Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Estimación de la cápsula no-convexa de nubes de puntos por partición de la unidad

Tesis de Licenciatura en Ciencias de la Computación

Shai Bianchi

Director: Dr. Francisco Gómez Fernández

Buenos Aires, 2019

Estimación de la cápsula no-convexa de nubes de puntos por partición de la unidad

Un paso clave en los procedimientos de reconstrucción 3D suele ser el procesamiento de nubes de puntos que representan objetos de la vida real. Damos una taxonomía de los algoritmos empleados para dicha tarea e identificamos como nuestro foco a los que toman nubes de puntos orientadas y devuelven funciones de distancia con signo. Elegimos dos de particular interés, *Non-Convex Hull Surfaces (NCH)* y *Multi-level Partition of Unity Implicits (MPU)*, estudiamos sus fundamentos teóricos y los contrastamos con observaciones acerca de su comportamiento ante distintas configuraciones de sus parámetros. En el caso de NCH, aportamos un nuevo análisis que compara detalladamente sus diferentes variantes en base a la teoría y sus resultados de reconstrucción. Por su parte, mostramos que la parametrización de MPU sugerida en el artículo original no resulta de utilidad para nuestro conjunto de datos y proponemos un valor que nos provee reconstrucciones de mayor calidad. Por último, presentamos un algoritmo nuevo que resulta de una combinación de los enfoques de los dos anteriores. Analizamos en detalle su comportamiento en base a una evaluación cuantitativa rigurosa de su calidad de reconstrucción utilizando el framework de benchmarking *Reconbench*. El nuevo método, bautizado *Partition of Unity Non-Convex Hull (PUNCH)*, exhibe una mejora muy marcada de tiempos de ejecución en comparación con su predecesor NCH sin perder capacidad de reconstrucción. Mostramos por medio de una evaluación cuantitativa y cualitativa que PUNCH se compara favorablemente tanto con los dos algoritmos en los que se inspira como con *Screened Poisson*, un método estándar en el área.

Palabras claves: Reconstrucción de superficies 3D - Funciones implícitas - Partición de la unidad.

Partition of Unity Non-Convex Hull Surfaces

Algorithms for processing point clouds are often key to 3D surface reconstruction procedures. We focus on a subset of such algorithms which take oriented 3D point clouds as input and produce signed-distance functions as output. We turn our attention to two in particular, *Non-Convex Hull Surfaces (NCH)* and *Multi-level Partition of Unity Implicits (MPU)*, and study the behavior of their parametrization. In particular, we provide a new detailed comparative analysis of the three different variations of the NCH algorithm. As for MPU, we show that the parameter configuration suggested by the method's original article produces unfavorable results when applied to our data sets, and we suggest a new value that leads to more satisfactory reconstructions. Finally, we present a new algorithm that builds on the previous two by merging both of their approaches. We present a detailed analysis of its behavior under a variey of parameters relying in part on the *Reconbench* framework for quantitative evaluation of reconstruction quality. The new algorithm, baptized *Partition of Unity Non-Convex Hull (PUNCH)*, demonstrates a significant improvement to its predecessor NCH in terms of running time without yielding reconstruction quality. We show by means of quantitative and qualitative evaluation that PUNCH compares favorably to MPU as well as to *Screened Poisson*, a standard reconstruction method.

Keywords: 3D surface reconstruction - Implicit surfaces - Partition of unity.

Agradecimientos

A mi director de tesis, Francisco.
Al director de mi director de tesis, Gabriel Taubín, por sus ideas y aportes.
A los correctores de la tesis por su interés y predisposición.

A Jennifer, phran, Felipe Garrote y Vrapper.

A Iglesias, Becher y Rosner por la chispa. A Mínimo.

A estas y otras olas, vientos,
y a mi familia, el faro.

When I heard the learn'd astronomer,
When the proofs, the figures, were ranged in columns before me,
When I was shown the charts and diagrams, to add, divide, and measure them,
When I sitting heard the astronomer where he lectured with much applause in the lecture-room,
How soon unaccountable I became tired and sick,
Till rising and gliding out I wander'd off by myself,
In the mystical moist night-air, and from time to time,
Look'd up in perfect silence at the stars.

— Walt Whitman

Índice general

Introducción

1.1 Motivación

En los últimos años se ha observado una tendencia en muchas áreas de investigación hacia la generación de datos geométricos 3D a través de técnicas ópticas como el escaneo láser y el procesamiento de tales datos mediante algoritmos especializados.

En la literatura, los algoritmos para medir y procesar geometría 3D no son nuevos [25]. Sin embargo, debido a las restricciones de memoria y cómputo además de las propias de los sensores 3D, los métodos desarrollados en el pasado han encontrado una aplicación limitada en el uso cotidiano. Actualmente, podemos identificar muchos casos en donde los algoritmos basados en 3D están siendo utilizados para reemplazar sus paralelos basados en mediciones 2D. Uno de estos casos es el de la muy popular Microsoft Kinect [29], que recibió gran atención por parte de la comunidad científica por ser un sensor RGB-D de bajo costo que permite obtener video 3D en tiempo real.

En particular, motivado por una variedad de aplicaciones académicas e industriales, en los campos de la *visión por computadora* y la *computación gráfica* surge el problema de la *reconstrucción de superficies 3D*. En él, se desea obtener una representación computacional tridimnesional de objetos del mundo real de manera que se ajuste a distintos requerimientos de calidad y eficiencia impuestos por cada aplicación.

En muchos casos, una primera etapa del proceso de reconstrucción consiste en la obtención de una representación del objeto a reconstruir como una nube de puntos 3D (ejemplo en la figura 1.1). Algunos de los más populares a cargo de esta etapa, que denominamos *etapa de digitalización*, son el de la *luz estructurada*, el *escaneo láser*, la *visión estéreo* o *structure from motion (SFM)*. Una característica de estos métodos que es fundamental de reconocer y manejar es que, a menudo, generan nubes del orden del millón de puntos para objetos de variados tamaños y con gran cantidad de detalles. Asimismo, suelen producir *ruido* en las ubicaciones de los puntos, uniformidad variable en su densidad e imprecisiones como regiones con puntos faltantes.

En una segunda etapa, se procesan estas nubes de puntos mediante algoritmos diseñados para inferir la geometría 3D del objeto a reconstruir. Una información clave en el que suelen apoyarse estos algoritmos es la *orientación* de la nube. Esta información suele codificarse asociando a cada punto un vector unitario que determina la orientación del plano tangente al objeto en ese punto. Una nube que cuenta con estos vectores normales se denomina *nube de puntos orientada*. Algunos métodos de digitalización proveen una estimación de las normales asociadas a cada punto, mientras que otros sólo proveen coordenadas para los puntos. En el último caso, se acude a un algoritmo de estimación de normales que pasa a formar parte de un preprocesamiento necesario antes de la aplicación de algoritmos de la segunda etapa.

El resultado de este segundo paso, que denominamos *etapa de reconstrucción*, suele ser

un modelo tridimensional visualizable, a menudo representado por medio de una *malla poligonal*. Por el volumen de datos que pueden tener las nubes de entrada de estos algoritmos, procesarlas suele presentar importantes desafíos computacionales debido a las restricciones de tiempo de procesamiento y memoria que imponen la mayoría de las aplicaciones actuales y el hardware disponible para ellas. Por eso, se necesita de técnicas eficientes y a su vez precisas que permitan obtener reconstrucciones en tiempos aceptables y de buena calidad. Esta tesis se enfoca en estudiar y proponer un análisis detallado de técnicas de esta etapa, teniendo en consideración justamente los aspectos de eficiencia, calidad y los tradeoffs que puedan existir entre estas propiedades.

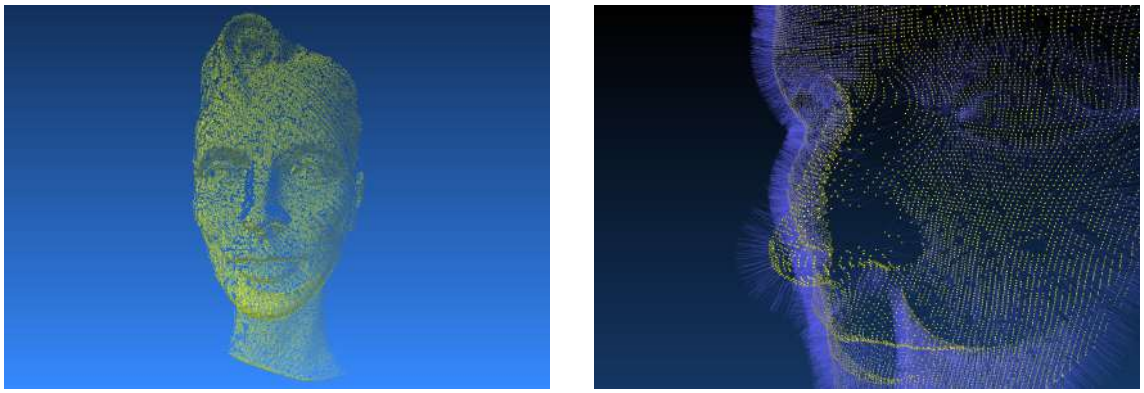


Fig. 1.1: Ejemplo de nube de puntos obtenida de un escaneo láser. Se puede observar fácilmente la no-uniformidad en la distribución de los puntos y regiones con datos faltantes. En la imagen de la derecha se visualizan normales asociadas a los puntos.

1.2 Aplicaciones

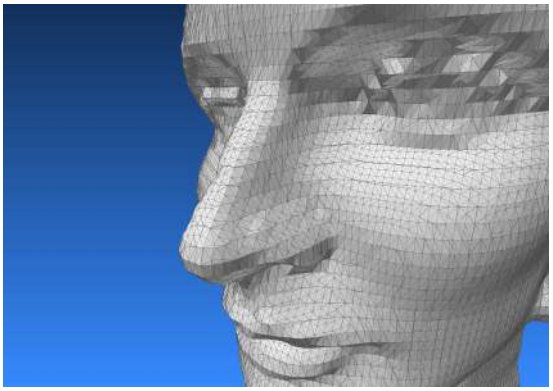
En la medicina existen variadas aplicaciones de visión computacional y reconstrucción. Un ejemplo de aplicación muy exitosa es la tomografía computada. En la misma se distinguen claramente las dos etapas de reconstrucción comentadas previamente: en primer lugar se obtienen datos geométricos crudos por medio de un escaneo rotativo basado en rayos X, y luego se procesan esos datos para producir la imagen que interpreta el personal médico.

Otra aplicación en medicina es la detección temprana y prevención de enfermedades motrices en personas. Por ejemplo, utilizar modelos tridimensionales de las manos de pacientes tomados instante a instante permite clasificar los patrones de movimiento en *patológicos* y *sanos* en base a una estimación matemática precisa de ubicación y desplazamiento.

Las aplicaciones industriales de métodos de digitalización y reconstrucción aparecen con mucho protagonismo también en ramas de la ingeniería. Reconstrucciones 3D precisas permiten a los ingenieros de producción corregir rápidamente errores de manufactura, lanzar productos a la fabricación en masa de manera más eficiente en tiempo y costos o llevar fácilmente objetos antiguos de los que no se dispone una información 3D digital a un entorno de diseño como CAD (*Computer-Aided Design*).

1.3 Taxonomía y antecedentes

Resolución de 100^3



Resolución de 100^3

Resolución de 500^3

Resolución de 500^3

Fig. 1.2: Dos mallas poligonales construidas con marching cubes con una grilla de $100 \times 100 \times 100$ en la primera y $500 \times 500 \times 500$ en la segunda. En las imágenes de la derecha de cada caso se visualiza el contorno de los triángulos de la malla para resaltar la diferencia de resolución.

La bibliografía en reconstrucción de superficies a partir de nubes de puntos es bastante extensa, y abarca más de dos décadas [10, 7]. A pesar de su larga historia, el área sigue siendo muy activa.

En muchos casos, los procesos de la etapa de reconstrucción consisten en un primer paso que computa una función implícita para la superficie a reconstruir, y etapas posteriores que completan la reconstrucción en base a esa función y producen la superficie propiamente dicha, por ejemplo en forma de malla poligonal. Un ejemplo de categoría de algoritmos que no sigue este esquema es aquella en la que reconstrucción se realiza utilizando directamente a todos o algunos de los puntos de entrada como vértices de polígonos de una malla poligonal [8]. Muchos de estos métodos son combinatorios por naturaleza [13], y algunos vienen con calidad de reconstrucción garantizada [3, 4, 12, 15].

Otra manera de categorizar a los métodos de reconstrucción es separar en algoritmos *interpolantes* y *aproximados*. Los del primer tipo producen superficies que interpolan los puntos de la nube de entrada. Los algoritmos mencionados anteriormente que utilizan a los propios puntos como vértices de una malla poligonal pertenecen claramente a este

conjunto. Un problema que presentan los algoritmos interpolantes es que la calidad de sus resultados está necesariamente afectada por errores en los puntos de entrada: una nube ruidosa, al utilizarse como entrada de un algoritmo interpolante, llevará naturalmente a una superficie ruidosa también.

Los algoritmos aproximados, por su parte, proveen a menudo mayor robustez ante tales errores de digitalización. Una analogía que ejemplifica esta distinción es la utilización de cuadrados mínimos para ajustar un determinado modelo lineal a mediciones hechas en un laboratorio. En el gráfico 1.3 se muestra un tal ejemplo en el que se ajusta a las mediciones un modelo lineal. Si en vez de aproximar los datos del gráfico de esa manera se intentara interpolarlos, quedaría un modelo caótico y demasiado acoplado al ruido producido naturalmente por las técnicas de medición. De manera análoga al ejemplo del gráfico, en algunos métodos aproximados de reconstrucción de superficies se realiza la aproximación ajustando funciones cuadráticas bivariadas, que producen superficies suaves, a subconjuntos de puntos de la nube. Esto mismo se hace en el algoritmo *MPU* ([24]) que se estudia en detalle más adelante.

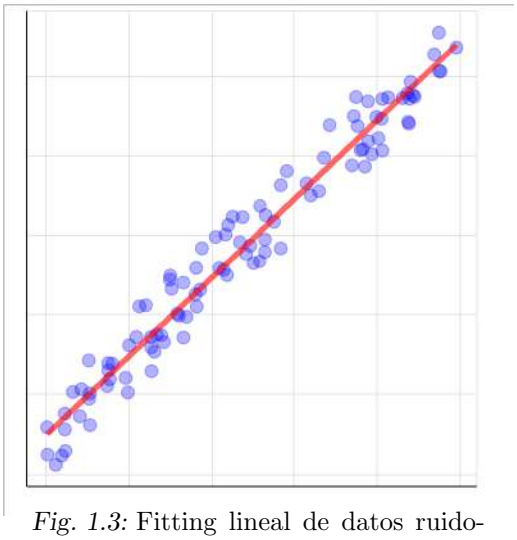


Fig. 1.3: Fitting lineal de datos ruidosos

En particular, la mayoría de los métodos aproximados de reconstrucción se hacen en base a una función implícita, ya que dicha forma de representación puede garantizar superficies suaves. Ejemplos de estos métodos incluyen [18, 19, 2, 23] que reconstruyen una función indicatriz binaria; [17, 9, 11] que estima una función de distancia con signo; y [1, 16, 24] basados en el ajuste y combinación de funciones locales, o *Moving Least Squares*. La mayoría de estos métodos reducen el problema a uno de optimización numérica que involucra resolver grandes sistemas lineales poco densos.

Por su parte, el algoritmo de reconstrucción de superficies de la *cápsula no-convexa Non-convex Hull (NCH)* [27] se encuentra entre dos de las categorías descriptas antes. Es un algoritmo interpolante, pero a diferencia de los interpolantes citados, su salida es una función de distancia con signo (*signed distance function*, o *SDF*) que puede ser consultada luego para la confección de la superficie final. Estas funciones son un caso particular de representación por función implícita que devuelven para cada punto de su dominio la distancia que presenta a la superficie; de esta manera, la superficie queda representada implícitamente como el conjunto de puntos para el cual vale 0.

El algoritmo *Multi-level Partition of Unity Implicits*, o MPU, propone un método aproximado que calcula también una SDF. El mismo calcula y combina funciones de soporte local para producir su SDF de salida. La combinación es realizada en base a la técnica de *partition of unity*, o *partición de la unidad*, utilizada a menudo en distintas áreas para extender construcciones locales a una construcción global. Para elegir las regiones donde computar las aproximaciones locales, el método divide el espacio que ocupa la nube de manera adaptativa: considera subdivisiones de mayor o menor granularidad de acuerdo a la complejidad geométrica que infiere en las distintas partes de la nube. Luego, en cada localidad, utiliza un método distinto de reconstrucción local de acuerdo a una serie de

tests que pretenden descubrir si la superficie en esa región presenta o no características "filosas" como bordes o esquinas. El algoritmo toma como parámetro un máximo error aceptable de aproximación y realiza esta adaptación en la subdivisión del espacio de modo de cumplir con dicho parámetro. De esta manera, el costo computacional del algoritmo varía de acuerdo a un parámetro de calidad aceptable de reconstrucción, produciendo subdivisiones más granulares y por ende mayor cantidad de reconstrucciones locales a medida que se exija un error más bajo.

En la mayoría de los casos, un primer resultado visualizable de una reconstrucción es representado como una malla poligonal. Tales mallas son calculadas por algoritmos llamados *algoritmos de poligonalización* que toman como entrada una función implícita resultante de algún otro método como los mencionados anteriormente. Dicha representación consiste en un conjunto de polígonos, conectados en sus vértices, que intentan aproximar la superficie del objeto a reconstruir. Una mayor concentración de polígonos generalmente permite una visualización más fiel del objeto y sus detalles finos (ver ejemplo en la figura 1.2). Esta característica de una malla poligonal se denomina su *resolución*.

En particular, cuando la función implícita a poligonalizar se trata de una SDF, el tipo de algoritmo de poligonalización adecuado es el de los métodos de *extracción de isosuperficies*. Este nombre proviene del hecho de que buscan determinar el conjunto de puntos del espacio para el cual la función implícita evalúa a una determinada constante. Cuando la implícita es una SDF, esta constante es 0. Un ejemplo de esta familia de algoritmos es Marching Cubes [22] que se utilizará a lo largo de este trabajo. Este algoritmo evalúa la SDF en los vértices de una grilla tridimensional regular que encierra el volumen ocupado por la nube de puntos original, y ubica polígonos en las celdas de la grilla de acuerdo al signo del valor devuelto por la SDF en cada vértice. Dado este esquema, es inmediato ver que cuanto mayor sea la resolución deseada de la malla, mayor cantidad de celdas deberá tener la grilla utilizada. En el contexto de un algoritmo tal, la resolución de la malla suele denotarse por las dimensiones de la grilla tridimensional usada. Dado que el costo de evaluar la función implícita en una grilla regular de alta resolución es a menudo muy elevada, algunos métodos utilizan estructuras volumétricas adaptativas tales como octrees [24, 19, 23, 11] que requieren algoritmos más complejos de creación de polígonos pero permiten realizar una cantidad mucho menor de evaluaciones de la SDF. Es importante notar que los algoritmos de cálculo de función implícita también juegan un papel fundamental en el costo computacional de la etapa de extracción de superficies, dado que los algoritmos de extracción de isosuperficies deben evaluar la función implícita en una cantidad habitualmente muy alta de puntos. Cuanto más eficiente sean la estructura y algoritmos utilizados para representar la función implícita y devolver su valor en puntos del espacio, más eficiente será la extracción final de la superficie.

1.4 Objetivos

El objetivo principal de esta tesis es el análisis, la implementación y la evaluación de métodos de reconstrucción de superficies 3D que toman nubes de puntos orientadas y producen SDFs. Elegimos en primer lugar dos algoritmos interesantes de estudiar, y por último proponemos un nuevo algoritmo que combina los enfoques de los primeros dos.

El primero es el método *Non-Convex Hull Surfaces (NCH)* [27] (sección 3), que propone una manera sencilla y precisa de calcular una SDF de la superficie a reconstruir. Analizamos la calidad de sus reconstrucción en función de distintos tipos de entrada, con-

trastando esos resultados con la teoría detrás del algoritmo (sección 3.6). Asimismo, el método da lugar a tres variantes distintas. Contrastamos estas variantes en detalle, presentado sus fortalezas y debilidades (secciones 3.5 y 3.7). Mostramos que el método tiene como desventaja su poca eficiencia computacional, teniendo una complejidad temporal cuadrática en la cantidad de puntos de entrada (3.4).

En gran parte, la eficiencia de NCH se ve aminorada por tratarse de un método de reconstrucción *global*, es decir, que considera al mismo tiempo la totalidad de la nube de puntos de entrada. En contraste, los métodos *locales* de reconstrucción parten la nube en localidades que son procesadas por separado y luego combinadas para dar una función implícita final de toda la superficie. El segundo algoritmo que estudiamos, el anteriormente mencionado Multi-level Partition of Unity Implicits [24] (sección 4), es un ejemplo de método local. Presentamos la teoría detrás de este algoritmo y algunos de sus resultados. Hacemos foco en el esquema de localización en el que se basa el método, enfoque que resulta pilar para el tercer algoritmo del trabajo (secciones 4.1 y 4.2). Además, proponemos una modificación sencilla y útil a uno de los criterios involucrados en dicho esquema (4.2.1). Por último, realizamos un análisis de los parámetros con los que puede configurarse el comportamiento de este algoritmo, proponiendo finalmente en base a nuestros resultados una elección de parámetros distinta a la sugerida por los autores del método (sección 4.7).

Por último, proponemos una extensión al método de NCH que utiliza un enfoque local inspirado en el de MPU con el fin de estimar de manera correcta y más eficiente la NCH de una nube de puntos (sección 5). Denominamos al algoritmo resultante de esta extensión *Partition of Unity Non-Convex Hull (PUNCH)*. Motivamos y explicamos la parametrización del algoritmo (5.1) y estudiamos su comportamiento en función de diferentes valores interesantes de sus parámetros (5.3), basándonos en un estudio de sus tiempos de ejecución y en un framework particular de evaluación cualitativa de calidad de reconstrucción. Como resultado de este análisis, proponemos una heurística para configurar el algoritmo en función de cada entrada a la que se lo desee aplicar.

Fundamentos

En este capítulo se abordan algunos de los fundamentos teóricos y prácticos que se tratan en el resto del trabajo. Entre ellos se presentan los frameworks de experimentación en los que se va a basar la evaluación de los algoritmos que se estudian.

2.1 Reconstrucción 3D

En el capítulo anterior se presentó el concepto de reconstrucción, algunas de sus etapas, aplicaciones y antecedentes de algoritmos relacionados. En particular, se detalló una taxonomía para métodos de reoconstrucción.

A lo largo de este trabajo nos enfocamos principalmente en una etapa específica del proceso de reconstrucción: aquella que parte de una nube de puntos y produce una representación computacional de la superficie que se intenta reconstruir. Específicamente, se estudian algoritmos que toman nubes de puntos *orientadas* y devuelven SDFs. Para poder visualizar y evaluar los resultados de estos algoritmos, utilizamos *marching cubes* para extraer una superficie concreta de la función implícita devuelta en cada corrida. En cada ocasión donde se busque comparar entre resultados de algoritmos de función implícita, se utilizará siempre marching cubes con la misma resolución.

Dado que el nombre "proceso de reconstrucción" puede dar a entender que se trata de todos los pasos desde la digitialización de un objeto real hasta la obtención de una malla, por claridad, nos referiremos a cada etapa por su nombre, específicamente la de cálculo de SDF y la de extracción de poligonalización. Dado que las únicas dos etapas tratadas en este texto son estas dos, cuando hablemos de "reconstrucción" usualmente se tratará de ambos pasos.

A continuación definimos algunos conceptos relacionados a la reconstrucción 3D que aparecerán recurrentemente en lo que sigue del texto.

Nube de puntos. Definimos una nube de puntos P sencillamente como un conjunto de puntos del espacio euclídeo tridimensional representad representados por sus coordenadas cartesianas: $P \subset \mathbb{R}^3$.

Nube de puntos orientada. En una nube de puntos orientada se le asocia un vector unitario a cada punto de una nube de puntos. Dicho vector, denominado la *normal* del punto, representa la orientación de la superficie en el sentido de ser normal al plano tangente a la superficie en el punto asociado. En nuestra notación, de manera consistente con la mayoría de los papers originales de los algoritmos citados, representamos una nube de puntos orientada definiendo por un lado una nube de puntos como un conjunto ordenado $P = \{p_1, \ldots, p_n\}$, y por otro definiendo un conjunto de normales $N = \{n_1, \ldots, n_n\}$ que se corresponden uno-a-uno a los elementos de P.

Caja delimitadora. Una *caja delimitadora* de una nube de puntos, del inglés *bounding box*, es un prisma rectangular que contiene a todos los puntos de la nube. Es un elemento básico de muchísimos algoritmos relacionados al procesamiento de datos 3D que da información de la escala de la nube y permite definir la región donde se hará el procesamiento de los datos. Suele darse por supuesto que la caja delimitadora de una nube contiene a sus puntos de manera ajustada, i.e minimizando la distancia entre los puntos y las caras de la caja.

Cubo delimitador. Un *cubo delimitador* de una nube de puntos es una caja delimitadora cúbica de esa nube.

Nube de puntos ruidosa. Las nubes de puntos que se calculan por medios de digitialización como el escaneo láser, luz estructurada o SFM, suele tener algo de ruido de medición como es inevitable al realizar mediciones por medios ópticos de un objeto real. Decimos que una nube es "ruidosa" cuando presenta el tipo de error que puede originarse en mediciones de este tipo. Tales nubes habitualmente van a incluir ruido tanto en la ubicación de sus punto como en la dirección de las normales asociadas a ellos. Tiene sentido hablar de una nube *sin* ruido realmente sólo cuando se trata de una nube generada sintéticamente mediante un programa hecho con ese propósito. A lo largo del trabajo trabajamos con ambos tipos de nubes para analizar el comportamiento de los distintos algoritmos.

Densidad de una nube de puntos. La densidad de una nube de puntos da una medida de cuán cerca se encuentran los puntos de la nube entre sí. Las técnicas de digitalización a menudo ofrecen la posibilidad de producir resultados más o menos densos. Asimismo, existen técnicas de *simplificación* de nubes de puntos que pretenden reducir la densidad de una nube dada perdiendo la menor cantidad posible de información útil para una reconstrucción posterior. Dada una caja delimitadora de una nube de puntos, la densidad es sencillamente proporcional a la cantidad de puntos de la nube.

Representación por función implícita. Matemáticamente, una superficie tridimensional S puede quedar unívocamente definida mediante una *función implícita* $f : \mathbb{R}^3 \to \mathbb{R}$ de la siguiente manera: $S = \{x \in \mathbb{R}^3 : f(x) = y_0\}$ para un y_0 fijo. Es decir, dada f y fijando un valor real y_0, podemos entender a S como el conjunto de todos aquellos puntos del espacio donde f vale y_0. Se dice que esta función define implícitamente a S en el sentido de que no provee una expresión paramétrica que permite conocer sus puntos. Por ejemplo, podemos considerar la función $f(x, y, z) = x^2 + y^2 + z^2$. Tomando $y_0 = 1$, f representa la esfera de radio 1 centrada en $(0, 0, 0)$. Contando sólo con la definición de f, a menos que se conozca de antemano su definición o la superficie que representa implícitamente, es imposible saber qué puntos conforman la superficie sin realizar un análisis que permita concluir a través de qué conjunto de puntos f vale y_0. En cambio, una representación paramétrica de la misma esfera mediante $f' : \mathbb{R}^2 \to \mathbb{R}^3$, $f'(\theta, \phi) = (cos\theta sin\phi, sin\theta sin\phi, cos\phi), 0 \leq \theta \leq 2\pi, 0 \leq \phi \leq \pi$ permite conocer explícitamente puntos de la esfera evaluando a f' en cualquiera de los valores válidos de sus argumentos.

Signed-distance function (SDF). Una función de distancia con signo, o SDF por las siglas de su nombre en inglés *signed-distance function*, es una función que determina implícitamente una superficie. Mapeando $\mathbb{R}^3$ a $\mathbb{R}$ como corresponde según la definición anterior, devuelve para cada punto del espacio un real cuyo valor absoluto es la distancia del punto a la superficie representada. Considerando que un punto está en la superficie si y sólo si su distancia a la misma es 0, la superficie representada por f es la que corresponde a tomar $y_0 = 0$ considerando la definición anterior. Además, garantiza una propiedad de

gran utilidad: devuelve valores con distinto signo según si el punto en el que se evalúa se encuentra "de un lado" de la superficie o del otro. En este sentido, la SDF divide al espacio en tres conjuntos de puntos: la superficie en sí, donde vale 0, la región donde es positiva y la región donde toma valores negativos. Para cuerpos cerrados, esto se corresponde con definir unívocamente un interior y un exterior para el cuerpo en cuestión. Esta propiedad es fundamental para la extracción de isosuperficies por medio de marching cubes dado que dicho algoritmo se basa en la diferencia de signo entre distintas evaluación de la función en vértices de su grilla para inferir dónde se encuentra la superficie. Por convención e intuición, generalmente, se define la SDF de manera que tome valores positivos fuera del cuerpo y negativos dentro del mismo.

Isosuperficie. El término se refiere simplemente a la superficie representada como el conjunto de puntos que evalúa a un y_0 fijo a través de una función implícita $f : \mathbb{R}^3 \to \mathbb{R}$, viniendo del *iso* en latín que significa "igual". En ocasiones nos referiremos al proceso de extracción de isosuperficies como *isosurfacing*.

Malla poligonal. Consiste en un conjunto de polígonos conectados por sus vértices, produciendo la visualización de una superficie. Puede tener una mayor o menor *resolución* según la densidad de polígonos que se utilicen. En nuestro contexto, se utiliza como forma de visualizar la superficie definida por una función implícita ya calculada. En particular, el mencionado algoritmo de extracción de isosuperficies *marching cubes* da como resultado una malla poligonal con una resolución arbitraria. Cuanto mayor sea la resolución, más fiel será la visualización de detalles finos de la superficie, pero mayor será el costo computacional requerido para obtenerla. Además, una malla extraída por un algoritmo de isosuperficies es el primer elemento del proceso de reconstrucción que constituye efectivamente un modelo 3D del objeto reconstruido, que es el objetivo del proceso. Una vez obtenido, permite no sólo su visualización, sino que es la base de cualquier post-procesamiento que se desee hacer, desde cálculos que analizan la geometría del objeto hasta una texturización o coloración del mismo.

Artefactos de reconstrucción. Es común que los métodos de reconstrucción produzcan *artefactos*, provenientes en la gran mayoría de los casos de ruido en la nube de entrada. La forma de estos artefactos, su cantidad y su detrimento de la calidad final varía en base a cada algoritmo.

2.2 Frameworks de evaluación

Los algoritmos de reconstrucción, especialemente aquellos que producen como resultado una función implícita, presentan desafíos a la hora de ser evaluados en términos de la calidad de sus reconstrucciones. En general, podemos dividir las formas de evaluar la calidad de reconstrucción de los algoritmos en dos grandes familias: la cualitativa y la cuantitativa.

El primer tipo es el más común y accesible de realizar. Se basa esencialmente en obtener reconstrucciones del algoritmo evaluado y observar con detenimiento esos resultados con el foco puesto en evaluar ciertas propiedades de interés. Algunas de estas pueden ser la representatividad general que presenta la superficie analizada del objeto que debería capturar cuando este se conoce de antemano, la *pleasantness* general a la vista de la superficie resultante, su *intuitividad*, el ruido que haya heredado de un input ruidoso o artefactos que se hayan generado. Parte del desafío del análisis cualitativo reside en la

elección de nubes cuyas reconstrucciones se van a evaluar, cuidando de seleccionar casos interesantes de acuerdo a cada algoritmo.

El segundo tipo es el más complejo de realizar. En principio, requiere de alguna manera definir métricas y algoritmos que permitan evaluarlas. Además, es deseable poder hacerlo comparando los resultados analizados contra un modelo 3D que haga las veces de *ground truth*. Para ellos, se requeriría tener efectivamente tanto un modelo de ground truth como datos realistas que correspondan a esa misma figura y permitan llevar a cabo esa comparación. El framework de evaluación que se propone en [6], denominado *Reconbench*, ofrece exactamente esto.

Por último, una propiedad fundamental a evaluar en un algoritmo de reconstrucción es su eficiencia temporal y espacial. En este trabajo nos concentraremos especialmente en los tiempos de ejecución de los algoritmos, que son el aspecto de su eficiencia que más desafíos presenta. En particular, como se ha mencionado, veremos que uno de los algoritmos presentados tiene una complejidad temporal prohibitiva y presentaremos una extensión del mismo que se propone con el fin de mejorar sus tiempos de ejecución.

2.2.1 Metro

En [14] se propone una herramienta de comparación cuantitativa entre mallas denominada *Metro*. La comparación se basa en el concepto de *distancias punto-a-superficie*. Dadas dos mallas a comparar M_1 y M_2, se elige una de las mallas, por ejemplo M_1, como pivote. Se realiza un sampleo de puntos de M_1 y se calcula, para cada punto p del sampleo, la distancia punto-a-superficie de p a M_2. Dicha distancia se define como la mínima distancia euclídea de p a puntos de M_2. Tomando el máximo entre las distancias punto-a-superficie con M_1 como pivote, se obtiene la *distancia unilateral* de M_1 a M_2. Lo análogo puede hacerse tomando a M_2 como pivote y calculando tanto las distancias punto-a-superficie de los puntos de un sampleo de M_2 como la distancia unilateral de M_2 a M_1. Tomando el máximo entre las dos distancias unilaterales, se obtiene la métrica denominada *distancia de Hausdorff* que se utilizará en algunos análisis a lo largo de esta tesis.

Metro provee una serie de herramientas de comparación entre dos mallas en base a las distancias punto-a-superficie de una malla hacia la otra y las distancias unilaterales. De esas herramientas, a lo largo del trabajo utilizamos principalmente una. Tomando una de las mallas como pivote y las distancias punto-a-superficie correspondientes a un sampleo de la misma, Metro nos da una *comparación por superposición* que mapea dichas distancias a una escala de colores (en nuestro caso la RGB) y produce una nueva malla texturizada con los colores correspondientes. Más específicamente, la malla resultante tiene un polígono por cada polígono de la malla pivote, y cada uno de ellos se colorea de acuerdo al promedio de las distancias punto-a-superficie que presentan los puntos del sampleo de la malla pivote que caen dentro de ese polígono.

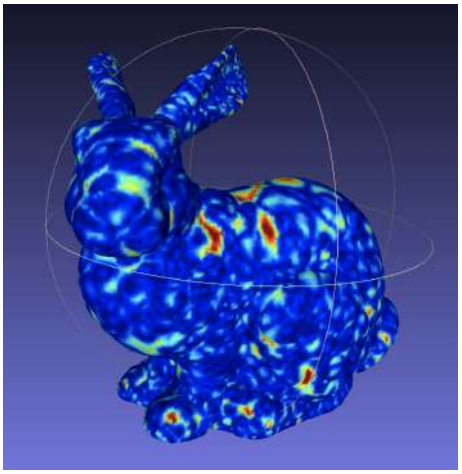


Fig. 2.1: Mapeo de distancias entre mallas a RGB

Es importante destacar que Metro surge para un uso específico muy distinto al nuestro. Originalmente, la herramienta se motiva en el problema de simplificación de mallas, en el que se busca reducir la cantidad de polígonos de una malla eliminando y combinándolos de manera de conservar lo más posible la forma original. En ese contexto, Metro se propone como manera de poder comparar entre la malla poligonal original y la simplificada, dando métricas cuantitativas de cuánta información se perdió en un proceso de simplificación y proveyendo así una manera de ajustar el algoritmo de simplificación a los requerimientos de precisión de cada aplicación. En cambio, en nuestro caso, lo que nos interesa es evaluar un algoritmo de reconstrucción. En este contexto, entonces, sólo existe una malla en cada ejecución del algoritmo estudiado, que es la que captura la reconstrucción realizada. Por ende, Metro nos sirve más en los casos donde contemos con un modelo de ground truth o cuando nos interese simplemente comparar entre reconstrucciones de distintos algoritmos.

Vale remarcar también que, como puede suponerse en base a las definiciones dadas arriba, la forma de sampleo de la malla pivote es determinante para la evaluación. Metro provee distintas estrategias que permiten obtener un sampleo de cierta densidad y que cumpla con ciertas propiedades deseables. Sin embargo, es importante notar que, denso como sea el sampleo, al hacerse sobre una superficie ya poligonalizada es inevitable el fenómeno de "facetización" del sampleo: todos los puntos que se muestrean de un mismo polígono resultarán necesariamente co-planares cuando en el objeto de la vida real, en caso de que tal objeto exista, pueden no serlo. Esta es una limitación de este framework inherente al hecho de basarse en la comparación de dos mallas, nuevamente un hecho explicado por el contexto de simplificación de mallas en el que se motiva Metro originalmente.

