(P(y_{t+1} \mid \boldsymbol{f}_D))}$$



Visualización de la aproximación online del proceso posterior, donde $p_{post} = p_N$ en el paso N.

Regresión Dispersa

- Se tiene un conjunto de vectores base (BV) con una capacidad máxima.
- Dado un nuevo dato de entrenamiento $(\boldsymbol{x}, y)$:
  - El vector $\boldsymbol{x}$ se descompone como la suma de sus proyecciones en el espacio generado por el conjunto BV más el vector residual v_{res}.
  - Si $||v_{res}||$ es menor que una tolerancia dada, entonces el punto nuevo se considera redundante y $\boldsymbol{x}$ es proyectado en el espacio generado por el conjunto BV.
  - En caso contrario, el vector $\boldsymbol{x}$ es incluido en el conjunto BV
  - Si el conjunto BV ha sobrepasado su capacidad, el vector que contribuya menos a la representación del GP es removido.

Kernels

Nombre	Parámetros	Kernel
Radial Basis Function	A, b, $\mathbf{w}$	$Ae^{\frac{-1}{2d}\sum_{i=1}^{d}\mathbf{w}_i(\mathbf{x}_i-\mathbf{y}_i)^2}+b$
RBF simple	w	$e^{\frac{-1}{2d}w\|\mathbf{x}-\mathbf{y}\|^2}$
Polinomial	α, c, d	$(\alpha\mathbf{x}^T\mathbf{y}+c)^d$
Thin Plate	R	$2\|\mathbf{x}-\mathbf{y}\|^3-3R\|\mathbf{x}-\mathbf{y}\|^2+R^3$

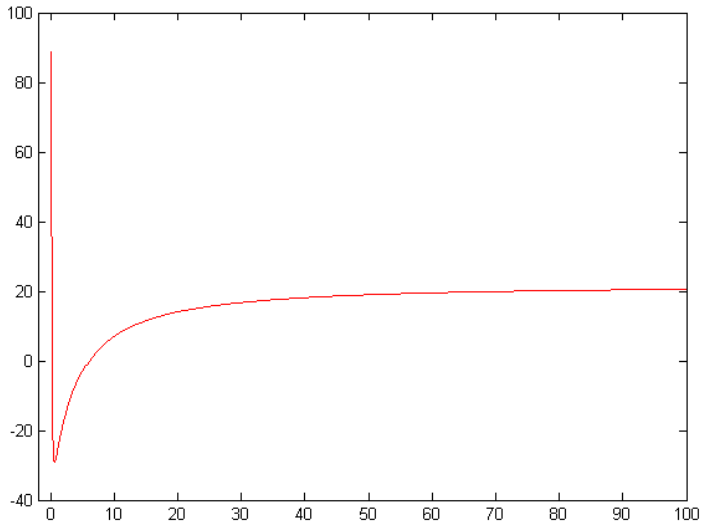
Optimización de parámetros

El proceso Gaussiano depende de un conjunto de parámetros θ:

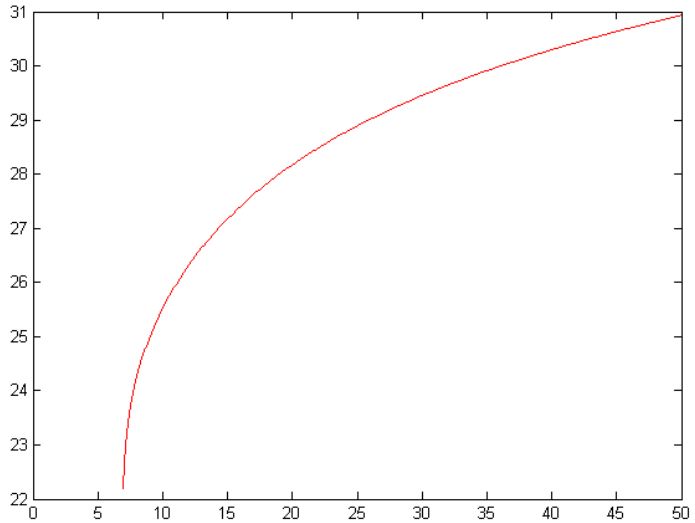
- La varianza del ruido de los datos s_0^2
- Otros parámetros usados por el kernel

Una forma determinar θ es buscando el valor que haga que el conjunto de entrenamiento tenga la mayor probabilidad de haber ocurrido. Esto puede hacerse maximizando la verosimilitud $L(\theta)$

Estructura de log-verosimilitud



$-\log(L(\theta))$ usando el kernel RBF simple.



$-\log(L(\theta))$ usando el kernel Thin Plate

Algoritmo MPU paralelizable

Entrada: Un conjunto de puntos **P**
Salida: El valor pesado w de cada punto en **P**

1: Inicia el recorrido del Octree en la raíz
2: Se realiza el aprendizaje en el nodo actual
3: **Si** se cumplen las condiciones de división **entonces:**
4: El nodo se divide
5: Continúa el recorrido en todos los hijos (paso 2)
6: **Para** cada punto **p** en **P :**
7: Inicia el recorrido del Octree en la raíz
8: **Si p** cae dentro del soporte compacto del nodo **entonces:**
9: **Si** el nodo actual es una hoja **entonces:**
10: Se realiza la predicción para **p**
11: Se calcula el peso de **p**
12: Se agrega el valor predecido y el peso a w, siguiendo la fórmula ??
13: **Si no**
14: Continúa el recorrido en todos los hijos (paso 8)

Condiciones de división

Un nodo se divide si se cumple alguna de las siguientes condiciones:

- El error de predicción es mayor al máximo permitido
- La cantidad de puntos en el nodo es mayor al máximo permitido

Sin embargo, aunque se cumplan las condiciones anteriores, un nodo no puede dividirse si se cumple alguna de las siguientes condiciones:

- La profundidad de los hijos excedería el máximo permitido
- La cantidad de puntos en el nodo actual es menor al mínimo permitido

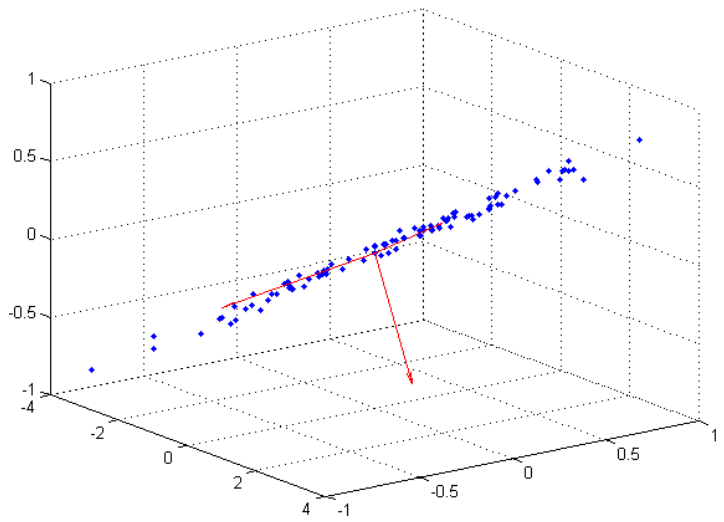
Proceso de aprendizaje

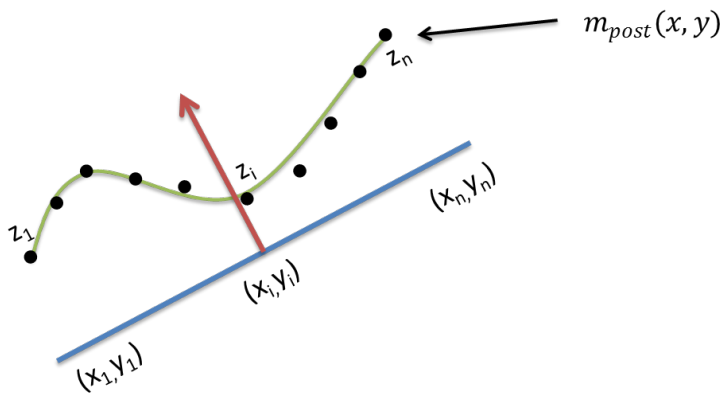
Entrada: Un conjunto de puntos D, un radio inicial r_0, la cantidad mínima de puntos p_{min}, el factor de aumento λ

Salida: Una función de predicción

1: Consigue el conjunto P de puntos en D que caen dentro del soporte compacto del nodo usando el radio $r = r_0$
2: **Mientras** $|P| < p_{min}$ **:**
3: Aumenta el radio: $r = r + \lambda r_0$
4: Consigue el conjunto P de puntos en D que caen dentro del soporte compacto del nodo con radio r
5: Calcula la función de aproximación con los puntos P
6: Calcula la el error de la función de aproximación

