UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN
SEÑALES, IMÁGENES Y AMBIENTES VIRTUALES

HERRAMIENTAS COMPUTACIONALES PARA EXTRAER, REPRESENTAR Y ANALIZAR RASGOS FACIALES

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
SAÚL ALEXIS HEREDIA PÉREZ

TUTOR PRINCIPAL:
DR. JORGE ALBERTO MÁRQUEZ FLORES
CCADET, UNAM

COTUTOR:
DR. MIGUEL ÁNGEL PADILLA CASTAÑEDA
CCADET, UNAM

CIUDAD DE MÉXICO, DICIEMBRE 2016

AGRADECIMIENTOS

A mis padres y hermana por su cariño, apoyo y confianza.

A mis compañeros y amigos por su apoyo y motivación.

A mis profesores, especialmente al Dr. Jorge Márquez, Dr. Miguel Padilla y Dr. Alfonso Gastélum por su valiosa ayuda y orientación.

A los profesores Dra. Elena Martínez y Dr. Boris Escalante por su colaboración como sinodales.

Al personal administrativo del Posgrado en Ciencia e Ingeniería de la Computación por toda su ayuda.

Al CONACYT por la Beca de Estudios de Maestría.

CONTENIDO

TABLA DE ILUSTRACIONES

RESUMEN

En este trabajo se presenta un sistema asistido por computadora para la extracción interactiva de puntos de referencia en mallas tridimensionales de rostros humanos, así como una metodología para el análisis estadístico de dichos puntos. En el software desarrollado, se emplean técnicas de renderizado en tiempo real y selección interactiva mediante detección de colisiones y retroalimentación háptica, para la interacción intuitiva del usuario con el modelo virtual. Explotando la información geométrica de las mallas, mediante el cálculo y visualización de las curvaturas e índice de forma, se ofrece al usuario una mejor percepción de los modelos 3D mediante el uso de mapas de colores. El método propuesto fue probado en una base de datos de 35 cabezas de individuos mexicanos sanos, obtenidos con un sistema de visión estereoscópica con luz estructurada de bajo costo. Uno de los objetivos de este estudio fue determinar la variabilidad de un conjunto de 19 puntos de referencia faciales, en los ojos, boca, nariz, mejillas y mentón. Para determinar la validez de los puntos seleccionados a mano, se realizó un análisis del Error Técnico de Medición (TEM). Después de extraer los puntos, se aplicó un registro rígido de los puntos de referencia, hacia los puntos de un modelo de referencia de la cabeza humana, mediante una transformación rígida óptima determinada a partir de la matriz de covarianza cruzada. Para obtener los puntos de referencia promedio y los modos de variación, se realizó un Análisis de Componentes Principales (PCA) basado en la matriz de covarianza. Por último, se obtuvo la forma aproximada de la cara promedio de la población de estudio, deformando el modelo de la cabeza de referencia mediante un método de registro deformable basado en jaulas. En dicho enfoque, el modelo de alta resolución es asociado a una malla de baja resolución, denominada malla de control o jaula, que encierra el modelo detallado, mediante el uso de Coordenadas de Valor Medio (MVC), cada vértice de la malla detallada es representado como una combinación lineal de los vértices de la malla de control, permitiendo la deformación de la malla de alta resolución, a través de la deformación la malla de control mientras se preservan los detalles, luego la malla de control es deformada de manera iterativa mediante deformación laplaciana, con el fin de minimizar la suma del cuadrado de las distancias entre puntos de referencia correspondientes. El software desarrollado, además de trabajar con puntos y medidas lineales y angulares, permite también al usuario seleccionar de forma interactiva caminos y contornos sobre la superficie de la malla, para la obtención de longitudes y áreas; e incluso trabajar sobre imágenes. Finalmente, el sistema permite el análisis de otras estructuras anatómicas complejas, y potencialmente tener otras aplicaciones como la planificación de cirugía craneofacial, el reconocimiento facial en ciencia forense, y la generación de rostros aleatorios para la creación de avatares en ambientes virtuales, así como construir bases de datos compactas de rostros 3D.

ABSTRACT

In this work, we present a computer aided system for interactive extraction of anthropometrical points (landmarks) on 3D human face meshes, and a methodology for a statistical analysis of the anthropometrical points. In the developed software, we employed real time rendering techniques and interactive picking through collisions detection and haptic feedback, to allow intuitive user interaction with the virtual model. We also exploit the geometric information of the meshes by computing and displaying the curvatures and shape index, to give to the user a better understanding of the 3D data by using color maps. The proposed method was tested on a database of 35 faces from healthy Mexican individuals, obtained with a low cost structured light stereovision system. One of the objectives of this study, was to determine the statistical variability of a set of 19 face landmarks, which define facial features of the eyes, mouth, nose, cheeks and chin. To validate the reliability of the hand-extracted landmarks, a Technical Error of Measurement (TEM) analysis was performed. After the points were extracted, a rigid registration of the landmarks, to those from a reference head model, was applied by determining an optimal rigid transformation consisting of a unit quaternion and a translation vector obtained from the cross-covariance matrix. To obtain the mean landmarks set and the modes of variation, a Principal Components Analysis (PCA) based on the covariance matrix was employed. Finally, we approximated the average facial shape of the population under study, by deforming the reference head model through cage-based registration. In such approach, the high resolution model is attached to a rough mesh or cage, which encloses the detailed model, by using Mean Value Coordinates (MVC) each vertex of the detailed mesh is represented as a linear combination of the cage mesh vertices, which allow detail preserving deformation of the high polygonal mesh by displacement of the vertices from the coarse mesh, then the cage is iteratively deformed through Laplacian deformation in order to minimize the squared distance between corresponding landmarks. Besides working with points, and linear and angular measurements, the developed software also allows interactively to select paths and contours on the mesh surface, obtaining geodesic distances and areas; and even working on images. Finally, our work can be extended for the analysis of other complex anatomical structures, and potentially it may have other applications such as facial recognition on forensics, random face generation for *avateering* in virtual environments, and building compact 3D face databases.

1 INTRODUCCIÓN

En este trabajo se presenta el desarrollo de herramientas computacionales para el análisis de modelos de cabezas tridimensionales, mediante la construcción de interfaces que permiten de manera intuitiva la selección de los puntos de referencia que definen los rasgos faciales. Mediante el uso de técnicas de computación gráfica y visualización científica, se ofrece al usuario una mejor percepción de la geometría del objeto de estudio, facilitando de esta forma la interpretación de la información con el fin de representar de manera compacta los rasgos faciales, así como contar con métricas objetivas que permiten estudiar mejor las características morfológicas de los objetos. Además, se propone el uso de un modelo deformable de la cabeza, el cual puede ajustarse a cualquier configuración de los puntos de referencia, con el fin de obtener una forma promedio de la cabeza en la población de estudio y generar de manera aleatoria cabezas humanas que pertenecen a la misma población, o sea que siguen la misma distribución estadística.

1.1 ANTECEDENTES

Los puntos de referencia del rostro humano y su configuración espacial, son de utilidad para el reconocimiento facial y su caracterización morfológica. Con el uso extendido de los escáneres 3D y otros métodos de adquisición, se han desarrollado muchos algoritmos y metodologías para la extracción automática de puntos de referencia. En esta sección se analizan algunos métodos para la extracción automática de puntos de referencia, con el fin de identificar los parámetros que estos métodos emplean como discriminantes para la detección de esos puntos, así como los métodos estadísticos utilizados para el procesamiento de la información.

En (Ruiz and Illingworth, 2008) primero un conjunto de 14 puntos de referencia es colocado a mano sobre modelos de entrenamiento, que luego de ser alineados en sistema de referencia común, se realiza un análisis de componentes principales (PCA) obteniendo el promedio, y los valores y vectores propios de la población. Luego, cada punto de referencia es clasificado de acuerdo al índice de forma que se calcula en un parche centrado en el punto. Para encontrar los puntos de referencia en un rostro, se realiza primero una aproximación inicial utilizando el método iterativo del punto más cercano (ICP) (Besl and McKay, 1992), posteriormente se aplica un modelo activo de forma con la información estadística de los puntos de referencia y los índices de forma, para ajustar iterativamente los puntos de referencia sobre el modelo.

La curvatura de las superficies, como propiedad local, ha sido ampliamente utilizada como descriptor de forma, ya que tiene la ventaja de ser invariantes a transformaciones rígidas. En (Ansari et al., 2007) se presentó un modelo genérico del rostro que se ajusta sobre una imagen de profundidad (range image) 3D, utilizando la curvatura gaussiana para detectar la punta de la nariz y las esquinas internas de los ojos para realizar reconocimiento facial sobre una base de datos. En (Nair and Cavallaro, 2009) el índice de forma y la curvidad (*Curvedness*) son utilizados como características para la localización automática de puntos de referencia y segmentación facial, empleando para ello un modelo de distribución de puntos del rostro. En (Gupta et al., 2010) se utilizan los valores de curvatura gaussiana y curvatura media para detectar puntos de referencia sobre la nariz, los ojos y la boca en imágenes de profundidad 3D, mientras que en (Mahmood et al., 2013) el índice de forma se usa como descriptor para detectar la punta de la nariz sobre una nube de puntos en 3D del rostro.

En (Kaushik et al., 2010) primero se realiza una normalización del conjunto de puntos a través de una transformación afín para colocar la cabeza en una configuración estándar, haciendo coincidir el plano sagital con el plano yz, y la punta de la nariz con el origen. Luego se aplica un algoritmo para extraer las regiones de interés basado en una rejilla rectangular y las normales en cada celda, segmentando las regiones correspondientes a las cejas, nariz y labios. Finalmente, los puntos de referencia se extraen mediante un análisis de los mínimos y máximos locales en las coordenadas de las regiones segmentadas.

En (Whitmarsh et al., 2006) la extracción automática de puntos de referencia se realiza empleando un modelo deformable de un rostro a baja resolución, del cual se conocen los puntos de referencia. En primer lugar, se realiza un registro rígido del modelo como una primera aproximación utilizando el análisis de componentes principales, para normalizar la orientación de los modelos. Finalmente, el modelo deformable es ajustado a los datos escaneados utilizando el método iterativo del punto más cercano (ICP) (Besl and McKay, 1992).

Un enfoque similar se presenta en (Liang et al., 2013) en donde los puntos de referencia sobre un modelo utilizado como plantilla, son transferidos hacia otro modelo. Primero se aplica un proceso de normalización para que la cabeza esté centrada en el origen, y con el rostro mirando en la dirección del eje z. Posteriormente se realiza un registro deformable utilizando una serie de transformaciones que minimizan una función de energía a través de un proceso iterativo. En dicho enfoque la malla es deformada vértice a vértice como se detalla en (Allen et al., 2003).

En el trabajo presentado en (El-Hussuna, 2003) se realiza un análisis estadístico sobre modelos tridimensionales de rostros obtenidos mediante un escáner laser 3D, utilizando un conjunto de 64 puntos de referencia anatómicos colocados a mano sobre los modelos. Mediante un algoritmo de correspondencia densa se alinean los modelos utilizando pseudo-puntos de referencia tomando como parámetro la distancia euclidiana más corta. Mediante un análisis de componentes principales se caracterizaron los modos de variación para un conjunto de ocho escaneos de rostros de varones caucásicos de entre 25 y 35 años.

En (Scheenstra, 2005) se realizó un análisis sobre las distancias entre puntos de referencia para caracterizar la variación mediante una prueba de Fisher con el objetivo de hallar los puntos de referencia que son más adecuados para comparación facial. En (Enciso et al., 2003) se realizó un análisis antropométrico sobre datos tridimensionales adquiridos usando luz estructurada y se validaron las mediciones contra las realizadas utilizando un digitalizador 3D sobre un maniquí. Por otra parte en (Majid et al., 2005) se compara el uso de fotogrametría utilizando luz estructurada para la adquisición de rostros 3D con el escáner laser 3D para formar una bases de datos craneofacial en Malasia realizando pruebas sobre maniquíes.

En (Carnicky and Jr., 2006) se utilizó un sistema de adquisición de rostros 3D mediante luz estructurada para medir la variación entre rasgos, definidos por las distancias entre puntos de referencia. Se escanearon en total 25 rostros y se caracterizó la variación de las mediciones del ancho y altura de la boca, así como del ancho de la nariz para los varones y las mujeres dentro de la población estudiada.

Un análisis más completo se presentó en (López, 2015) en donde se utilizó un sistema de adquisición de bajo costo mediante estereovisión con luz estructurada para la adquisición de los modelos

tridimensionales del rostro y se realizó un análisis de componentes principales para caracterizar la correlación por género de los índices, distancias y ángulos entre puntos de referencia colocados a mano en un total de 35 personas de nacionalidad mexicana. En el presente trabajo se utilizó una versión mejorada de dicho sistema.

1.2 JUSTIFICACIÓN

Este proyecto es de utilidad para los antropólogos a fin de caracterizar los rasgos craneofaciales en grupos étnicos. En ciencia forense es de utilidad para la identificación de personas con base en sus rasgos craneofaciales sobre bases de datos. En proyectos con realidad virtual es posible crear avatares con características faciales específicas, exageradas o cambiantes, de acuerdo a la aplicación, por ejemplo, en estudios cognitivos o de psicofísica y en psicoterapia.

1.3 OBJETIVO

El objetivo general de este trabajo es la creación de un software para la extracción de puntos de referencia sobre modelos tridimensionales de la cabeza humana, con el fin de reunir de manera compacta, información sobre los rasgos faciales de la población mexicana, y proporcionar una herramienta de análisis con mayor información estadística que la normalmente utilizada con métodos manuales. Para ello se utilizarán técnicas de computación gráfica y visualización científica para desarrollar interfaces que, de manera intuitiva, permitan la visualización y selección de los puntos característicos en tiempo real; para ello se despliega sobre el modelo, información sobre las curvaturas e índice de forma, explotando así la información geométrica del modelo para brindar al usuario una mejor percepción de los rasgos del modelo. Adicionalmente se desarrolló un modelo deformable de la cabeza humana, para aproximar mediante el desplazamiento de los puntos de referencia, la forma de los rostros con pocos puntos de referencia, obteniéndose además la forma de la cabeza promedio, y pudiéndose generar de manera aleatoria, modelos de cabezas que siguen la misma distribución estadística que la población estudiada.

Objetivos específicos:

- Desarrollar herramientas de software para la extracción semiautomática de los puntos, medición de distancias, ángulos e índices de interés.
- Estimar y visualizar las curvaturas principales en mallas triangulares de los modelos.
- Realizar un análisis de componentes principales sobre el conjunto de puntos de diferentes cabezas.
- Desarrollar un modelo deformable de la cabeza humana usando como puntos de control los puntos de referencia.
- Aproximar con el modelo deformable la forma de los rostros, obtener la cabeza promedio y generar modelos aleatorios de cabezas humanas con base en la información estadística.

1.4 ORGANIZACIÓN DE LA TESIS

Este trabajo está organizado de la siguiente manera:

- En el capítulo 2 se expondrán los materiales utilizados en la realización de este trabajo, haciendo énfasis en los recursos computaciones, tanto hardware como software, así como los datos empleados.
- En el capítulo 3 se detallan los métodos utilizados en este trabajo, resaltando la descripción matemática y algorítmica de cada método.
- En el capítulo 4 se describe la implementación de las herramientas computacionales, desarrolladas en este trabajo, para la selección interactiva de los puntos de referencia, haciendo énfasis en la ingeniería de software.
- En el capítulo 5 se discuten los resultados de este trabajo, mostrando la evidencia respectiva, en forma de imágenes y gráficas, resaltando la naturaleza estadística de los datos obtenidos.
- En el capítulo 6 se resumen los resultados más relevantes, exponiendo las contribuciones más importantes de este trabajo.

2 MATERIALES

2.1 BASE DE DATOS DE MÉXICO

Se utilizó como población de estudio, un conjunto de mallas triangulares correspondientes a los modelos tridimensionales de las cabezas pertenecientes a 35 sujetos mexicanos. Estos datos fueron adquiridos en el Laboratorio de Imágenes, del Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET), de la Universidad Nacional Autónoma de México (UNAM), mediante la técnica de estereovisión por luz estructurada (López, 2015). El sistema de adquisición, compuesto por tres pares de cámaras, se ilustra en la (Figura 1).

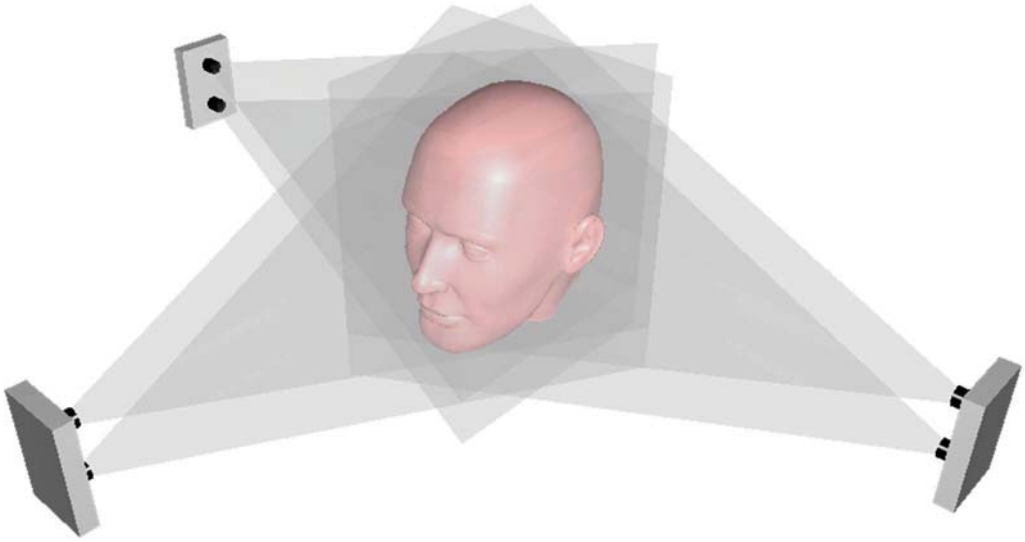


Figura 1. Sistema de adquisición con tres pares de cámaras.

