UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SEGMENTACIÓN DENTAL BASADA EN DETECCIÓN DE PUNTOS CLAVES Y CAMPOS ARMÓNICOS

MEMORIA PARA OPTAR AL TÍTULO DE

INGENIERO CIVIL EN COMPUTACIÓN

JAVIER IGNACIO LIBERMAN SALAZAR

PROFESOR GUÍA:
NANCY HITSCHFELD KAHLER

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
JOSÉ SAAVEDRA RONDO

SANTIAGO DE CHILE
2018

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: JAVIER IGNACIO LIBERMAN SALAZAR
FECHA: 2018
PROF. GUÍA: NANCY HITSCHFELD KAHLER

SEGMENTACIÓN DENTAL BASADA EN DETECCIÓN DE PUNTOS CLAVES Y CAMPOS ARMÓNICOS

La ortodoncia invisible consta de alineadores o placas que son fabricadas en base a un escaneo tridimensional de la dentadura del paciente. Cada placa es diseñada en un software especializado, y es fabricada posteriormente con impresión 3D.

El alumno fue intervenido con un tratamiento de ortodoncia invisible, durante el cual se detectaron múltiples falencias. Debido a esto, se ideó un sistema para realizar tratamientos de esta índole solucionando las deficiencias observadas. Esta misión se enmarca en el nacimiento de la empresa TecDent, cuyo único propósito es entregar estos tratamiento de manera económica y cómoda.

La presente memoria consistió en el diseño del primer paso para poder realizar un tratamiento una vez que la dentadura del paciente ya ha sido digitalizada. Este primer paso es identificar de manera automática y correcta cada una de las estructuras anatómicas presentes en el escaneo tridimensional, es decir, cada uno de los dientes y encía. Este paso es fundamental, ya que para poder modificar la posición de los dientes cualquier programa o sistema requiere saber su posición inicial.

Después de cuatro implementaciones de algoritmos fallidos, aquella que resultó exitosa fue una adaptación obtenida desde la literatura. En esta, se detectan puntos claves dentro de los dientes de manera automática y se utilizan campos armónicos para detectar la forma final y posición de cada diente.

El trabajo fue probado con un total de veintidós modelos. Seis de estos obtenidos de medios públicos y los dieciséis restantes siendo tomados por una asistente dental en yeso, de pacientes que activamente buscaban un tratamiento de ortodoncia, para ser digitalizados posteriormente por un escáner tridimensional.

Como resultado el programa desarrollado logró segmentar correctamente las estructuras anatómicas en los modelos tridimensionales, debido a que en base a esta segmentación se fabricaron y ejecutaron tratamientos de ortodoncia de manera exitosa. El proceso de segmentación requiere aún en algunos casos de mínima intervención humana, siendo un sistema semi-automático.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

En este capítulo se explica el contexto en el cual se desarrolla esta memoria, la motivación que impulsó a desarrollar este trabajo, los objetivos del trabajo a realizar y la metodología para implementarlo. También se presentan brevemente todos los capítulos de esta memoria.

1.1. Contexto

El trabajo de memoria se encuadra en una solución, centrada en procesos computacionales, a los tratamientos de mal oclusión. La mal oclusión, es el alineamiento incorrecto de los dientes, que puede producir problemas funcionales que impactan en la salud bucal, y problemas estéticos.

Cuando la mal oclusión es lo suficientemente pronunciada se realiza un tratamiento de ortodoncia. Cuando el caso es extremadamente acentuado se realiza cirugía y posiblemente en combinación con aparatología fija, también conocida como brackets o frenillos. En casos en que la cirugía no es necesaria, se da una solución a la mordida únicamente mediante brackets.

Los brackets tienen varios formatos. Los más comunes, y económicos, son los metálicos, elaborados en acero inoxidable. Puesto que son muy notorios, existen alternativas más estéticas, en las que el material de fabricación es claro, habitualmente cerámica, transparente, fabricados en zafiro, o colocados de una forma sublingual (por las caras internas de los dientes).

Los brackets y técnicas de ortodoncia han sido revolucionadas desde la invención de *Invisalign*. Zia Chishti y Kelsey Wirth, establecen en la patente 6,217,325[4] (usualmente referida como '325) el primer sistema de alineamiento dental ocupando placas invisibles fabricadas por impresoras en 3D. Estas placas van paulatinamente cambiando desde la posición inicial del paciente a la posición deseada.

Desde su aparición, múltiples laboratorios competidores han iniciado actividades con un producto similar, con sutiles diferencias para circunnavegar la patente mencionada. A groso

Figura 1.1: Molde Dental

Figura 1.2: Placa *Invisalign*

modo los sistemas de alineadores invisibles, como el de *Invisalign* descrito en la patente '325, operan en 4 pasos:

1. Un ortodoncista obtiene un molde de yeso tradicional. Figura 1.1.
2. El molde es escaneado para ser procesado por computador, transformándose en un archivo 3D (.obj, .ply, .stl u otros).
3. El archivo es la entrada de un programa que permite realizar modificaciones manuales en la posición de los dientes. El laboratorio genera así varios archivos 3D, con los dientes ligeramente desplazados desde la situación anterior, hasta llegar a la posición final deseada. Este software alerta al laboratorista si está realizando un movimiento más allá que lo médicamente recomendable.
4. Una impresora 3D imprime cada uno de los archivos mencionados anteriormente. Aplicando plástico sellado al vacío sobre cada impresión, se obtienen las placas que utilizará el paciente. Figura 1.2.

En estas tecnologías, el diseño del plan de ortodoncia, es decir, los movimientos que se espera realizar sobre los dientes, son diseñados por el laboratorio. Asimismo sucede con la ejecución de este plan, puesto que las placas también son impresas por el laboratorio, a diferencia de los brackets tradicionales. De esta manera, el ortodoncista actúa principalmente como vendedor-asegurador-monitoreador de la solución, dejando su rol original, de implementador global, bastante relegado.

1.2. Motivación

El alumno fue tratado con el sistema *Invisalign*. Se fue a cotizar en conjunto con una segunda persona, en marzo del 2016, momento en que la solución valía un total de $3.000.000. Ya que para su situación era necesario, el alumno decidió tomar el sistema de todas formas, sin embargo, frente al prohibitivo costo su acompañante no tomó la solución.

Por otro lado, una vez que el alumno había empezado su tratamiento, se encontró un

estudiante de diseño 3D, Amos Dudley, que al informarse de los costos de *Invisalign* decidió realizar el tratamiento por su propia cuenta[6]. Amos, generó manualmente cada una de los alineadores necesarios, utilizando sus conocimientos en diseño 3D para modificar los archivos de sus moldes dentales mediante software de edición tridimensional. Logró realizar su tratamiento de forma exitosa por un costo total menor a sesenta dólares americanos.

En base a esto, se desprende que el costo real del material del tratamiento es muy bajo, y que la diferencia de precios entre el material y el tratamiento en sí, está dado principalmente por la comisión del ortodoncista, el laboratorista, y el valor del software que permite la edición de placas y la validación de que estas no realizan movimientos muy bruscos.

Frente a esta diferencia de precios surge la idea de realizar un sistema computacional que permita entregar una solución más barata a los usuarios finales, dentro de la cual se enmarca esta memoria. El objetivo es reducir las horas hombre del técnico laboratorista y ortodoncista requeridas para elaborar un tratamiento. Considerando los elevados costos de hora de ortodoncista, la única forma de alcanzar precios accesibles, es que su labor sea muy breve, siendo únicamente un "notario" del tratamiento, certificando que puede ejecutarse.

El sistema que se desea implementar, se encuadra dentro de un modelo que consta de las partes detalladas en la figura 1.3, donde la parte del sistema que atinge a esta memoria es la destacada en la figura, la obtención de la posición de cada diente. Cabe destacar que para poder mover los dientes dentro de un sistema computacional, sea de manera manual o automática, es fundamental tener la posición original de estos, luego este primer paso es indispensable.

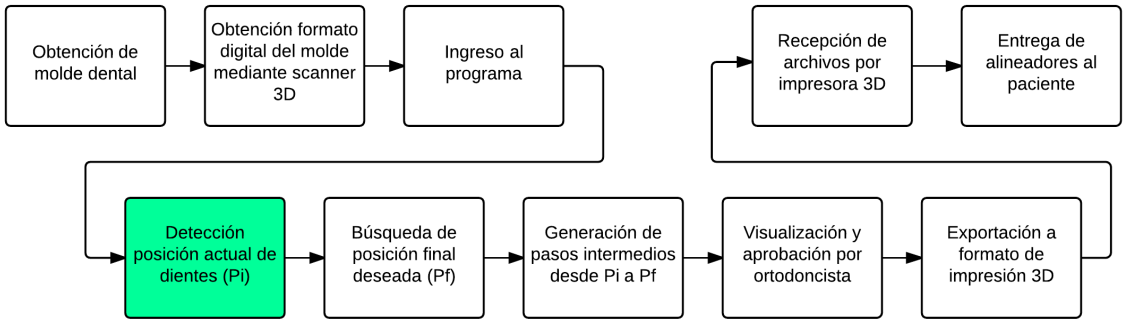


Figura 1.3: Proceso de Tratamiento

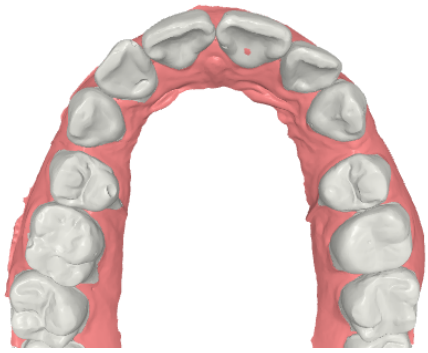
1.3. Objetivos

El objetivo principal es construir un software gráfico, que a partir de una entrada de un archivo tridimensional en formato .ply con el molde dental de una persona, reconozca cada uno de los dientes, separándolo de las encías y otros dientes, entregando como salida un mapa que indique por cada vértice de la malla a que diente pertenece o si es parte de la encía.

Los objetivos específicos son la correctitud, siendo un requisito de calidad, la automaticidad y salida interpretable, siendo requisitos de funcionalidad.

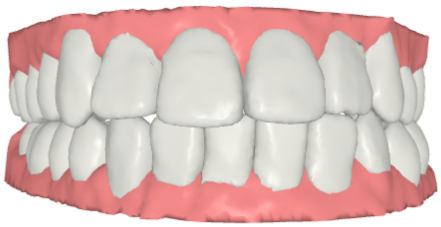
1.3.1. Correctitud

Puesto que el proceso de segmentación es el paso cero del sistema global, es fundamental que la tasa de reconocimiento correcto sea absoluta, a costo de tiempo computacional. Hay casos de mal oclusión severa, en que el sistema tiene permitido no detectar los dientes correctamente. Esto es puesto que los tratamientos que se pueden realizar con alineadores invisibles están limitados a casos simples[16], detallados en la figura 1.4



(a) Espacios

(b) Apiñamiento



(c) Mordida Cruzada

(d) Rotación e Inclinación

Figura 1.4: Casos Tratables

Para todos los casos de la figura 1.4, y aquellos sin problemas de oclusión, o sea, denticiones normales, se espera que el sistema reconozca siempre correctamente los dientes.

1.3.2. Automaticidad

Se espera que el sistema opere de forma completamente automática. Esto se refiere a que el modo de uso, consistirá en ingresar únicamente un archivo con los moldes dentales para obtener el resultado del programa, sin más interacción por el usuario. Aún con esto en consideración, y en función del objetivo de tener correctitud, se puede solicitar al usuario ingresar el número de dientes en el molde, para entregar al programa una verdad absoluta con

la cual puede comparar sus resultados. Aunque esto es aceptable, no es el comportamiento deseado.

1.3.3. Output Interpretable

Idealmente se espera obtener un archivo .csv que indique a qué diente pertenece cada vértice. Con la información de este archivo se debe poder colorear cada diente y encía de distinta forma, para que sea reconocible tanto que fueron detectados correctamente, como para que sea un archivo interpretable por otros programas dentro del proceso de tratamiento. En consecuencia, este archivo debe tener un número distinto para cada diente y para la encía, alguno de estos números será asignado a cada vértice según el diente o encía al que pertenezca.

En resumen, se espera que con una entrada de malla geométrica como la que se observa en la figura 1.5, se obtenga un archivo .csv que para cada vértice de la malla (figura 1.6) indique a qué diente o encía pertenece. En consecuencia, que permita colorear la malla como se observa en la figura 1.7. Esto debe ser realizado de manera automática y correcta.

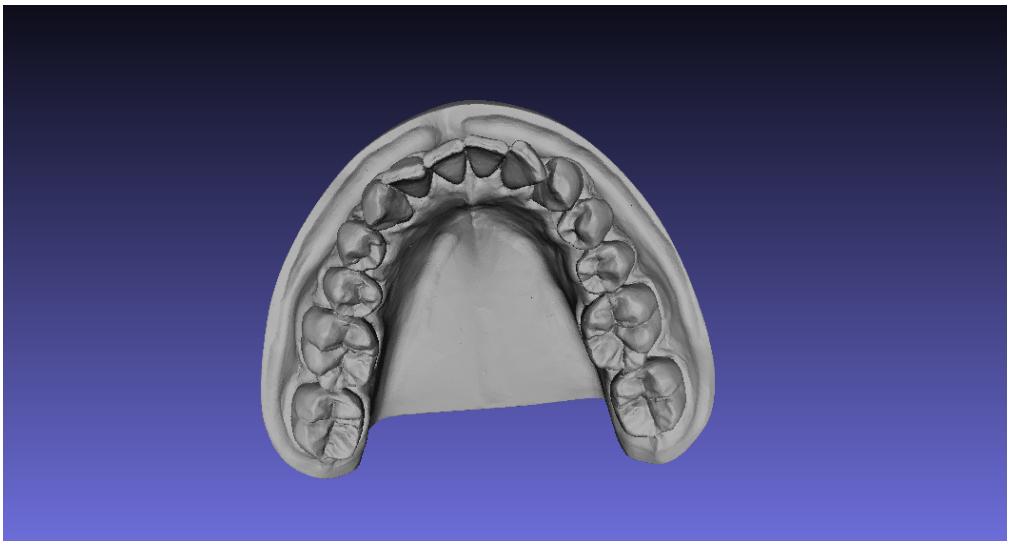


Figura 1.5: Visualización de la malla tridimensional de entrada al programa (archivo .ply)

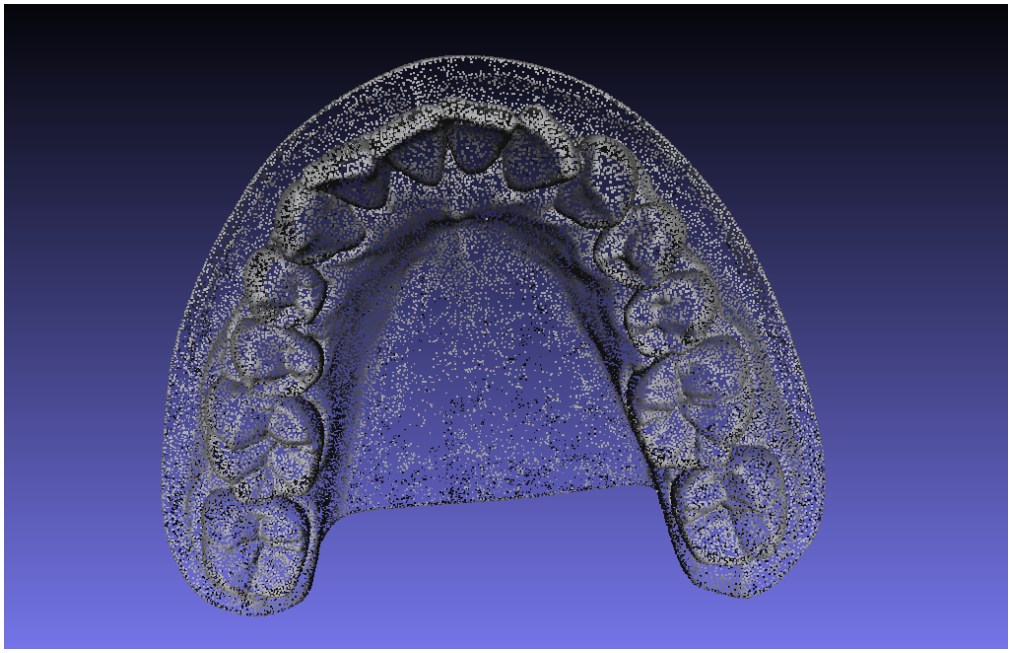


Figura 1.6: Visualización de los vértices de la malla tridimensional de entrada al programa (archivo .ply)

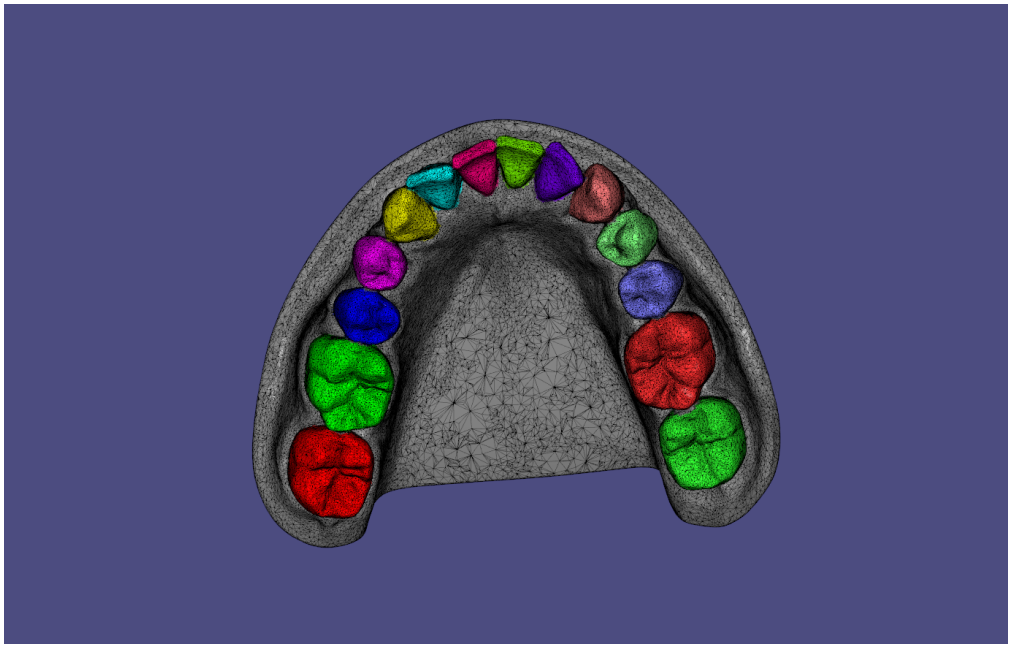


Figura 1.7: Visualización de los dientes y encía coloreados correctamente acorde a archivo .csv de salida

1.4. Metodología

La metodología consistió en buscar publicaciones con soluciones existentes para detectar dientes e implementar sus algoritmos. Viendo sus resultados se modificó y mezcló algoritmos para llegar al desempeño deseado.

En particular, se seleccionaron tres publicaciones[21][14][11] para segmentar dientes con los mejores resultados entre las existentes en la literatura, que son detalladas más adelante dentro de este trabajo. Estas son implementadas y modificadas hasta encontrar aquella que cumple con los objetivos ya explicados. Se valida de manera visual que las segmentaciones sean correctas, probando el sistema implementado en veintidós moldes dentales.

1.5. Contenido de la Memoria

El trabajo presente está dividido en seis capítulos. El capítulo dos detalla la documentación encontrada y el estado del arte existente en segmentaciones dentales, describiendo las tres publicaciones que se implementan y las herramientas con las que se lleva a cabo. El capítulo tres presenta en detalle los algoritmos implementados, cuyos resultados están considerablemente lejos de una segmentación dental correcta. Con la información obtenida en el capítulo tres, el cuarto capítulo describe el algoritmo implementado que se acerca a los objetivos planteados en la memoria. Los resultados de este algoritmo se presentan en el capítulo cinco. Concluyendo en el capítulo seis con la verificación de los objetivos, los aprendizajes obtenidos, el trabajo futuro y los desafíos post segmentación. Finalmente, se detallan los tratamientos actualmente realizados con el sistema creado en este trabajo.

Capítulo 2

Preliminares

En este capítulo se presentan algoritmos y conceptos que se utilizarán frecuentemente durante este trabajo. También se detallan las tres publicaciones encontradas en la literatura con los mejores resultados. Finalmente se explican los lenguajes de programación, librerías y tecnologías utilizadas, junto con la justificación de su uso.

2.1. Algoritmos y Conceptos

2.1.1. Concepto: Mallas Geométricas Triangulares

En esta memoria se trabaja únicamente con mallas triangulares. Una malla triangular es un conjunto de triángulos con sus puntos en el espacio XYZ, los cuales pueden estar unidos en sus bordes o en sus esquinas. Una representación computacional simple de este concepto es la utilizada por *libigl*[12]. Donde los puntos son una matriz V de de n por 3, donde n es el número de puntos, y las caras son una matriz F de m por tres, donde m es el número de caras.

$$V = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 3 & 2 \end{pmatrix}$$

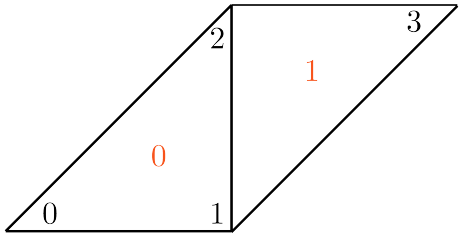


Figura 2.1: Malla simple de 2 triángulos y 4 vértices

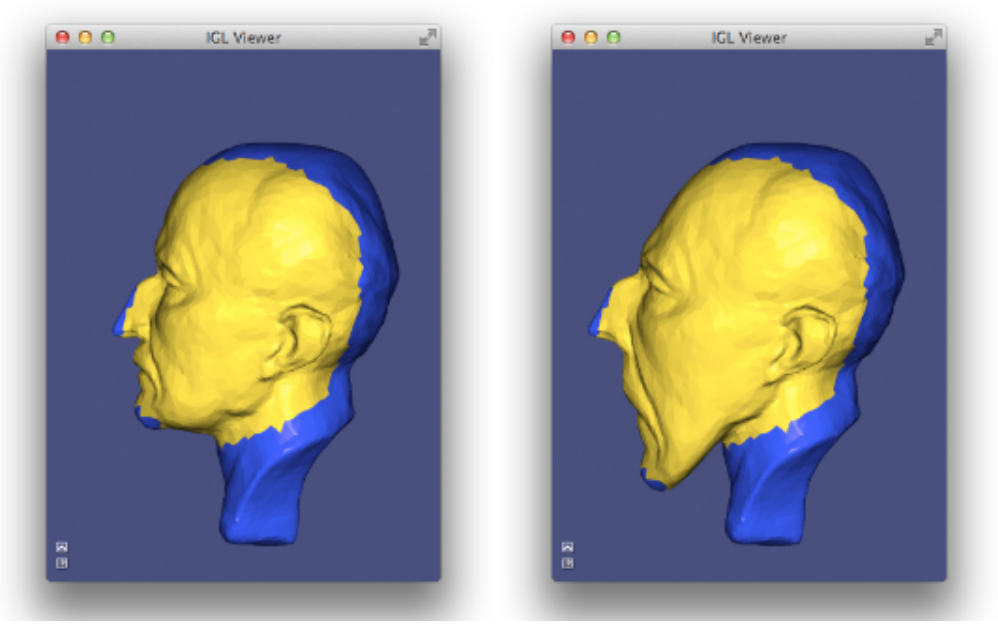


Figura 2.2: Deformación de cabeza de hombre. Puntos en dorado son libres, puntos azules condiciones de dirichlet.

2.1.2. Concepto: Campos Armónicos en Mallas Geométricas

Los campos armónicos son un herramienta matemática utilizable en mallas geométricas que resulta de suma utilidad para realizar deformaciones (Figura: 2.2), mapeo de figuras (Figura: 2.3), segmentaciones (Figura: 2.4) y otras. En este trabajo se utilizarán los campos armónicos únicamente para realizar segmentaciones. Los campos armónicos modelan un proceso de difusión dentro de una malla, es decir, permiten saber qué tan relevante es un vértice sobre otro.

A modo de ejemplo, en la deformación de la figura 2.2 se puede observar que el mentón del hombre es empujado hacia abajo. Los vértices más cercanos al mentón, ubicados en la boca y quijada son mayormente deformados, sin embargo nariz, ojos, orejas y frente son deformadas de menor manera, ya que se encuentran más alejadas del mentón. Este proceso de deformación difundida, donde paulatinamente los vértices son deformados de menor manera a medida que se alejan del mentón es lo que permite modelar un campo armónico.

La base matemática de los campos armónicos proviene de las funciones armónicas. Una función armónica es una función doblemente continua y diferenciable que satisface la ecuación de Laplace:

$$\frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2} + \cdots + \frac{\partial^2 f}{\partial x_n^2} = 0$$

Un campo armónico es el resultado de resolver una ecuación de Laplace, o Laplaciano, con ciertas condiciones de borde. En este trabajo se utilizan condiciones de borde de Dirichlet.

Puesto que una malla geométrica es una representación discreta del espacio, el Laplaciano debe ser discretizado. Esto puede ser realizado de múltiples formas. La manera más simple

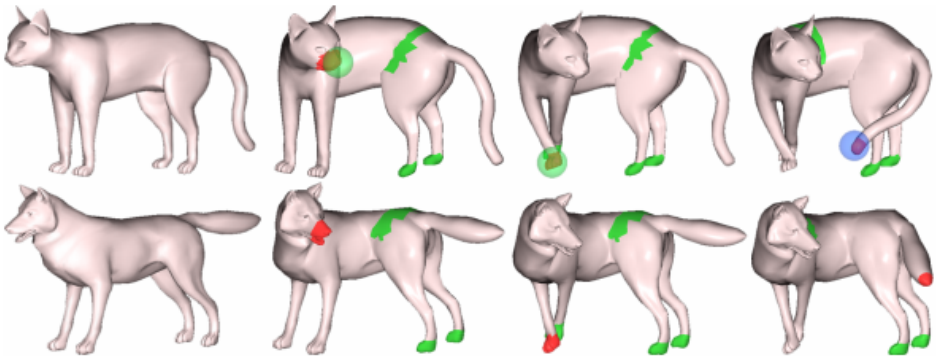


Figura 2.3: Mapeo de movimientos en un gato a un lobo utilizando campos armónicos. Este sistema se ocupa para reutilizar animaciones preexistentes en nuevas mallas.

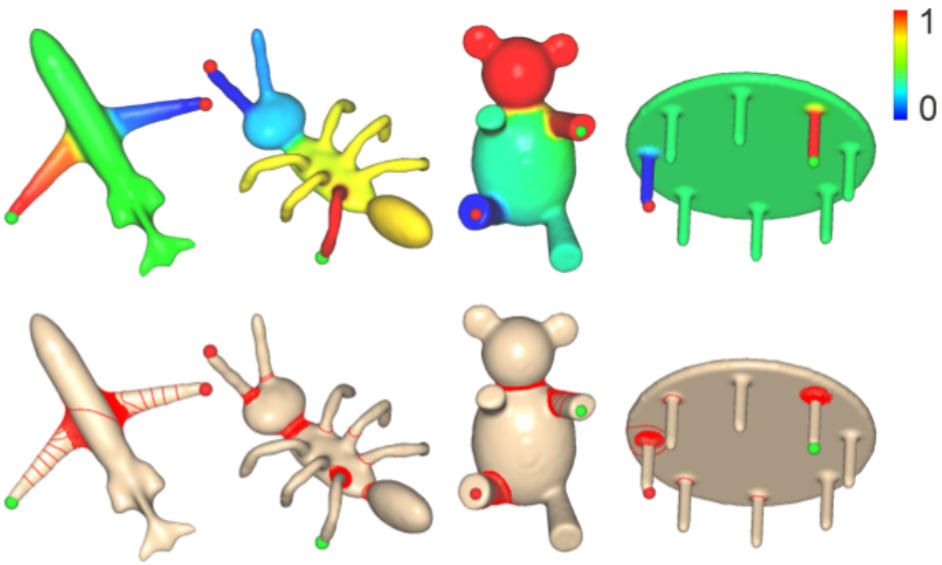


Figura 2.4: Segmentación de partes de mallas geométricas utilizando campos armónicos.

es utilizar directamente las distancias entre los vértices. Sin embargo, un esquema habitualmente utilizado en mallas geométricas es la discretización de cotangentes. Las ecuaciones que permiten realizar esta discretización serán vistas en detalle en la sección del algoritmo de segmentación mediante campos armónicos.

Lo que se desea destacar en esta sección preliminar, es que los campos armónicos en mallas geométricas, son una herramienta matemática que permite modelar procesos de difusión, que tienen diversas aplicaciones.

2.1.3. Algoritmo: Region Growing

El algoritmo de *region growing* es utilizado para segmentar grupos en una malla de puntos. Funciona eligiendo un punto dentro de la malla y agrupándolo iterativamente con sus vecinos que no tienen un cambio abrupto de curvatura hasta que no quedan más puntos a agregar. En este instante elige un nuevo punto de la malla que no esté agrupado y vuelve a realizar el procedimiento de detectar vecinos hasta que no queden puntos para agregar. Así continúa hasta haber obtenido todos los grupos de la malla.

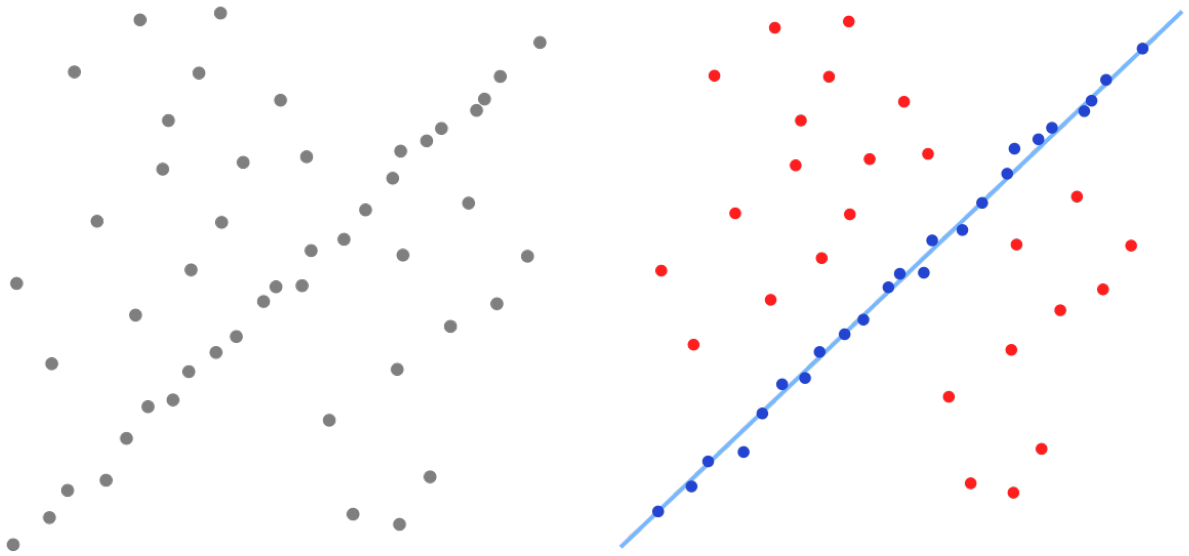
Un buen ejemplo de su uso es cuando se tiene una mesa con un conjunto de objetos. Este algoritmo permitirá entregar cada uno de estos objetos y la mesa como un grupo o clúster de puntos distinto.

El algoritmo sigue los siguientes pasos:

1. Los puntos de la malla son ordenados de mayor a menor según su curvatura.
2. Se elige el punto de menor curvatura y se lo agrega a un conjunto de puntos, sea este *semillas*.
3. Se encuentran todos los vecinos de este punto. Si la curvatura entre el punto original y el vecino es menor que un parámetro predefinido, se agrega al conjunto de *semillas*.
4. Se verifican iterativamente los vecinos de los puntos recientemente agregados al conjunto de *semillas*.
5. Cuando en una iteración no se agrega ningún punto a *semillas*, *semillas* es descontado de la malla original y se vuelve al paso 2.