Análisis de componentes principales





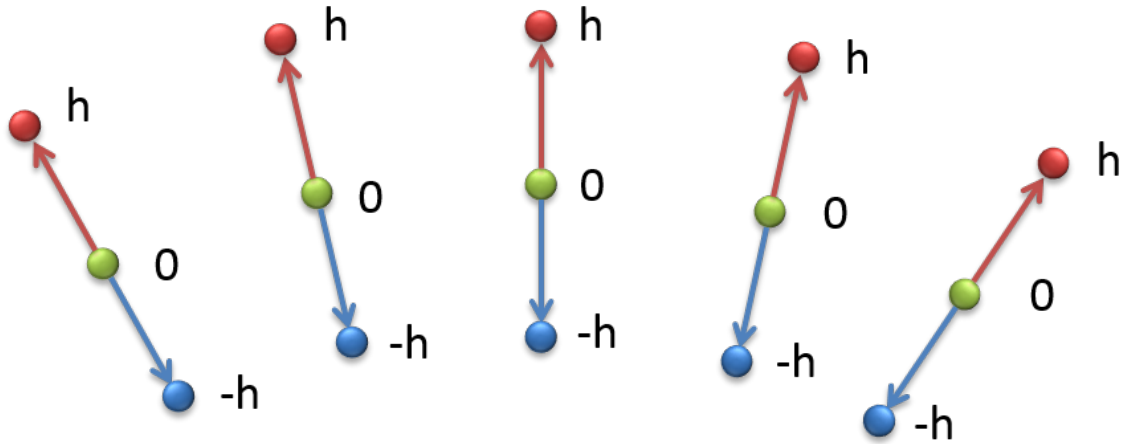
Análisis de componentes principales

- Cuando se obtienen los planos es posible que en dos nodos adyacentes se obtengan orientaciones opuestas.
- Es posible invertir la orientación del plano bajo condiciones como las siguientes:
  - $\mathbf{n} \cdot \overline{\mathbf{N}} < 0$, donde $\mathbf{n}$ es la normal al plano y $\overline{\mathbf{N}}$ es la media de las normales de los puntos del nodo
  - La cantidad de nodos adyacentes con orientaciones opuestas es mayoría.

Desplazamiento de normales

Se utilizan los puntos extendidos (x,y,z,w), donde w indica la *altura* del punto con respecto a la superficie:

- Positivo en el exterior de la superficie
- Negativo en el interior de la superficie
- 0 en la superficie



Cálculo del error

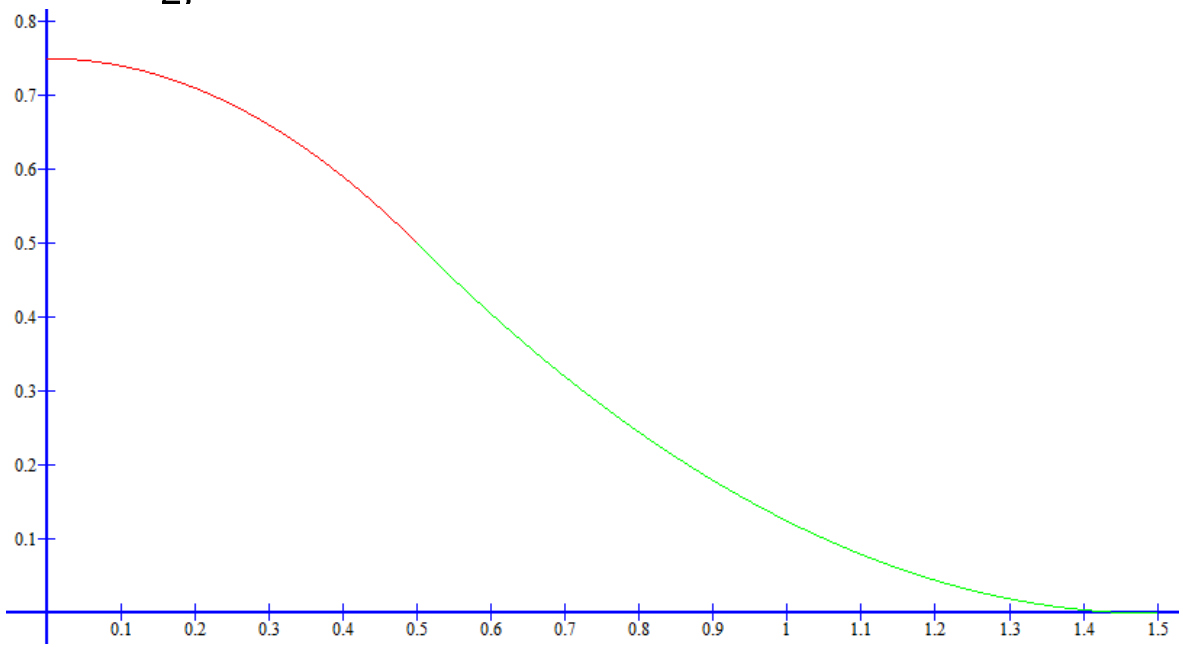
El error puede obtenerse evaluando la aproximación obtenida en los puntos de entrada. Esto nos da un conjunto de diferencias entre el valor calculado y el "real", de las cuales puede obtenerse una sola medida de error de las siguientes maneras:

- Error máximo
- Media aritmética
- Media cuadrática
- Método de Taubin

Se escogió la media cuadrática

Pesos

El método elegido utiliza la distancia al centro y el radio r_i tomado como $\frac{3}{4}$ de la diagonal del nodo, usando un *spline* cuadrático $b(d)$, donde $d = \frac{3|(x,y,z)-c_i|_2}{2r}$:



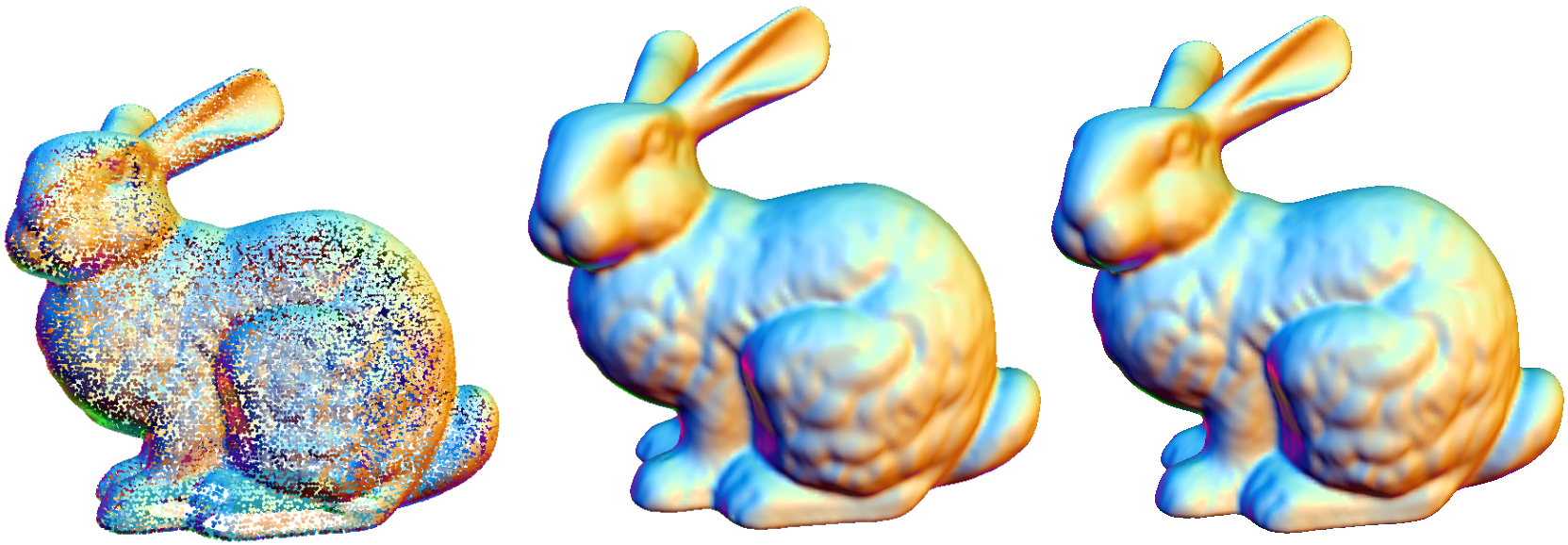
Complejidad

Proceso Gaussiano	Media	Varianza
Batch	$O(l\bar{n}^3 + d^3\bar{n})$	$O(l\bar{n}^3 + d^3\bar{n}^2)$
Disperso	$O(lk^2\bar{n} + d^3k)$	$O(lk^2\bar{n} + d^3k^2)$

- l es la cantidad de hojas del *octree*
- $\bar{n}$ es la cantidad promedio de puntos por nodo
- d es la cantidad de de divisiones del espacio a usar en la discretización para el algoritmo *Marching Cubes*
- k es la capacidad elegida para el Proceso Gaussiano disperso