2.2.2 Reconbench

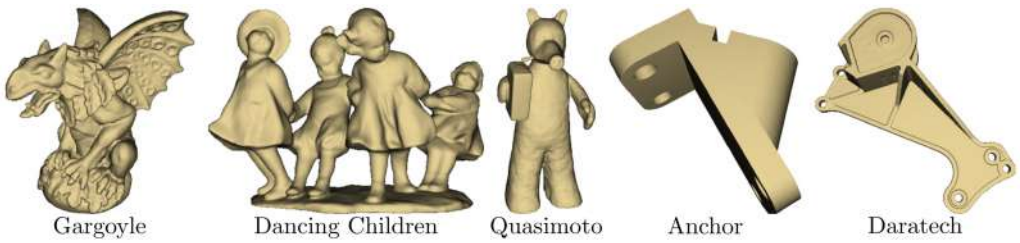


Fig. 2.2: Figuras del conjunto de datos de Reconbench con distintas características interesantes de evaluación. Fuente: [6]

En [6] se plantea un framework de evaluación cuantitativa de algoritmos de reconstrucción que producen funciones implícitas. El framework ofrece 5 figuras de test (figura 2.2), elegidas por los autores del paper de modo de formar un conjunto de figuras que reúna varias de las características que consideran interesantes en un input para estudiar este tipo de algoritmos. En más detalle:

- Gargoyle: detalles finos de distintos tamaños como las garras y las crestas en las alas. En promedio $\approx 79k$ puntos.
- Dancing children: topología no-trivial, incluyendo túneles de distintos tamaños y detalles como el borde del sombrero y las arrugas en la ropa. En promedio $\approx 62k$ puntos.

- Quasimoto: representa formas con partes articuladas, como las piernas los brazos y la cabeza en este caso. En promedio $\approx 47k$ puntos.
- Anchor: rasgos filosos, túneles de tamaño medio y una concavidad profunda. En promedio $\approx 67k$ puntos.
- Daratech: rasgos filosos, túneles pequeños y láminas de sueprficie delgadas. En promedio $\approx 49k$ puntos.

Además, las 5 figuras pueden clasificarse en dos clases útiles: los modelos Gargoyle, Dancing Children y Quasimoto constituyen superficies de rasgos mayoritariamente suaves, mientras que Anchor y Daratech representarán los modelos con rasgos filosos.

Para cada una de estas figuras el framework cuenta con un modelo de ground truth. A diferencia de otros frameworks de evaluación, en este caso el modelo de ground truth se mantiene como función implícita suave, y no se utiliza una malla de ningún tipo para él. Evitar utilizar una malla poligonal como ground truth tiene varias ventajas, sacando del camino los errores resultantes de representar características suaves de superficies mediante polígonos que los "aplanan". En particular, de manera parecida a Metro, Reconbench también se basa en un sampleo de puntos de una superficie para realizar comparaciones; sin embargo, en este caso la superficie sampleada es la de ground truth que está capturada en una función implícita suave. Por esto, el sampleo no puede sufrir del problema de facetización que existe en Metro. En otras palabras, al guardar el modelo de ground truth como función implícita suave, se puede comparar cada resultado del algoritmo evaluado con un modelo de ground truth suave, lo que hace que el proceso se parezca lo más posible a comparar esa malla con un objeto de la vida real.

Para obtener resultados de cada algoritmo evaluado que puedan ser comparados con los modelos de ground truth, el framework provee para cada figura un conjunto de nubes de puntos de test distintas. El conjunto de test para cada figura se calcula por parte de los autores del framework realizando simulaciones de escaneo 3D aplicadas sobre el modelo de ground truth. En este sentido, contar con una función implícita como ground truth ofrece una nueva ventaja: un escaneo real se aplica sobre la superficie suave de un objeto real; entonces, es mucho más realista correr una simulación de escaneo sobre el modelo suave capturado por la función implícita de ground truth que hacerlo usando una malla poligonal. De este modo, el conjunto de prueba para cada figura ofrece datos realistas, teniendo las nubes incluídas en él propiedades que es común encontrar en nubes obtenidas por métodos de digitalización: ruido en los puntos, ruido en las normales, uniformidad variable en la distribución de los puntos y datos faltantes.

Además de proveer datos de prueba, Reconbench plantea un pipeline de evaluación cuantitativa de la calidad de reconstrucción de un algoritmo en base a las nubes de prueba que proporciona y ofrece programas para ejecutarlo.

Pipeline de evaluación

En primer lugar, se elige una de las figuras y un subconjunto de sus nubes de prueba. Se corre el algoritmo a evaluar para cada nube, produciendo una malla poligonal en cada corrida. Después, se calculan errores de reconstrucción para cada malla usando tres métricas específicas que se detallan luego. Por último, en base a los valores de esas métricas computados para cada una de las mallas, se calcula la distribución agregada de valores

de error de cada métrica a lo largo del conjunto entero de nubes utilizado. De esta forma, dada una figura en particular, se obtiene un panorama mucho más representativo del rendimiento del algoritmo ante esa figura y sus caracetrísticas geométricas contemplando los errores que produjo para una variedad de nubes en vez de medir esos errores para una sola.

Tomando una misma figura, puede realizarse este pipeline con algún subconjunto de sus nubes de prueba utilizando una serie de algoritmos y realizar una comparación cuantitativa entre ellos en igualdad de condiciones. Este esquema constituye la forma de cabecera de realizar comparaciones cuantitativas entre algoritmos en lo que sigue este trabajo.

Métricas de error

Dada una malla M generada por el algoritmo evaluado y la función implícita de ground truth G, el framework construye lo que denomina *mapas de mínima distancia*. Tomando un sampleo denso y uniforme S de G, que incluye tanto puntos como normales asociados a los mismos calculados en base a la función G, se establece un mapa entre los puntos S y sus puntos más cercanos en el sentido euclídeo en M. En base a esa correspondencia se calculan valores para las tres métricas de error principales. Esto es similar al funcionamiento de Metro, con la diferencia notable de que en este caso se samplea sólo una de las superficies y la misma está en forma de función implícita y no como malla poligonal.

Distancia promedio. Para cada par de puntos de la correspondencia entre M y G se puede calcular la distancia euclídea que los separa. En esta métrica se toma el promedio de todas esas distancias. Para esta métrica principalmente es que tiene importancia la uniformidad del sampleo S además de su densidad, ya que es importante para lograr un promedio representativo de las distancias.

Distancia de Hausdorff. De acuerdo a la definición anterior de esta métrica, la *distancia de Hausdorff* aquí se calcula tomando el máximo de las distancias entre pares de puntos de la correspondencia entre M y G.

Desvío de ángulos promedio. En teoría, estableciendo una manera de calcular una normal para los puntos de cada par del mapa de mínima distancia de M y G, podría calcularse sencillamente el ángulo formado entre las dos normales para obtener una medida de desviación en cada par. En Reconbench, esto se hace tomando la normal sampleada para S del lado del modelo de ground truth, y calculando la normal al polígono que contiene el punto correspondiente en la malla poligonal M.

En resumen, dada una figura de las propuestas por el framework y un algoritmo a evaluar, Reconbench nos provee una manera de obtener, en base a tres métricas particulares, la distribución de errores que presenta el algoritmo ante una serie de nubes con ruido realista correspondientes a la figura elegida. Corriendo un conjunto de algoritmos sobre una misma serie de nubes de prueba se puede realizar una comparación cuantitativa de calidad de reconstrucción entre esos algoritmos.

Non-Convex Hull Surfaces

En [27], se propone un método de reconstrucción que produce una SDF interpolante. Este método se denomina *el método de la cápsula no-convexa* y se abrevia *NCH* por su nombre en inglés. La principal virtud de este método es su simpleza, pudiendo implementarse de manera sencilla y con pocas líneas de código. Además, asumiendo una nube de puntos de entrada sin ruido o con ruido leve, lleva a mallas de alta calidad. Tiene como desventaja principal su complejidad temporal que es cuadrática en la etapa de estimación y lineal en la de evaluación.

Como construcción inicial, el trabajo define la cápsula *convexa* de una nube de puntos orientada, y luego refina esta definición para dar lugar a la de la cápsula *no-convexa*.

3.1 Cápsula convexa orientada

Tomando una función $f(x)$ continua en un dominio $U \subseteq \mathbb{R}^3$, se define un *semiespacio* H como $H = \{x \mid f(x) \leq 0\}$. En este contexto, decimos que f es la *función de borde* de H. Un semiespacio se dice *de soporte* de un conjunto de puntos P si cumple que todo punto de P pertenece a él y al menos uno de ellos evalúa a 0 a través de f, es decir, se encuentra "sobre el borde" de H. En particular, tal semiespacio de soporte se denomina *lineal* cuando f es lineal.

Dada una nube de puntos $P = \{p_1, ..., p_N\}$ orientada con normales unitarias $N = \{n_1, ..., n_N\}$, el trabajo define para cada punto p_i un semiespacio de soporte lineal $H_i = \{x \mid f_i(x) \leq 0\}$ usando la función $f_i(x) = n_i^t(x - p_i)$. Considerando el punto p_i, esta definición de H_i esencialmente divide el espacio en una mitad que contiene a todo P y otra que no contiene ningún punto de P, trazando una línea que atraviesa a p_i.

La cápsula convexa orientada (*Oriented Convex Hull*) de P se define como $OCH(P) = \bigcap_i H_i$. Como cada H_i es convexo y la intersección conserva esta propiedad, $OCH(P)$ es también convexo.

Como aproximación de una superficie, la OCH de una nube de puntos tiene la natural desventaja de no poder representar concavidades que la misma podría tener, al tratarse de un conjunto convexo. Sin embargo, si se extiende la construcción anterior de manera de incluir semiespacios de soporte no-convexos, esta limitación se podría superar. Esta extensión se puede hacer de manera sencilla.

3.2 Cápsula no-convexa

Consideramos un reemplazo de las funciones de borde por

$$f_i(x) = \frac{1}{2r}(r^2 - \|x - (p_i + rn_i)\|^2) \tag{3.1}$$

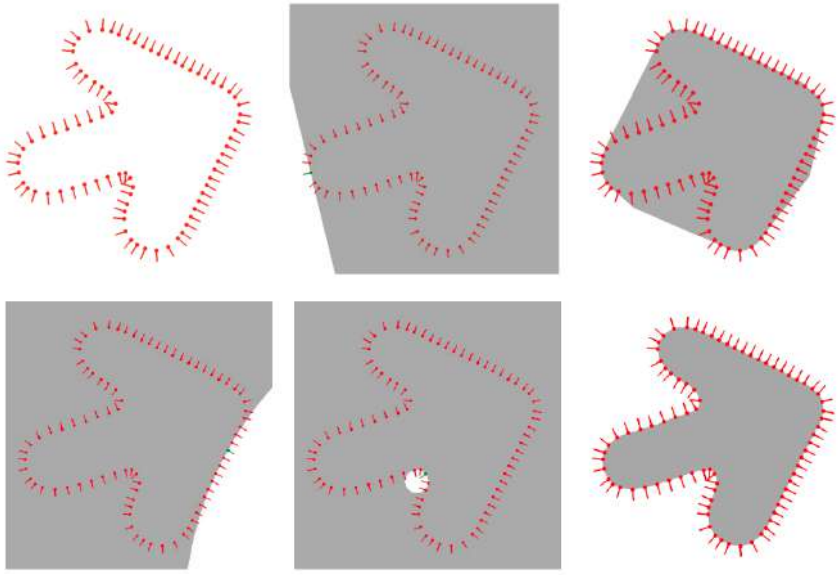


Fig. 3.1: Cápsula convexa orientada y cápsula no-convexa. En gris se marcan valores negativos de las funciones de borde. Primera fila: una nube de puntos orientada, el semiespacio lineal de uno de los puntos y la intersección de todos ellos. Segunda fila: dos ejemplos de semiespacios de soporte de complemento esférico y la intersección de todos ellos. Imagen tomada de [27].

En esta definición, la función vale 0 en la frontera de la esfera de radio r_i centrada en el punto $p_i + r_i n_i$, es negativa fuera de esa esfera, positiva dentro de la misma y toma su máximo valor en su centro. Por construcción, entonces, $f_i(p_i) = 0$ dado que p_i dista r_i de $p_i + r_i n_i$ en dirección de n_i y por ende yace sobre la esfera. Si además elegimos r_i de modo que el tamaño de la esfera sea lo suficientemente chico para que todo $p_j \in P, j \neq i$ quede sobre el borde o fuera de la esfera, i.e $f_i(p_j) \leq 0$, tenemos que H_i definido con la nueva f_i es un semiespacio de soporte de P. Eligiendo el máximo r_i que cumple esta propiedad, el trabajo denomina a las H_i *semiespacios de soporte de complemento esférico.*

Contando con esta nueva construcción de semiespacios de soporte, se define a la cápsula no-convexa como $NCH(P) = \bigcap_i H_i$. En esta definición, el conjunto resultante sí es capaz de capturar las concavidades que pueda tener el cuerpo a reconstruir. El contraste en la capacidad de capturar concavidades entre la cápsula convexa y la no-convexa se muestra con un ejemplo en la figura 3.1. Notemos que, permitiendo $r = \infty$, la definición de esta nueva familia de semiespacios de soporte puede incluir también semiespacios lineales como los utilizados para la cápsula convexa, que pueden resultar útiles para representar mejor algunas partes del cuerpo en cuestión.

3.3 Función de distancia con signo de la NCH

Dada la definición de la NCH como la intersección de los semiespacios de complemento esférico, podemos pensar a la NCH como un semiespacio en su propio derecho que divide al dominio en esa intersección por un lado y la unión de las esferas de radios r_i, el complemento, por otro. De esta manera, el borde de este semiespacio compuesto está formado por sectores de esas esferas. Este semiespacio, al igual que los definidos hasta ahora, también puede expresarse por medio de una función de borde. Esta función es

$$f(x) = \max_i f_i(x)$$

y podemos definir la NCH de P como $NCH(P) = \{x \mid \max_i f_i(x) \leq 0\}$. De esta manera, esta función hace las veces de SDF para la NCH de P. En el contexto de esta definición, las funciones de borde f_i asociadas a cada punto p_i son las que el trabajo original denomina las *basis functions* de la NCH.

Es relevante destacar que el signo que tomen puntos del espacio que se encuentran en una región o la otra de la división dada por f depende de la orientación de las normales. Ante un volteo de la orientación de las normales, se invertirá el signo del interior y el exterior de la superficie dada por la SDF (ver figura 3.2). Más aún, la superficie resultante de la reconstrucción cambiará ya que los semiespacios utilizados serán estrictamente distintos, y por ende se produciría una SDF distinta. En cualquier caso, la reconstrucción sigue correspondiendo a la definición $S = \{x \mid f(x) = 0\}$ que corresponde a la estimación de la superficie de nivel 0.

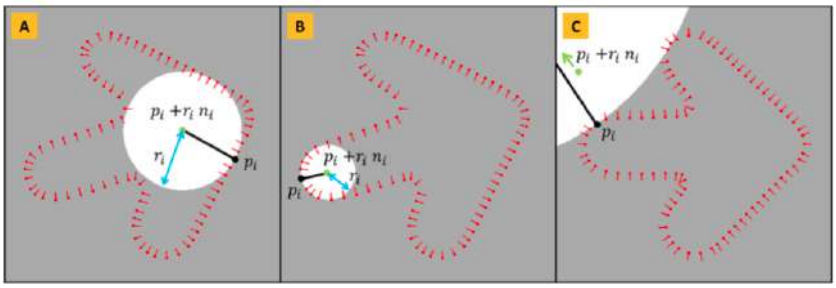


Fig. 3.2: Semiespacios de soporte de complemento esférico. En A y B las normales de la nube están orientadas hacia el interior de la figura y la región negativa queda fuera. En C las normales se orientan al revés y la región negativa queda dentro de la figura. Imagen tomada de [27].

Para proveer una mayor intuición de la definición de la SDF como $f(x) = \max_i f_i(x)$, podemos considerar algunos casos particulares. Supongamos que las normales están orientadas "hacia afuera"del cuerpo a reconstruir, de manera que la SDF toma valores negativos dentro de la misma y positivos fuera de ella. Si evaluamos la SDF en un punto x_0 y obtenemos un valor negativo y_0, sabemos que ninguna f_i evaluó x_0 a un valor positivo ya que, por construcción, y_0 es el máximo entre todos esos valores. Entonces, sabemos que x_0 se encuentra del lado negativo de los N semiespacios utilizados y por ende debe yacer en el interior de la figura. En cambio, razonando de manera similar, si obtenemos un y_0 positivo sabemos que existe un p_i en P para el cual x_0 se encuentra en la región positiva de su semiespacio. Equivalentemente, sabemos que se encuentra dentro de la esfera definida por la i-ésima función de borde. En caso tal, corresponde que la SDF evalúe a un valor positivo en x_0 ya que alcanza con estar del lado positivo de un solo semiespacio de los N definidos para saber que el punto no está dentro de la figura.

3.4 Algoritmo de la NCH

Dada una nube de puntos de entrada, se desea computar la SDF definida en la ecuación 3.3 para poder obtener su valor en puntos arbitrarios del dominio. Por construcción, evaluarla en un punto particular p requiere tomar el máximo entre evaluaciones de las $f_i, i = 1 \ldots N$ en ese punto. A su vez, como se puede apreciar en la definición de f_i en la

ecuación 3.1, lo único que se precisa para poder evaluar cada una es hallar en primer lugar su correspondiente radio r_i.

En [27] se propone una reescritura de la definición de f_i que resulta muy conveniente para esta tarea:

$$f_i(x) = n_i^t(x - p_i) - \rho_i \|x - p_i\|^2$$

donde $\rho_i = \frac{1}{2r_i}$. Esta expresión equivalente pone la definición de f_i en términos de la incógnita ρ_i, que se puede encontrar mediante una simple búsqueda de máximo:

$$\rho_i = \begin{cases} 0 & \text{si } J_i = \emptyset \\ \text{máx}\{\frac{n_i^t(p_j - p_i)}{\|p_j - p_i\|^2} \mid j \in J_i\} & \text{caso contrario} \end{cases}$$

donde J_i es el conjunto de índices $j = 1, ..., N$ tal que $n_i^t(p_j - p_i) > 0$.

El signo de $n_i^t(p_j - p_i)$ determina la posición del j-ésimo punto de la nube respecto del plano posicionado en p_i con normal n_i: si da negativo, quiere decir que p_j se encuentra "detrás del punto". Si todos los otros puntos de la nube presentan este posicionamiento, poner $\rho_i = 0$, o equivalentemente $r_i = \infty$, corresponde a elegir para p_i un semiespacio de soporte lineal que coincide con el plano mencionado. Esto es apropiado ya que el radio infinito es en efecto el máximo radio para el cual $f_i(p_j) \leq 0$ para todo j, de acuerdo a la definición de $NCH(P)$.

De lo contrario, si existen puntos $p_j, j \neq i$ "delante"de p_i según el plano con normal n_i, el radio r_i tendrá que ser el valor finito máximo entre los que dejan a todos los otros p_j dentro del semiespacio complementario a la i-ésima esfera. Este máximo r_i se encuentra minimizando el valor de ρ_i que le es inversamente proporcional.

Vale destacar como virtud de este método su naturaleza analítica. Por medio de un planteo algebraico, logra contrastar positivamente con los procedimientos iterativos de minimización de error usualmente empleados en el cálculo de parámetros en este tipo de algoritmos. Esto le otorga sencillez y mayor robustez como algoritmo.

El siguiente pseudocódigo resume el algoritmo recién expuesto y permite su implementación casi directa en cualquier lenguaje de programación estándar:

estimarNCH():
  para i en $1, .., N$ **hacer**:
    $\rho_i \leftarrow 0$
    para j en $1, ..., N$ **hacer**:
      si $j \neq i$ **entonces**:
        $a \leftarrow n_i^t(p_j - p_i)$
        $b \leftarrow \|p_j - p_i\|^2$
        si $a - \rho_i b > 0$ **entonces**:
          $\rho_i \leftarrow a/b$

evaluarNCH(x):
  $f_x \leftarrow -\infty$
  para i en $1, ..., N$ **hacer**:
    $a \leftarrow n_i^t(x - p_i)$
    $b \leftarrow \|x - p_i\|^2$
    $c \leftarrow a - \rho_i b$
    si $c > f_x$ **entonces**:
      $f_x \leftarrow c$
  devolver f_x

El procedimiento **estimarNCH** se encarga de calcular los ρ_i, obteniendo así toda la información necesaria para luego evaluar la SDF resultante. Notar que es visiblemente cuadrático en el tamaño de la entrada, dando una complejidad de $\Theta(N^2)$. Luego, en $evaluarNCH$ se busca el máximo valor entre las evaluaciones de las f_i en el argumento x. Este último procedimiento es claramente $\Theta(N)$.

3.5 Orientación de las normales y variante simétrica

Como se ha mencionado, tener las normales de la entrada apuntando hacia un lado o el opuesto cambia el resultado de la reconstrucción. Como puede apreciarse en la figura 3.2, las características de los semiespacios que se utilizan para calcular la SDF cambian en cada caso de orientación de las normales. Cuando las mismas apuntan hacia el interior de la figura, las esferas están limitadas por este interior y el mismo queda definido como la unión de N esferas "chicas". En el caso inverso, el exterior queda definido por la unión de N esferas "grandes".

Estas características de cada orientación en particular pueden beneficiar algunos rasgos de un cuerpo a reconstruir y perjudicar otros. Este fenómeno se expone en más detalle en la siguiente sección. Para contar con un término medio, es posible definir una variante "simétrica" de la SDF que tiene en cuenta esferas construidas tanto de un lado como del otro del cero. Vale destacar que esto no requiere modificar la entrada; sólo se trata de una extensión al algoritmo.

Para ello, en la iteración i, j de **estimarNCH**, se tiene en cuenta tanto $n_i^t(p_j - p_i)$ como $-n_i^t(p_j - p_i)$, considerando así tanto el caso donde la i-ésima normal tiene su orientación original (cualquiera sea esta) como aquel en el que tiene la orientación inversa. Para cada una de estas dos orientaciones se calcula un valor distinto de ρ_i, que nombramos ρ_i^+ y ρ_i^- respectivamente. A su vez, cada uno de estos valores da lugar a una función de borde distinta para el semiespacio de p_i, que denominamos f_i^+ y f_i^- respectivamente. Notar

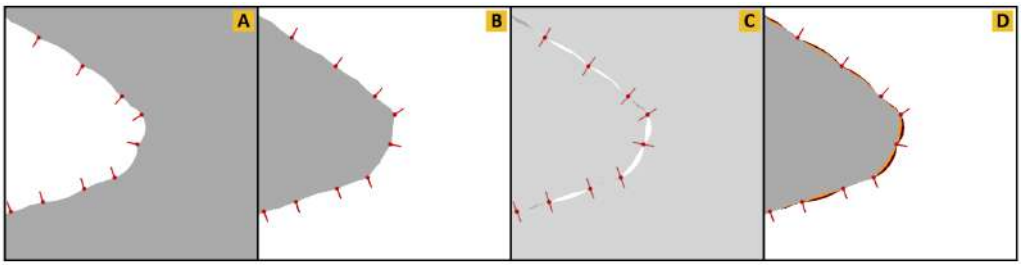


Fig. 3.3: A: NCH con normales hacia adentro. B: NCH con normales hacia afuera. C: superposición de ambas orientaciones. D: NCH simétrica. Imagen tomada de mesh.brown.edu/nch.

que f_i^+ es exactamente la misma función que se obtendría en esta iteración corriendo el procedimiento **estimarNCH** original a la nube con sus normales sin modificar, y lo análogo vale para f_i^- si se corre el procedimiento **estimarNCH** original a la nube con sus normales dadas vuelta. En este contexto, denominamos *NCH positiva* al resultado de hacer lo primero y *NCH negativa* al de hacer lo segundo.

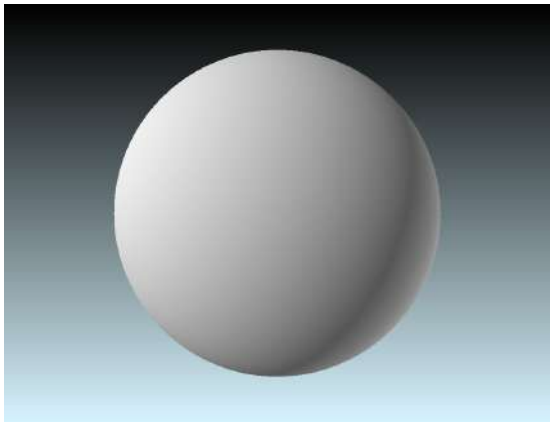
Finalmente, contando con f_i^+ y f_i^- para cada i, se define $f_i(x) = (f_i^+(x) - f_i^-(x))/2$. En general, la mayoría de los puntos de una entrada se encontrarán o bien del lado positivo de la NCH positiva y del lado negativo de la NCH negativa, o bien del lado negativo de la NCH positiva y del lado positivo de la NCH negativa (ver figuras A y B en 3.3). En el primer caso, $f_i^+(x) - f_i^-(x)$ tendrá signo positivo y $f_i(x)$ será el promedio de los módulos de ambos valores, y en el segundo $f_i^+(x) - f_i^-(x)$ tendrá signo negativo y $f_i(x)$ será la negación del promedio de los módulos. Vale destacar la importancia de que la nueva definición de f_i conserve la propiedad de dividir el espacio en una región positiva y otra negativa. Definida así, la positividad y negatividad del interior y exterior de la figura es consistente con la formulación original de la NCH: si originalmente las normales estaban orientadas hacia adentro, f_i es positiva adentro y negativa afuera, y si estaban orientadas hacia afuera, lo contrario. Si ambos valores son positivos o ambos negativos (en figura C de 3.3, áreas blancas y grises oscuras respectivamente), la cuenta $f_i^+(x) - f_i^-(x)$ restará sus módulos. Los casos de este tipo donde ambos módulos en particular resulten iguales serán aquellos que se encuentran en el borde del i-ésimo semiespacio.

3.6 Análisis de resultados

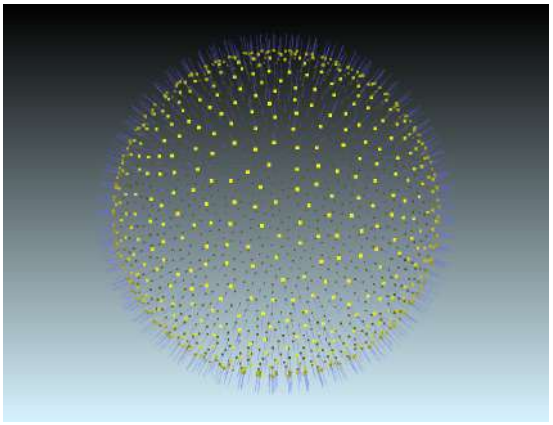
En esta sección se exponen resultados de este algoritmo. Se eligen primero entradas sintéticas y controladas para mostrar características específicas del método relacionándolas con la teoría desarrollada. Luego se muestran resultados para inputs más generales. Destacar estas propiedades de la reconstrucción por este método será de gran relevancia a la hora de analizar los resultados producidos por PUNCH, ya que NCH es parte central de este último algoritmo.

3.6.1 Características esféricas y planas

En la figura 3.4 se exhiben resultados de reconstrucción con NCH para una esfera sintética. La misma se genera creando programáticamente una malla de alta resolución. Luego, se extrae una nube de puntos muestreando esa malla y se aplica NCH con las variantes positiva, negativa y simétrica.

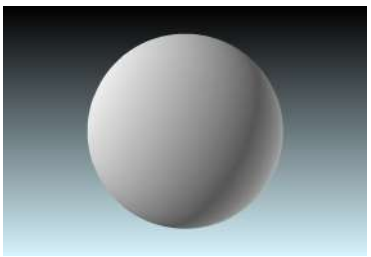


(a) Figura sintética



(b) Nube de puntos orientada con las normales hacia afuera

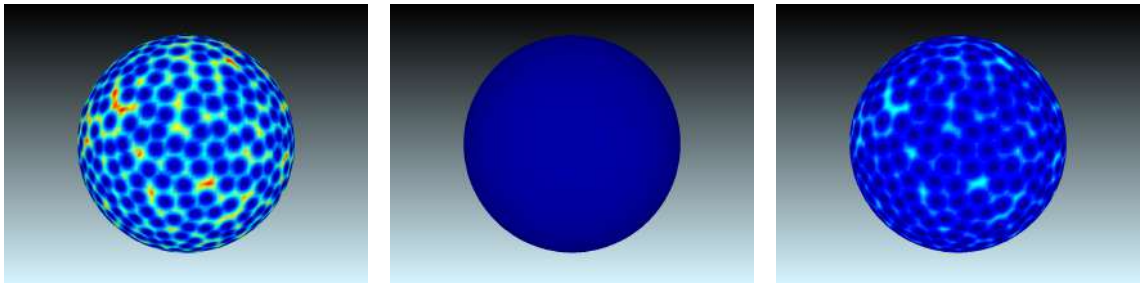
(c) NCH positiva (normales hacia afuera)



(d) NCH negativa (normales hacia adentro)

(e) NCH simétrica

Fig. 3.4: Resultados de reconstrucción para una esfera sintética



(a) NCH positiva *(b)* NCH negativa *(c)* NCH simétrica

Fig. 3.5: Comparación por superposición a escala de colores entre la esfera sintética original y las reconstrucciones por NCH

Podemos apreciar en las imágenes que la reconstrucción negativa es la más exitosa. Esto ocurre ya que, al tener la entrada las normales orientadas hacia adentro, la reconstrucción se realiza definiendo esferas en el interior de la figura. Como la figura *es* una esfera, resulta que cada semiespacio coincide de manera exacta a efectos prácticos con la esfera entera que se está reconstruyendo. En cambio, en la variante positiva, cada semiespacio define una esfera de radio infinito en el exterior de la figura, generando cada uno un parche plano en la reconstrucción. La variante simétrica muestra, como es de esperar, un resultado intermedio. La forma imperfecta de los semiespacios en el exterior de la figura aún se notan, pero de manera menos acentuada que en el caso de la reconstrucción positiva.

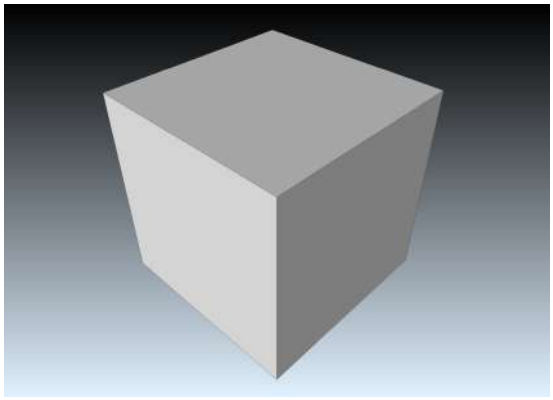
Con el fin de hacer una apreciación un poco más precisa de estas diferencias, en la figura 3.5 mostramos una comparación por superposición entre la reconstrucción obtenida y la figura original utilizando Metro.

Fig. 3.6: Escala RGB utilizada en Metro

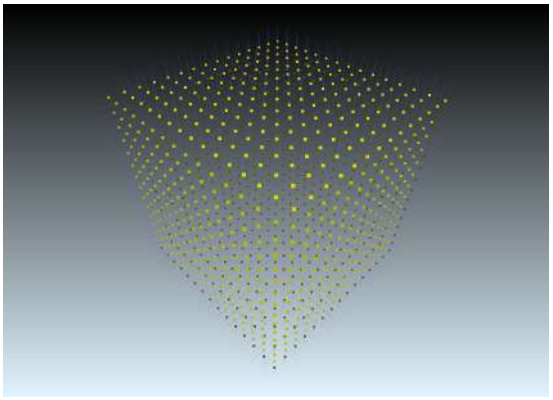
Entre otras cosas, el framework toma dos mallas y realiza mediciones de distancia desde una malla hacia la otra y produce un mapeo de esas distancias a una escala de colores RGB como la exhibida en la figura 3.6. En la figura 3.5 aplicamos este procedimiento comparando cada reconstrucción con la malla de la figura sintética original de la esfera. Notemos que los colores azules representan las distancias más bajas, los verdes distancias intermedias y los colores rojos las distancias más altas.

En la reconstrucción positiva vemos parches azules que corresponden a los planos construidos por NCH para cada punto. Es esperable un valor cercano a cero en esos puntos ya que, por construcción, dichos planos interpolan a los mismos. En las regiones donde los parches planos de cada punto se conectan entre sí es donde vemos mayores valores de error (colores verdes, amarillos y rojos). La NCH negativa, al ser prácticamente la misma esfera que se está reconstruyendo, naturalmente nos da una distancia nula entre la reconstrucción y la esfera original en todos lados. En el caso simétrico, observando este mapeo a colores, tenemos una buena corroboración del carácter "intermedio" que presenta esta reconstrucción en comparación con la positiva y la negativa.

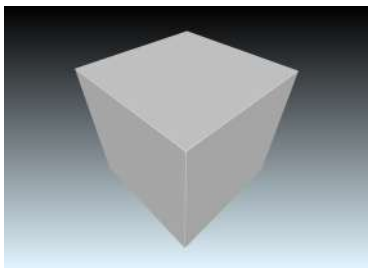
En la Figura 3.7, podemos ver los resultados de reconstrucción para un cubo sintético generado programáticamente de manera similar a la esfera. En calidad de reconstrucción, ocurre lo opuesto que con la esfera: la mejor reconstrucción es la positiva, mientras que la peor es la negativa. Esto se debe a que, al apuntar las normales de la nube hacia afuera, la reconstrucción positiva genera todos sus semiespacios con esferas de radio infinito y logra reconstruir con éxito la naturaleza plana de las caras del cubo. La reconstrucción negativa,



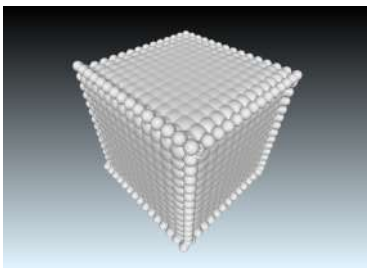
(a) Figura sintética



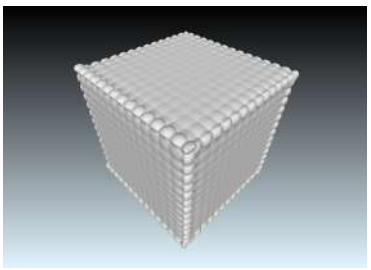
(b) Nube de puntos orientada con las normales hacia afuera



(c) NCH positiva (normales hacia afuera)

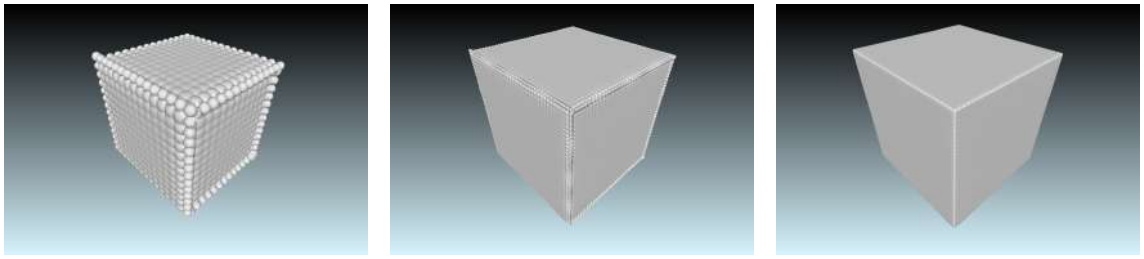


(d) NCH negativa (normales hacia adentro)



(e) NCH simétrica

Fig. 3.7: Resultados de reconstrucción para un cubo sintético



(a) Nube con $\approx 1k$ puntos *(b)* Nube con $\approx 10k$ puntos *(c)* Nube con $\approx 100k$ puntos

Fig. 3.8: Mejora en la calidad de la reconstrucción negativa en función de la densidad de la nube

en cambio, sufrió una consecuencia similar a la positiva en el caso de la esfera debiendo aproximar una forma no-esférica con esferas. En este caso, estando las normales orientadas hacia adentro, queda visible el exterior de las esferas de los semiespacios, contrario al caso anterior donde se visualiza la parte interna de los mismos al apuntar las normales hacia afuera. De manera parecida al caso de la esfera, la reconstrucción simétrica provee un término medio entre las positiva y la negativa, produciendo igualmente una malla de calidad afectada negativamente por la geometría de los semiespacios.

Otra observación interesante para relacionar con la teoría es la evidente variación en el radio de las esferas. Recordemos que la esfera del semiespacio del i-ésimo punto se define con el mayor radio que deja al resto de los puntos sobre la frontera de la esfera o fuera de ella (cumpliendo $f_i(p_j) \leq 0$ para todo $j \neq i$). Cuanto más lejos se encuentra un punto ubicado sobre una cara del cubo del resto de sus caras, mayor es la esfera que se puede definir para su semiespacio. Este fenómeno se observa claramente en las reconstrucciones negativa y simétrica de la figura 3.7, donde los puntos más lejanos del borde de la cara a la que pertenecen presentan un aspecto “más plano” teniendo su semiespacio un radio mayor que el de los puntos más cercanos a los bordes.