2.1.4. Algoritmo: RANSAC

RANSAC o *random sample consensus* es un método iterativo para ajustar un conjunto de datos a un modelo matemático. Por ejemplo, si se tiene un conjunto de puntos que se sabe que se pueden aproximar a una recta, pero se desconoce la ecuación de la recta en sí misma, se puede utilizar *RANSAC* para encontrarla. Siguiendo este ejemplo, el algoritmo creará una ecuación de la recta cualquiera y verá que porcentaje de los datos se ajustan a ella, con cierto grado de tolerancia. Luego de realizar este proceso un determinado número de veces, definido por el implementador, entregará la recta que obtuvo un mayor porcentaje de ajuste. Los resultados de *RANSAC* acorde al ejemplo se pueden observar en la figura 2.5



(a) Datos que se pueden ajustar a una ecuación de la recta, pero esta es desconocida.

(b) Ecuación de la recta en azul que se ajusta al modelo. En rojo *outliers*.

Figura 2.5

En mallas geométricas, *RANSAC* sigue el mismo proceso que en el ejemplo sólo que en tres dimensiones. Cabe destacar que solo puede ajustar a modelos definibles por una ecuación matemática, como esferas, conos, cilindros o planos. Para objetos que no tienen una clara ecuación matemática que los define en tres dimensiones, como un diente, u otro objeto cualquiera de mayor complejidad, no es posible utilizar *RANSAC* para detectarlos con exactitud.

2.1.5. Algoritmo: Watershed

El algoritmo de *watershed* se utiliza para encontrar mínimos o máximos locales de manera eficiente dentro de un conjunto de datos. Para este trabajo los datos serán alturas de puntos dentro de mallas geométricas. Este algoritmo se para en cada punto de la malla y desciende (o asciende) hacia su vecino más profundo (o alto), hasta que llega a un punto en el cual todos sus vecinos son más altos (o bajos) que él, y lo define como un mínimo (o máximo) local. Se puede observar un ejemplo en dos dimensiones en la figura 2.6.

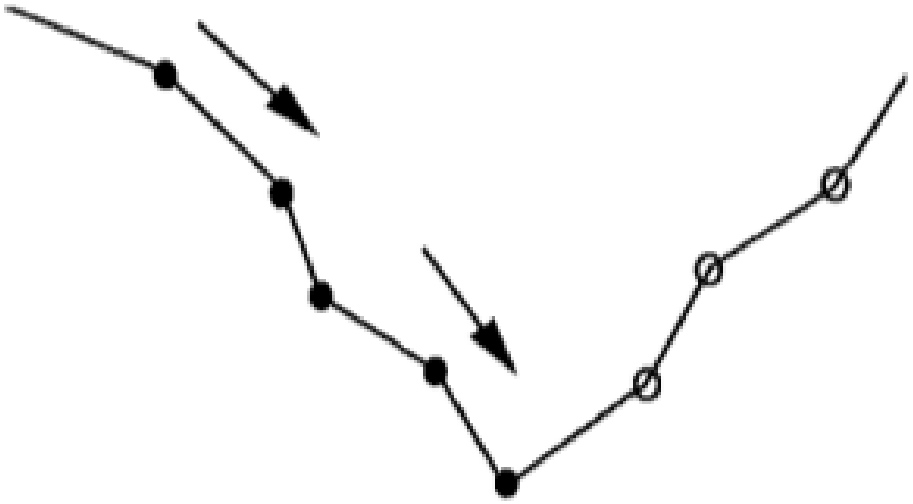


Figura 2.6: Se observa el descenso de un punto indicado por las flechas hasta un mínimo local

2.2. Publicaciones

2.2.1. Método basado en detección de puntos céntricos

El primer paper, "*Semi-automatic teeth segmentation in 3D models of dental casts using a hybrid methodology*"[21] describe un sistema que consta de tres pasadas por distintos algoritmos. Primero se utiliza un algoritmo de *region growing*[18] para segmentar la encía de los dientes. Como toda la encía es un área con curvas suaves, y al llegar a los dientes hay un cambio abrupto, se obtiene un gran clúster con todos los puntos pertenecientes a la encía. Esto queda ejemplificado en la figura 2.7.

Una vez removida esta, se realiza *NARF*[20] (*Normal Aligned Radial Features*) para obtener puntos clave de los dientes. *NARF* es un algoritmo que detecta bien puntas de superficies. Revisa que las distancias entre los puntos de un área sean relativamente constantes, y resalta los puntos en que hay un cambio abrupto. Por ejemplo, en una mesa se obtiene una diferencia de distancias entre un punto y otro relativamente constante en toda la superficie interna, hasta llegar a un borde, donde el cambio es abrupto. Esos puntos son destacados como puntos claves. El algoritmo de segmentación, espera que al aplicar *NARF* sobre los dientes, los puntos extraídos sean idealmente detectados en el centro de cada diente. Se puede observar un caso ideal en la figura 2.8.

Posteriormente a esto, se realiza un algoritmo de *min-cut* aplicado a nubes de puntos[9], que es usualmente usado para segmentar cilindros de un medio con ruido. Al ser aplicado en un diente se obtiene un resultado aproximado apreciable en la figura 2.9.

Esta publicación testea sus datos en seis modelos dentales, y entrega sus diferencias de resultados contra una segmentación perfecta de forma visual (sin mediciones).

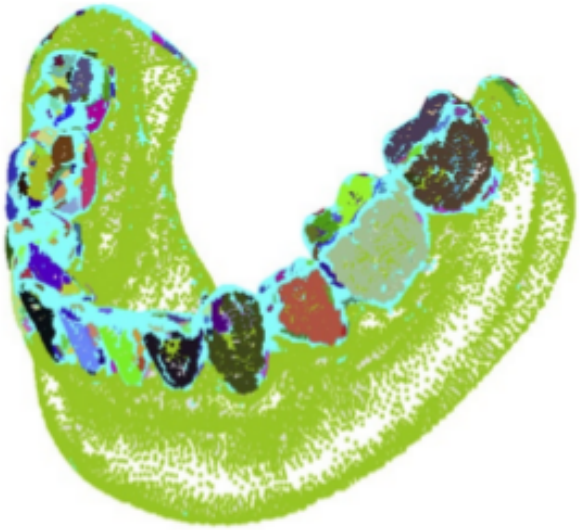


Figura 2.7: Malla geométrica dental. La encía ha sido detectada con *region growing* quedando destacada en verde. El resto de los colores corresponde a los demás clústers.

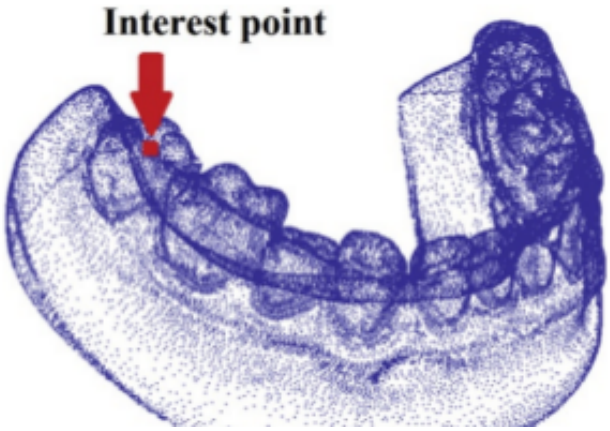


Figura 2.8: Punto céntrico de molar detectado con NARF

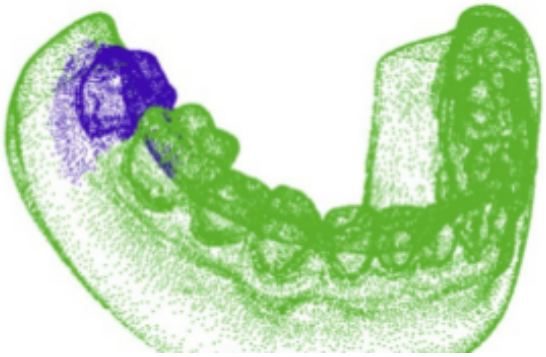


Figura 2.9: Segmentación dental con Min-Cut

Figura 2.10: Modelo dental con los puntos más altos marcados en negro.

2.2.2. Método basado en detección de planos en intersticios

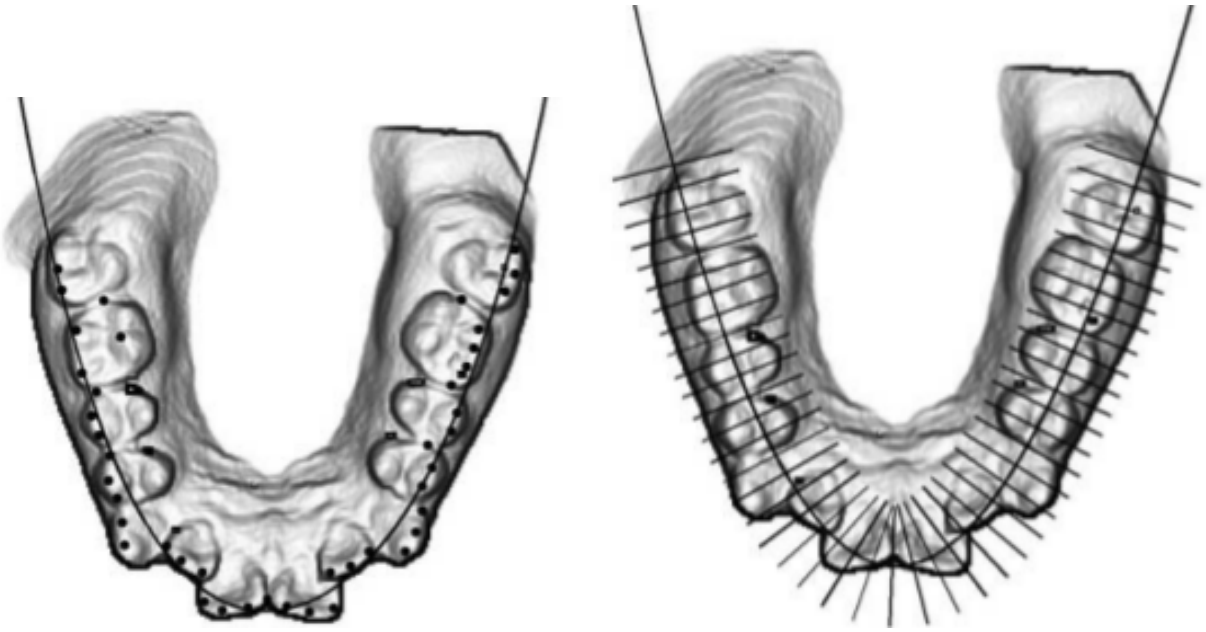
La segunda *publicación*, "*Tooth Segmentation of Dental Study Models Using Range Images*"[14] busca encontrar los intersticios dentales o planos que dividen a un diente con otro usando un algoritmo exclusivamente pensado para esto, con conciencia de la geometría de la dentadura humana. Primero se extraen los puntos claves de los dientes (siendo estas las partes más altas de los dientes) con un algoritmo de *watershed* identificándolos como máximos locales, esto se aprecia en la figura 2.10.

En base a la identificación de estos puntos, se estima una curva de orden cuarto que se aproxime bien a ellos. Sobre esta curva se establecen todos los posibles planos que dividen los dientes. La curva y planos posibles se pueden observar en la figura 2.11.

Para cada uno de estos planos se revisa si la curvatura de los puntos en ese plano es alta, y si el punto más alto en el plano es bajo. Si ambas condiciones se dan, entonces el plano es un intersticio dental. Se documenta que se realizaron pruebas en treinta modelos dentales, equivocándose únicamente en seis intersticios dentales. El resultado final del algoritmo se refleja en la figura 2.12.

2.2.3. Método basado en detección de puntos claves y campos armónicos

La tercera publicación, "*Automatic Tooth Segmentation of Dental Mesh Based on Harmonic Fields*"[11] propone extensiones sobre la segunda. Plantea que segmentar los dientes en base a un plano, o al intersticio calculado, puede generar cortes indeseados, como sobre o bajo segmentaciones, debido a que la orientación del diente no es necesariamente igual al plano de corte. Por esto, propone ocupar campos armónicos para segmentar. Estos campos



(a) Curva detectada del arco dental (b) Posibles planos de división

Figura 2.11: Curva detectada y posibles planos de división

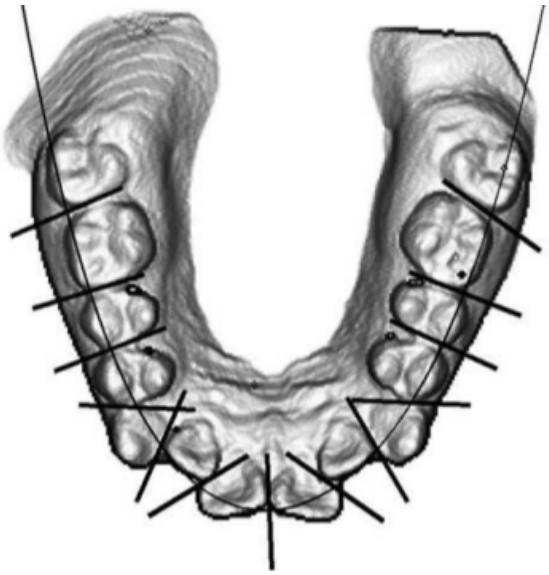


Figura 2.12: Planos de intersticios correctamente detectados.

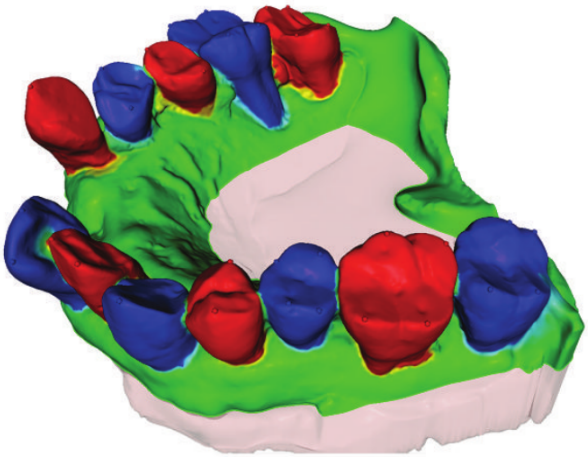


Figura 2.13: Dientes coloreados intercaladamente y encía coloreada en verde

obtienen sus condiciones de borde en base al sistema de la segunda publicación, es decir, se segmentan los puntos claves de los dientes usando planos, y estos puntos son las condiciones de Dirichlet de los campos armónicos. En esta publicación se produce una segmentación mucho más exacta que en los anteriores, validando el sistema en sesenta moldes. El error promedio es documentado como un milímetro, y el error máximo es de 1.2 milímetros. Se puede observar un modelo segmentado con este algoritmo en la figura 2.13.

2.3. Tecnologías y Materiales de Trabajo

Para el análisis de los modelos tridimensionales, se utilizan dos librerías. Una, la *Point Cloud Library*[19], es una librería de visión computacional implementada en C++. La segunda, *libigl* es una librería de manejo tridimensional para uso científico implementada en C++, diferenciada de las más conocidas *VTK* o *CGAL* por sus estructuras de datos simples y fácil utilización. El uso de estas tiene como objetivo aprovechar la batería de algoritmos existentes en ellas, permitiendo así prototipar rápidamente, para ver qué algoritmos funcionan y cuáles no, sin tener que asumir el costo de una implementación desde cero. Por otra parte, para realizar visualizaciones y ajustes manuales a las mallas, se utilizó el manipulador geométrico *Meshlab*[5].

Capítulo 3

Implementación de Algoritmos Deficientes

En este capítulo se detallan todos los algoritmos implementados que no entregaron segmentaciones adecuadas para poder realizar un tratamiento de ortodoncia. Se explican dos algoritmos obtenidos de la literatura y dos algoritmos originales. Finalmente se entregan la conclusiones y explicaciones del por qué los resultados fueron fallidos.

3.1. Algoritmo basado en puntos céntricos

Se implementó el algoritmo explicado en una publicación encontrada en la literatura, "*Semi-automatic teeth segmentation in 3D models of dental casts using a hybrid methodology*"[21].

Este algoritmo asume que cada diente puede ser aproximado a un cilindro y que la encía es una masa tridimensional uniforme, sin grandes irregularidades. También asume que existen puntos de grandes diferencias con el entorno cerca del centro de cada diente.

En base a esto se reconocen puntos claves de gran diferencia con el entorno en el archivo tridimensional de entrada. Para cada uno de estos puntos se estima un cilindro alrededor y se determina que este es un diente. Por otro lado se reconoce el mayor conjunto de vértices conectados sin mayores irregularidades y son determinados como la encía.

Para la implementación se utilizaron los procedimientos de *region growing*, *narf* y *min-cut* implementados por la *Point Cloud Library*, y explicados en la sección 2.2.1.

Los pasos a seguir por el algoritmo son:

1. Retirado de la encía mediante *Region Growing*.
2. Detección de puntos claves mediante *NARF*.
3. Para cada punto clave detectado con *NARF*, estimar el cilindro que lo rodea mediante

min-cut, definiéndolo como un diente.

4. Entregar la encía y dientes detectados como salida.

En pseudo código:

Algoritmo 1 Algoritmo basado en puntos céntricos

```
puntos ← Nube de puntos original
clustersBajosCambios ← REGIONGROWING(puntos)
encia ← MAX(clustersBajosCambios)
puntosSinEncia ← (puntos − encia)
puntosClaves ← NARF(puntosSinEncia)
teeth ← ∅
for all puntosClaves do
    tooth ← MINCUT(puntosClave)
    teeth.PUSH(tooth)
csv ← TOCSV(teeth, encia)
    return csv
```

Este algoritmo se implementó con resultados extremadamente deficientes, lejos a los esperados y a los declarados en la publicación. Es posible que la implementación realizada haya contenido errores. Sin embargo, el remover la encía y puntos no pertenecientes a la dentición propiamente tal mediante *Region Growing* sí dio resultados utilizables, y también se obtuvo valiosa experiencia con la *Point Cloud Library*. Con esta implementación, se descartó el uso de *NARF* más *Min-Cut* para segmentar los dientes entre sí.

3.2. Algoritmo Original, RANSAC-RG

Por los malos resultados obtenidos con el algoritmo basado en puntos céntricos, se decidió modificar la implementación pero manteniendo la premisa de que cada diente puede ser aproximado a un cilindro y que la encía es una masa tridimensional uniforme.

Manteniendo esta premisa geométrica se diseño el algoritmo original *RANSAC-RG*. En este algoritmo se cortan las áreas inferiores de los dientes. Con el objetivo de obtener cilindros más precisos, como se observa en la figura 3.1c. Hecho esto, los cilindros son detectados y se asume que cada uno es la parte superior de un diente. A estos puntos detectados se le agregan las partes inferiores del diente cortadas anteriormente.

Para esta implementación se utiliza *region growing* explicado anteriormente y el algoritmo *RANSAC*[8] que intenta ajustar como modelo un plano y un cilindro.

Los pasos a seguir por el algoritmo son:

1. Retirado de la encía mediante *Region Growing*. Figura 3.1b.
2. Corte planar de los dientes mediante *RANSAC*. Se puede observar de la figura anterior que un plano captura la mayor cantidad de puntos cortado todos los dientes cerca de

la parte superior, dejando los pequeños cilindros esperados. Figura 3.1c.

3. Detección de cilindros sobre el plano mediante *RANSAC*, figura 3.1d, crecimiento posterior mediante *Region Growing* (para evitar detectar varias veces el mismo círculo), y retirado de la nube de puntos del corte hasta que se haya cubierto un 90 % de los puntos. Figura 3.1e.
4. Extensión de estos clústers a las partes inferiores de los dientes mediante *Region Growing*. Figura 3.1f.

En pseudo código:

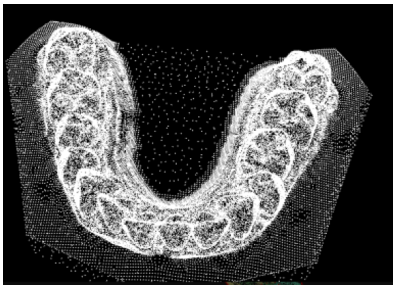
Algoritmo 2 Algoritmo RANSAC-RG

```
puntos ← Nube de puntos original
clustersBajosCambios ← REGIONGROWING(puntos)
encia ← MAX(clustersBajosCambios)
puntosSinEncia ← (puntos − encia)
planoCilindros ← RANSAC(puntosSinEncia, plano)
tamañoPlanoCilindrosOriginal ← planoCilindros.size
cilindros ← ∅
while tamañoPlanoCilindrosOriginal ∗ 0,9 > cilindros.size do
    cilindroActual ← RANSAC(planoCilindros, cilindro)
    cilindroActual ← REGIONGROW(cilindroActual)
    cilindros.PUSH(cilindroActual)
    planoCilindros ← (planoCilindros - cilindroActual)
teeth ← ∅
for all cilindros do
    tooth ← REGIONGROWING(cilindro)
    teeth.PUSH(tooth)
csv ← TOCSV(teeth, encia)
    return csv
```

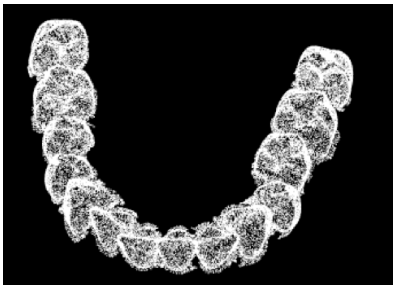
Se puede observar en la figura 3.1e que la realización de *region growing* sobre el cilindro no es perfecta, puesto que se absorbe parte del diente adyacente, más aún, empeora bastante en la extensión final observada en la figura 3.1f, identificando hasta tres dientes como uno solo. Puesto que estos resultados tampoco satisfacían los objetivos, se intentó un último algoritmo con la premisa que un cilindro se ajusta bien a un diente, detallado a continuación.

3.3. Algoritmo Original, Random-RG

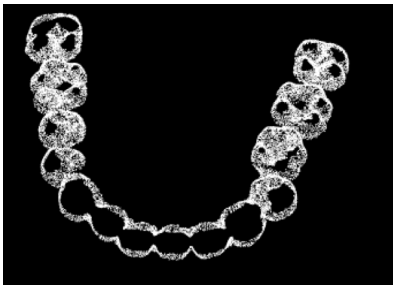
Como se explicó anteriormente, en el algoritmo RANSAC-RG, se absorbe parte del diente adyacente al realizar *region growing* sobre el modelo de un cilindro. Debido a esto, se decidió invertir el orden de estas operaciones. Es decir, el algoritmo Random-RG selecciona aleatoriamente una región de la malla geométrica y verifica posteriormente si esta es o no un cilindro. Si lo es, se asume que es un diente, y se sigue iterando con los puntos restantes de la malla.



(a) Original

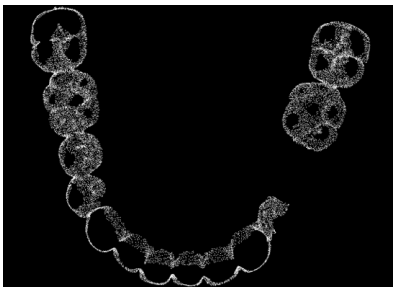


(b) Eliminada encía mediante *Region Growing*

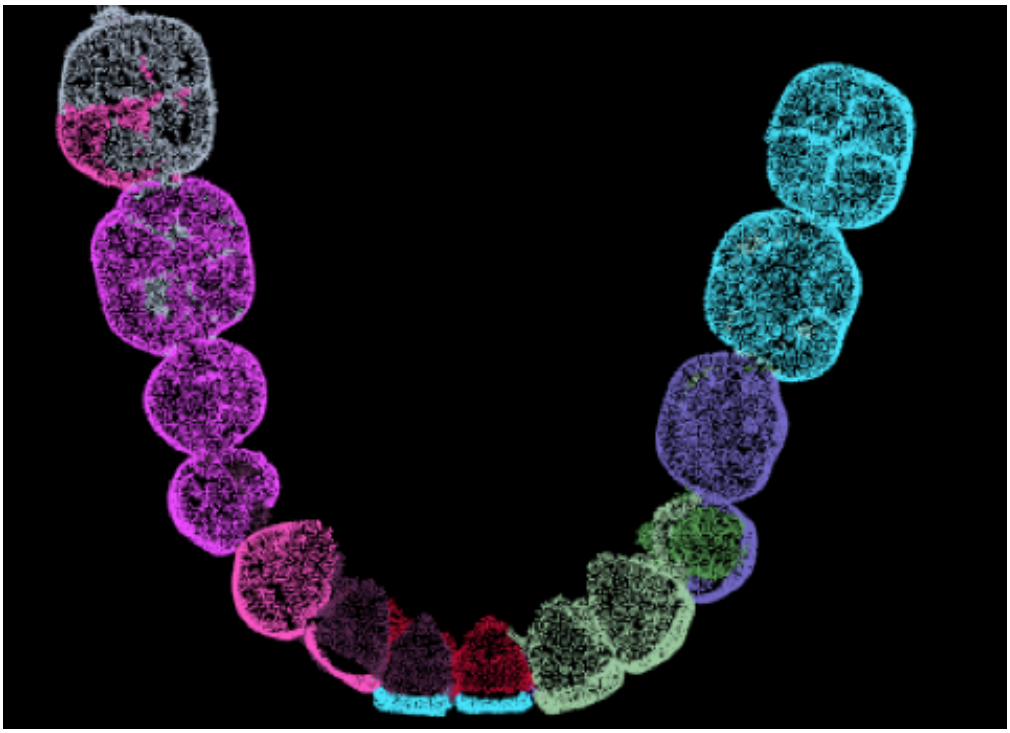


(c) Corte planar mediante *RANSAC*

(d) Detección de cilindro mediante *RANSAC*. Cilindro detectado en rojo.



(e) Eliminación de cilindro, una vez realizado *Region Growing*



(f) Extensión de los Clústers

Figura 3.1: Pasos Algoritmo RANSAC-RG

Para su implementación, se realiza *region growing* sobre la malla iterativamente. Como ya fue explicado el algoritmo de *region growing*, va uniendo puntos vecinos cuando existe una baja diferencia de curvatura entre estos, siendo la diferencia de curvatura un parámetro modificable. En cada iteración se cambia este parámetro y se obtiene un subconjunto distinto de regiones, o candidatos a dientes. Estos candidatos son verificados por una función que determina si son o no un diente. Si es un diente, es eliminado del espacio de búsqueda, agregado al listado de dientes y se prosigue con nuevos parámetros.

Puesto que los incisivos tienen dos caras divididas por un área de muy alta curvatura, *region growing* difícilmente las considera como el mismo clúster. Debido a esto, el algoritmo aleatoriamente une dos candidatos a dientes, esperando unir las dos caras de un incisivo.

La función que verifica si un conjunto de puntos es un diente o no, revisa dos condiciones. Una, todos los puntos deben formar una única componente conexa. La otra, que los puntos deben ajustarse mejor al menos un 50 % a un cilindro que a un plano. Esto se hace realizando *RANSAC* con ambas figuras sobre los puntos, y comparando la cobertura que estas producen.

En pseudo código:

Algoritmo 3 Algoritmo Random-RG

```
puntos ← Nube de puntos original
clustersBajosCambios ← REGIONGROWING(puntos)
encia ← MAX(clustersBajosCambios)
puntosSinEncia ← (puntos − encia)
teeth ← ∅
while puntosSinEncia.size > 0 do
    regionGrowingCurvature ← RANDOM(0,1, 10)
    candidatosTooth ← REGIONGROWING(puntos)
    if RANDOM(0, 1) > 0,7 then
     UNIRCANDIDATOS(candidatosTooth)
    for all candidatosTooth do
        if ISTOOTH(candidatoTooth) then
            teeth.PUSH(candidatoTooth)
            puntosSinEncia ← (puntosSinEncia - candidatoTooth)
csv ← TOCSV(teeth, encia)
     return csv
```

El gran problema de este algoritmo, fuera de su tiempo de ejecución de alrededor de cuatro minutos por dentadura, es que la mitad de un premolar, es sumamente similar a un canino, por lo que la función que verifica si los puntos entregados son un diente o no, acepta esta mitad de diente por confundirla con un canino.

En la figura 3.2 se puede observar que tres de cuatro premolares, y un molar tienen mitades que han sido detectadas como caninos generando dos dientes cuando existe uno solo. El problema es que teniendo una visión de que un diente se puede ajustar a un cilindro, no se puede evitar este error. Luego, después de tres implementaciones fallidas, se descarta la opción de modelar los dientes como cilindros al ser una aproximación demasiado burda, y

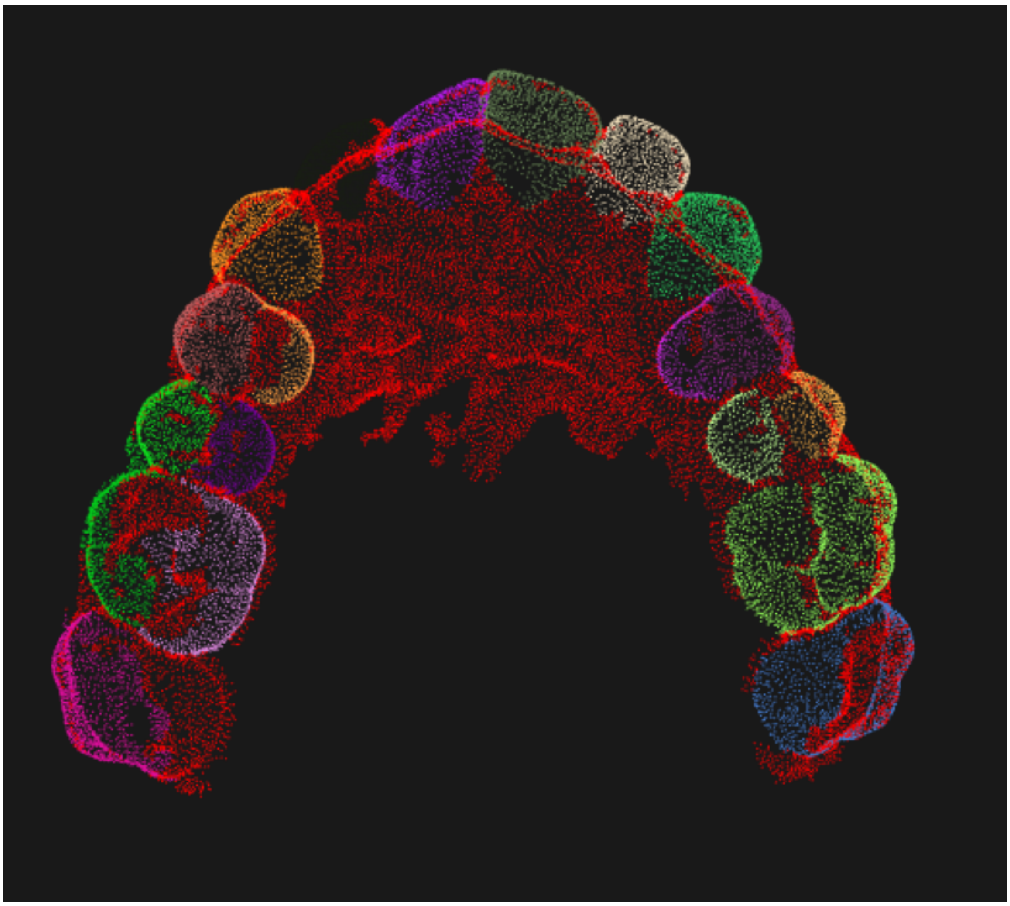


Figura 3.2: Resultado Algoritmo Random-RG. Se observa que tres de cuatro premolares y un molar fueron detectados como dos dientes en vez de uno.