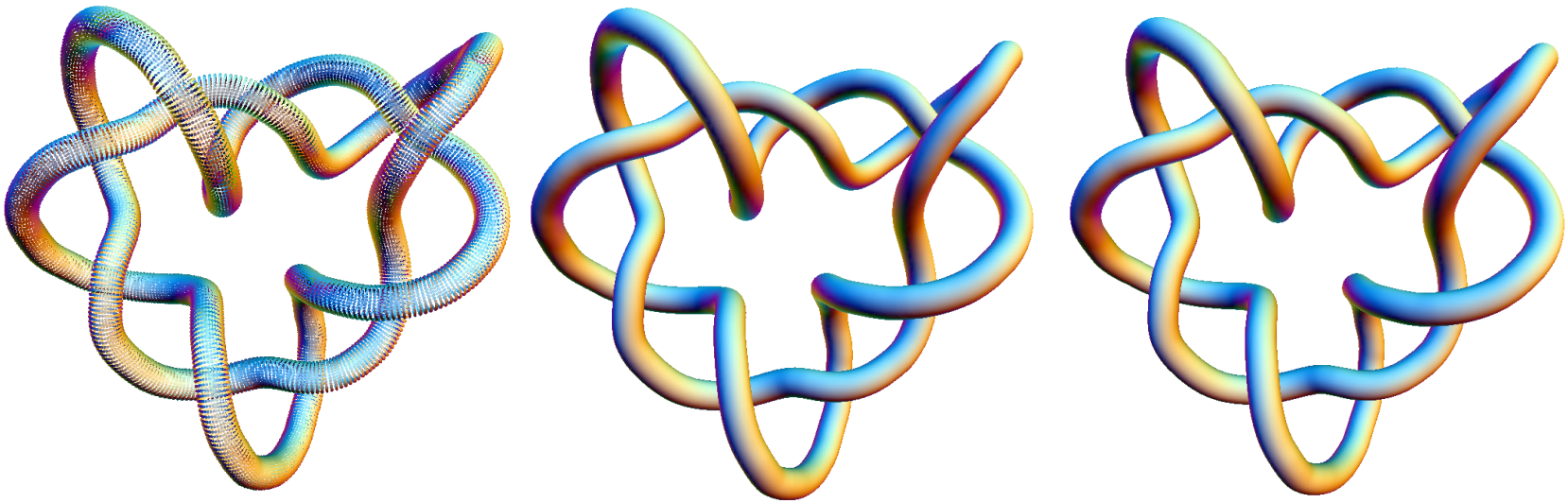
Resultados

Modelos con suficientes puntos



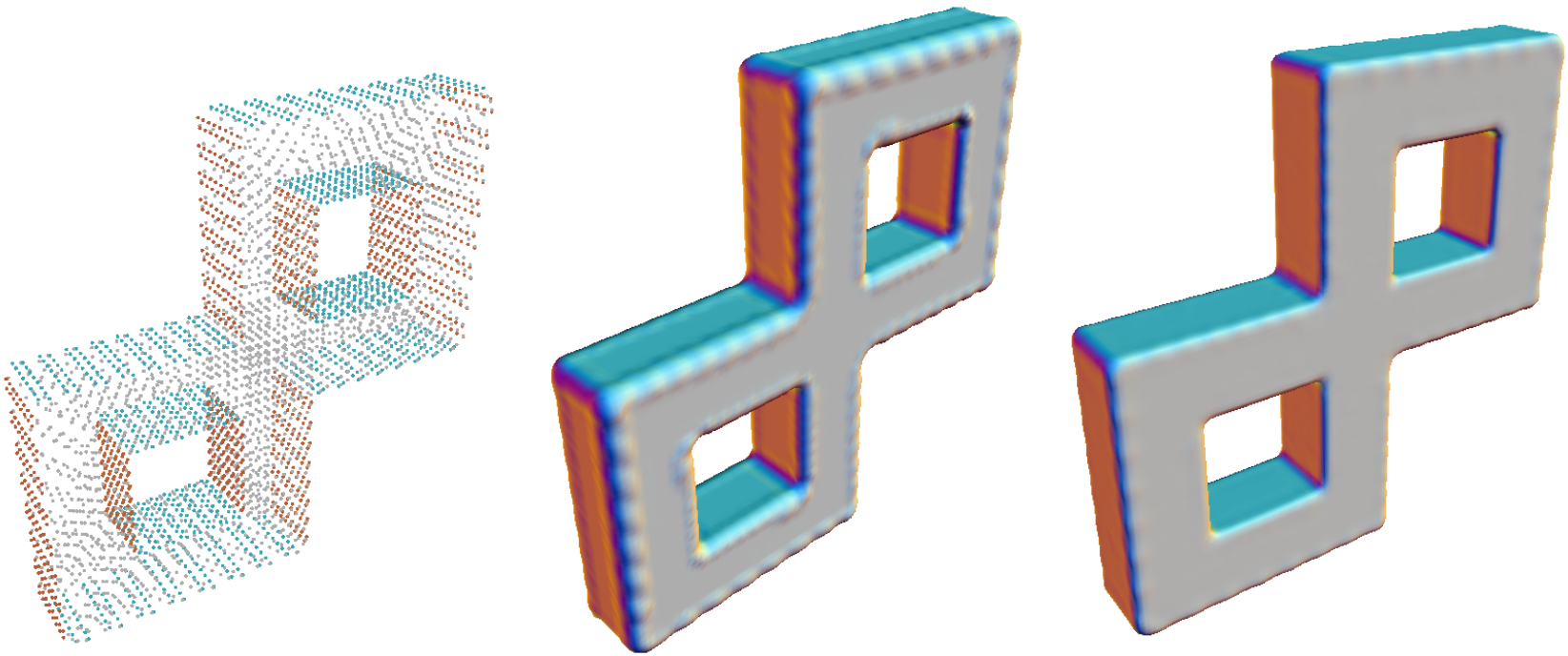
Modelo Stanford Bunny. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.

Modelo Armadillo. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.

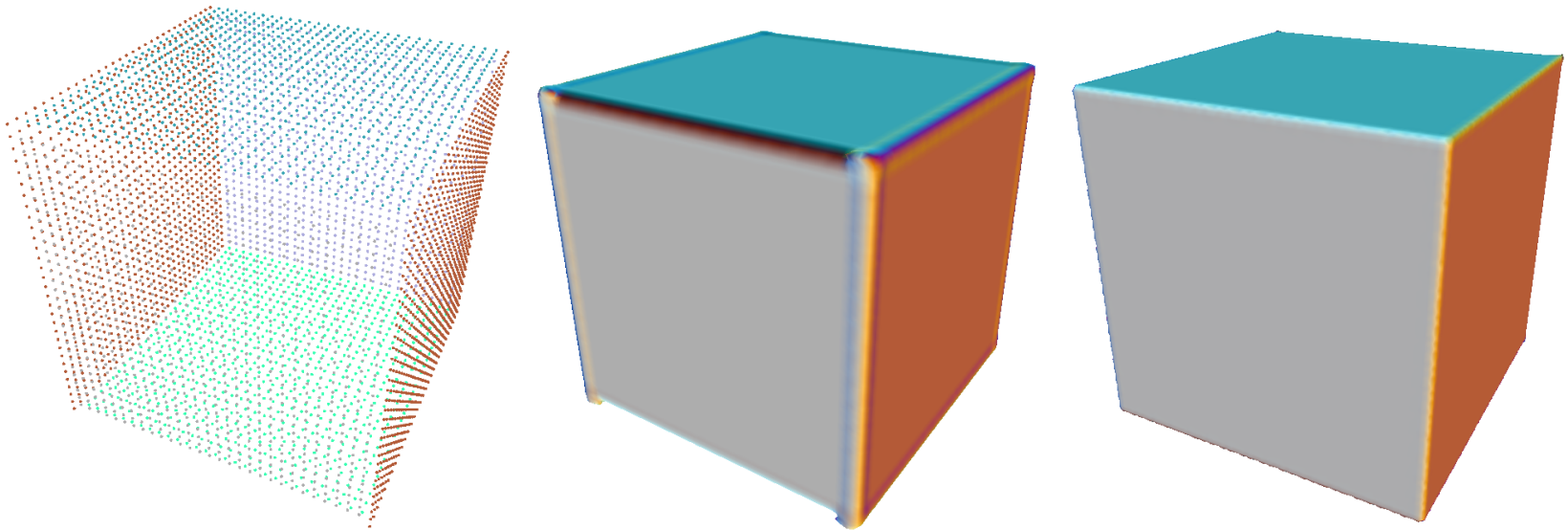


Modelo Knot. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.