3.6.2 Densidad de la nube

Las imágenes anteriores también sugieren que existe una relación estrecha entre el efecto sobre la calidad de reconstrucción de los fenómenos exhibidos y la densidad de la nube de entrada. Tanto con las esferas como con la nube, una mayor densidad de puntos significa que las cáscaras visibles de semiespacios esféricos serían más chicos en relación a la reconstrucción entera. Para visualizar esta dependencia de la densidad de puntos, podemos observar en la figura 3.8 cómo varía la calidad percibida de reconstrucción en función de esa característica de la entrada. Notemos que el radio de las esferas no cambia con el aumento de densidad ya que las dimensiones del cuerpo que la nube representa son las mismas; lo que sí ocurre es que se construyen muchas más de estas esferas y se visualiza una parte mucho más chica de la cáscara de cada una. Sin embargo, vale destacar también que aún con un aumento considerable en la densidad de la nube, cerca de los bordes del cubo se siguen notando ciertos artefactos. Estos se ven debido a que las esferas de puntos cercanos al borde siguen recibiendo un radio muy chico. Cuanto más angosto sea el ángulo encerrado en un borde tal, más se notará la forma esférica asociada a los puntos cercanos a ese borde.

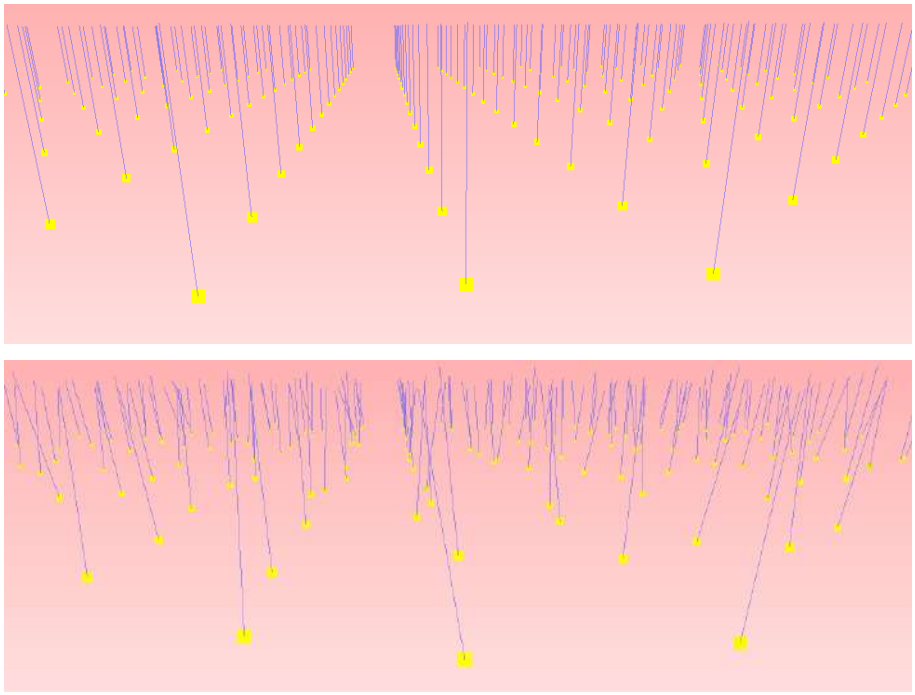


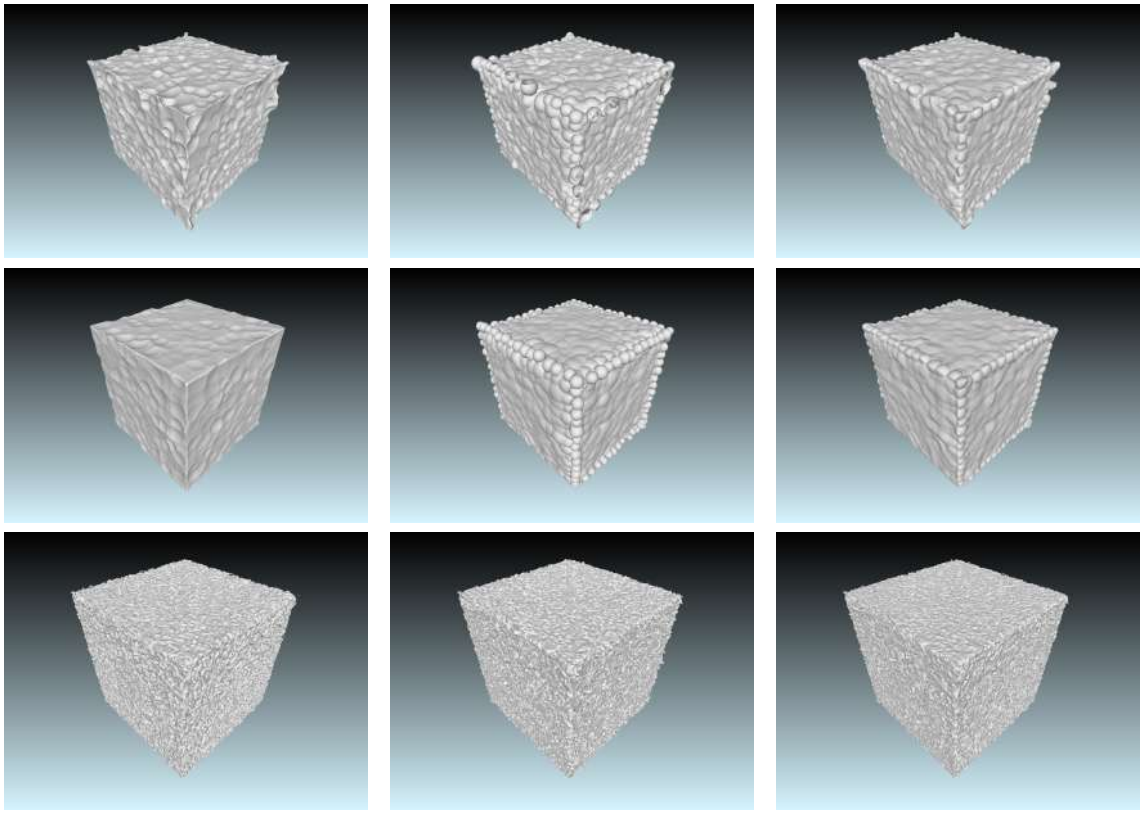
Fig. 3.9: Compración entre puntos sin ruido (arriba) y puntos con un pequeño ruido gaussiano en sus posiciones y sus normales (abajo).

3.6.3 Ruido en la nube

Habitualmente, las nubes de puntos obtenidas por medios ópticos como escáneres 3D tienen cierta cantidad de *ruido*, tanto en la posición de los puntos como en las normales asociadas a ellos. Los casos exhibidos hasta ahora, como se ha explicado, fueron generados programáticamente sin ningún tipo de ruido. Por ese motivo es que, tomando como ejemplo el caso de la NCH negativa de la esfera, el método supo construir para cada punto un semiespacio cuyo borde esférico coincide con la esfera a reconstruir. De haber estado la normal de algún punto algo desviada, esto no hubiera resultado así y se tendría en ese punto un semiespacio de radio menor. Cuanto más desviada esté la normal, menor será el radio de la esfera de su semiespacio y más notable resultará a la vista en el resultado final. Puede observarse un ejemplo de nube ruidosa en la figura 3.9.

Algo similar ocurre con el ruido en la posición de los puntos. Tomemos como ejemplo esta vez el caso de la reconstrucción positiva del cubo, en la que el método definió para cada punto un semiespacio de radio infinito. Recordemos que esto ocurre cuando el algoritmo encuentra, dado un punto en particular, que el resto de los puntos se encuentran "detrás" del plano normal al radio de ese punto. Si a causa de ruido un punto de una cara del cubo se encontrara corrido hacia el interior del mismo, pasaría de repente a tener otros puntos de la nube "delante" y ya no se le asignará al semiespacio de ese punto un radio infinito.

Para visualizar estos efectos, podemos referirnos a la figura 3.10. En ella, se toma el caso del cubo y se le agrega un pequeño ruido gaussiano en las normales por un lado, en las posiciones de los puntos por otro y luego ambos tipos de ruido juntos. Se nota claramente el efecto negativo del ruido en la calidad percibida de la reconstrucción, especialmente interesante en la variante positiva de la NCH anteriormente muy exitosa. Vale aclarar que, si bien lo habitual es trabajar con nubes reales que presentan ruido de ambos tipos, aquí



(a) NCH positiva *(b)* NCH negativa *(c)* NCH simétrica

Fig. 3.10: Reconstrucciones para nubes con ruido. Primera fila: nube esparsa con ruido en las normales. Segunda fila: nube esparsa con ruido en las posiciones. Tercera fila: nube densa de $\approx 100k$ puntos con ruido tanto en las normales como en las posiciones.

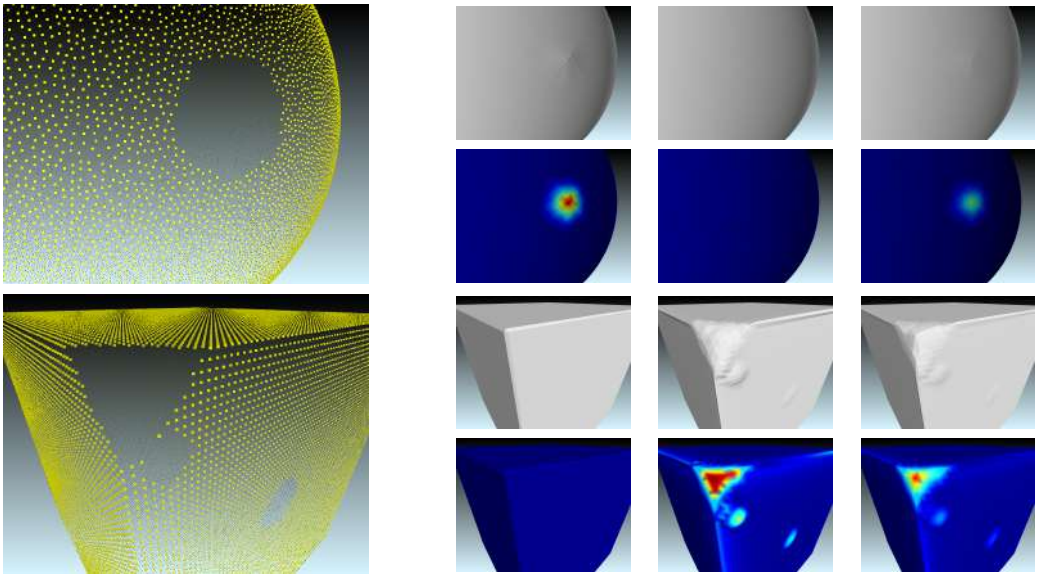


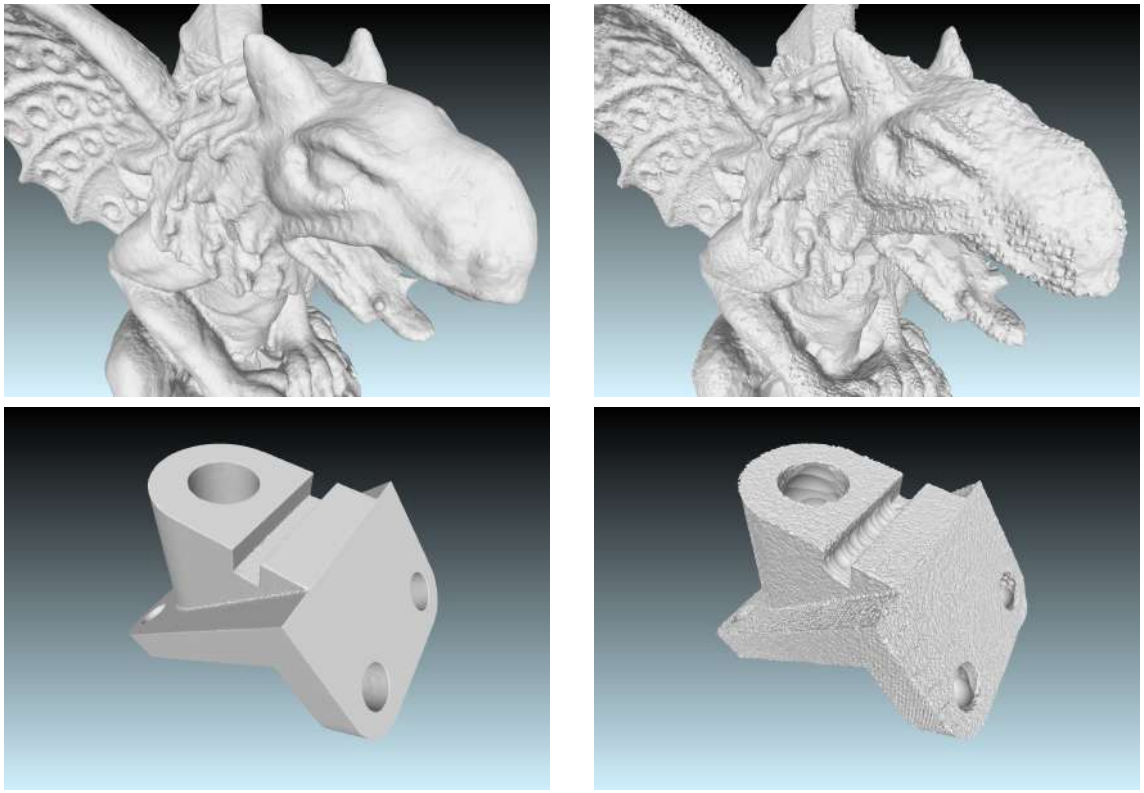
Fig. 3.11: Reconstrucciones para figuras sintéticas en base a nubes densas de $\approx 50k$ puntos con agujeros

se utilizó un ruido de intensidad fija y arbitraria con una distribución particular con el fin de relacionar estos fenómenos con la teoría. De todas maneras, es útil analizar casos de este tipo de manera sintética y controlada ya que es de esperar que este comportamiento se dé en la práctica también.

3.6.4 Agujeros en la nube

Otra clase de error que suele darse en nubes de puntos obtenidas por medios ópticos es la de parches de puntos faltantes, o "agujeros" en la nube. Es interesante entonces observar también cómo se comporta un método como este ante tal tipo de características de la nube.

En la figura 3.11 mostramos reconstrucciones de las mismas figuras sintéticas ya introducidas utilizando nubes densas con agujeros grandes. Los casos anteriormente exitosos de reconstrucción para la esfera y el cubo siguen dando resultados altamente precisos a pesar de los agujeros. Esto es de esperar: las esferas definidas por los puntos que sí están presentes suplen perfectamente a las que se definirían por los puntos ausentes, dado que estos generarían cada uno exactamente la misma esfera. Lo análogo ocurre en la NCH positiva del cubo con los planos que se computan por cada punto. Como también es esperado, la variante positiva en el caso de la esfera y la negativa en el del cubo presentan resultados inferiores. Sin embargo, es razonable decir que en esta comparación en particular los resultados exitosos lo son casi desmedidamente. Vistos fuera de contexto, pueden dar la idea de que el algoritmo es capaz de inferir de manera casi perfecta la forma a reconstruir a pesar de las considerables ausencias de datos. No obstante, conociendo el algoritmo, sabemos que esto resulta así dado que el mismo lleva a cabo la reconstrucción calculando justamente esferas y planos. Con esto en mente, podemos destacar de este método que, aún en las variantes menos exitosas, sigue produciendo una superficie sólida que "tapa" los agujeros. Quizás más importante aún, es robusto ante los agujeros en el sentido de



(a) Usando una nube rin ruido

(b) Usando una nube con ruido realista

Fig. 3.12: Reconstrucciones por NCH de dos figuras de Reconbench. Gargoyle, la primera, se reconstruye por la NCH simétrica. Anchor, la segunda, por NCH positiva.

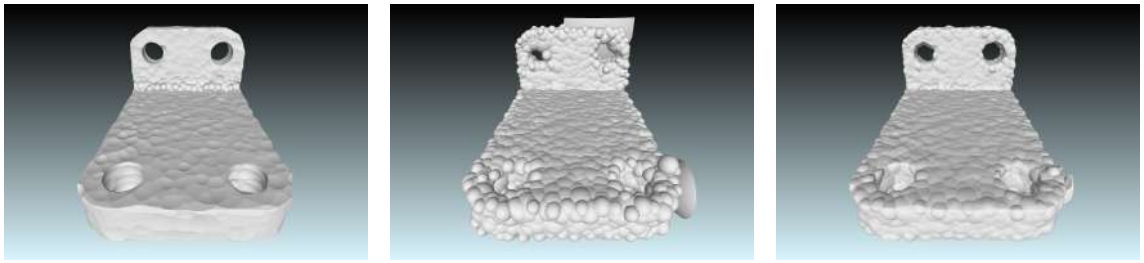
que la reconstrucción no es afectada por ellos en el resto de la nube: donde la nube sigue intacta, la reconstrucción lo es también.

Por último, volvemos a observar cómo la variante simétrica de NCH constituye un término medio entre la positiva y la negativa.

3.6.5 Cuerpos complejos

Los resultados anteriores fueron obtenidos para entradas sintéticas y controladas con el fin de exhibir y discutir características particulares de este algoritmo. Para concluir esta discusión, mostramos en la figura 3.12 reconstrucciones hechas para nubes de puntos de cuerpos más realistas y complejos. Utilizamos algunas de las nubes del conjunto de datos que en las próximas secciones se usará de manera extensiva para evaluar el algoritmo PUNCH.

Las nubes limpias se obtienen realizando programáticamente un muestreo de puntos de superficies *ground truth*. Las ruidosas, tomadas del conjunto de datos propuesto en [6], se calculan también programáticamente, pero simulando un escaneo 3D como el que resultaría de usar una máquina con un objeto real como se explica en el artículo citado. Vemos, cómo a pesar de su sencillez matemática y de los recursos geométricos básicos que utiliza, el método logra reconstrucciones intuitivas y de apreciación cualitativa satisfactoria aún cuando es llevado nubes de puntos de figuras complejas. Asimismo, en los casos ruidosos, podemos detectar a simple vista artefactos compatibles con los que obtuvimos en los casos



(a) Variante positiva *(b)* Variante negativa *(c)* Variante simétrica

Fig. 3.13: Comparación de las tres variantes ante una nube ruidosa. Se utiliza una versión esparsa de una nube de Anchor preparada mediante un algoritmo de simplificación de nubes de puntos con el fin de permitir una mejor visualización de la participación de cada semiespacio.

ruidosos discutidos anteriormente.

3.7 Comparación entre las variantes de NCH

La principal ventaja de la negativa es el poder ofrecer parches curvos para regiones convexas del cuerpo reconstruido, y su principal desventaja es el estar limitada a utilizar esferas que quepan en el interior del cuerpo. La principal ventaja de la positiva es el contar con la posibilidad de definir parches planos, teniendo como desventaja el deber utilizar tales parches para regiones curvas y convexas. Sin embargo, en base a la apreciación visual que hemos hecho, parecen ser más aceptable el peor caso de la variante positiva que el de la negativa. Cualitativamente, resultan más intuitivos los parches planos que aproximan insatisfactoriamente convexidades curvas (ver 3.4c) que las esferas en el interior de la figura que aproximan insatisfactoriamente regiones planas (ver 3.7d y 3.8), especialmente cerca de bordes y esquinas e incluso cuando la densidad se incrementa. Desde el punto de vista teórico, nubes más densas serían más favorables aún a la variante positiva debido a que la parte visible del sector plano de cada semiespacio sería mucho más chica cuanto más juntos se encuentren los puntos. Asimismo, tampoco hay un empeoramiento en la *reconstrucción positiva* de *características curvas* de la manera que lo hay en la *reconstrucción negativa* de *parches planos*: un plano siempre es igual de plano, mientras que una sección esférica como aproximación de una región plana varía su calidad cuanto en función radio de las esferas que utiliza, siendo peor cuanto menor sea el radio (fenómeno observable en la figura 3.8). En el caso de agujeros en las nubes, la variante positiva también parece ofrecer una reconstrucción más intuitiva de la esfera que la negativa del cubo (figura 3.11).

Por su parte, la variante simétrica ofrece consistentemente un intermedio entre la positiva y la negativa, hecho notable tanto visualmente en las mallas como en las comparaciones hechas por superposición con las mallas sintéticas originales (como en 3.5). La utilidad de un intermedio tal merece una discusión. A priori, parece ser mayor el beneficio obtenido por utilizar la variante positiva ante bordes de láminas planas, donde se aproximan dichas láminas directamente colocando planos (como en 3.7c), que el beneficio de promediar la positiva con la negativa para mitigar la reconstrucción de características curvas con parches planos (como en 3.4e). En otras palabras, desde el punto de vista cualitativo, es mayor la mejora de 3.7c cuando se compara con 3.7d y 3.7e que la mejora de 3.4e comparado con 3.4c. Al mismo tiempo, es menor la pérdida de calidad de la variante positiva por usar

parches planos en características curvas (como en 3.4c) que el empeoramiento introducido en bordes de láminas planas al promediar la positiva con la negativa (como en 3.7e).

Dicho esto, vale observar que estas consideraciones favorables a la variante positiva se sostienen siempre que analicemos resultados provenientes de nubes sintéticas sin ruido, como hicimos recién. Con eso en mente, vale la pena discutir aparte el comportamiento ante nubes ruidosas, que son el caso de uso más realista y que seguiremos teniendo a lo largo de este trabajo con el conjunto de prueba de Reconbench.

En nubes ruidosas, como ya explicamos, puntos que deberían ser co-planares no lo son, y vectores que deberían ser normales a un mismo plano no lo son tampoco. Peor que eso, puede ocurrir que se tengan normales casi invertidas. En la práctica, ante la presencia de ruido de estas características, las tres variantes presentan un comportamiento que se adhiere al tipo de error de cálculo de semiespacios explicado en la sección sobre nubes con ruido de este capítulo (sección 3.6.3), calculándose semiespacios de radio mucho menor al ideal.

En la figura 3.13 se muestran resultados de cada variante aplicada a entradas ruidosas de Reconbench. En la variante negativa, que ubica esferas en el interior del cuerpo reconstruido, esto se traduce en "protuberancias" producto de las secciones visibles exteriores de esas esferas. En la positiva, por el contrario, que ubica esferas en el exterior del cuerpo, se observan "pozos" correspondientes al interior de las esferas de los semiespacios. La variante simétrica, por su parte, ofrece como siempre su efecto intermedio, pero que no alcanza a compensar satisfactoriamente las protuberancias con los pozos. Esto se debe a que la simétria "promedia" los resultados negativos y positivos, y las esferas de la variante negativa son de radio mucho menor a las producidas por la positiva: las primeras están limitadas a caber en el interior del cuerpo reconstruido, mientras que los radios de las segundas no tienen esa restricción.

Además, haciendo referencia nuevamente a 3.13, es evidente la peor performance de la variante negativa en cuanto a artefactos. Esto se debe a que, ante normales muy ruidosas, el algoritmo negativo a veces coloca esferas ubicadas completamente por fuera del cuerpo. Si bien la variante positiva también coloca esfera en el exterior del cuerpo, dichas esferas son acompañadas de esferas exteriores vecinas. Ausente esa condición, la esferas exteriores en la negativa se vuelven visibles y detrimentales a la calidad cualitativa de sus resultados. En la reconstrucción positiva, en cambio, una normal invertida a lo sumo dispararía la ubicación de una esfera interior, pero eso, naturalmente, no lleva a un artefacto tan visible. La variante simétrica, al compensar las esferas exteriores de la negativa con las interiores de la positiva, elimina esos artefactos. Sin embargo, como ya se observó, la compensación de la positiva y la negativa hecha por la simétrica no alcanza a ser cualtiativamente satisfactoria. En última instancia, tanto la teoría como la evaluación cualitativa son favorables a la positiva.

Multi-level Partition of Unity Implicits (MPU)

En el método propuesto por [24], se define de manera implícita una superficie que aproxima la nube de puntos de entrada. Como se ha explicado, este método es local: define aproximaciones de distintas regiones de la nube que luego son combinadas en una aproximación global. Cada aproximación local es una SDF en su propio derecho, y la global lo es también. La combinación se hace utilizando funciones de ponderación que conforman una *partición de la unidad*, esto es, suman uno globalmente.

El método se propone proveer aproximaciones rápidas, de alta calidad, robustas ante agujeros en la nube y ruido en los puntos, y que conserven características "filosas" de la figura como esquinas o bordes, todas propiedades generalmente deseables en reconstrucción de superficies 3D. El hecho de ser un método aproximado y no interpolante es importante para lograr la robustez planteada ante escasez de puntos y/o ruido en los mismos. Naturalmente, la interpolación de datos ruidosos produce resultados que lo son también.

Para capturar de manera fiel bordes y esquinas, el método cuenta con una serie de pruebas que, ante la porción de la nube residente en una localidad dada, determinan si la misma presenta una característica de ese tipo. En base a lo que resulte de esas pruebas, el procesamiento se adapta y elige entre distintas variantes de aproximación local para conservar dichos detalles lo mejor posible y no "sobre-suavizarlos", como ocurre por ejemplo en los métodos de Kazhdan et al. [21, 20].

Una característica adicional y central del enfoque de este método es su naturaleza error-adaptativa. El método cuenta con una medición de error de aproximación en base a la *Taubin distance* [26], y permite al usuaria definir un valor máximo permisible para esta medida de error. El algoritmo intenta satisfacer esta restricción realizando una reconstrucción de manera de tener un error estrictamente menor al solicitado en la mayor cantidad de localidades que pueda. Esta adaptabilidad guiada por error es lograda mediante un esquema de subdivisión del espacio que parte de un cubo contenedor de la nube de puntos y da por resultado una partición del mismo en celdas de distintos tamaños. Cada una de esas celdas denota el centro de una localidad esférica en la que se llevará a cabo una reconstrucción local individual. Luego de definir todas las aproximaciones locales, la evaluación de la función implícita en un punto se realiza ubicando la celda de la subdivisión en la que ese punto cae. El valor de la SDF devuelto para ese punto será el de la aproximación local de esa celda, combinado en mayor o menor medida con los de celdas vecinas en función de cuán cerca se encuentre el punto de ellas; cuanto más cerca de una celda vecina, más influirá la SDFs de esa celda en el resultado de la evaluación.

Vale destacar que, como se explica en las secciones siguientes, la subdivisión se realiza utilizando *octrees*, estructura de árbol que provee búsquedas en tiempo logarítmico. Entonces, la evaluación de la función implícita es rápida al poder ubicar la celda que contiene un punto en tiempo de peor caso logarítmico en el tamaño de la nube. Es interesante

destacar que la complejidad del algoritmo para una nube dada es *output-sensitive*, como la describen los autores del método, en el sentido de que los requerimientos de tiempo y espacio para una entrada dependen del parámetro de error solicitado por la usuaria y de la complejidad del cuerpo a reconstruir.

4.1 Partición de la unidad

La partición de la unidad es una técnica algorítmica con variadas aplicaciones y un fundamento matemático teórico riguroso [5]. Como técnica en general, permite la división del dominio de datos en partes más chicas que son tratadas por separado y luego combinadas para dar el resultado final. En nuestro caso particular, esto se hace "mezclando" las SDFs locales mediante funciones suaves denominadas "funciones de partición de la unidad". Estas funciones se definen una por cada localidad, pero siempre de modo que sumen uno en todo el dominio, y de ahí su nombre.

Concretamente, supongamos un dominio acotado Ω en $\mathbb{R}^3$ y un conjunto de funciones de partición de la unidad $\{\phi_i\}$ tal que $\sum_i \phi_i = 1$. Cada ϕ_i está definida en un subdominio denominado $supp(\phi_i)$, o *soporte* de ϕ_i, contenido en Ω. Supongamos además que contamos con un conjunto de aproximaciones locales V_i asociado a la i-ésima localidad para todo i, definidas en $supp(\phi_i)$. Recordemos que cada localidad se define como una esfera en torno al centro de una celda resultante de la subdivisión previa de un cubo delimitador de la nube de entrada.

De esta manera, una aproximación global de una función $f(x)$ está dada por $f(x) \approx \sum_i \phi_i(x) Q_i(x)$, donde $Q_i \in V_i$. Dadas funciones no-negativas $\{w_i(x)\}$ soportadas compactamente de manera que $\Omega \subset \cup_i supp(w_i)$, nuestras funciones de partición de la unidad se definen como $\phi_i(x) = \frac{w_i(x)}{\sum_{j=1}^{n} w_j(x)}$. En el caso del método en cuestión, cada w_i es una función de peso definida según $w_i(x) = b(\frac{3|\mathbf{x}-\mathbf{c_i}|}{2R_i})$, soportada dentro de su localidad esférica centrada en la celda i, donde

$$b(t) = \begin{cases} -t^2 + 3/4 & \text{si } \ t < 0{,}5 \\ \frac{1}{2}(\frac{3}{2} - t)^2 & \text{caso contrario} \end{cases}$$

es la función de *B-spline cuadrática*, $\mathbf{c_i}$ es el centro de la celda y R_i el radio de la esfera.

Vale destacar que la definición de las funciones de peso es central al funcionamiento de un método de este tipo. Como se explica en Thacker et al. [28], las funciones de peso definidas en el algoritmo original de Shepard otorgan mucha influencia a datos lejanos de la localidad para la que están definidas, y tienen su soporte en el dominio entero. Por el contrario y de manera similar a los pesos planteados aquí, las funciones de peso en la modificación al método de Shepard propuesta por Franke y Nielson, conocidas como funciones de peso por distancia inversa (*inverse distance weighting*), sólo toman valores positivos en un soporte esférico centrado en cada localidad. Más aún, las funciones de peso por distancia inversa pueden generalizarse con un parámetro que controla la influencia ejercida por datos lejanos a la localidad.

Asimismo, al tener estas funciones de peso un soporte local esférico, también entra en juego el tamaño de dicha esfera: cuanto mayor sea, más influencia tendrán datos vecinos. De acuerdo a los autores de este método, cuanto mayor sea la esfera, mayor calidad tendrá la aproximación global resultante, pero también se incurrirá un mayor costo computacional.

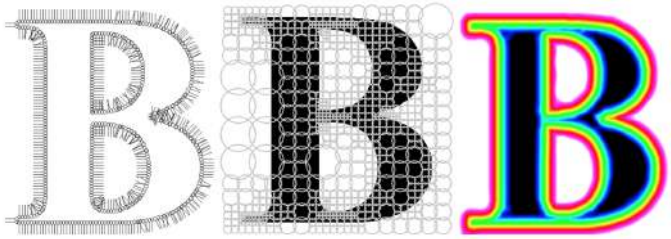


Fig. 4.1: Definición de una SDF por MPU. Izquierda: nube de puntos orientada. Medio: soportes restulantes de una subdivisión adaptativa, exhibidos al 50 % de su tamaño real. Se puede ver cómo la granularidad de la subdivisión aumenta para características como curvas, bordes o puntas. Derecha: SDF visualizada a escala de color CMYK. El conjunto de nivel cero se encuentra entre el amarillo y el verde, siendo el negro negativo y el magenta positivo. Figura tomada de [24].

4.2 Subdivisión y criterios de corte

El mecanismo de subdivisión del método es central para su capacidad de controlar el error de aproximación y de proveer una evaluación rápida de la función implícita que sólo depende de la vecindad del punto de evaluación. Implementativamente, este mecanismo es guiado por la construcción de un octree cuyos nodos representan celdas que se producen a medida que se subdivide el espacio ocupado por la nube. Al realizar la división de una celda, la misma se parte en 8 octantes de igual volumen. En el octree, la celda dividida es un nodo y los octantes resultantes de la subdivisión son sus ocho hijos. De esta manera, las celdas resultantes de la subdivisión finalmente son las hojas del octree.

El proceso comienza con un cubo delimitador de la nube alineado con los ejes cartesianos con diagonal de tamaño 1. En caso de que la nube sea demasiado grande, se cambia su escala para que entre en un cubo tal. A esta altura del proceso, la división del cubo consiste en una sola celda, o equivalentemente, nuestro octree tiene un solo nodo. En esta celda inicial se construye una SDF que aproxima los puntos de la nube contenidos en ella, que en este caso conforman la nube entera. Se calcula el error de aproximación de esta SDF según el criterio basado en la distancia de Taubin, y si supera el parámetro de error máximo ϵ_0 definido por la usuaria, la SDF calculada se descarta y la celda es subdividida. El proceso se repite recursivamente sobre las celdas hijas.

En caso de que el error calculado para la SDF local de una celda resulte menor que ϵ_0, esta aproximación local se conserva y la celda en cuestión no se vuelve a subdividir, quedando como hoja del octree de subdivisión. Esta SDF local pasará a participar de la evaluación de la función global en puntos contenidos por su celda o que resulten lo suficientemente cercanos a ella.

Con este esquema de subdivisión guiado por error, la subdivisión tiende a ser más granular en regiones que presentan mayores dificultades de aproximación y deja localidades más grandes para las regiones más fáciles de aproximar. Esto se distingue claramente de una reconstrucción basada en una subdivisión regular, donde para tener cierta granularidad alta en las regiones más complejas de la figura debe incurrirse el costo de tener esa granularidad en regiones más sencillas que no la requieren y para las cuales una aproximación más gruesa alcanza. Esto puede apreciarse con un ejemplo en la figura 4.1.

Para determinar el soporte local de una celda con centro $\mathbf{c}$, siendo los soportes esferas centradas en su celda, basta con definir un radio de soporte R. Esto se hace en primera

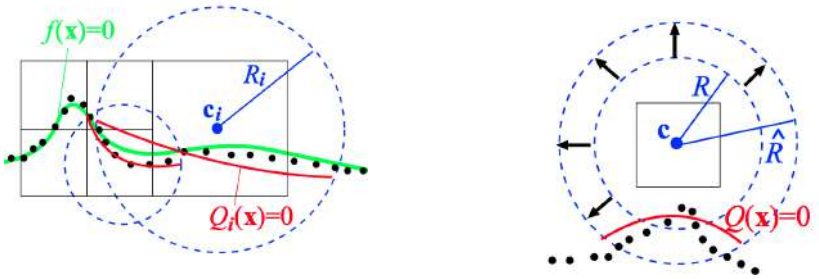
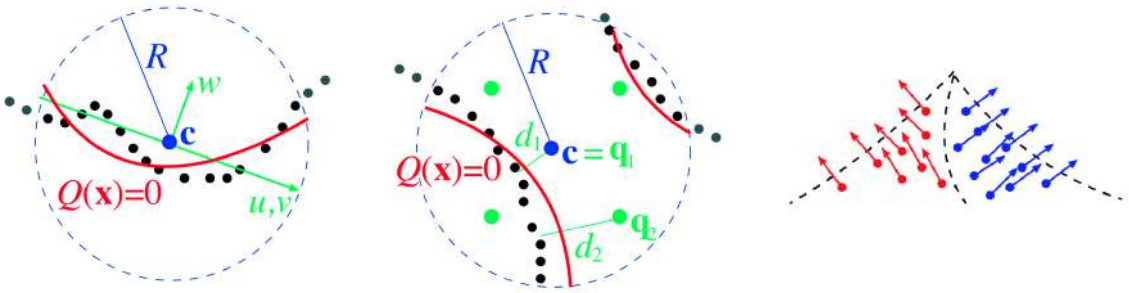


Fig. 4.2: Definición de soportes locales. En rojo se muestran aproximaciones locales y en verde la aproximación resultante de combinarlas mediante funciones de peso. Izquierda: celdas de distintos tamaños y los soportes de dos de ellas. Derecha: expansión del radio de un soporte local para el cual se tenían demasiados pocos puntos. Se exhibe la aproximación local realizada para los puntos del soporte expandido.

instancia de manera proporcional al tamaño de la celda, definiendo $R = \alpha d$ donde d es el tamaño de su diagonal y α es algún multiplicador positivo. Los autores proponen un valor de $\alpha = 0{,}75$ remarcando, como se ha mencionado, que un mayor valor produce una mejor aproximación global pero resulta detrimental a la eficiencia agregando un costo temporal aproximadamente cuadrático en α. Asimismo, puede ocurrir que la bola resultante de radio R contenga una cantidad de puntos chica e insuficiente para obtener una aproximación fiel de su subconjunto de la nube de puntos. Los autores definen un mínimo de puntos como $N_{min} = 15$, y en caso de que la cantidad de puntos dentro de la bola de radio R y centro $\mathbf{c}$ sea menor a N_{min}, se aplica la iteración $R_j = R_{j-1} + \lambda R, j > 1$ hasta obtener para algún j un soporte cuyo radio R_j reúne N_{min} puntos o más. Los autores sugieren un valor de $\lambda = 0{,}1$ para realizar la iteración.