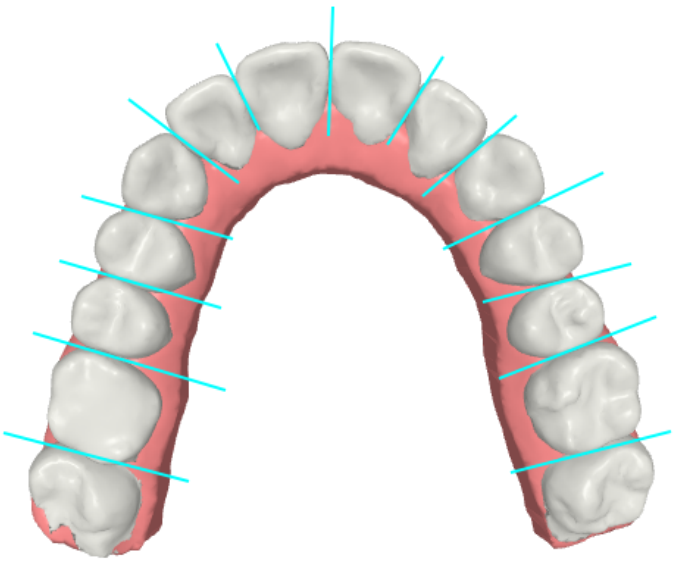


Figura 3.3: Dentadura con intersticios demarcados en celeste

se afirma que es necesario otro modelo de los dientes para poder realizar una segmentación exacta.

3.4. Algoritmo de Cortes Planares

En la búsqueda de un mejor modelo para representar y segmentar los dientes, se acude a la literatura, implementando un algoritmo de cortes planares[15]. A diferencia de los métodos planteados anteriormente, este algoritmo no busca encontrar los dientes en sí, si no que los intersticios dentales que los dividen, observables en la figura 3.3.

Para esto, se asume que un intersticio tiene una mezcla de dos características. La primera, es que los puntos alojados en un intersticio tienen una menor altura que los puntos alojados en dientes. La segunda, es que los puntos de un intersticio tienen una mayor curvatura que los puntos de los dientes. Estas dos características se pueden observar en las figuras 3.4, 3.5, 3.6 y 3.7.

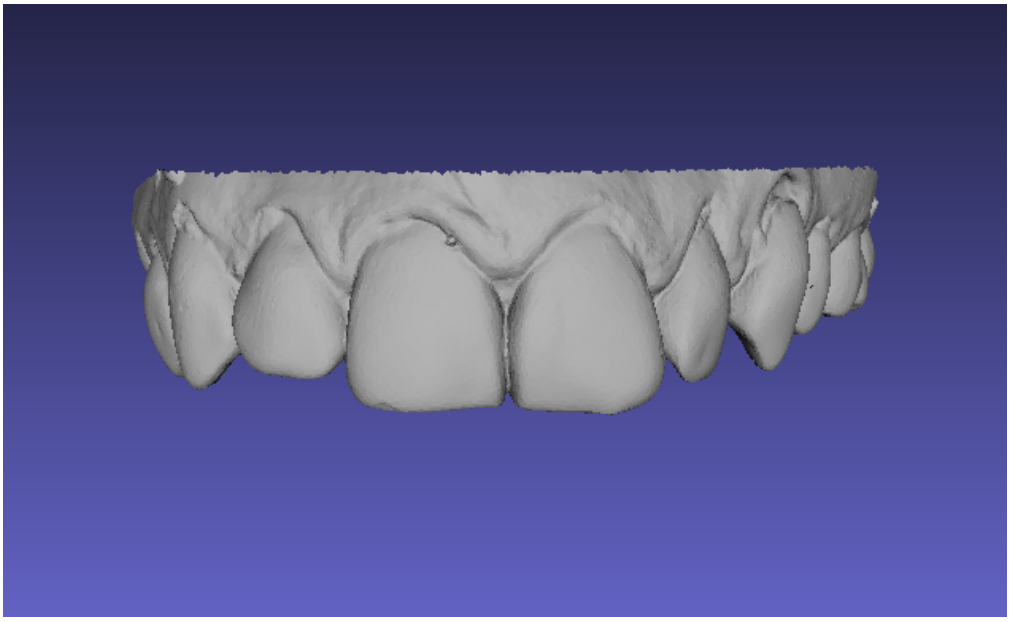


Figura 3.4: Vista de los dientes anteriores. Se puede observar que hay depresiones de altura entre los dientes.

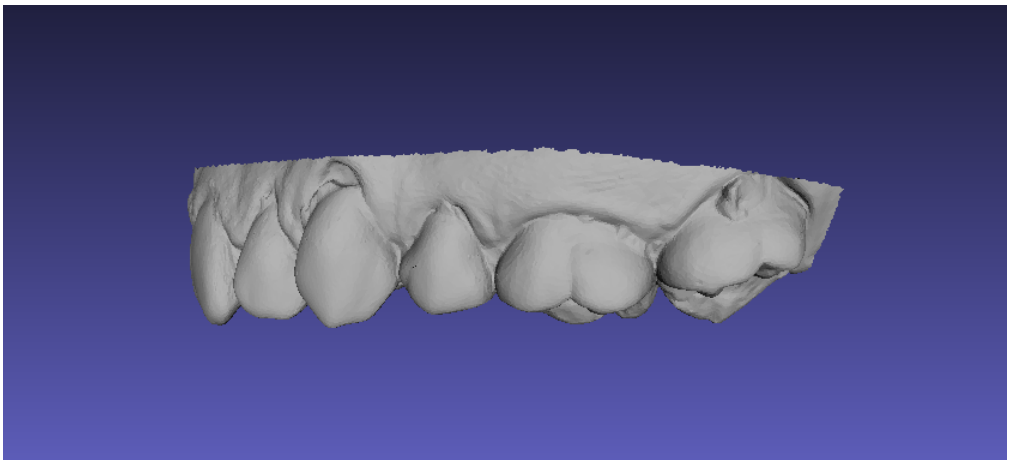


Figura 3.5: Vista de los dientes posteriores. Las depresiones de altura entre los dientes son más pronunciadas que en los dientes anteriores.

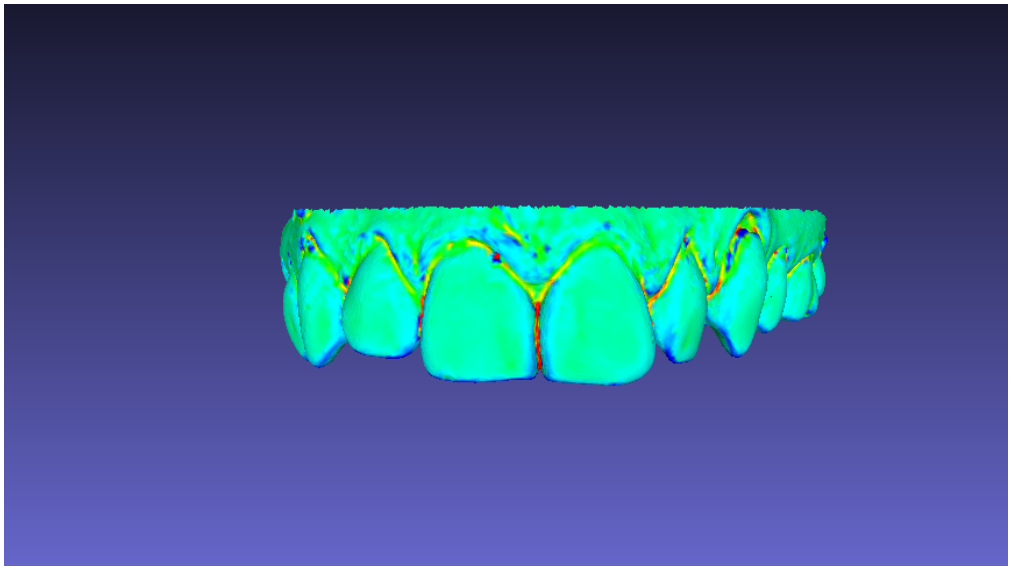


Figura 3.6: Vista de mapa de calor de la curvatura de los dientes anteriores. Máxima curvatura en rojo. Mínima en azul.

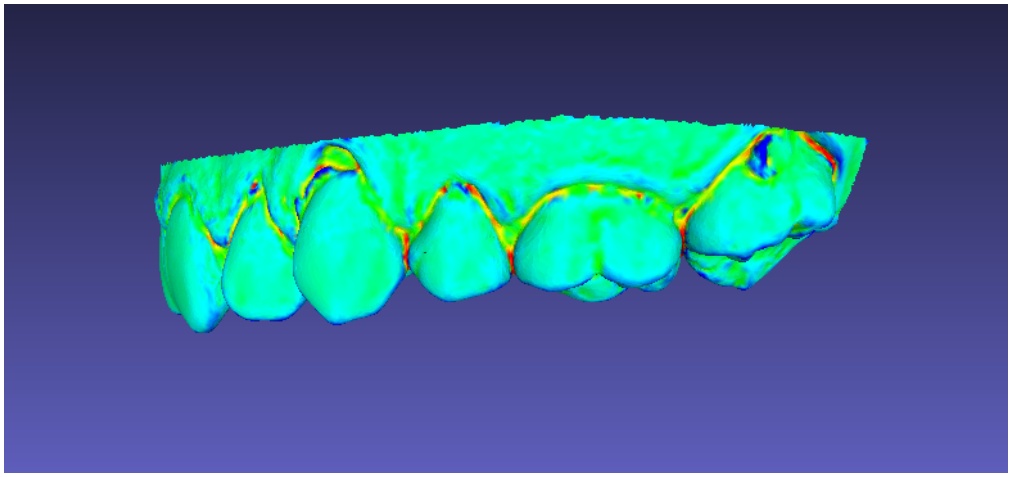


Figura 3.7: Vista de mapa de calor de la curvatura de los dientes posteriores. Máxima curvatura en rojo. Mínima en azul.

Debido a que la publicación es del año 2004, sus algoritmos están implementados ocupando imágenes, para evitar el costo de manejar archivos 3D. El alumno, con ayuda de la literatura[11], se encargó de migrar estos algoritmos a tres dimensiones.

El sistema adaptado de la publicación, estima una curva que se ajusta al arco de los dientes, apreciable en la figura 3.8, y sobre esta se definen múltiples posibles intersticios dentales (figura 3.12) para posteriormente ir verificando uno a uno cuales cumplen las condiciones anteriormente explicadas, de menor altura y mayor curvatura.

Una de las limitantes de este algoritmo, puesto que va midiendo la altura de los intersticios, es que el modelo tridimensional debe estar alineado con los ejes XYZ de la forma descrita

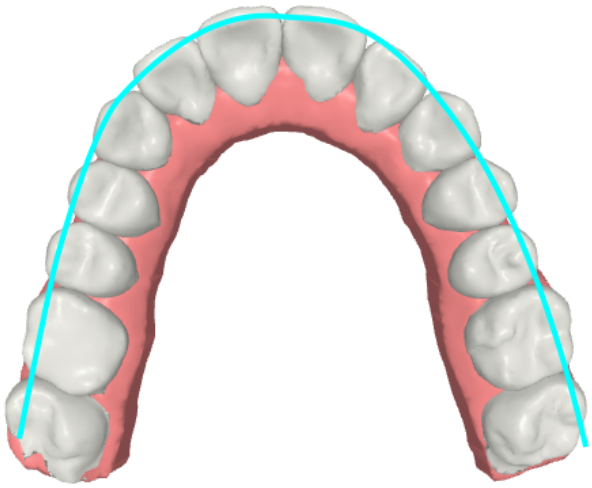
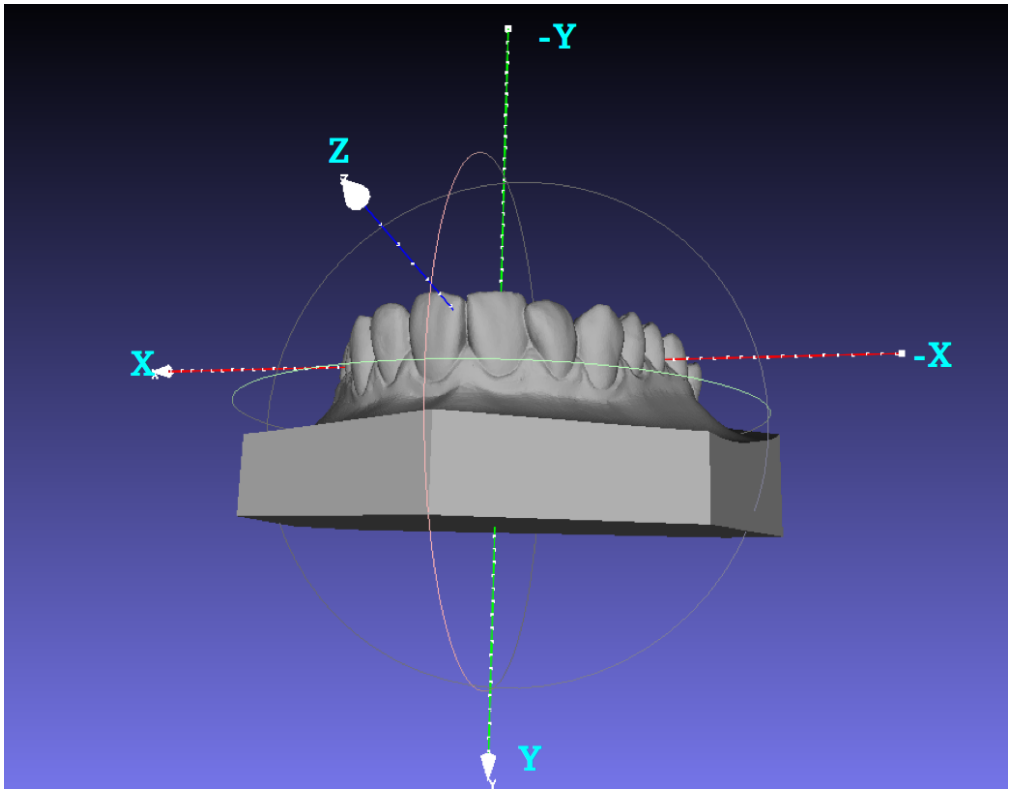


Figura 3.8: Curva del arco dental en celeste

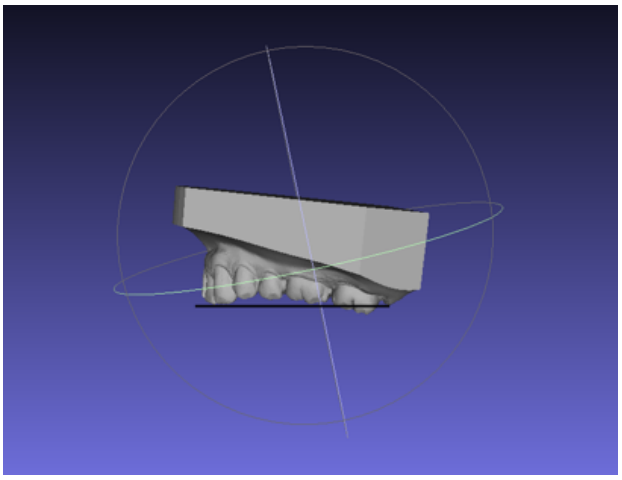
en la figura 3.9. El eje Y es mínimo en la punta de los dientes, y el eje Z es máximo en los incisivos y mínimo a la altura de los molares. Actualmente, esa alineación es realizada a mano. Por esto se propone un programa, previo al ingreso al programa de detección de dientes, que se encargue de que esta alineación esté emplazada correctamente.

Una vez modificada la entrada, el algoritmo sigue los siguientes pasos:

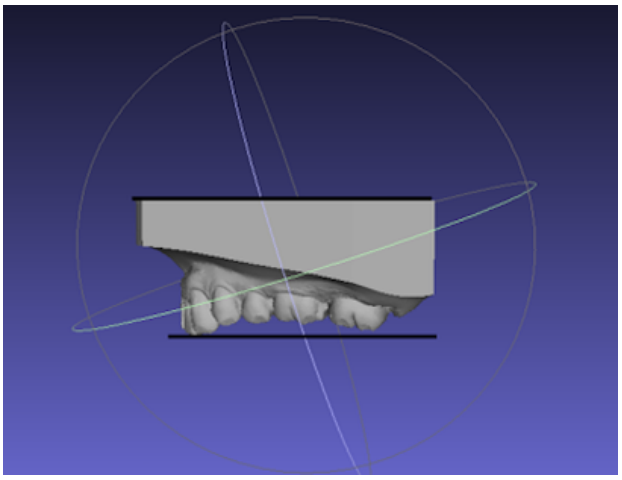
1. Detección de puntos claves mediante algoritmo de *watershed*[17]. La analogía sobre la cual se realiza este algoritmo es dejar una gota de agua en cada punto de la malla. Esta gota resbala hacia su punto vecino más bajo, hasta que ya no existe algún vecino más bajo. Así se obtienen mínimos locales, que resultan ser los puntos claves de los dientes, cúspides en los caninos, premolares y molares y puntos en los bordes incisales en los incisivos y laterales, figura 3.10. Estos puntos claves, permiten estimar la curva del arco.
2. Se ajusta una curva de orden cuatro a los puntos encontrados. Para esto se ocupa *Ceres Solver*[2], una librería de optimización y ajuste de curvas en C++. Figura 3.11
3. A lo largo de toda la curva se definen intersticios dentales hipotéticos, muy cercanos entre sí. Estos son planos que intersectan perpendicularmente a la curva, y se añaden seis inclinaciones para cada uno, $\pm 10, \pm 20, \pm 30$. Figura 3.12.
4. Para cada una de las inclinaciones de un mismo plano, se detecta su menor punto en el eje Y, sea este pm_y. Entre más grande sea el valor de pm_y, más profunda será la zona sobre la cual se encuentra el plano, posiblemente marcando un intersticio (recordando que el eje Y es mínimo en la parte más alta de los dientes, y un intersticio dental es más bajo que la parte más alta de los dientes). En base a esto se selecciona como plano candidato entre todas las inclinaciones aquel que tenga un pm_y de valor más alto.
5. En conjunto con pm_y, se ocupa la curvatura de la zona que demarca el plano como un indicador de intersticio. El valor pm_y explicado anteriormente y el promedio de



(a) Alineación



(b) Alineación Plano Z-X Incorrecta



(c) Alineación Plano Z-X Correcta

Figura 3.9: Alineación del los Ejes

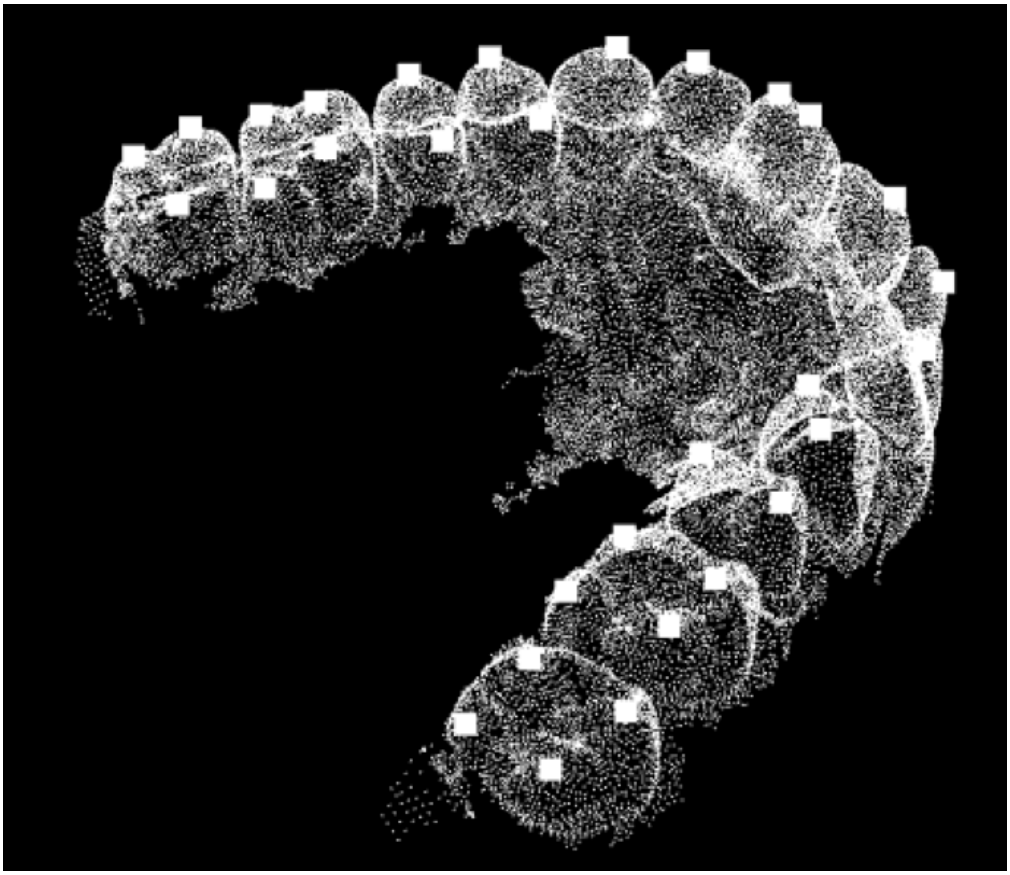


Figura 3.10: Puntos claves detectados con *Watershed*. Puntos claves representados por cuadrados blancos de mayor tamaño.

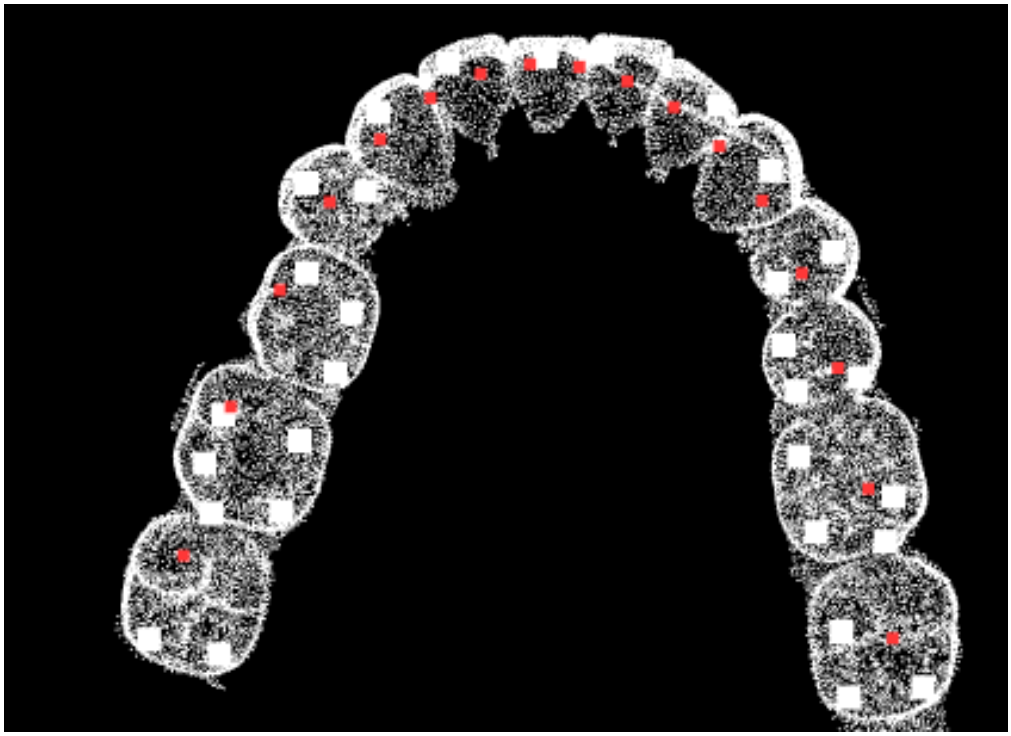


Figura 3.11: Curva estimada sobre puntos claves. Puntos claves representados por cuadrados blancos de mayor tamaño. Curva representada por cuadrados rojos.

la curvatura, son normalizados y promediados para generar un valor indicador, que a su vez es vuelto a normalizar. Es decir, este indicador es un valor representativo del nivel de curvatura y altura de los puntos de este intersticio hipotético. Estos valores se pueden observar en el gráfico 3.13a. Para eliminar el ruido del indicador, los valores son promediados con el indicador de sus dos planos vecinos, resultando en un valor suavizado, como se aprecia en el gráfico 3.13b. Para este nuevo valor suavizado, se obtiene sus máximos, y para cada máximo, si el valor es mayor a 0,45 es considerado como un intersticio.

6. Se generan polígonos no simples entre intersticios adyacentes. O sea, se conecta cada intersticio detectado con sus vecinos, dado que en el interior de este polígono se encuentra el diente (representado en figura 3.14. Se debe tomar una medida especial para generar los polígonos correctos, evitando la unión que se observa en la figura 3.15. Esta consiste de unir planos adyacentes únicamente con líneas que no crucen la curva del arco previamente calculada.

7. Con los polígonos calculados, se ocupa la técnica de *raycasting* para ver qué puntos se encuentran dentro de cada polígono. Esta técnica dibuja un rayo hacia el infinito (en la implementación es hacia un punto muy lejano) y detecta el número de intersecciones con las líneas que componen al polígono. Si el número de intersecciones es impar, se está dentro del polígono. Por el contrario, si es par, se encuentra afuera. Para evitar errores de aritmética de punto flotante, se implementan dos rayos hacia el infinito con distintas direcciones, y se verifica que la condición de intersecciones impares suceda en ambos para determinar que un punto esta dentro del polígono, esto mejora sustancialmente la detección, como se observa en la figura 3.16

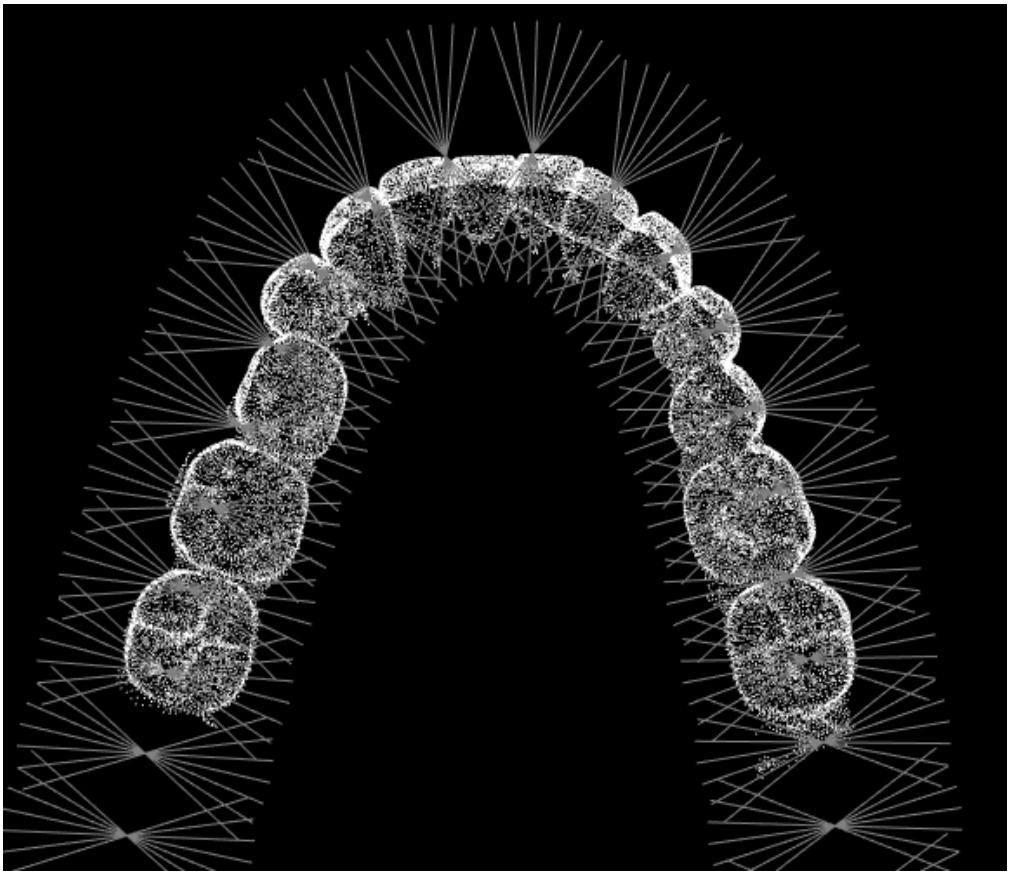
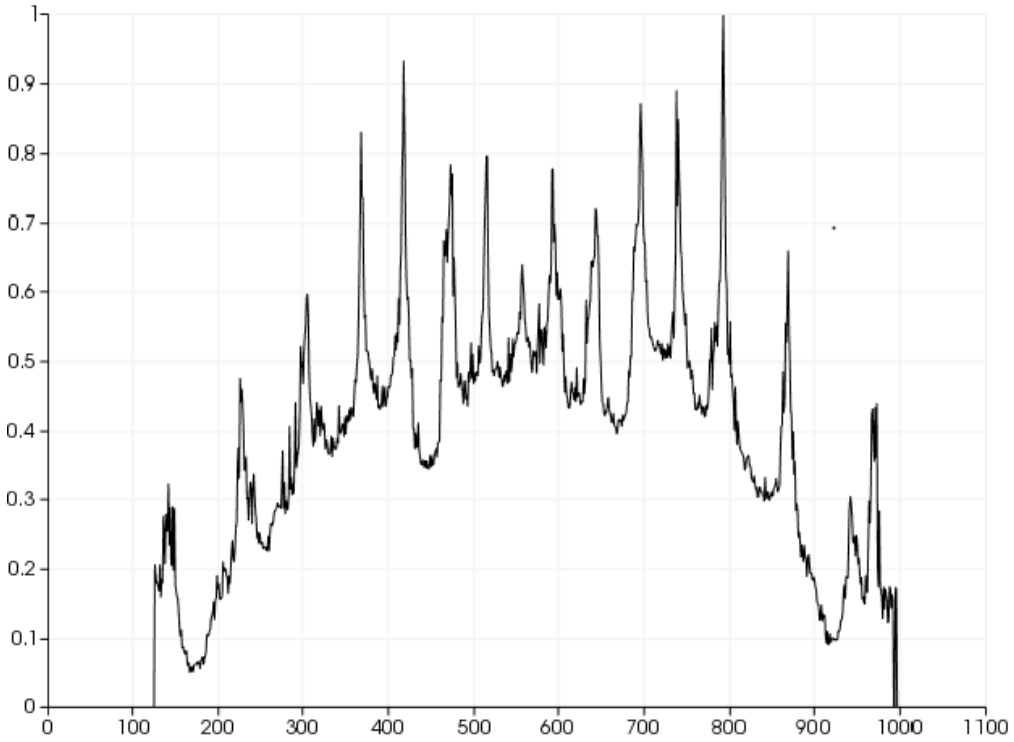
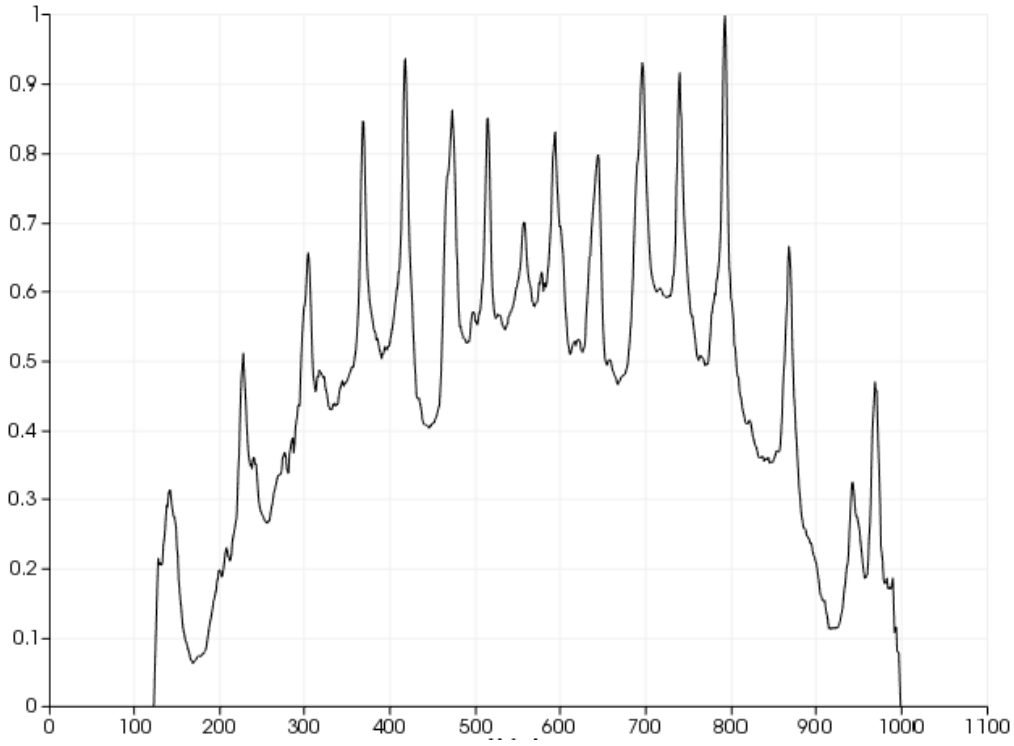