Modelos con bordes

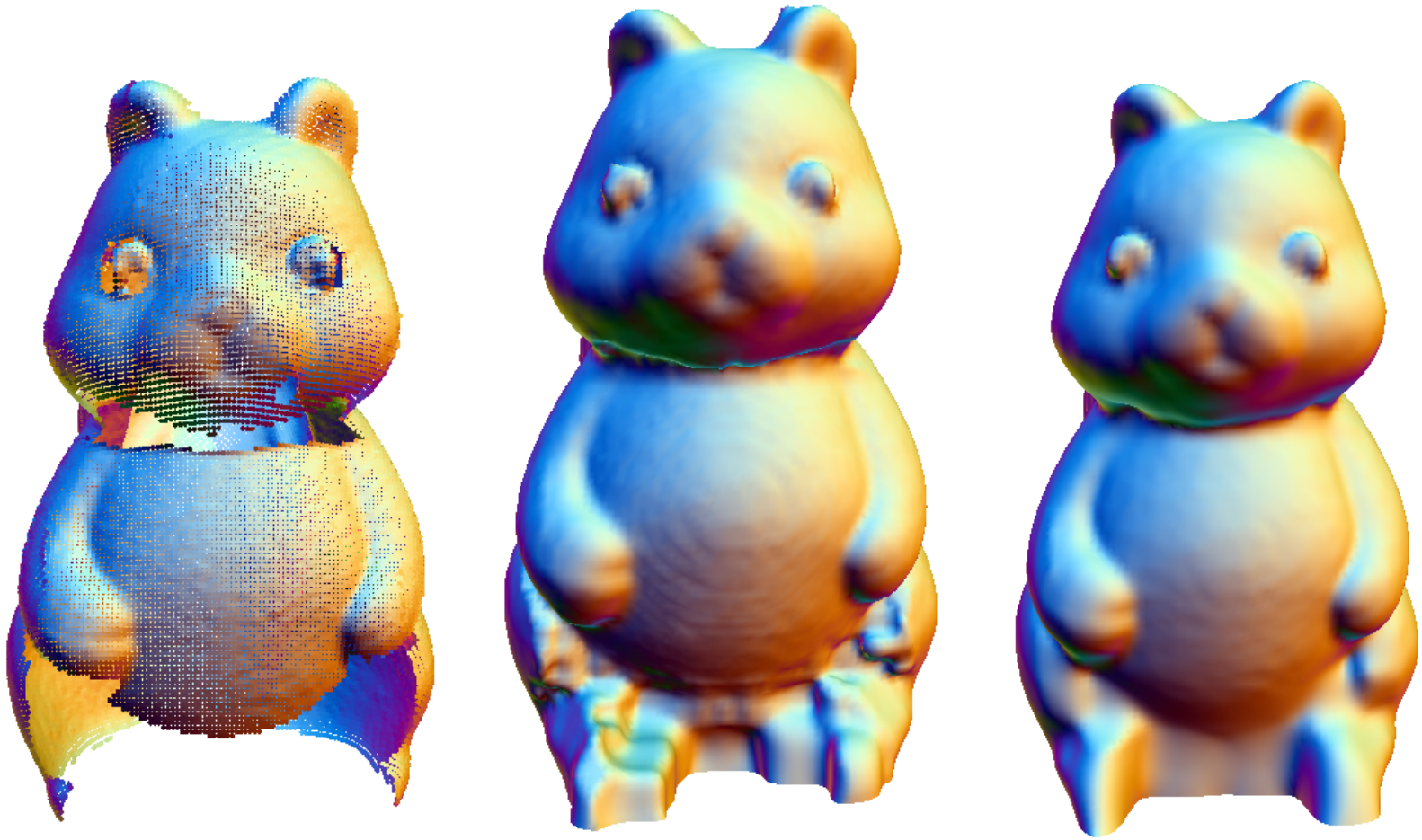


Modelo Double toro. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.



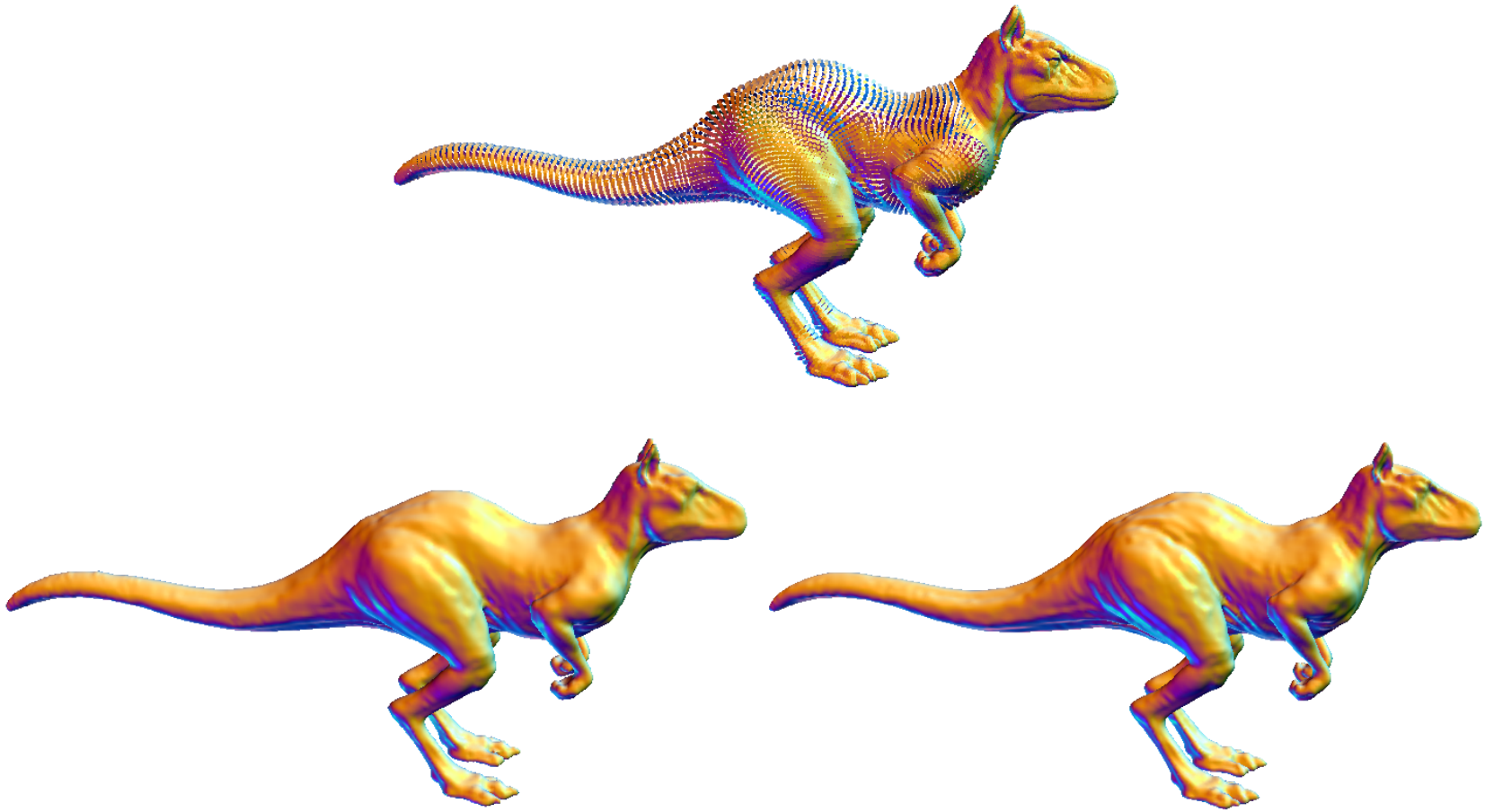
Modelo Cubo. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.

Modelos con regiones faltantes

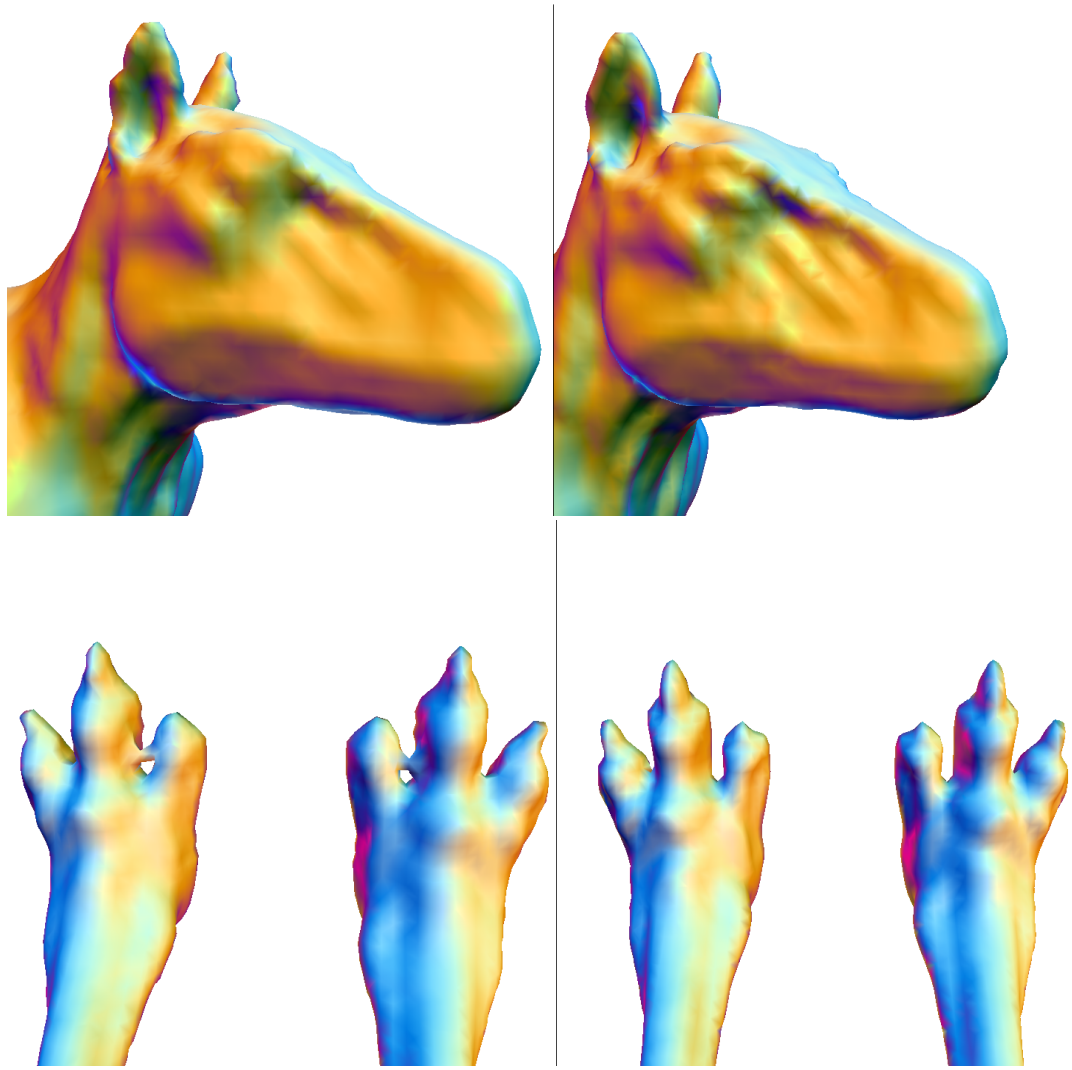


Modelo Squirrel. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.