Los sujetos mexicanos, hombres y mujeres, que conforman la base de datos, tienen en común las siguientes características:

- Ser amerindio o mestizo, es decir, que al menos haya un amerindio en sus tres generaciones anteriores.
- Ser adulto joven (20 a 40 años).
- No tener perforaciones, maquillaje o vello facial prominente al momento de la adquisición.
- No haber sufrido cirugías o fracturas en el rostro.

La base de datos está conformada por 35 archivos de mallas triangulares en formato VRML[1]. Para facilitar la manipulación de las mallas y preservar la información de los colores asociados a los vértices, los modelos fueron convertidos al formato PLY[2] utilizando el software MeshLab[3]. En algunos casos fue necesario eliminar manualmente componentes no conexos, que no formaban parte del rostro, así como triángulos degenerados, resultado de los artefactos del algoritmo de

[1] http://www.w3.org/MarkUp/VRML/
[2] http://www.dcs.ed.ac.uk/teaching/cs4/www/graphics/Web/ply.html
[3] http://meshlab.sourceforge.net/

reconstrucción 3D. Algunos de los modelos que forman parte de la base de datos se muestra en la (Figura 2), en donde se aprecia el patrón de luz estructurada en bandas usado para la reconstrucción 3D.

Figura 2. Algunos modelos de la base datos de modelos tridimensionales de cabezas de sujetos mexicanos utilizados en este trabajo. Las mallas de los modelos están formadas, en promedio, por 45555 vértices y 97614 triángulos

2.2 MODELO DE REFERENCIA DE LA CABEZA HUMANA

Con el propósito de obtener un modelo de la cabeza humana de referencia en este estudio, se utilizó el software MakeHuman[1], el cual forma parte de un proyecto de código abierto, escrito en lenguaje Python, para la creación de humanos realistas en 3D. Este software emplea interpolación de forma para el modelado paramétrico del cuerpo humano, permitiendo variar rasgos como edad, género, etnicidad, musculatura, peso, entre otros. El mallado generado por MakeHuman está compuesto en su totalidad por cuadriláteros, y la topología está optimizada para operaciones de subdivisión de mallas y animación. Con este software se generó un modelo del cuerpo humano con rasgos neutros sintéticos. Las unidades del modelo fueron establecidas en milímetros, y se exportó la malla de alta resolución en el formato COLLADA[2] para su posterior edición (Figura 3).

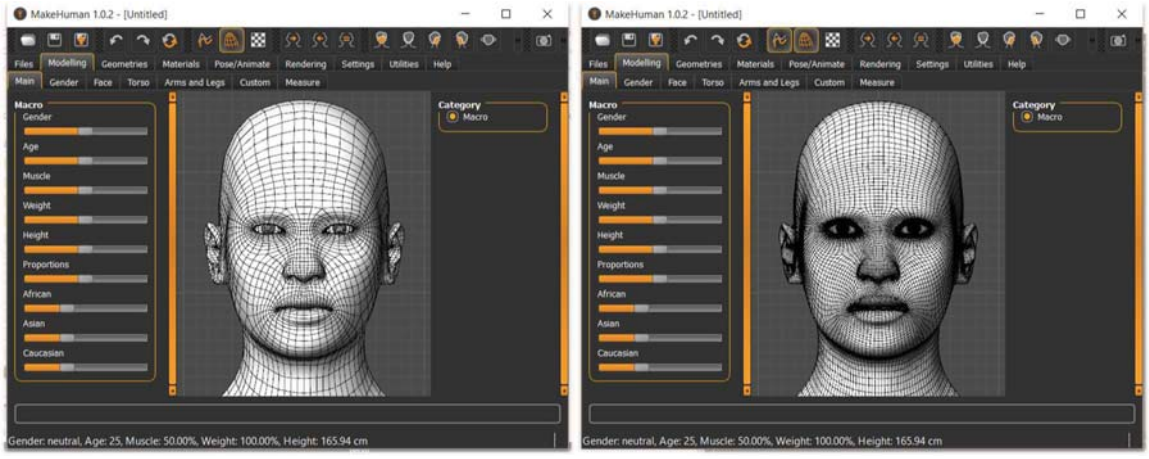


Figura 3. El modelo original de baja resolución está formado por 14444 vértices y 14398 cuadriláteros, mientras que el modelo de alta resolución tiene 57678 vértices y 57592 cuadriláteros.

[1] www.makehuman.org/
[2] https://www.khronos.org/collada/

El modelo de cuerpo completo fue posteriormente editado en el software Autodesk Maya[1], para extraer la región de la cabeza, tomando como referencia la mitad de la longitud del cuello. Los globos oculares del modelo fueron retirados, y se cerró la abertura del cuello. Finalmente se trasladó el centroide del modelo hacia el origen y se triangularon las caras. La malla resultante es cerrada y conexa (Figura 4).

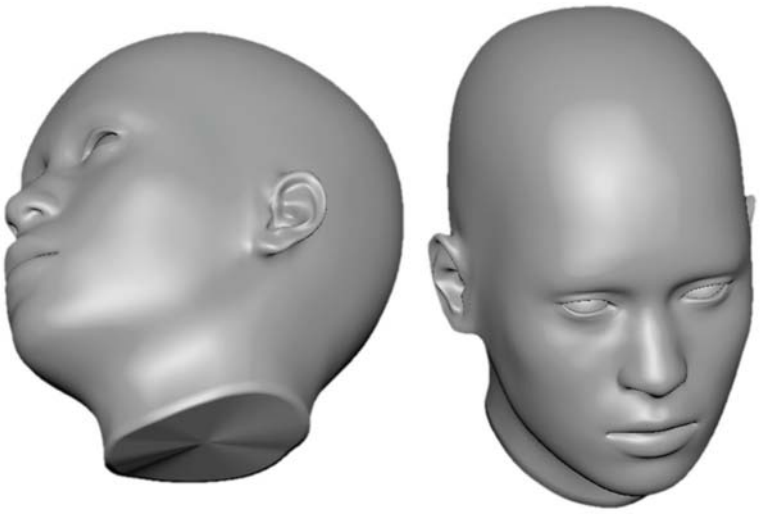


Figura 4. El modelo editado está formado por 17680 vértices y 35356 triángulos.

2.3 PUNTOS DE REFERENCIA

Un punto de referencia (landmark[2]) es un punto sobre cada objeto dentro de una población que coinciden entre sí (Dryden and Mardia, 1998). Pueden clasificarse como:

- Puntos de referencia anatómicos: Puntos asignados por un experto que corresponden a características biológicas en organismos.
- Puntos de referencia matemáticos: Puntos localizados sobre un objeto de acuerdo a propiedades geométricas calculadas numéricamente.
- Pseudo-puntos de referencia: Puntos construidos sobre un objeto sobre las líneas entre puntos de referencia anatómicos o matemáticos.
- Puntos de referencia etiquetados: Puntos de referencia asociados con una etiqueta la cual es utilizada para identificar puntos de referencias coincidentes.

Otra clasificación de los puntos de referencia (Bookstein, 1997):

- Tipo I: Uniones entre tejidos y huesos.
- Tipo II: Propiedades locales tal como las curvaturas máximas.
- Tipo III: Puntos extremos o puntos de referencia definidos arbitrariamente.

[1] http://www.autodesk.mx/products/maya/overview

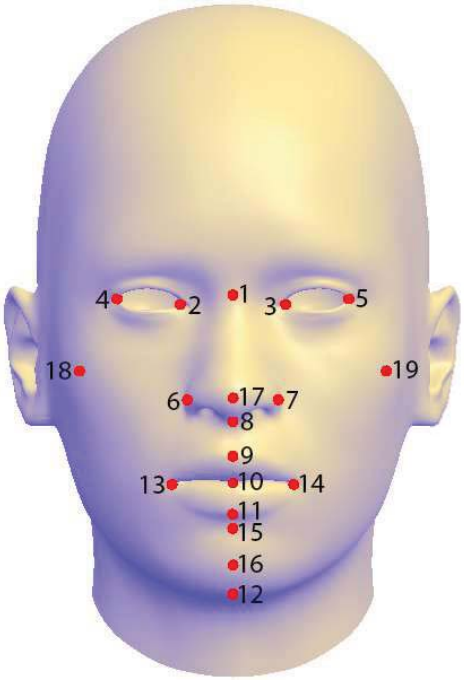
[2] Otros términos en español son puntos somatométricos o marcas fiduciarias, en este trabajo se traducirá como puntos de referencia, por ser la traducción literal del término en inglés.

2.3.1 Puntos de referencia en este trabajo

Se utilizarán los mismos puntos que en (López, 2015) utilizando las definiciones de (Kolar and Salter, 1997). Los puntos se enlistan en la (Tabla 1) y su localización se detalla en la (Figura 5).

Tabla 1. Puntos de referencia utilizados en este trabajo.

Nombre	Símbolo	Definición
Nasion	n	El punto medio de la sutura nasal-frontal.
Endocanthion	en	Las esquinas internas de las fisuras de los ojos.
Exocanthion	ex	Las esquinas externas de las fisuras de los ojos.
Alare	al	El punto más lateral del ala nasal.
Subnasale	sn	La unión entre la división de los orificios nasales y el labio superior.
Labiale superius	ls	El punto medio del borde bermellón del labio superior.
Stomion	sto	El punto medio de la fisura labial cuando los labios están cerrados.
Labiale inferius	li	El punto medio del borde bermellón del labio inferior.
Gnathion	gn	El punto más bajo de la mandíbula.
Cheilion	ch	La esquina externa de la boca.
Sublabiale	sl	El punto más profundo entre el labio inferior y el mentón visto de perfil.
Pogonion	pg	El punto más prominente de la barbilla.
Pronasale	prn	El punto más prominente de la punta de la nariz.
Zygion	zy	El punto más lateral sobre el arco cigomático.



1. Nasion (n)
2. Endocanthion$_{right}$ (en$_r$)
3. Endocanthioin$_{left}$ (en$_l$)
4. Exocanthion$_{right}$ (ex$_r$)
5. Exocanthioin$_{left}$ (ex$_l$)
6. Alare$_{right}$ (al$_r$)
7. Alare$_{left}$ (al$_l$)
8. Subnasale (sn)
9. Labiale superius (ls)
10. Stomion (sto)
11. Labiale inferius (li)
12. Gnathion (gn)
13. Cheilion$_{right}$ (ch$_r$)
14. Cheilion$_{left}$ (ch$_l$)
15. Sublabiale (sl)
16. Pogonion (pg)
17. Pronasale (prn)
18. Zygion$_{right}$ (zy$_r$)
19. Zygion$_{left}$ (zy$_l$)

Figura 5. Puntos de referencia utilizados en este trabajo.

2.4 HARDWARE

Para la implementación y desarrollo de las herramientas computacionales necesarias para la realización de este trabajo se utilizó una computadora portátil con las siguientes características:

- Procesador: Intel® Core™ i7-4710 HQ (6M Cache, 2.5 GHz)
- Sistema Operativo: Windows 10
- Gráficos: NVIDIA GeForce GTX 860M GDDR5 2GB
- Memoria RAM: 8 GB
- Disco duro: 1 TB

Como interfaz háptica se utilizó un dispositivo PHANTOM OMNI[1] con las siguientes características:

- Espacio de trabajo: 160 x 120 x 70 mm
- Peso del dispositivo: 3 lb 15 oz
- Resolución: 0.055 mm
- Fuerza máxima: 3.3 N
- Rigidez: eje x 1.26 N/mm, eje y 2.31 N/mm, eje z 1.02 N/mm
- Inercia: 45 g
- Interface: IEEE-1394 Puerto Firewire de 6 a 6 pines
- Sensores de posición: x, y, z, yaw, pitch, roll
- Retroalimentación de fuerza: x, y, z

2.5 SOFTWARE

En el desarrollo de este trabajo se utilizaron los siguientes recursos computacionales:

Software:

- Microsoft Visual Studio 2013[2]. Entorno de desarrollo integrado, utilizado para la implementación en lenguaje C++ de los proyectos.
- MeshLab[3]. Software para la edición de mallas 3D, utilizado para la edición y conversión de formatos de los modelos 3D.
- MakeHuman[4]. Software para la generación de modelos 3D del cuerpo humano, utilizando modelado paramétrico, se utilizó para obtener un modelo de referencia de la cabeza humana.
- Autodesk Maya 2016[5]. Software para modelado y animación 3D, utilizado para la edición y renderizado de modelos 3D.
- MATLAB R2015a[6]. Entorno de programación para el procesamiento de datos matriciales, utilizado para el análisis estadístico de los datos, y generación de algunas de las gráficas mostradas en este trabajo.

[1] http://www.dentsable.com/haptic-phantom-omni.htm
[2] https://www.visualstudio.com/
[3] http://meshlab.sourceforge.net/
[4] http://www.makehuman.org
[5] http://www.autodesk.mx/products/maya/overview
[6] http://www.mathworks.com/products/matlab/

- Visual Paradigm for UML 10.1. Entorno para el modelado de diagramas UML, utilizado para la edición de los diagramas de clases.

Bibliotecas de programación:

- Qt 5.5[1]. Es una biblioteca escrita en lenguaje C++ para la programación de aplicaciones multiplataforma con interfaces gráfica de usuario. Para este trabajo se utilizó la versión Open Source para Windows de 32 bits compatible con Visual Studio 2013.
- OpenGL 4.0[2]. Biblioteca en lenguaje C para la programación de gráficos en 3D.
- GLEW[3]. Biblioteca en lenguaje C para administrar las extensiones de OpenGL.
- GLM[4]. Biblioteca en lenguaje C++ de algebra matricial, con estructuras de datos y métodos similares a los implementados por el lenguaje de sombreado GLSL de OpenGL.
- Eigen[5]. Biblioteca de algebra lineal, utilizada para la descomposición en vectores y valores propios.
- ASSIMP[6]. Biblioteca para cargar archivos de mallas en 3D.
- DevIL[7]. Biblioteca para importar y exportar imágenes, se utilizó para importar las imágenes utilizadas como textura por los modelos, así como para generar capturas de pantalla.
- Sensable OpenHaptics Toolkit[8]. Biblioteca en lenguaje C para la integración y control de dispositivos hápticos.

[1] http://www.qt.io/
[2] https://www.opengl.org/
[3] http://glew.sourceforge.net/
[4] http://glm.g-truc.net/
[5] http://eigen.tuxfamily.org/
[6] assimp.sourceforge.net/
[7] http://openil.sourceforge.net/
[8] http://www.geomagic.com/es/products/open-haptics/overview/

3 MÉTODOS

3.1 VISUALIZACIÓN CIENTÍFICA

En esta sección se describirán los métodos empleados para la visualización de los objetos 3D, describiendo de forma breve los sistemas de coordenadas y transformaciones de vértices durante el proceso de proyección a través de la cámara virtual, los modelos de sombreado e iluminación durante el proceso de renderizado, y finalmente se abordará el uso de mapas de colores para la visualización de datos sobre el mallado.

3.1.1 Cámara virtual

Durante las etapas de procesamiento de vértices para la visualización, se tienen diferentes sistemas de coordenadas en cada etapa y transformaciones entre éstas, representadas mediante matrices de transformación en coordenadas homogéneas (Figura 6).

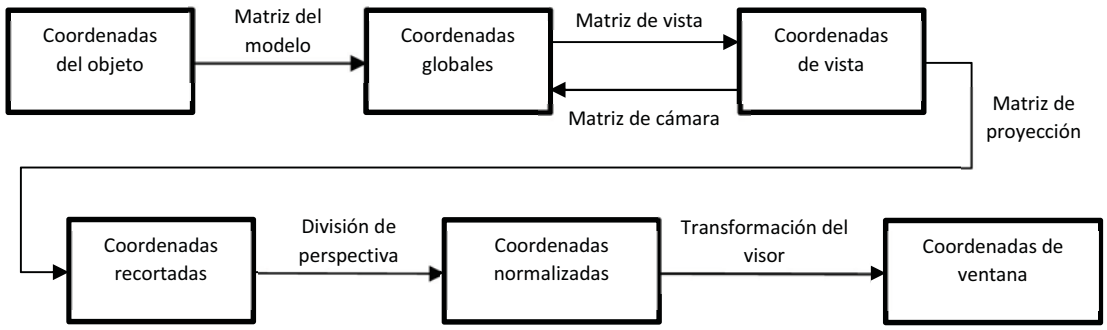


Figura 6. Sistemas de coordenadas y transformaciones entre sistemas de coordenadas.

3.1.1.1 Coordenadas del objeto

Es el sistema de coordenadas local del objeto con respecto un punto de referencia dentro del mismo (típicamente el centro geométrico o de masa); sirve para referenciar localmente las posiciones de los vértices tal y como son originados.