Figura 3.12: Intersticios hipotéticos (En la figura se han reducido el número de planos por claridad. En la implementación su densidad es 80 veces mayor)



(a) Gráfico del indicador dentado



(b) Gráfico del indicador suavizado promediando con valores vecinos

Figura 3.13: Gráficos del Indicador de Intersticio

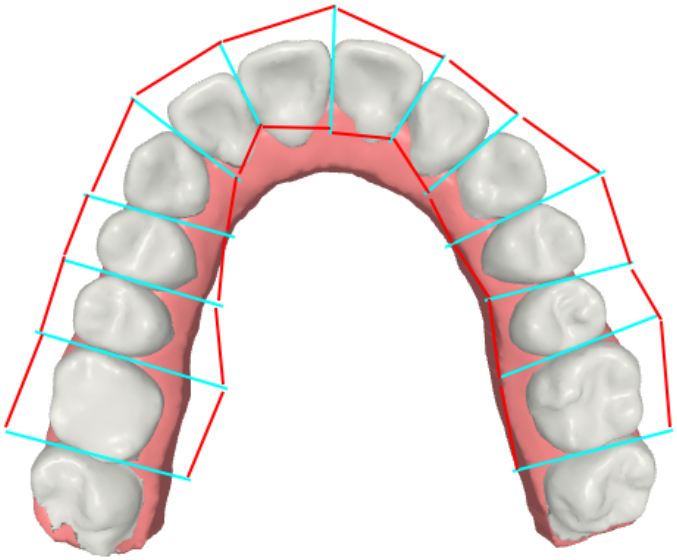


Figura 3.14: Intersticios (Azul) | Uniones (Rojo)

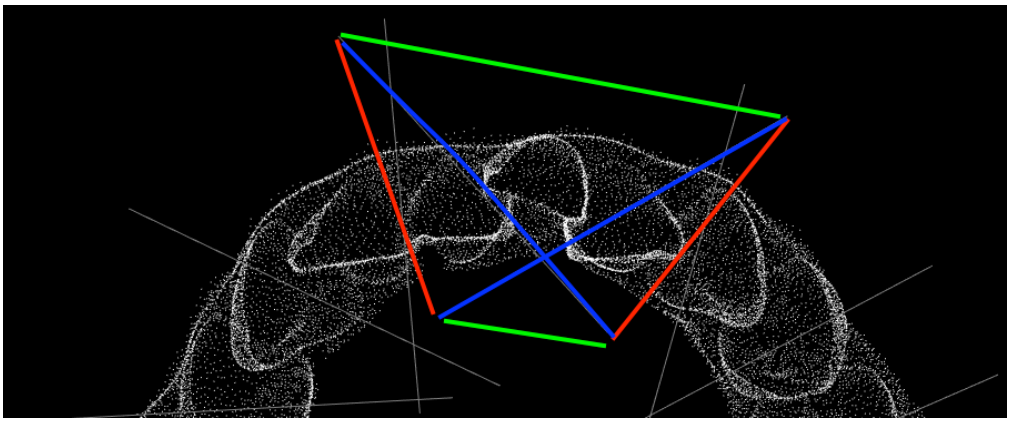
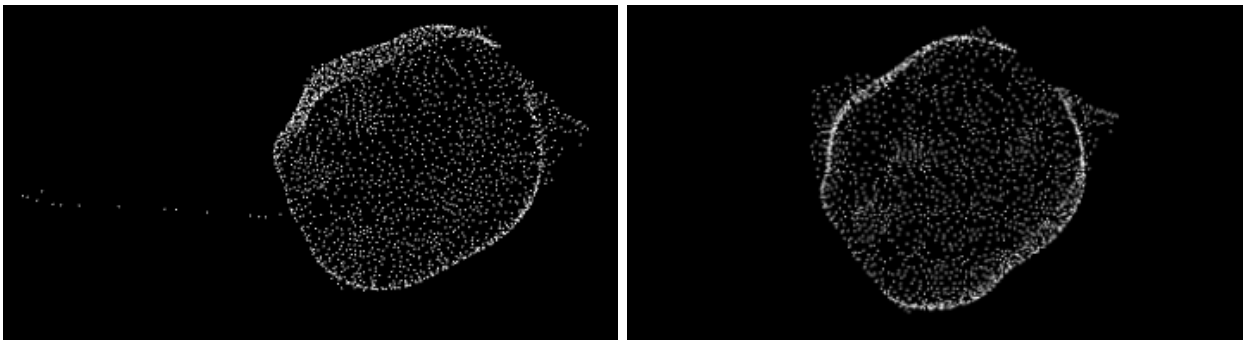


Figura 3.15: Uniones incorrectas (Rojo) | Uniones correctas (Verde) | Planos calculados (Azul)



(a) Segmentación incorrecta, 1 raycast (b) Segmentación correcta, 2 raycasts

Figura 3.16: Resultados segmentaciones con raycasts

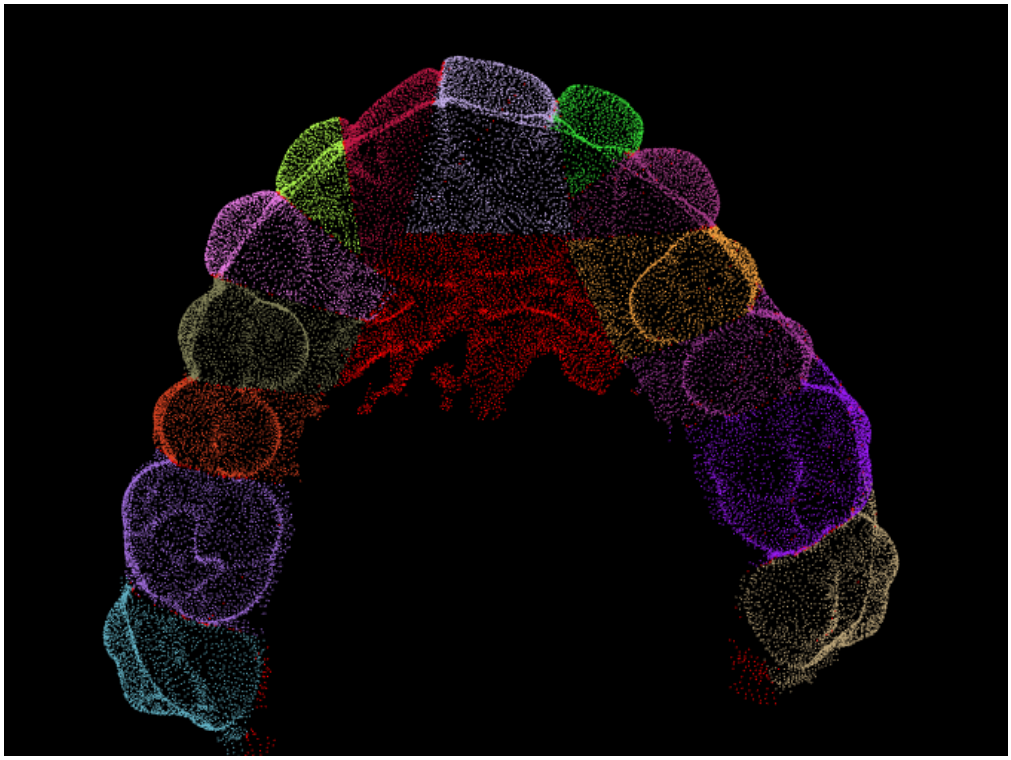


Figura 3.17: Resultado final segmentación por planos

El resultado final del algoritmo se puede observar en la figura 3.17. Es evidente, que es muy superior a los algoritmos previamente descritos, sin embargo, como por definición se buscaba los intersticios dentales y no la forma de los dientes en si mismos, no se logra detectar de manera correcta los contornos de los dientes con la encía.

3.5. Conclusiones

El aprendizaje obtenido de la implementación de estos algoritmos deficientes, dice relación con el uso de parámetros o constantes dentro de los mismos. Cada vez que se define una constante, como por ejemplo, los parámetros de *region growing*, hacen que el algoritmo siempre sea deficiente, puesto que se sobre o bajo ajustan a cada caso. Es decir, es posible realizar una segmentación casi perfecta con el algoritmo RANSAC-RG, sin embargo los parámetros tanto de los algoritmos de *RANSAC* y *region growing* funcionarían para un único caso, requiriendo que estos se cambien para otro modelo dental. En consecuencia, al ingresar parámetros al programa, definitivamente se requiere de algún usuario que pueda modificarlos caso a caso. El óptimo es encontrar algoritmos sin parámetros o que los parámetros dependan de la entrada. Si esto no es posible, la segunda mejor alternativa es tener una interfaz de uso sencillo para que un usuario pueda ajustarlos.

Capítulo 4

Implementación de Algoritmo de Segmentación mediante Campos Armónicos

En este capítulo se describe el algoritmo implementado que logró entregar la segmentación adecuada para poder realizar un tratamiento de ortodoncia. Se explica en detalle cada uno de los pasos que se siguen dentro de este. Finalmente se detalla su paralelización y formato de salida.

4.1. Descripción General

El algoritmo que cumple con los objetivos, surge como una mezcla entre lo planteado en "*Automatic Tooth Segmentation of Dental Mesh Based on Harmonic Fields*"[11], que se encuentra fuertemente basado en una publicación anterior de los mismos autores principales, "*Interactive tooth partition of dental mesh base on tooth-target harmonic field*"[13] y mejoras realizadas por el alumno en base a la experimentación. Como se detalla más adelante, estas mejoras surgen ya que en la experimentación los métodos planteados por las publicaciones no dan los resultados esperados y requieren pequeños ajustes.

La premisa de este sistema de segmentación, es que todo diente se encuentra rodeado de una zona cóncava. Esto sucede puesto que en la zona en la que la encía recubre el diente y en las intersecciones con otros dientes se producen grietas cóncavas, como se aprecia en la figura 4.1. Se probará en el capítulo de resultados experimentales que esto es cierto para un gran número de modelos dentales.

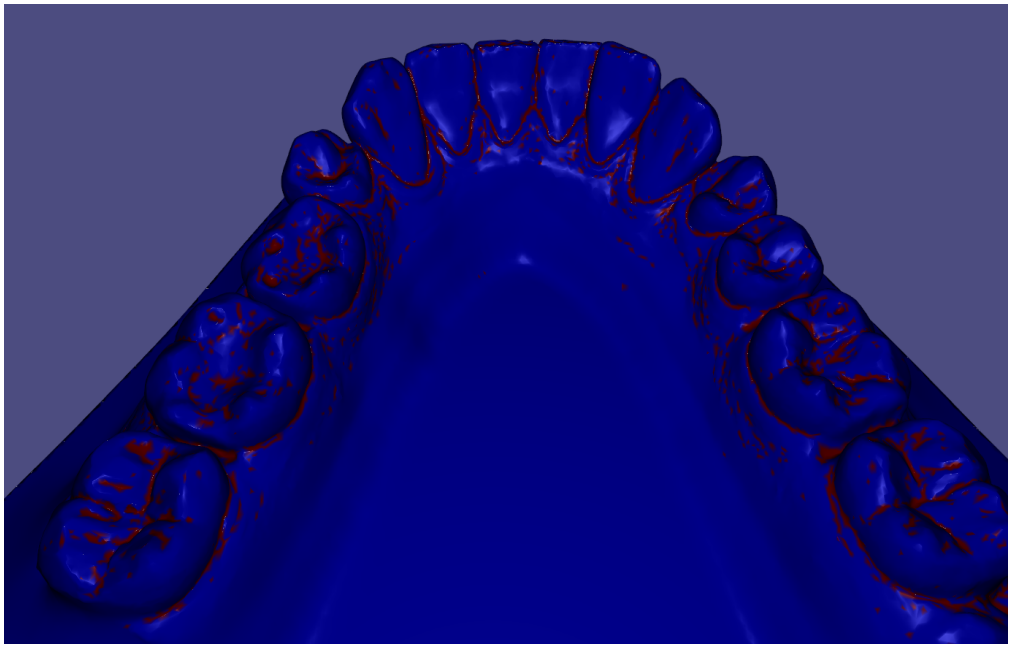


Figura 4.1: Grietas cóncavas en rojo

En este algoritmo de segmentación se utiliza el algoritmo de cortes planares anteriormente explicado, pero con una modificación, se segmentan únicamente los puntos claves detectados por *watershed*. Luego, estos se transforman en las condiciones de dirichlet de un campo armónico. La solución de este campo permite entregar un valor escalar como el que se observa en la figura 4.2. En esta figura se puede observar que hay cinco puntos sumamente rojos, estos son los puntos claves que han sido asignados la condición de borde 1, todos los otros puntos claves han sido asignados un valor 0.

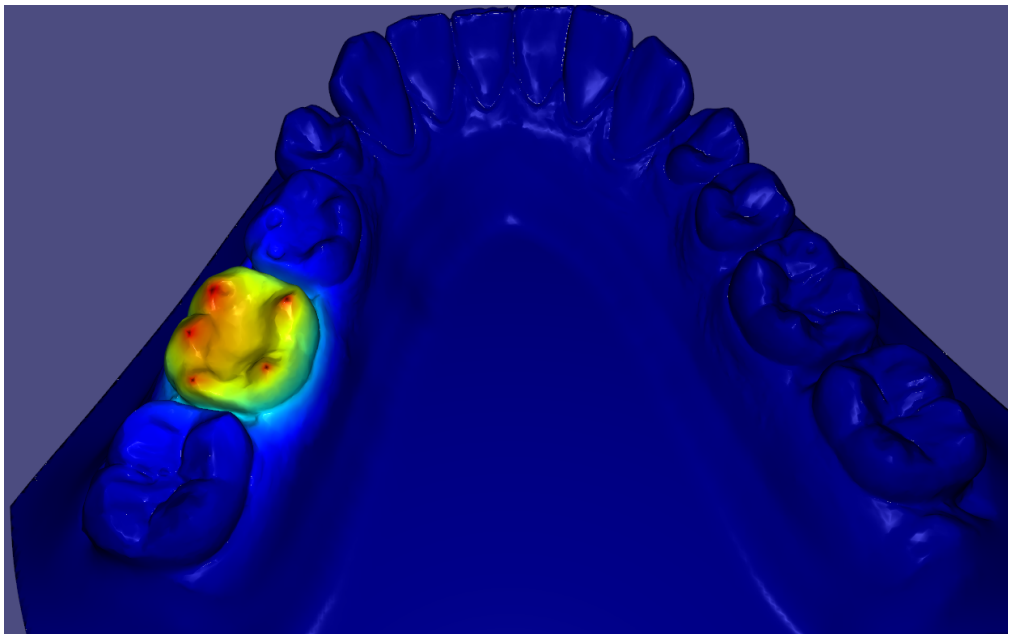


Figura 4.2: Mapa de calor de la solución del campo armónico. Rojo indica valor 1, azul valor 0

Una buena analogía para entender los campos armónicos es imaginar que el modelo tridimensional de la dentadura esta fabricado en papel u otro material absorbente. En cada punto clave que se asigna valor 1, se coloca una gota de colorante rojo. Por otro lado, a cada punto clave con valor 0 se le coloca una gota de colorante azul. Como el campo armónico modela un proceso de difusión, estas gotas de colorante se difundirán hasta mezclarse unas con otras resultando en un color morado intermedio, que en el caso de la visualización del software es verde-calipso. El resultado es el que se observa en la figura 4.2.

Posteriormente se muestrean isolíneas de este campo, como se observa en la figura 4.3. Finalmente se elige la mejor isolínea para realizar la segmentación, con criterios que se explicarán más adelante.

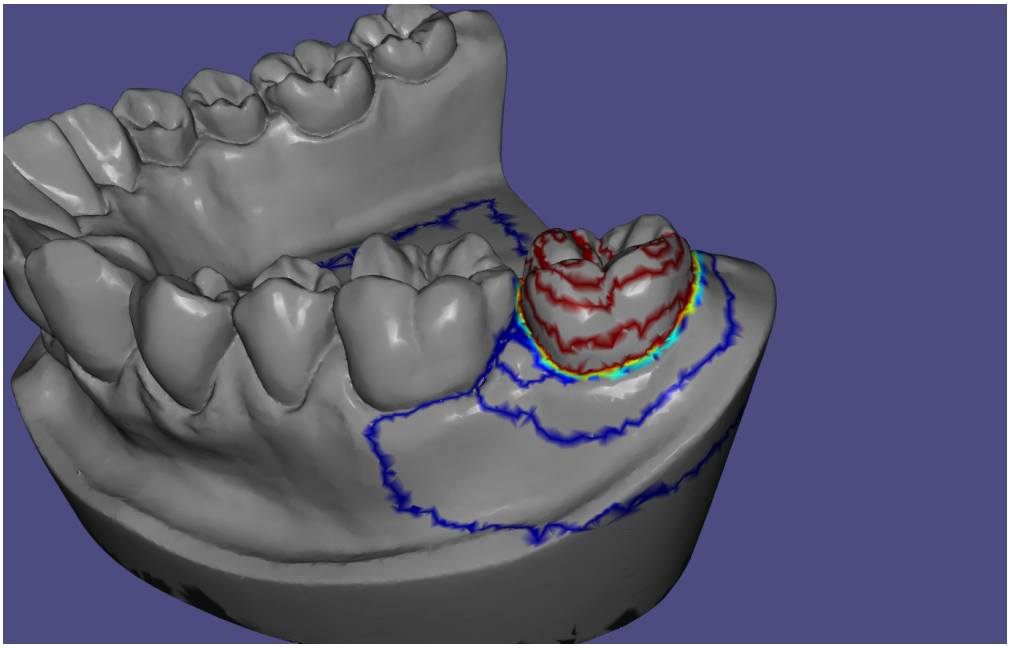


Figura 4.3: Isolíneas agrupadas en el borde del diente

En resumen, esta segmentación sigue los siguientes pasos:

1. **Cortes Planares:** Se utiliza el algoritmo de cortes planares explicado anteriormente con una ligera modificación. En vez de segmentar todo el diente en base al plano, se segmentan únicamente los puntos claves detectados por watershed. Además se calculan los puntos de la encía.
2. **Cálculo y Solución del Laplaciano del Campo Armónico Modificado:** Se calcula el laplaciano de un campo armónico, modificado en base a la concavidad, sobre el modelo tridimensional. Este es solucionado con condiciones de Dirichlet, obteniendo un campo escalar.
3. **Muestreo y Filtrado de Isolíneas Óptimas:** Se buscan los contornos o isolíneas del campo escalar, seleccionando las óptimas como bordes de cada diente.
4. **Corte en Base a Isolíneas:** Se corta cada uno de los dientes en base a su isolínea, y se resuelven ambigüedades.

Como las condiciones de Dirichlet usadas son los puntos claves obtenidos por los cortes planares, parte del proceso es utilizar este algoritmo previamente implementado. Con el objetivo de no sobrecargar los programas, en particular por las diferentes dependencias y el tiempo de compilación que toma unirlas, el programa de segmentación de dientes se divide en dos. Uno que realiza el paso 1, y otro que toma como entrada el resultado del primer programa y realiza los pasos 2, 3 y 4.

4.2. Cortes Planares

Tal como fue expuesto en la descripción de este algoritmo, su mayor falencia es que al realizar cortes basado en planos, se ignoran los contornos dentales con la encía. Como los planos son calculados a la altura de las puntas de los dientes, los errores de segmentación se agravan al ir descendiendo hacia la encía.

Por lo mismo, este algoritmo es bastante robusto segmentando las partes superiores de los dientes. Más aún, su precisión aumenta entre más cerca del centro del diente se esté. Es decir, es posible que el algoritmo se equivoque en la parte superior del diente estando cerca de su borde, pero es altamente improbable que detecte la parte central de un diente erróneamente.

En base a lo anterior, se desprende que realizando cortes planares se pueden segmentar correctamente los puntos claves (cúspides, puntas de caninos, y puntos bordes de incisivos) en base a los cuales se calcula la curva del arco. Estos puntos claves están tanto en la parte superior de los dientes, como hacia el centro de estos.

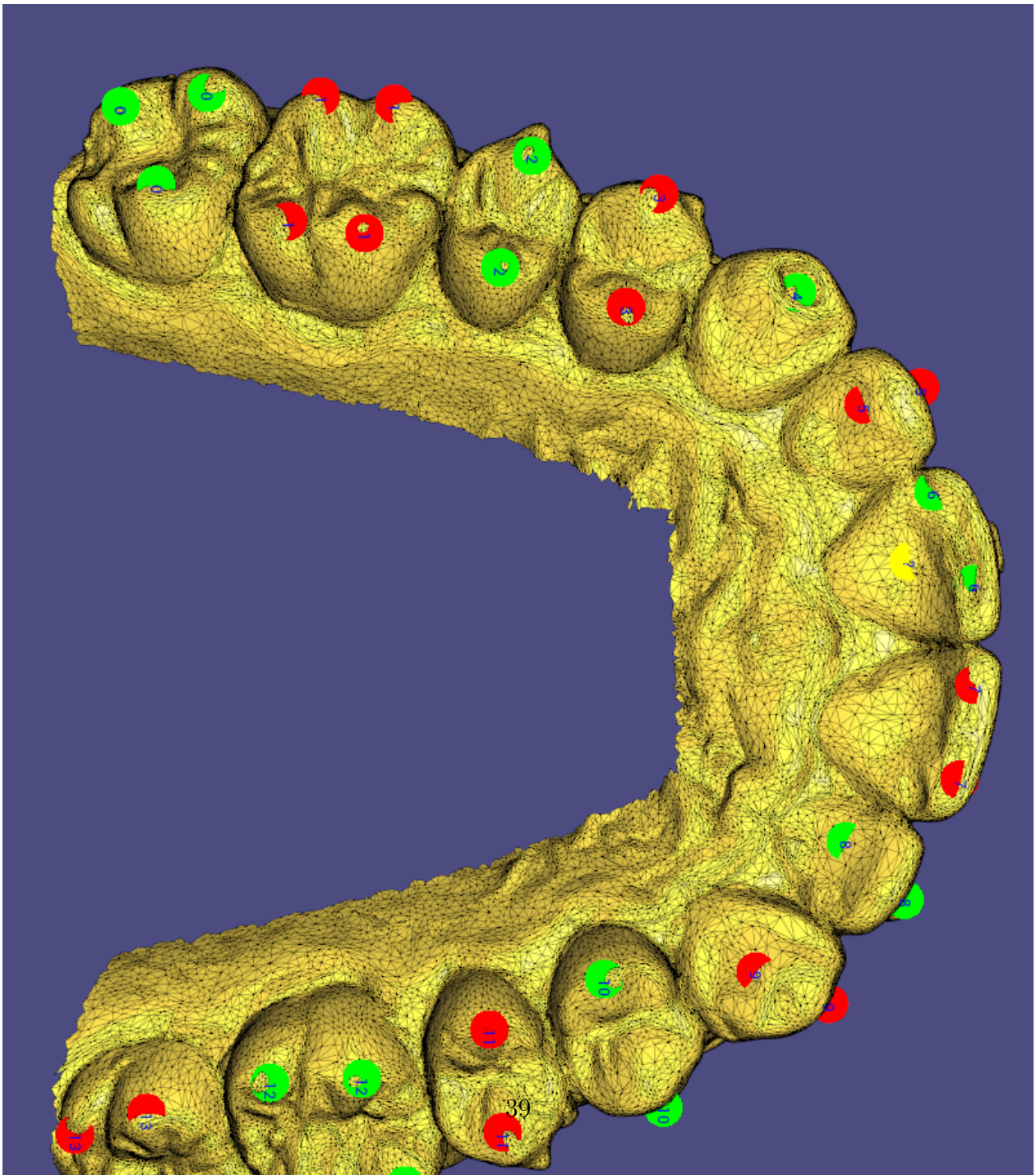
Por otro lado, se calculan los puntos de los cuales se tiene una alta confianza que no son parte de ningún diente, o sea, aquellos pertenecientes a la encía y base del modelo, acorde al mecanismo de *region growing* explicado en el capítulo anterior.

Finalmente el programa termina con un mapeo de puntos claves a dientes, donde los puntos claves encontrados entre el primer y segundo plano de intersticios en la curva corresponden al diente cero, entre el segundo y el tercero al diente uno, y así sucesivamente. Los puntos pertenecientes a la encía son mapeados a un valor -1. Este resultado se ejemplifica en la tabla 4.1 y en la figura 4.4

Los resultados de este mapeo son escritos a un archivo .csv para poder ser recibidos por el programa de cálculo armónico.

Diente	Vértice
0	5297
0	7863
0	14691
1	14729
⋮	⋮
13	35804
-1	0
⋮	⋮
-1	49979

Tabla 4.1: Ejemplo de Mapeo Vértices-Dientes



4.3. Cálculo y Solución del Laplaciano del Campo Armónico Modificado

4.3.1. Laplacianos Armónicos

Por definición un campo armónico ψ sobre una malla tridimensional, de vértices y arcos (V, A), cumple con:

$$\nabla\psi = 0$$

siendo ∇ el operador laplaciano.

Se utiliza un campo armónico de cotangentes modificado sobre la malla tridimensional. Los campos armónicos tradicionales de cotangentes ocupan una función de peso de la forma:

$$w_{\mathrm{i}j} = \cot\alpha + \cot\beta$$

Donde α y β son los ángulos opuestos al arco entre los puntos i y j.

Este peso es utilizado en la definición del laplaciano. El laplaciano es una matriz de tanto filas como columnas iguales al número de vértices en la malla, con valores para cada celda determinados por la siguiente ecuación:

$$L_{\mathrm{i}j} = \begin{cases} -w_{\mathrm{i}j} & (\mathrm{i}, j) \in A, \\ 0 & (\mathrm{i}, j) \notin A, \\ -\sum_{k \neq \mathrm{i}} L_{\mathrm{i}k} & \mathrm{i} = j, \end{cases}$$

Luego podemos redefinir $\nabla\psi = 0$ como:

$$L\psi = 0$$

4.3.2. Soluciones al Sistema y Condiciones de Borde

Al sistema $L\psi = 0$ se le pueden asignar condiciones de borde de Dirichlet, permitiendo resolver el sistema de ecuaciones matricial, obteniendo así ψ. Estas condiciones son obtenidas por los cortes planares. El planteamiento en la literatura asigna valor 0 a los puntos identificados como un diente impar, valor 1 a aquellos identificados como un diente par, y 0.5 a los puntos identificados como no pertenecientes a los dientes. Con esto el sistema es resuelto, y se obtiene un campo como se ve en la figura 4.5. En este trabajo, se utilizó el solver

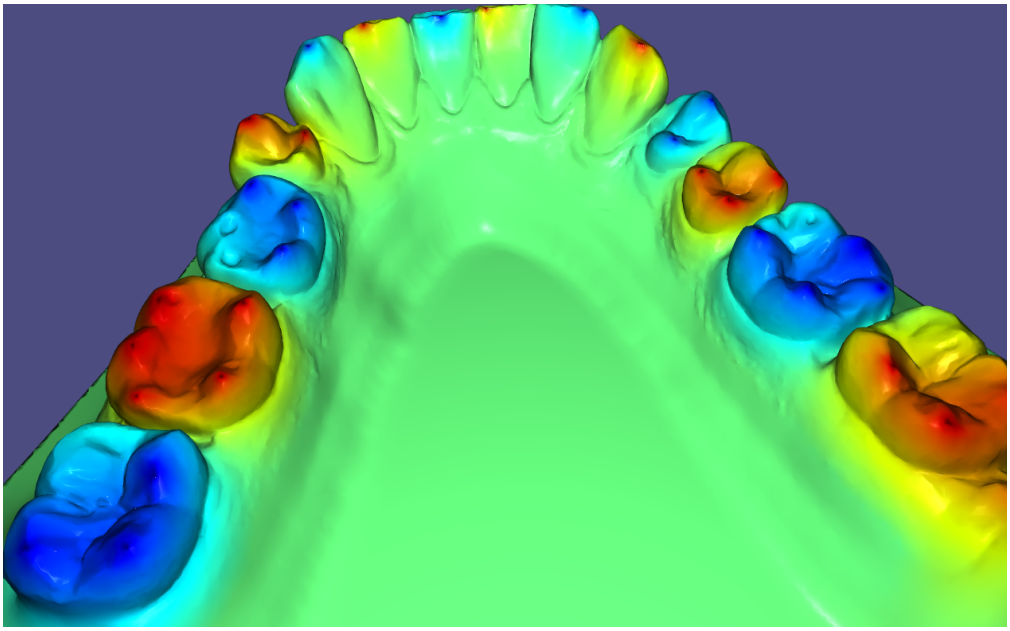


Figura 4.5: Mapa de calor del campo armónico global. Rojo indica valor 1, azul valor 0.

de matrices esparsas implementado por Eigen[7] `Eigen::SimplicialLDLT` que resuelve los sistemas utilizando factorizaciones de Cholesky.

En la literatura se resuelve un único sistema con todos los dientes, en vez de resolver un sistema por diente. El problema de resolver un único sistema, es que se puede introducir ruido o fugas de campo desde un diente a otro, ya que se tienen condiciones de borde del mismo valor en diferentes dientes, aún considerando que están alejados y separados por puntos con condiciones de borde opuestas. El resolver un único campo también dificulta el paso de muestreo y filtrado de isolíneas óptimas, problema ahondado en la sección del mismo nombre.

Por otra parte, se encontró en *libigl*[12] un mecanismo para resolver sistemas de ecuaciones, y mezclado con las capacidades de la librería *Eigen* se puede resolver los múltiples sistemas en tiempo similar al de resolver un solo sistema.

Resolver los sistemas rápidamente es posible porque los vértices de condiciones de borde se mantienen constantes, el cambio es únicamente su valor. Por esto, utilizando el módulo de matrices esparsas de *Eigen*, se puede realizar una sola vez la descomposición de Sparse-Cholesky del sistema y resolver para distintos valores de borde.

Así se resuelve para cada uno de los dientes, un sistema en que los vértices pertenecientes a ese diente tienen condición de borde de valor 1, y todos los que no pertenecen valen 0.

Se produce de esta manera un campo distinto ψ_{i} para cada diente i como se aprecia en la figura 4.6.

En algunas de las figuras hacia adelante, se mostrará el campo único que se define en la literatura, esto es solamente debido a que permite dar una visión general de la dentición en vez de particular de un diente, como lo es la solución de múltiples ψ_{i} implementada por el alumno.

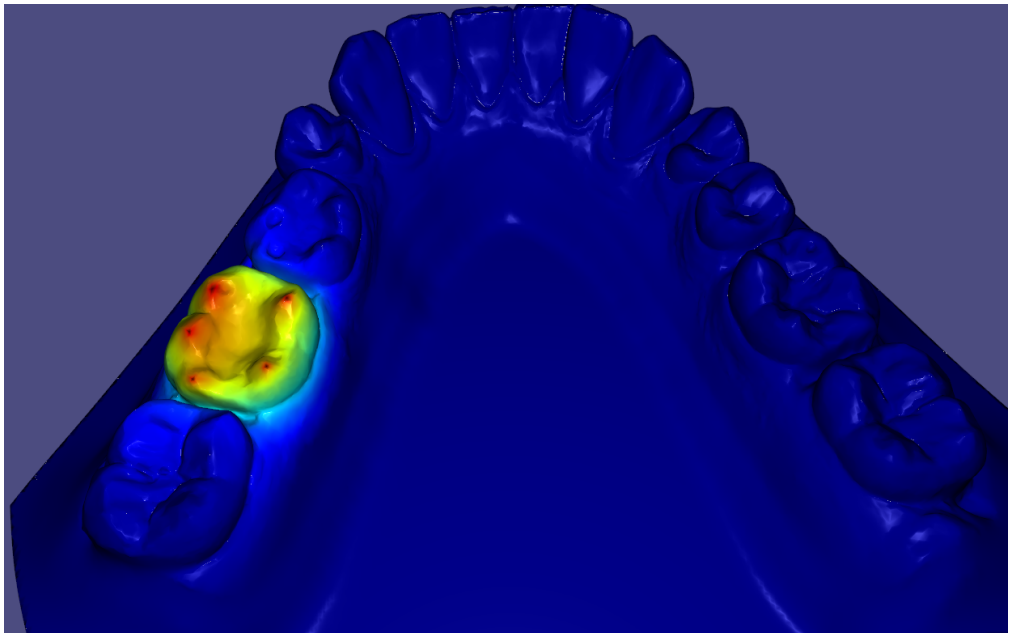


Figura 4.6: Mapa de calor del campo armónico por diente. Rojo indica valor 1, azul valor 0.