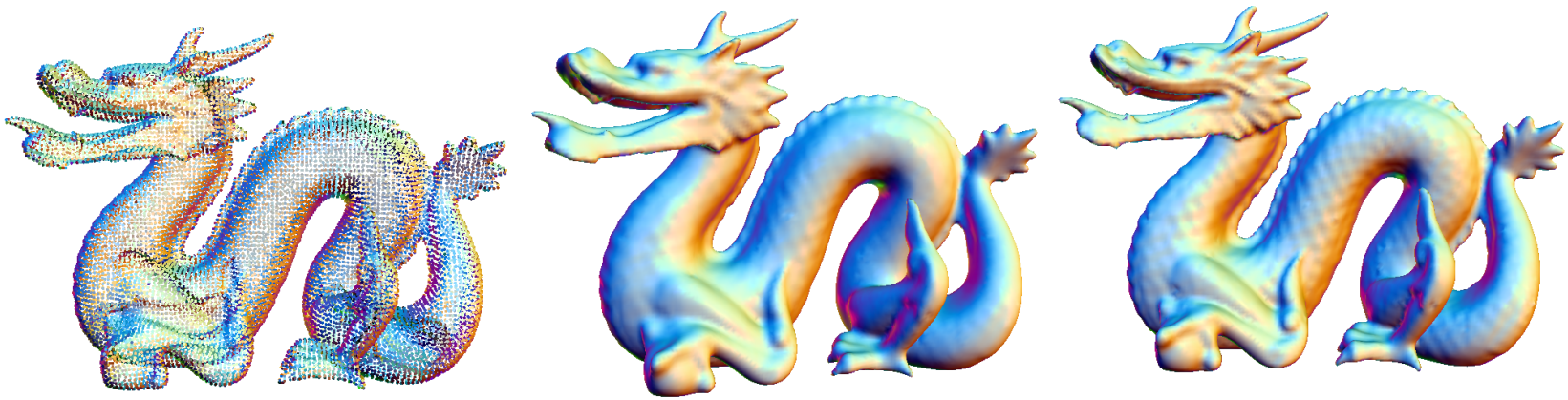
Modelos con detalles finos



Modelo Dino. Arriba: conjunto original de puntos. Medio: reconstrucción usando método original. Abajo: reconstrucción usando método propuesto.



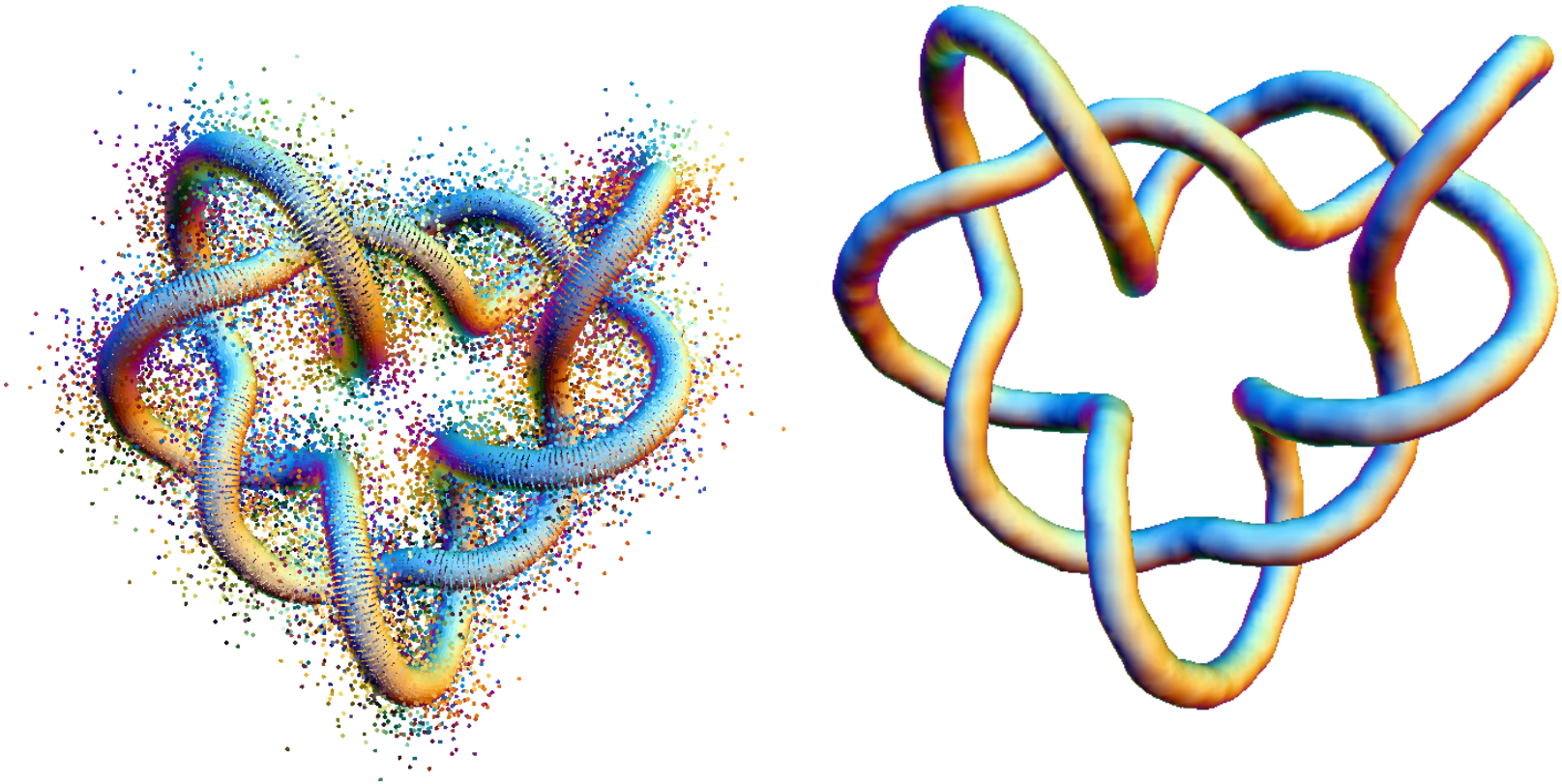
Acercamiento del modelo Dino. Izquierda: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.



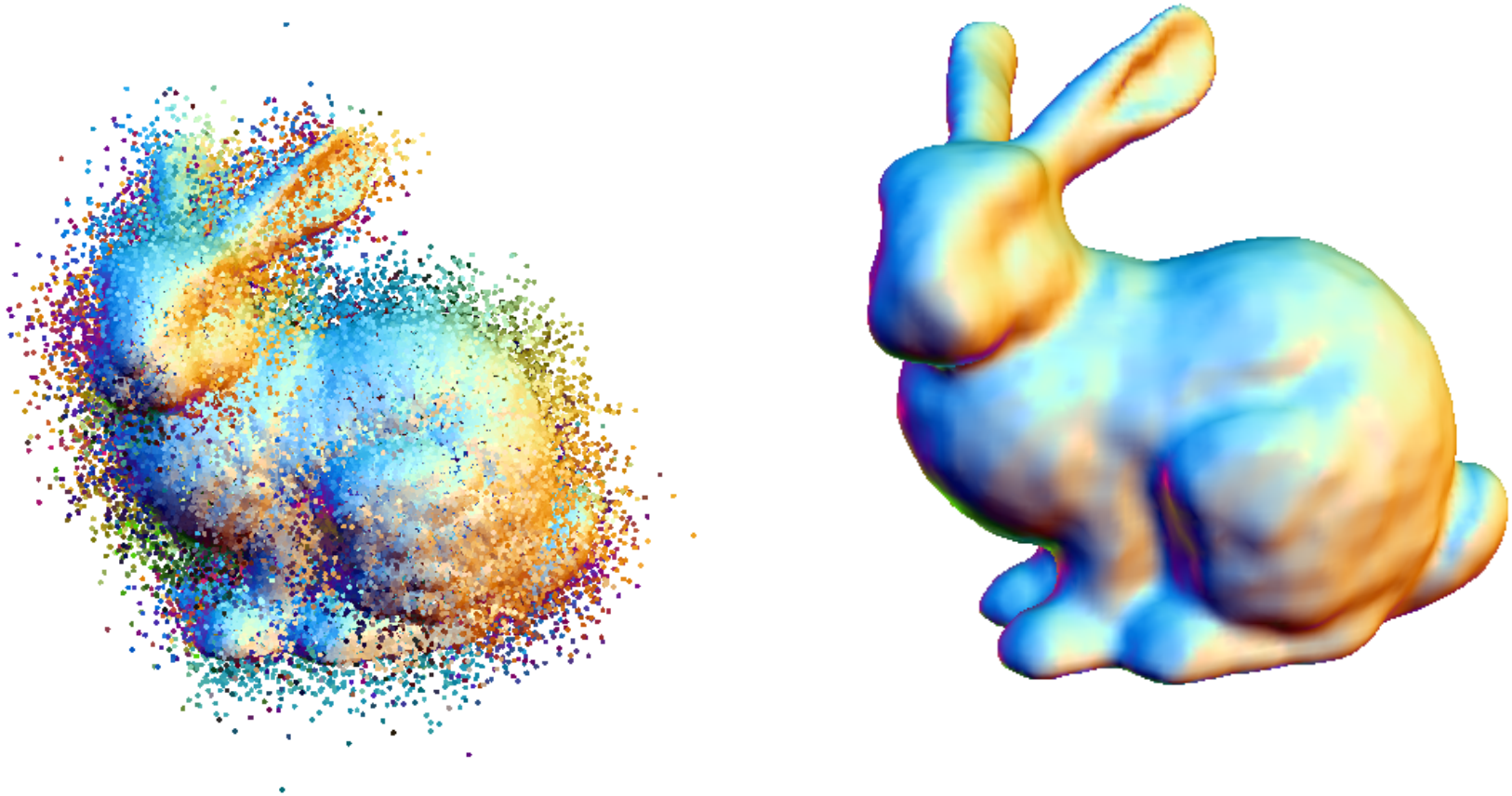
Modelo Dragon. Izquierda: conjunto original de puntos. Medio: reconstrucción usando método original. Derecha: reconstrucción usando método propuesto.