3.1.1.2 Coordenadas globales

Son las coordenadas del objeto con respecto al sistema de coordenadas del mundo, una vez que este es posicionado, orientado y escalado dentro de la escena, luego de aplicar la matriz de modelado. Para linealizar las transformaciones, se utilizan coordenadas homogéneas $\begin{pmatrix} x & y & z & w \end{pmatrix}$ con $w \neq 0$.

$$\begin{pmatrix} x & y & z & w \end{pmatrix}^{\mathrm{T}}_{\mathrm{world}} = \mathbf{M}_{\mathrm{model}} \begin{pmatrix} x & y & z & w \end{pmatrix}^{\mathrm{T}}_{\mathrm{object}} \tag{1}$$

3.1.1.3 Coordenadas de vista

Son las coordenadas con respecto a la posición, orientación y escala de la cámara virtual. La cámara se localiza en el origen y mirando hacia la dirección $-z$, y son los objetos los que son transformados por la matriz inversa de la cámara.

$$\begin{pmatrix} x & y & z & w \end{pmatrix}_{\text{eye}}^{\text{T}} = \mathbf{M}_{\text{view}} \begin{pmatrix} x & y & z & w \end{pmatrix}_{\text{world}}^{\text{T}} \tag{2}$$

$$\begin{pmatrix} x & y & z & w \end{pmatrix}_{\text{eye}}^{\text{T}} = \mathbf{M}_{\text{camera}}^{-1} \begin{pmatrix} x & y & z & w \end{pmatrix}_{\text{world}}^{\text{T}} \tag{3}$$

3.1.1.4 Coordenadas recortadas

Se llaman así porque los vértices transformados son delimitados por el volumen de visualización (*frustum*) aplicando la matriz de proyección.

$$\begin{pmatrix} x & y & z & w \end{pmatrix}_{\text{clip}}^{\text{T}} = \mathbf{M}_{\text{projection}} \begin{pmatrix} x & y & z & w \end{pmatrix}_{\text{eye}}^{\text{T}} \tag{4}$$

3.1.1.5 Coordenadas normalizadas del dispositivo

Se obtienen dividiendo las coordenadas de recorte por la componente w, las coordenadas resultantes están en el intervalo $\left[-1,\ 1\right]$ en cada eje.

$$\begin{pmatrix} x & y & z \end{pmatrix}_{\text{NDC}}^{\text{T}} = \begin{pmatrix} x/w & y/w & z/w \end{pmatrix}_{\text{clip}}^{\text{T}} \tag{5}$$

3.1.1.6 Coordenadas de ventana

Las coordenadas normalizadas del dispositivo de despliegue gráfico son mapeadas a las dimensiones físicas del visor, para después ser pasadas al proceso de rasterización.

$$\begin{pmatrix} x & y & z \end{pmatrix}_{\text{window}}^{\text{T}} = \left(\frac{w}{2}x + \left(x_0 + \frac{w}{2} \right) \quad \frac{h}{2}y + \left(y_0 + \frac{h}{2} \right) \quad \frac{f-n}{2}z + \frac{f+n}{2} \right)_{\text{NDC}}^{\text{T}} \tag{6}$$

En donde $\left(x_0, y_0 \right)$ son las coordenadas en pixeles de la esquina inferior izquierda del visor de ancho w y alto h, y n, f son correspondientemente la distancia a los planos de recorte (clipping) cercano y lejano.

Para hacer intuitivo al usuario el proceso de navegación dentro de la escena 3D se propuso descomponer la matriz de cámara $\mathbf{M}_{\text{camera}}$ en una serie de transformaciones de traslación, rotación y escala, con el fin de obtener las siguientes características:

- La cámara mira hacia un punto de interés o pivote $\mathbf{p} = \left(p_{\mathcal{X}}, p_{\mathcal{Y}}, p_{\mathcal{Z}} \right)$ en el mundo.
- La cámara está alejada del pivote una distancia d sobre el eje óptico.
- La cámara gira en torno al pivote, su orientación está definida por los ángulos de Euler $\mathbf{r} = \left(r_{\mathcal{X}}, r_{\mathcal{Y}}, r_{\mathcal{Z}} \right)$.
- La cámara puede ampliar o reducir objetos en la escena (zoom), mediante un factor de escalamiento isotrópico ζ.

Sea $\mathbf{T}\left(x, y, z \right)$ la matriz homogénea de traslación definida como:

$$\mathbf{T}(x,y,z)=\begin{pmatrix}1 & 0 & 0 & x\\ 0 & 1 & 0 & y\\ 0 & 0 & 1 & z\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{7}$$

Sea $\mathbf{R}_{\mathcal{X}}(\theta)$ la matriz de rotación en torno al eje $\mathcal{X}$ un ángulo θ en sentido anti-horario:

$$\mathbf{R}_{\mathcal{X}}(\theta)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{8}$$

Sea $\mathbf{R}_{\mathcal{Y}}(\phi)$ una matriz de rotación en torno al eje $\mathcal{Y}$ un ángulo ϕ en sentido anti-horario:

$$\mathbf{R}_{\mathcal{Y}}(\phi)=\begin{pmatrix}\cos\phi & 0 & \sin\phi & 0\\ 0 & 1 & 0 & 0\\ -\sin\phi & 0 & \cos\phi & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{9}$$

Sea $\mathbf{R}_{\mathcal{Z}}(\psi)$ una matriz de rotación en torno al eje $\mathcal{Z}$ un ángulo ψ en sentido anti-horario:

$$\mathbf{R}_{\mathcal{Z}}(\psi)=\begin{pmatrix}\cos\psi & -\sin\psi & 0 & 0\\ \sin\psi & \cos\psi & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{10}$$

Sea $\mathbf{S}(s)$ una matriz de escalamiento isotrópico definida como:

$$\mathbf{S}(s)=\begin{pmatrix}s & 0 & 0 & 0\\ 0 & s & 0 & 0\\ 0 & 0 & s & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{11}$$

La matriz de cámara está dada por:

$$\mathbf{M}_{\text{camera}}=\mathbf{T}(p_{\mathcal{X}},p_{\mathcal{Y}},p_{\mathcal{Z}})\mathbf{S}(1/\zeta)\mathbf{R}_{\mathcal{Z}}(r_{\mathcal{Z}})\mathbf{R}_{\mathcal{Y}}(r_{\mathcal{Y}})\mathbf{R}_{\mathcal{X}}(r_{\mathcal{X}})\mathbf{T}(0,0,d) \tag{12}$$

Luego la matriz de vista es

$$\mathbf{M}_{\text{view}}=\mathbf{M}_{\text{camera}}^{-1} \tag{13}$$

Para la matriz de proyección $\mathbf{M}_{\text{projection}}$ existen dos casos (Ahn, 2008).

3.1.1.7 Proyección perspectiva

La matriz de proyección perspectiva está dada por:

$$\mathbf{M}_{\text{projection}} = \begin{pmatrix} \frac{e}{a} & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & \frac{-f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{14}$$

En donde a es el aspecto o proporción entre el ancho y el alto del visor, n, f son respectivamente la distancia a los planos de recorte cercano y lejano, y e es la longitud focal sin unidades o normalizada (Szeliski, 2010), definida como:

$$e = \frac{1}{\tan\left(\frac{\text{fovy}}{2}\right)} \tag{15}$$

En donde fovy es el campo de visión vertical.

3.1.1.8 Proyección ortográfica

La proyección ortográfica está dada por

$$\mathbf{M}_{\text{projection}} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

En donde l, r, b, t, n, f son respectivamente las coordenadas x de los planos de recorte izquierdo y derecho, las coordenadas y de los planos de recorte inferior y superior, y coordenadas z de los planos de recorte cercano y lejano.

Para que el plano sobre el pivote coincida en ambas proyecciones, los valores son ajustados a

$$\mathbf{M}_{\text{projection}} = \begin{pmatrix} \frac{e}{ad} & 0 & 0 & 0 \\ 0 & \frac{e}{d} & 0 & 0 \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{17}$$

3.1.2 Sombreadores (Shaders)

Los sombreadores, más comúnmente conocidos como shaders, son programas definidos por el usuario dentro del pipeline de renderizado de OpenGL (Figura 7), para determinar la apariencia final de los objetos virtuales. Antes de hablar de los modelos de iluminación utilizados en este trabajo se analizará brevemente las etapas de renderizado.

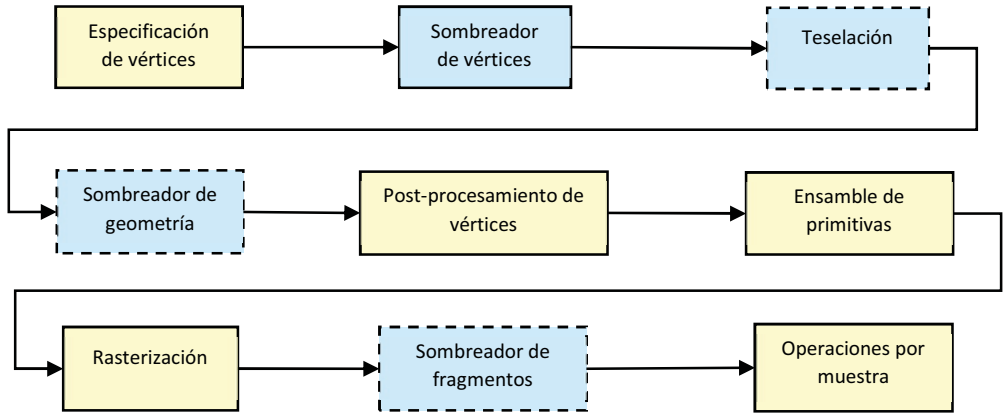


Figura 7. Diagrama del pipeline de renderizado de OpenGL. Las etapas en color azul con programables, las etapas programables señaladas con líneas discontinuas son opcionales[1].

3.1.2.1 Especificación de vértices

En esta etapa la aplicación establece la lista ordenada de vértices que representan las primitivas y las envía al *pipeline*. Los vértices son representados por un conjunto de atributos arbitrarios cuyo significado depende de la forma en que se procesen en las etapas siguientes. Son atributos, por ejemplo, la posición y el vector normal.

3.1.2.2 Sombreador de vértices

En esta etapa se procesan individualmente los vértices mediante un programa arbitrario definido por el usuario. Esta etapa incluye, pero no está limitada, a la transformación de las posiciones de los vértices en coordenadas locales a coordenadas recortadas. Los sombreadores de vértices pueden tener salidas definidas por el usuario.

Sean $\mathbf{v}_0$, $\mathbf{n}_0$ la posición y vector normal correspondientemente del vértice antes de ser transformado. La posición final $\mathbf{v}$ y orientación final del vector normal $\mathbf{n}$ están dados por:

$$\begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} = \mathbf{M}_{\text{ftransform}} \begin{pmatrix} \mathbf{v}_0 \\ 1 \end{pmatrix} \tag{18}$$

$$\begin{pmatrix} \mathbf{n} \\ 0 \end{pmatrix} = \mathbf{M}_{\text{normal}} \begin{pmatrix} \mathbf{n}_0 \\ 0 \end{pmatrix} \tag{19}$$

En donde la matriz de transformación homogénea $\mathbf{M}_{\text{ftransform}}$ combina las matrices de modelado, de vista y de proyección:

[1] https://www.opengl.org/wiki/Rendering_Pipeline_Overview

$$\mathbf{M}_{\text{ftransform}} = \mathbf{M}_{\text{projection}}\mathbf{M}_{\text{view}}\mathbf{M}_{\text{model}} \tag{20}$$

La matriz de las normales $\mathbf{M}_{\text{normal}}$ se calcula como:

$$\mathbf{M}_{\text{normal}} = \left(\left(\mathbf{M}_{\text{view}}\mathbf{M}_{\text{model}}\right)^{-1}\right)^{\text{T}} \tag{21}$$

Si la matriz de modelado y de vista contienen solo escalamiento isotrópico, la matriz de las normales es simplemente:

$$\mathbf{M}_{\text{normal}} = \mathbf{M}_{\text{view}}\mathbf{M}_{\text{model}} \tag{22}$$

Finalmente se obtiene el vector normal unitario:

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|} \tag{23}$$

3.1.2.3 Teselación

En esta etapa las primitivas son subdivididas con el fin de obtener una malla más suave o con más detalles (Figura 8). Esta etapa es opcional y no entraremos en detalles ya que no se emplea en este trabajo.

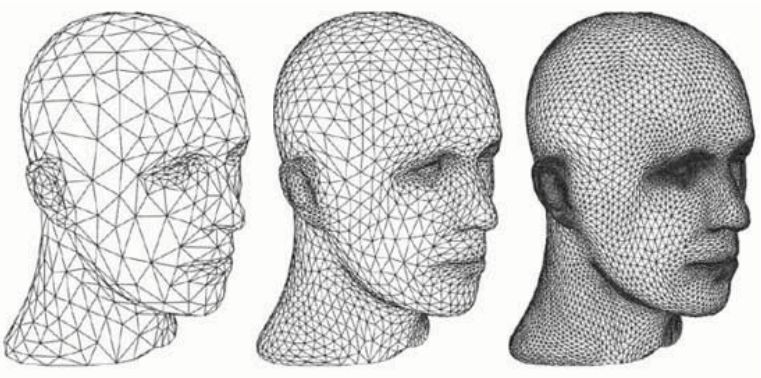


Figura 8. Ejemplo de teselación de una malla triangular, con uno y dos niveles de subdivisión. Extraído de http://www.mantis.es/directx11.

3.1.2.4 Sombreador de geometría

Esta etapa procesa las primitivas de entrada y genera cero o más primitivas, además puede convertir las primitivas en otros tipos. Esta etapa es opcional y no entraremos en detalles ya que no se emplea en este trabajo.

3.1.2.5 Post-procesamiento de vértices

En esta etapa las primitivas se recortan, esto es, son descompuestas en otras primitivas de forma tal que estén contenidas dentro el volumen de visualización. Las posiciones de los vértices son transformadas mediante la división de perspectiva y la transformación del visor.

3.1.2.6 Ensamble de primitivas

En esta etapa se recolectan los datos de los vértices de las etapas anteriores y se ensamblan para formar secuencias de primitivas. Las primitivas que no están contenidas dentro del volumen de visualización o cuya faceta esté invertida, son descartadas.

3.1.2.7 Rasterización

Las primitivas son descompuestas en fragmentos, cada fragmento contiene atributos de posición en coordenadas de ventana y la lista de datos arbitrarios provenientes de las etapas anteriores, obtenidos mediante interpolación de los vértices que forman las primitivas.

3.1.2.8 Sombreador de fragmentos

En esta etapa se define el color final y el valor de profundidad que son escritos en los búferes de color y de profundidad correspondientemente. En esta etapa es en donde se aplican los diferentes modelos de iluminación. Esta etapa es opcional, si no se especifica, solo el búfer de profundidad es escrito con los valores de profundidad obtenidos en la etapa anterior.

3.1.2.9 Operaciones sobre fragmentos

Los datos de salida de los fragmentos pasan por el test de profundidad para descartar fragmentos ocluidos y finalmente se aplican las operaciones de mezclado de color, para efectos de transparencias.

3.1.3 Modelos de sombreado

En graficación por computadora, la palabra sombreado se refiere al proceso de determinar el color de los fragmentos originados a partir de una superficie rasterizada, usualmente aplicando un modelo de iluminación. Existen tres modelos de sombreado que corresponden básicamente a sombreado por polígono, sombreado por vértice y sombreado por fragmento (Figura 9).

3.1.3.1 Modelo de sombreado de Warnock

También conocido como sombreado plano, utiliza las normales por polígono para calcular el color aplicando el modelo de iluminación, luego ese color es utilizado en todos los fragmentos del polígono (Figura 9 a).

3.1.3.2 Modelo de sombreado de Gouraud

Utiliza las normales por vértice para el cálculo de los colores en cada vértice aplicando el modelo de iluminación, luego los colores de los fragmentos en el interior son obtenidos mediante interpolación bilineal (Gouraud, 1971). Este método presenta el artefacto de acentuar las discontinuidades en las aristas (Figura 9 b).

3.1.3.3 Modelo de sombreado de Phong

Este modelo de sombreado interpola bilinealmente las normales por vértice al interior del polígono rasterizado (Phong, 1975), luego el color de cada fragmento se determina aplicando el modelo de iluminación utilizando la normal interpolada (Figura 9 c).

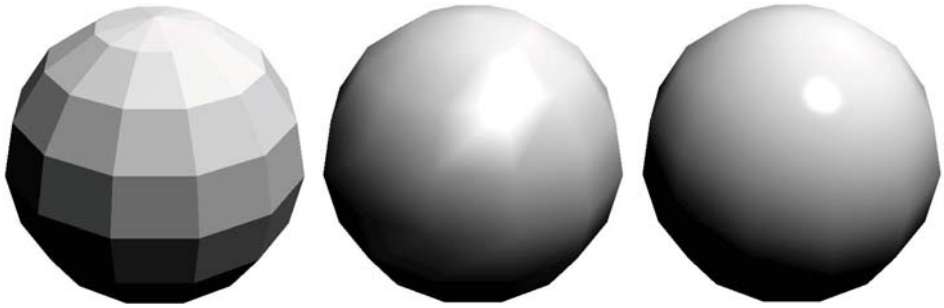


Figura 9. Distintos modelos de sombreado aplicados al mismo objeto con las mismas propiedades materiales: sombreado por polígono, sombreado de Warnock o sombreado plano (a), sombreado por vértice o sombreado de Gouraud (b), y sombreado por fragmento o sombreado de Phong (c).

3.1.4 Modelos de iluminación

Los modelos de iluminación determinan la interacción entre los materiales y las fuentes de luz para determinar la intensidad de iluminación en cada punto de la escena.