4.3.3. Modificación del Campo Armónico

Al ocupar el esquema de pesos de cotangentes, hay más difusión que la deseada en la malla. El objetivo es que los campos tengan un bajo costo para avanzar a través de espacio convexo, y por el contrario, que tengan dificultad para avanzar por los espacios cóncavos. Con este objetivo se aplica un nuevo esquema de pesos, según lo detallado en la publicación[11]:

$$w'_{ij} = \gamma_{ij} \cdot w_{ij}$$

Donde:

$$\gamma_{ij} = \begin{cases} 10^{-4} & \text{si i o } j \text{ cóncavos,} \\ 1 & \text{si no} \end{cases}$$

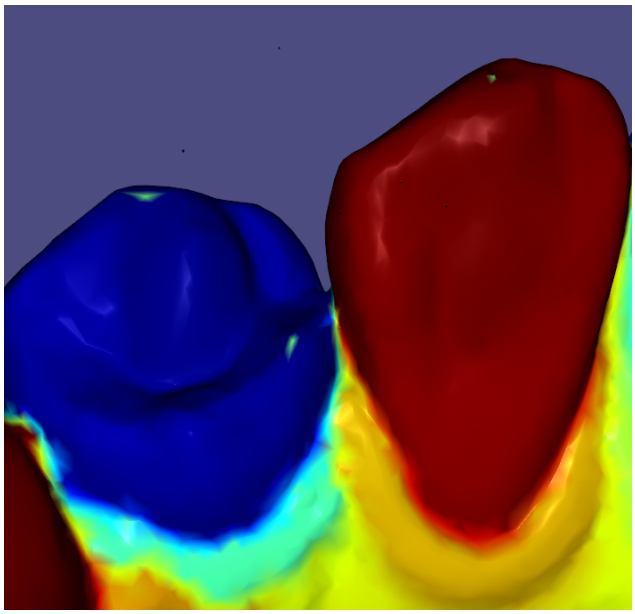
Sin embargo, en la implementación se descubrió que para obtener segmentaciones correctas, se debe ocupar un peso:

$$w''_{ij} = |w'_{ij}|$$

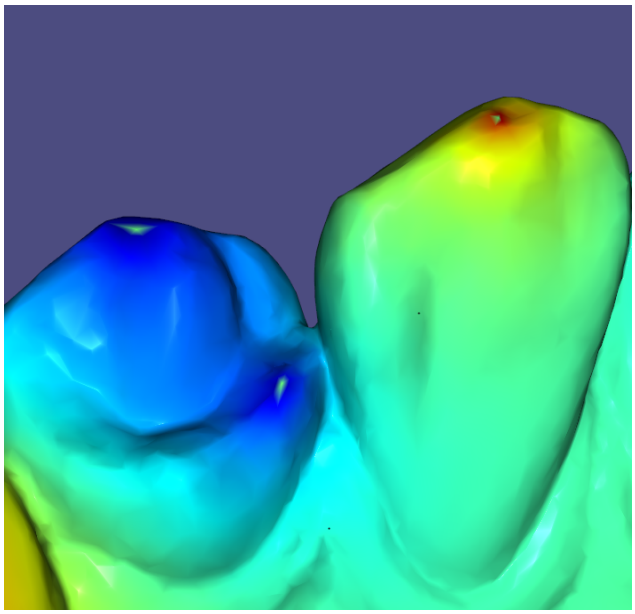
De no ser aplicada la función de valor absoluto el sistema de resolución de matrices de *Eigen* no logra resolver el sistema.

Con este nuevo esquema de pesos se obtiene una partición más precisa, como se ve en la

figura 4.7a, al poner resistencia al circular por los espacios cóncavos, el campo varia fuertemente justo en los bordes de los dientes,



(a) Campo armónico con pesos w''_{ij}. Alta resistencia para circular por áreas cóncavas, y baja resistencia en áreas no cóncavas



(b) Campo armónico con pesos w_{ij}. Resistencia uniforme para circular en el campo

Figura 4.7: Diferencias con cambios en pesos de campos armónicos

4.3.4. Concavidad

Se define en un trabajo anterior[13] que un punto i es cóncavo si:

$$(v_{avg} - v_{\mathrm{i}}) \bullet n_{\mathrm{i}} > \gamma$$

Donde v_{i} es la posición (x, y, z) del punto i, v_{avg} es la posición promedio de los puntos que tienen arcos que conectan con el punto i, n_{i} es la normal del punto i y γ es un valor constante pequeño, en la implementación 10^{-6}. El alumno descubrió en la implementación que se debe normalizar $v_{avg} - v_{\mathrm{i}}$ antes de realizar la operación de producto punto.

En está definición de concavidad se encontraron dos problemas. El primero es que existen espacios no cóncavos erróneamente detectados como tales, como se observa en la figura 4.8a. El segundo problema es que existen áreas cóncavas, demarcadas en la figura 4.8b, en particular en la zona lingual de los incisivos, en las cuales a pesar de sí ser un área cóncava, no es un área delimitante, impactando en el campo calculado como se aprecia en el campo de la figura 4.9.

Debido a esto, se realizó una segunda implementación de calculo de concavidad, definida en "*Mesh Segmentation with Concavity-aware Fields*"[1]. En esta publicación se determina que un punto i es cóncavo si algún vecino j cumple con:

$$\frac{(v_{\mathrm{i}} - v_j)}{||v_{\mathrm{i}} - v_j||} \bullet (n_j - n_{\mathrm{i}}) > \varsigma$$

Donde v_{i} y v_j son las posiciones de los puntos i y j, n_{i} y n_j son sus normales, y ς es una constante evaluada en 10^{-3}.

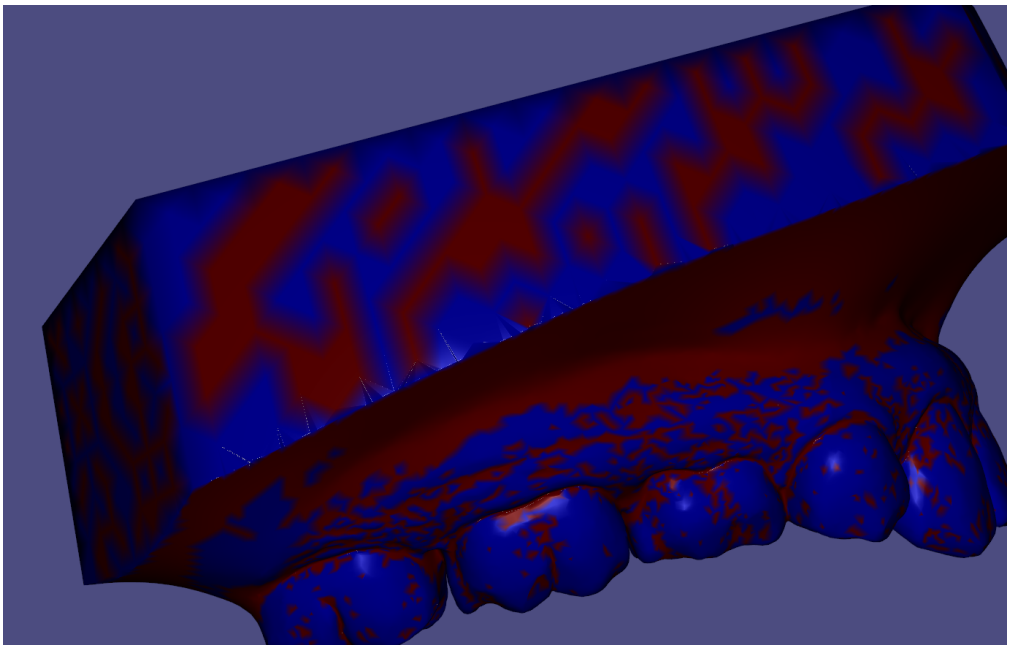
Como se puede apreciar en la figura 4.10, este campo soluciona el primer problema de detección de áreas cóncavas incorrectamente, sin embargo no soluciona el segundo problema de encontrar las áreas cóncavas delimitantes.

Para solucionar esto, el alumno realiza su propia implementación, en la cual define que un punto i es cóncavo, si cumple con:

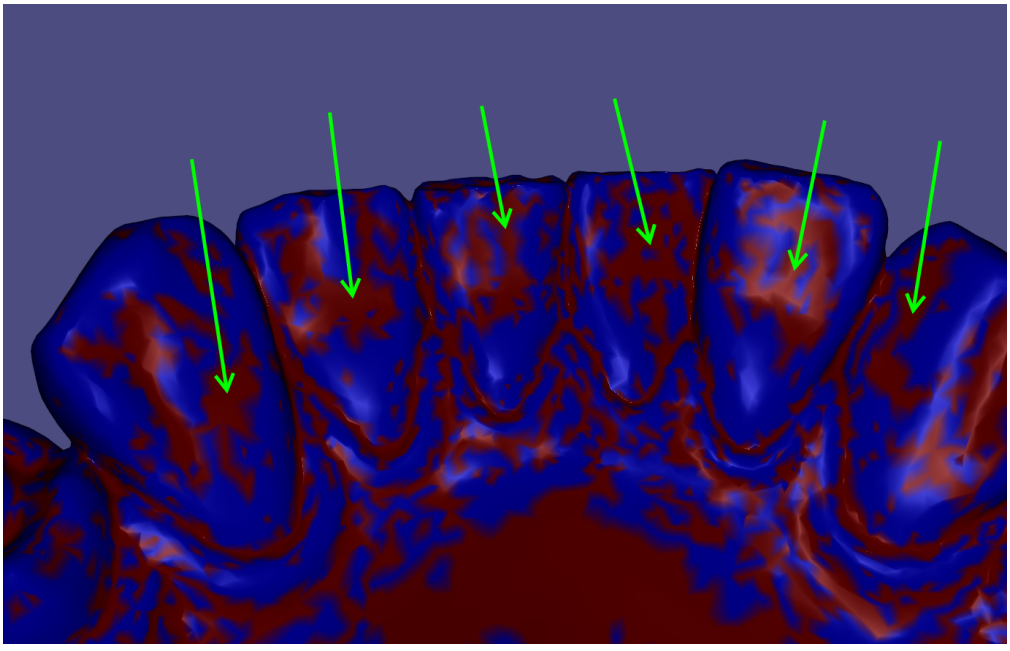
$$(v_{avg} - v_{\mathrm{i}}) \bullet n_{\mathrm{i}} > \gamma \wedge c_{\mathrm{i}} \geq \varkappa$$

Donde los primeros valores son equivalentes a la primera definición de concavidad, c_{i} es la curvatura del punto i y $\varkappa$ es una constante de valor 1. Este sistema soluciona ambos problemas, detectando únicamente las grietas cóncavas, como se observa en la figura 4.11. Con estos pesos se obtiene el campo armónico de la figura 4.12.

El objetivo de este esquema mixto, es ocupar tanto la curvatura como la concavidad. Luego, si un área es cóncava pero no tiene una gran curvatura, es improbable que sea una división dental.



(a) Detección de concavidad en áreas planas



(b) Áreas cóncavas indeseadas

Figura 4.8: Problemas en la concavidad - Rojo: Áreas Cóncavas, Azul: Áreas Convexas

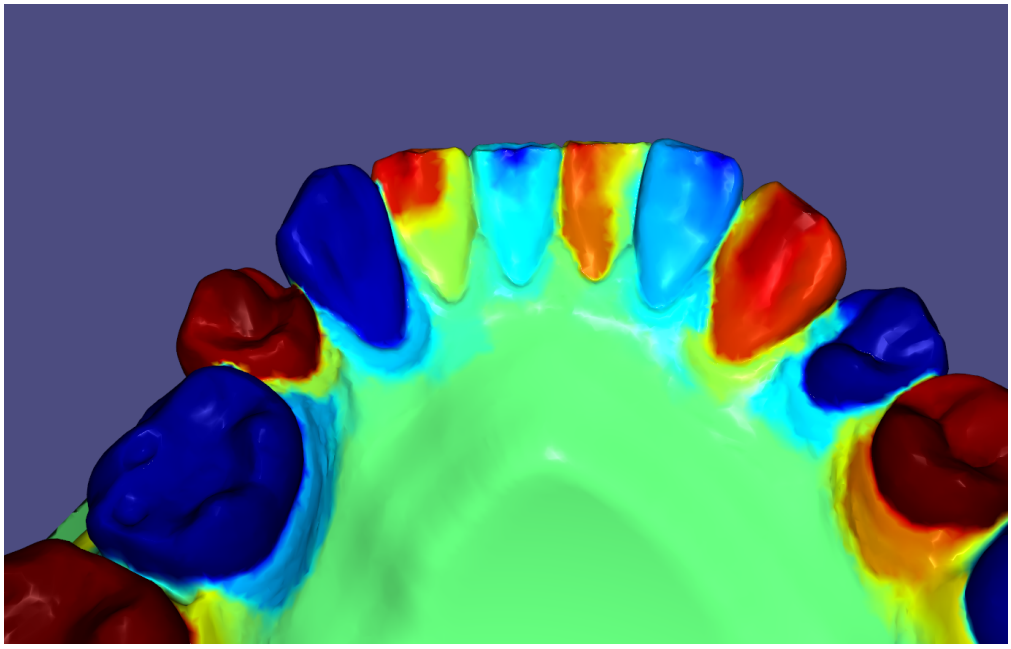
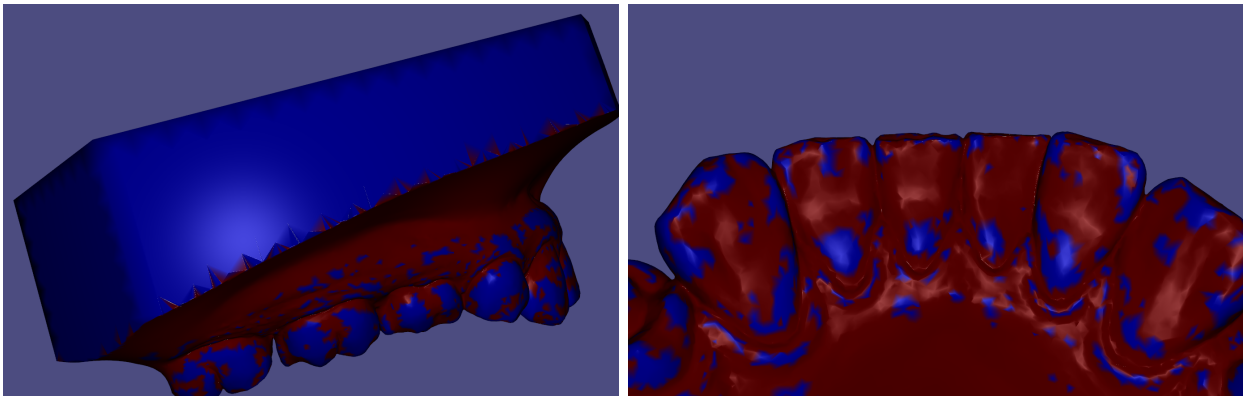


Figura 4.9: Errores en campo armónico por fallas en concavidad. Se aprecia que los dientes no están correctamente coloreados, con fugas y partes faltantes



(a) Detección correcta de áreas planas (b) Áreas cóncavas indeseadas se mantienen

Figura 4.10: Problemas en la concavidad - Rojo: Áreas Cóncavas, Azul: Áreas Convexas

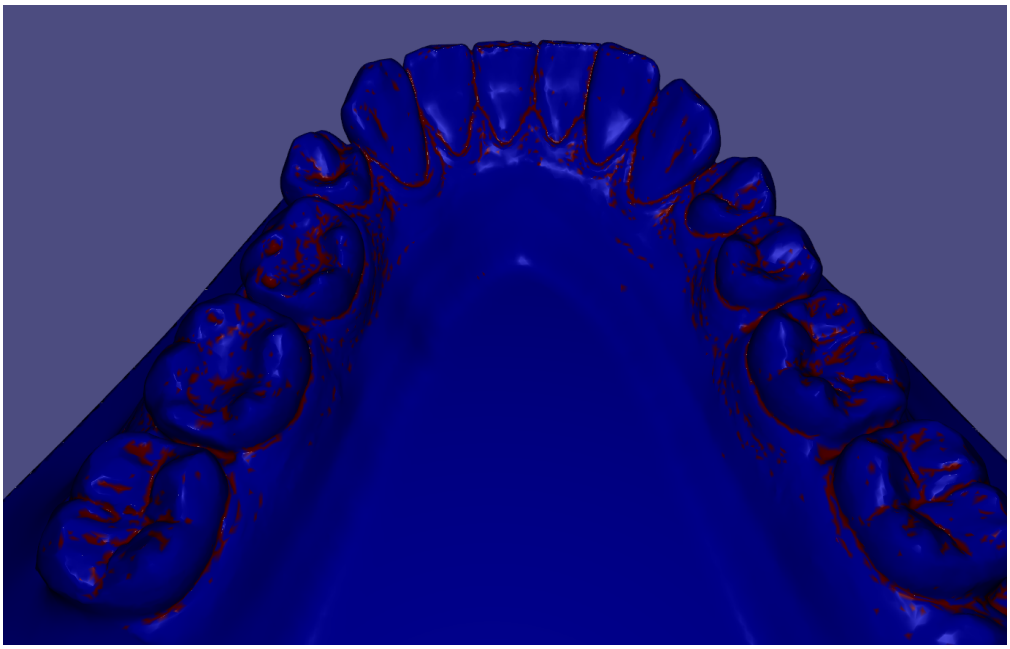


Figura 4.11: Grietas cóncavas detectadas con algoritmo original - Rojo: Áreas Cóncavas, Azul: Áreas Convexas

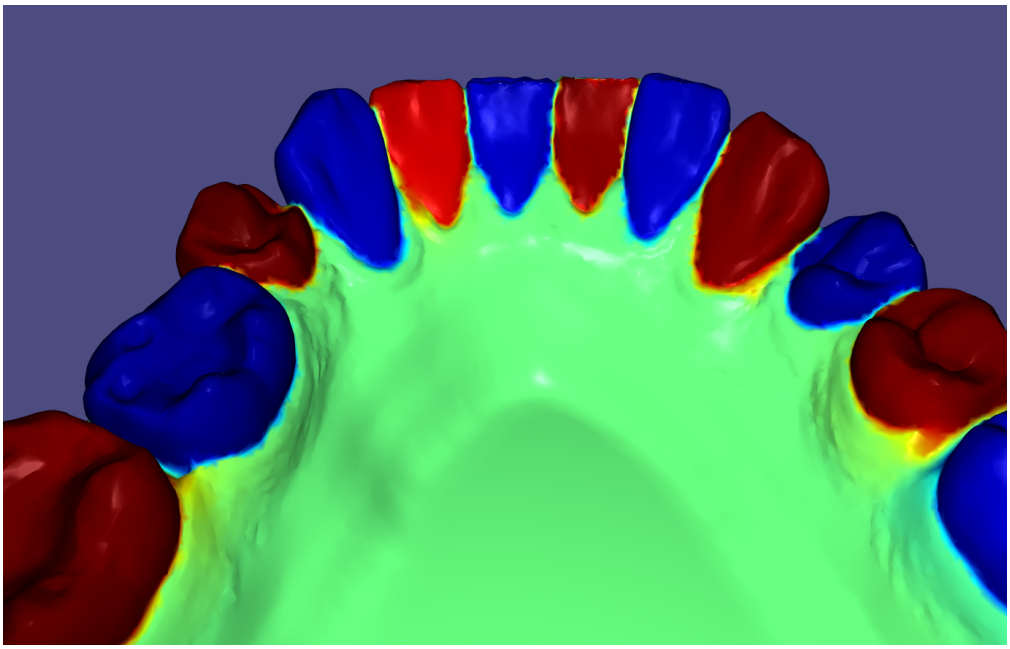


Figura 4.12: Campo armónico correcto

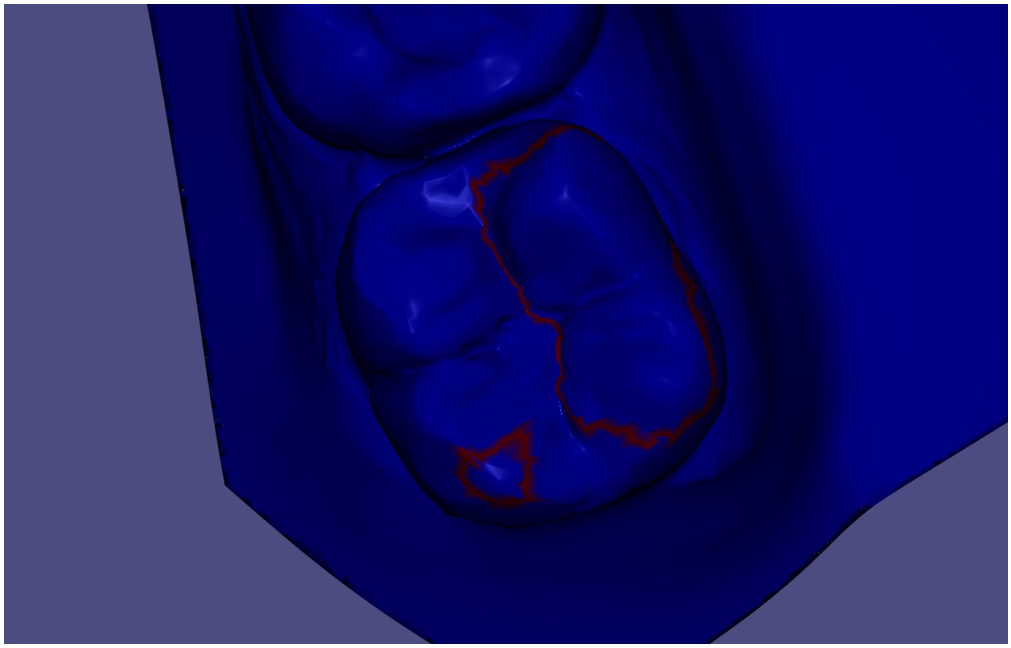


Figura 4.13: Múltiples isolíneas en rojo de valor 0.8 en campo armónico

4.4. Muestreo y Filtrado de Isolíneas Óptimas

4.4.1. Muestreo de Isolíneas

Una vez calculado el campo armónico, se buscan los contornos, o isolíneas de este como candidatos a un corte dental. Una isolínea es una línea circuncidante a la geometría sobre la que existe el campo, donde este tiene un valor constante.

Todas las publicaciones analizadas que realizan segmentación tanto de dientes como de otros objetos tridimensionales utilizando campos armónicos[11][13][1][24], calculan isolíneas para usar como zonas de corte. Sin embargo, ninguna explica ni cita algún algoritmo de búsqueda de contornos en mallas tridimensionales. Por esto, y después de una búsqueda infructuosa en la literatura, se diseña un algoritmo sencillo para muestrear isolíneas. A pesar de su simplicidad, a dado resultados aceptables en todos los experimentos.

Se define que un vértice i está en una isolínea de valor η si se cumple para algún vecino j:

$$\psi(\mathrm{i}) \leq \eta \wedge \psi(j) > \eta$$

Donde $\psi(x)$ es el valor del campo escalar para el punto x.

Al realizar esta selección de vértices se obtienen todos los contornos del campo. Sin embargo, estos no son necesariamente conexos, como se nota en la figura 4.13.

Para resolver esto, se separan todos los vértices encontrados en sus respectivas componen-

tes conexas, calculadas bajo un simple algoritmo de *breadth first search*.

Ocupando este sistema, para cada campo armónico se muestrean isolíneas para cincuenta valores, acorde a lo establecido en "*Dot Scissor: A Single-Click Interface for Mesh Segmentation*" [24]. Estos valores son espaciados igualmente entre el mínimo y el máximo de los valores del campo.

4.4.2. Filtrado de Isolíneas Óptimas

Para cada uno de los campos y de sus isolíneas muestreadas, se busca aquella que segmente de mejor manera el diente. Para esto se sigue un esquema de votación, similar a lo planteado en *Dot Scissor*[24]. En está publicación se establece que la isolínea que esté sobre el área más cóncava, será probablemente la isolínea de corte. Sin embargo, se encontró mayor robustez en el esquema de votación planteado por "*Mesh Decomposition with Cross-Boundary Brushes*"[23]. En la publicación se determina, que utilizando los campos escalares modificados anteriormente explicados, las isolíneas del campo tienden a agruparse cerca del área que se desea cortar, como se observa en la figura 4.3.

Luego las isolíneas más atractivas para realizar un corte, son aquellas que tienen una menor distancia con sus isolíneas vecinas. Para implementar esto cada una recibe un puntaje o *score*. Este *score*, en adelante *Siso*, se calcula siguiendo los siguientes pasos:

Primero se define un radio r_{i} para la isolínea i de largo l_{i} como:

$$r_{\mathrm{i}} = \frac{l_{\mathrm{i}}}{2\pi}$$

A pesar de ser lejanas a un círculo, esta definición de radio a dado buenos resultados experimentalmente. La métrica más simple para detectar la variación radio es:

$$\triangle_{\mathrm{i}} = 2r_{\mathrm{i}} - r_{\mathrm{i}-1} - r_{\mathrm{i}+1}$$

Donde $\triangle_{\mathrm{i}}$ es la variación del radio entre la isolínea i con sus vecinas.

Sin embargo esta medida se ve demasiado afectada por la localidad, por lo que se consideran una comparación con todas las isolíneas no sólo las directamente adyacentes, incluyendo los casos de borde. Para esto se define:

$$\triangle_{\mathrm{i}k} = \begin{cases} 2r_{\mathrm{i}} - r_{\mathrm{i}-k} - r_{\mathrm{i}+k} & \text{si i} - k > 1 \wedge \mathrm{i} + k \leq n, \\ 0 & \text{si no} \end{cases}$$

Donde n es el numero total de isolíneas, luego:

$$\triangle'_{\mathrm{i}} = \sum_{k=0}^{n} \triangle_{\mathrm{i}k}$$

El indicador $\triangle'_{\mathrm{i}}$ compara la variación de radio entre una isolínea y todas las demás. Por lo tanto entre más cerca del promedio este el radio de una isolínea, más cercano a cero será el valor del indicador. Sin embargo, este indicador tiene el problema de que elimina completamente la localidad, y es posible que existan varios sectores donde se agrupan isolíneas, como se observa en la figura 4.14. En consecuencia, pueden existir varios valores promedio de relevancia, por lo que se debe reintroducir la localidad para distinguir cual es el más correcto.

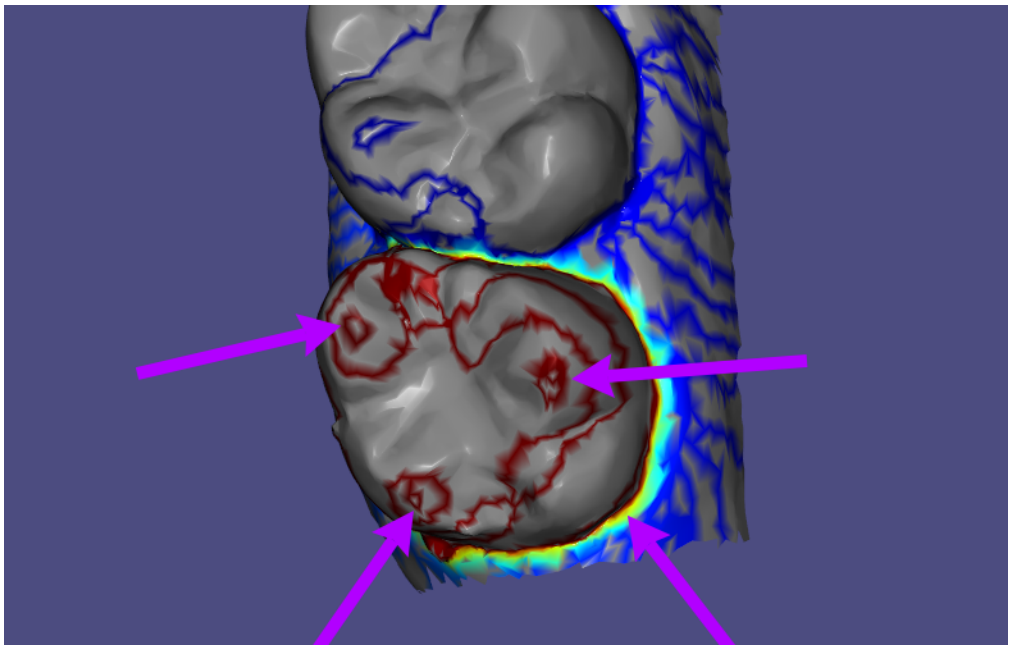


Figura 4.14: Múltiples agrupaciones de isolíneas

Por esto, se asignan pesos mayores a las isolíneas cercanas y menores a las lejanas realizando la convolución con la función gaussiana de la forma:

$$f(k) = \varepsilon^{-\frac{(k-1)^2}{2\sigma^2}}$$

Donde σ toma valor 2. Este producto resulta en un nuevo cálculo de variación de radio de la forma:

$$\triangle''_{\mathrm{i}} = \frac{\sum_{k=0}^{n} f(k) \triangle_{\mathrm{i}k}}{\sum_{k=0}^{n} f(k)}$$

Finalmente el *Siso* entregado a la isolínea i es:

$$Siso_{\mathrm{i}} = \frac{1}{1 + \triangle_{\mathrm{i}}''^2}$$

Nótese que $Siso_{\mathrm{i}}$ será más cercano a 1 mientras más pequeño sea $\triangle_{\mathrm{i}}''$ lo que sucede justamente cuando una isolínea tiene un radio similar al global, considerando con mayor ponderación a sus vecinos.

Por otra parte, en "*Mesh Decomposition with Cross-Boundary Brushes*", cada una de las caras también obtiene un *score*. Con este score cada cara emitirá su voto por cada isolínea que pasa por ella. Este *score* o número de votos, en adelante $Scara$, queda definido para cada cara i:

$$Scara_{\mathrm{i}} = \frac{\sum_{k=1}^{m} l_{\mathrm{i}k} \cdot Siso_k}{\sum_{k=1}^{m} l_{\mathrm{i}k}} \cdot g_{\mathrm{i}}$$

Donde m es el número de isolíneas que pasan por la cara i, $l_{\mathrm{i}k}$ es el largo de la isolínea en la cara i, y g_{i} es el valor del gradiente del campo armónico modificado. El gradiente del campo armónico modificado resulta ser un indicador de la concavidad de la cara, puesto que, como ya ha sido explicado, el campo varia más en las áreas de alta concavidad, viéndose esto reflejado en el gradiente.

Con $Scara$ calculado para todas las caras, estas emiten sus votos. Esto lo realizan sumando el valor de $Scara_{\mathrm{i}} \cdot l_{\mathrm{i}k}$ al total de votos de cada isolínea k que pasa a través de ellas. Realizado esto, simplemente se elige la isolínea que tenga más votos como aquella que segmenta correctamente el diente.

Este mecanismo de votación, hace particularmente complejo implementar un campo único. Como se observa en la figura 4.15, existen varias isolíneas posiblemente válidas para realizar un corte. Luego el proceso de filtrado no puede seleccionar a la mejor isolínea global, sino que se deben implementar medidas para seleccionar los mejores candidatos locales. Esto introduciría nuevos parámetros, como por ejemplo un radio estimado de diente, para encontrar los mejores candidatos dentro de aquel radio. Por el aprendizaje obtenido implementando los primeros algoritmos fallidos, se decide calcular siempre la mejor isolínea, evitando la introducción de un parámetro de localidad sensible.