4.2.1 Tratamiento de celdas vacías

Para la gran mayoría de las entradas, va a ocurrir durante la subdivisión que se obtengan celdas hijas vacías. En principio, para realizar una reconstrucción para una celda vacía, necesariamente debe llevarse a cabo una iteración de expansión como la explicada arriba. Una vez hecho esto, se tendrá un soporte con una cantidad de puntos mayor o igual a N_{min} y se computará una aproximación local. Si el error de aproximación resultante es menor a ϵ_0, la celda en cuestión se conservará como hoja junto a la aproximación que se le calculó. Sin embargo, si ocurre lo contrario y el error de aproximación da mayor o igual a ϵ_0, se procederá en principio a subdividir esta celda vacía. Cada hijo de la subdivisión será a su vez una celda vacía adicional, y se repetirá el procedimiento de expansión para cada una de ellas. Dado que sus soportes tendrán un alto grado de superposición al ser celdas que comparten como ancestro común a la celda vacía original, es esperable que todas terminen realizando aproximaciones para porciones de la nube muy parecidas y, peor aún, con chances similares de producir un error de aproximación parecido al de dicho ancestro común que fue mayor o igual a ϵ_0. De esta forma, puede darse una recursión infinita. Si bien la misma sería controlable mediante una restricción de límite de profundidad para el octree, todas las celdas del subárbol de profundidad máxima que cuelgue de la celda vacía inicial terminarán proveyendo aproximaciones para porciones de la nube casi iguales. Además, se trata de un caso en el que el corte a esta recursión puede realizarse de manera muy sencilla, potencialmente mucho antes de alcanzar ese límite de profundidad,



Aproximación local tipo **(a)**. Aproximación de tipo **(b)**. Aproximación de tipo **(c)**.

Fig. 4.3: Los distintos tipos de aproximación local. En todas, los puntos negros son el subconjunto de puntos de la nube correspondiente a una localidad, y Q es la aproximación local resultante. En (a) y en (b) se muestran en verde puntos y ejes auxiliares que se computan localmente para realizar la aproximación. En (c) se muestra la clusterización en normales de dos láminas para las cuales se calculará un polinomio distinto. Figuras tomadas de [24]

simplemente evitando la subdivisión de celdas vacías.

Con este objetivo, en el paper original se establece que, dada una celda vacía, se le realiza una reconstrucción previa expansión iterativa de su soporte, pero no se subdivide más independientemente del error de aproximación que resulte. No obstante, en la implementación utilizada para este trabajo se eligió utilizar un criterio un poco más estricto, cortando la recursión en celdas de cantidad de puntos inicial menor o igual a 1, en lugar de esperar a celdas con cantidad nula. Se comprueba empíricamente que los resultados de reconstrucción con esta modificación son prácticamente iguales, y los tiempos de ejecución mejoran notablemente al evitar una cantidad de reconstrucciones en el orden de 8 por cada celda con un solo punto, y frecuentemente muchas más por las subdivisiones sucesivas que se puedan realizar en celdas cuyo soporte sigue conteniendo al punto.

4.2.2 Profundidad máxima

En coincidencia con la implementación del método presentada por los autores, es posible establecer como criterio complementario de corte el de máxima profundidad del octree de subdivisión. Esto permite un mayor control sobre el comportamiento de la reconstrucción y la cantidad de aproximaciones locales que se computan y que luego participan en la evaluación de la SDF global.

4.3 Aproximaciones locales

De acuerdo a lo explicado, cada celda de una subdivisión en proceso es candidata a permanecer como celda hoja. Dada una celda tal, se calcula una aproximación local para los puntos de su soporte. Luego, según el error de aproximación que resulte y el parámetro de máximo error tolerable, se decide si la celda se subdivide o si se mantiene como hoja conservando su SDF local.

En la etapa de cómputo de la aproximación local de una celda entra en juego otra de las características más importantes de MPU. El método cuenta con tres tipos distintos de posible aproximación local, entre los cuales elige de acuerdo a las características particulares de cada celda:

(a) Una cuádrica general.

(b) Un polinomio cuadrático bivariado.

(c) Aproximación de bordes y esquinas mediante superficies suaves de a trozos.

Todos estos procesos implican un procedimiento iterativo de minimización por cuadrados mínimos para encontrar los parámetros correspondientes a cada trozo de superficie suave a utilizar en la aproximación local resultante.

Dada la porción de la nube de puntos P' residente en el soporte esférico de una celda (luego de expandirlo, de ser necesario), se realiza una serie de tests que determinan el tipo de aproximación a utilizar entre **(a)**, **(b)** y **(c)**. Estas pruebas son función de la cantidad de puntos en cuestión $|P'|$ y su distribución en el espacio. Como primera medida para las evaluaciones de la distribución, se calcula el promedio del conjunto de normales N' asociadas a los puntos en consideración y se toma la máxima desviación θ existente de ese promedio entre las normales de N'.

Si $|P'| > 2N_{min}$ y $\theta \geq \pi/2$, se intenta aproximar con **(a)**. De acuerdo a pruebas subsiguientes basadas en la distribución de normales de los puntos cercanos a los vértices de la celda, esta aproximación local se lleva a cabo o no. En caso de que las pruebas indiquen que no es pertinente ajustar a los puntos una cuádrica general, la celda se subdivide. Si $|P'| > 2N_{min}$ y $\theta < \pi/2$, se realiza la aproximación por medio de **(b)**.

Si ocurre que $|P'| \leq 2N_{min}$, se realizan pruebas para determinar si puede tratarse de un borde o una esquina. Estas pruebas se hacen sólo en el caso de tener una tal cantidad de puntos porque las mismas son costosas en tiempos de ejecución y se desea evitar realizarlas para cantidades grandes de puntos; notar que, de no tener esta restricción, estas pruebas ya podrían realizarse para la primera celda de todas que será la raíz del octree de subdivisión, cuyo soporte local contendrá todos los puntos de la nube. Al mismo tiempo, naturalmente, bien pueden existir soportes que contengan $|P'| > 2N_{min}$ y que tengan este tipo de característica geométrica. Respecto de esto, los autores plantean que, de ser lo suficientemente exigente el parámetro de máximo error que se elige para la reconstrucción, el proceso de subdivisión se encargará del caso de una celda con una característica filosa y más de $2N_{min}$ puntos detectando un error de aproximación elevado y por ende forzando una subdivisión. Esto tiene sentido dado que en tal situación se estaría tratando de ajustar superficies suaves a características geométricas que no lo son. Luego de las subdivisiones sucesivas que sean necesarias, finalmente las parte de la nube con esa característica se encontrarán contenidas en un soporte local de $2N_{min}$ puntos o menos, y su presencia se detectará mediante las pruebas pertinentes.

En caso de realizar las pruebas y sospechar efectivamente de la presencia de un borde o una esquina, se separan los puntos en clusters de acuerdo a sus normales, esperando que cada cluster contenga normales similares y represente una de las "láminas" suaves de la cuadrática de a trozos a calcular. De esta manera, para cada cluster de puntos se calcula una cuadrática separada, y se obtiene como aproximación local la combinación de estas láminas. El método soporta desde dos hasta tres o cuatro láminas diferentes para una misma característica filosa de la superficie.

4.4 Evaluación de la SDF

Al término del proceso de reconstrucción, la SDF obtenida está lista para ser evaluada, usualmente con el propósito de generar una visualización de la reconstrucción por medio de

un procedimiento de isosurfacing. El algoritmo de este tipo conocido como *marching cubes* a menudo requerirá una cantidad de evaluaciones en el orden de las decenas o cientos de millones en caso de desear una visualización de alta calidad. Por ello es importante contar con un mecanismo de evaluación de la SDF que sea eficiente. La naturaleza local de MPU también presenta una ventaja en este aspecto, ya que el valor de la SDF en un punto sólo depende de la reconstrucción local de la celda de la subdivisión en la que cae y de las cercanas.

Si f es la función calculada por el método, recordemos que por construcción $f(x) \approx \sum_i \phi_i(x) Q_i(x)$, donde Q_i es la aproximación local de la i-ésima celda. Por la definición de las ϕ, esto equivale a $f(x) \approx \frac{1}{\sum_{j=1}^{n} w_j(x)} \sum_i w_i(x) Q_i(x)$. Recordemos además que las funciones de peso sólo tienen soporte en el radio de la esfera centrada en la celda a la que corresponden, y valen 0 fuera de ese soporte por definición.

De esta manera, la evaluación de la función en un punto se traduce en ubicar las celdas cuyo soporte contiene al punto, realizar la evaluación de la SDF local de cada una y promediar dichos valores de manera ponderada por los pesos locales correspondientes. No hace falta hacer ninguna evaluación en ninguna otra celda, ya que las funciones de peso de celdas cuyo soporte no contiene al punto darían 0 cuando se evalúan en él. Las evaluaciones locales tienen tiempo constante al tratarse de funciones cuadráticas cuyos parámetros ya han sido hallados. La ubicación de las celdas, por otra parte, se hace en tiempo logarítmico en el peor caso, aprovechando la estructura de octree en base a la cual se realizó la subdivisión. Puede verse un pseudocódigo de este procedimiento al final de la sección.

4.5 Variante interpolante

Los autores de MPU proponen una manera de convertir en interpolante a una implementación existente del método aproximado original. Esto se hace cambiando el criterio de subdivisión de manera de asegurar que cada hoja del octree contenga a lo sumo un solo punto. Luego, se definen soportes centrados en cada celda de manera similar a la versión aproximada, se realizan reconstrucciones mediante polinomios cuadráticos bivariados en cada una y se evalúa la SDF resultante usando partición de la unidad por medio del mismo mecanismo de exploración del octree de subdivisión.

Por su parte, las funciones de peso cambian y pasan a ser las de *inverse distance weighting*, las mismas del método interpolante de Franke y Nielson. De hecho, podemos pensar a esta variante del método de MPU como una adaptación del mismo al Modified Shepard propuesto por Franke y Nielson; en efecto, lo que resulta es un método interpolante local que calcula un polinomio por cada punto de la nube y que promedia esos resultados individuales en uno global mediante *inverse distance weighting*.

4.6 Pseudocódigo

Los algoritmos basados en este framework de subdivisión y partición de la unidad pueden esquematizarse en el siguiente pseudocódigo generalizado.

```
reconstruirGlobal(celda):
    subdividir? ← procesarCelda_A(celda)
    si subdividir? entonces:
        hijos ← subdividir(celda)
        para h en hijos:
            reconstruirGlobal(h)
    si no:
        guardarReconstruccionLocal_A(celda)

evaluarGlobal(x, celda):
    suma ← 0
    pesoTotal ← 0
    evaluarGlobalRec(x, celda)
    devolver suma/pesoTotal

evaluarGlobalRec(x, celda):
    si enSoporte?(x, celda) entonces:
        si esHoja?(celda):
            w ← pesoLocal(x, celda)
            v ← evaluarLocal(x, celda)
            suma ← suma + w × v
            pesoTotal ← pesoTotal + w
        si no:
            para h en hijos(celda):
                evaluarGlobalRec(x, h)
```

El único procedimiento abstracto en este esquema es *procesarCelda*, marcado apropiadamente en el pseudocódigo, que deberá tener una implementación distinta de acuerdo al algoritmo concreto del que se trate. Los métodos de obtención y consulta de los hijos de una celda y *esHoja?* son propias de una estructura general de octree. Por su parte *evaluarLocal* simplemente evalúa a la SDF local asociada a la celda en cuestión de manera agnóstica a su procedencia.

Para el método de MPU, *procesarCelda* deberá llevar a cabo todos los pasos relativos a la decisión de subdivisión: determinar el soporte de la celda, realizar una reconstrucción local y calcular el error de aproximación producido. El método *guardarReconstrucciónLocal*, que se ejecutará en el caso de no subdividir la celda, simplemente se encarga de conservar la SDF computada en el procesamiento de la celda y asociarla a la celda en cuestión, que quedará como hoja del octree. Es una importante característica de este esquema de implementación el hecho de descartar las reconstrucciones intermedias hechas para celdas que se subdividen, evitando guardar reconstrucciones más gruesas de las deseadas de acuerdo a los criterios de subdivisión que se establezcan.

4.7 Análisis del parámetro de error

En el paper original del método, los autores sugieren un parámetro de error de 10^{-4} como manera de asegurar una captura fiel de detalles finos. Sin embargo, se comprue-

ba empíricamente que este parámetro de error muy bajo no permite al algoritmo, según criterios de evaluación cualitativa, suavizar lo suficiente la reconstrucción ante la presencia de ruido. Como consecuencia, se obtienen reconstrucciones de una calidad cualitativa menor a la deseable. Esto puede ocurrir debido a que, a pesar de tratarse de un método aproximado, un parámetro de error tan exigente fuerza que la reconstrucción se ajuste demasiado a los datos ruidosos y no alcance a suavizar esos errores.

Además, en principio, al realizar una reconstrucción de una superficie determinada con este algoritmo y un parámetro de error en particular, puede no quedar claro hasta qué punto ese error realmente puede lograrse. En teoría, es posible ejecutar el algoritmo con un parámetro tan exigente que no pueda satsifacerse en absolutamente ninguna localidad de la subdivisión que se realiza, cortándola el algoritmo siempre en los casos base alternativos como el de profundidad máxima o celdas con un solo punto.

Con estas observaciones en mente, realizamos un análisis estadístico del comportamiento de la subdivisión del algoritmo ante distintos parámetros de error para nuestro conjunto de datos de Reconbench. El objetivo de este análisis es poder sacar conclusiones acerca de la capacidad del algoritmo de cumplir con un parámetro de error exigente como el sugerido en el paper para nuestro conjunto de datos, y eventualmente elegir un parámetro que se amolde mejor al uso intencionado en este trabajo.

4.7.1 Parámetro original

Exhibimos en la figura 4.4 los resultados del análisis estadístico aplicados a la superficie Daratech. Los resultados para el resto de las superficies de prueba de Reconbench son muy similares y pueden encontrarse en el anexo. Los gráficos incluídos muestran información acerca del comportamiento de la subdivisión y el cumplimiento del parámetro de error de 10^{-4} que tiene el algoritmo cuando se corre para una de las entradas del conjunto de nubes de puntos para Daratech.

Para el primer gráfico se muestra la proporción de las localidades de la subdivisión en los que se alcanzan en la práctica determinados niveles de error de aproximación. Los rangos de error incluidos en el gráfico cubren el 99.9 % de las hojas de la subdivisión con errores asociados más bajos. De acuerdo a este resultado, solamente el 29.9 % de las localidades alcanzan errores menores a 10^{-4}. Es decir que aproximadamente el 70.1 % de las localidades del algoritmo tienen un error mayor al requerido por el parámetro. En términos de performance, esto también sugiere que en ese porcentaje alto de casos se fueron realizando muchas reconstrucciones y consecuentes subdivisiones que nunca iban a poder alcanzar el nivel de error pedido y terminaron la subdivisión en el caso base de un solo punto.

En el segundo gráfico se muestra información acerca de la distribución de los niveles de altura del octree a los que se encuentran las hojas resultantes de la subdivisión realizada. Concretamente, se indica el mínimo, el máximo y la mediana de niveles del octree correspondientes a las hojas que caen en cada rango de error de aproximación de los definidos para el gráfico anterior. Vemos niveles del octree que se encuentran mayormente entre 5 y 12 con una mediana de 8, y algunos peores casos de profundidad de 13, 14 y 15. Para poner en perspectiva estos valores, notemos que octrees completos de estas profundidades tienen cantidades de hojas de 8^{13}, 8^{14} y 8^{15}, valores que se encuentran en el orden de 10^{11}, 10^{12} y 10^{13} respectivamente. Tener en memoria, por ejemplo, 8^{15} hojas donde cada hoja ocupa sólo un byte, requeriría aproximadamente 35 terabytes de memoria.

Las observaciones anteriores son compatibles con lo que muestra el tercer gráfico,

dando la proporción de hojas de la subdivisión por cada cantidad de puntos inicial que contiene su soporte previo a expandirlo. Las cantidades de puntos iniciales que aparecen en el gráfico cubren el 99.9 % de las hojas. En el 96.8 % de los casos se llegó a tener un solo punto inicialmente en la localidad, alcanzando el criterio de corte de la subdivisión que se activa cuando no se puede cumplir localmente con el error de aproximación requerido. Notemos que las cantidades de puntos iniciales de las hojas son muy chicas en la enorme mayoría de los casos, aún dejando de lado los de un solo punto, resultando por ende en reconstrucciones locales numerosas y con muy poca información acerca de la superficie. Este hecho puede ayudar a explicar la sobre-adaptación mencionada a las nubes ruidosas que le fueron proporcionados como entrada al algoritmo en estas pruebas, observable en 4.5.

4.7.2 Parámetro alternativo

Realizando pruebas con una variación granular de parámetros de error, se elige 2×10^{-3} para realizar el resto de las reconstrucciones con MPU que se harán en el marco de este trabajo. Este valor es el más bajo que cumple con criterios de evaluación cualtitativa aplicados a mallas resultantes de reconstruir todas las figuras de Reconbench con alguna de sus nubes. Observando la figura 4.5 se puede hacer una comparación cualitativa entre las reconstrucciones presentadas antes para el parámetro de 10^{-4} y las realizadas para 2×10^{-3}. Como referencia, se agrega en la figura 4.6 una comparación similar para la superficie Gargoyle, incluyendo también una reconstrucción utilizando un parámetro poquito más permisivo de 3×10^{-3}.

Tomando 2×10^{-3} como parámetro de error, también llevamos a cabo el análisis realizado antes para el parámetro de 10^{-4}, presentado en la figura 4.7. En este caso, vemos resultados mucho más favorables. En primer lugar, el 99.2 % de las localidades hojas presentan un error de aproximación que cae dentro del requerido. Además, se evidencia que el algoritmo logra cumplir con este parámetro requiriendo una menor profundidad del octree. De manera compatible con esta última observación, se observa en el tercer gráfico una distribución muy distinta de hojas en función de la cantidad inicial de puntos de su soporte. Esta nueva distribución significa que las reconstrucciones locales utilizadas por el algoritmo para generar la global se hacen con porciones de mayor tamaño y por ende más representativas de la superficie. Por último, también es positivo tener mayores cantidades de puntos iniciales ya que así se evita una gran cantidad de operaciones de expansión de soportes.

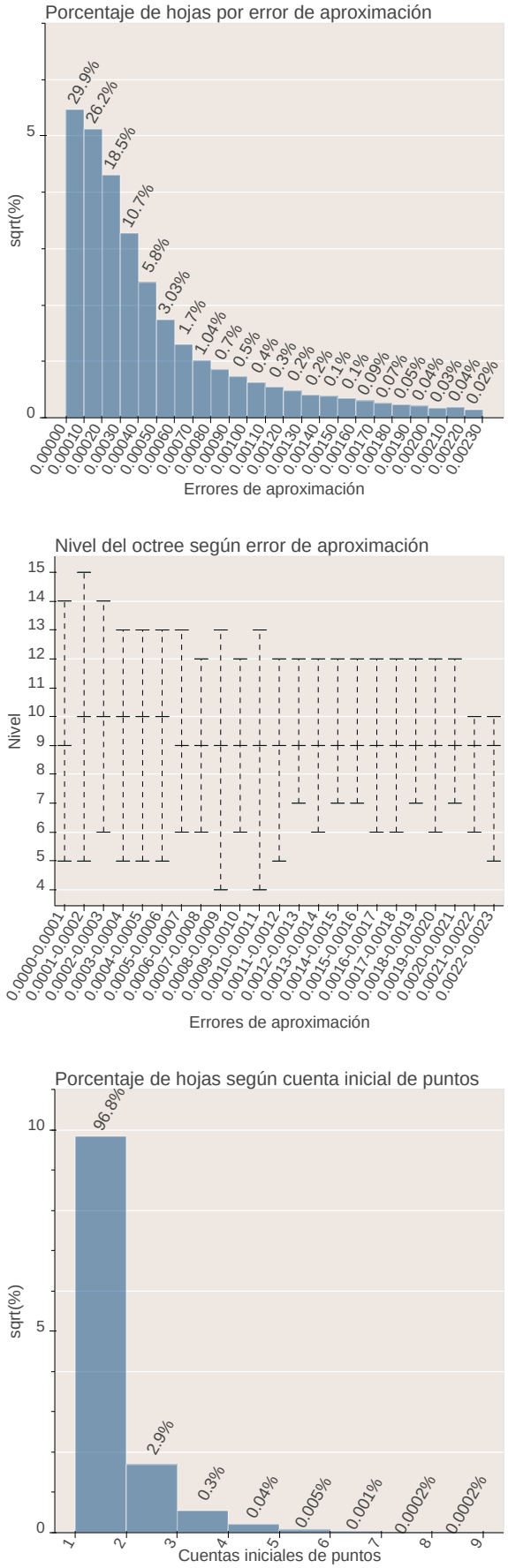


Fig. 4.4: Análisis de subdivisión para Daratech con parámetro de error 0.0001.

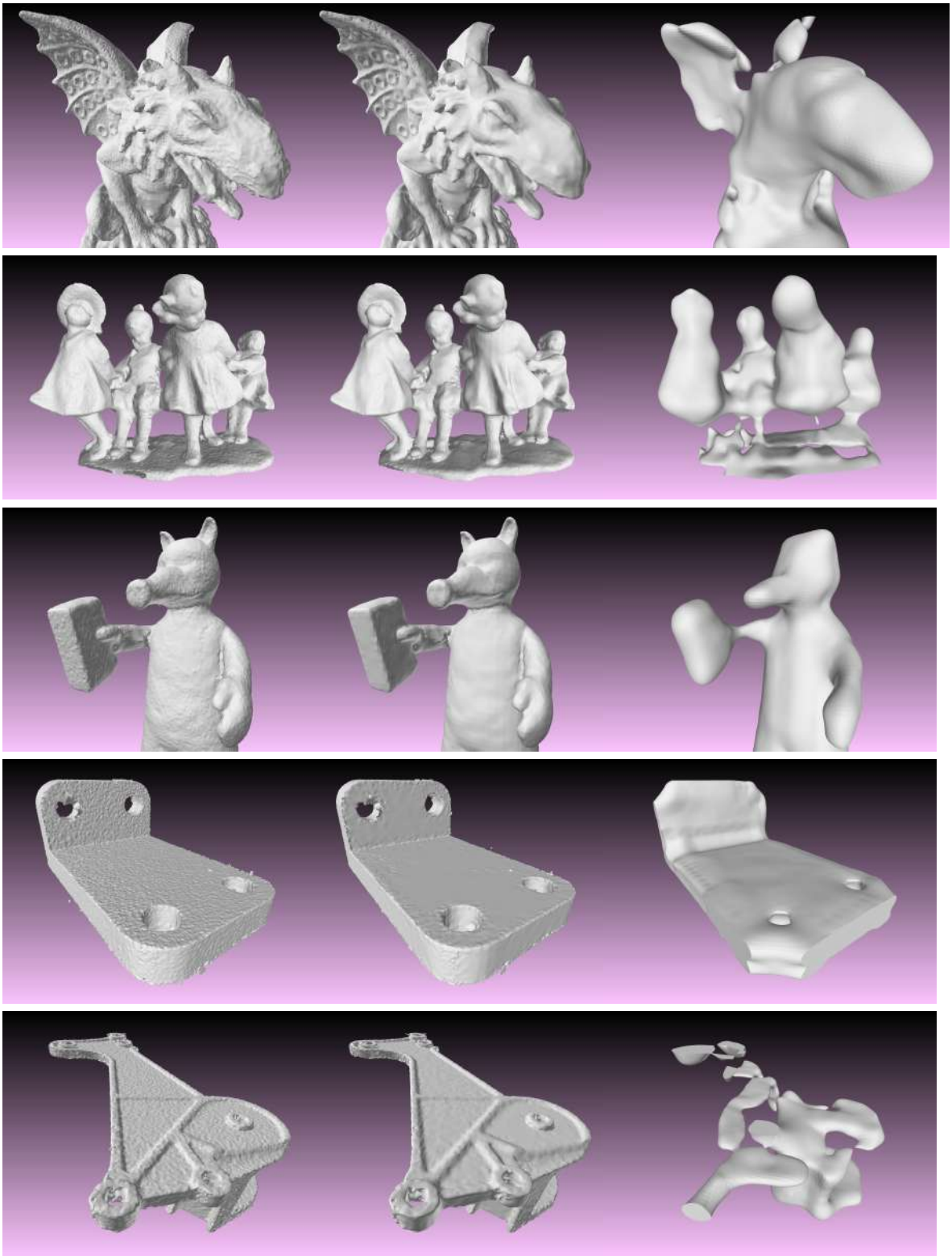


Fig. 4.5: Comparación de reconstrucciones hechas usando el parámetro original de 10^{-4} (columna izquierda) con reconstrucciones hechas utilizando el alternativo 2×10^{-3} (columna del medio). Como referencia, se agregan reconstrucciones con un parámetro particularmente laxo de 5×10^{-2}. La diferencia en calidad de reconstrucción se ve claramente en muchas regiones de todas las superficies, como por ejemplo en las láminas planas de Anchor y Daratech o en el objeto sostenido por Quasimoto. A pesar de ser más suaves, las superficies de nuestro parámetro alternativo aún conservan detalles finos como los círculos en las alas de Gargoyle o las caras de los niños en Dancing Children.

(a) Error máximo 10^{-4} *(b)* Error máximo 2×10^{-3} *(c)* Error máximo 3×10^{-3}

Fig. 4.6: Comparación de reconstrucciones de Gargoyle utilizando el parámetro de error original de 10^{-4}, el alternativo de 2×10^{-3} y uno un poco más permisivo de 3×10^{-3}. Tanto *(b)* como *(c)* suavizan el ruido observable en *(a)*, pero *(c)* ya pierde detalles notables como los círculos en el ala.

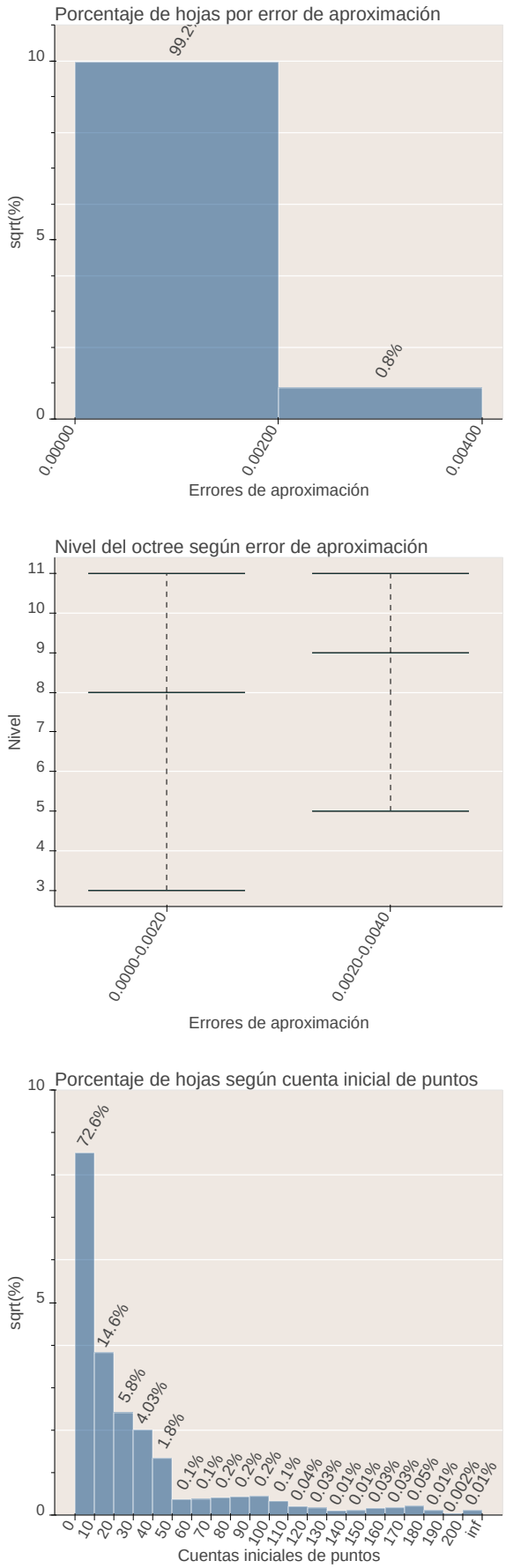


Fig. 4.7: Análisis de subdivisión para Daratech con parámetro de error 0.002.

PUNCH: NCH utilizando partición de la unidad

Recordemos que el algoritmo de NCH presenta una desventaja en cuanto a sus tiempos de ejecución, teniendo una complejidad cuadrática en su etapa de cálculo de SDF y lineal en cada evaluación que se hace de la SDF resultante.

Con la motivación de mejorar este aspecto del método de la NCH, se presenta en esta sección un algoritmo de reconstrucción nuevo que combina los enfoques de NCH y MPU. Dicha combinación da por resultado un algoritmo que, de manera similar al método NCH original, produce una SDF interpolante de la nube de puntos, pero la construye combinando reconstrucciones locales hechas con NCH. De forma parecida a MPU, se realiza la subdivisión de un cubo delimitador de la nube de puntos mediante un octree. En cada celda hoja resultante se calcula un soporte esférico centrado en ella, se confecciona la porción de la nube correspondiente a ese soporte y se estima la NCH de ese subconjunto de puntos. Luego, al evaluar la SDF, se combinan los valores de las celdas cercanas al punto de consulta utilizando funciones de peso adecuadas. De este modo, el algoritmo nuevo se amolda perfectamente al framework de subdivisión y partición de la unidad esquematizado en el pseudocódigo de la sección 4.6. En la sección 5.2 se completa el pseudocódigo de PUNCH. El objetivo de este nuevo enfoque es lograr la estimación de la NCH de una nube de puntos de una manera más eficiente en tiempos de ejecución, sin disminuir la calidad de las reconstrucciones resultantes.

Para hacer las veces de funciones de peso se eligen las de *inverse distance weighting* en base a algunos antecedentes ya mencionados: el método de Franke y Nielson es un algoritmo interpolante que combina interpolaciones locales mediante funciones de *inverse distance weighting*, y la versión interpolante del método de Ohtake et al.[24] también utiliza esta misma familia de funciones para realizar la combinación de interpolaciones locales. Cada celda de la subdivisión cuenta con su propia función de distancia inversa, parametrizada según el radio de su soporte en particular.

A su vez, el soporte esférico de una celda también se define inicialmente con un radio de $R = \alpha d$ donde d es la diagonal principal de la celda, pero en este caso $\alpha = 1{,}25$. La elección de este valor de α se basa en el precedente puesto por [24], donde se usa este valor para la variante interpolante de su método en lugar del valor de 0,75 configurado para su método aproximado original. Este precedente se justifica teniendo en cuenta que también elegimos usar las mismas funciones de peso que en la variante interpolante de [24] (en lugar de las funciones de peso basadas en la función de B-spline cuadrática), dado que el radio de los soportes determina la cantidad de SDFs locales que serán combinadas usando dichas funciones en la evaluación de la SDF global en un punto del espacio.

Definido todo lo anterior, lo único que resta determinar para este nuevo algoritmo es el criterio según el cual se subdividen las celdas del octree. Esto se discute en la siguiente

sección.

5.1 Criterio de subdivisión y parametrización

Como se ha explicado, la complejidad temporal de la etapa de SDF del algoritmo de NCH es cuadrática en la cantidad de puntos de la nube, y la de evaluación es lineal. Este orden de complejidad torna inviable para muchos usos prácticos la aplicación directa del algoritmo sobre nubes de tamaño medio; la reconstrucción de nubes de un tamaño un orden de magnitud mayor que las del conjunto de Reconbench, por ejemplo, puede llevar varias horas. Como manera de mitigar este hecho, se plantea en PUNCH un criterio de división diseñado para controlar el detrimento de eficiencia producido por esa complejidad. Considerando que la misma es directamente función de la cantidad de puntos sobre la que se corre NCH tanto en su etapa de cálculo de la SDF como en su etapa de consulta, este control se basa en proveer garantías acerca de la cantidad de puntos que resultará en cada localidad donde se corra el algoritmo de NCH.

De este criterio se desprende la necesidad de definir algunos parámetros. Supongamos inicialmente una implementación básica en la que se toma como entrada un *máximo* para la población de puntos admisible por localidad. Para cumplir con ese valor, el algoritmo subdividiría una celda dada si la cantidad de puntos contenidos en su soporte esférico es mayor a ese máximo y pararía la subdivisión en celdas cuyo soporte contiene una cantidad de puntos menor o igual a él. Tener este máximo nos provee un control sobre el costo individual de cada corrida local de NCH, pudiendo limitar la cantidad de puntos involucrada en su reconstrucción cuadrática y evaluación lineal.

Sin embargo, puede ser deseable también controlar el mínimo de puntos que queda por celda. Sólo imponer el máximo y dejar el mínimo libre puede generar mucha variabilidad en la cantidad de puntos final: con un máximo de 100, por ejemplo, puede quedar por celda cualquier cantidad de puntos entre 0 y 100. En la práctica esta variabilidad efectivamente se observa, y ocurre en mayor o menor medida según características en principio no-controladas de la nube como su distribución de puntos o la rotación en la que se encuentra la nube respecto de los ejes cartesianos. Este fenómeno supone una menor predicibilidad de los tiempos de ejecución de PUNCH, y celdas con pocos puntos pueden producir estimaciones locales y en consecuencia globales de menor calidad. Desde un punto de vista teórico, tiene sentido pensar que esto pueda pasar ya que, como se explica en detalle en la sección 3, el cálculo del semiespacio que se asigna a cada punto de la nube depende de todos sus otros puntos. Por ende, tomar una porción de la nube y correrle NCH sin dudas limita la información utilizada por el algoritmo para la definición de cada semiespacio. Cuanto más chica la porción, mayor será esta limitación y mayor será la cantidad de puntos afectada por ella.

Se motiva así el agregado de un parámetro adicional que define la *mínima* cantidad de puntos aceptaba por celda. Esta restricción se cumple consiguiendo primero tener en una celda dada una población menor al máximo indicado por el parámetro anterior de la misma manera y luego, si no cumple con el mínimo de puntos requerido, expandiendo su soporte esférico en la medida de lo necesario para alcanzar ese mínimo. La expansión se realiza iterativamente siguiendo el mismo procedimiento ya presentado en la sección 3: $R_i = R_{i-1} + \lambda R, i > 1$ siendo R el radio original de la celda y λ, a priori, un valor fijo elegido experimentalmente.

No obstante, es importante observar que una elección de mínimo y máximo cercanos,

si bien proveerá mayor control y predicibilidad sobre la ejecución del algoritmo, tendrá como consecuencia que más soportes deban ser expandidos. Así, más celdas dispararán el proceso iterativo de incremento del radio de soporte y agregarían el costo en tiempos de ejecución correspondiente.

Asimismo, el nuevo parámetro de mínimo para las poblaciones de las celdas introduce un nuevo problema. Si λ no es lo suficientemente fino, puede ocurrir que ciertas combinaciones de máximo y mínimo sean imposibles de cumplir en algunos soportes. Por ejemplo, supongamos que debemos satisfacer un máximo de M puntos por soporte y un mínimo de m, y contamos con un parámetro de expansión λ_0. Sea una celda C de soporte esférico inicial de radio R_0 en la que caen $k < m$ puntos. Dicha celda cumple con tener menos de M puntos ya que $m < M$, pero debemos aplicar el procedimiento de expansión para conseguir una cantidad de puntos $\tilde{k}$ que cumpla $m \leq \tilde{k} \leq M$. Sin embargo, en principio, es perfectamente viable para algún $j > 0$ que los puntos contenidos en el soporte con $R_j = R_{j-1} + \lambda_0 R_0$ sumen una cantidad menor a m, y en el soporte con $R_{j+1} = R_j + \lambda_0 R_0$ sumen una cantidad mayor a M. En tal caso, podemos concluir o bien que esta combinación particular de parámetros de mínimo y máximo es simplemente imposible de cumplir y descartarla o manejarla con alguna heurística, o bien podemos concluir que la elección de λ_0 como parámetro de expansión no provee una granularidad suficiente para cumplir con las dichas restricciones sobre la población del soporte de la celda C. Las localidades en las que no se pueda satisfacer la restricción de M y m se denominan *casos imposibles*.

En la práctica, se comprueba que un valor de $\lambda = 0{,}1$ como el sugerido por Ohtake et al.[24] para su método produce una proporción alta de casos imposibles, e induce a buscar menores valores para este multiplicador. Por otro lado, cuanto menor sea su valor, más iteraciones y por ende más tiempo requerirá cada procedimiento de expansión, de los cuales habrá una mayor proporción cuanto más cercanos sean el máximo y el mínimo de poblaciones por celda que se exijan. Más adelante estudiamos el efecto de distintos valores para λ en presencia de combinaciones diferentes de máximos y mínimos.

En resumen, PUNCH se configura con tres parámetros: el máximo de población por celda (M), el mínimo de población por celda (m) y la tasa de expansión de soportes (λ). Si se encuentra un caso imposible durante la expansión de un soporte, el algoritmo corta la expansión y deja la población resultante de la primera iteración que se pasó de M.

Vale aclarar también que el agregado de puntos a cada localidad en el proceso de expansión no es "permanente": los soportes se expanden tomando "prestados" puntos vecinos para asegurar una determinada calidad de reconstrucción local, pero luego los radios resultantes de expandir los soportes de las celdas se descartan. Después de calculadas todas las localidades y a la hora de evaluar la SDF global en un punto, los radios que se utilizan para determinar qué reconstrucciones locales participarán de la evaluación son los originales, $R = \alpha d$ con $\alpha = 1{,}25$. Esta decisión también se hereda del mecanismo de partición de la unidad de MPU en el que se basa PUNCH.