3.1.4.1 Modelo de iluminación de Lambert

Es el modelo de iluminación más simple basado en la ley de coseno de Lambert. La intensidad de la luz reflejada es proporcional al coseno del ángulo entre el vector normal de la superficie y la dirección de la luz incidente, simulando un difusor perfecto (Whitted, 2005). A las superficies que reflejan o emiten luz en todas direcciones y cuya intensidad varía de acuerdo a la ley del coseno de Lambert se le conoce como superficies lambertianas (Figura 10).

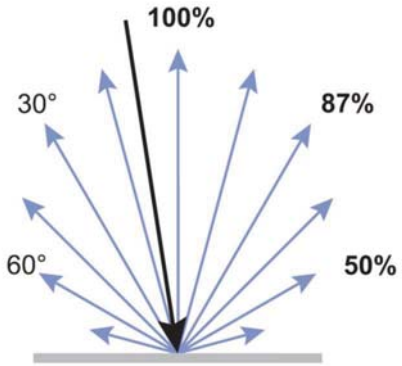


Figura 10. Reflexión en una superficie lambertiana. Extraído de (Taylor, 2000).

El modelo de iluminación de Lambert es una aproximación burda del efecto físico de reflexión y no considera las propiedades especulares del material, no obstante, genera superficies mate u opacas de apariencia razonable. La intensidad de iluminación I está dada por:

$$I = I_\alpha + k_d \sum_j \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}_j \tag{24}$$

En donde I_α es la intensidad de la luz ambiente, k_d es la constante de reflexión difusa del material, $\hat{\mathbf{n}}$ es el vector normal de la superficie y $\hat{\mathbf{l}}_j$ es el vector en la dirección de la fuente de luz j. Es

importante mencionar que sólo se suman las contribuciones positivas de los productos escalares $\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}_j$. Un ejemplo de sombreado usando el modelo de iluminación de Lambert se muestra en la (Figura 11 a).

3.1.4.2 Modelo de iluminación de Phong

Es un modelo empírico simplificado para iluminar puntos de una escena, considerando la posición del observador y las propiedades especulares del objeto (Phong, 1975). El modelo de Phong asume que las fuentes de luz y el observador están en el infinito y considera que la luz reflejada por un objeto puede ser de tres tipos:

- Luz ambiental: Proviene de todas las direcciones e ilumina todas las caras del objeto por igual.
- Luz difusa: proviene de una dirección, pero de refleja en todas direcciones.
- Luz especular: proviene de una dirección y se refleja sólo en una dirección.

La intensidad de iluminación I en un punto se calcula como la suma de los tres tipos de iluminaciones (Whitted, 2005):

$$I = I_\alpha + k_d \sum_j \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}_j + k_s \sum_j \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}'_j \right)^n \tag{25}$$

En donde I_α es la reflexión debido a la luz ambiental, k_d es la constante de reflexión difusa del material, $\hat{\mathbf{n}}$ es el vector normal de la superficie, $\hat{\mathbf{l}}_j$ es el vector en la dirección de la fuente de luz j, k_s es el coeficiente de reflexión especular, $\hat{\mathbf{l}}'_j$ es el vector en la dirección intermedia entre el observador y la fuente de luz j, y n es un exponente que depende de la brillantez (shininess) de la superficie. Es importante mencionar que sólo se suman las contribuciones positivas de los productos escalares $\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}_j$, y de que sólo se toman en consideración para aplicar el exponente los valores positivos del producto $\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}'_j$. El efecto de usar distintos valores de exponente en el modelo de iluminación de Phong se muestra en la (Figura 11 b-d).

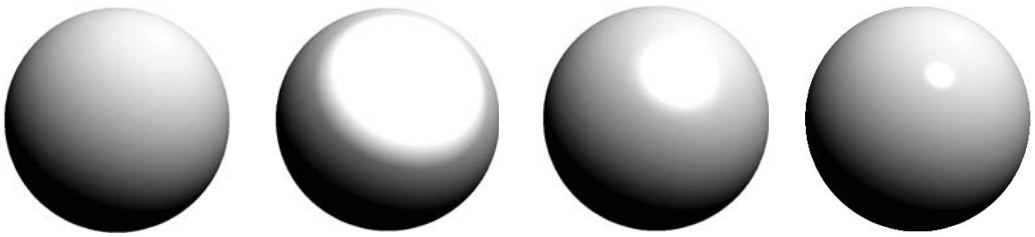


Figura 11. Esferas sombreadas usando el modelo de iluminación Lambert (a) y usando el modelo de iluminación de Phong con exponente de 1 (b), 10 (c) y 100 (d).

3.1.4.3 Modelo de Iluminación de Gooch

Introducido por (Gooch et al., 1998), es un modelo de iluminación no foto-realista inspirado en ilustraciones técnicas realizadas a mano por artistas. Está basado en la percepción de la temperatura

del color combinando cambios de luminancia y de matiz para transmitir la noción de profundidad en las superficies acentuando los detalles. La temperatura del color se define como cálidos (rojo, anaranjado y amarillo), fríos (azul, violeta y verde) o templados (rojo-violeta y amarillo-verde). La pista de profundidad viene dada por la percepción de que los colores fríos se alejan mientras que los colores cálidos avanzan. El color RGB a ser desplegado I está dado por:

$$I = \left(\frac{1 + \hat{\mathbf{l}} \cdot \hat{\mathbf{n}}}{2} \right) \mathbf{k}_{\text{cool}} + \left(1 - \frac{1 + \hat{\mathbf{l}} \cdot \hat{\mathbf{n}}}{2} \right) \mathbf{k}_{\text{warm}} \tag{26}$$

En donde $\hat{\mathbf{l}}$ es el vector unitario en la dirección de la fuente de luz y $\hat{\mathbf{n}}$ es el vector normal a la superficie, y

$$\mathbf{k}_{\text{cool}} = \mathbf{k}_{\text{blue}} + \alpha \mathbf{k}_d \tag{27}$$

$$\mathbf{k}_{\text{warm}} = \mathbf{k}_{\text{yellow}} + \beta \mathbf{k}_d \tag{28}$$

En donde $\mathbf{k}_{\text{blue}} = (0, 0, b)$, $\mathbf{k}_{\text{yellow}} = (y, y, 0)$, y $\mathbf{k}_d$ es el color difuso del material. Los valores de b e y determinan la intensidad del cambio de temperatura del color, y los valores α y b determinan la contribución del color del objeto y la intensidad del cambio de iluminación (Figura 12). Comúnmente a los objetos sombreados utilizando el modelo de iluminación de Gooch se les agregan las líneas de contorno para una mejor percepción de la forma (Figura 13).

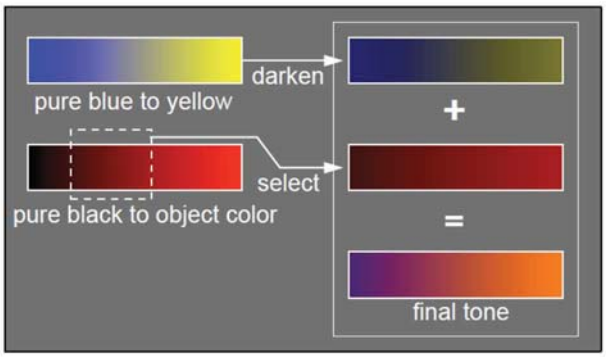


Figura 12. El tono final para un objeto rojo se obtiene sumando los tonos del azul al amarillo y del rojo oscuro al rojo. Extraído de (Gooch et al., 1998).

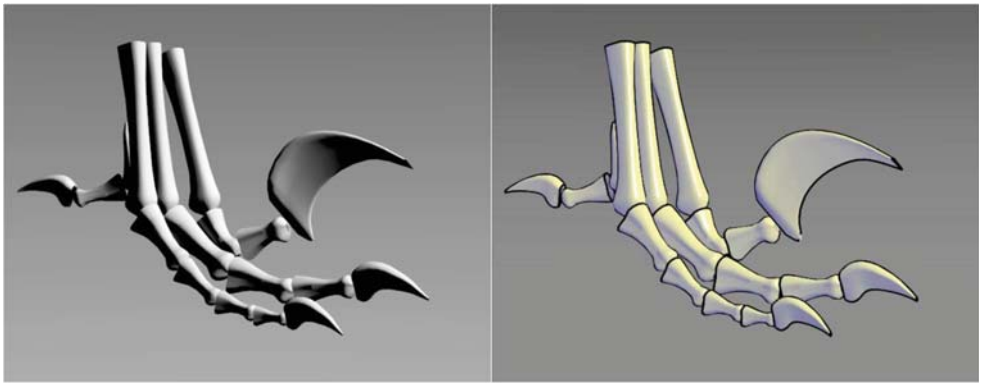


Figura 13. (Izquierda) Modelo de iluminación de Lambert y (derecha) modelo de iluminación de Gooch. Extraído de (Gooch et al., 1998).

3.1.5 Mapas de Color

Los mapas de color son utilizados para representar visualmente los datos, uno de los más utilizados en visualización científica es el denominado mapa de color arcoíris, el cual está basado en el orden de los colores en el espectro de luz visible (Figura 14) y es el más común en la literatura (Borland and Taylor, 2007).

Figura 14. Mapa de color arcoíris. Extraído de (Moreland, 2009)

En (Brewer, 2004) se dividen los mapas de colores en tres clases: cualitativo, secuencial y divergente. Los mapas de color cualitativos son usados para representar una colección discreta de clases sin un orden. Los mapas de colores secuenciales sirven para representar datos escalares partiendo de un color saturado y disminuyendo la saturación e incrementando la luminancia conforme se incrementa el valor representado. Los mapas de color divergente sirven para representar datos escalares con un valor significativo en la media pasando de un color saturado a otro color saturado (Figura 15).

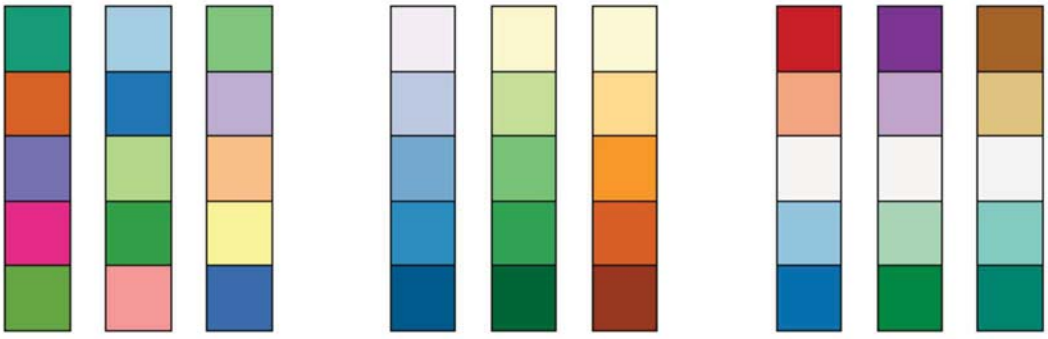


Figura 15. Ejemplos de mapas de color: (a) Cualitativo, (b) secuencial y (c) divergente. Extraído de (Brewer, 2004)

En (Moreland, 2009) se propone un mapa de color divergente con la propiedad de uniformidad perceptual, esto significa que todos los pares de colores adyacentes deben lucir igual de diferentes entre sí. Para ello se propone un espacio de color alternativo denominado Msh el cual es la versión

polar del espacio de color CIELAB (Cuyos componentes se representan como L^*, a^*, b^*). La conversión entre estos dos espacios de color está dada por:

$$M = \sqrt{L^{*2} + a^{*2} + b^{*2}} \tag{29}$$

$$s = \arccos\left(\frac{L^*}{M}\right) \tag{30}$$

$$h = \arctan\left(\frac{b^*}{a^*}\right) \tag{31}$$

$$L^* = M\cos(s) \tag{32}$$

$$a^* = M\sin(s)\cos(h) \tag{33}$$

$$b^* = M\sin(s)\sin(h) \tag{34}$$

Moreland demuestra que interpolando linealmente por partes en el espacio Msh entre dos colores saturados se obtiene un mapa de color divergente que satisface la condición de uniformidad perceptual. El mapa de color divergente mostrado en la (Figura 16) según Moreland es más adecuado para la visualización científica ya que divide lógicamente los valores escalares en tres regiones: valores bajos, valores medios y valores altos, y la selección de los colores se basa en la percepción de la temperatura del color, asociando valores negativos a los colores fríos (azul, verde) y valores positivos a los colores cálidos (rojo, amarillo).

Figura 16. Mapa de color divergente. Extraído de (Moreland, 2009)

Tabla 2. Valores RGB para la escala de color frio a cálido. Extraído de (Moreland, 2009).

Escalar	Rojo	Verde	Azul	Escalar	Rojo	Verde	Azul
0	59	76	192	0.53125	229	216	209
0.03125	68	90	204	0.5625	236	211	197
0.0625	77	104	215	0.59375	241	204	185
0.09375	87	117	225	0.625	245	196	173
0.125	98	130	234	0.65625	247	187	160
0.15625	108	142	241	0.6875	247	177	148
0.1875	119	154	247	0.71875	247	166	135
0.21875	130	165	251	0.75	244	154	123
0.25	141	176	254	0.78125	241	141	111
0.28125	152	185	255	0.8125	236	127	99
0.3125	163	194	255	0.84375	229	112	88
0.34375	174	201	253	0.875	222	96	77
0.375	184	208	249	0.90625	213	80	66

0.40625	194	213	244	0.9375	203	62	56
0.4375	204	217	238	0.96875	192	40	47
0.46875	213	219	230	1	180	4	38
0.5	221	221	221				

En este trabajo el mapa de color se implementó utilizando el lenguaje sombreador GLSL. Los valores escalares v son pasados por cada vértice al sombreador de vértices y son normalizados en el intervalo $[0,1]$ mediante:

$$s = \frac{v - v_{\min}}{v_{\max} - v_{\min}} \tag{35}$$

En donde los valores $v_{\max}, v_{\min}$ son correspondientemente el valor mínimo y máximo del conjunto de valores asociados a cada vértice, luego el escalar s es pasado al sombreador de fragmentos en donde es interpolado bilinealmente entre vértices, este valor se utiliza como coordenada de textura unidimensional para muestrear una textura generada a partir de los valores en la (Tabla 2).

3.2 SELECCIÓN INTERACTIVA

En graficación por computadora, la selección interactiva (Picking) se refiere al proceso de determinar sobré que objeto virtual proyectado en la pantalla el usuario está interactuando por medio de un dispositivo apuntador, por ejemplo, el ratón. Esta es una operación muy importante en los softwares interactivos. En el caso de la selección interactiva de objetos en 3D proyectados sobre un visor en 2D, cada pixel de la pantalla equivale a un rayo en el espacio 3D, reduciéndose al problema de determinar el punto de intersección más cercano, desde el punto de vista del observador, de dicho rayo contra cada uno de los objetos virtuales. Debido a que esta operación forma parte del flujo de interacción, se requiere de una solución en tiempo real.

3.2.1 Intersección entre un rayo y un triángulo

En un análisis presentado por (Jiménez et al., 2014) se enlista los más prominentes algoritmos para la prueba de intersección entre un rayo y un triángulo como sigue:

- Algoritmo de Badouel. Determina la intersección entre el rayo y el plano del triángulo, la subsecuente proyección sobre un plano, y el cálculo del punto de intersección sobre el triángulo se realiza usando coordenadas baricéntricas en 2D.
- Algoritmo de Möller-Trumbore. Se basa en la solución de un sistema de ecuaciones formado por la ecuación del rayo y la ecuación del punto de intersección entre un rayo y un triángulo usando coordenadas baricéntricas (Möller and Trumbore, 1997), es una optimización del algoritmo de Badouel y es considerado como la solución más rápida para un caso genérico.
- Algoritmo de Segura. Calcula el volumen con signo del tetraedro formando por los vértices del triángulo y los puntos finales del segmento, en este caso el signo indica de qué lado del triángulo está el punto a probar. Este método provee un resultado booleano de la intersección, pero no calcula el punto de intersección.

- Algoritmo de Jiménez. Se basa en el cálculo de la coordenadas baricéntricas del punto de intersección con respecto al tetraedro formado por el triángulo y uno de los puntos del segmento (Jiménez et al., 2010).
- Otros algoritmos. Incluye el uso de coordenadas de Plüker, optimizaciones para trazado de rayos, y el algoritmo estanco (*Watertight Ray/Triangle Intersection*) (Woop et al., 2013). Este último se explica en detalle a continuación ya que es el que se utilizó en este trabajo.

3.2.1.1 Algoritmo estanco

Es un algoritmo para la intersección entre un rayo y un triángulo propuesto por (Woop et al., 2013), tiene la ventaja de ser robusto para todas las configuraciones de triángulos. Aplicando una transformación afín se coloca el rayo, de longitud unitaria, en el origen apuntando hacia uno de los ejes coordenados. Esto simplifica el problema de la intersección a un problema 2D en donde las aristas pueden ser verificadas de manera conservativa, utilizando aritmética de punto flotante de precisión simple.

Considérese un rayo $\mathbf{R} = (\mathbf{p},\ \mathbf{d})$ con origen $\mathbf{p} \in \mathbb{R}^3$ y dirección $\mathbf{d} \in \mathbb{R}^3$, primero se encuentra una transformación que genere el rayo unitario $\mathbf{R}'$, con origen $\mathbf{p}' = (0 \quad 0 \quad 0)^{\mathrm{T}}$ y dirección $\mathbf{d}' = (0 \quad 0 \quad 1)^{\mathrm{T}}$.