Luego, se calculará para cada diente una solución de campo armónico con sus propias condiciones, como fue detallado anteriormente, donde cada punto perteneciente al diente tiene un valor 1 y cada punto externo a este tiene un valor 0. El campo de un único diente y su mejor isolínea resultante se puede observar en la figura 4.16.

4.5. Corte en Base a Isolíneas

Teniendo ya la isolínea correcta, el sistema de corte es relativamente sencillo. La isolínea se presenta como un contorno cerrado dentro de la malla geométrica. Esto divide a la malla

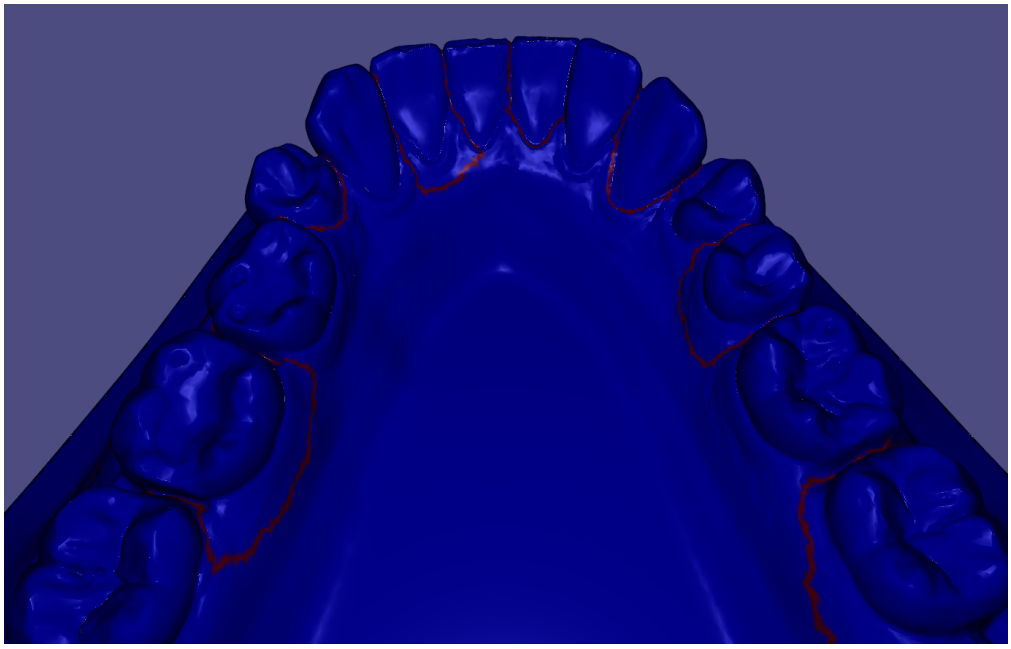
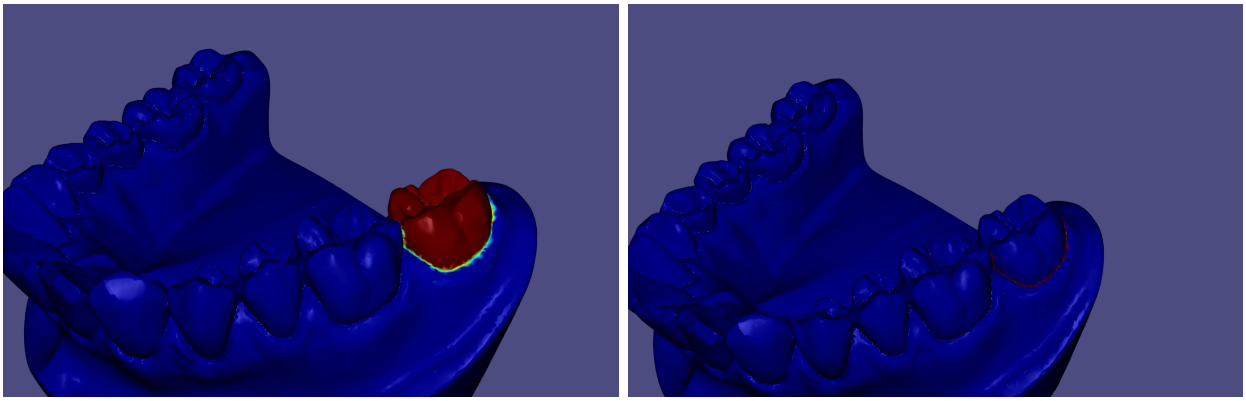


Figura 4.15: Múltiples contornos de valor 0.51 en campo armónico múltiple



(a) Campo único (b) Isolínea del Campo Único

Figura 4.16: Problemas en la concavidad - Segundo algoritmo

en dos, o sea, existen dos componentes conexas que no pueden conectarse sin cruzar la isolínea. Sabiendo esto, basta con demarcar todos los vértices pertenecientes a la isolínea como borrados o incruzables. Hecho esto, se toma un punto al azar p_0 que no pertenece a la isolínea, y se encuentra su componente conexa. Esta es una de las dos componentes descritas anteriormente, siendo la otra todos los puntos restantes de la malla, descartando la componente ya encontrada y los puntos de la isolínea. Se asume que la componente conexa que contiene el menor número de vértices corresponde al diente, determinando el resultado final de la segmentación. Este corte se realiza para cada campo, entregando la segmentación total de la malla.

4.6. Complejidad computacional y tiempos de ejecución

El sistema de segmentación consta de múltiples partes, luego el cálculo de su complejidad no es trivial. Es más, la complejidad teórica se hace aún más compleja puesto que la malla geométrica de entrada determinará cuanto se demorará el algoritmo no solo por el número de puntos, sino que por su forma anatómica. El número de intersticios dentales posibles ralentiza considerablemente el algoritmo. Modelos que sean más grandes en sus dimensiones X, Z o que tengan una forma de arco más larga involucrarán un mayor número de intersticios factibles.

En la etapa de preprocesamiento todos los modelos son reducidos a un total de 100.000 caras y 50.000 vértices. Esto se debe a que los modelos deben ser presentados al usuario luego de la segmentación y se ha encontrado experimentalmente que este número de vértices y caras mantienen una buena relación entre la calidad de los modelos y peso considerando tiempos de descarga. Luego la complejidad del algoritmo frente al número de puntos no es muy relevante ya que este es constante.

Lo que sí es relevante es el tiempo de ejecución. En realizar el algoritmo de cortes planares, para obtener los puntos claves que actúan como condiciones de borde en el campo armónico se obtuvo un promedio de tiempo de ejecución en todos los modelos de 25.42 segundos. El algoritmo de corte armónico tuvo un tiempo de ejecución promedio de 62.84 segundos (o 22.57 segundos considerando paralelización). Luego la segmentación total, al concatenar ambos programas tiene un tiempo promedio de 88,26 (o 47,99 segundos paralelizado). Todas las pruebas fueron realizadas en un computador con un procesador *Intel® Core™ i7-4790K*.

4.7. Paralelización

Debido a que cada campo de cada diente, y su isolínea correspondiente, es absolutamente independiente del otro, la paralelización de este algoritmo cae dentro de la categoría de lo *embarrassingly parallel*. En consecuencia, se optó por paralelizar esta parte del programa. Para hacerlo, el proceso de segmentación de cada diente corre en un *thread* distinto, obteniendo una optimización teórica con una relación 1 a 1 con el número de núcleos de la máquina en la que se esta corriendo. Es decir, si la máquina cuenta con sólo un núcleo, la optimización será nula o de 1x, por otro lado si la máquina cuenta con cuatro núcleos la optimización es de

4x. El *speedup* experimental con el procesador anteriormente descrito fue de 2,78 veces. No se alcanzó un *speedup* de cuatro veces por los tiempos de escritura y lectura de los archivos de input y output.

4.8. Salida

La salida del programa es únicamente un archivo .csv, manteniéndose fiel al objetivo de entregar un output interpretable por cualquier otro programa. Este archivo .csv contiene cada uno de los vértices de la malla donde se indica por cada vértice a que número de diente pertenece, o si pertenece a la encía es indicado con el valor -1.

Capítulo 5

Resultados Experimentales

5.1. Obtención de Archivos de Prueba

5.1.1. *Picza* y Modelos Públicos

En el inicio del trabajo se intentó obtener archivos tridimensionales, para poder realizar las primeras pruebas, de dos maneras. La primera fue la obtención de moldes de yeso de la dentadura, superior e inferior, del alumno, y de un molde de dentadura superior de otra persona. Todos tomados por un estudiante de odontología. Hecho esto, se recibió una capacitación en el FabLab de la Facultad de Ingeniería y Ciencias de la Universidad de Chile, para utilizar una máquina *Picza* de escaneo 3D. Posteriormente los modelos fueron escaneados usando la máquina, obteniendo archivos 3D .stl. La segunda manera de obtención, fue descargar moldes previamente escaneados por terceros con licencias de uso libre.

Ambos métodos de obtención presentaron grandes desafíos. El primero, la obtención de moldes de yeso para posterior escaneo con la máquina *Picza*, tiene la dificultad de que la máquina de escaneo *Picza* fue diseñada para escanear objetos que no tengan socavados profundos, o en caso de tenerlos, requiere que el usuario realice múltiples escaneos y los aúne con otra herramienta de software.

El costo en tiempo de realizar un escaneo en la máxima resolución de la máquina *Picza* es de aproximadamente sesenta minutos. Para obtener todos los socavados de un modelo de yeso se requirió realizar cinco escaneos, llegando a un total de cinco horas en tiempo de escaneo. Posteriormente cada escaneo requería ser alineado con la herramienta "Align" de Meshlab. Al obtener resultados deficientes múltiples veces, y al no tener certeza de que se conseguiría la precisión deseada se desistió con el método de la máquina *Picza*.

El segundo método tenía el beneficio de que los modelos ya estaban escaneados, se encontraban completos y sin imperfecciones con respecto a las mallas geométricas. Sin embargo presentaban el problema de ser únicamente tres modelos, y que se desconocía la precisión con la que fue realizada la toma de moldes y el escaneo.

Finalmente se exploró la alternativa de realizar un trabajo en conjunto con la facultad de odontología de la Universidad de Chile. Esta facultad contaba con numerosos modelos en yeso, pero el escáner tridimensional con el que trabajaban era marginalmente mejor al scanner *Picza*, y entablar una relación de trabajo fue difícil puesto que el uso de esta memoria sería con fines de lucro para la empresa TecDent.

Considerando las dificultades expuestas se decidió buscar en el mercado una forma de obtener moldes de manera independiente.

5.1.2. *3Shape Trios* y Centro Dental

Se exploró el mercado chileno en búsqueda de escaners tridimensionales diseñados específicamente para la odontología. Se encontró que aquel que mejor se acomodaba para realizar un escaneo rápido es el escaner intraoral *3Shape Trios*. En conjunto con escanear fácilmente moldes de yeso, permite escanear directamente dentro de la boca, evitando el paso por moldes de yeso y capturando con más detalle las estructuras de esta. En conjunto con esto, fue el escaner que más se ajustaba a las necesidades económicas de la empresa TecDent.

Figura 5.1: Máquina de escaneo *3Shape Trios*

Se recibió una capacitación por parte del reseller del aparato para realizar la toma de moldes, habilitando al alumno como tomador de moldes. Hecho esto, se empezó una alianza con un centro dental para realizar tomas de moldes en sus dependencias, con el objetivo de una vez segmentados los modelos con la metodología expuesta en este trabajo, realizar el

tratamiento de ortodoncia invisible para poder ejecutar pruebas clínicas. En este centro se tomaron 16 moldes de yeso, 8 superiores y 8 inferiores, a pacientes que aceptaron un consentimiento informado para aceptar ser parte de los primeros tratamientos de esta implementación de la tecnología y de ocupar sus modelos y fotografías para análisis y publicaciones. Los modelos en yeso fueron tomados por una asistente dental capacitada, y posteriormente fueron digitalizados con el *3Shape Trios*. Tres de estos ocho pacientes iniciaron su tratamiento a la fecha, por lo que se cuenta con sus escaneos intraorales también.

Luego en total se cuenta con 22 modelos. 6 obtenidos de fuentes públicas y 16 obtenidos por cuenta propia. Son particularmente interesantes los modelos obtenidos por cuenta propia, ya que pertenecen a personas interesadas en iniciar una ortodoncia, por lo que prueban al sistema en su caso de uso real.

5.2. Primeras Pruebas con Modelos Propios

Al pasar desde los modelos publicados con licencias de uso libre a los modelos internos producidos por el *3Shape Trios*, inmediatamente se encontraron tres problemas a abordar.

5.2.1. Problema 1 - Falta de Base de Yeso

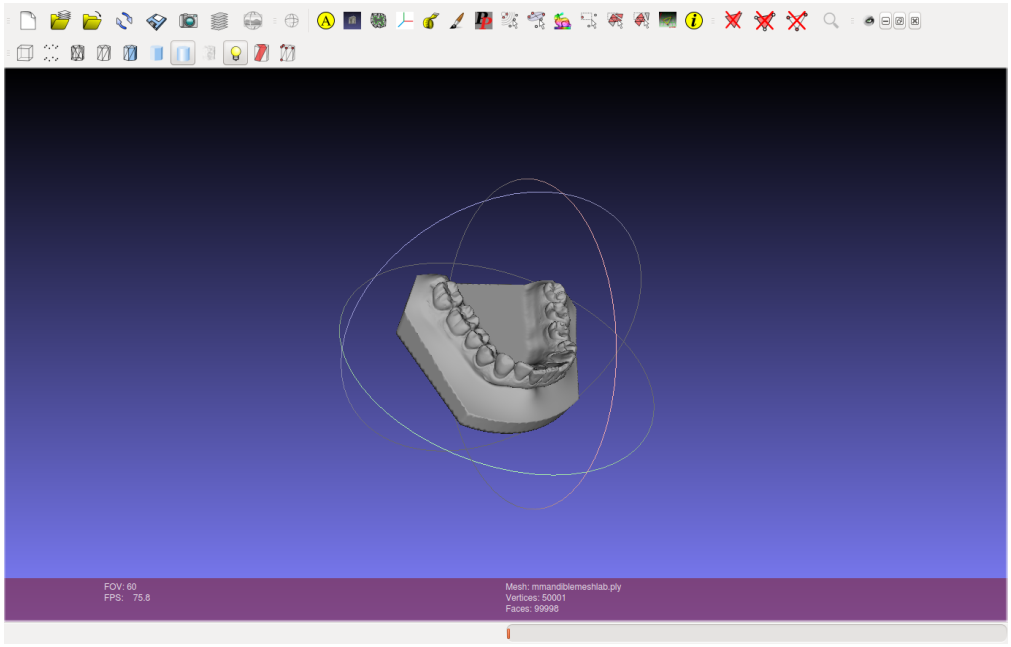
El primer problema es que los modelos obtenidos por internet cuentan con una base de yeso. Los modelos obtenidos por el *Trios* de manera intraoral no cuentan con esa base, como se observa en la figura 5.2.

Incluso si son escaneados a partir de yeso no se realiza un escaneo de la base del modelo, puesto que no es importante para el posterior tratamiento y consumiría más tiempo en escanear.

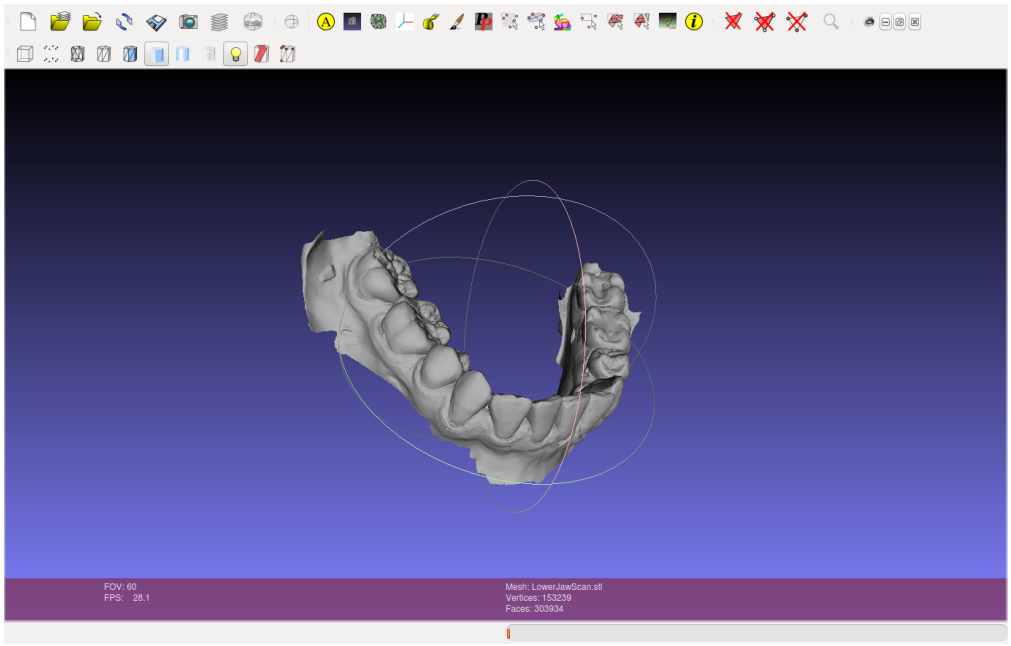
El primer paso dentro del programa era una limpieza de la base de yeso, como se aprecia en 5.3. La base era removida porque contenía más de la mitad de puntos del modelo y en consecuencia ralentiza considerablemente el programa de manera innecesaria. Además estos puntos limpiados, eran indicados por la salida del programa como la parte del modelo que no era diente. Esto causó que el programa de detección de cúspides empezara a fallar inmediatamente por el procedimiento de region growing que se realizaba para eliminar esa base de yeso. Algunos puntos que pertenecían a los dientes eran eliminados incorrectamente como se observa en la figura 5.4.

5.2.2. Problema 2 - Anatomías no Implementadas

El segundo problema consistió en anatomías detalladas por el paper de segmentación con campos armónicos, que no se encontraron en los modelos por internet, por lo que no se implementaron las soluciones que se describían para poder procesar modelos que las contengan.

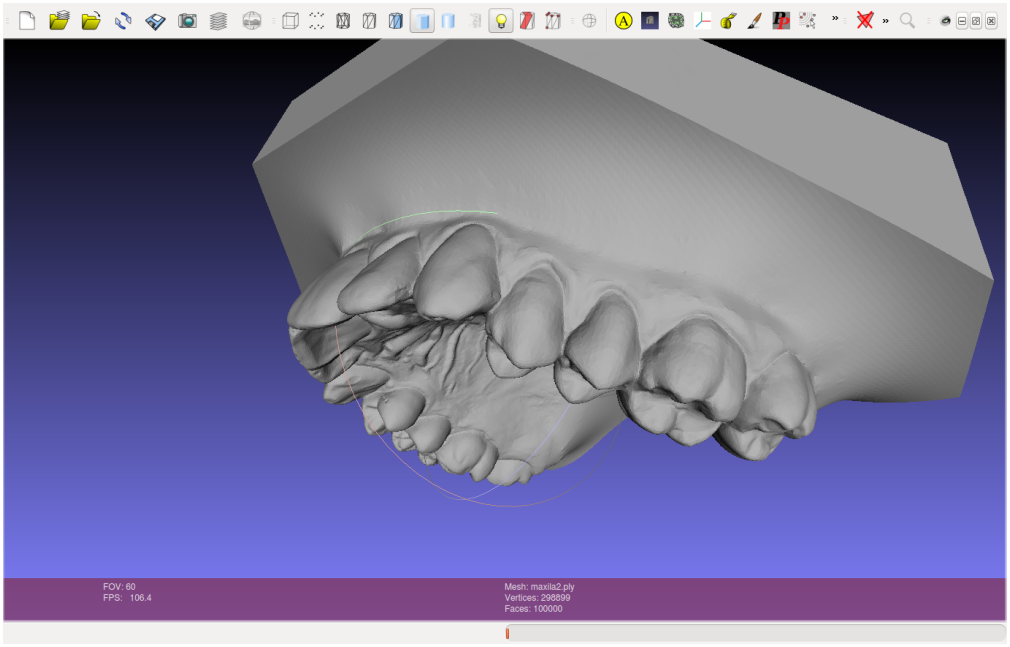


(a) Modelo publicado en internet

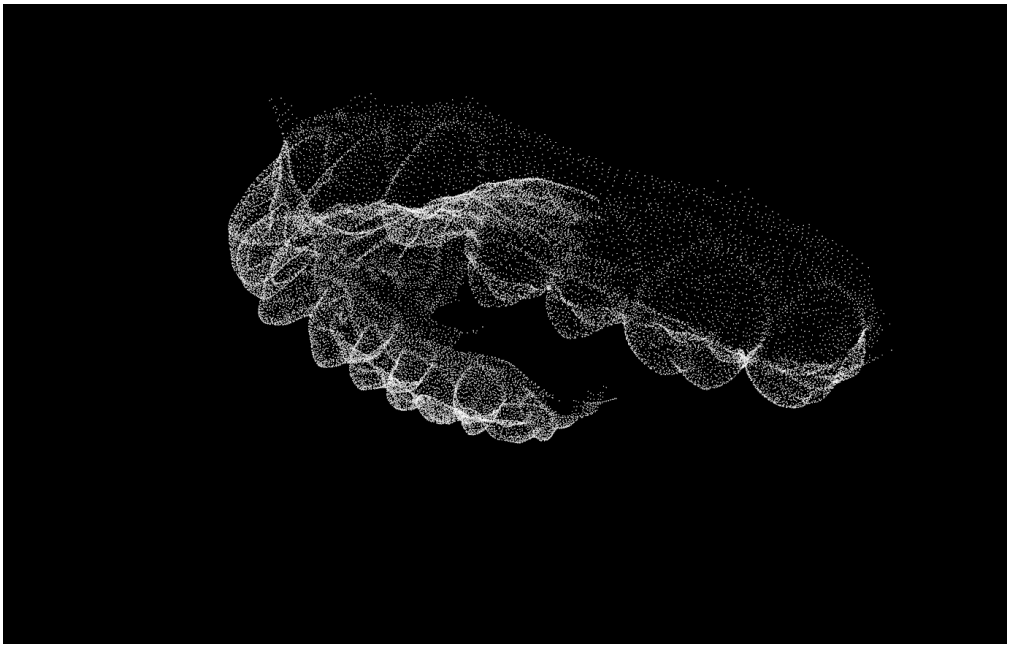


(b) Modelo de paciente escaneada con *3Shape Trios*

Figura 5.2: Diferencia de Base de Yeso

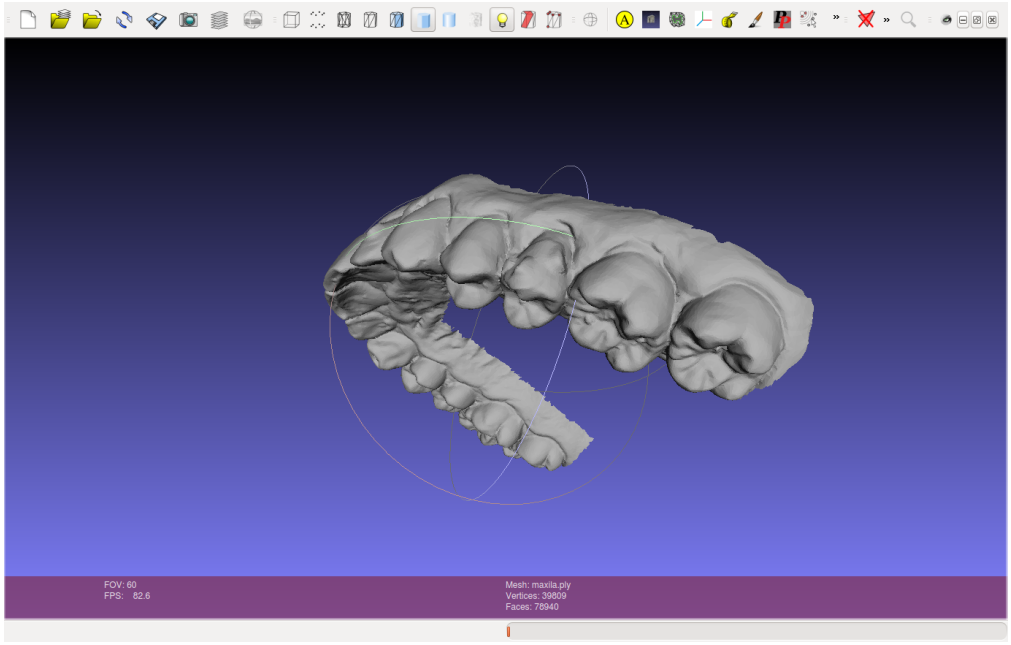


(a) Modelo Internet Original

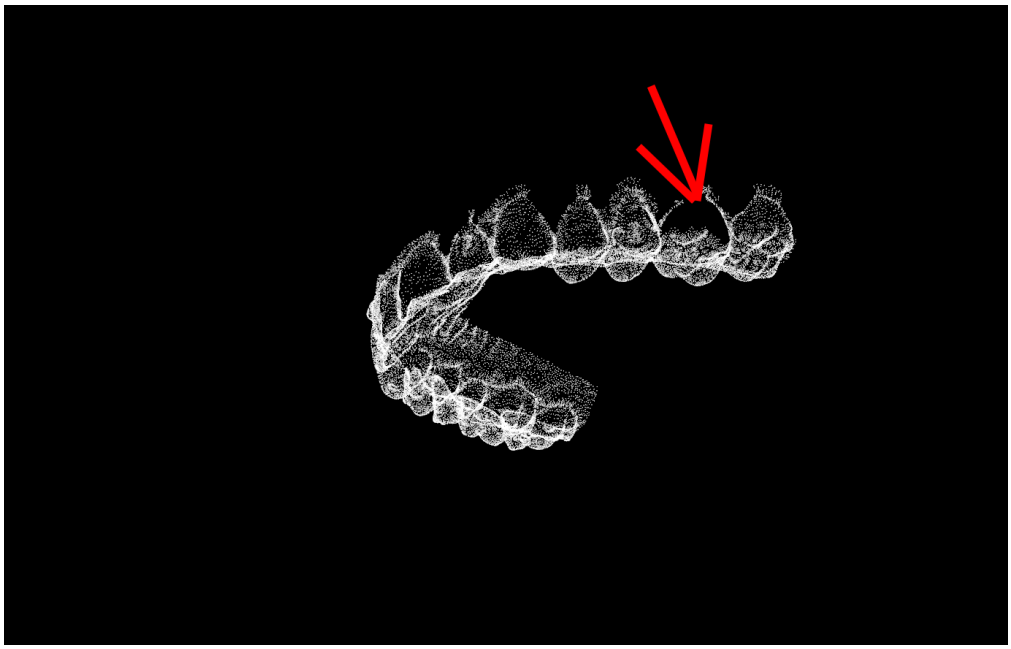


(b) Modelo Internet Después de Limpieza RG

Figura 5.3: Limpieza de la Base del Modelo



(a) Modelo del alumno original



(b) Modelo del alumno después de limpieza RG (Puntos incorrectos demaracados por flecha)

Figura 5.4: Puntos válidos limpiados incorrectamente

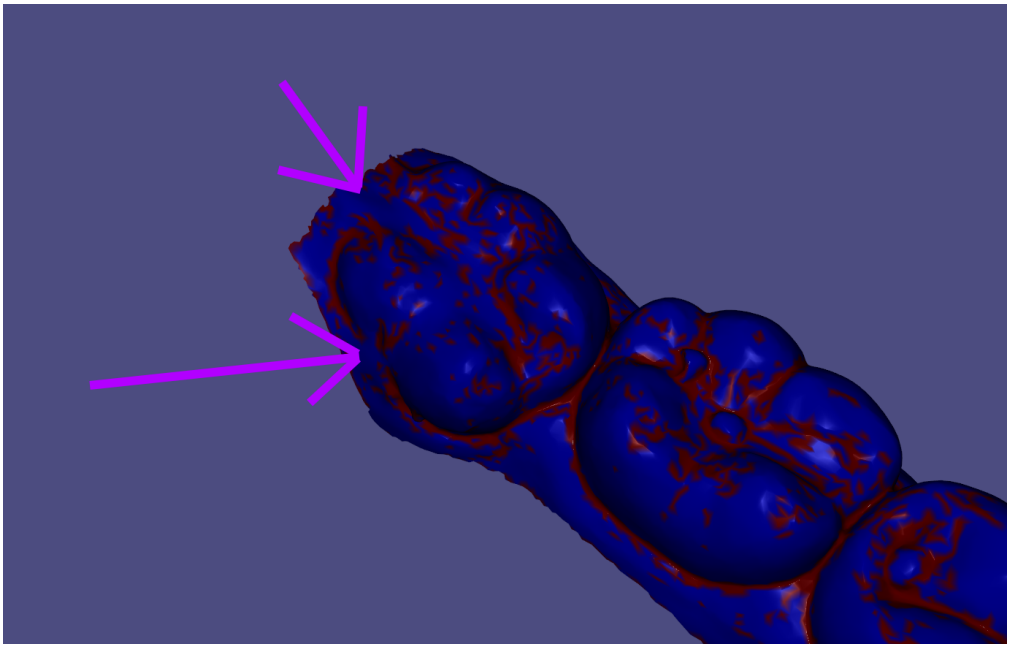


Figura 5.5: Áreas cóncavas faltantes - Concavidades en rojo

Estas anatomías constaban de dientes que en su área lingual, no contenían un borde cóncavo, esto indican las flechas en las figura 5.5.

Como la segmentación por campos armónicos considera que todo diente se encuentra en su gran mayoría encerrado por un borde cóncavo, falla al encontrarse con esta área considerable que no lo contiene, creando una fuga como se observa en la figura 5.6.

En el paper se resuelve este problema con una curva de NURBS (Non-uniform rational B-spline). Pudo haberse implementado la solución planteada en la publicación, pero dado que no resolvería el tercer problema se decidió por otra alternativa, que se detallará más adelante, para dar una solución global.

5.2.3. Problema 3 - Anatomías no Detalladas

El tercer problema consistió en anatomías no detalladas por el paper. Estas son las concavidades en los surcos de las superficies oclusales de los incisivos. Con el paso del tiempo y el uso de la dentición, las superficies oclusales se van sometiendo a desgaste, o en jerga técnica, atrición. Esto se exacerba considerablemente en las personas que sufren de bruxismo[10]. Este desgaste se puede observar en la figura 5.7.

En múltiples modelos se encontraron estas concavidades, con distintos niveles de severidad. En particular en el caso del alumno la atrición es bastante avanzada, como se observa en las áreas cóncavas indicadas por flechas en la figura 5.8. Estas áreas debiesen ser convexas, sin embargo, al tener bruxismo y una mordida cruzada antes de iniciar el tratamiento de ortodoncia, hubo un desgaste considerable dejándolas cóncavas.