5.2 Pseudocódigo

Haciendo referencia al esquema general de reconstrucción localizada por medio de un octree planteado en 4.6, para dar el pseudocódigo de PUNCH sólo debemos definir el método *procesarCelda*.

procesarCelda$(celda)$:
  $R \leftarrow \alpha d$
  $puntos \leftarrow confeccionarSoporte(celda, R)$
  si $|puntos| > M$ **entonces**:
    devolver verdadero // *se debe subdividir*
  si $|puntos| < m$:
    $puntos \leftarrow ajustarSoporte(celda, R)$ // *asegurar* $m \leq |puntos| \leq M$
  estimarNCH() // usando "puntos" como entrada
  devolver falso

donde d es la diagonal de la celda, M y m son los parámetros de máximo y mínimo respectivamente para la población de las celdas y *ajustarSoporte* se encarga de realizar la expansión iterativa del radio inicial R hasta cumplir con la condición de población. Luego, haciendo referencia nuevamente a 4.6, la función *evaluarLocal* realizará una invocación de *evaluarNCH* con cada punto con el cual se llame.

5.3 Análisis de parámetros

De acuerdo a lo explicado, este algoritmo se parametriza eligiendo un máximo M y un mínimo m para la cantidad de puntos de cada celda y un parámetro λ que determina la tasa de expansión de los soportes locales. En esta sección analizamos el comportamiento del algoritmo en función de distintos valores de estos parámetros, observando su performance en términos de tiempos de ejecución y de calidad de reconstrucción utilizando el pipeline de evaluación de Reconbench. El objetivo del análisis que sigue es doble: por un lado, estudiar distintos aspectos del comportamiento del algoritmo, y por otro elegir una configuración de los parámetros para llevar a cabo una comparación global de rendimiento entre los tres algoritmos estudiados en este trabajo. Los experimentos de tiempos de ejecución se realizan en procesadores Intel i7 4.20GHz 64-bit con procesos single-threaded. Todos los tiempos exhibidos son el promedio de 10 o más ejecuciones.

5.3.1 Variante de NCH

La discusión que compara entre las tres variantes de NCH en la sección 3.7 nos llevaría, en principio, a optar siempre por la variante positive. Sin embargo, si bien las observaciones realizadas al respecto siguen siendo válidas en contextos donde se desee aplicar NCH a una nube, tiene sentido preguntarse si las mismas se sostienen al querer correr NCH de manera localizada como se hace en PUNCH.

En la sección 3.7 se observa que tanto la NCH positiva como la negativa presentan artefactos en la reconstrucción a causa de ruido en la nube de puntos sobre la que se aplican, y que la variante simétrica, a pesar de ofrecer un intermedio entre la positiva y la negativa, no compensa satisfactoriamente los efectos indeseados de sus dos contrapartes. El motivo de esto último es que la variante negativa, al deber calcular esferas que quepan en el interior del cuerpo recontruido, produce radios mucho menores a los de las esferas correspondientes en la versión positiva.

Sin embargo, esta última limitación de la variante negativa se reduce dramáticamente en PUNCH, que corre NCH sobre subconjuntos de la nube de puntos y no sobre la nube entera. Suponiendo una subdivisión en localidades lo suficientemente granular, es posible

imaginar cómo, en una gran mayoría de casos si no en todos, el algoritmo de NCH se correría en cada localidad desconociendo los puntos que se encuentran del lado opuesto del cuerpo en la nube completa. Es decir, la NCH negativa se computaría sin contemplar los puntos de la nube que limitarían el radio de las esferas que elige para sus semiespacios.

Es en este contexto que el intermedio ofrecido por la variante simétrica cobra mayor utilidad, generando un intermedio entre esferas que presentan una distribución de radios menos despareja, estando una de las variantes mucho menos limitada que antes. En estas nuevas condiciones, la variante simétrica puede permitirnos tomar ventaja de la naturaleza geométricamente complementaria de los artefactos generados por la positiva y la negativa: dado un punto para el que se define un semiespacio de radio chico a causa de ruido, la negativa colocaría una esfera "saliente" de la frontera del cuerpo (produciendo así su "protuberancia"), mientras que la positiva colocaría una esfera "entrante" (produciendo así su "pozo"). Asumiendo que el radio de las esferas es similar, como es más razonable hacer dada la localización de la aplicación de NCH que realiza PUNCH, optar por la opción simétrica nos provee un intermedio entre la "protuberancia" y el "pozo", construcción de la que intuitivamente se esperan resultados cualitativamente más satisfactorios que los provistos por las otras dos variantes.

Dada esta discusión, las reconstrucciones que se realicen con PUNCH de ahora en adelante se harán todas utilizando localmente la NCH simétrica. Vale notar que no hemos atendido en estas consideraciones la cuestión de cuán granular debe ser la subdivisión para eliminar la limitación mencionada a la variante negativa en una proporción razonablemente alta de localidades. No obstante, como hemos anticipado, en lo que sigue de esta sección se presenta un análisis cuantitativo de la calidad de reconstrucción de PUNCH en función de distintos valores para sus parámetros. Entre esos parámetros se encuentra M que ejerce un control directo sobre la subdivisión. Entonces, de ser importante el impacto mencionado aquí de la granularidad de subdivisión, es razonable esperar que el mismo se vea reflejado en los resultados que siguen.

5.3.2 Máximo de población por celda

En primer lugar, analizamos el comportamiento del algoritmo en función de distintos valores para M. Es interesante tomar la variación de este parámetro como un primer eje de experimentación ya que tiene una influencia particularmente importante en el funcionamiento del algoritmo. A diferencia de los otros dos, este parámetro determina la subdivisión que se hará en cada ejecución: el octree se subdivide hasta satisfacer en cada celda una población de cantidad de puntos menor o igual a M y no se modifica más, estando el mecanismo de expansión a tasa de λ encargado de cumplir con el mínimo m. En otras palabras, dado un valor concreto de M, el algoritmo construye un único octree independientemente de m y λ; estos dos últimos parámetros influyen luego en qué soporte queda definido para cada hoja de ese octree.

En la figura 5.1 se presentan datos acerca del rendimiento del algoritmo ante diferentes valores de M en materia de tiempos de ejecución, tanto de la etapa de cálculo de la SDF como la de isosurfacing con Marching Cubes. Para asistir al análisis de estos resultados, se grafican además de los tiempos otras dos características de las ejecuciones correspondientes de PUNCH. En los figuras de la etapa de cálculo de la SDF se grafica la cantidad de hojas resultante en el octree de subdivisión para cada valor de M. Por su parte, en los gráficos de la etapa de isosurfacing se agrega para cada valor de M el promedio de la cantidad de soportes en los que caen los puntos en los que Marching Cubes evalúa la SDF producida

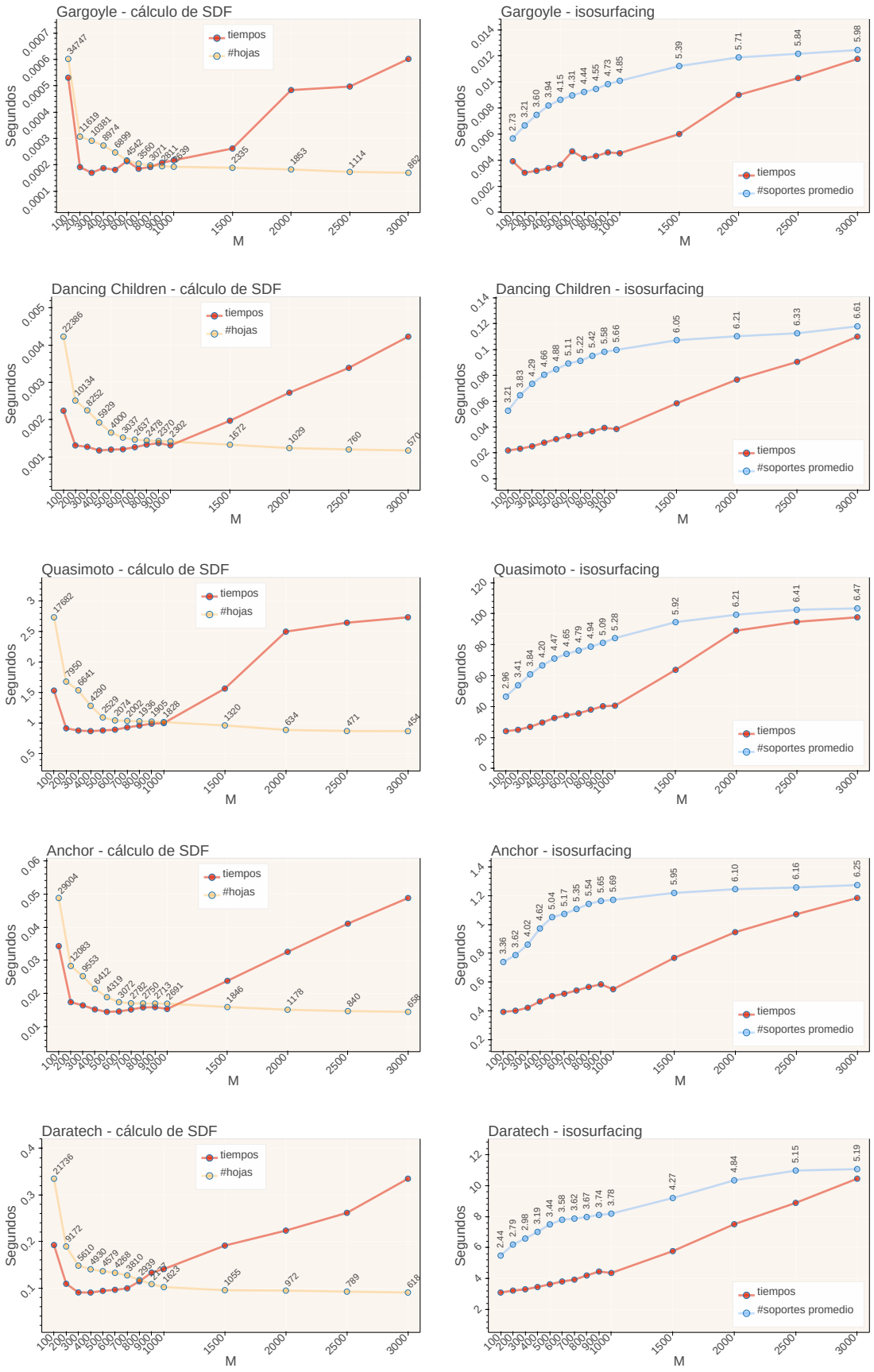


Fig. 5.1: Tiempos de cálculo de la SDF y de isosurfacing usando PUNCH con distintos máximos de población por celda. Isosurfacing con una resolución de 150^3.

en la etapa previa. El sentido de estos datos adicionales se entenderá en la discusión que sigue.

En estas ejecuciones se deja fijo m en 5, un valor muy chico cuya elección prácticamente equivale a que el mínimo para las poblaciones de las celdas esté completmanete libre. Se elige ese valor como alguna cantidad arbitraria chica pero mayor a 1 para asegurar que tenga sentido correr NCH en las localidades que resulten de la subdivisión. Se asegura empíricamente que en ninguna de las ejecuciones incluídas en estos resultados se disparan expansiones de soportes, por lo que puede dejarse de lado por el momento ese aspecto del costo computacional del algoritmo.

La variación de tiempos de cálculo de SDF de PUNCH en función de M es muy consistente entre las distintas superficies de nuestro conjunto de prueba: para valores chicos de M los tiempos son relativamente altos, luego bajan rápidamente hasta alcanzar un mínimo global y finalmente pasan a crecer paultainamente con el aumento de M. Este comportamiento se puede explicar desde la teoría considerando la variación de las características de la subdivisión en función del cambio de M. Cuando el límite superior de puntos por celda es muy bajo, el criterio de subdivisión de PUNCH produce una cantidad alta de hojas con pocos puntos cada una, llevando a que se calculen SDFs locales por NCH en localidades muy chicas pero en una cantidad muy numerosa. En cambio, cuando el límite superior de cantidad de puntos por celda es muy alto, se produce una cantidad de hojas órdenes de magnitud más baja pero con muchos más puntos cada una, debiendo luego realizarse una cantidad chica de corridas de NCH sobre poblaciones mucho más grandes. Este comportamiento se ve reflejado en los gráficos de la figura 5.1 que muestran la cantidad de hojas por valor de M. Como se hace evidente en dichos gráficos, en los valores chicos de M los tiempos de SDF están controlados por la elevada cantidad de corridas locales de NCH que se deben hacer más que por el tamaño de la entrada de cada una, mientras que para valores altos de M las poblaciones crecen tanto que el tamaño de la entrada de cada corrida de NCH pasa a tener mayor influencia en el rendimiento que la cantidad baja de localidades. Importante en este hecho es la complejidad temporal de NCH, cuadrática en el tamaño de su entrada. Esta variación encuentra consistentemente un balance en valores de M aproximadamente entre 300 y 700. Para ejemplificar este análisis, podemos tomar el caso de Gargoyle en el que calcular la SDF con $M = 100$, que requiere realizar 34747 corridas locales de NCH, lleva un tiempo casi igual que hacerlo con $M = 2500$ donde se requieren 1114 corridas de NCH, una cantidad 30 veces menor que con $M = 100$. Al mismo tiempo, en el mínimo global de tiempos de ejecución correspondiente a $M = 300$, el algoritmo tarda aproximadamente 3 veces menos que con $M = 100$ o $M = 2500$.

Las pequeña variabilidad en la tendencia general de variación de los tiempos, como la manifestada en el caso de Gargoyle con los pequeños picos locales en $M = 400$ y $M = 600$, es esperable considerando que, al prácticamente estar libre m, los tamaños de poblaciones de celdas que efectivamente se dan en cada ejecución es variable y sujeta a factores en principio no-controlados como la geometría de la figura reconstruida y distribución de los puntos de su nube.

Por su parte, el rendimiento de la etapa de isosurfacing también es muy consistente entre las distintas superficies de prueba. A medida que M aumenta, así lo hacen también los tiempos de isosurfacing. A diferencia de la etapa de cálculo de la SDF, sin embargo, el crecimiento de tiempos de esta etapa es prácticamente monótono. Veremos que esto también es coherente desde el punto de viste de la teoría, de la que se desprende un

crecimiento casi lineal.

Recordemos que esta fase de la reconstrucción se lleva a cabo por medio del algoritmo de Marching Cubes. En estas pruebas, todas las ejecuciones se realizan con una misma grilla de $150 \times 150 \times 150$. Esto significa que, en todos los casos, se disparan exactamente las mismas consultas de valores de la SDF global computada en la etapa anterior de la reconstrucción. Además, recordemos que cada una de estas consultas se resuelve en PUNCH buscando, en primera instancia, todas las celdas en cuyo soporte cae el punto de consulta y luego consultando las SDFs locales correspondientes a cada uno de estos soportes para combinar sus valores mediante las funciones de peso y devolver el resultado final. La búsqueda de soportes se ejecuta recorriendo el octree de subdivisión desde la raíz hasta las hojas que correspondan, operación de tiempo acotado por el logarítimo del tamaño de la nube, mientras que el cálculo del valor del punto de consulta en cada localidad lleva un tiempo lineal en la cantidad de puntos del su soporte, siendo esta la complejidad de la evaluación de una SDF generada por una corrida de NCH.

Dado que es fija la cantidad de evaluaciones que hace Marching Cubes de la SDF global, si suponemos que tanto las alturas del árbol como la cantidad de soportes en las que caen los puntos de la grilla son independientes de M, se sigue que el crecimiento de los tiempos en función de M es lineal. Informalmente, podemos expresar este crecimiento como $O(G \times (logN + SM))$ donde G es la cantidad de evaluaciones hechas por Marching Cubes y S es una constante que acota superiormente a la cantidad de soportes en los que puede caer uno de los puntos evaluados.

Ahora bien, estas suposiciones no son estrictamente correctas. A medida que M crece, la altura del árbol disminuye dado que las hojas de la subdivisión incluyen cada vez más puntos. Por su parte, la cantidad de soportes en los que se incluye un dado punto del espacio ha de ser creciente en M dado que aumentan los tamaños de las celdas, y en consecuencia lo hacen los radios de los soportes esféricos que son proporcionales a la diagonal de su celda. Este fenómeno se visualiza en los gráficos de la etapa de isosurfacing de la figura 5.1 consultando los promedios de cantidad de localidades que soportan los puntos en los que se consulta la SDF.

Por un lado, la incidencia de la disminución en la altura del árbol como producto del crecimiento de M en los tiempos de ejecución del isosurfacing es negligible tratándose de logaritmos en base 8. A su vez, el aumento de la cantidad de localidades que soportan los puntos de la grilla de Marching Cubes podría ser mayor: no sólo suma a la cantidad de evaluaciones de SDFs que se realizan, sino que estas son lineales en poblaciones cada vez más numerosas. No obstante, si bien es creciente, el aumento en la cantidad de soportes también es prácticamente despreciable, subiendo de 2.44 en el caso más bajo a solamente 6.6 en el más alto mientras M se varía de 100 a 3000, un aumento del 30,000 %.

En el caso del isosurfacing también es importante seguir remarcando la variabilidad de las características de la subdivisión y los soportes que resultan de variar M dados las propiedades geométricas y de distribución no-controlados de las nubes, especialmente dejando libre m. En particular, ante la variación de M y los consecuentes cambios en la subdivisión, un mismo vértice de la grilla puede caer en cantidades de soportes que no crecen uniformemente con el incremento de M, y las poblaciones de esos soportes presentan cada vez más variación a medida que M y m distan cada vez más.

El otro aspecto importante a considerar del efecto de variar M es el de la calidad de reconstrucción. Para analizarla, sería interesante poder comparar la calidad entre todas las instanciaciones de M y todas las superficies de prueba que venimos utilizando. Entre

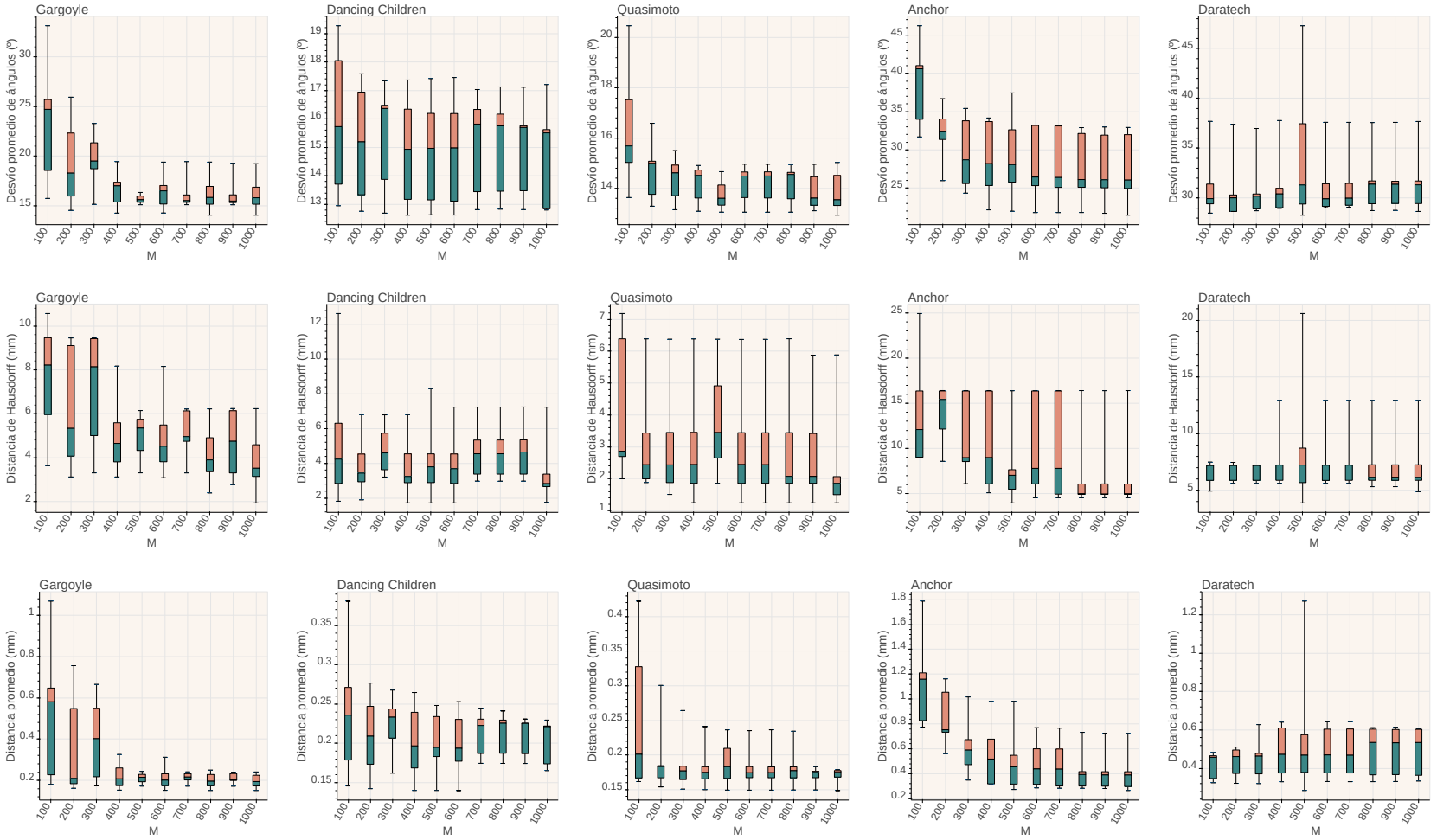


Fig. 5.2: Distribuciones de error en función de M para todas las superficies de Reconbench. Por fila: desvío promedio de ángulos, distancia de Hausdorff y distancia promedio (ver sección 2.2.2).

otros motivos, para comparar una cantidad tal de reconstrucciones es que resulta particularmente conveniente realizar una evaluación cuantitativa de su calidad que nos permita contrastar métricas numéricas asociadas a los resultados. Con tal fin, para esta comparación nos basamos en el pipeline de evaluación de Reconbench presentado en la sección 2.2. Las distribuciones de error exhibidas resultan de correr el pipeline con 10 nubes distintas para cada figura del conjunto de pruebas (Gargoyle, Dancing Children, Quasimoto, Anchor y Daratech).

En la figura 5.2 presentamos los resultados de nuestra comparación cuantitativa de la calidad de reconstrucción de PUNCH en función de una variación de los valores de M, nuevamente con $m = 5$. El rango de M se elige teniendo en cuenta que los tiempos de ejecución medidos en función de M tienden a dispararse después de 1000. Cada gráfico exhibe la distribución de valores de una métrica en particular en un conjunto de nubes de una superficie de Reconbench en particular. Para cada valor de M se grafica el valor mínimo, el máximo y los percentiles 25, 50 (mediana) y 75. Los gráficos están ordenados con una métrica por fila y una superficie por columna.

Como tendencia general, se observa una disminución en los valores de error con el aumento de M. Vale notar que la única superficie que no sigue este patrón es Daratech, donde los errores se mantienen casi quietos e incluso aumentan ligeramente con M; fuera de ese caso, consistentemente se registran mejoras en todas las métricas a medida que M aumenta. En más detalle, a lo largo del rango de máximos entre 100 y 500 se observan las mejores más acentuadas, estableciéndose luego las medidas de error en valores parecidos. Sólo en Anchor y en Quasimoto para las métricas de distancia se tienen mejoras consistentes en rangos más altos de 800 a 1000. Esta falta de respuesta de Daratech a la expansión de soportes puede explicarse por su forma mucho más "plana" que el resto de las superficies que se extienden más a lo largo de las tres dimensiones (ver figura 2.2) y son por ende más susceptibles a cambios en los volúmenes donde se realizan las reconstrucciones locales.

El comportamiento observado es consistente con la teoría, considerando que localidades con cantidades muy chicas de puntos reducen la cantidad de información disponible en cada reconstrucción local y aumentan proporcionalmente la participación de las funciones de peso en el resultado global, funciones que existen no para aportar a la geometría de la reconstrucción sino sólo como herramienta para mitigar los efectos de la partición del dominio en localidades. Ahondando en el punto de vista teórico de este comportamiento, también es razonable esperar que la calidad de reconstrucción "converja", o al menos reduzca considerablemente su pendiente de mejora, a partir de cierto tamaño que alcancen las celdas. Para explicar lo último, podemos plantearlo en términos de extremos. Si utilizamos localidades de ínfimo tamaño que contienen dos o tres puntos cada una, obtendremos muchísimas reconstrucciones muy carentes de información cada una y con una excesiva participación de las funciones de peso. En cambio, si ponemos M lo suficientemente grande, podemos obtener una sola localidad que contiene a todos los puntos de la nube, y nuestra SDF resultante será exactamente la misma que correr el algoritmo NCH original sobre la nube entera. Trivialmente, este último caso es el de la mejor calidad de reconstrucción posible, considerando que nuestra meta con PUNCH es mejorar los tiempos de ejecución manteniendo, mínimamente, la misma calidad de reconstrucción. Así, vemos claramente cómo valores muy chicos de M darían una calidad de reconstrucción muy inferior a valores muy grandes de este parámetro.

Con el fin de elegir un valor concreto de M para utilizar en los experimentos que sigan,

debemos tener en consideración además el comportameiento del algoritmo en términos de tiempos de ejecución. En referencia al análisis correspondiente, recordamos que los tiempos alcanzaban un rango mínimo para valores de M entre 300 y 700. Incentivados a elegir el menor valor posible de este rango por la tendencia creciente general de los tiempos y contemplando las observaciones hechas recién sobre los resultados en la figura 5.2, fijamos M en 500.

5.3.3 Mínimo de población por celda

Habiendo fijado M en 500, procedemos a estudiar el comportamiento de PUNCH cuando se varía m. Por su parte, λ se deja en el mismo valor fijo que antes. En esta experimentación, en la que vamos acercando m a M, sí empiezan a darse casos en los que deben expandirse los soportes de algunas hojas para satisfacer la configuración requerida de M y m. Dicho eso, se garantiza que el valor de λ fijado es suficientemente chico para que no existan casos imposibles.

En la figura 5.3 se presentan los tiempos de PUNCH en la etapa de SDF y la de isosurfacing con $M = 500$ para diferentes valores de m. Estos valores se eligen de modo de proveer mayor granularidad en los valores más chicos y los más altos del rango total de posibles m, naturalmente comprendido entre 0 y 500. En estos resultados se observa un crecimiento casi monótono de los tiempos a medida que m aumenta.

Consideramos primero la etapa de SDF. Recordamos que, al estar fijo M, la subdivisión que realiza PUNCH es exactamente la misma en todos los casos. En particular, esto quiere decir que la cantidad de localidades es fija para todos los valores de m. Por ende, tenemos una cantidad fija de reconstrucciones locales, siendo lo único que varía la cantidad de puntos sobre los que se realiza cada una. A medida que aumenta m, crece también la cantidad de localidades en las que la cantidad de puntos del soporte inicial es menor a m y debe realizarse una expansión del soporte para cumplir con ese mínimo. De esa forma, la cantidad de cómputo que debe llevarse a cabo de un valor de m al siguiente está determinada por la cantidad de soportes que tienen que expandirse, ya que esta es la cantidad de reconstrucciones locales que pasan de realizarse sobre una determinada cantidad de puntos menor a m a hacerse sobre una cantidad de puntos mayor o igual a m. Corroboramos esta correlación graficando en 5.3 para cada caso el crecimiento de la proporción de soportes que deben ser expandidos. La forma superlineal del crecimiento de los tiempos se explica por el hecho de ser las reconstrucciones cuadráticas en sus poblaciones.

En cuanto a los tiempos de isosurfacing, al igual que en los experimentos en función de M, tenemos una cantidad fija G de evaluaciones de la SDF que se ejecutan. Estando fijo M y por ello el octree resultante del proceso de subdivisión, la cantidad de soportes en los que cae cada punto evaluado por Marching Cubes es fija. Sin embargo, sigue siendo creciente en función de m la cantidad de puntos sobre la que se construyen las SDFs que se evalúan. Como evaluar una SDF es lineal en su cantidad de puntos, crecen los tiempos de isosurfacing de manera aproximadamente lineal. La variabilidad de los tiempos y el aspecto no estrictamente lineal de su crecimiento se explica por la ya discutida naturaleza variable de las poblaciones de las celdas dado cada par de valores de M y m. En particular, dado un punto concreto en el que Marching Cubes evalúa la SDF global y las celdas en las cuales se evalúan SDFs local para devolver ese resultado, de un valor de m a otro puede ocurrir que algunas de esas celdas aumenten la cantidad de puntos sobre los que se corre su reconstrucción local y otras no.

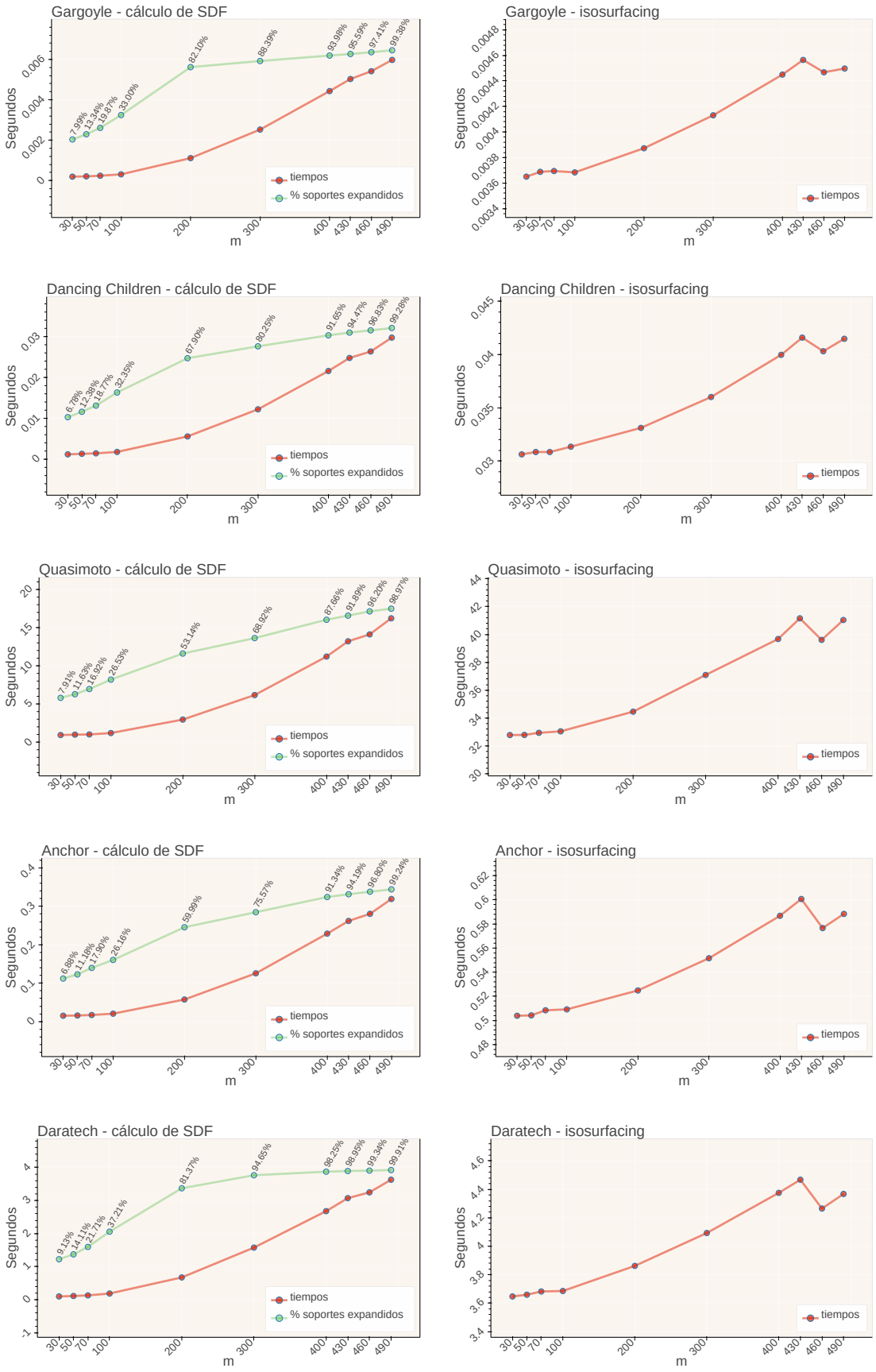


Fig. 5.3: Tiempos de cálculo de la SDF y de isosurfacing usando PUNCH con distintos mínimos de población por celda y un máximo fijo de 500. Isosurfacing con una resolución de 150^3.

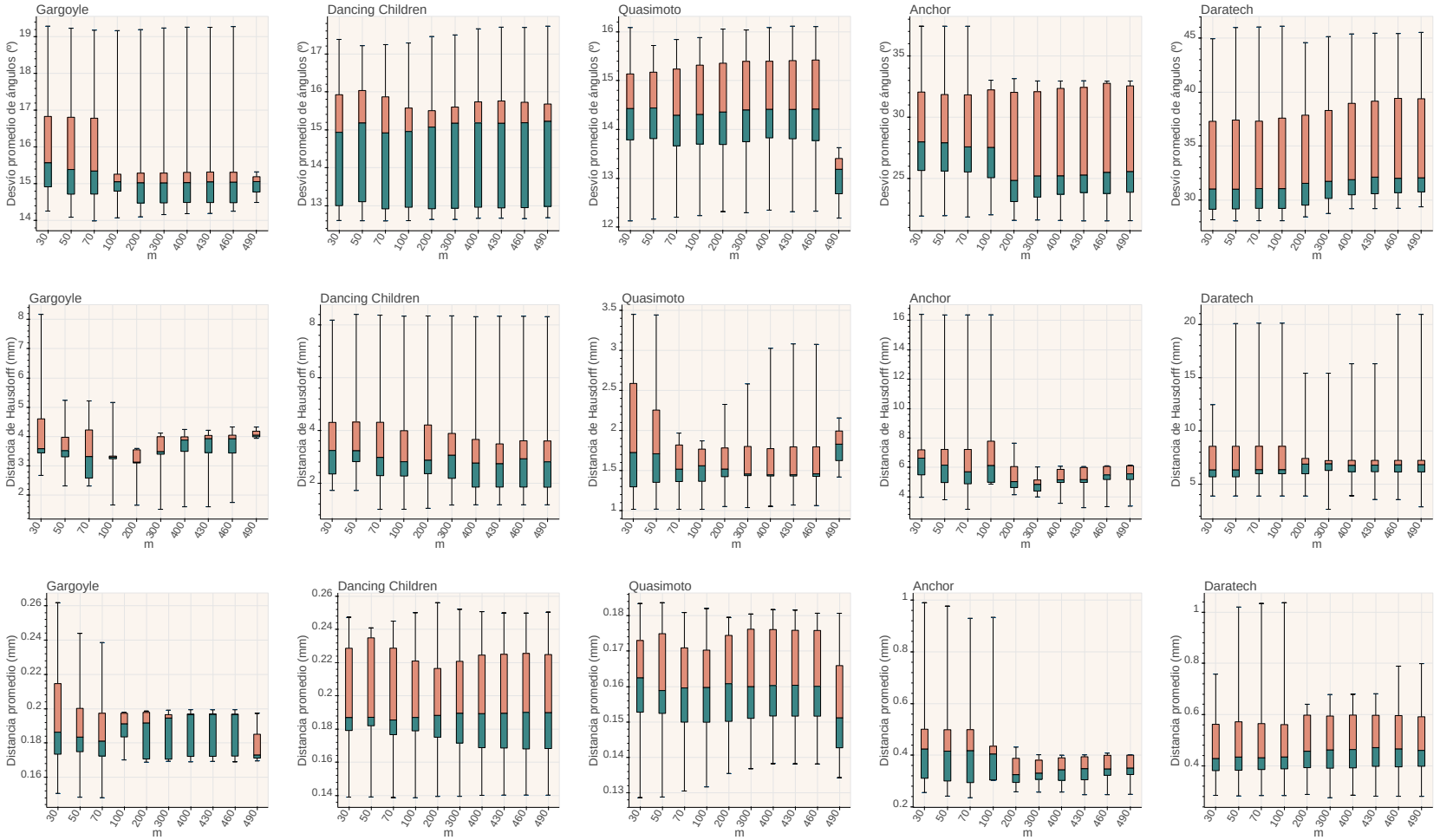


Fig. 5.4: Distribuciones de error en función de m con $M = 500$. Por fila: desvío promedio de ángulos, distancia de Hausdorff y distancia promedio (ver sección 2.2.2)

De manera similar a lo hecho para M, analizamos también la variación de la calidad de las reconstrucciones de PUNCH para diferentes valores de m en base a la evaluación cuantitativa de Reconbench. En la figura 5.4 exhibimos los resultados correspondientes a este análisis. Los mismos demuestran consistentemente reducciones en las métricas de error para cerca de $m = 100$ y $m = 200$. Que existan valores bajoes de m a partir de los cuales mejora la calidad de reconstrucción es compatible con nuestro análisis de calidad para el parámetro M, donde planteamos que celdas con poblaciones demasiado chicas implican reconstrucciones locales realizadas con poca información y un aumento en la participación de las funciones de peso. Fuera de las mejoras alrededor de los de $m = 100$ y $m = 200$, es difícil discernir una clara tendencia general de la variación de las calidades de reconstrucción en función de m. Esto puede deberse a que los cambios en las reconstrucciones locales que se producen dados los diferentes valores de m no alcanzan a exhibir cambios significativos al tratarse de un rango relativamente chico de valores.