Modelos con ruido[1]

[1] Y normales

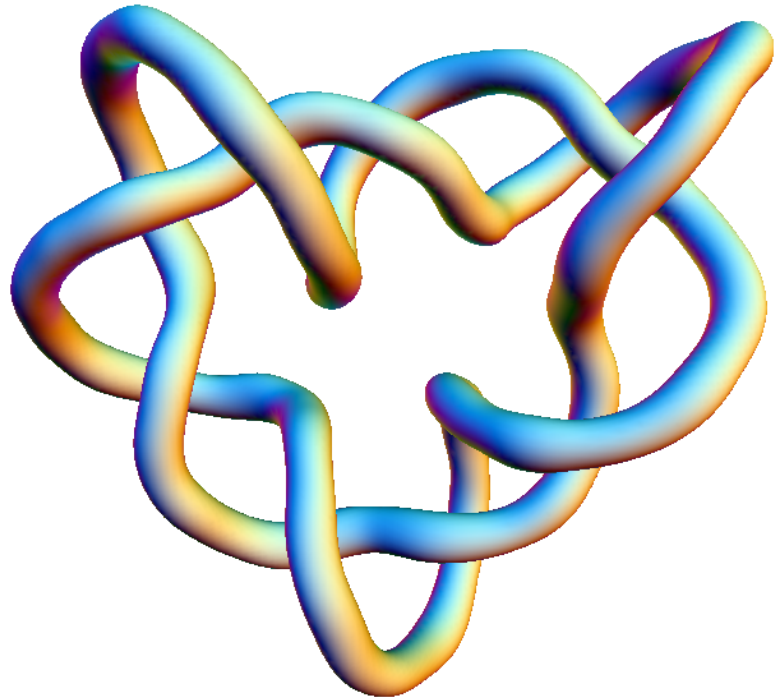


Izquierda: modelo Knot con ruido agregado. Derecha: reconstrucción usando método propuesto

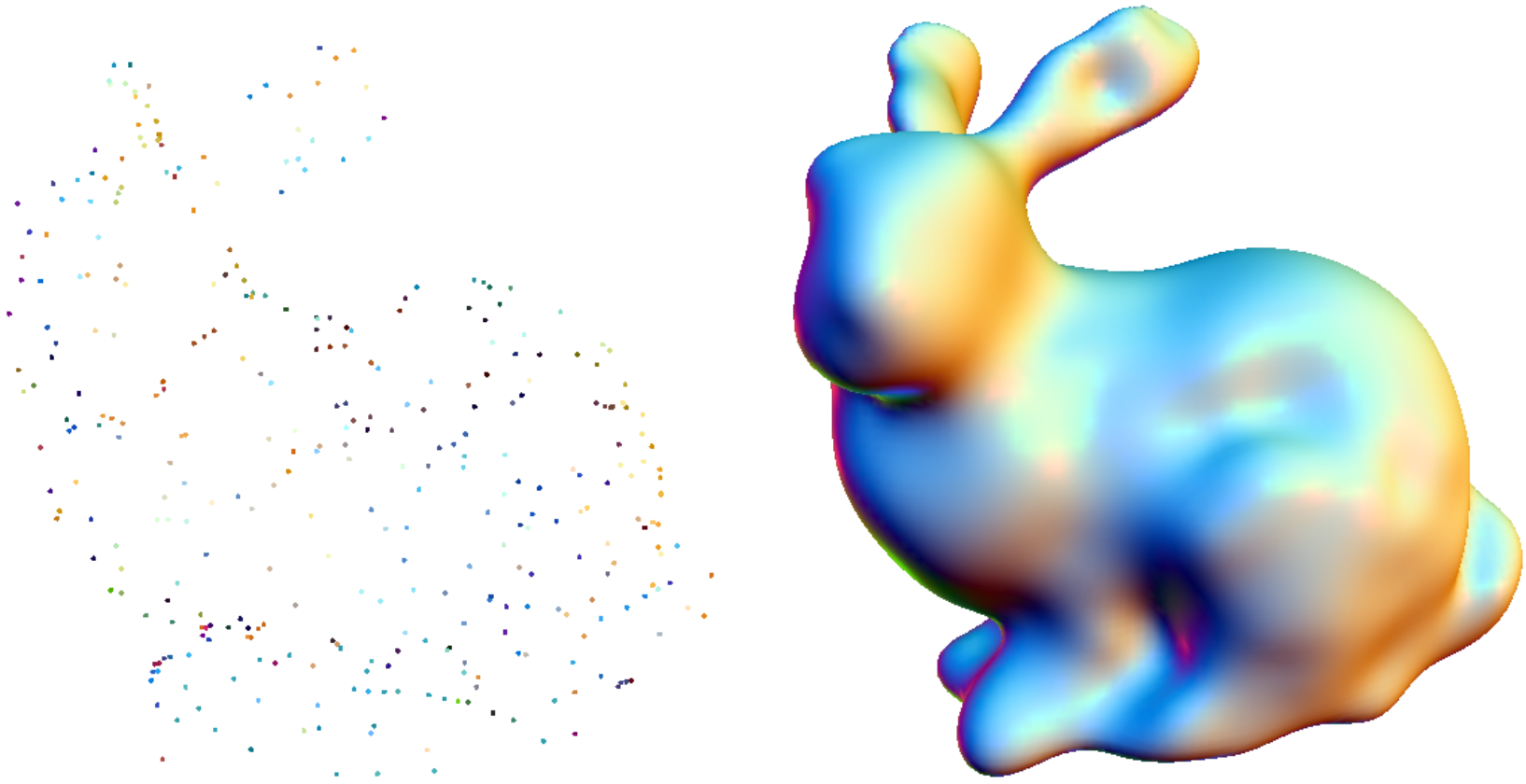


Izquierda: modelo Stanford Bunny con ruido agregado. Derecha: reconstrucción usando método propuesto