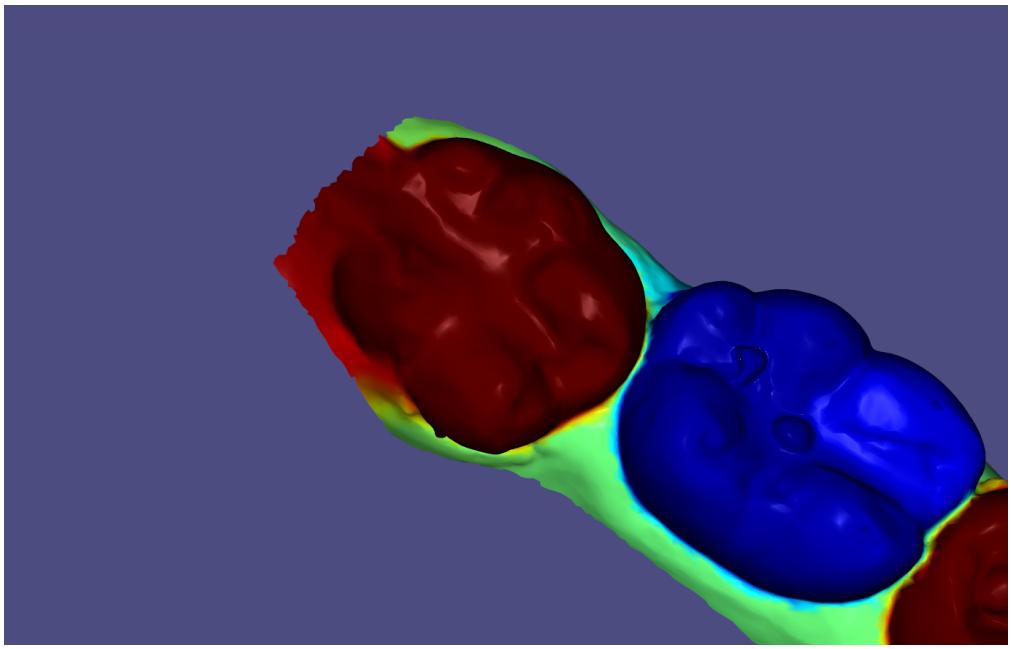


Figura 5.6: Fugas en la segmentación

Figura 5.7: Atrición severa

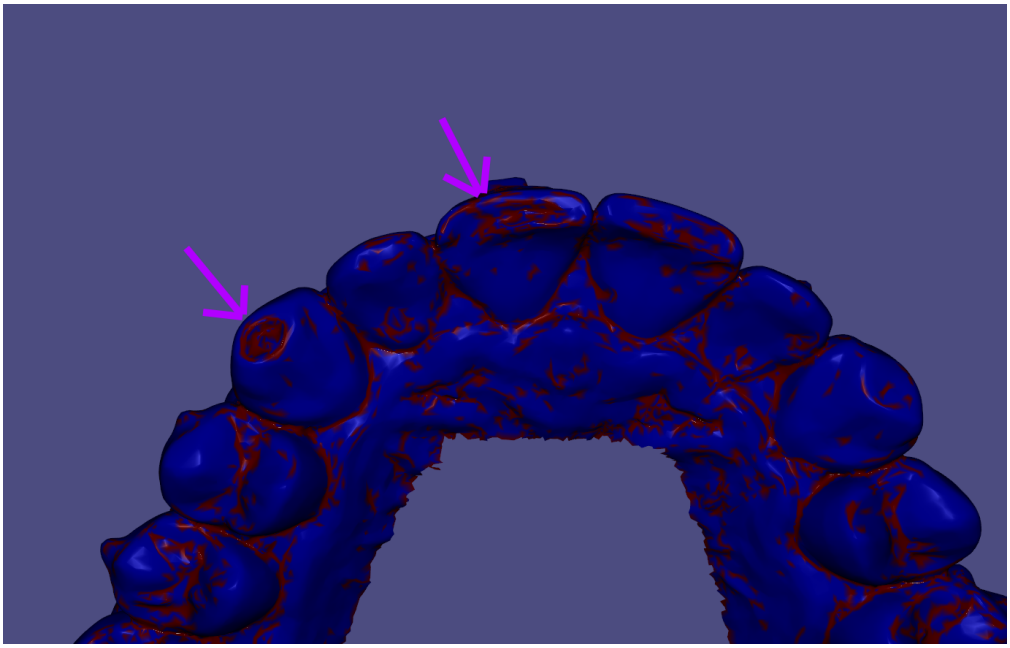


Figura 5.8: Atrición del alumno

El problema que esto produce en la segmentación es que el algoritmo asume que toda componente convexa de cada diente contiene al menos un punto clave. Esto sucede tanto en molares como premolares, ya que el algoritmo de detección de puntos claves detecta varios puntos en las distintas zonas oclusales. Por otra parte, en los incisivos y caninos el algoritmo solo encuentra puntos en la superficie oclusal que puede estar sometida a atrición, dejando una parte convexa sin punto clave como se observa en la figura 5.9.

Esto finalmente produce errores en la segmentación, observables en la figura 5.10.

5.2.4. Soluciones

Resolver el primer problema, de la ausencia de base, constó de cambios en el programa de detección de puntos claves. Estos cambios fueron relativamente simples ya que bastó con remover la parte del código que se encargaba de la eliminación de los puntos de la base y encía. Por otro lado, el segundo problema pudo haber sido resuelto utilizando la implementación de las curvas de NURBS. Pero al considerar que el paper no contempló las atriciones en los incisivos se determinó que se agregaría un módulo de edición manual. En este módulo el usuario puede agregar o eliminar nuevos puntos claves, para garantizar que la segmentación final sea correcta.

5.3. Editor de Puntos Claves

Para resolver los problemas planteados anteriormente se creó la herramienta de edición de puntos claves, mostrada en la figura 5.11. Esta herramienta pretende ser usada únicamen-

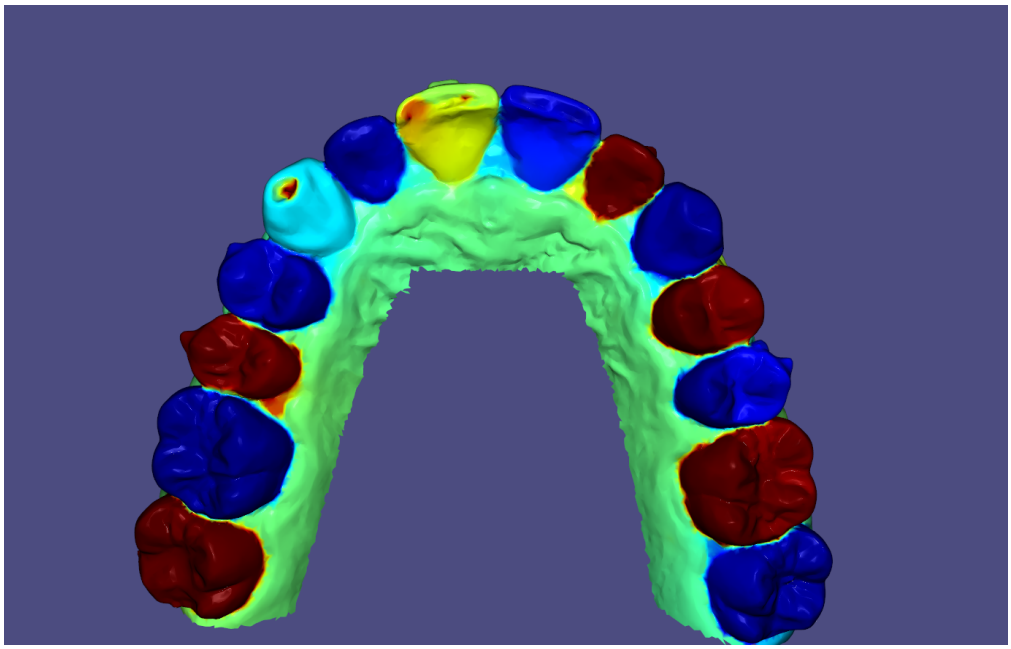


Figura 5.9: Problemas causados por atrición en el campo armónico. El coloreo no cubre totalmente los dientes

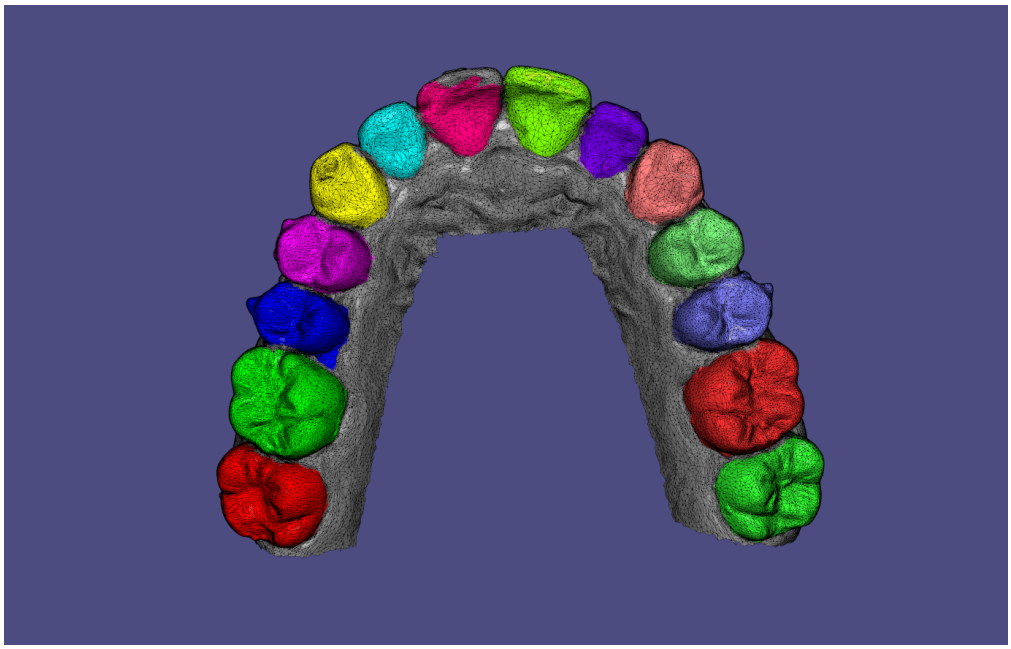


Figura 5.10: Problemas en segmentación por atrición

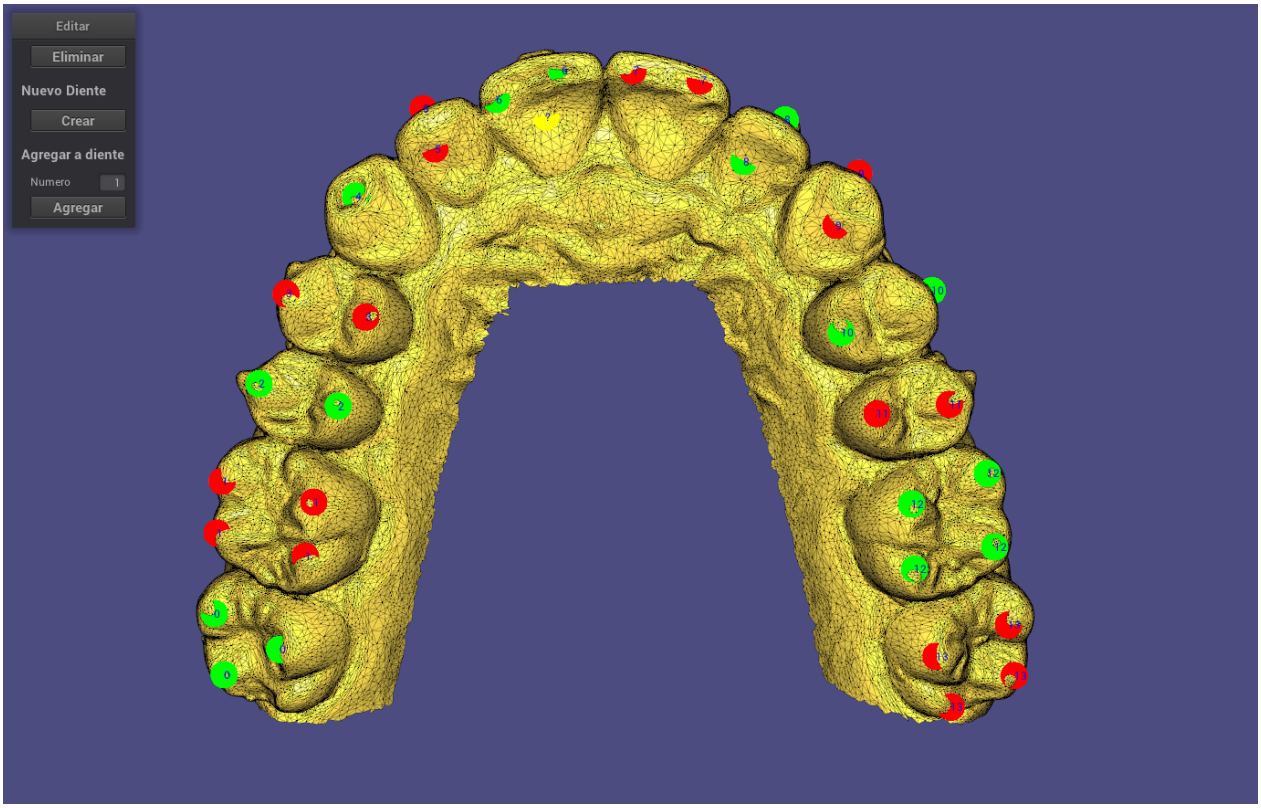


Figura 5.11: Editor de puntos claves. Cada punto en rojo y verde corresponde a un punto clave que se puede eliminar. Pueden agregarse también nuevos puntos.

te cuando los programas de detección automática y campo armónico no llegan al resultado deseado en conjunto. La creación de esta herramienta va en contra del objetivo inicial de segmentación completamente automática. Mas debe considerarse que el objetivo de la correctitud prepondera sobre la automaticidad. Desde una perspectiva de negocios una vez que se ha obtenido un paciente, convencido de iniciar el tratamiento, escaneado digitalmente, no es sustentable comercialmente no proseguir con el tratamiento ya que el segmentador automático no logra procesar su modelo.

Al ir agregando puntos claves antes del programa de campo armónico, se garantiza una segmentación correcta, como se explicará más adelante.

5.4. Desempeño del Algoritmo de Detección de Puntos Claves

Para evaluar el desempeño del programa de detección de puntos claves se medirán las siguientes variables:

1. Número de puntos claves incorrectamente detectados.
2. Número de piezas dentales no detectadas.

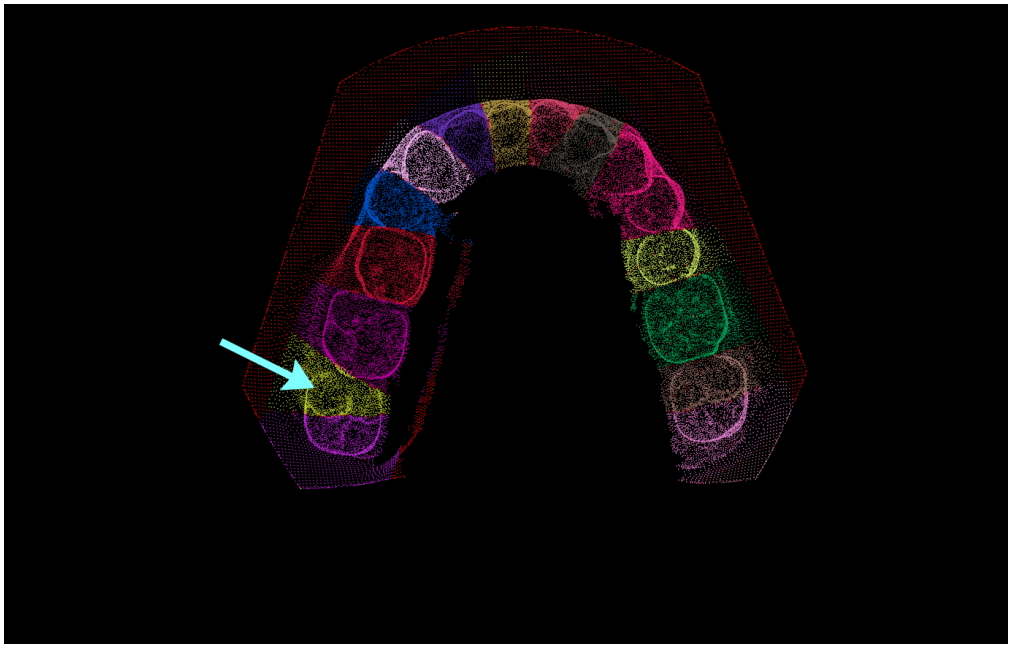


Figura 5.12: Molar subdividido en dos partes incorrectamente

3. Número de piezas dentales sobrantes, o divisiones de dientes que no corresponden a una única unidad. Ver figura 5.12.

Estos parámetros son relevantes porque el algoritmo armónico asume que hay al menos un punto clave por área cóncava del diente. Si es que hay dientes no detectados, o subdivisiones de dientes, es altamente problable que el algoritmo de campo armónico falle. Asimismo, si hay puntos claves de dientes detectados en zonas que no lo son, como la encía, el algoritmo armónico también aumenta su probabilidad de fallar.

Vale notar que no es trivial determinar los puntos correctamente detectados frente aquellos que no. Por ejemplo, en la figura 5.13 se puede observar dos piezas dentales detectadas como una sola. Ambas demarcadas con puntos claves verdes. Se podría determinar que los seis puntos del premolar, aquellos vecinos a los puntos celestes, son los detectados de manera correcta. Pero se podría también decir que los correctos son los ocho puntos detectados en el molar, hacia la parte inferior de la imagen, y que los del premolar son incorrectos. Se determinó la convención de que la pieza que tenga la mayor cantidad de puntos correctamente detectados determinará que esos puntos son correctos. Cualquier punto clave detectado por el programa como la misma pieza, pero posicionado fuera de esta, es entonces incorrecto. En el caso del ejemplo, los ocho puntos del molar son determinados como los correctos por ser un número mayor a los seis puntos del premolar.

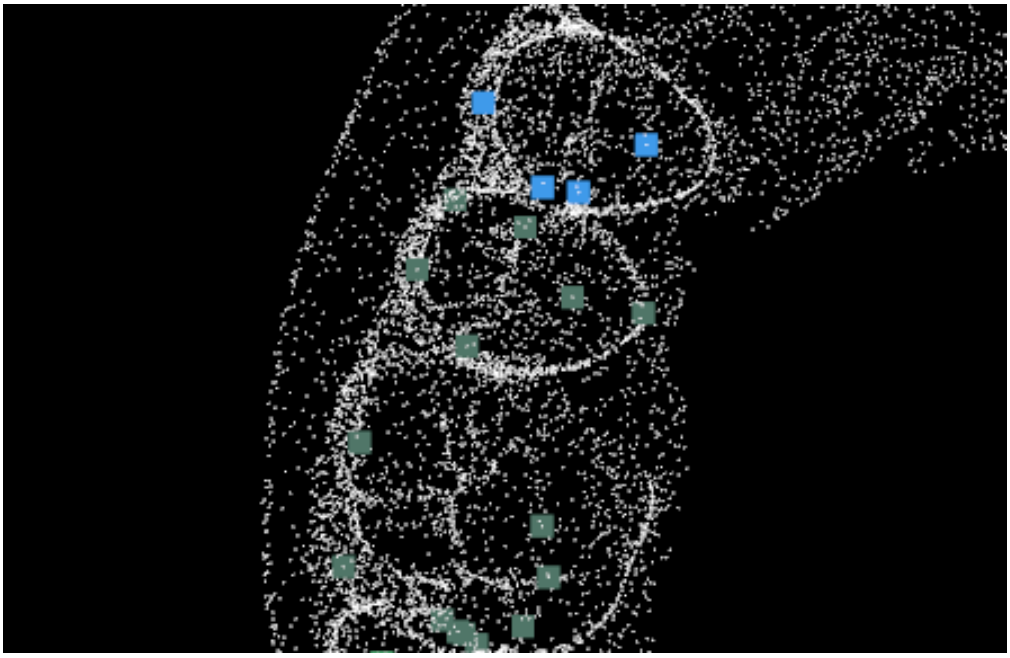


Figura 5.13: Puntos claves pertenecientes a distintos dientes detectados como uno solo en verde. Puntos claves correctamente detectados en azul

Parámetro	Tasa de Éxito
Sin Piezas Sobrantes	86 %
Sin Piezas Faltantes	63 %
Sin Piezas Sobrantes ni Faltantes	50 %
Sin Puntos Claves Incorrectos	36 %
Sin Ninguna de las Anteriores	31 %

Tabla 5.1: Resultados Obtenidos

Un 31 % de los modelos alcanzaron una detección perfecta. La dificultad para alcanzarla, se encuentra en que al ajustar el sistema para ser más restrictivo con algún caso de error, existe otro error inversamente correlacionado que aumenta. En particular, al ser más prohibitivo con la detección de puntos claves, disminuye la tasa de puntos claves detectados incorrectamente, sin embargo aumenta la cantidad de piezas que no tienen ningún punto clave detectado. Asimismo al aumentar el nivel de curvatura y profundidad requerido para detectar un intersticio el número de piezas sobrantes disminuye pero las piezas faltantes aumentan, lo mismo ocurre de manera inversa. Por esto llegar a la detección perfecta en la totalidad de los casos es un problema de alta complejidad. Se decidió por no abordar este problema, ya que incluso si se llegara a una detección perfecta, la segmentación final con el algoritmo armónico no está garantizada, como se detallará en la siguiente sección.

Modelo	Fuente	Arcada	Piezas Sobrantes	Piezas Faltantes	Puntos Claves Incorrectos
1	Pública	Superior	0	1	9
2	Pública	Inferior	2	0	6
3	Pública	Superior	0	0	0
4	Pública	Inferior	0	0	0
5	Pública	Superior	2	1	5
6	Pública	Inferior	0	0	0
7	Propia	Superior	0	0	0
8	Propia	Inferior	0	0	0
9	Propia	Superior	1	0	1
10	Propia	Inferior	0	1	2
11	Propia	Superior	0	0	1
12	Propia	Inferior	0	2	3
13	Propia	Superior	0	1	3
14	Propia	Inferior	0	0	1
15	Propia	Superior	0	0	2
16	Propia	Inferior	0	1	2
17	Propia	Superior	0	0	0
18	Propia	Inferior	0	2	2
19	Propia	Superior	0	0	0
20	Propia	Inferior	0	0	0
21	Propia	Superior	0	0	3
22	Propia	Inferior	0	4	5

Tabla 5.2: Detalle de resultados de pruebas de detección de puntos claves

5.5. Desempeño del Algoritmo de Campo Armónico

La primera pregunta al medir el desempeño de este algoritmo es determinar que se quiere medir. A diferencia del programa de detección automática de cúspides, el programa de segmentación utilizando campo armónico debe arrojar un resultado que sea utilizable para fabricar un tratamiento de ortodoncia invisible. Se detallarán los pasos a seguir post segmentación en el próximo capítulo. Acorde a lo documentado en ellos y a la experimentación empírica, para poder realizar un tratamiento se requiere que cada diente no capture partes de otro diente, ni que su error con respecto a su límite real con la encía sea mayor a un milímetro.

5.5.1. Con el Input Resultante del Programa de Detección de Puntos Claves

Al realizar las pruebas del algoritmo de campo armónico ocupando de entrada la salida del algoritmo de detección de puntos claves, es importante notar que es imposible obtener una segmentación correcta con una entrada que tiene piezas sobrantes o faltantes. Mas sí puede ejecutarse correctamente si tiene puntos claves detectados de manera incorrecta sin piezas faltantes ni sobrantes.

A modo de ejemplo, en las figuras 5.14 y 5.15 se pueden observar los resultados con piezas faltantes y sobrantes respectivamente.

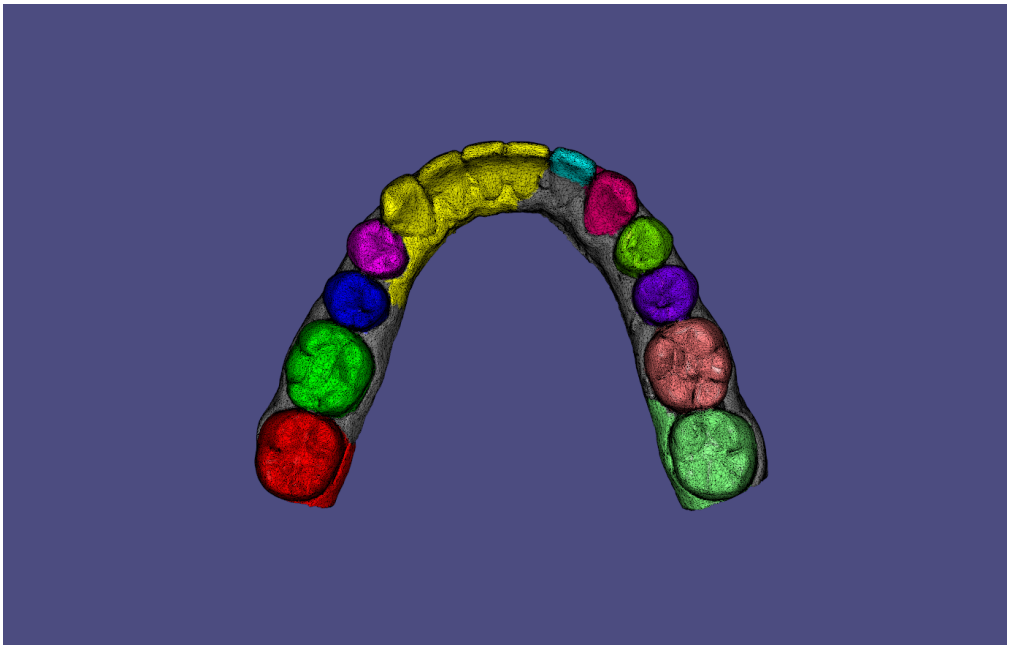


Figura 5.14: Piezas faltantes

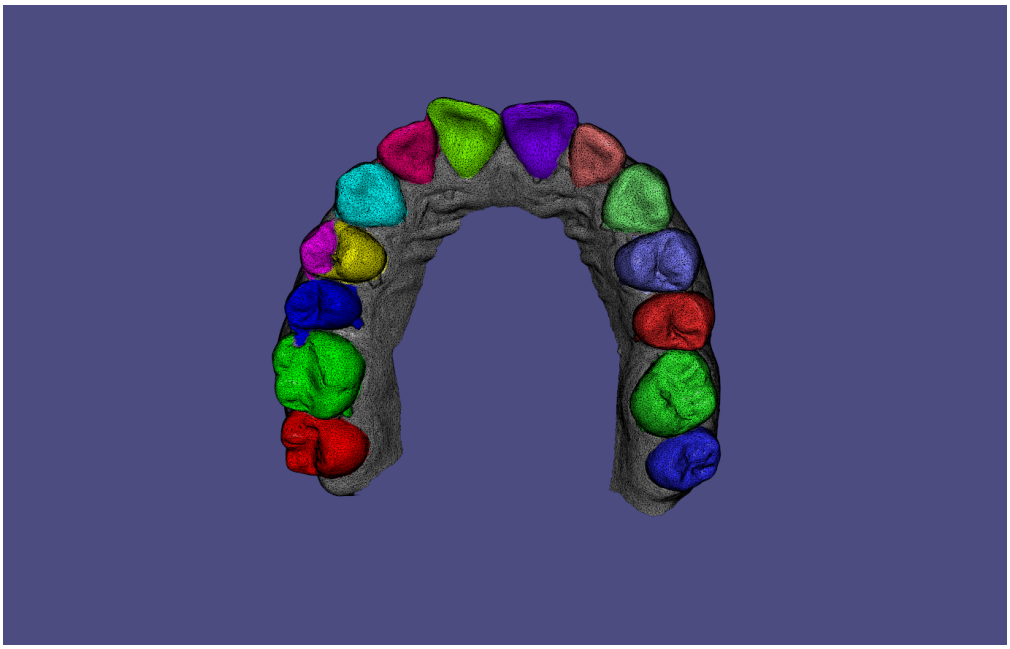


Figura 5.15: Piezas sobrantes

En consecuencia, para analizar la efectividad de la segmentación con campo armónico, utilizando la detección automática del programa de puntos claves, se ocuparán solo aquellos casos en que la detección automática no cuenta con piezas faltantes ni sobrantes. Esto representa un 50 % de los 22 modelos originales.

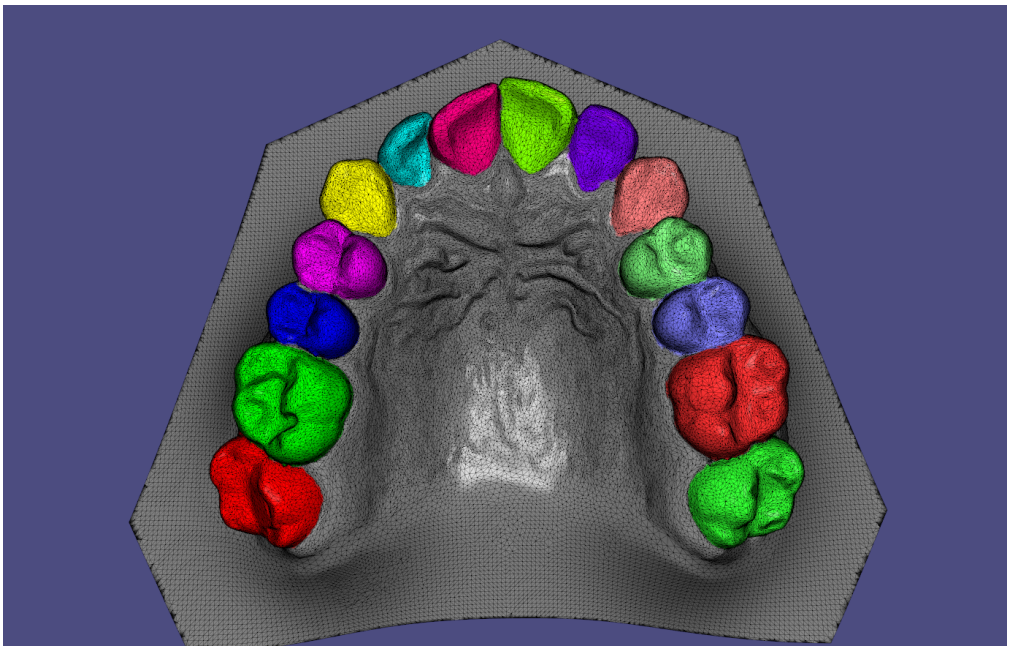
De los 11 modelos probados, únicamente 2 consiguieron una segmentación correcta, observable en la figura 5.16. Cabe destacar que estos dos modelos pertenecen al conjunto de modelos que se encontraba en dominio público mediante internet. Al ser modelos publicados con el objetivo de mostrar los detalles de la dentición en alta fidelidad, resulta lógico que se haya obtenido una segmentación correcta.

El problema más común fueron casos de dientes que indicaban que una gran parte de la encía que los circundaba les pertenecía, debido a áreas cóncavas faltantes, o fugas cóncavas, problema que se detalló anteriormente en la figura 5.5. Este problema sucedió en un 54 % de los casos.

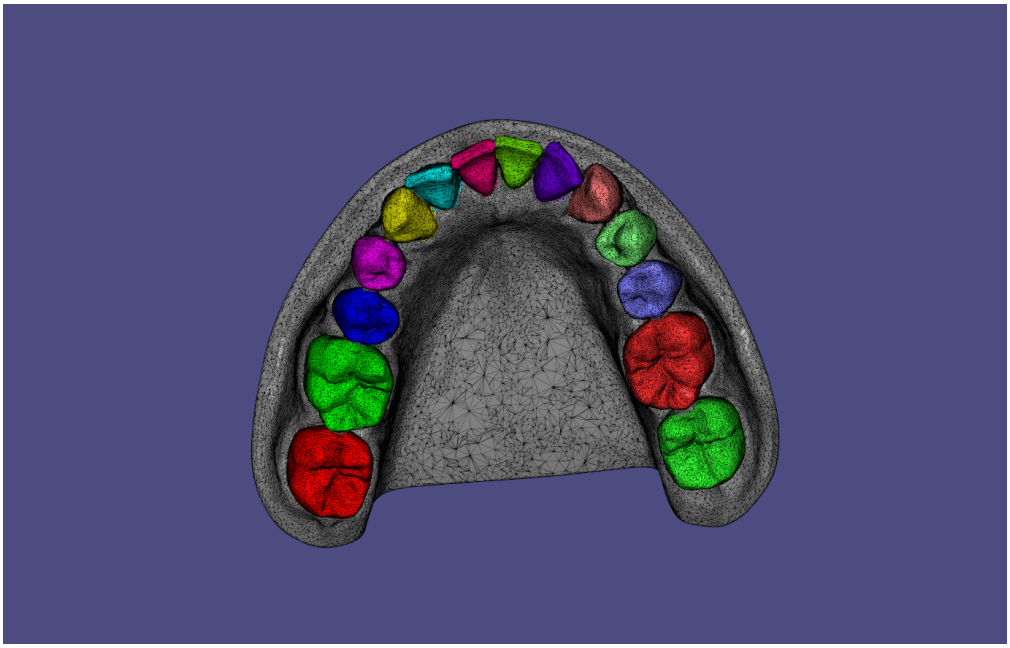
El otro problema fue la falta o sobra de algún punto clave. En algunos casos un diente poseía un área cóncava que no estaba indicada por ningún punto clave, como se observa en el primer molar derecho de la figura 5.17.