Con el fin de seleccionar un valor de m, tenemos la motivación del rendimiento de tiempos de ejecución para elegir el menor valor que nos provea una calidad aceptable. Más aún, cuanto menor sea m, más lejos estará de M y menor será la probabilidad de debe acudir a expansiones de soporte que agregan tanto al tiempo requerido para calcular la SDF como a las cantidades de puntos sobre las que se hacen reconstrucciones locales y a la cantidad de soportes en los que debe evaluarse un punto donde se consulte la SDF en la etapa de isosurfacing. Con esto en consideración, tomamos $m = 100$.

5.3.4 Tasa de expansión de soportes

Para tener visibilidad de la interacción de λ y m cuando $M = 500$, graficamos en la figura 5.5 la proporción de soportes que suponen casos imposibles para cada combinación de esos dos parámetros. Los resultados mostrados son del caso de Anchor; el resto de los casos exhiben un comportamiento muy similar, y los gráficos correspondientes a ellos se agregan en el anexo. Variamos λ entre 0.001, valor elegido experimentalmente, y 0.1, el sugerido por los autores de MPU para la expansión de soportes de su método. Por su parte, variamos m entre 250 y 500. No mostramos resultados para ms menores ya que para ninguno de esos valores se dan casos imposibles en nuestro conjunto de datos utilizando λs del rango elegido. En el mapa de calor se observa el comportamiento esperable, dándose una mayor proporción de casos impsoibles a medida que λ y m aumentan. Dicho de otra forma, en valores bajos de m, ciertas configuraciones elevadas de λ ya demuestran ser demasiado gruesas para cumplir con una cantidad de puntos mayor o igual a m pero menor o igual a 500, al mismo tiempo que en valores más altos de m se requeiren elecciones de λ muy chicas para cumplir con las restricciones impuestas de población.

La consideración para una elección de λ debe tener en cuenta el siguiente tradeoff. Un mayor λ implica que se requieran menos iteraciones de expansión para cumplir con el par seleccionado de m y M, pero que el resultado final de población de los soportes sea más variable por tener un λ más grueso que agrega una mayor cantidad de puntos en cada paso del procedimiento de expansión. Un menor λ toma más iteraciones para cumplir con m y M, pero provee mayor control sobre la cantidad de puntos que se termina teniendo en los soportes al ser el procedimiento de inclusión de puntos más fino. Dependiendo de lo que se considere más prioritario, es útil tener en cuenta información como la proporcionada por la figura 5.5, que permite elegir los valores de λ que evitan los casos imposibles por completo o en la medida que se considere necesario. Recordamos que el efecto indeseado de tener casos imposibles es el perder la capacidad de limitar las poblaciones sobre las

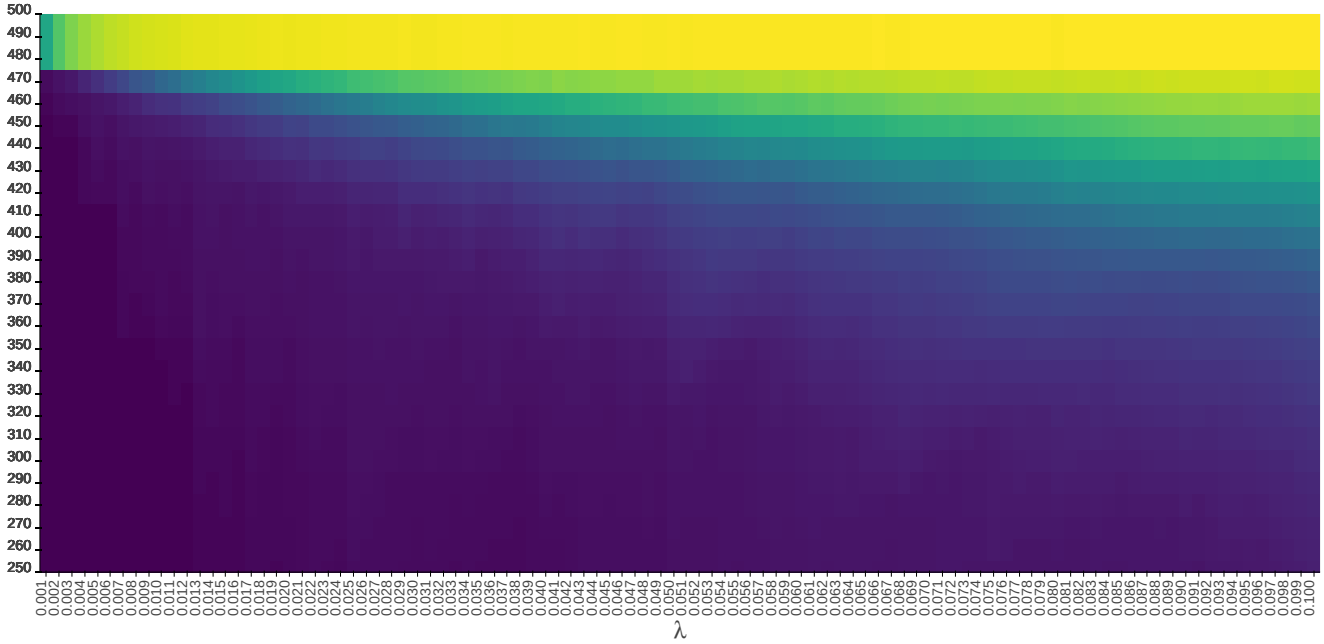


Fig. 5.5: Mapa de calor de la proporción de casos imposibles para distintas combinaciones de λ y m. Superficie: Anchor. Eje y: m. Escala de color: *viridis*. Para cada (λ, m) el valor corresponde a la raíz cuadrada del porcentaje de casos imposibles.

que se hacen las reconstrucciones cuadráticas de NCH, lo cual va en contra del objetivo fundacional de PUNCH que es controlar la complejidad cuadrática de NCH mediante un enfoque localizado.

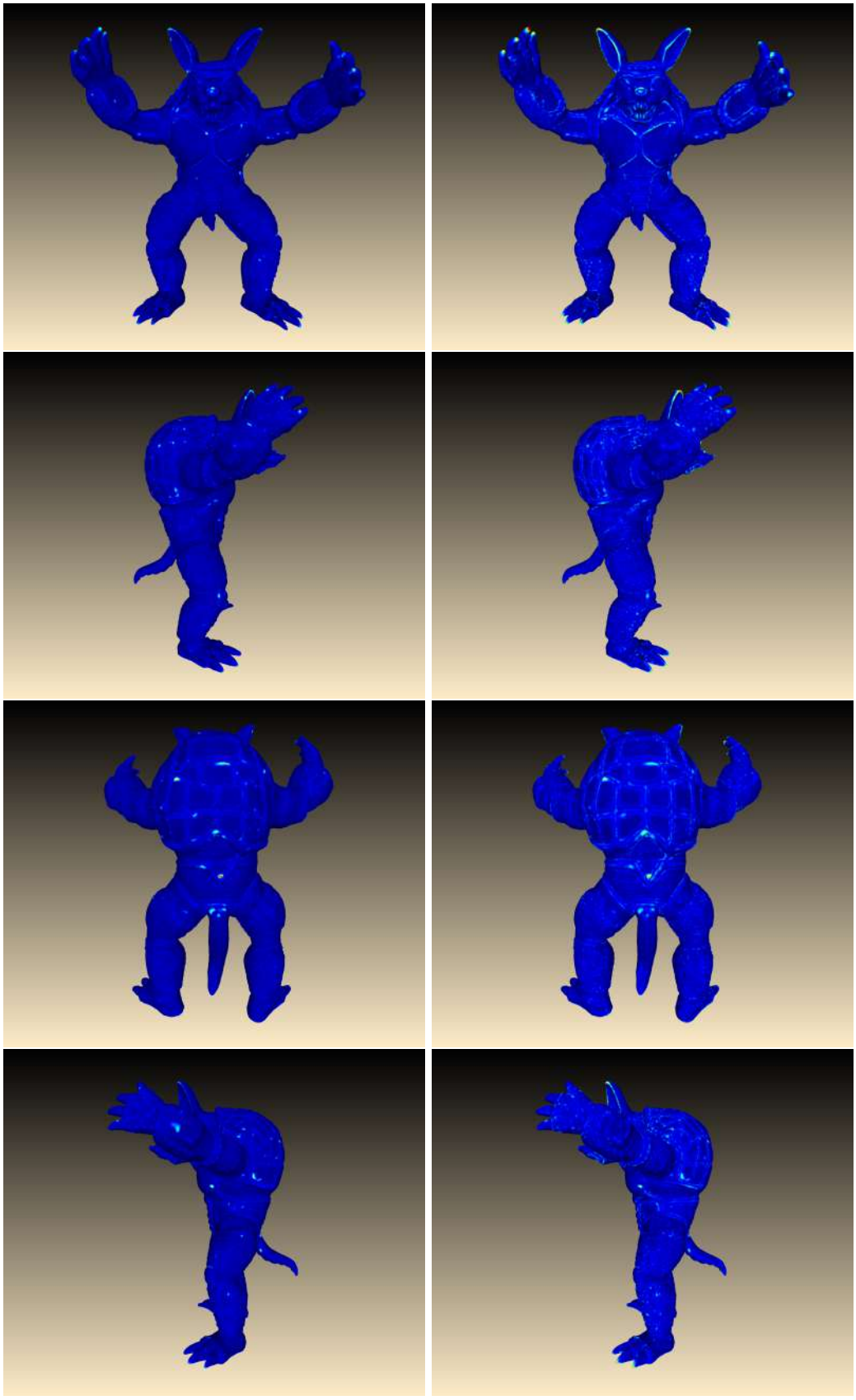
Para $m = 100$, nuestro mínimo elegido, como se comentó, no ocurren casos imposibles para ninguna de estas opciones de λ. Esto en principio podría permitirnos elegir un valor alto de λ que disminuya las iteraciones de expansión necesarias. Sin embargo, para el resto de nuestros experimentos, seguiremos utilizando $\lambda = 0{,}001$, que es el valor en el que hemos configurado todas las ejecuciones de PUNCH llevadas a cabo hasta aquí. Un tal valor relativamente bajo significa que serán necesarias más iteraciones en cada expansión y que su cantidad crecerá en función de m, pero nos dará mayores garantías de evitar casos imposibles al aplicar PUNCH a nubes nuevas por fuera de nuestro conjunto de pruebas principal; en un contexto distinto, podría priorizarse la reducción de la cantidad de iteraciones por expansión por encima de la evitación de casos imposibles y así optar por un λ más alto. Al mismo tiempo, es útil destacar que desde el punto de vista teórico el crecimiento de la cantidad de iteraciones es despreciable al lado del crecimiento del cómputo de reconstrucción causado por el aumento de m: el orden de cantidad de iteraciones necesarias para incluir k puntos expandiendo una esfera a una tasa de expansión fija es $O(\sqrt[3]{k})$, mientras que el tiempo de cómputo de la SDF para k puntos es $O(k^2)$.

Más aún, vale destacar que el costo agregado por las iteraciones de expansión está naturalmente limitado por la cantidad de expansiones que se realicen. Observando los resultados expuestos en la figura 5.3, vemos una proporción de soportes expandidos relativamente baja para $m = 100$, promediando el 31 % entre los cinco casos.

Por último, seguir utilizando la elección λ en 0.001 nos permite mantener el mismo control fino sobre las poblaciones locales que tuvimos hasta ahora, asegurándonos de no cambiar las localidades que produjo el algoritmo para los distintos casos de Reconbench y los valores de $M = 500$ y $m = 100$ que determinamos en esas pruebas. Dado que seguiremos utilizando las superficies de Reconbench para contrastar a PUNCH con otros algoritmos, es conveniente para una comparación justa no variar el comportamiento de nuestro algoritmo en cuanto a las características de sus reconstrucciones y su consecuente rendimiento ante las diferentes métricas de error que propone este framework de evaluación.

5.3.5 Complejidad temporal y espacial

Desde el punto de vista teórico, es difícil dar una expresión analítica de la complejidad del algoritmo, tanto temporal como espacial. Esto se debe a la naturaleza no-determinística de la subdivisión, la cantidad de puntos que quedaría en las celdas en cada caso y el nivel de superposición que se dé entre los soportes de acuerdo a los parámetros de control de poblaciones que se configuren. En cuanto a los tiempos, presentamos resultados empíricos y encontramos que se condicen con nuestra teoría del algoritmo. Respecto de la memoria, damos aquí algunos resultados de referencia y dejamos como trabajo futuro plantear una experimentación similar a la de los tiempos que estudie la variación del consumo en función de los parámetros de PUNCH. En la tabla 5.1 indicamos la cantidad de memoria pico consumida por PUNCH usando una nube de cada figura de Reconbench. Dicho pico siempre es alcanzado cuando ya se tiene computada en memoria toda la SDF.



PUNCH, $M = 2706$, $m = 541$ Screened Poisson

Fig. 5.6: Comparación entre reconstrucciones de Armadillo por PUNCH con M^+ y m^+ y Screened Poisson contra una malla de ground truth de Stanford.

Gargoyle	Dancing Children	Quasimoto	Anchor	Daratech
2.98GB	1.57GB	1.03GB	1.71GB	1.56GB

Tab. 5.1: Consumo pico de memoria de PUNCH para una nube de cada caso de Reconbench.

5.4 Reparametrización proporcional

En nuestra evaluación de performance de PUNCH, hemos contado principalmente con el framework cuantitativo de Reconbench y el conjunto de datos que ofrece. Hacerlo nos permitió sacar conclusiones acerca del comportamiento del algoritmo ante distintas configuraciones de sus parámetros.

Los tres parámetros tienen que ver con restringir las poblaciones que yacen en los soportes locales de la subdivisión con el fin de controlar el crecimiento cuadrático de los tiempos de reconstrucción de NCH. Asimismo, vimos experimentalmente en la sección 5.3.2 que estos parámetros sirven para encontrar un balance entre una subdivisión demasiado profunda, del tipo que lleva a reconstrucciones locales chicas pero muy numerosas, y una subdivisión demasiado superficial que conduce a hacer pocas reconstrucciones pero demasiado costosas. Vimos que ambos extremos significan un peor rendimiento en tiempos, y en el caso de la subdivisión profunda también tenemos evidencia parcial de una peor calidad en los resultados (según la figura 5.2).

Ahora bien, observemos que nuestra definición actual de estos parámetros es absoluta, en el sentido de no ser relativa de ninguna manera a la entrada sobre la que se corre el algoritmo. Reconstruyendo nubes cada vez más densas, el uso de parámetros fijos llevará necesariamente a que las localidades incluyan porciones de la nube cada vez más chicas. Dada la sospecha de que esto pueda influir negativamente en la calidad de reconstrucción y la certeza de que llevaría a una subdivisión profunda de las caracetrísticas que hemos reconocido como detrimentales a la eficiencia del algoritmo, es interesante pensar una manera de ajustar los parámetros de acuerdo a la densidad de la nube a la que se aplique. Recordemos que algoritmos basados en nuestro esquema de localización por partición de la unidad deben comenzar con una nube de puntos contenida en un cubo delimitador de diagonal 1; por ello, suponiendo una delimitación ajustada, al ser fijo el volumen de la nube la densidad es directamente proporcional a su cantidad de puntos.

Para estudiar esta idea de manera práctica, consideramos una nube de una cantidad de puntos un orden de magnitud por encima de las provistas por Reconbench. Se trata de una nube limpia, obtenida de muestrear una malla de referencia de la figura *Armadillo* ofrecida por Stanford Computer Graphics Laboratory[1] (SCGL), que contiene aproximadamente $330k$ puntos. Esto implica una densidad más de 5 veces mayor a la promedio de las nubes de Reconbench.

Proponemos un criterio inicial que relativiza la configuración de estos parámetros como función de la densidad de la nube de entrada. El objetivo de este criterio es lograr una subdivisión de profundidad similar a la que determinamos deseable en los casos de Reconbench. El promedio de cantidades de puntos de las nubes de Reconbench sobre las que llevamos a cabo aquella evaluación cuantitativa que resultó en la elección de $M = 500$ y $m = 100$ es de $61k$. Calculamos una estimación de la proporción que suponen estos

[1] https://graphics.stanford.edu/

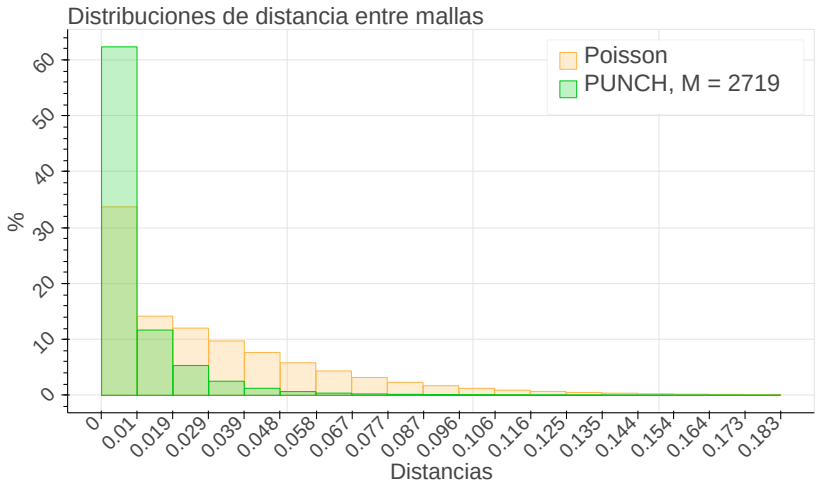


Fig. 5.7: Distribuciones de distancia entre mallas con respecto al ground truth de SCGL.

M y m concretos en relación a aquellas nubes, definiendo $\mu_M = 500/61000 = 0{,}0082$ y $\mu_m = 100/61000 = 0{,}00164$. Luego, utilizamos μ_M y μ_m como multiplicadores que permiten obtener la configuración de M y m para nubes nuevas. En el caso de Armadillo, que cuenta con $330k$ puntos, tenemos $M^+ = \mu_M \times 330000 = 2706$ y $m^+ = \mu_m \times 330000 = 541$. Vale destacar que, al ser M, m y el tamaño de la nube todas medidas volumétricas, i.e que representan todas una cantidad repartida en un espacio de tres dimensiones, es válido en el sentido algebraico tomar este enfoque de *reparametrización proporcional*.

Por su parte, en este criterio de extensión de los parámetros de PUNCH a nubes de otras densidades, el parámetro de λ no requiere modificación. Al no cambiar el volumen en el que se realiza la subdivisión, cada paso de expansión con este mismo λ agrega la misma cantidad de volumen a los soportes que expanda hasta cumplir con los M y m que se eligen. Al ajustar M y m proporcionalmente a la densidad de la nube, no tenemos motivos para pensar que pueda hacer falta dar pasos de expansión más o menos granulares para satisfacer la nueva configuración de parámetros.

Aplicamos PUNCH con M^+ y m^+ a Armadillo. Corremos también sobre la misma nube el algoritmo estándar *Screened Poisson* [19] (de ahora en más *Poisson*), como referencia para evaluar la calidad de reconstrucción de PUNCH reparametrizado. En la figura 5.6 mostramos comparaciones por superposición de PUNCH y Poisson contra la misma malla de referencia provista por SCGL de modo de tener cierta evaluación acerca de la calidad del resultado. En un principio, podemos considerar exitoso el resultado de PUNCH viendo que se ajusta satisfactoriamente a la malla de Stanford y lo hace incluso algo mejor que Poisson juzgando por las distancias entre mallas observables en el resultado.

Con el fin de sustentar esta comparación en una evaluación cuantitativa, presentamos además en la figura 5.7 las distribuciones de distancia entre mallas correspondientes a las reconstrucciones de la figura 5.6. Vemos reflejado en las distribuciones de distancia el comportamiento que exhiben las comparaciones por superposición de la figura 5.6 donde en Poisson se manifiestan mayores valores de distancia que PUNCH con M^+ y m^+.

En la tabla 5.2 damos datos acerca de la subdivisión resultante de la reparametrización en comparación con los tomados de una corrida de PUNCH con los parámetros anteriores.

	Hojas en total	**Soportes expandidos**	**Casos imposibles**
$M = 500, m = 100$	22694	6940	0
M^+, m^+	3533	1158	0

Tab. 5.2: Datos acerca de la subdivisión en las aplicaciones de PUNCH a Armadillo.

Es interesante observar cómo en Armadillo obtenemos con $M = 500$ una cantidad de hojas en el orden de las que tuvimos para $M = 100$ en las figuras de Reconbench según evidencia la figura 5.1, mientras que la cantidad de hojas resultante de usar en Armadillo los parámetros M^+ y m^+ es comparable con las que la figura 5.1 indica para $M = 500$. Es decir, con M^+ y m^+ logramos efectivamente una subdivisión de granularidad similar a la que obtuvimos con nuestros parámetros seleccionados en los casos de Reconbench. Esta observación corrobora en principio nuestro criterio de reparametrización proporcional.

Por último, aportamos en la figura 5.8 datos sobre el funcionamiento de la reparametrización en forma de un heatmap análogo al de la figura 5.5. En este caso, proviene de aplicar PUNCH a Armadillo con M^+ mientras se varía m entre valores ampliados proporcionalmente en comparación con los de la figura 5.5. Notamos el mismo patrón y valores comparables de ocurrencia de casos imposibles que en las nubes de Reconbench, al igual que la nulidad de casos imposibles que indica 5.2 en la nueva combinación de máximo y mínimo habiendo mantenido $\lambda = 0{,}001$.

En cuanto al efecto de la reparametrización en la calidad de reconstrucción, presentamos en la figura 5.9 una comparación de las distribuciones de distancia entre mallas que contrapone las dos parametrizaciones de PUNCH. Los datos del gráfico muestran una distribución muy similar entre las dos configuraciones, incluso con una ligera diferencia a favor de la original. Interpretamos esto como evidencia de que la reparametrización no perdió calidad de reconstrucción; el hecho de que no la haya ganado amerita un estudio más extensivo. Una posibilidad es que, a medida que aumenta la densidad de la nube, bajan los valores de límites de población debajo de los cuales comienza a perderse precisión.

Cualquiera sea el caso, seguirá siendo importante considerar el tradeoff de recursos computacionales, recordando que parametrizaciones con límites proporcionalmente chicos pueden llevar a subdivisiones detrimentalmente granulares. Asimismo, no debe dejarse de lado el hecho de que esta reparametrización produce límites de población crecientes en función de la densidad de la entrada, por lo que a partir de cierta cantidad de puntos de la nube se manifestará cada vez más el carácter cuadrático de la complejidad de las NCHs locales, esto es, la desventaja cuyos efectos la subdivisión justamente se propone contener. En última instancia, ante el prospecto de aplicar PUNCH a nubes arbitrarias, se podrá contar con el criterio de reparametrización proporcional como una herramienta más en la consideración de todos estos factores.

5.5 Comparación con NCH y otros algoritmos

A continuación comparamos el comportamiento de PUNCH con otros algoritmos de reconstrucción. Con el fin de analizar el cumplimiento del objetivo de PUNCH de mejorar los tiempos de ejecución de NCH sin perder de su calidad de reconstrucción, presentamos en primer lugar un benchmark de tiempos de estos dos algoritmos. Luego, contrastamos resultados de PUNCH y NCH cualitativamente. Por último, damos una comparación

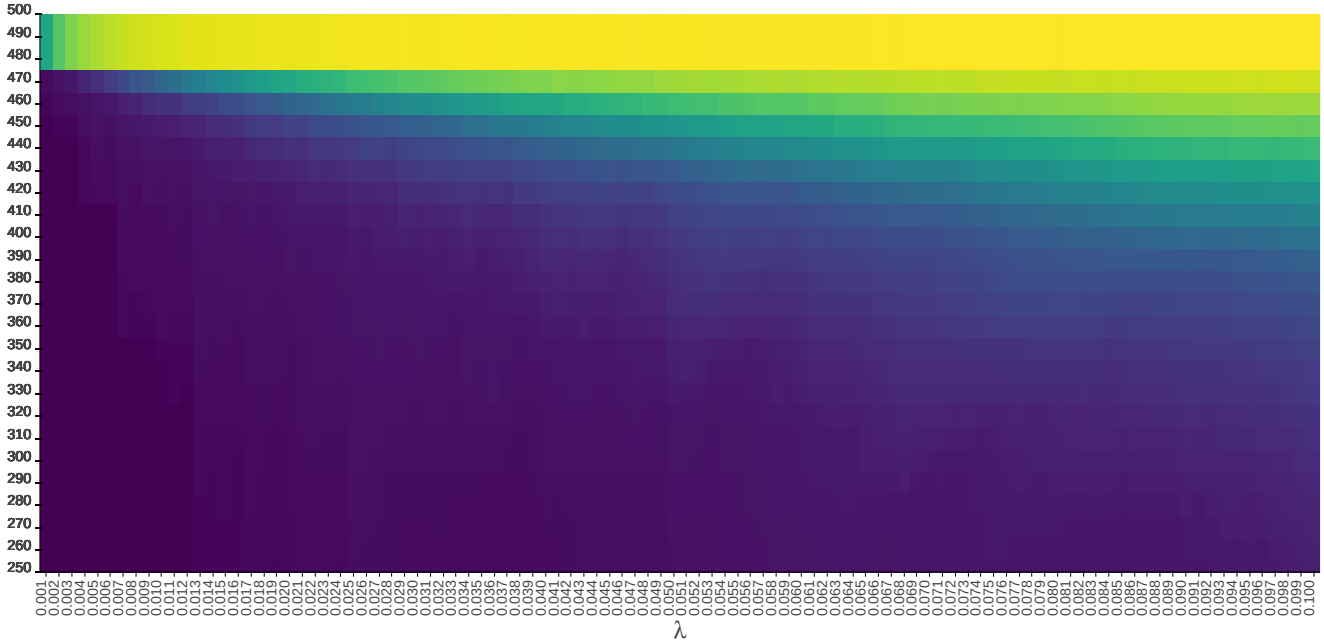


Fig. 5.8: Mapa de calor de la proporción de casos imposibles para distintas combinaciones de λ y m. Superficie: Armadillo. Eje y: m. Escala de color: *viridis*. Para cada (λ, m) el valor corresponde a la raíz cuadrada del porcentaje de casos imposibles.

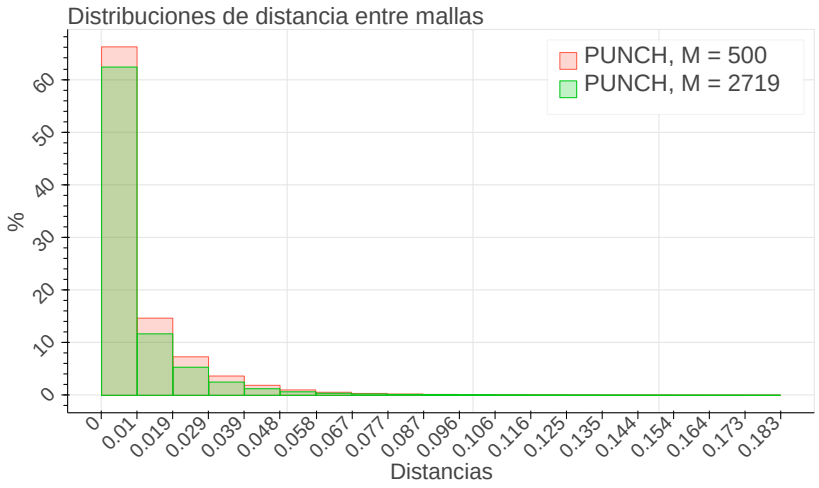


Fig. 5.9: Distribuciones de distancia entre mallas con respecto al ground truth de SCGL.

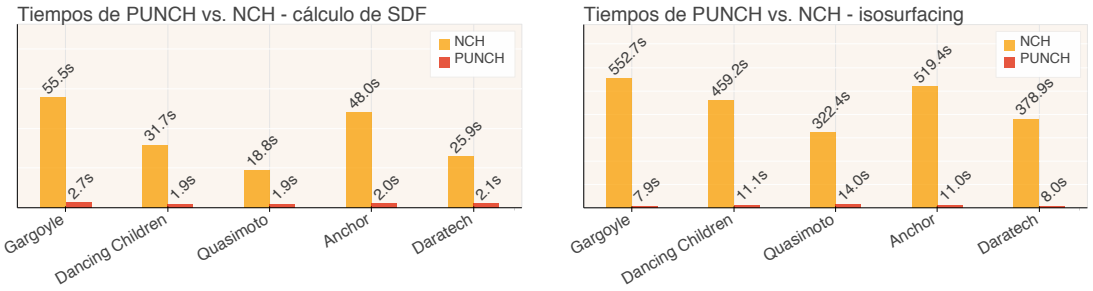


Fig. 5.10: Comparación de tiempos de ejecución entre PUNCH y NCH para una nube de cada una de las superficies de prueba de Reconbench. Los tiempos de isosurfacing son para una grilla de Marching Cubes de 100^3.

cuantitativa que abarca a estos dos algoritmos, MPU y Poisson.

5.5.1 Comparación de tiempos de ejecución con NCH

Recordando que el objetivo principal de PUNCH es tratar los elevados tiempos de ejecución del algoritmo de la NCH, tiene sentido comparar el rendimiento de estos dos algoritmos. Habiendo elegido una configuración para los parámetros de PUNCH, corremos este algoritmo al igual que la NCH positiva para una nube de cada una de las superficies de prueba de Reconbench. Los resultados de esta comparación se muestran en la figura 5.10.

Como es de esperar, los tiempos de PUNCH son considerablemente más bajos que los de NCH, con diferencias de $\times 10$ hasta $\times 24$. En la etapa de SDF, la localización de ejecuciones del algoritmo de la NCH en poblaciones chicas y controladas por nuestra parametrización de PUNCH lleva a reducir el tiempo global de reconstrucción de manera significativa, no permitiendo que dispare los tiempos crecimiento cuadrático de las operaciones en cada localidad. Los tiempos de SDF de la NCH, en cambio, son puramente el cuadrado del tamaño de cada nube.

En la etapa de isosurfacing se da un comportamiento similar, con diferencias de $\times 23$

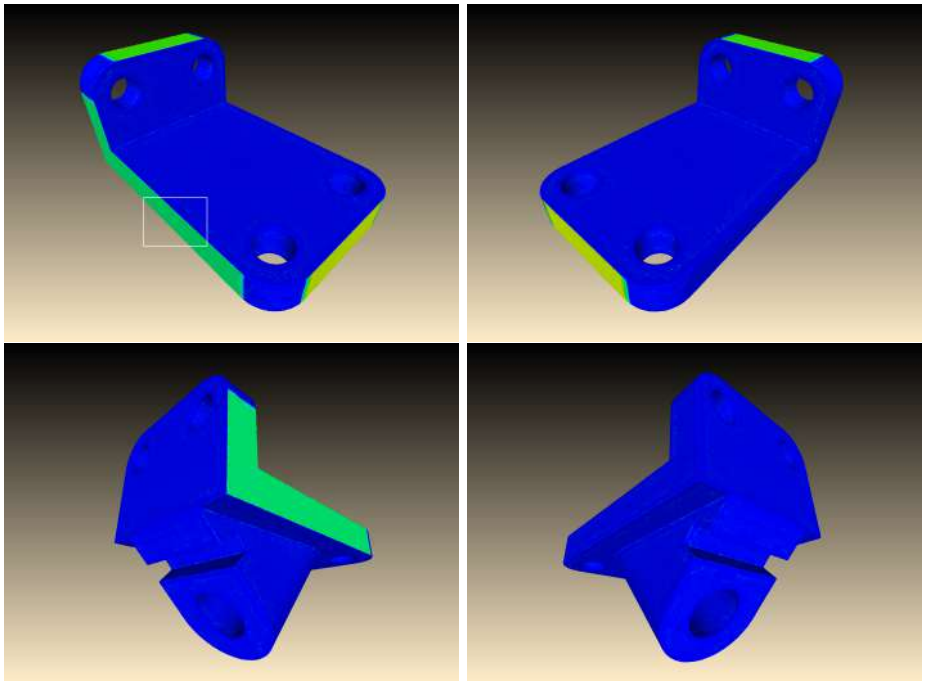


Fig. 5.11: Comparación entre reconstrucciones con PUNCH y NCH simétrica de una nube limpia de Anchor. En la primera imagen se señala la región enfocada en la figura 5.12.

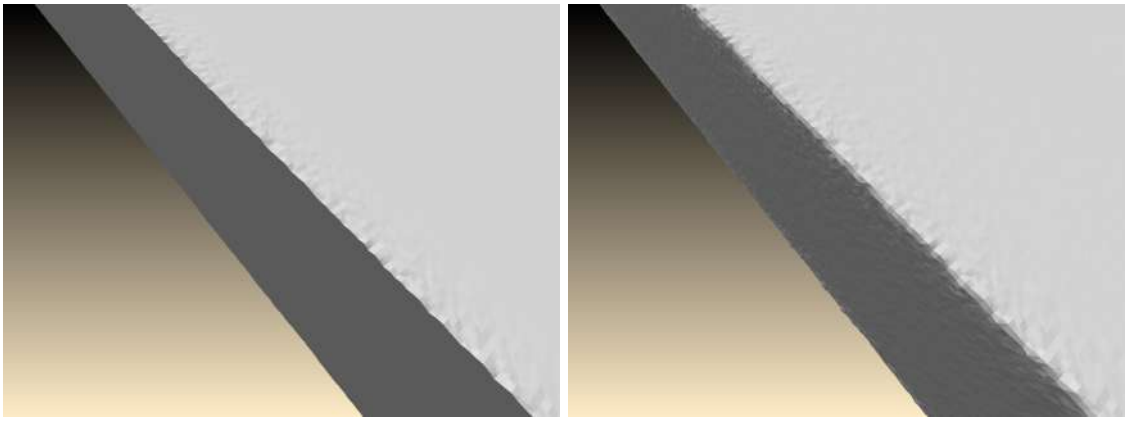
hasta $\times 70$. En esta etapa, el aspecto de estos algoritmos que entra en juego es el procedimiento de evaluación de su SDF en puntos del espacio. Al ser evaluada cada SDF en una cantidad muy elevada de puntos, en este caso en el orden de 10^3 al ser esta la resolución elegida para la grilla de Marching Cubes, es muy elevada la cantidad de veeces que entra en juego la diferencia de complejidad algorítimica favorable a PUNCH: cada evaluación de NCH es lineal en el tamaño de la nube entera, mientras que cada evaluación de PUNCH es lineal en el tamaño de una cantidad reducida de soportes locales. Debido a esto, las diferencias se amplían en esta etapa con respecto a la de cálculo de la SDF.

5.5.2 Comparación cualitativa con NCH

En la figura 5.11 mostramos una comparación por superposición de reconstrucciones de PUNCH y NCH de Anchor utilizando Metro. Elegimos para esta primera evaluación una nube limpia de esta superficie, proveniente de un sampleo del modelo de ground truth de Reconbench (sección 2.2.2). Con el fin de observar de la manera más directa posible los efectos de localizar la aplicación de NCH en comparación con correrlo globalmente, para este análisis en particular utilizamos reconstrucciones producidas por la NCH simétrica recordando que esta es la variante que elegimos para los resultados locales de PUNCH.

Salvo en determinadas regiones que saltan a la vista, podemos apreciar que las reconstrucciones son prácticamente iguales. En particular, este resultado no provee evidencia de efectos detrimentales de la combinación de reconstrucciones locales que, de existir, sería coherente que emerjan de esta comparación directa entre la versión local y la global.

Por su parte, llaman la atención las porciones de la superficie que exhiben distancias tan marcadas entre las dos reconstrucciones, especialmente dadas las similitudes entre todas estas regiones. Por un lado, todas tienen en común el ser láminas planas de la



PUNCH NCH simétrica

Fig. 5.12: Mayor zoom sobre una de las láminas planas con diferencias significativas de reconstrucción. Corresponde a la región señalada en la primera imagen de la figura 5.11.

superficie. Por otro, casi toda la superficie está formada por láminas planas, y la mayoría de ellas no exhiben esta distancia entre las reconstrucciones.

En la figura 5.12 hacemos zoom en una de las regiones que presentan este comportamiento, mostrando lado a lado las mallas resultantes de correr PUNCH y NCH. Se hace evidente que dicha lámina en el caso de PUNCH es más "correcta" que la producida por NCH, pareciéndose mucho más a una región planar, mientras que la de NCH presenta "perturbaciones" en la superficie.