Modelos dispersos

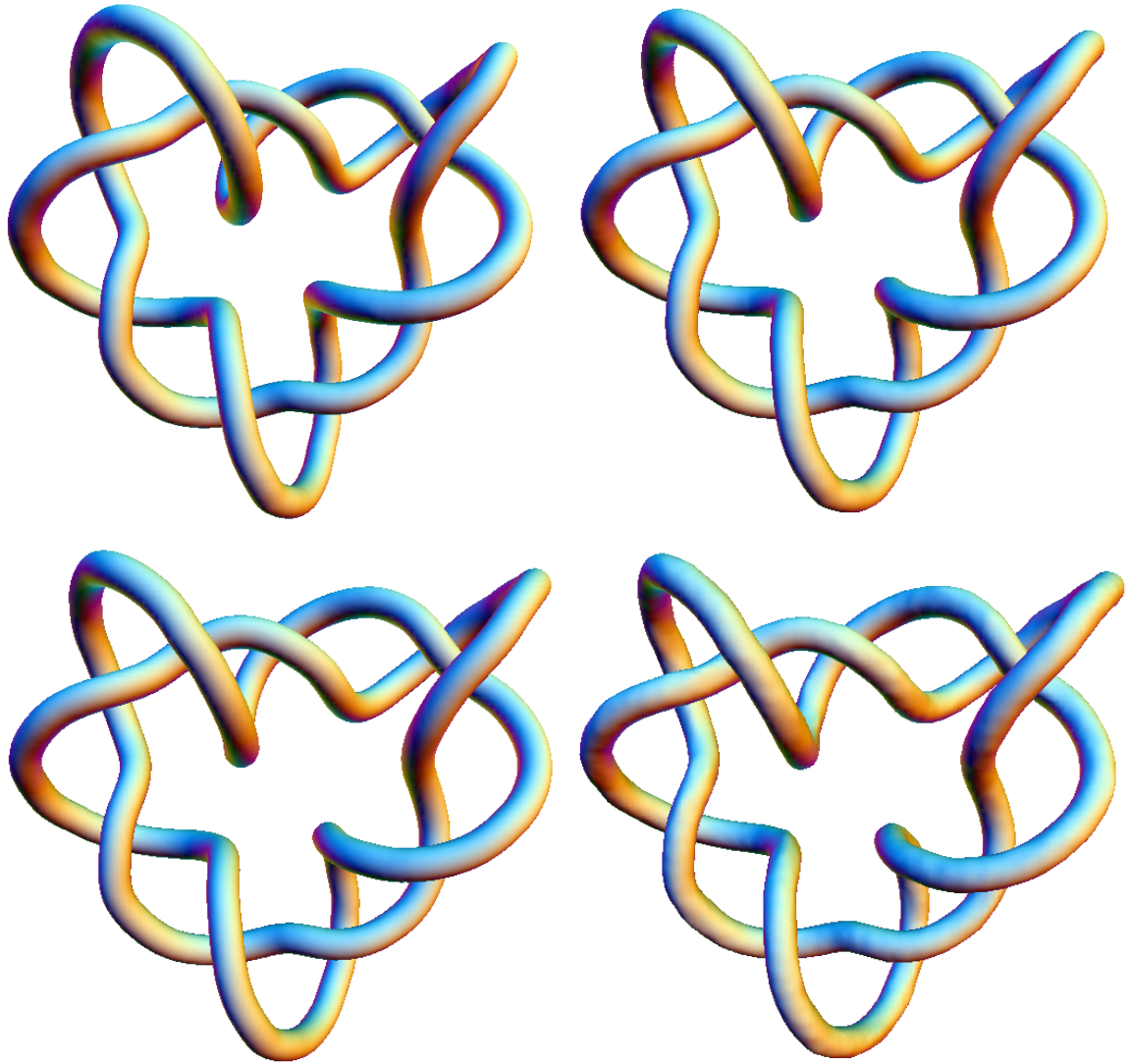


Izquierda: modelo Knot con 350 puntos. Derecha: reconstrucción usando método propuesto

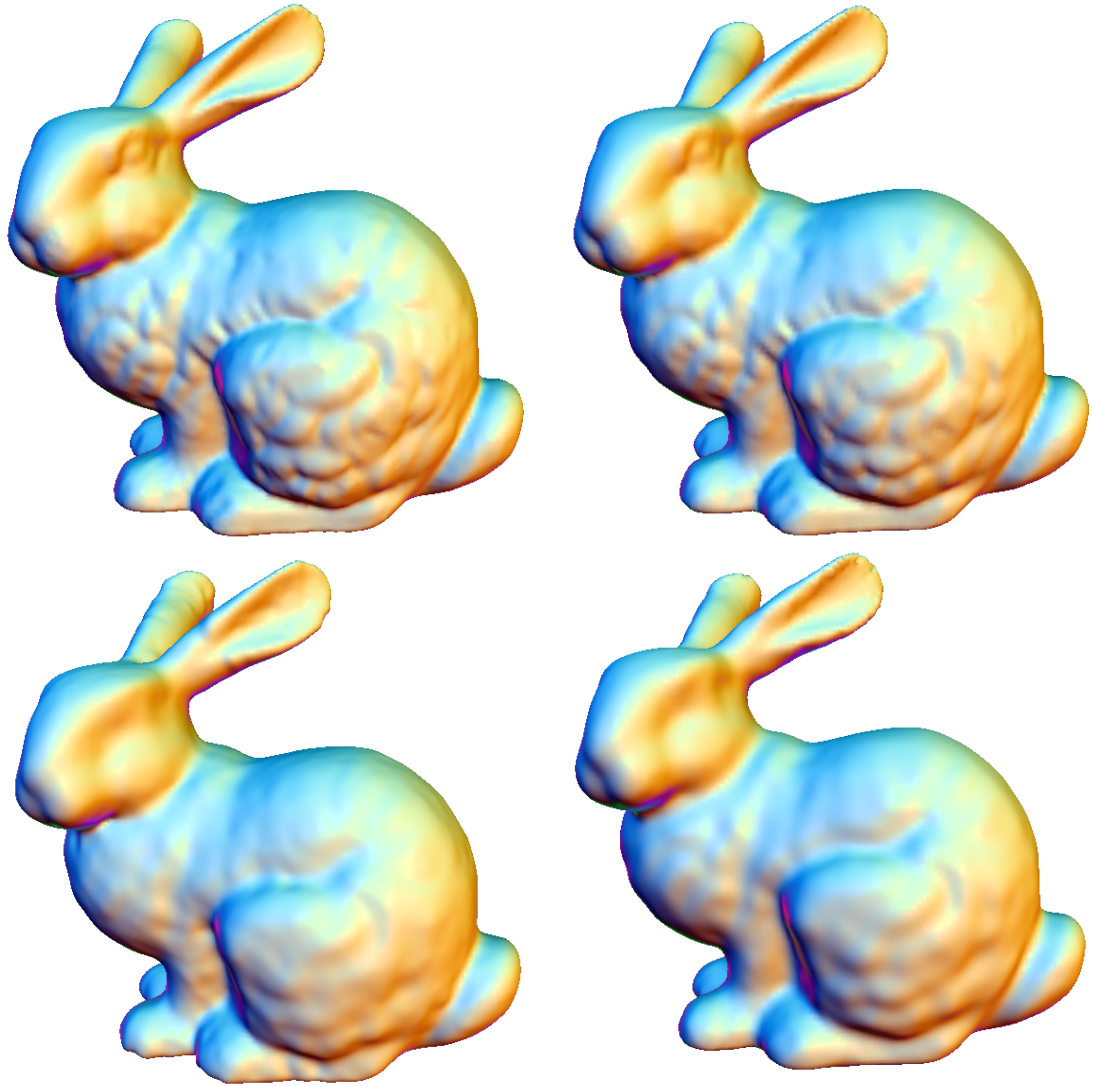


Izquierda: modelo Stanford Bunny con 350 puntos. Derecha: reconstrucción usando método propuesto

Proceso Gaussiano online disperso

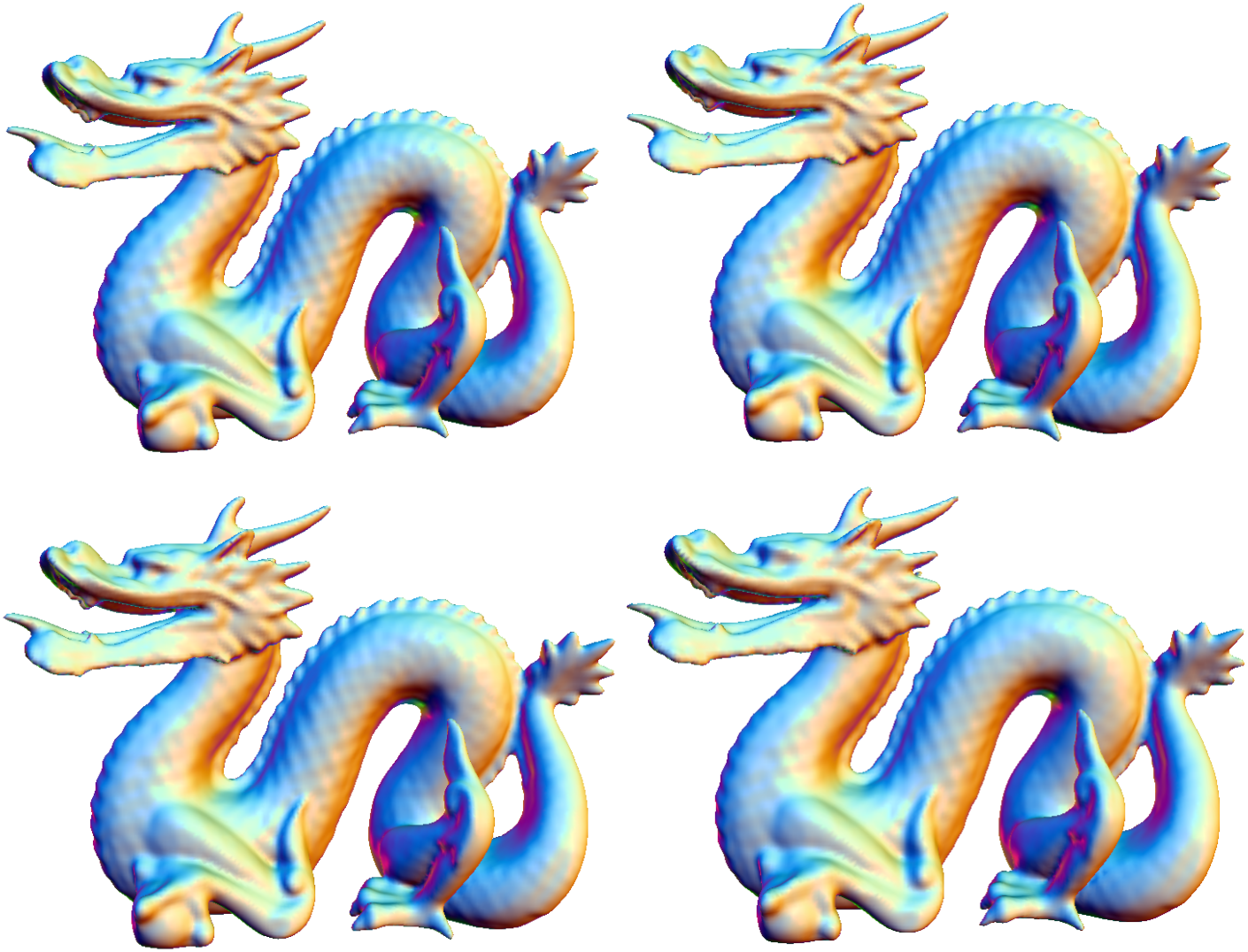


Modelo Knot reconstruido usado 100% (superior izquierda), 50% (superior derecha), 25% (inferior izquierda) y 12.5% (inferior derecha) de los datos.