5.5.2. Con Input Iterativo

Ocupando el programa de edición de puntos claves, es trivial entender que este garantiza la segmentación correcta en todos los casos. Para hacerlo, en el peor caso, basta con que el usuario determine el valor correcto para cada vértice en la malla, realizando él la segmentación de manera completamente manual. En la práctica la solución es muchísimo más eficiente. El usuario solamente debe añadir los puntos de las piezas faltantes, eliminar los puntos incorrectamente detectados y entregarle esa nueva detección asistida al programa de campo armónico. Si el programa armónico presenta errores de puntos claves faltantes y/o fugas cóncavas, el usuario debe volver al editor de puntos claves hasta llegar a la segmentación correcta. Se probó segmentar los 22 modelos de manera correcta, y se comprobó que en el peor caso se requirió tres iteraciones y la añadidura de 19 puntos claves para llegar a la segmentación correcta.



(a) Maxilar



(b) Mandíbula

Figura 5.16: Segmentación correcta

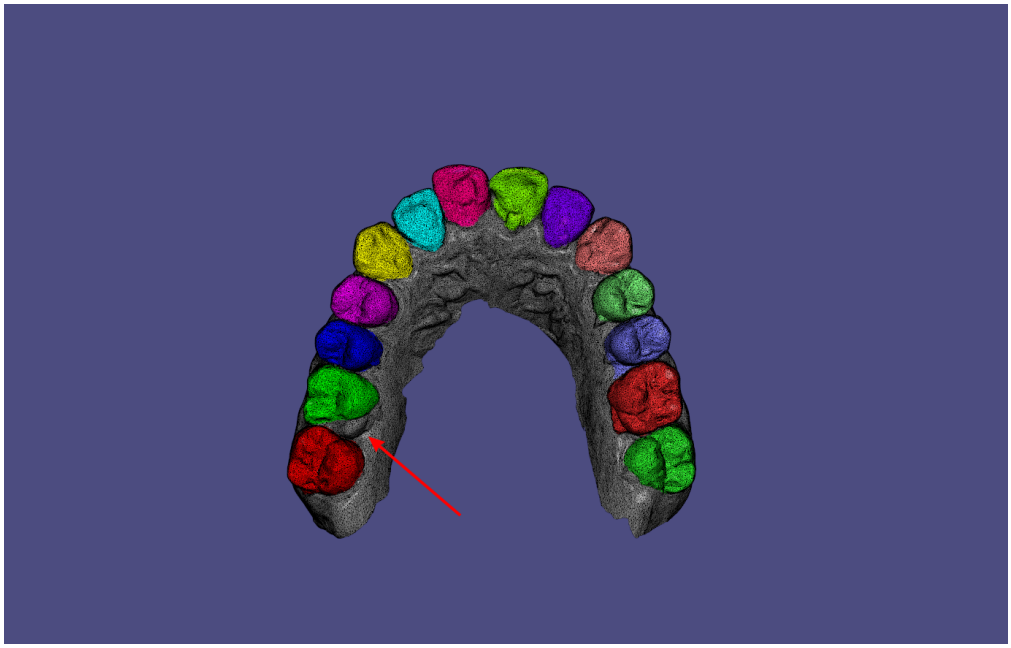


Figura 5.17: Punto clave faltante, produce segmentación parcial del molar indicado por la flecha

Capítulo 6

Conclusiones

6.1. Resultados Obtenidos

El software desarrollado cumple la meta de segmentar correctamente los dientes con ayuda de un usuario, permitiendo desarrollar y entregar tres tratamientos de ortodoncia a la fecha. Como se detalló en el capítulo quinto, no se logró obtener una segmentación completamente automática. Esto se debió principalmente a que se encontraron dos tipos de anatomías no detalladas en las publicaciones implementadas, siendo estas, las áreas cóncavas por atrición y las fugas cóncavas.

Se plantearon tres objetivos en el inicio de la memoria, siendo estos, correctitud, automaticidad y output interpretable. El camino a seguir para implementar estos objetivos fue implementar tres publicaciones y sus algoritmos de segmentación dental. Todos los algoritmos fueron implementados y testeados. Los primeros dos algoritmos fueron descartados como absolutamente insuficientes, y al tercer algoritmo, la segmentación utilizando campos armónicos se le agregó una entrada manual para lograr segmentar cualquier caso. En consecuencia, se sacrificó el objetivo de automaticidad a cambio de la correctitud, permitiéndole al programa segmentar cualquier entrada con la ayuda de un usuario. La salida del programa consta de archivos .csv y .ply, siendo interpretables por otros programas como se demuestra en la sección de trabajo futuro. En síntesis se cumplieron dos de tres objetivos planteados, resultando en un programa de salida correcta, interpretable y semi automático.

6.2. Aprendizaje

El objetivo incompleto de este trabajo es de tener una automaticidad absoluta. Se descubrió la alta dificultad de tener un sistema que funcione para el total de los casos sin tener ningún grado de input humano. Dado que el programa requiere funcionar para todo caso que ingrese a este, se debió asumir desde un principio que el sistema requeriría un asistente humano. Se concluye que en general un programa que requiera una correctitud total, debe

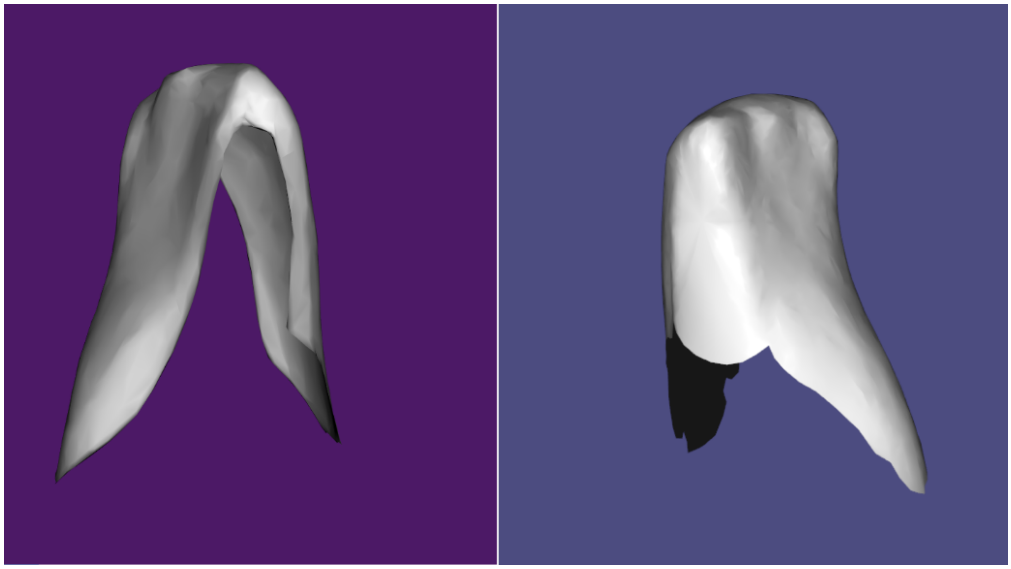


Figura 6.1: Fill and Fair de Áreas Faltantes

tener una opción de modificación manual humana.

Considerando que la automaticidad total no era factible, se estima que hubiese sido más eficiente crear un interfaz completamente manual para ir posteriormente optimizando las partes más demandantes de tiempo del usuario y más fáciles de optimizar.

6.3. Trabajo Futuro

El programa de segmentación no requiere mayores mejoras en cuanto a su funcionalidad. Sin embargo, sí se podría mejorar la interfaz gráfica para no tener que lanzar los programas mediante una línea de comando permitiendo el uso de usuarios no expertos. De todas maneras los mayores desafíos vienen post segmentación. Cada uno de los dientes debe recibir estimaciones de las formas no presentes en el escaneo, realizadas con algoritmos de *fill and fair* o triangulación y suavizado como se demuestra en la figura 6.1.

Con las partes faltantes estimadas se puede proceder a editar sus posiciones con un programa de edición dental, desarrollado también por el alumno, cuyos resultados se pueden observar en la sección 6.4.

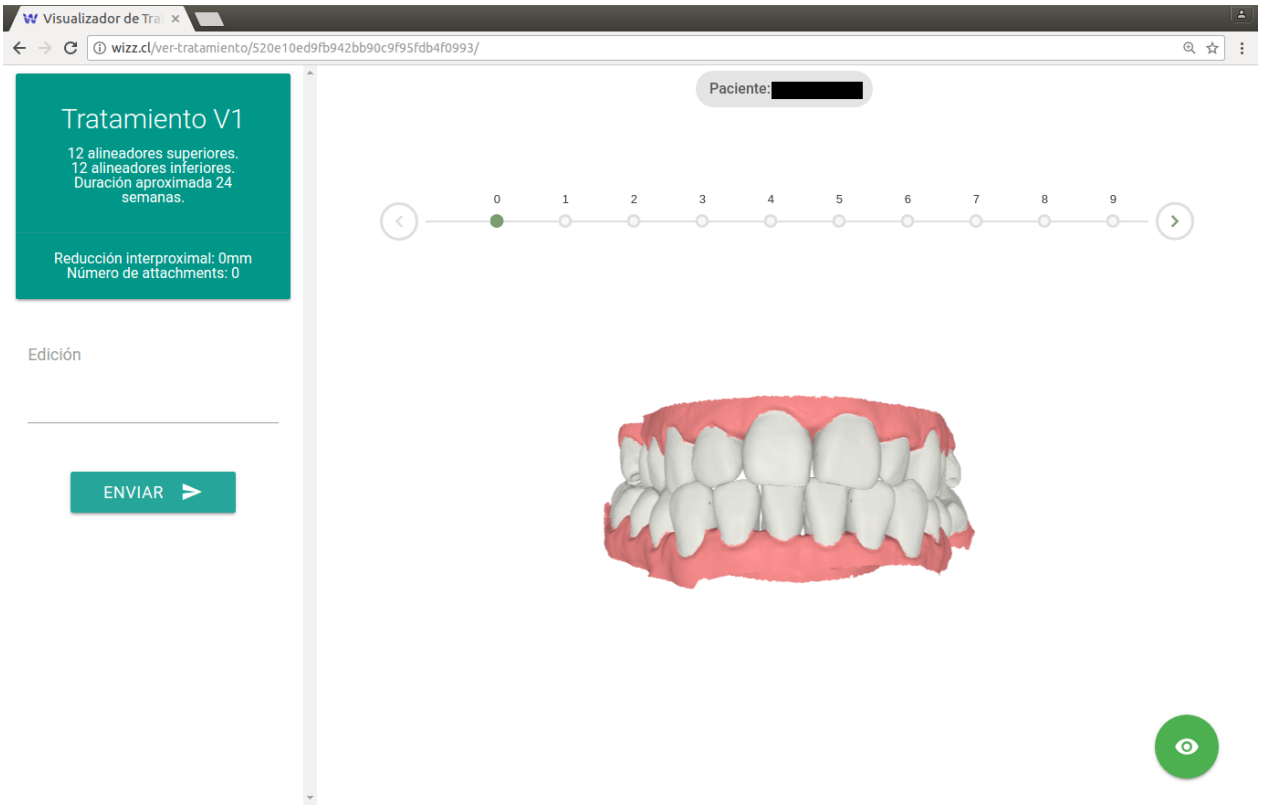
6.4. Tratamientos Realizados

Se adjuntan los estados iniciales y finales del tratamiento de ortodoncia de una paciente. El tratamiento de ortodoncia de esta paciente ha sido diseñado únicamente con software desarrollado por el alumno. El tratamiento ha sido fabricado con impresión 3D y termoformado y al momento de escritura, la paciente ya ha avanzado por más de la mitad de este, con buenos

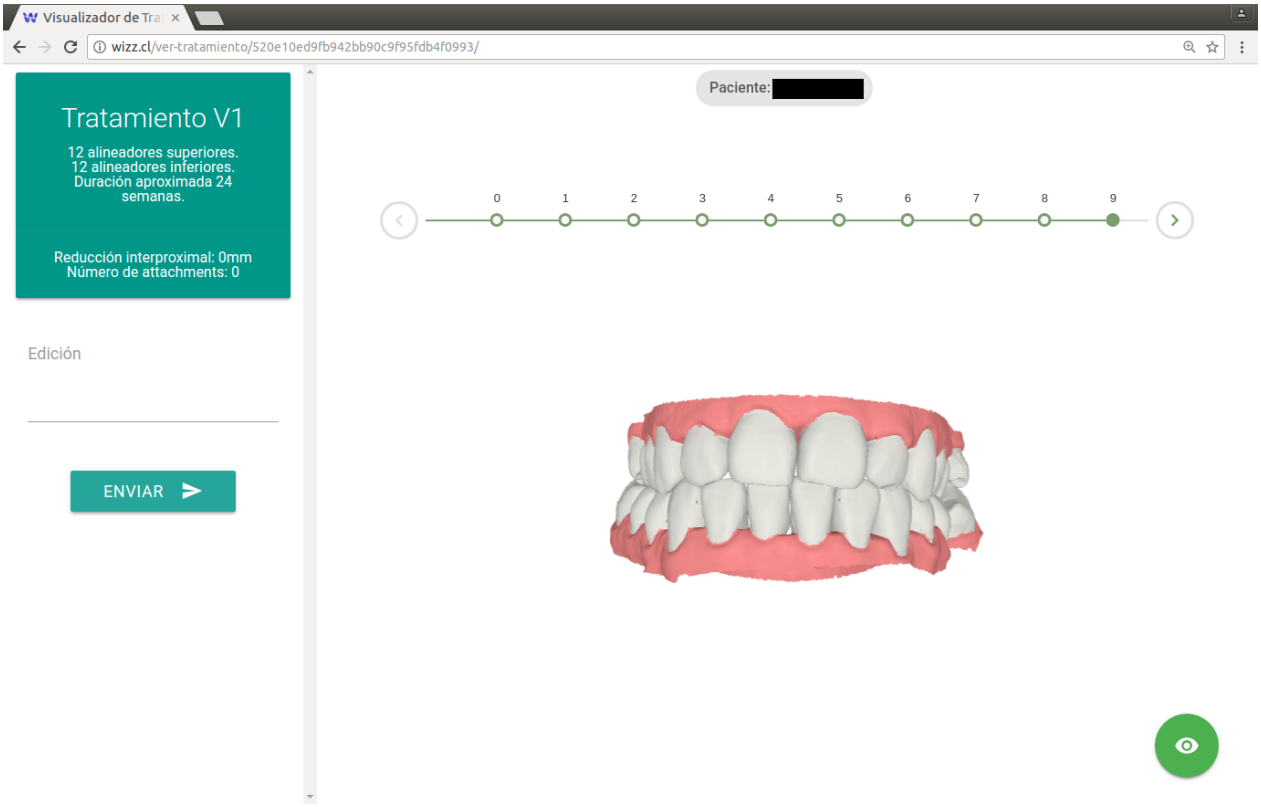
resultados verificados por el ortodoncista tratante.

La paciente presenta una mordida cruzada y la preocupación estética de la posición de su lateral superior derecho. Se presenta un tratamiento de 12 alineadores superiores e inferiores corrigiendo tanto la mordida como la estética.

Los pasos iniciales y finales se pueden observar en las figuras 6.2, 6.3 y 6.4.

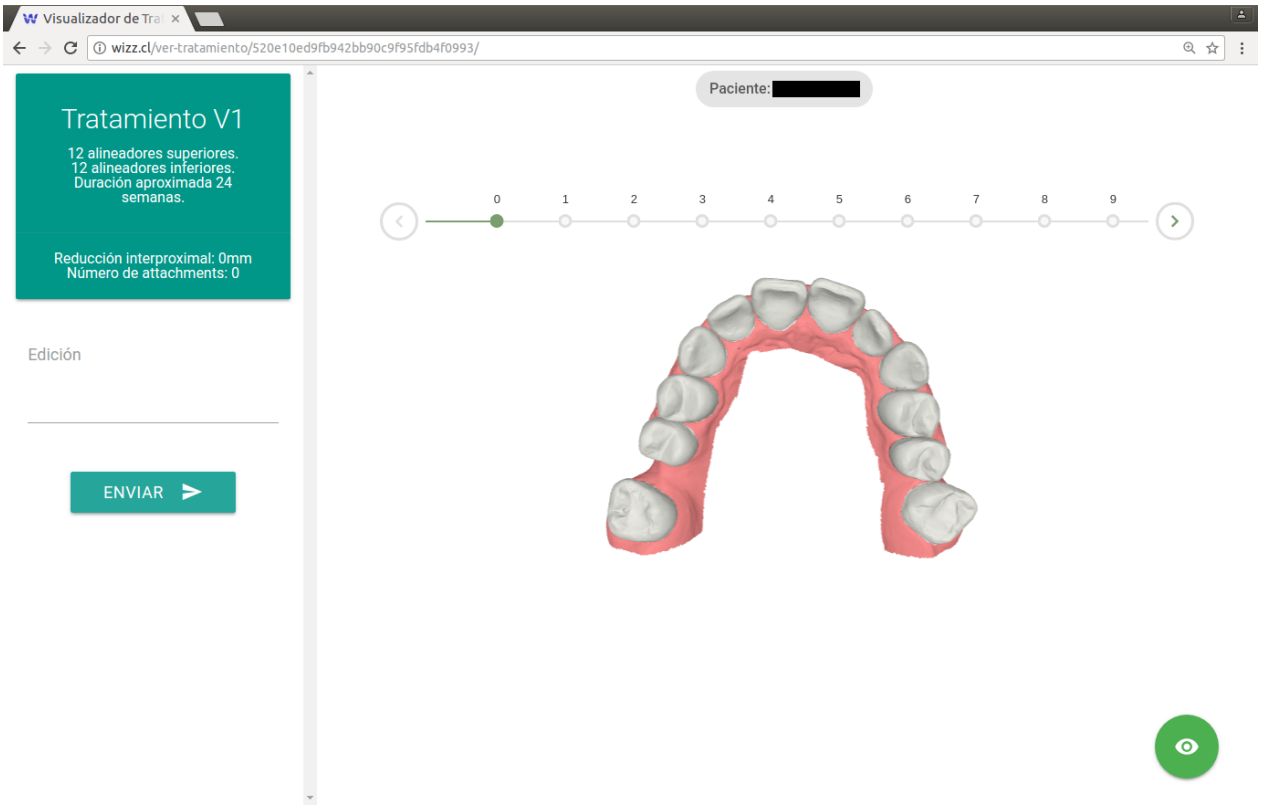


(a) Inicial

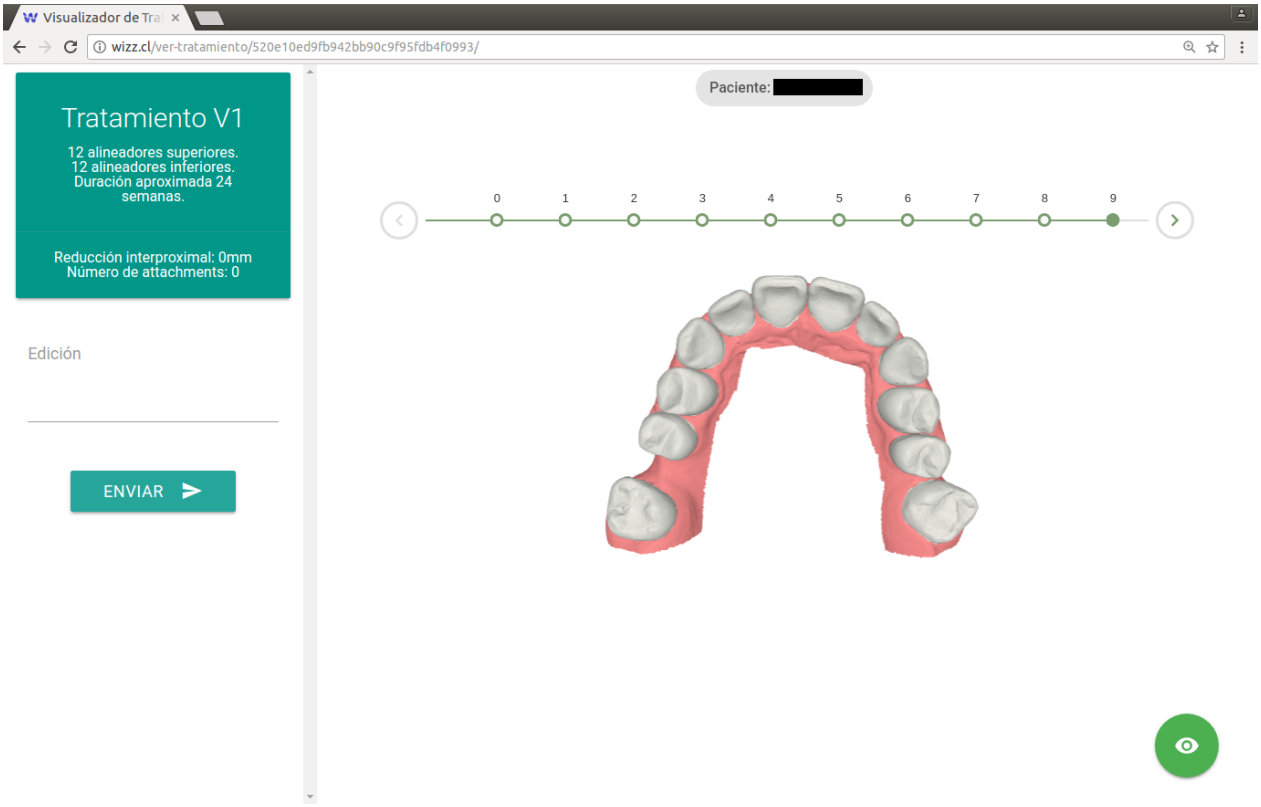


(b) Final

Figura 6.2: Oclusión

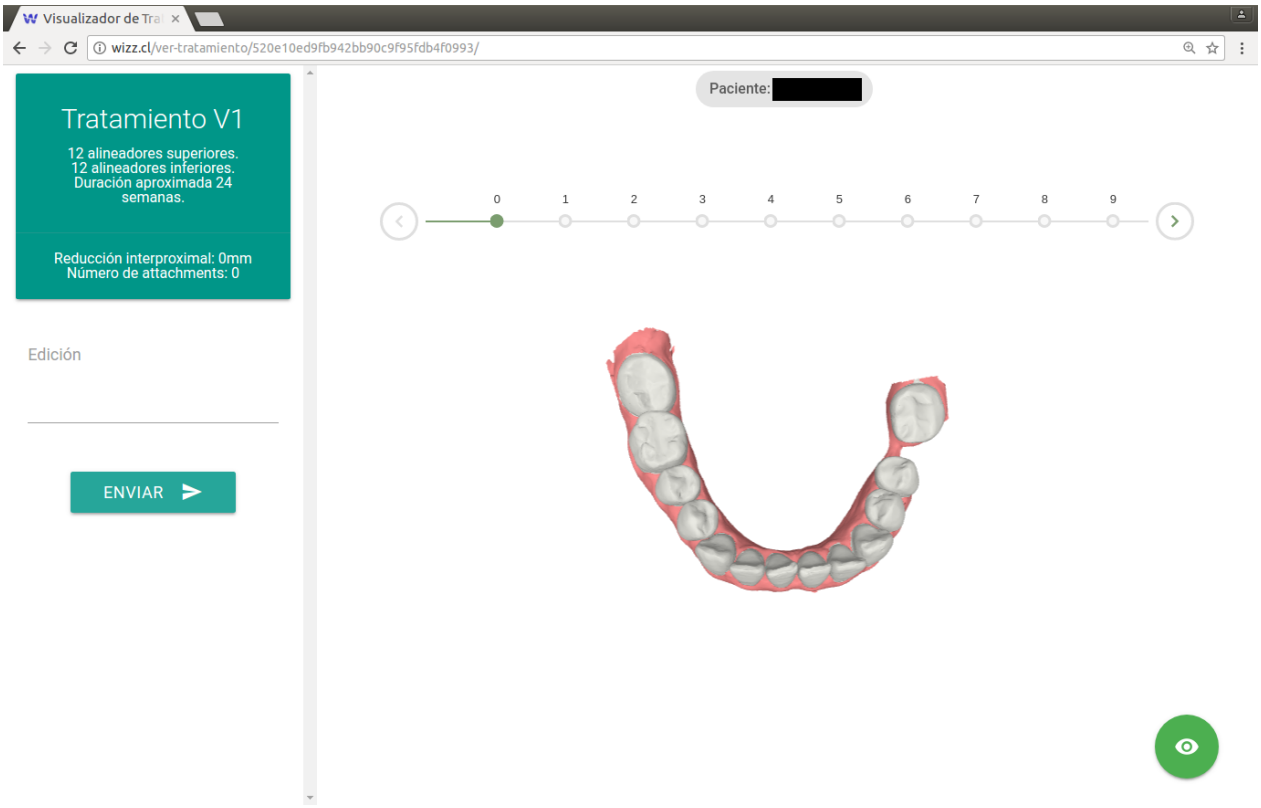


(a) Inicial

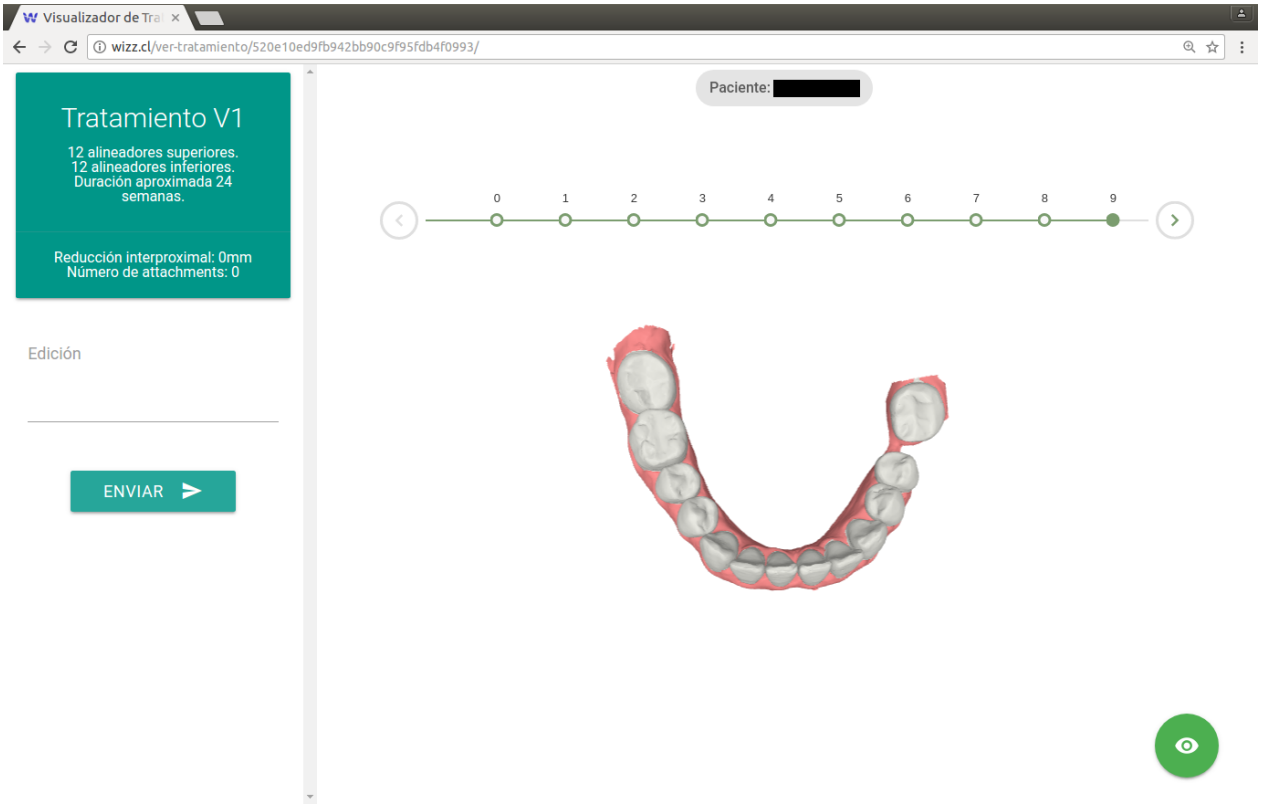


(b) Final

Figura 6.3: Arcada Superior



(a) Inicial



(b) Final

Figura 6.4: Arcada Inferior

Capítulo 7

Bibliografía

[1] Oscar Kin-Chung Au, Youyi Zheng, Menglin Chen, Pengfei Xu, and Chiew-Lan Tai. Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1125–1134, jul 2012.

[2] Ceres solver. `http://ceres-solver.org`. Última visita: 2017-01-24.

[3] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, jul 2007.

[4] Muhammad Chishti and Richard Ridgley. Method and system for incrementally moving teeth, April 17 2001. Patente Estadounidense 6.217.325.

[5] Visual Computing Lab ISTI CNR. Meshlab. http://meshlab.sourceforge.net/.

[6] Amos Dudley. Orthoprint, or how i open-sourced my face. `http://amosdudley.com/weblog/Ortho`. Última visita: 2017-01-24.

[7] Eigen. `https://eigen.tuxfamily.org/dox-devel/group__TopicSparseSystems.html`. Última visita: 2017-01-26.

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in Computer Vision*, pages 726–740. Elsevier BV, 1987.

[9] Aleksey Golovinskiy and Thomas Funk. Min-cut based segmentation of point clouds. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. Institute of Electrical and Electronics Engineers (IEEE), sep 2009.

[10] John O Grippo, Marvin Simring, and Steven Schreiner. Attrition, abrasion, corrosion and abfraction revisited: a new perspective on tooth surface lesions. *The Journal of the American Dental Association*, 135(8):1109–1118, 2004.

[11] Sheng hui Liao, Shi jian Liu, Bei ji Zou, Xi Ding, Ye Liang, and Jun hui Huang. Au-

tomatic tooth segmentation of dental mesh based on harmonic fields. *BioMed Research International*, 2015:1–10, 2015.

[12] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2016. http://libigl.github.io/libigl/.

[13] Bei ji Zou, Shi jian Liu, Sheng hui Liao, Xi Ding, and Ye Liang. Interactive tooth partition of dental mesh base on tooth-target harmonic field. *Computers in Biology and Medicine*, 56:132–144, jan 2015.

[14] T. Kondo, S.H. Ong, and K.W.C. Foong. Tooth segmentation of dental study models using range images. *IEEE Transactions on Medical Imaging*, 23(3):350–362, mar 2004.

[15] T. Kondo, S.H. Ong, and K.W.C. Foong. Tooth segmentation of dental study models using range images. *IEEE Transactions on Medical Imaging*, 23(3):350–362, mar 2004.

[16] DA Kuncio. Invisalign: current guidelines for effective treatment. *The New York state dental journal*, 80(2):11–14, 2014.

[17] A.P. Mangan and R.T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[18] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International archives of photogrammetry, remote sensing and spatial information sciences*, 36(5):248–253, 2006.

[19] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (PCL). In *2011 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers (IEEE), may 2011.

[20] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. Point feature extraction on 3d range scans taking into account object boundaries. In *2011 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers (IEEE), may 2011.

[21] J.D. Tamayo-Quintero, S. Arboleda-Duque, and J.B. Gómez-Mendoza. Semi-automatic teeth segmentation in 3d models of dental casts using a hybrid methodology. In *Emerging Trends in Image Processing, Computer Vision and Pattern Recognition*, pages 433–446. Elsevier BV, 2015.

[22] Federico Tombari and Luigi Di Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*. Institute of Electrical and Electronics Engineers (IEEE), nov 2010.

[23] Youyi Zheng and Chiew-Lan Tai. Mesh decomposition with cross-boundary brushes. *Computer Graphics Forum*, 29(2):527–535, may 2010.

[24] Youyi Zheng, Chiew-Lan Tai, and O. K-C Au. Dot scissor: A single-click interface

for mesh segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1304–1312, aug 2012.