Sin pérdida de generalidad, asumiremos que la componente de $\mathbf{d}$ en la dirección $\mathcal{Z}$ tiene mayor valor absoluto. En caso de no ser así intercambiaremos los componentes de forma tal que se conserve la orientación (*Winding*) de los ejes del sistema coordenado. En caso de que $d_{\mathcal{Z}}$ sea negativo se intercambian las coordenadas $\mathcal{X}$, $\mathcal{Y}$ preservando así la orientación del triángulo. Después de los cambios de coordenadas se obtiene la transformación $\mathbf{M}$ como:

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & -d_{\mathcal{X}}/d_{\mathcal{Z}} \\ 0 & 1 & -d_{\mathcal{Y}}/d_{\mathcal{Z}} \\ 0 & 0 & 1/d_{\mathcal{Z}} \end{pmatrix} \tag{36}$$

Luego se transforman los vértices del triángulo $\triangle\mathbf{abc}$ con respecto al origen $\mathbf{p}$ del rayo usando $\mathbf{M}$:

$$\mathbf{a}' = \mathbf{M}(\mathbf{a} - \mathbf{p}) \tag{37}$$

$$\mathbf{b}' = \mathbf{M}(\mathbf{b} - \mathbf{p}) \tag{38}$$

$$\mathbf{c}' = \mathbf{M}(\mathbf{c} - \mathbf{p}) \tag{39}$$

Y se calculan las coordenadas baricéntricas con escalamiento de acuerdo a:

$$u = \mathbf{d}' \cdot (\mathbf{c}' \times \mathbf{b}') \tag{40}$$

$$v = \mathbf{d}' \cdot (\mathbf{a}' \times \mathbf{c}') \tag{41}$$

$$w = \mathbf{d}' \cdot (\mathbf{b}' \times \mathbf{a}') \tag{42}$$

Ya que $\mathbf{d}' = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}$, esto se reduce a un caso 2D:

$$u = c'_{\mathcal{X}} b'_{\mathcal{Y}} - c'_{\mathcal{Y}} b'_{\mathcal{X}} \tag{43}$$

$$v = a'_{\mathcal{X}} c'_{\mathcal{Y}} - a'_{\mathcal{Y}} c'_{\mathcal{X}} \tag{44}$$

$$w = b'_{\mathcal{X}} a'_{\mathcal{Y}} - b'_{\mathcal{Y}} a'_{\mathcal{X}} \tag{45}$$

Si $u < 0 \vee v < 0 \vee w < 0$, el rayo no intersecta el triángulo. En caso contrario se calcula el determinante del sistema de ecuaciones como $\det = u + v + w$. Si el determinante es cero, el rayo y el triángulo son coplanares, y por convención diremos que no existe intersección.

Posteriormente se calcula la distancia de intersección t interpolando las coordenadas $\mathcal{Z}$ de los vértices transformados:

$$t = ua'_{\mathcal{Z}} + vb'_{\mathcal{Z}} + wc'_{\mathcal{Z}} \tag{46}$$

Si $t \leq 0$ el rayo está detrás del triángulo y no lo intersecta, en caso contrario se calculan las coordenadas baricéntricas normalizadas $\begin{pmatrix} u/\det & v/\det & w/\det \end{pmatrix}$ y la distancia de intersección real $t/\det$.

3.2.2 Cálculo del rayo

Para determinar el origen y dirección del rayo se requiere conocer la posición del puntero del ratón en la pantalla, así como los parámetros de la cámara virtual. Sea $\begin{pmatrix} m_{\mathcal{X}}, m_{\mathcal{Y}} \end{pmatrix}$ la posición del puntero del ratón sobre el área de visualización con respecto a la esquina inferior expresada pixeles. Para obtener el origen del rayo se proyecta la posición del puntero del ratón sobre el plano cercano de la cámara virtual, siguiendo la dirección del rayo según el modelo de proyección.

3.2.2.1 Proyección perspectiva

El origen del rayo se obtiene como:

$$\begin{pmatrix} P \\ 1 \end{pmatrix} = \mathbf{M}_{\text{camera}} \begin{pmatrix} \dfrac{n}{he}\left(m_{\mathcal{X}} - \dfrac{w}{2} \right) & \dfrac{n}{he}\left(m_{\mathcal{Y}} - \dfrac{h}{2} \right) & -n & 1 \end{pmatrix}^{T} \tag{47}$$

La dirección del rayo está dada por:

$$\begin{pmatrix} D \\ 0 \end{pmatrix} = \frac{\mathbf{M}_{\text{camera}} \begin{pmatrix} m_{\mathcal{X}} - w/2 & m_{\mathcal{Y}} - h/2 & -he & 0 \end{pmatrix}^{\mathrm{T}}}{\left| \mathbf{M}_{\text{camera}} \begin{pmatrix} m_{\mathcal{X}} - w/2 & m_{\mathcal{Y}} - h/2 & -he & 0 \end{pmatrix}^{\mathrm{T}} \right|} \tag{48}$$

3.2.2.2 Proyección ortográfica

El origen del rayo se calcula de acuerdo a:

$$\begin{pmatrix} P \\ 1 \end{pmatrix} = \mathbf{M}_{\text{camera}} \begin{pmatrix} m_{\mathcal{X}} - \dfrac{w}{2} & m_{\mathcal{Y}} - \dfrac{h}{2} & -n & 1 \end{pmatrix}^{T} \tag{49}$$

La dirección del rayo es simplemente:

$$\begin{pmatrix} D \\ 0 \end{pmatrix} = \frac{\mathbf{M}_{\text{camera}} \begin{pmatrix} 0 & 0 & -1 & 0 \end{pmatrix}^{\mathrm{T}}}{\left| \mathbf{M}_{\text{camera}} \begin{pmatrix} 0 & 0 & -1 & 0 \end{pmatrix}^{\mathrm{T}} \right|} \tag{50}$$

3.2.3 Intersección entre un rayo y una malla triangular utilizando un octree

Para hallar la intersección entre un rayo y una malla triangular en tiempo real, en este trabajo se propone utilizar una estructura de datos conocida como octree o árbol octal. En este enfoque el espacio que ocupa el objeto es dividido de manera recursiva en octantes, hasta alcanzar determinado nivel de detalle (Figura 17). En cada subdivisión se generan ocho celdas que conforman los nodos de la estructura de árbol. En el último nivel del octree cada nodo hoja contiene referencias a los elementos que están contenidos en su volumen.

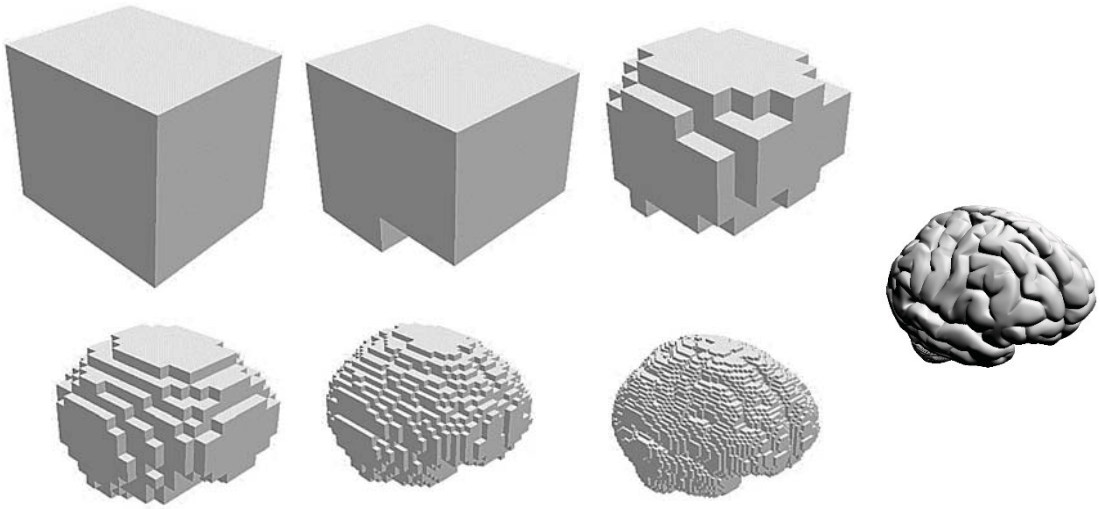


Figura 17. Visualización de seis niveles de resolución del octree de un modelo de cerebro[1] comparado con el modelo original.

En la implementación desarrollada en este trabajo, cada nodo del octree tiene información sobre la profundidad a la que se encuentra y sobre el espacio que ocupa, almacenando la posición del centro de la celda y de las extensiones medias en cada uno de los ejes coordenados. Para realizar el recorrido del octree, cada nodo contiene apuntadores a cada uno de sus ocho hijos y un apuntador a su celda padre. En el caso del nodo raíz, el puntero de nodo padre es nulo. Algunos punteros a los nodos hijos de los nodos internos pueden ser nulos en caso de que las celdas correspondientes no contengan elementos. Las celdas hoja, que están en el máximo nivel de profundidad del octree poseen todos los apuntadores de sus hijos con valores nulos. Cada nodo hoja mantiene una lista con los índices de los triángulos de la malla que están contenidos total o parcialmente dentro del volumen de la celda respectiva. El octree en sí solo contiene un nodo raíz y una referencia a la malla triangular sobre la cual se construye.

Para la construcción del octree primero se determina el centro y las extensiones medias la caja envolvente de la malla, las cuales son asignadas al nodo raíz. Luego se insertan una a una las caras

[1] Modelo obtenido en http://gfx.cs.princeton.edu/proj/sugcon/models/

triangulares dentro del octree, esto es, se realiza una búsqueda para determinar a qué hojas pertenece cada cara. Para evitar el uso de recursividad, se utiliza una pila en la cual se van insertando los nodos que contienen parcial o totalmente al triángulo. En cada iteración se saca el nodo en el tope de la pila y se verifican cuáles de las celdas hijas contienen al triangulo, y se insertan en la misma pila. Cuando se alcanza la profundidad máxima del octree, el índice del triángulo es insertado en la lista de índices del nodo hoja correspondiente. El algoritmo para la construcción del octree se describe en el (Pseudocódigo 1).

```
Crear el nodo raíz y asignarle el centro y extensiones medias de la caja envolvente de la malla
NodeStack es una pila de nodos de octree vacía
Para cada triángulo de la malla
        Insertar en NodeStack el nodo raíz
        Mientras NodeStack tenga elementos hacer
                Parent es el nodo extraído del tope de NodeStack
                Si la profundidad del nodo Parent es igual a la máxima profundidad entonces
                        Si la lista de triángulos del nodo Parent es nula entonces
                                Crear la lista de triángulos.
                        Fin si
                        Insertar el triángulo en la lista de triángulos del nodo Parent.
                Si no
                        Para cada nodo hijo de Parent hacer
                                Si el hijo es nulo entonces
                                        Calcular el centro y extensiones medias de la celda hija
                                        según el octante.
                                        Si el triángulo intercepta con la celda calculada entonces
                                                Crear el nodo hijo e insertarlo en NodeStack
                                        Fin si
                                Si no, si el triángulo intersecta con la celda del nodo hijo entonces
                                        Insertar el nodo hijo en NodeStack
                                Fin si
                        Fin para cada
                Fin si
        Fin mientras
Fin para cada
```

Pseudocódigo 1. Algoritmo para la construcción del octree a partir de una malla triangular.

La profundidad del octree se determina como:

$$n = \left\lceil \frac{\log m}{\log 8} \right\rceil \tag{51}$$

En donde m es el número de triángulos en la malla. Este cálculo se basa en el caso ideal en que cada nodo hoja solo contiene un único triángulo y proporciona una cota inferior que es adecuada a nuestros propósitos, ya que no se busca aproximar la superficie de la malla con precisión, sino agilizar la selección interactiva sin incrementar demasiado la cantidad de memoria adicional requerida.

La inserción de un triángulo en el octree toma un tiempo lineal en la profundidad del octree. Por lo tanto, la complejidad del algoritmo de construcción del octree es $O(n \log n)$ en el número n de triángulos de la malla.

Debido a que los nodos son creados conforme son requeridos, y los nodos que no contienen elementos nunca se crean, no existe relación entre la distribución espacial de las celdas que corresponden a los nodos, y el orden de los nodos representados en memoria, con lo cual es necesario el uso de punteros para navegar dentro del octree. No obstante, el requerimiento de memoria es mucho menor a si se tratara de un octree completo, en que todos los nodos en cada una de las profundidades son reservados y dispuestos linealmente en memoria con tal de hacer eficiente el cálculo de los índices. La memoria requerida por un octree completo de profundidad n es:

$$1+8+64+\cdots+8^{n}=\frac{8^{n+1}-1}{7} \tag{52}$$

Cada celda correspondiente a cada nodo, está definida de manera explícita por las coordenadas del centro de la celda, y por las extensiones medias del volumen de la celda en cada eje coordenado, las cuales no necesariamente son las mismas en los tres ejes. Los volúmenes de todas las celdas en un mismo nivel no se traslapan. Inicialmente, la celda del nodo raíz corresponde a la caja envolvente de la malla, cuyo centro no necesariamente coincide con el origen. En general, las extensiones medias de las celdas hijas son la mitad de las de la celda padre y corresponden a la división en octantes de la celda padre, los centros de cada celda hija son asignados en función del índice del nodo hijo, y se calculan como:

$$\mathbf{c}_{\text{child}}\left(i\right)=\mathbf{c}_{\text{parent}}+\left(s_{\mathcal{X}}\left(i\right),s_{\mathcal{Y}}\left(i\right),s_{\mathcal{Z}}\left(i\right)\right)\frac{\mathbf{h}_{\text{parent}}}{2} \tag{53}$$

En donde $\mathbf{c}$ es la posición del centro de la celda, $\mathbf{h}$ son las extensiones medias de la celda en cada eje coordenado e i es el índice del octante, las funciones $s_{\mathcal{X}},s_{\mathcal{Y}},s_{\mathcal{Z}}$ son factores de desplazamiento en cada eje coordenado según el octante de acuerdo a la (Tabla 3).

Tabla 3. Tabla de factores de desplazamiento por octante

i	$s_{\mathcal{X}}\left(i\right)$	$s_{\mathcal{Y}}\left(i\right)$	$s_{\mathcal{Z}}\left(i\right)$
1	-1	-1	-1
2	-1	-1	$+1$
3	-1	$+1$	-1
4	-1	$+1$	$+1$
5	$+1$	-1	-1
6	$+1$	-1	$+1$
7	$+1$	$+1$	-1
8	$+1$	$+1$	$+1$

Para la prueba de traslape entre una celda y un triángulo, para determinar si un triángulo en particular pertenece o no a una celda durante el proceso de construcción del octree, se utiliza el algoritmo rápido propuesto por (Akenine-Möller, 2005) basado en el *Teorema de los Ejes Separados*. Dicho teorema establece que dos poliedros convexos, A y B, son disjuntos si pueden ser separados por un eje paralelo a la normal de una cara de A o de B, o a lo largo de un eje formado

por el producto cruz de una arista de A con una arista de B. El algoritmo de Akenine-Möller se enfoca en encontrar la intersección entre una caja envolvente alineada con los ejes (*Axis-Aligned Bounduing Box*), en adelante AABB, definida por un centro $\mathbf{c}$ y un vector $\mathbf{h}$ de longitudes medias, contra un triángulo $\triangle \mathbf{u}_0\mathbf{u}_1\mathbf{u}_2$. Para simplificar la prueba, primero se traslada el triángulo de forma tal que la caja esté centrada en el origen, $\mathbf{v}_i = \mathbf{u}_i - \mathbf{c},\ i \in \{0,1,2\}$ (Figura 18).

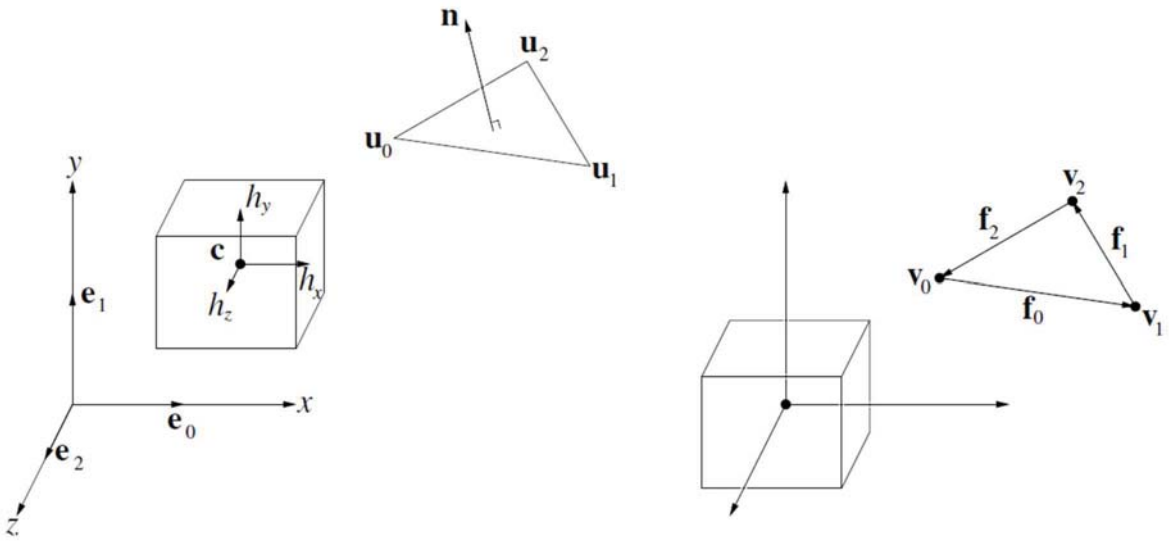


Figura 18. (Izquierda) Posición inicial de la caja y el triángulo, (derecha) la caja y el triángulo trasladados para que el centro de la caja coincida con el origen. Extraído de (Akenine-Möller, 2005).