Este comportamiento se explica desde la teoría haciendo referencia a un aspecto en particular de nuestra justificación de elegir la variante simétrica de NCH para la aplicación local en PUNCH (sección 5.3.1). Si bien el razonamiento detrás de dicha elección tiene que ver con mitigar los efectos de ruido en la nube, la observación central del razonamiento aplica en general a PUNCH independientemente del nivel de ruido en la entrada: en muchas localidades, la porción de la nube sobre la que se hace una reconstrucción local desconoce puntos de la nube que limitarían el radio de las esferas que utiliza. Por ende, el aspecto perfectamente planar que parece exhibir la lámina de PUNCH en la figura 5.12 se debe a que la subdivisión permitió a las corridas locales de NCH asignar esferas de radio infinito en sus semiespacios.

Vale recordar también que, a priori, la subdivisión que efectivamente resulta en cada caso depende de factores no-controlados como la distribución de los puntos y la orientación de la nube respecto de los ejes cartesianos. Considerando esto, tiene sentido que pueda no darse esta diferencia entre las mallas en láminas como, por ejemplo, la opuesta a la que se señala en la primera imagen de la figura 5.11. Notemos que con conocer sólo algunos puntos ubicados en las láminas perpendiculares ya podría alcanzar para limitar el radio de las esferas y dar resultados más parecidos a la NCH global.

En cuanto al resto de las láminas planas de Anchor, todas tienen en común una distinción clave respecto de las que sí presentan este tipo de diferencias con la reconstrucción global: distan mucho menos de su pared opuesta. Por eso, es mucho más probable que las localidades en las que caen incluyan puntos de esa lámina opuesta y que por ello los radios de sus esferas interiores se encuentren con la misma limitación que en la NCH global. Este interpretación es compatible con el modo en el que las distancias entre las mallas de la comparación se aminoran hasta desaparecer en cercanía de los cuatro agujeros superiores

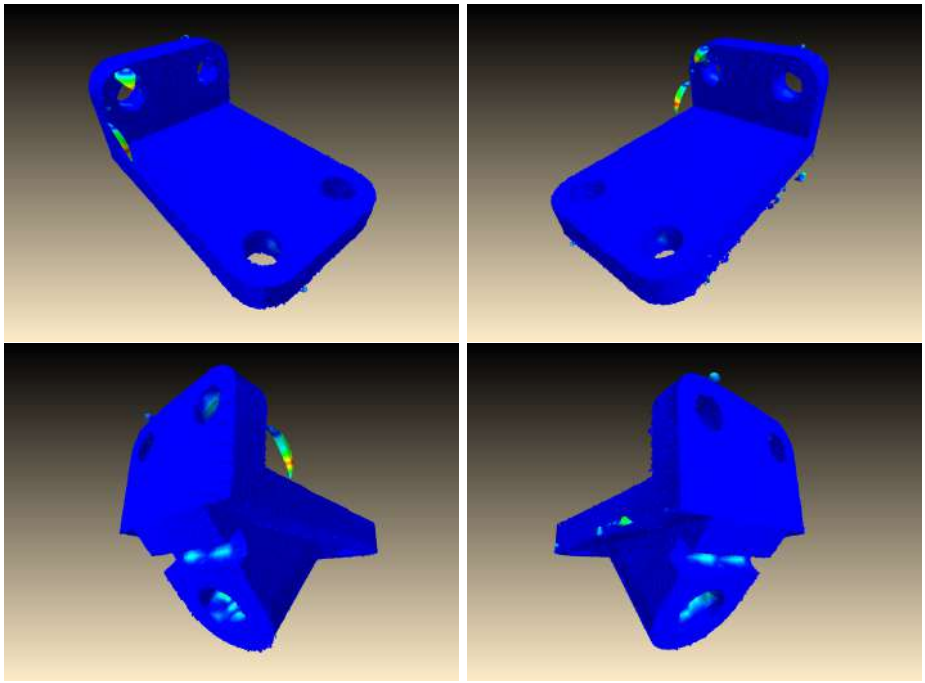
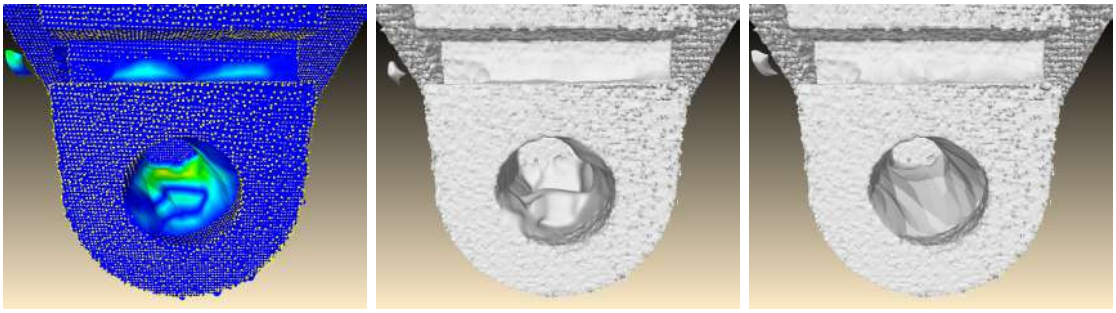


Fig. 5.13: Comparación entre reconstrucciones con PUNCH y NCH simétrica de una nube ruidosa de Anchor. Distancia de Hausdorff: 0.092568.

de Anchor, donde los puntos de la cara externa que presenta las diferencias con NCH pasan a tener frente a ellos los puntos de las láminas cilíndricas de los agujeros, muchos más ceranos que los de la pared opuesta.

En la figura 5.13 damos resultados del experimento anterior usando una nube ruidosa de Anchor en lugar de una limpia. Agregamos la distancia de Hausdorff como referencia cuantitativa básica en compelmento de las imágenes. Como es de esperar dados los resultados de nubes ruidosas que discutimos en la sección 3.6, ventajas como la posiblidad de utilizar semiespacios de radio infinito se anulan en la presencia de ruido en la entrada, debiendo acudir las reconstrucción por NCH a semiespacios de radio menor. Ausente esa capacidad, las reconstrucciones resultan mayormente iguales. Nuevamente no se evidencian efectos detrimentales de la combinación de resultados locales.

Aún así, se notan con claridad regiones con diferencias significativas entre las dos reconstrucciones. Es interesante observar que dichas regiones se encuentran en marcadas concavidades de la superficie. La nube de entrada corresponde al conjunto de datos realistas de Reconbench que se confecciona realizando simulaciones elaboradas de escaneos láser. Una de los puntos débiles del escaneo láser como método de digitalización es justamente la generación de nubes con datos faltantes en concavidades del cuerpo al que se aplica, donde el láser se encuentra con un impedimento físico a obtener datos de esas regiones.



Superposición con la nube PUNCH NCH simétrica

Fig. 5.14: Mayor zoom sobre las regiones con diferencias significativas entre las reconstrucciones de PUNCH y NCH. En la imagen de la izquierda superponemos la nube de puntos con la comparación por metro evidenciando la relación entre las diferencias de reconstrucción y la falta de datos en la nube.

En la figura 5.14 hacemos zoom en las regiones con diferencias de reconstrucción y encontramos que, efectivamente, las mismas se corresponden con agujeros en la nube de puntos. En la imagen de la izquierda se nota claramente esta correspondencia. En la imagen del medio y de la derecha podemos apreciar a simple vista las diferencias de reconstrucción entre los dos algoritmos. Como referencia, agregamos en la figura 5.15 la misma vista de este túnel de anchor en una malla reconstruida en base a la misma nube sin ruido usada en la figura 5.11.

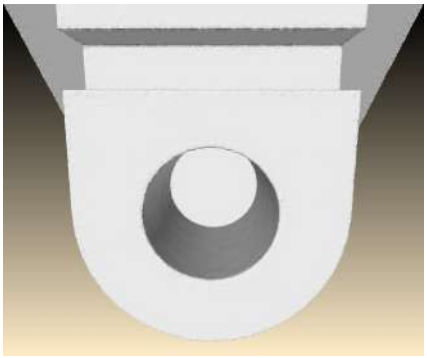


Fig. 5.15: Reconstrucción limpia por PUNCH del túnel de Anchor.

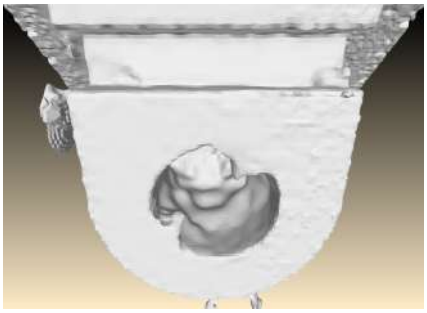


Fig. 5.16: Reconstrucción ruidosa por MPU túnel de Anchor.

Desde el punto de vista teórico, tiene sentido que se den tales diferencias. En la NCH simétrica global, por un lado, todo punto visible de la malla es directamente parte de algún promedio entre esferas correspondientes a semiespacios de soporte. Cuando se trata de agujeros en la nube, lo que visualizamos corresponde a los semiespacios de los puntos aledaños al agujero, dado que en NCH una falta de puntos se traduce en falta de semiespacios para agregar a la reconstrucción. En otras palabras, estamos visualizando la manera de *extrapolar* de NCH su reconstrucción a regiones sin datos. Por ello, en la imagen de la derecha de la figura 5.14, vemos en este túnel de Anchor una combinación de esferas de la NCH negativa, de radio relativamente grande ya que existen menos puntos alrededor que delimiten su participación. En PUNCH, por otro lado, lo que visualizamos es el resultado de una interacción algo más compleja. En ella también participan extrapolaciones de NCH pero numerosas, hechas con menos información aún al computarse de manera local y ponderadas por funciones de peso. El tratamiento por parte de PUNCH de los datos faltantes resulta menos intuitivo que el de NCH, pero también parece acercarse más en términos de distancia promedio al resultado ideal. Como referencia adicional para respaldar nuestra interpretación de esta extrapolación de PUNCH como originada en el esquema de combinación

por funciones de peso, agregamos en la figura 5.16 el túnel resultante de aplicar MPU con parámetro 2×10^{-3} a la misma nube ruidosa. En esta última reconstrucción observamos una extrapolación cualitativamente muy similar a la de PUNCH.

Con el fin de complementar esta evaluación comparativa con otra superficie, utilizamos Gargoyle del conjunto de Reconbench ya que presenta otro tipo de propiedades como detalles finos y características curvas y suaves. En la figura 5.18 contrastamos PUNCH y la NCH simétrica aplicados a una nube limpia. Se nota cómo las reconstrucciones son prácticamente idénticas, salvo en una variedad de pequeñas regiones distribuidas en toda la superficie. La diferencia más marcada en una de las puntas del ala izquierda se debe a un error de visualización independiente de las reconstrucciones propiamente dichas.

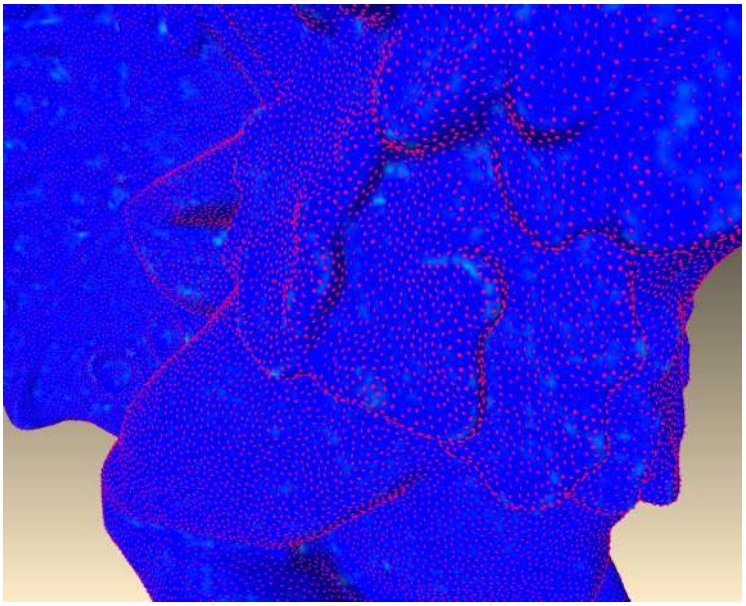


Fig. 5.17: Zoom en algunas diferencias entre PUNCH y NCH para Gargoyle limpio, superpuestas con la nube.

No se observa en esta comparación el comportamiento descrito en realción a las diferencias en láminas planas de Anchor, hecho esperable dada la ausencia de tal tipo de característica en Gargoyle. No obstante, las pequeñas diferencias que sí existen se pueden atribuir al mismo fenómeno: diferencias en los semiespacios asignados a cada punto, originados en que la NCH promedia esferas de gran radio definidas en la variante positiva con las esferas negativas cuyo radio es limitado por el interior del cuerpo, mientras que PUNCH cuenta con las mismas esferas positivas pero no es sujeto a esa limitación para las negativas. Esta explicación es corroborada en la figura 5.17, donde hacemos zoom en la comparación por Metro superpuesta con la nube de puntos. Si observamos con detenimiento, podemos ver que en absolutamente todos los casos las diferencias se dan *entre* puntos. En otras palabras, nunca vemos un punto "ubicado *encima*" de una diferencia entre las mallas. Esto se debe primariamente a que las dos mallas coinciden perfectamente *en* los puntos, debido a que que ambas se generan por métodos interpolantes. Lo interesante es notar que luego difieren de manera gradualmente creciente en función de la distancia a los puntos: donde hay diferencias, cuanto más lejos de los puntos involucrados miremos, mayor es la distancia. La ocurrencia y el grado de estas distancias depende de cómo se dé la subdivisión, ya que la misma es la que determina qué localidades de PUNCH se segmentan de manera de desconocer puntos vecinos que limitan sus semiespacios negativos.

Al igual que con Anchor, contrastamos también reconstrucciones de Gargoyle sobre una nube ruidosa. En la figura 5.19 mostramos los resultados correspondientes, agregando aquí también como referencia numérica la distancia de Hausdorff. En primer lugar, observemos que las diferencias pequeñas entre las mallas como las que vimos en el caso limpio desaparecen. Este hecho sirve como corroboración adicional de nuestra interpretación de aquel comportaemiento, de manera análoga al caso de Anchor: cuando se introduce ruido

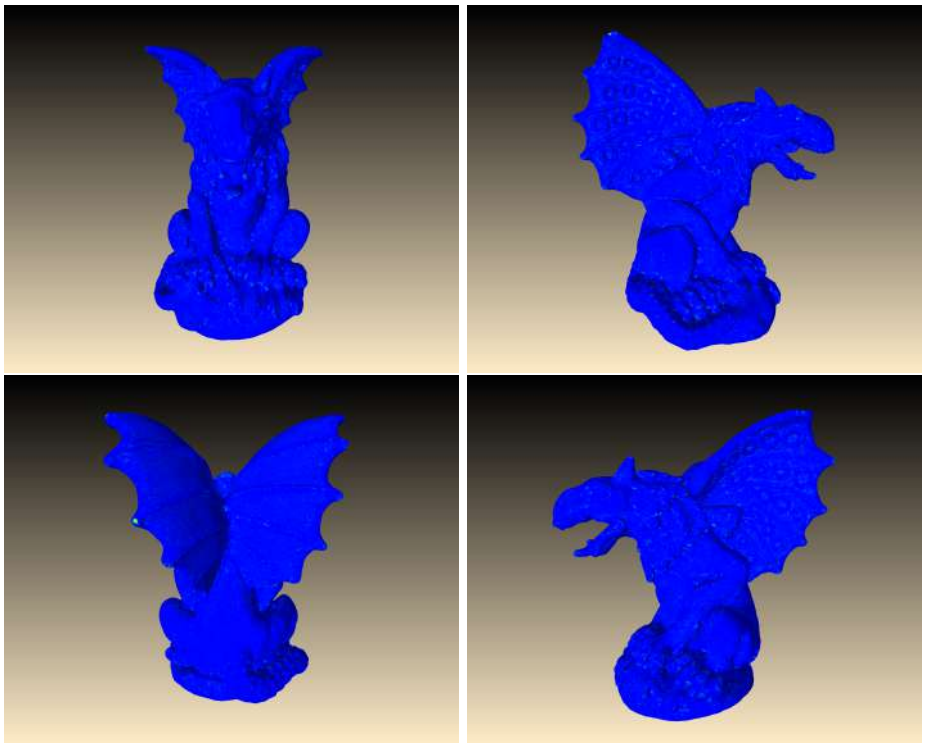


Fig. 5.18: Comparación entre PUNCH y NCH simétrica usando una nube limpia de Gargoyle.

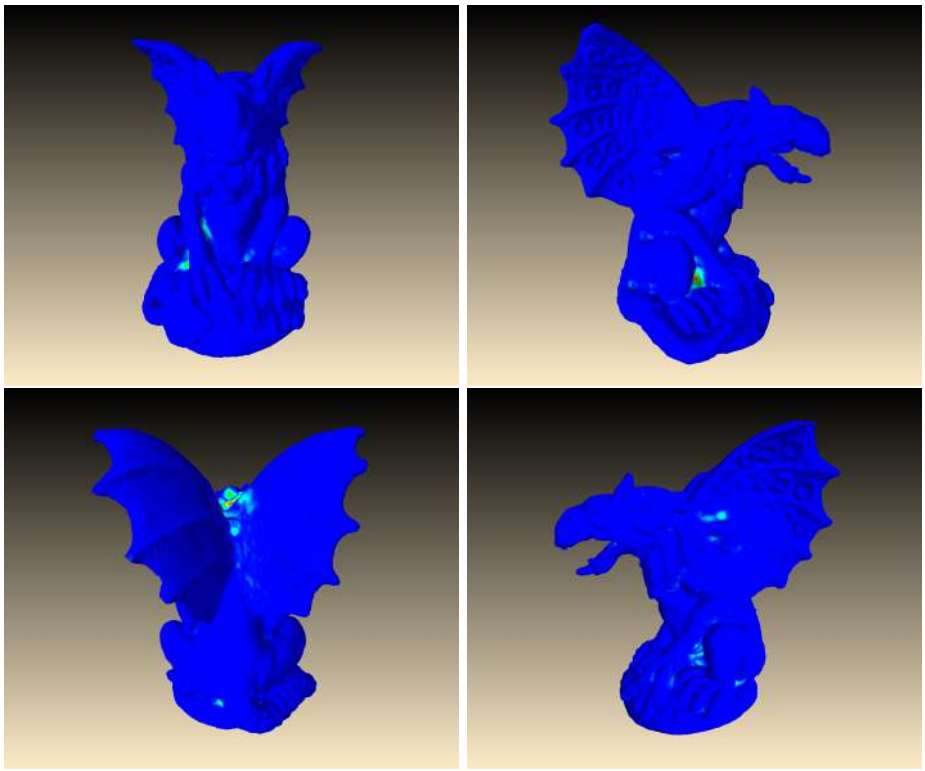
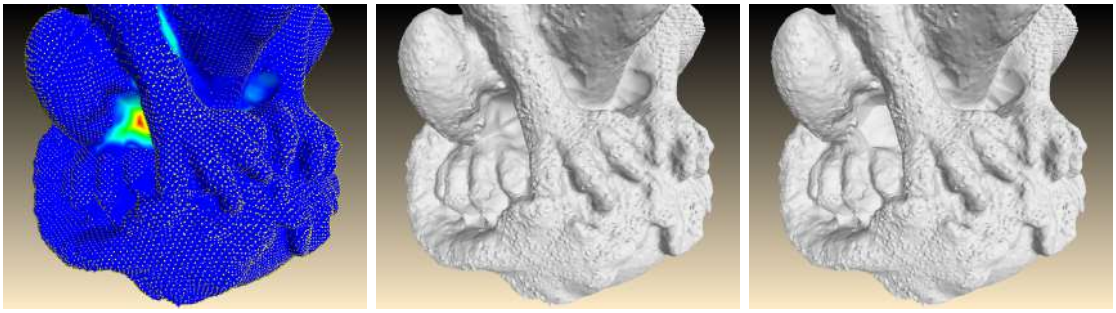


Fig. 5.19: Comparación entre PUNCH y NCH simétrica usando una nube ruidosa de Gargoyle. Distancia de Hausdorff 0.076227.



Superposición con la nube PUNCH NCH simétrica

Fig. 5.20

Superposición con la nube PUNCH NCH simétrica

Fig. 5.21

en la nube, el radio de todas las esferas utilizadas en reconstrucciones de NCH, globales o locales, disminuye considerablemente.

También de manera muy similar a Anchor, vemos diferencias acentuadas entre las reconstrucciones de PUNCH y NCH en concavidades profundas del cuerpo. Este comportamiento aquí también se explica por regiones de datos faltantes. Mostramos algunos de estos agujeros en la nube en las figuras 5.20 y 5.21, donde vemos la misma falta de puntos en las regiones de grandes distancias entre mallas y el mismo tipo de extrapolación en PUNCH y NCH. Nuevamente, las reconstrucciones de estas regiones resultan algo anti-intuitiva en términos estéticos, posiblemente más aún en el caso de PUNCH.

5.5.3 Comparación cuantitativa

Para poner la evaluación de calidad de PUNCH en un contexto más amplio, realizamos una evaluación cuantitativa por Reconbench que lo compara con NCH y MPU, al igual que con Poisson como referencia. Configuramos PUNCH según $M = 500$ y $m = 100$. Para NCH, ponemos en consideración tanto la variante positiva como la simétrica, recordando que es la que se aplica en PUNCH. En el caso de MPU, en referencia a nusetro análisis de parámetros correspondiente, lo agregamos tanto configurado con un error de 10^{-4} como con 2×10^{-3}. Los resultados se presentan en la figura 5.22.

En la métrica de desvío promedio de ángulos, PUNCH presenta resultados favorables. Si bien tiene una performance ligeramente inferior a Poisson, se encuentra consistentemente en un rango similar o inferior de valores de error en comparación con sus contrapartes globales, las NCH simétrica y positiva. Esto indica que el mecanismo de combinación de

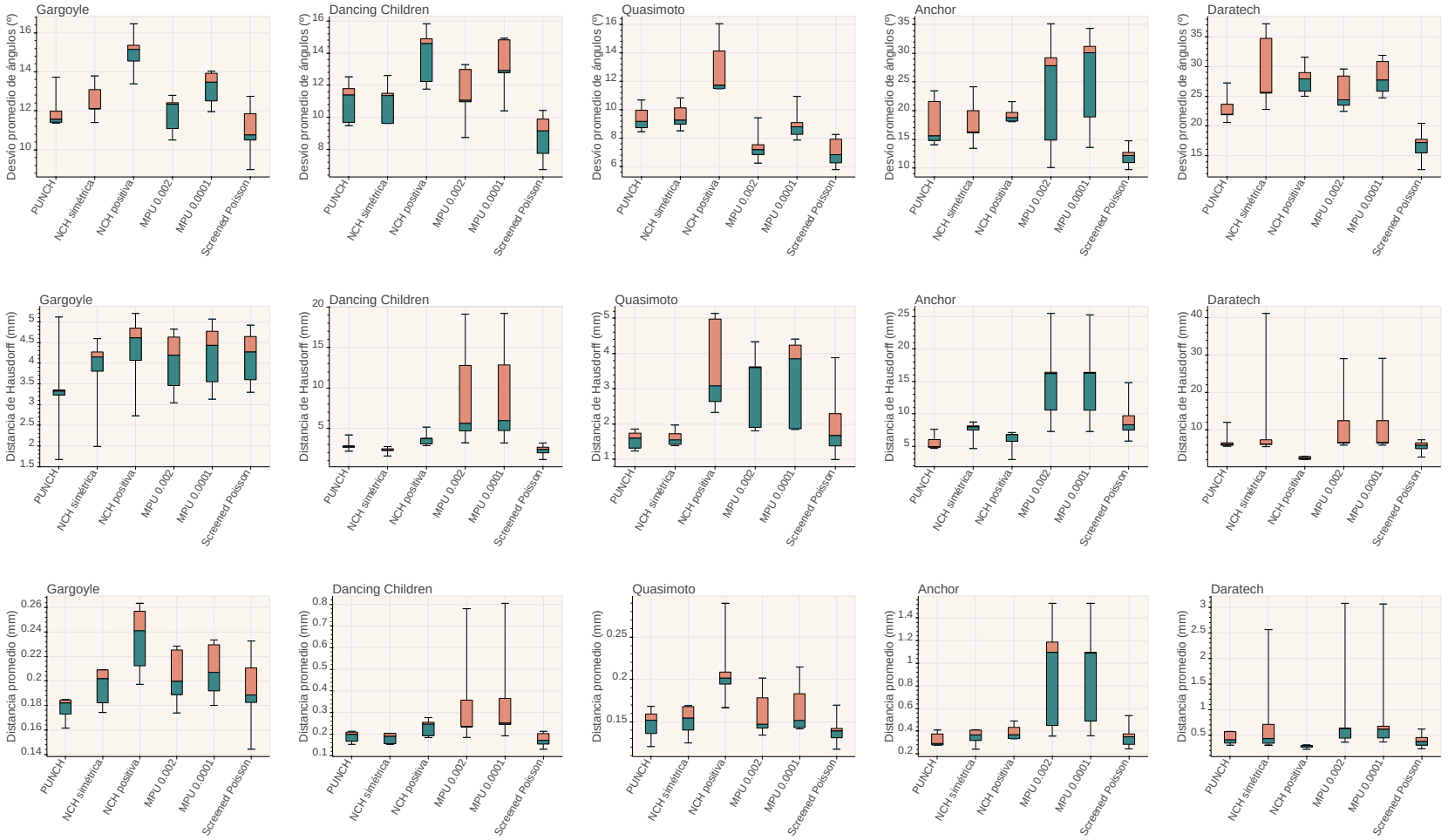


Fig. 5.22: Comparación de distribuciones de error entre PUNCH y otros algoritmos.

localidades no influye negativamente en esta métrica. El éxito de PUNCH relativo a la variante positiva se puede explicar por esferas de radio demasiado chico que se usan en la reconstrucción debido a ruido en los puntos y sus normales, cuyo efecto es mitigado en la variante simétrica y más aún en PUNCH que particiona los datos y su componente de la variante negativa está menos limitada en el radio de sus esferas, como ya hemos analizado. Asimismo, salvo en el caso de Quasimoto, muestra mejores resultados que las instancias de MPU. Quasimoto es la figura más suave y la de menos detalles finos de todas, hecho que puede explicar el mayor éxito de MPU que trabaja con la combinación de superficies cuadráticas.

Entre las dos métricas restantes, ambas medidas de distancia, el comportamiento es parecido. PUNCH da resultados comparables o mejores que el resto de los algoritmos. Esto en particular aplica si constrastamos a PUNCH con las NCHs, observación que indica que la localización de reconstrucciones de PUNCH tampoco es perjudicial en términos de desviación en distancia. Tiene sentido la favorabilidad tanto de PUNCH como de las NCHs en comparción con MPU en estas métricas dada su naturaleza interpolante, que lleva a que sus superficies se peguen más al modelo de ground truth, mientras que MPU aproxima y tiende a sobre-suavizar y así mantener mayores distancias al ground truth. Poisson tiene la misma tendencia, lo cual explica sus resultados algo peores ante estas métricas. En el caso de la distancia de Hausdorff en particular, vemos una distribución de errores inestable a través del conjunto de figuras. Recordando los resultados que discutimos en la sección 5.5.2, donde observamos un comportamiento difícil de predecir de estos algoritmos ante regiones de datos faltantes, especialmente en el caso de PUNCH donde la extrapolación de esas partes introdujo relieves anti-intuitivos, podemos atribuir esta variabilidad a una capacidad de extrapolación relativamente impredecible.

Conclusiones y trabajo futuro

Presentamos el método de la NCH ([27]), un algoritmo de SDF global e interpolante, estudiamos su teoría y analizamos su comportamiento ante distintas familias de entradas. En particular, aportamos una comparación detallada entre sus variantes en base a sus fundamentos teóricos y evaluaciones cualitativas. Dicha discusión resultó favorable a la variante *positiva* del algoritmo, reconociendo que sus ventajas se ven aminoradas ante la presencia de ruido en la entrada. La variante alternativa a considerar en base a esta comparación es la *simétrica* siendo que de la restante, la *negativa*, se esperan y observan resultados consistentemente inferiores. En nuestra evaluación cuantitativa global (figura 5.22), en cambio, encontramos un rendimiento algo peor de la variante positiva en comparación con la simétrica en algunos casos. Es interesante notar que esto ocurre principalmente en nuestro subconjunto de superficies con rasgos suaves y detalles finos, en contraste con las figuras de láminas planas y rasgos filosos donde la variante positiva presenta mejores resultados que su contraparte simétrica, distinción que se condice con nuestro análisis comparativo inicial. Estas observaciones motivan posiblemente el diseño de algún criterio que sirva para elegir dinámicamente una variante en función de características de la nube sobre la que se va a aplicar el algoritmo. En ausencia de un criterio tal, el análisis hecho aquí puede resultar informativo a la hora de deber una usuaria aplicar este algoritmo a sus datos.

También de nuestras evaluaciones cualitativas se desprenden las desventajas del carácter interpolante de NCH en la presencia de ruido, al igual que las de su mecanismo de unión de esferas que produce resultados poco intuitivos cerca de bordes donde es necesaria la reducción de los radios de sus semiespacios. A causa de ese mismo mecanismo, aspecto esencial del método, se produce también su forma de extrapolación algo llamativa a la vista en regiones de datos faltantes, donde no tiene alternativa más que rellenar con esferas de grandes radios. Una contracara favorable de este mecanismo es lo sencillo que resulta el método, permitiendo una implementación simple y una facilidad particular de acudir a la teoría para explicar resultados observados en sus mallas.

En nuestros experimentos con MPU vimos reflejada claramente su naturaleza aproximada y error-adaptativa, observando cómo reconstruye aproximaciones híper-gruesas para un parámetro de error laxo y se sobreadapta a nubes ruidosas utilizando un parámetro muy estricto. Planteamos un estudio que nos dio como resultado un parámetro de error más adecuado a nuestro conjunto de datos, y por consiguiente a nuestra inquietud por presentar una comparación global entre los algoritmos que pone en juego a todos en sus mejores condiciones. Nuestras observaciones cualitativas iniciales acerca de los resultados en función de estos valores del parámetro son corroboradas en la evaluación cuantitativa global, logrando la instancia de MPU con el parámetro nuevo resultados iguales o superiores consistentemente. Pensamos que esta forma de análisis del parámetro de error puede

tomarse como punto de partida para una generalización a otros métodos guiados por subdivisión, especialmente los que sean error-adaptativos, pudiendo obtener en base a ella un panorama estadístico de la adherencia del algoritmo estudiado a su configuración y los efectos negativos en materia de recursos computacionales que ello pueda conllevar, en el caso de MPU manifestados en subdivisiones que fallan en cumplir con una parametrización exigente a pesar de ser maximalmente profundas.

De manera similar a nuestras conclusiones acerca de NCH, desprendemos de este análisis también la idea de un trabajo futuro que provea un criterio adaptativo encargado de determinar dinámicamente el parámetro de error a configurar de acuerdo a cada nube. La naturaleza de la métrica de error utilizada por MPU, basada en distancias, motiva plantear un criterio basado en densidad: cuanto más densa sea la nube, mayores deberían ser los parámetros de error viables de exigir. Notemos que esto se corresponde con la intuición si pensamos a la densidad de la entrada en términos de información acerca del cuerpo objetivo: cuanto más densa una nube, más información provee acerca de la superficie que representa, y por ende debería poderse exigir una mejor adaptación con un parámetro más estricto de máximo error permisible.

Observando la performance de las dos instancias de MPU en la comparación cuantitativa global para los casos de Anchor y Daratech, nuestras figuras representativas de superficies de rasgos filosos y bordes, no encontramos resultados particularmente favorables a MPU a pesar de su especial énfasis en la reconstrucción fiel de ese tipo de características. Es esperable que un éxito ante esa misión específica se vea reflejado en los resultados de desvío promedio de ángulos en superficies caracterizadas por sus rasgos filosos y bordes, pero los valores de error exhibidos por MPU en esta métrica son los más elevados en el caso de Anchor y pierden claramente en comparación con Poisson y PUNCH en el caso de Daratech. Vale agregar que un comportamiento compatible fue reportado por los autores de Reconbench [6], que explicitan sus observaciones acerca de pobres resultados en la aproximación de rasgos filosos como parte de su motivación por utilizar en su propio trabajo una modificación de MPU diseñada por ellos.

Consideramos mayormente cumplidos los objetivos de PUNCH en tanto estos se enmarquen en la mejora de los tiempos de ejecución de NCH emparejada con la mantención de su calidad. Contemplando especialmente la evaluación cuantitativa global como la cualitativa contra las reconstrucciones de NCH, tenemos vasta evidencia de que el hecho de combinar NCHs locales a través de funciones de peso no influyó negativamente en la SDF global, a excepción del caso de regiones con puntos faltantes donde la extrapolación resultó menos intuitiva. En este sentido, planteamos el interés en un trabajo futuro de explorar funciones de peso que mantengan esta calidad de unión de reconstrucciones locales y a la vez mejoren la extrapolación del método a regiones sin puntos. Vale aclarar que tal análisis probablemente deba ser específico a cada estrategia local a la que se apele; en el caso de PUNCH, por ejemplo, se beneficiaría de aprovechar el conocimiento concreto de que NCH extrapola utilizando esferas de radios grandes.

Además, el hecho de que no se haya encontrado ninguna falla en la combinación de localidades motiva la pregunta de si puede relejarse el criterio de superposición de soportes, basado en nuestro caso en tomar un radio mayor o igual al 125 % de la diagonal de cada celda (de acuerdo al multiplicador $\alpha = 1{,}25$). Es concebible, en principio, que pueda lograrse una calidad de reconstrucción indistinguible por nuestras métricas que se obtenga definiendo soportes más chicos, modificación que en principio podría producir mejoras en los tiempos de ejecución y en el consumo de memoria al permitir una subdivisión más

gruesa y un menor grado de redundancia de puntos soportados entre las localidades.

Es interesante observar también el carácter paralelizable, en principio, de este esquema de partición incorporado en PUNCH. En la etapa de cálculo de SDF, cada reconstrucción local es independiente y puede ejecutarse en cualquier orden respecto del resto. La etapa de evaluación también admite la paralelización de las evaluaciones locales previo a ser ponderadas para dar el resultado global. Contemplando esto, sería interesante soportar paralelización en PUNCH y analizar la mejora en tiempos de ejecución así obtenida. Vale notar que NCH también tiene una naturaleza paralelizable, siendo el cálculo de cada semiespacio independiente del resto de los semiespacios. Por ello, en caso de querer contraponer los tiempos de tal implementación de PUNCH con los de NCH, sería importante hacerlo paralelizando la ejecución del último también para asegurar una comparación justa.

Tomando en cuenta estos resultados y observaciones, cabe plantearse si podemos dar una respuesta fundamentada a lo que probablemente constituya la pregunta más interesante que podemos postular acerca de lo hecho con PUNCH: ¿qué efectos tiene en el rendimiento de un algoritmo de reconstrucción global el llevarlo a un enfoque local? Nuestros experimentos parecen indicar que hacerlo mediante partición de la unidad con funciones de peso apropiadas tiene el potencial de lograr resultados favorables, siempre que se tengan en cuenta las características particulares del método en cuestión; en el caso de NCH, esto implicó decisiones como limitar las poblaciones de las localidades para controlar la complejidad cuadrática de NCH, imponer un mínimo a la cantidad de puntos usada en cada reconstrucción y reconsiderar la elección de variante de NCH ante la presencia de la partición de los datos en localidades. En el caso de que el método global pueda tener tiempos prohibitivos, llevarlo a la aplicación en localidades controladas con una estructura de búsqueda logarítmica como un octree puede sin dudas tener un resultado satisfactorio como el que hemos observado aquí. Más aún, la mejor performance que exhibe PUNCH en contraste con las NCHs en nuestra comparación cuantitativa global es interpretable como testigo de que NCH puede beneficiarse de considerar sólo una vecindad de cada punto al computarle su semiespacio, a diferencia de la inclusión en su cálculo de la totalidad de la nube de puntos. En última instancia, estos resultados motivan la pregunta de hasta qué punto podría generalizarse un esquema que localiza algoritmos de reconstrucción globales arbitrarios, y de qué tipo de flexibilidad debería ser capaz para acomodar las particularidades de cada uno.

Por último, vale remarcar la utilidad de los frameworks de evaluación cualitativa y cuantitativa como los que hemos usado en este trabajo, destacando la importancia de complementar evaluaciones de ambos tipos al igual que de eficiencia temporal y espacial a la hora de contrastar y razonar acerca de algoritmos de reconstrucción. Dejamos abierta la tarea de producir otras métricas de error para evaluaciones cuantitativas que ayuden a acortar distancias entre las conclusiones que puedan sacarse en base a evaluaciones de cada uno de estos tipos por separado. Aparte, al diseñar, purgar o estudiar particularmente algoritmos en cuyo corazón acciona un mecanismo de subdivisión complejo y no-determinístico, sería de gran utilidad valerse de una herramienta que permita visualizar la subdivisión que resulta en cada ejecución del algoritmo.