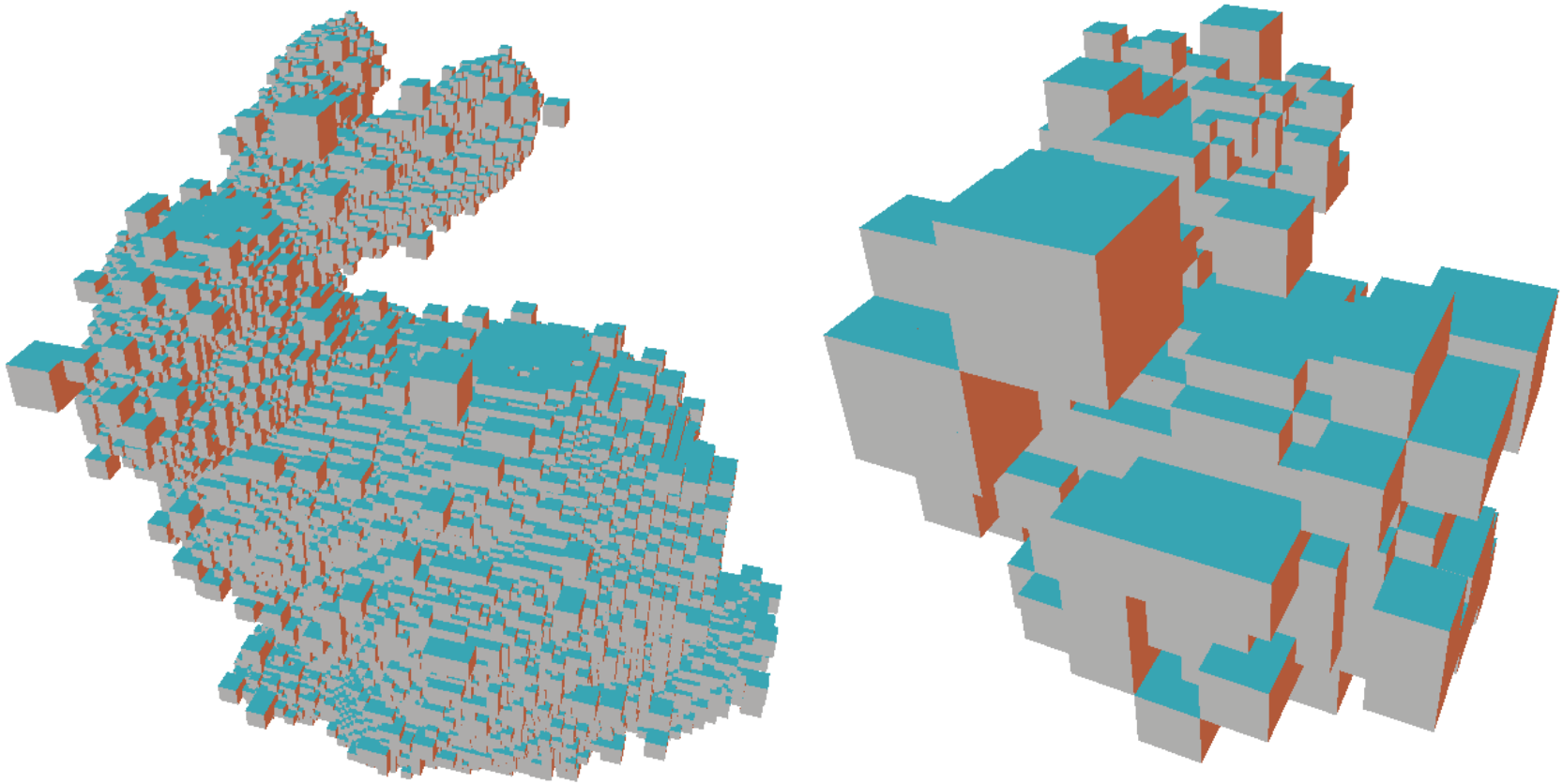
Modelo Stanford Bunny reconstruido usado 100% (superior izquierda), 50% (superior derecha), 25% (inferior izquierda) y 12.5% (inferior derecha) de los datos.

Modelo Squirrel reconstruido usado 100% (superior izquierda), 50% (superior derecha), 25% (inferior izquierda) y 12.5% (inferior derecha) de los datos.

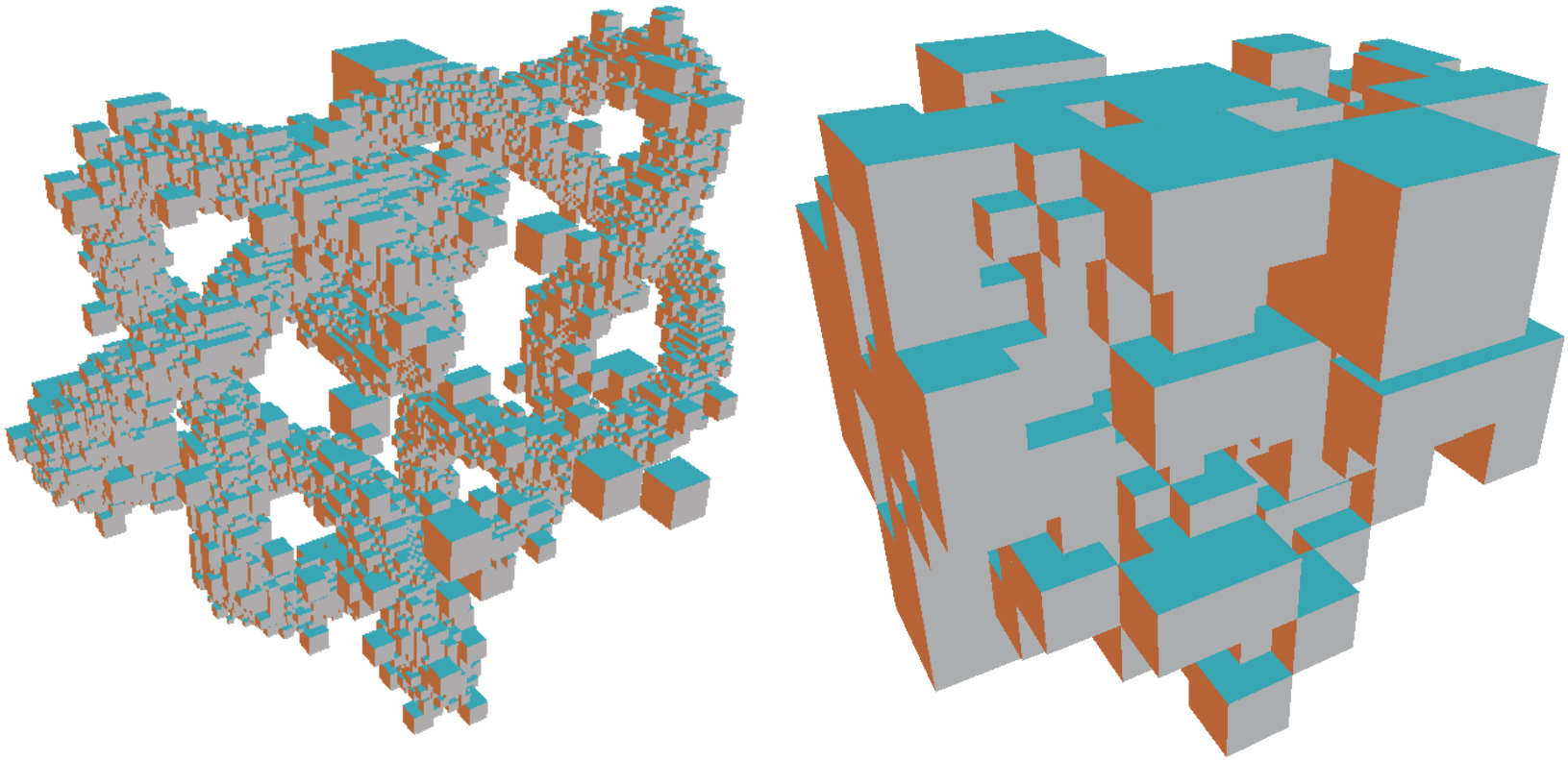


Modelo Dragon reconstruido usado 100% (superior izquierda), 50% (superior derecha), 25% (inferior izquierda) y 12.5% (inferior derecha) de los datos.

Cantidad de divisiones



Octree del modelo Stanford Bunny. Izquierda: método original. Derecha: Método propuesto.



Octree del modelo Knot. Izquierda: método original. Derecha: Método propuesto.

Conclusiones

- Por primera vez se combina el método MPU con Procesos Gaussianos.
- Se logró una reconstrucción mejor de las zonas con una baja densidad de puntos o regiones faltantes
- Los detalles fueron recuperados más notoriamente
- Se lograron usar sólo los datos más representativos en la versión online dispersa
- Se preservó la topología.

Trabajo futuro

- Investigar más opciones para kernels
- Buscar otras formas de eliminar la necesidad de las normales
- Conseguir robustez al ruido de una forma que no requiera normales
- Mejorar la calidad y estabilidad del Proceso Gaussiano online disperso