Con base en el Teorema de los Ejes Separados, se prueban con los siguientes 13 ejes:

1. (3 ejes) $\mathbf{e}_0 = (1,0,0),\ \mathbf{e}_1 = (0,1,0),\ \mathbf{e}_2 = (0,0,1)$. Corresponden a las normales del AABB.
2. (1 eje) $\mathbf{n}$, la normal del triángulo.
3. (9 ejes) $\mathbf{a}_{ij} = \mathbf{e}_i \times \mathbf{f}_j,\ i,j \in \{0,1,2\}$ en donde $\mathbf{f}_0 = \mathbf{v}_1 - \mathbf{v}_0,\ \mathbf{f}_1 = \mathbf{v}_2 - \mathbf{v}_1,\ \mathbf{f}_2 = \mathbf{v}_0 - \mathbf{v}_2$.

Si todas las pruebas pasan, significa que no existe un eje separado, por lo tanto, el triángulo se traslapa con la caja. Para nuestro trabajo se utilizó la implementación del algoritmo en lenguaje C, disponible en (Akenine-Möller, 2001).

3.2.3.1 Descripción del algoritmo

Una vez que el octree ha sido construido, y dado que se trabaja con mallas estáticas, este no cambia. Para realizar las pruebas de intersección entre un rayo y la malla triangular envuelta por el octree, se requiere antes un algoritmo eficiente para la intersección entre un rayo y una caja, para ello se utiliza la implementación en lenguaje C++ del algoritmo propuesto en (Williams et al., 2005) disponible en (Williams, 2005). El algoritmo para la intersección de un rayo y una malla triangular usando un octree se describe en el (Pseudocódigo 2). En este algoritmo se emplea una pila para realizar iterativamente la búsqueda de la celda hoja intersectada por el rayo más cercana al origen del mismo.

En cada iteración se toma el nodo en tope de la pila y se prueban cuáles de sus nodos hijos son intersectados por el rayo, luego son ordenados e insertados en la misma pila de acuerdo a su distancia al origen del rayo. En la primera iteración la pila contiene al nodo raíz y en las iteraciones siguientes el nodo en el tope de la pila es el nodo intersectado por el rayo más cercano al origen del rayo resultado de la iteración anterior. Si el rayo no intercepta a ningún octante del modo raíz, la pila se vacía y el algoritmo devuelve falso indicando que el rayo no intersecta a la malla. En otro caso, se requerirán tantas iteraciones como profundidad del octree para llegar a un nodo hoja intersectado por el rayo que sea el más cercano al origen del rayo. Cuando se llega a un nodo hoja se realiza la prueba de intersección del rayo contra cada uno de los triángulos contenidos dentro del nodo utilizando el algoritmo de la sección 3.2.1.1. Si el rayo colisiona con alguno de los triángulos, entonces se toma el punto de intersección más cercano al origen del rayo y la ejecución del algoritmo termina devolviendo el valor verdadero (Figura 19). En caso contrario la ejecución continúa y se prueba con las siguientes celdas candidatas en el orden en que están en la pila.

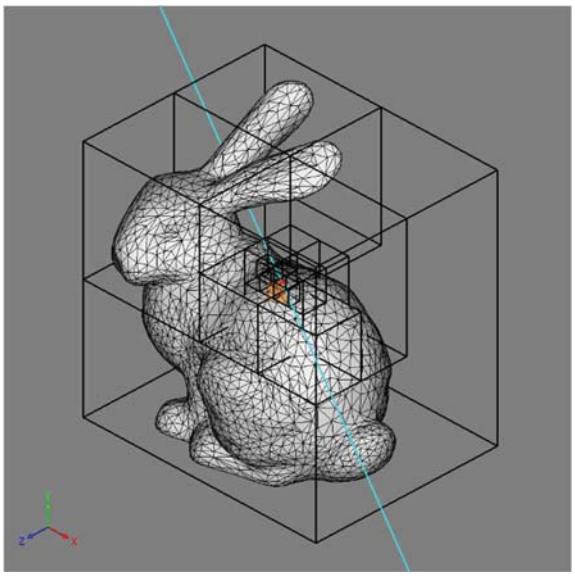


Figura 19. Colisión de un rayo con una malla triangular usando un octree.

La complejidad de este algoritmo de intersección de un rayo con una malla triangular utilizando un octree, en el caso promedio, es lineal en relación a la profundidad del octree, ya que equivale a un árbol de búsqueda binario. Pero como la profundidad del octree es logarítmica en el número de triángulos de la malla, entonces la complejidad del algoritmo es $O(\log n)$.

Ray Es un rayo formado por un punto de origen y el vector unitario de dirección
Intersection es el punto de intersección, es la variable de salida
NodeStack es una pila de nodos de octree vacía
NodeList es una lista de nodos de octree vacía
Insertar en *NodeStack* el nodo raíz del octree
Mientras *NodeStack* tenga elementos **hacer**
    Parent es el nodo extraído del tope de *NodeStack*
    Si *Parent* es nodo interno **entonces**
        Para cada nodo hijo de *Parent* **hacer**
            Si el nodo hijo es nulo entonces
                Continuar
            Fin si
            Si el rayo *Ray* colisiona con la celda del nodo hijo **entonces**
                Insertar el nodo hijo en *NodeList*
            Fin si
        Fin para cada
        Si *NodeList* está vacía **entonces**
            Continuar
        Fin si
        Ordenar *NodelList* por distancia del punto de colisión al origen del rayo de mayor a menor
        Recorrer *NodeList* e insertar cada elemento en *NodeStack*
        Vaciar *NodeList*
    Si no, si el rayo *Ray* colisiona con alguno de los triángulos contenidos en *Parent* **entonces**
        Asignar a *Intersecction* el punto de intersección más cercano al origen del rayo *Ray*
        Devolver verdadero
    Fin si
Fin mientras
Devolver falso

Pseudocódigo 2. Algoritmo para la intersección de un rayo y una malla triangular utilizando un octree.

3.3 RETROALIMENTACIÓN HÁPTICA

En este trabajo se implementó el uso de interfaces hápticas para la selección de puntos de referencia, como método alternativo al uso del ratón. La ventaja de dicho enfoque es que permite al usuario explorar el espacio 3D de forma más intuitiva, y mediante la retroalimentación de fuerza, tener una mejor percepción del objeto de estudio. En esta sección se introducirá el concepto de dispositivo háptico, y se explicará a detalle el algoritmo utilizado para el renderizado háptico.

3.3.1 Dispositivos Hápticos

La palabra *háptica* se deriva de la palabra griega $\alpha\pi\tau\acute{o}$ (hapto), que significa "tangible". Aunque esta palabra se emplee en un contexto científico, para designar al conjunto de tecnologías capaces de generar sensaciones de contacto de manera artificial, es aplicable para referirnos a cualquier sensación táctil sin importar su origen. Las sensaciones de contacto que experimentan los humanos son debidas a dos diferentes sistemas de sensores: los receptores en la piel, y los receptores en músculos y articulaciones. Aunque ambos sistemas actúan en conjunto para brindar una sensación

completa cuando se toca un objeto, son usados para percibir diferentes propiedades del objeto en la mano.

La piel contiene tres tipos de receptores cutáneos, lo cuales proporcionan sensaciones de dolor, temperatura y tacto. El último de estos tres tipos, es el encargado de detectar cambios de presión en la piel y vibraciones, y son capaces de determinar información textural y geometría local. Comúnmente las sensaciones producidas por esos receptores es lo que se denomina tacto. Los músculos y las articulaciones poseen receptores para determinar sus respectivas posiciones. Esto es lo que se conoce como propiocepción, refiriéndonos a la respuesta a los estímulos internos, resultado de las tensiones y posiciones de las extremidades. Este sentido, además de ser utilizado para el control de movimientos y posturas corporales, nos permite determinar pesos, fuerzas, fricción, viscosidad y formas. Comúnmente se conoce como el componente kinestésico de las sensaciones de contacto (Palmerius, 2007).

En ciencias de la computación, los dispositivos hápticos son un tipo de interface de computadora en donde el contacto es parte del flujo de información entre el usuario y la computadora. La retroalimentación háptica, acoplada con el despliegue visual, permite a los usuarios interactuar de manera intuitiva con los objetos virtuales, mejorando su habilidad en la realización de tareas complejas. Hoy en día los dispositivos hápticos más comunes son los dispositivos kinestésicos, los cuales se basan en transmitir fuerzas y posiciones entre el usuario y la computadora. La interacción se realiza mediante la manipulación de una herramienta, típicamente en forma de pluma (*Stylus*), acoplada a un dispositivo robótico formado por eslabones, el cual ejerce fuerzas en función de las acciones del usuario (Zilles, 1995). En este sentido el sujeto no toca directamente los objetos virtuales, sino que utiliza la punta de la pluma, llamada Punto de Interfaz Háptica, en adelante HIP (*Haptic Interface Point*), para explorar el ambiente virtual.

Los dispositivos hápticos kinestésicos se clasifican de acuerdo al número de grados de libertad, en adelante DOF (Degrees Of Freedoom). Esta propiedad describe el número de dimensiones independientes que el dispositivo es capaz de monitorear o controlar. Los dispositivos pueden tener diferentes DOF de entrada y de salida. Los dispositivos hápticos comerciales más comunes son los dispositivos kinestésicos con 6 DOF de entrada, para la posición-orientación del efector final, y 3 DOF de salida para la retroalimentación de fuerza. Como ejemplo de esta clasificación, está el dispositivo háptico PHANTOM OMNI[1], en adelante Phantom, utilizado en este trabajo.

[1] http://www.dentsable.com/haptic-phantom-omni.htm

Figura 20. Dispositivo háptico PHANTOM OMNI de 3 DOF

3.3.2 Renderizado Háptico

El renderizado háptico (*Haptic Rendering*) es una aplicación de los dispositivos hápticos, cuyo objetivo es lograr que los objetos virtuales simulen ser objetos físicos tangibles. El dispositivo háptico es utilizado para percibir e interactuar con los objetos virtuales, con frecuencia acompañado del despliegue gráfico. El renderizado háptico es análogo al renderizado gráfico, con la diferencia que los objetos virtuales son percibidos a través del tacto en lugar de la visión. Las interfaces hápticas han sido utilizadas en múltiples aplicaciones incluyendo entrenamiento quirúrgico, investigación sobre el sentido del tacto humano, acoplamiento molecular y pintura y modelado 3D (Sjöberg and Ylinenpää, 2009).

La propiedad háptica más básica a renderizar es la forma del objeto rígido, para percibir la superficie del objeto y explorar su forma. Otras propiedades que pueden ser añadidas son, por ejemplo, la fricción, rigidez y textura. Dada la posición del Punto de Interfaz Háptica (HIP), se requiere determinar la fuerza apropiada que aplicada al dispositivo háptico mantenga el HIP sobre la superficie de los objetos. Para este trabajo se utilizará el algoritmo de renderizado háptico conocido como el método del campo vectorial o método de penalización (*Vector-field/Penalty Methods*) propuesto por primera vez por (Massie and Salisbury, 1994). En este método la fuerza de reacción es directamente proporcional a la penetración dentro del volumen de los objetos virtuales. Para geometrías simples como esferas y planos, la dirección e intensidad de la fuerza es fácil de calcular (Figura 21 a). Las limitaciones de este método son que la fuerza puede cambiar abruptamente, si el HIP penetra en el objeto y se aleja de una cara y se acerca otra (Figura 21 b), y en el caso de los objetos delgados, cuando el HIP ha pasado más de la mitad del volumen, la fuerza cambia de dirección, haciendo que el HIP atraviese el objeto (Figura 21 c), esto es porque los objetos muy delgados carecen del volumen necesario para generar las fuerzas que prevenga que el HIP atraviese el obstáculo (Ruspini et al., 1997).

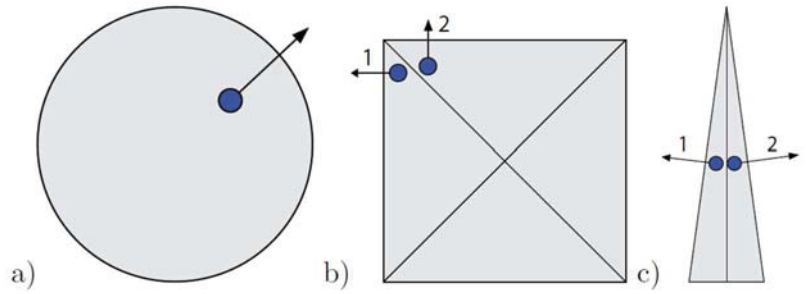


Figura 21. (a) La fuerza de reacción al tocar una esfera puede calcularse fácilmente. (b) La dirección de la fuerza puede cambiar abruptamente. (c) Los objetos delgados pueden atravesarse. Extraído de (Sjöberg and Ylinenpää, 2009).

En nuestra implementación, en lugar de pre-calcular el campo vectorial de fuerzas de reacción, primero se determina si el punto correspondiente a la posición del HIP está colisionando con la malla, es decir, si se encuentra en el interior del objeto. Para ello se utiliza una versión modificada del algoritmo de colisión entre un rayo y la malla triangular usando un octree de la sección 3.2.3.1, con la diferencia de que el algoritmo no termina con la primera colisión, sino que encuentra todos los puntos de intersección entre el rayo y la malla triangular, siendo la complejidad del algoritmo modificado $O(\log n)$. Para evitar contar dos veces el mismo punto de intersección se utiliza un algoritmo de eliminación de redundancias similar al utilizado para eliminar los vértices duplicados de la malla. Para determinar si un punto está en el interior de la malla, se aplica la regla par-impar para determinar la inclusión de un punto en un polígono cerrado (Figura 22), tomando ventaja de las propiedades topológicas de las superficies cerradas, se traza un rayo tomando como origen el punto y una dirección arbitraria, y si el número de intersecciones con la superficie de la malla es un número par, concluiremos que el punto está fuera de la malla, en caso contrario el punto está al interior de la malla. Dicha regla es una consecuencia del teorema de la curva de Jordan, el cual ha sido demostrado de que puede extenderse a dimensiones mayores (Kopperman et al., 1991). Por simplicidad no se considerarán los casos degenerados cuando el punto yace sobre la superficie de la malla, o cuando el rayo proyectado es tangente a la superficie.

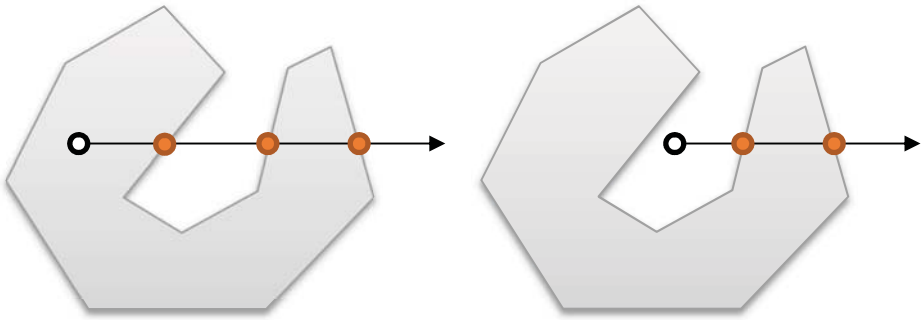


Figura 22. (Izquierda) El número de intersecciones entre el rayo y el contorno del polígono es impar, significa que el punto está en el interior del polígono cerrado. (Derecha) Si el número de intersecciones es par, entonces el punto está afuera.

Una vez que se sabe que el HIP está en el interior de la malla, se determina el punto más cercano sobre la superficie de la malla, para ello se utiliza el algoritmo del punto más cercano a una malla triangular utilizando un octree descrito a continuación en el Pseudocódigo 3. En el algoritmo propuesto se realiza una búsqueda dentro del octree hasta encontrar la celda más cercana al punto

correspondiente a la posición actual del HIP, mientras se recorre el octree en profundidad, las celdas son ordenas por distancia al punto e insertadas en una pila, de forma tal que para una iteración determinada la celda al tope de la pila es la más cercana al punto. Cuando se llega a un nodo hoja se analizan todos los triángulos en ese nodo y se actualiza la distancia más cercana y el punto más cercano hasta el momento. Siempre que se saque un nodo de la pila se compara su distancia al punto con la distancia más corta hasta ese momento y se descarta si la distancia de la celda es mayor. De esta forma se garantiza que siempre se encuentra el punto más cercano, mientras que las celdas que no pueden contener puntos más cercanos al más cercano hasta ahora son descartadas, con ello se evita un algoritmo de fuerza bruta de tiempo lineal. La complejidad de este algoritmo es $O(\log n)$ ya que es equivalente a un árbol binario de búsqueda.