Bibliografía

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, pages 3–15, 2003.

[2] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 39–48. Eurographics Association, 2007.

[3] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, 1999.

[4] N. Amenta, S. Choi, and R.K. Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid Modeling and Applications*, pages 249–266. ACM, 2001.

[5] A.V. Arhangel'skii, G.G. Gould, and M.M. Choban. *General Topology III: Paracompactness, Function Spaces, Descriptive Theory*. Encyclopaedia of Mathematical Sciences. Springer Berlin Heidelberg, 2014.

[6] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):20, 2013.

[7] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[8] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[9] J.D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Computational Geometry*, 22(1):185–203, 2002.

[10] R.M. Bolle and B.C Vemuri. On Three-Dimensional Surface Reconstruction Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[11] F. Calakli and G. Taubin. SSD: Smooth Signed Distance Surface Reconstruction. *Computer Graphics Forum*, 30(7), 2011.

[12] F. Cazals and J. Giesen. Delaunay triangulation based surface reconstruction. *Effective Computational Geometry for Curves and Surfaces*, pages 231–276, 2006.

[13] M.C. Chang, F.F. Leymarie, and B.B. Kimia. Surface Reconstruction from Point Clouds by Transforming the Medial Scaffold. *Computer Vision and Image Understanding*, 113(11):1130–1146, 2009.

[14] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring error on simplified surfaces. Technical report, Paris, France, France, 1996.

[15] T.K. Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*. Cambridge University Press, 2007.

[16] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24:544–552, July 2005.

[17] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. In *Proceedings of ACM Siggraph*. Citeseer, 1992.

[18] M. Kazhdan. Reconstruction of solid models from oriented point sets. In *Proceedings of the third Eurographics symposium on Geometry processing*, page 73. Eurographics Association, 2005.

[19] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70. Eurographics Association, 2006.

[20] Michael Kazhdan. Reconstruction of solid models from oriented point sets. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[21] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[22] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3d Surface Construction Algorithm. pages 163–169, 1987.

[23] J. Manson, G. Petrova, and S. Schaefer. Streaming surface reconstruction using wavelets. In *Computer Graphics Forum*, volume 27, pages 1411–1420. Wiley Online Library, 2008.

[24] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 463–470. ACM, 2003.

[25] G. Taubin and D. Lanman. *3D Photography*. Taylor & Francis, 2014.

[26] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(11):1115–1138, November 1991.

[27] Gabriel Taubin. Non-convex hull surfaces. In *SIGGRAPH Asia 2013 Technical Briefs*, page 2. ACM, 2013.

[28] William I. Thacker, Jingwei Zhang, Layne T. Watson, Jeffrey B. Birch, Manjula A. Iyer, and Michael W. Berry. Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data. *ACM Trans. Math. Softw.*, 37(3):34:1–34:20, September 2010.

[29] Zhengyou Zhang. Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, 19(2):4–10, 2012.

Apéndice

Apéndice de la Sección 4.7

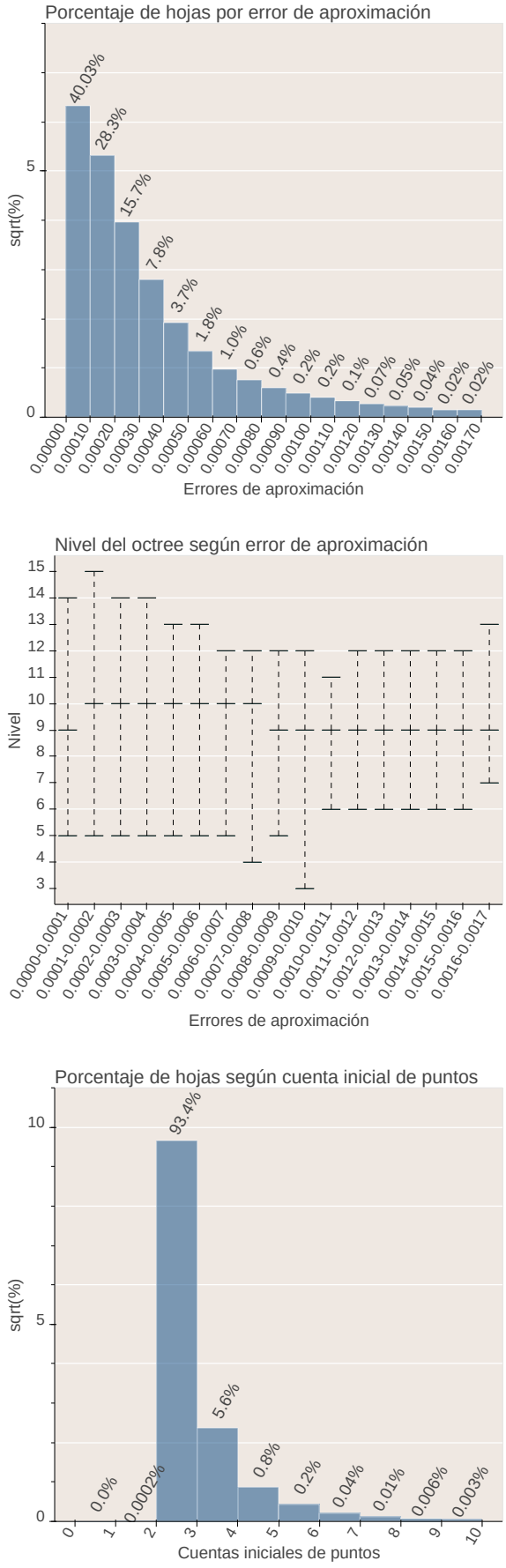


Fig. 7.1: Análisis de subdivisión para Gargoyle con parámetro de error 0.0001.

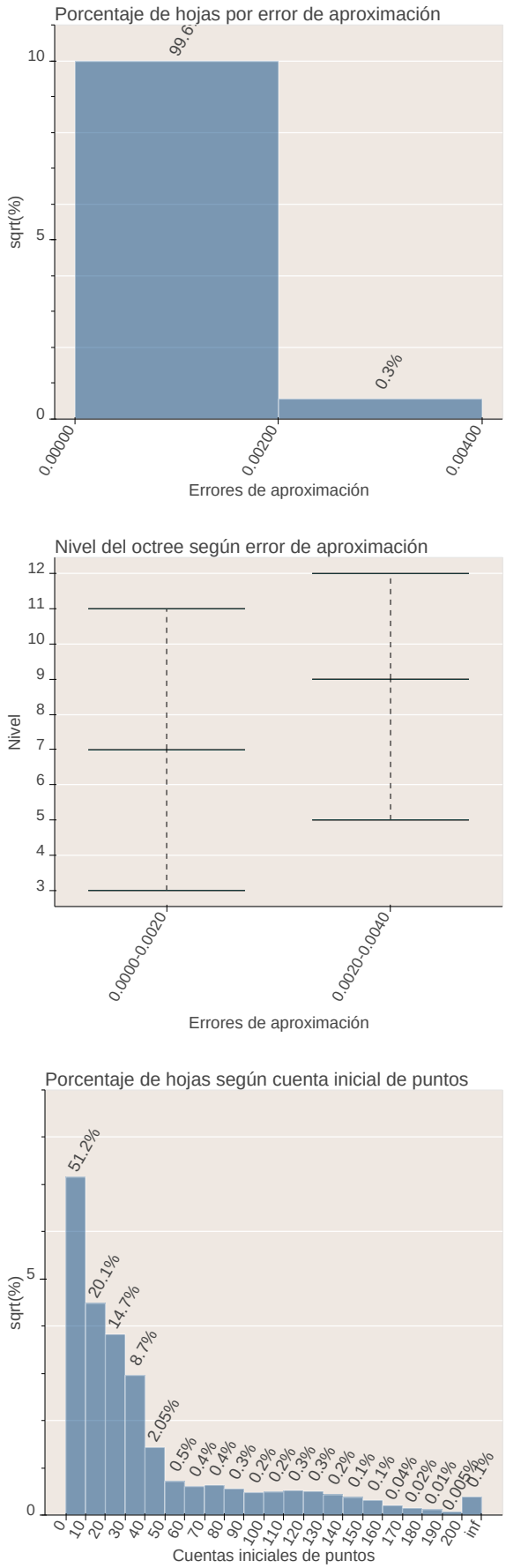


Fig. 7.2: Análisis de subdivisión para Gargoyle con parámetro de error 0.002.

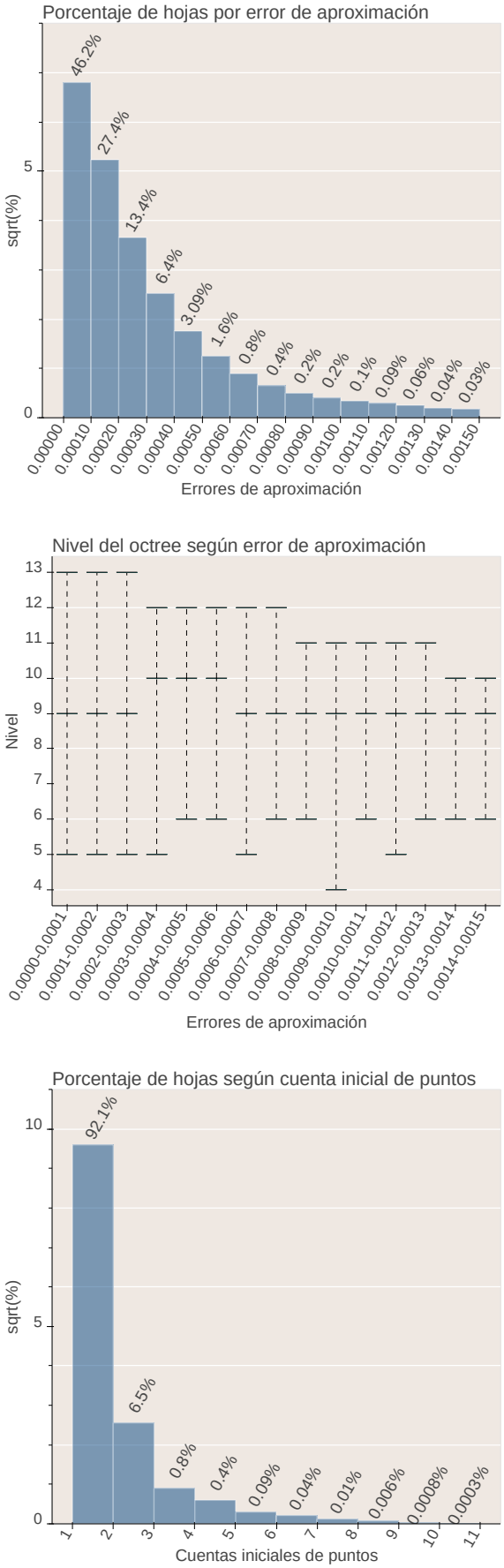


Fig. 7.3: Análisis de subdivisión para Dancing Children con parámetro de error 0.0001.

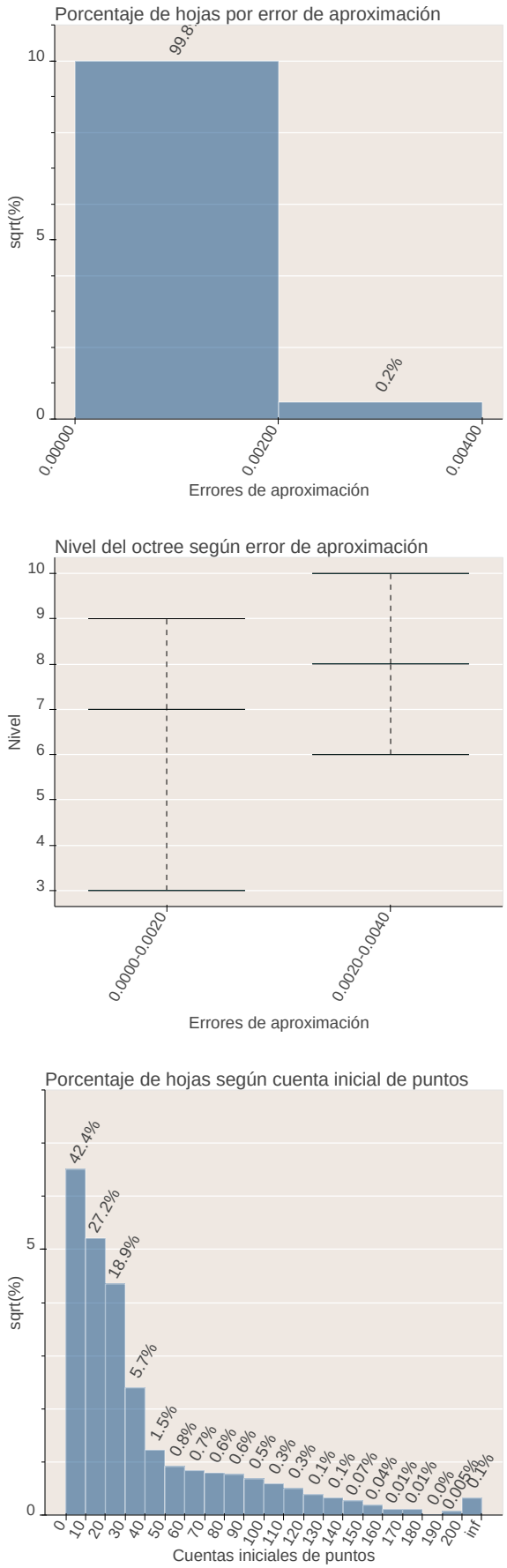


Fig. 7.4: Análisis de subdivisión para Dancing Children con parámetro de error 0.002.

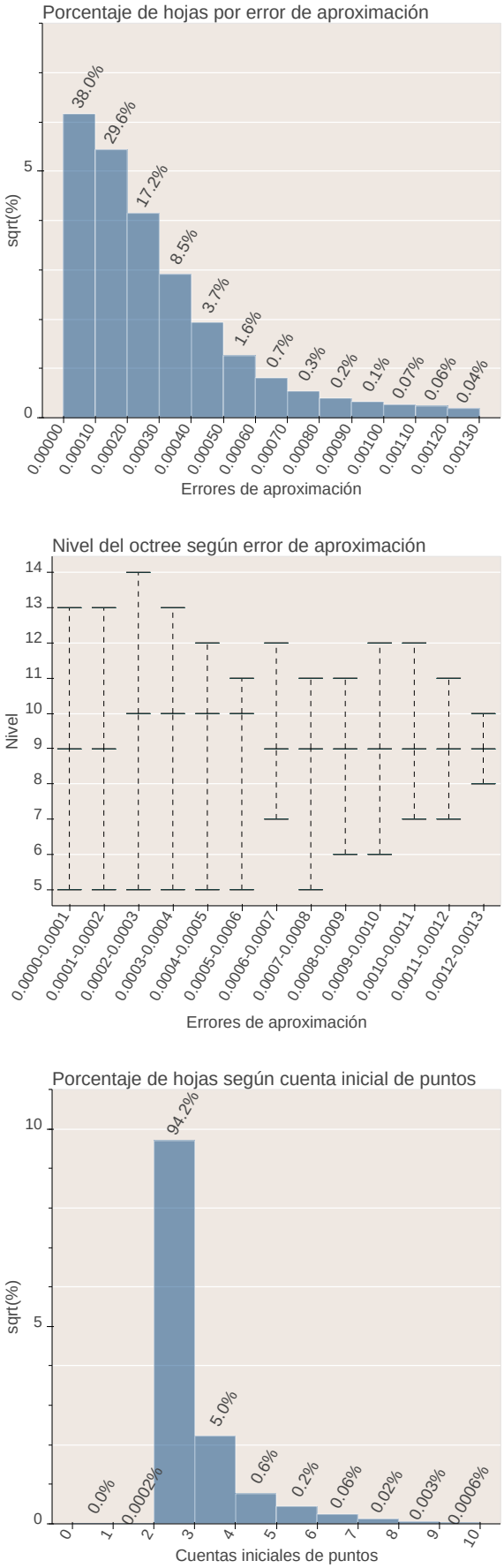


Fig. 7.5: Análisis de subdivisión para Quasimoto con parámetro de error 0.0001.

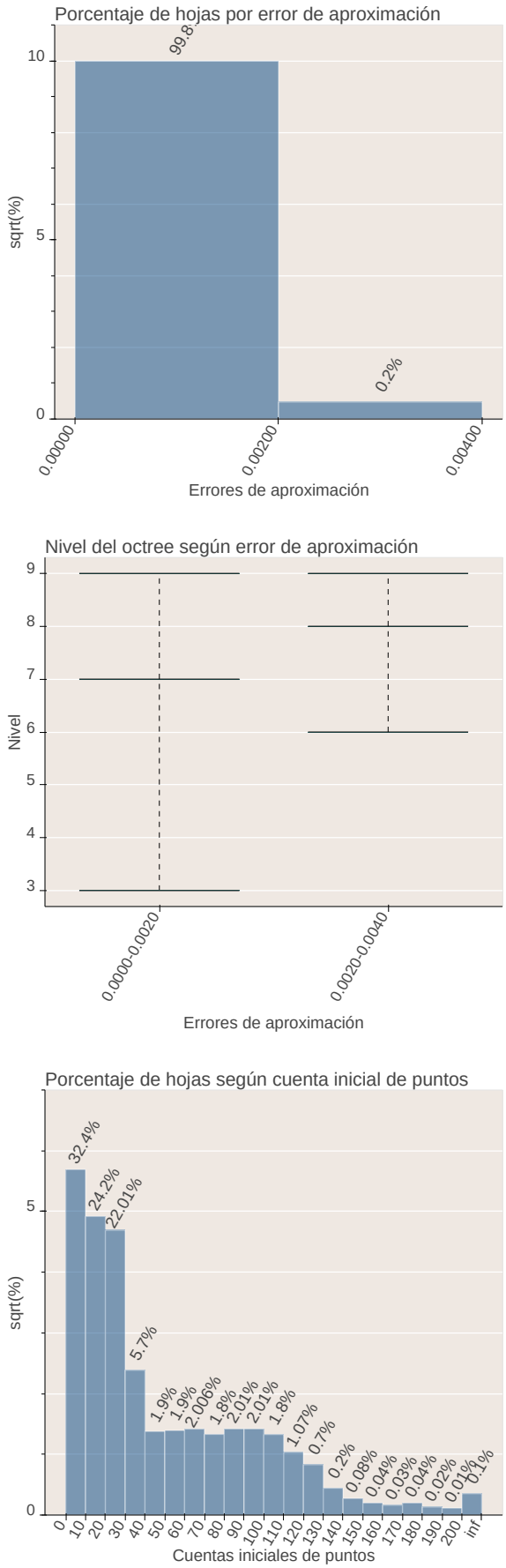


Fig. 7.6: Análisis de subdivisión para Quasimoto con parámetro de error 0.002.

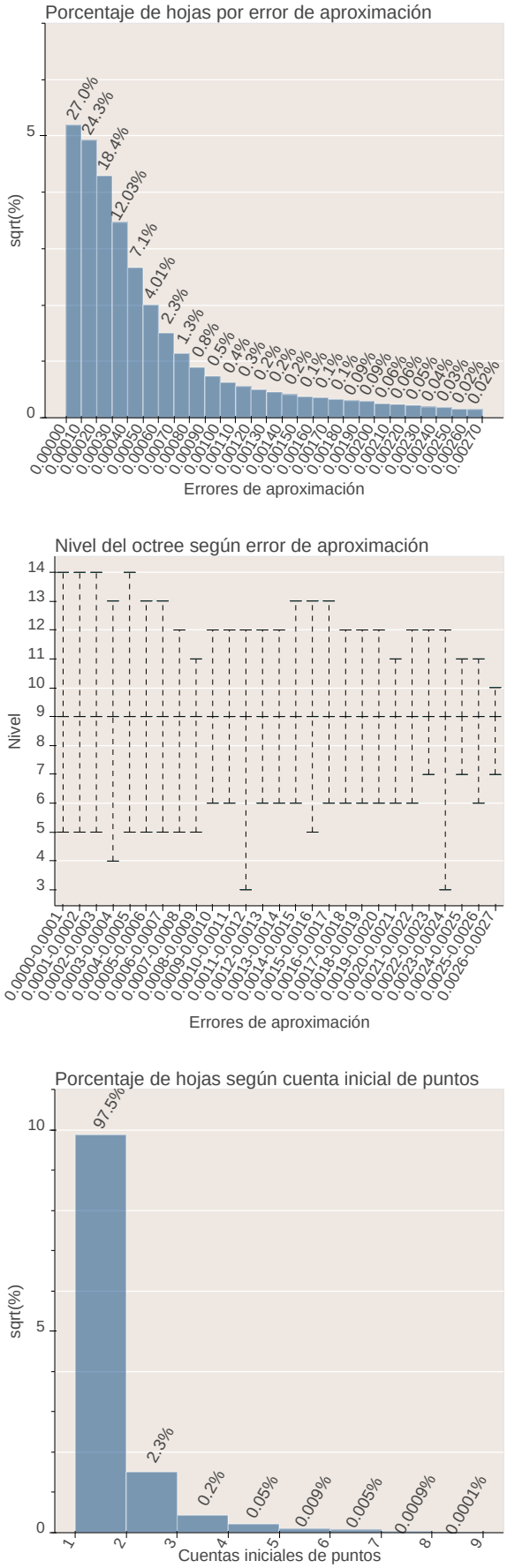


Fig. 7.7: Análisis de subdivisión para Anchor con parámetro de error 0.0001.

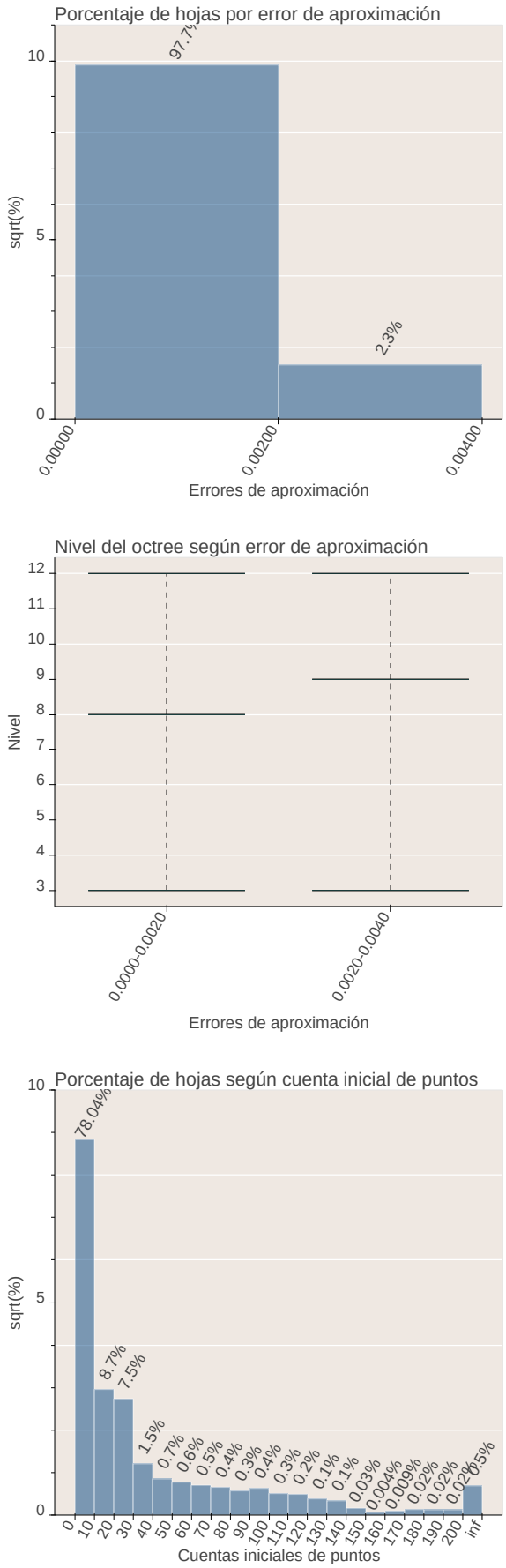
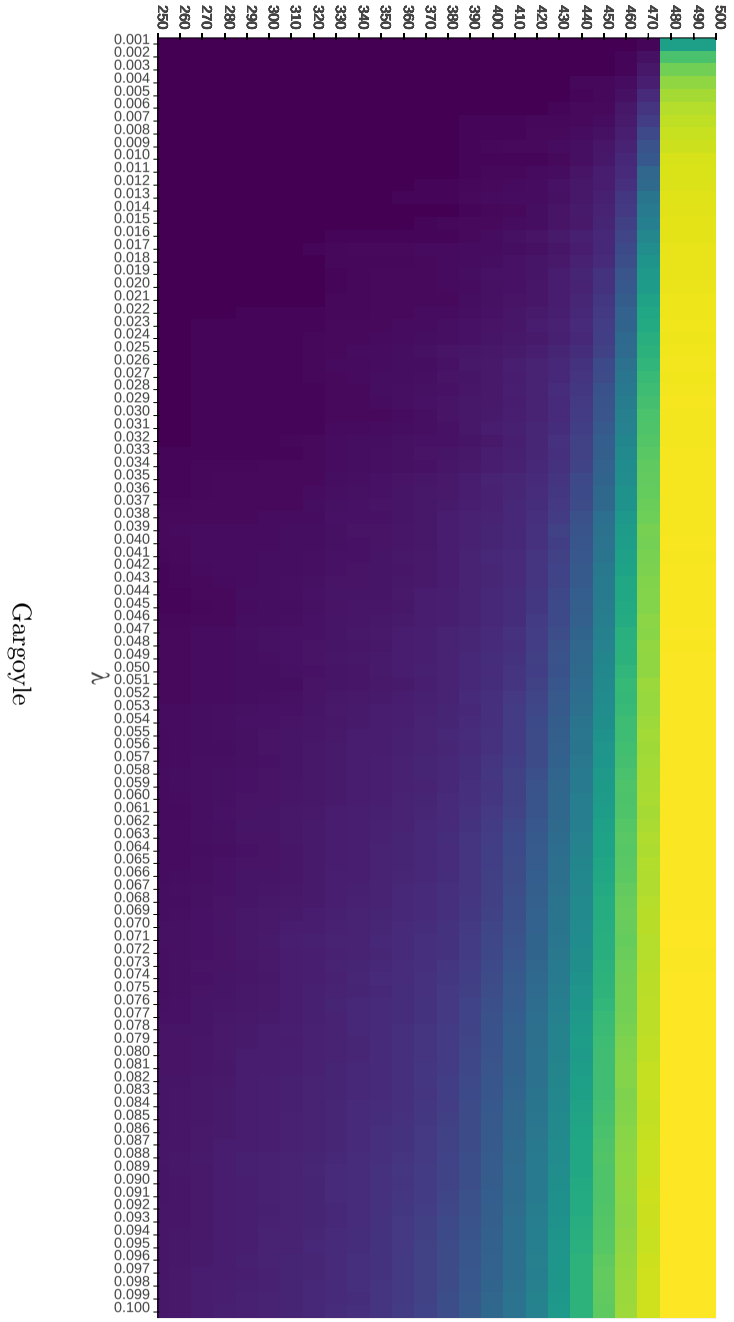
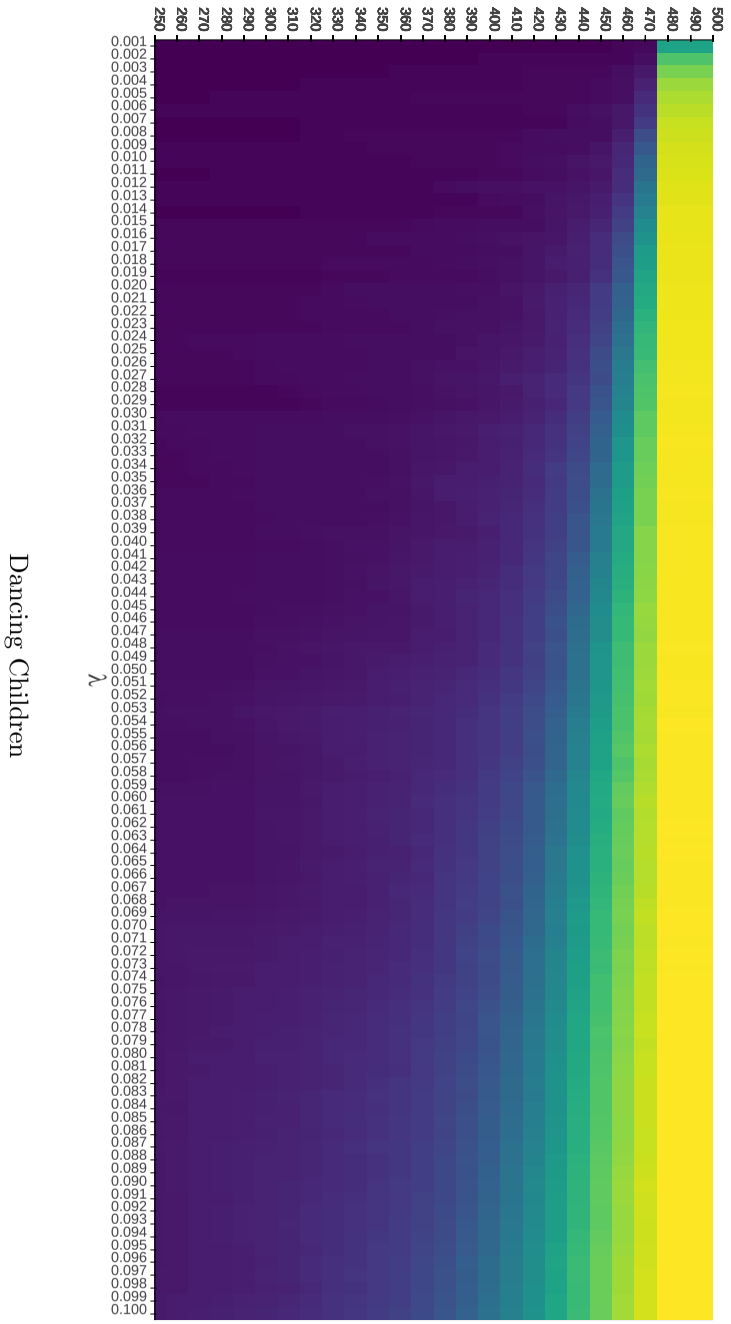
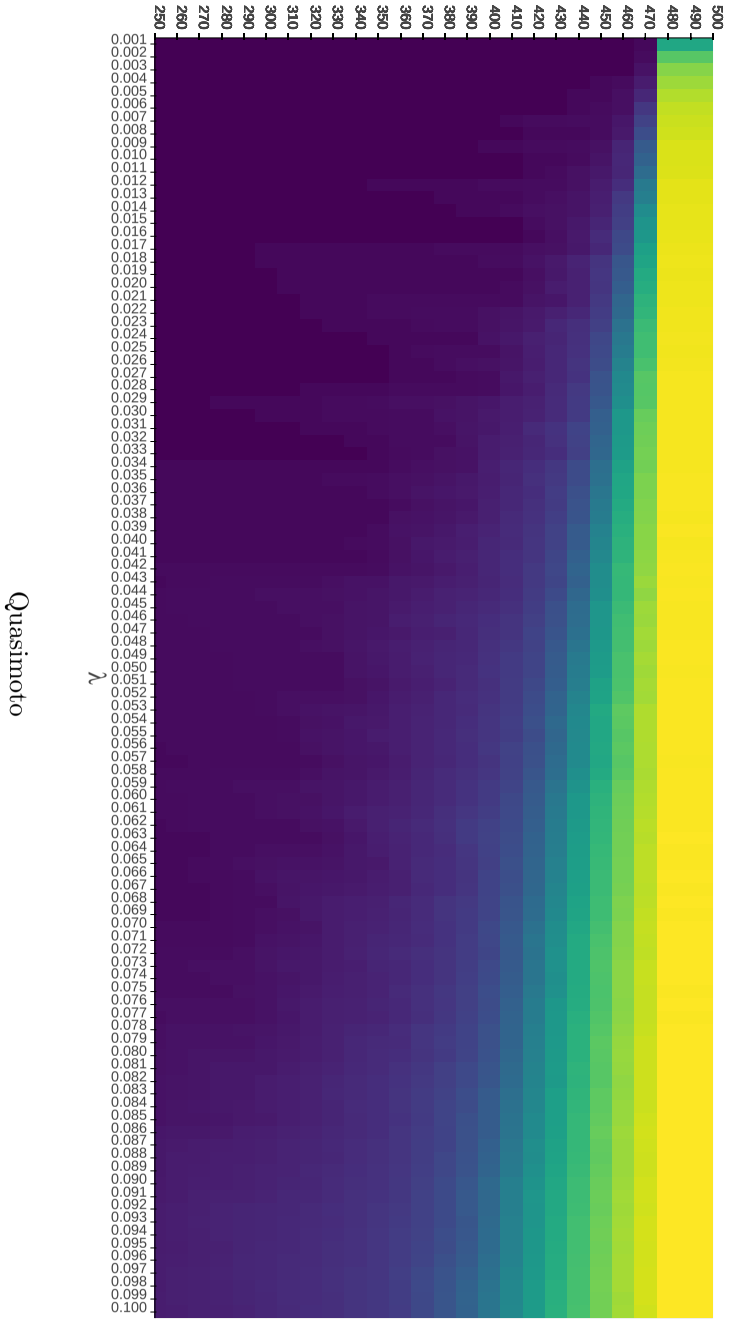


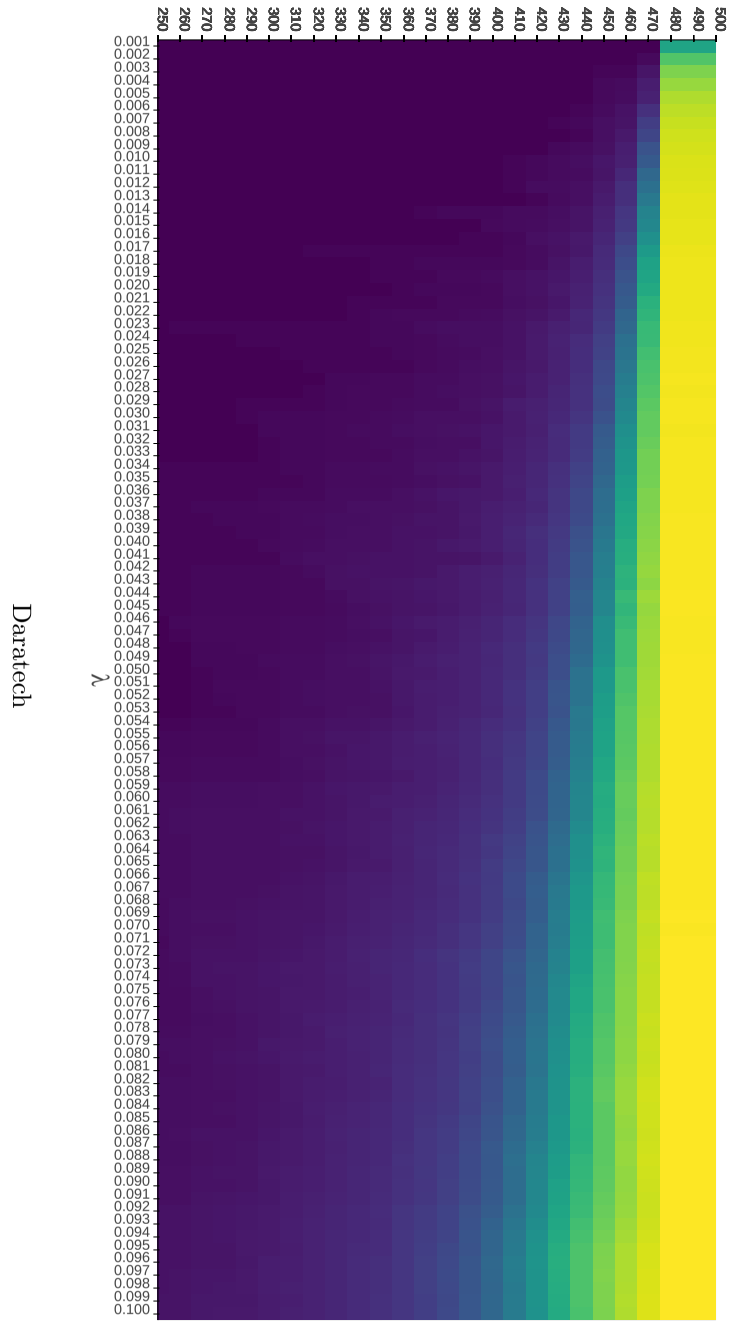
Fig. 7.8: Análisis de subdivisión para Anchor con parámetro de error 0.002.

Apéndice de la Sección 5.3


Gargoyle
γ
250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500
0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096 0.097 0.098 0.099 0.100


Dancing Children
γ
250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500
0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096 0.097 0.098 0.099 0.100


Quasimoto
γ
250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500
0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096 0.097 0.098 0.099 0.100


Daratech
γ
250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500
0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.017 0.018 0.019 0.020 0.021 0.022 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096 0.097 0.098 0.099 0.100