Point es el punto como argumento de entrada
ShortestDistance es la distancia más cercana inicializada en infinito
NearestPoint es el punto más cercano
NodeStack es una pila de nodos de octree vacía
NodeList es una lista de nodos de octree vacía
La distancia del nodo raiz es menos infinito
Insertar en *NodeStack* el nodo raíz del octree
Mientras *NodeStack* tenga elementos **hacer**
    Parent es el nodo extraído del tope de NodeStack
    Si la distancia del nodo *Parent* es mayor que *ShortestDistance* **entonces**
        Continuar
    Fin si
    Si *Parent* es nodo interno **entonces**
        Para cada nodo hijo de *Parent* **hacer**
            Si el nodo hijo es nulo **entonces**
                Continuar
            Fin si
            Si la distancia de *Point* a la celda del nodo hijo es menor que *ShortestDistance* **entonces**
                Asignar al nodo hijo la distancia calculada, Insertar el nodo hijo en *NodeList*
            Fin si
        Fin para cada
        Si *NodeList* está vacía **entonces**
            Continuar
        Fin si
        Ordenar *NodelList* por distancia
        Recorrer *NodeList* e insertar cada elemento en *NodeStack*
        Vaciar *NodeList*
    Si no, si la distancia de *Point* a alguno de los triángulos contenidos en *Parent* es menor que *ShortestDistance* **entonces**
        Actualizar *ShortestDistance* y *NearestPoint*
    Fin si
Fin mientras
Devolver *NearestPoint*

Pseudocódigo 3. Algoritmo para encontrar el punto más cercano en una malla triangular.

3.3.2.1 Retroalimentación de fuerza

Para determinar la fuerza necesaria para mover el HIP hasta el punto más cercano sobre la malla obtenido previamente, se utiliza un controlador Proporcional-Integral-Derivativo, en adelante PID, en el cual en cada ciclo de renderizado háptico se obtiene la posición del HIP y si colisiona con la malla se estable el valor de referencia del controlador en la posición del punto más cercano sobre la malla a la posición actual del HIP, luego se actualiza el estado del controlador y se obtiene la fuerza correspondiente a la acción de control. Finalmente se configura la fuerza del dispositivo háptico con la fuerza obtenida mediante el controlador PID. En caso de que el HIP no esté colisionando con la malla, se establece la fuerza de la interfaz háptica a cero. El ciclo de renderizado háptico se describe en el (Pseudocódigo 4).

```
Obtener la posición del HIP
Encontrar el punto sobre la malla más cercano al HIP
Establecer valor de referencia del controlador en el punto más cercano obtenido
Establecer la entrada del controlador en el del HIP
Ejecutar un ciclo de control del controlador PID
Si el HIP está en el interior de la malla entonces
        Obtener la fuerza de la salida del controlador PID
        Establecer la fuerza del Phantom en la fuerza calculada por el controlador PID
Si no, entonces
        Establecer la fuerza del Phantom en cero
Fin si
```

Pseudocódigo 4. Algoritmo de un ciclo de renderizado háptico utilizando un octree para la detección de colisiones y un controlador PID para el cálculo de la fuerza.

3.3.3 Controlador de posición del dispositivo háptico

Con el fin de mantener el HIP del dispositivo háptico en una posición dada, se implementó como ya se mencionó un controlador PID. El objetivo de este controlador es determinar la fuerza necesaria para mantener la posición del efector final del dispositivo háptico HIP en el punto dado como señal de referencia del controlador (equivalente al punto de anclaje más cercano a la malla con respecto al HIP). Para ello se implementó una versión en tiempo discreto de un controlador PID en lenguaje C++, considerando que el sistema opera siempre a 1000Hz para garantizar la estabilidad numérica del controlador. El sistema de control, compuesto por el dispositivo háptico acoplado con el controlador PID, se muestra en (Figura 23). La implementación del controlador PID se detalla en la (Figura 24).

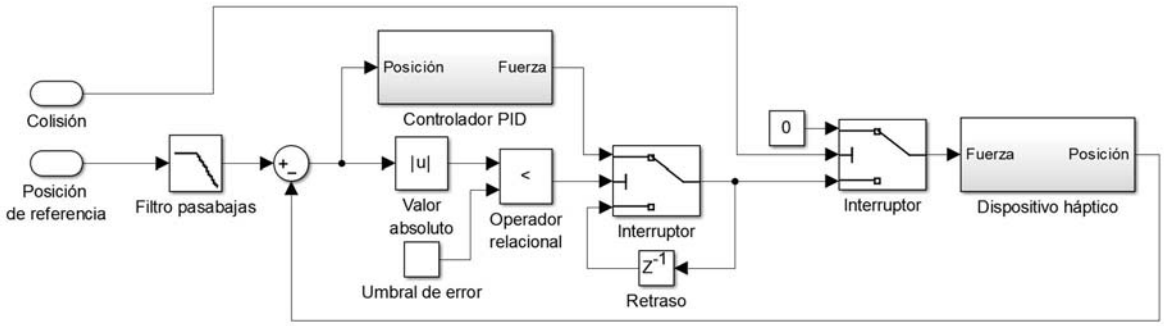


Figura 23. Diagrama de bloques del sistema de control.

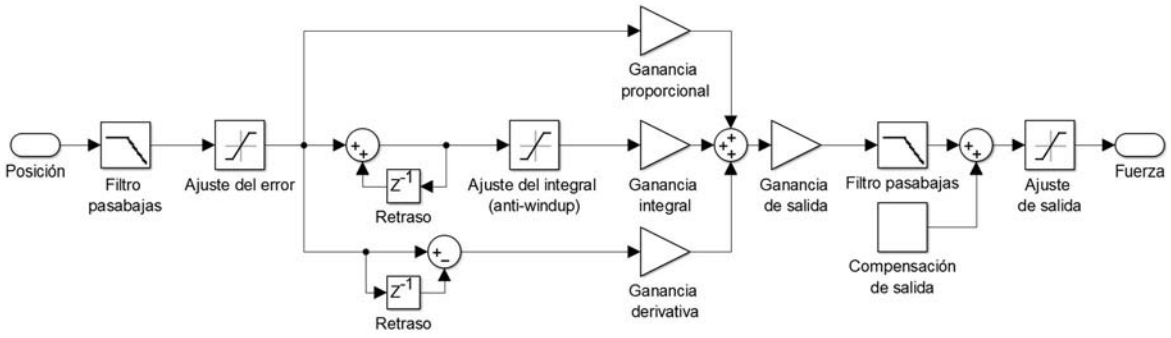


Figura 24. Diagrama de bloques del controlador PID.

Para mitigar los cambios abruptos en la señal de referencia, producto de la discretización de la malla, y en la señal de entrada del controlador para atenuar el ruido de alta frecuencia presente en las lecturas de posición del efector de posición del dispositivo háptico, se utiliza un filtro pasa bajas. También en la señal de salida del controlador PID, para evitar cambios abruptos en la fuerza aplicada al efector final del dispositivo háptico. El filtro se implementó en tiempo discreto como:

$$y^{t+} = \alpha x^{t+} + (\alpha - 1) y^{t} \tag{54}$$

En donde x, y son correspondientemente los valores de entrada y de salida del filtro, t, $t+$ denotan respectivamente los valores en la iteración anterior y en la iteración actual, y $\alpha \in [0,\ 1]$ es un factor de filtrado, que pondera la señal de entrada y la señal de salida del filtro en la iteración anterior. El efecto debido a distintos valores de α se muestra en la gráfica de la (Figura 25).

Figura 25. Filtro pasa bajas con distintos valores de factor de filtrado.

Sea ε la diferencia entre la señal de referencia y la señal de entrada actual después de pasar por el proceso de filtrado y de recorte. El error diferencial ε_D y el error integral ε_I, de manera discreta, están dados por:

$$\varepsilon_D^{t+} = \varepsilon^{t+} - \varepsilon^t \tag{55}$$

$$\varepsilon_I^{t+} = \varepsilon_I{}^t + \varepsilon^{t+} \tag{56}$$

La señal de salida del controlador F , en términos de fuerza, se obtiene como:

$$F^{t+} = P\varepsilon^{t+} + I\varepsilon_I^{t+} + D\varepsilon_D^{t+} \tag{57}$$

En donde $P,\ I,\ D$ son correspondientemente las ganancias proporcional, integral y derivativa del controlador PID, cuyos valores se obtuvieron experimentalmente siguiendo los siguientes pasos (Ang et al., 2005):

1. Establecer los valores I y D a cero.
2. Incrementar P hasta que la salida del lazo oscile.
3. Establecer P a aproximadamente la mitad del valor configurado previamente.
4. Incrementar I hasta que el proceso se ajuste en el tiempo requerido.
5. Finalmente, incrementar D hasta que el lazo sea lo suficientemente rápido para alcanzar su referencia tras una variación brusca de la carga.

Los valores obtenidos fueron:

$$P = 0.35$$
$$I = 0.01$$
$$D = 0.2$$

Antes de aplicar el controlador PID se evalúa la magnitud del error contra un umbral, en caso de que sea menor, la señal de salida del controlador conserva el valor de la iteración anterior. Para evitar la saturación de la señal de salida en caso de que el error crezca demasiado, debido a la acumulación en el tiempo del error integral (integral windup), el valor del error integral es ajustado para permanecer dentro de un intervalo fijo. Lo mismo se aplica en la señal de salida, para prevenir que se generen fuerzas muy grandes, que el dispositivo háptico no sea capaz de reproducir, el valor de la señal de salida es ajustado a un valor máximo predefinido. La respuesta al escalón del controlador PID implementado se muestra en la (Figura 26).

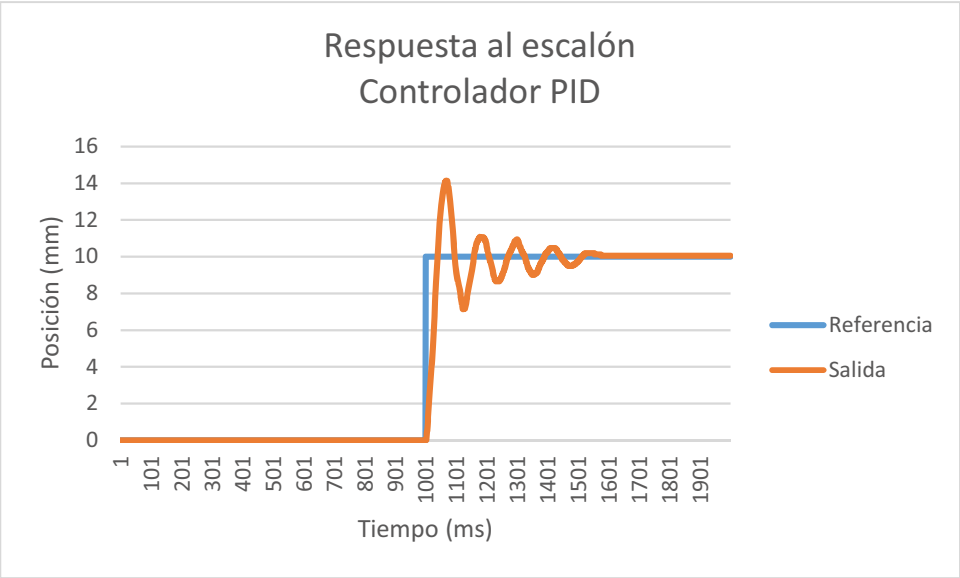


Figura 26. Respuesta del controlador PID a una señal de escalón.

3.4 DEFORMACIÓN DE MALLAS

Entre los objetivos de este trabajo, se desarrolló un modelo deformable de la cabeza humana con el fin de aproximar la forma de los modelos originales, mediante registro deformable, utilizando un conjunto reducido de puntos de referencia. En esta sección se expondrán en detalle los métodos de deformación de mallas utilizados para tal efecto.

3.4.1 Coordenadas Laplacianas

Las Coordenadas Laplacianas fueron introducidas en (Alexa, 2003). La idea principal es representar las coordenadas de los vértices con respecto a la de los vértices vecinos dentro de la malla en vez de con respecto a un marco de referencia global. El esquema presentado por Alexa para interpolación de forma, es invariante a traslaciones, pero no a transformaciones afines.

Sea $\mathbf{v}_i \in \mathbb{R}^3$ las coordenadas del vértice a ser representado, y sea $\bar{\mathbf{v}}_i \in \mathbb{R}^3$ las coordenadas del centroide de acuerdo a:

$$\bar{\mathbf{v}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{v}_j \tag{58}$$

La nueva representación del vértice es:

$$\boldsymbol{\delta}_i = \mathbf{v}_i - \bar{\mathbf{v}}_i \tag{59}$$

Siendo $\mathcal{N}_i$ el conjunto formado por los índices asociados a los vértices vecinos de $\mathbf{v}_i$ con conectividad simple (Figura 27).

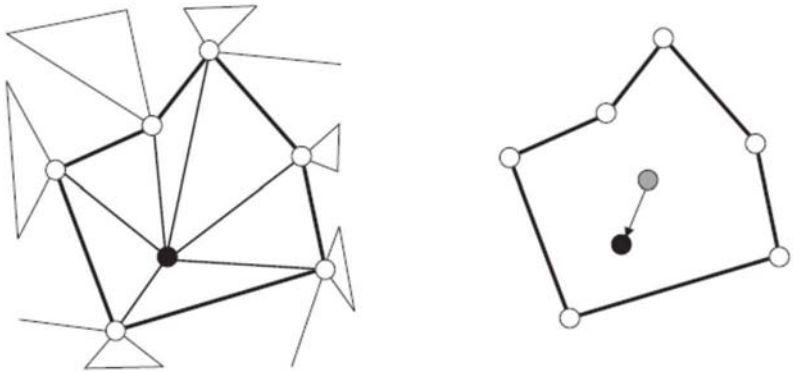


Figura 27. Un vértice (en negro) y sus vecinos con conectividad simple (blanco). En coordenadas laplacianas un vértice es representado como la diferencia al centroide de sus vecinos. Extraído de (Alexa, 2003).

La transformación de coordenadas absolutas $\mathbf{v}_i$ a coordenadas relativas $\boldsymbol{\delta}_i$ puede ser representada matricialmente como:

$$\mathcal{L}\mathbf{V} = \boldsymbol{\Delta} \tag{60}$$

En donde $\mathbf{V}$, $\boldsymbol{\Delta}$ son matrices cuyas filas son $\mathbf{v}_i$, $\boldsymbol{\delta}_i$ respectivamente. Siendo $\mathcal{L} = \mathbf{I} - \mathbf{DA}$ el operador laplaciano de la malla, en que $\mathbf{I}$ es la matriz identidad, $\mathbf{D}$ es una matriz diagonal con $d_{ii} = 1/\left|\mathcal{N}(i)\right|$ y $\mathbf{A}$ es la matriz de adyacencia de la malla.

La transformación inversa no es única, ya que $\mathcal{L}$ es singular. Resolver el sistema $\mathcal{L}\mathbf{V} = \boldsymbol{\Delta}$ para $\mathbf{V}$ no es posible de forma exacta, la alternativa es fijar algunos vértices y utilizar métodos iterativos para ajustar por mínimos cuadrados los vértices libres $\mathbf{V}'$ dentro de una región de interés tal que $\left|\mathcal{L}'\mathbf{V}'_i - \boldsymbol{\Delta}'_i\right|^2$ sea mínimo (Figura 28).

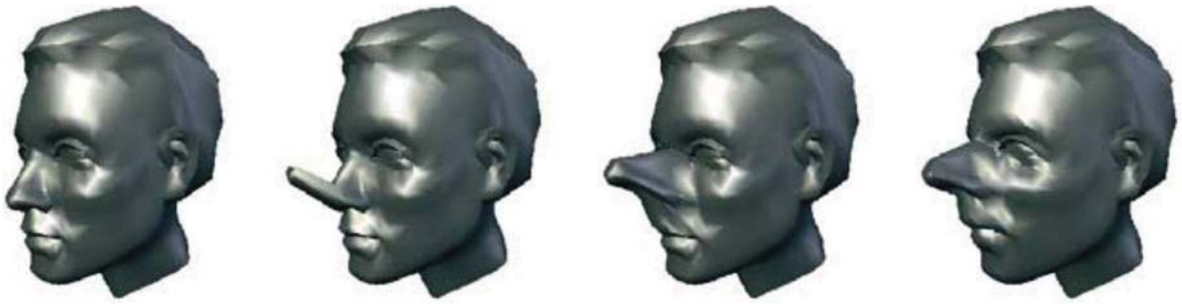


Figura 28. Uso de coordenadas laplacianas para deformación, primero se desplaza la punta de la nariz, luego el conjunto de vértices libres dentro de la región de interés son relajados para ajustar por mínimos cuadrados sus coordenadas. Extraído de (Alexa, 2003)

3.4.2 Coordenadas de Valor Medio

Las coordenadas de valor medio (*Mean Value Coordinates*), fueron introducidas por (Floater, 2003) como una generalización de las coordenadas baricéntricas para polígonos cerrados arbitrarios. Sean $\mathbf{v}, \mathbf{v}_1, \cdots, \mathbf{v}_i, \cdots, \mathbf{v}_n$ puntos en un plano con $\mathbf{v}_1, \cdots, \mathbf{v}_i, \cdots, \mathbf{v}_n$ dispuestos ordenadamente en sentido anti-horario en torno a $\mathbf{v}$ (Figura 29). Se define el conjunto de pesos $\lambda_1, \ldots, \lambda_n \geq 0$ tales que:

$$\sum_{i=1}^{n} \lambda_i (v) = 1 \tag{61}$$

$$v = \sum_{i=1}^{n} \lambda_i (v) v_i \tag{62}$$

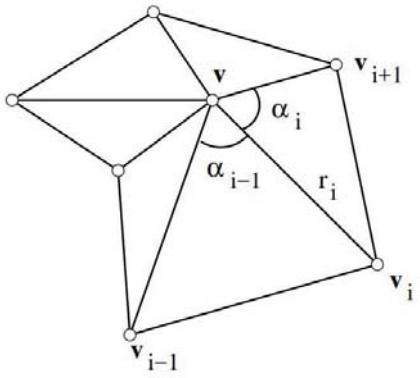


Figura 29. Polígono con forma de estrella. Extraído de (Floater et al, 2005).

Los pesos λ_i se derivan de la aplicación del *Teorema del valor medio* para funciones armónicas, con la observación de que la integral de todas las normales unitarias alrededor de un círculo es cero, como se demuestra en (Floater, 2003). Se calculan como:

$$\lambda_i = \frac{w_i}{\sum_{j=1}^{n} w_j} \tag{63}$$

$$w_i = \frac{1}{r_i}\left(\tan\frac{\alpha_{i-1}}{2} + \tan\frac{\alpha_i}{2} \right) \tag{64}$$

En donde $r_i = \left| \mathbf{v}_i - \mathbf{v} \right|$.

Posteriormente (Floater et al., 2005, Ju et al., 2005) extendieron las coordenadas de valor medio a mallas triangulares cerradas en 3D, de manera análoga al caso 2D, notando que la integral de todas las normales unitarias sobre una esfera es cero. Sea $\mathbf{v}$ un vértice en el interior de una malla triangular cerrada y $\left(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \right)$ una cara orientada de la malla. Considérese el tetraedro $\left(\mathbf{v}, \mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \right)$ con volumen positivo (Figura 30 a). Se definen los vectores unitarios $\mathbf{e}_i = \left(\mathbf{v}_i - \mathbf{v} \right) / r_i$, con $r_i = \left| \mathbf{v}_i - \mathbf{v} \right|$, y los nuevos puntos $\hat{\mathbf{v}}_i = \mathbf{v} + \mathbf{e}_i$. Los puntos $\hat{\mathbf{v}}_i$ están sobre una esfera S de radio unitario centrada en $\mathbf{v}$, y la proyección a través de $\mathbf{v}$ de cada triángulo $\mathbf{T} = \left(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \right)$ sobre esta esfera forma un triángulo esférico $\hat{\mathbf{T}}$ con vértices $\left(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j, \hat{\mathbf{v}}_k \right)$ (Figura 30 b).

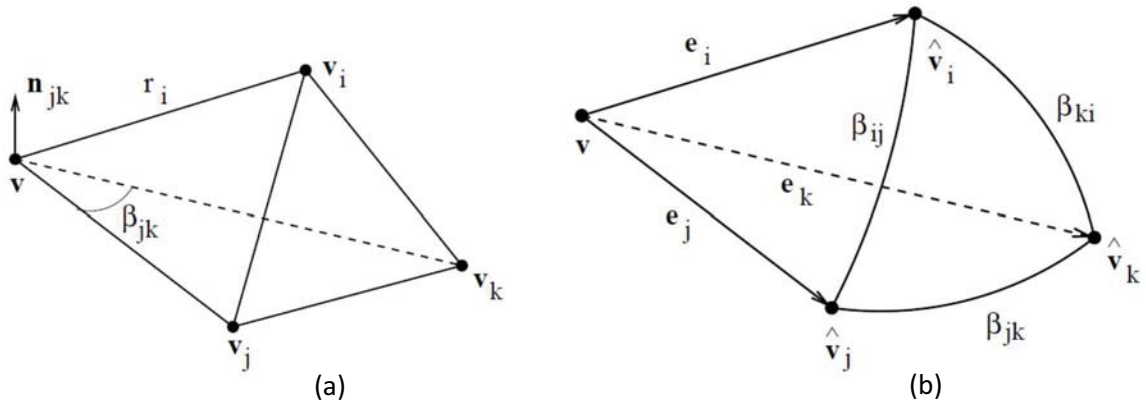


Figura 30. (a) Tetraedro y (b) triángulo esférico. Extraído de (Floater et al., 2005).

Los pesos w_i están dados por:

$$w_i = \frac{1}{r_i} \sum_{\mathbf{T} \ni v_i} \mu_{i,\mathbf{T}} \tag{65}$$

$$\mu_{i,\mathbf{T}} = \frac{\beta_{jk} + \beta_{ij}\mathbf{n}_{ij} \cdot \mathbf{n}_{jk} + \beta_{ki}\mathbf{n}_{ki} \cdot \mathbf{n}_{jk}}{2\mathbf{e}_i \cdot n_{jk}} \tag{66}$$

En donde $\mathbf{e}_a$ es el vector $\mathbf{v}_a - \mathbf{v}$, $\mathbf{n}_{ab}$ es el vector normal $\left(\mathbf{e}_a \times \mathbf{e}_b\right) / \left|\mathbf{e}_a \times \mathbf{e}_b\right|$, y β_{ab} es el ángulo en radianes comprendido entre $\mathbf{e}_a$ y $\mathbf{e}_b$, con $a, b \in \{i, j, k\}$ .según corresponda.

La derivación de las ecuaciones (65) y (66) se detalla en (Floater et al., 2005) y se basa en la integración en coordenadas baricéntricas esféricas, de los vectores normales a la superficie de $\hat{\mathbf{T}}$.

La deformación de mallas tridimensionales es una aplicación de las MVC propuesta por (Ju et al., 2005), para ello se emplean dos mallados diferentes: una malla de alta resolución y una malla de baja resolución cerrada que contiene a la primera, denominada malla de control o jaula. Para cada vértice $\mathbf{v}_i$ en la malla de alta resolución se calculan, en tiempo de inicialización, los pesos normalizados w_{ij} asociados a cada vértice $\mathbf{v}_j$ en la malla de control. La deformación se realiza desplazando los vértices en la malla de control (Figura 31). Sean $\tilde{\mathbf{v}}_j$ los vértices de la malla de control deformada, los vértices deformados en la malla de alta resolución $\tilde{\mathbf{v}}_i$ están dados por:

$$\tilde{\mathbf{v}}_i = \sum_{j=1}^{m} w_{ij} \tilde{\mathbf{v}}_j \tag{67}$$

Matricialmente se puede expresar como:

$$\mathbf{V} = \mathcal{W}\mathbf{C} \tag{68}$$

En donde $\mathbf{V}$ es una matriz de tamaño $n \times 3$, cuyas filas son los n vértices de la malla de alta resolución, $\mathcal{W}$ es la matriz $n \times m$ en donde cada renglón i contiene las coordenadas de valor medio del vértice $\mathbf{v}_i$, y $\mathbf{C}$ es una matriz de tamaño $m \times 3$, con los m vértices de la malla de control o jaula por filas.

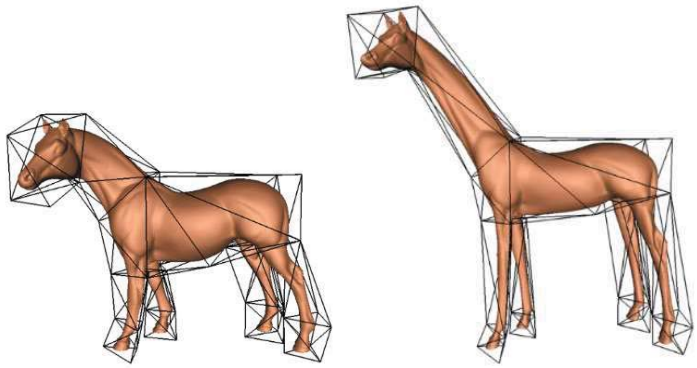


Figura 31. (izquierda) Modelo original con malla de control envolvente (derecha) al deformar la malla de control se induce la deformación en el modelo. Extraído de (Ju et al., 2005).

3.5 CURVATURA DE MALLAS

Los valores locales de las curvaturas medidos sobre la superficie de las mallas, han sido ampliamente utilizados como descriptores de forma, ya que describen de manera cuantitativa las propiedades morfológicas. En este trabajo se propuso obtener, de forma aproximada, los valores de curvatura de las mallas correspondientes a los rostros, y visualizarlos utilizando mapas de color, con el fin de apreciar mejor los detalles en las superficies, tales como las hendiduras, pliegues y líneas de cresta.

Los métodos para estimar las curvaturas principales y las direcciones sobre mallas triangulares discretas pueden ser clasificados en tres categorías generales (Rusinkiewicz, 2004):

- *Métodos de ajuste de parches*. Estos métodos ajustan una curva analítica a los puntos en una región local, luego calcula las curvaturas de la superficie ajustada de manera analítica. Estos métodos producen resultados exactos si los vértices están en la superficie ajustada. La debilidad de los algoritmos de ajuste de parches es que se vuelven inestables en condiciones degeneradas (Goldfeather and Interrante, 2004, Cazals and Pouget, 2003)
- *Métodos basados en curvaturas normales.* Éstos primero estiman la curvatura normal en la dirección de cada arista que parte de un vértice, luego utiliza los valores estimados para encontrar el segundo tensor fundamental conocido como *matriz de Weingarten*. Las curvaturas principales pueden ser encontradas como una función de los valores propios, lo cual es preciso sólo cuando la distribución de las direcciones de los vecinos es uniforme (Taubin, 1995, Hameiri and Shimshoni, 2003). En (Meyer et al., 2003) se realiza un ajuste por mínimos cuadrados para ajustar la curvatura normal.
- *Métodos de promediado del tensor.* Éstos calculan el promedio del tensor de curvatura sobre un área pequeña de la malla poligonal (Cohen-Steiner and Morvan, 2003, Alliez et al., 2003). La curvatura de un poliedro es cero dentro de una cara e infinita sobre las aristas, pero el promedio sobre una región de tamaño no cero es finito y bien definido. Los métodos

de promediado del tensor son elegantes y libres de configuraciones inestables, pero producen errores grandes bajo ciertas configuraciones de vértices.

Para este trabajo se utilizó el algoritmo propuesto en (Theisel et al., 2004). En este algoritmo el tensor de curvatura de la malla se estima tomando como entrada cada triángulo simple y las normales por vértice. La ventaja de este método está en que se la curvatura gaussiana K y la curvatura media H tienen formas cerradas en coordenadas baricéntricas, siendo el tensor de curvatura una función suave en el interior de cada triángulo de la malla (Figura 32). Además, los autores demostraron que el error de este método es comparable con el de los algoritmos de ajuste de parche de orden cúbico cuando las normales son aproximadas, reduciéndose significativamente si las normales son exactas.

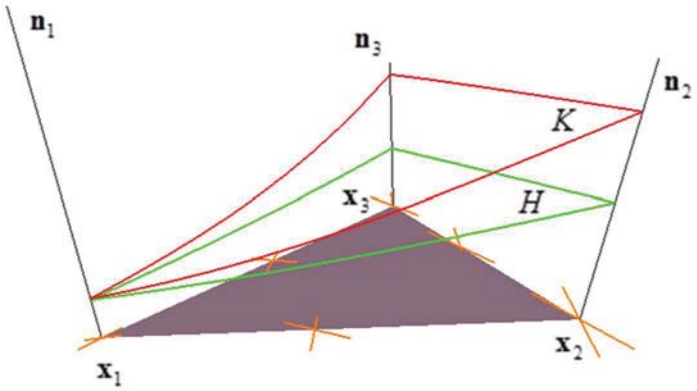


Figura 32. Un triángulo simple con visualizaciones de curvatura gaussiana y curvatura media. Extraído de (Theisel et al., 2004).

Considérese un triángulo no degenerado con vértices $\mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{x}_3$ y sus correspondientes normales no unitarias $\mathbf{n}_1,\ \mathbf{n}_2,\ \mathbf{n}_3$. Sea $(u,\ v,\ w)$ las coordenadas baricéntricas de un punto interior del triángulo, la curvatura gaussiana K y curvatura media G están dadas por:

$$K(u,\ v,\ w) = \frac{\det(\mathbf{n}_1 \quad \mathbf{n}_2 \quad \mathbf{n}_3)}{\tilde{\mathbf{n}}^2(\tilde{\mathbf{n}} \cdot \tilde{\mathbf{m}})} \tag{69}$$

$$H(u,\ v,\ w) = \frac{1}{2}\frac{(\tilde{\mathbf{n}} \cdot \mathbf{h})}{\|\tilde{\mathbf{n}}\|(\tilde{\mathbf{n}} \cdot \tilde{\mathbf{m}})} \tag{70}$$

En donde:

$$\tilde{\mathbf{n}} = u\mathbf{n}_1 + v\mathbf{n}_2 + w\mathbf{n}_3$$

$$\tilde{\mathbf{m}} = \mathbf{r}_3 \times \mathbf{r}_1 = \mathbf{r}_1 \times \mathbf{r}_2 = \mathbf{r}_2 \times \mathbf{r}_3$$

$$\mathbf{h} = (\mathbf{n}_1 \times \mathbf{r}_1) + (\mathbf{n}_2 \times \mathbf{r}_2) + (\mathbf{n}_3 \times \mathbf{r}_3)$$

$$\mathbf{r}_1 = \mathbf{x}_3 - \mathbf{x}_2,\ \mathbf{r}_2 = \mathbf{x}_1 - \mathbf{x}_3,\ \mathbf{r}_3 = \mathbf{x}_2 - \mathbf{x}_1$$

3.5.1 Curvaturas principales

Los valores k_1, k_2 denominados curvaturas principales, los cuales miden respectivamente la máxima y mínima curvatura en punto de la superficie, están relacionados con la curvatura Gaussiana K y la curvatura media H de acuerdo a:

$$K = k_1 k_2 \tag{71}$$

$$H = \frac{1}{2}\left(k_1 + k_2\right) \tag{72}$$

$$k_{1,2} = H \pm \sqrt{H^2 - K},\ k_1 > k_2 \tag{73}$$

3.5.2 Índice de forma

El índice de forma (*Shape Index*) es una medida cuantitativa de la forma de una superficie en un punto (Koenderink and van Doorn, 1992, Dorai and Jain, 1997). El índice de forma S_I se define como:

$$S_I = \frac{1}{2} - \frac{1}{\pi}\tan^{-1}\left(\frac{k_1 + k_2}{k_1 - k_2}\right) \tag{74}$$

El intervalo del índice de forma es $\left[0,\ 1\right]$, cada forma distintiva corresponde a un valor único de S_I (Figura 33). Los puntos en un plano tienen un valor indeterminado de S_I debido a que $k_1 = k_2 = 0$.

Figura 33. Relación entre el índice de forma y distintas superficies clásicas. Extraído de (Poó et al., 2013).

3.5.3 Curvidad

La curvidad o cantidad de curvatura (*Curvedness*) es una medida de la energía de flexión en un punto de la superficie (Ho and Gibbins, 2009, Dorai and Jain, 1997). La curvidad R de un punto sobre una superficie se define como:

$$R = \sqrt{\frac{k_1^2 + k_2^2}{2}} \tag{75}$$

3.5.4 Índice de curvatura

El índice de curvatura (*Curvature Ratio*) es una medida de curvatura invariante a la escala con valores en el intervalo $0 \le k_3 \le 1$ (Rugis and Klette, 2006), definido como:

$$k_3 = \frac{\min\left(|k_1|, |k_2|\right)}{\max\left(|k_1|, |k_2|\right)} \tag{76}$$

En caso de que k_1, k_2 sean igual a cero, k_3 se define como cero.

3.6 REGISTRO

El registro es una parte importante del análisis morfológico, puesto que es necesario que todos los objetos de la población estudiada estén alineados hacia un mismo sistema de referencia, eliminando de esta forma las diferencias debido a las variaciones de postura por parte de los sujetos al momento de realizar la adquisición. En esta sección se expondrán a detalle los métodos de registro utilizados en este trabajo. El concepto de registro generaliza el de alineación, al incluir transformaciones no lineales, realizarse entre objetos quizás diferentes, y utilizar rasgos de referencia que se deben corresponder.

3.6.1 Registro Rígido

El primer método de registro que se utilizó fue el registro rígido, el cual consiste en aplicar una transformación rígida, formada por una rotación y una traslación, al conjunto de datos que se desea registrar, con el fin de minimizar el error entre puntos correspondientes. Este método ha sido utilizado por (Loïc, 2007), para el registro de mallas triangulares de cabezas 3D, reconstruidas a partir de las imágenes de profundidad, obtenidas mediante escáner láser en pacientes reales. El método de registro rígido más utilizado es el algoritmo iterativo del punto más cercano ICP (*Iterative Closest Point*), el cual, en cada iteración, se definen pares de puntos correspondientes, asociando a cada punto de una malla, el punto más cercano de la otra, basándose en la distancia euclidiana. En cada iteración se obtiene la transformación rígida óptima, para el conjunto actual de pares de puntos correspondientes. Luego el conjunto de datos a registrar es transformado, y el método se repite hasta que el error en la alineación, converge hacia un mínimo local, como se demuestra en (Besl and McKay, 1992). En nuestro caso, debido a que se conocen de entrada, los pares de puntos correspondientes en ambos modelos, el problema se reduce a encontrar la transformación rígida óptima, que minimice el promedio de la suma de las distancias al cuadrado, en una sola iteración.

Sea $\mathbf{P} \in \mathcal{M}_{n\times 3}\left(\mathbb{R}\right)$ la matriz formada por las filas $\mathbf{p}_i \in \mathbb{R}^3$ que representan los puntos a alinear con los puntos $\mathbf{x}_i \in \mathbb{R}^3$ que conforman las filas de la matriz $\mathbf{X} \in \mathcal{M}_{n\times 3}\left(\mathbb{R}\right)$, tal que cada punto $\mathbf{p}_i$ corresponde al punto $\mathbf{x}_i$ con $i \in \{1, \cdots, n\}$. Se desea encontrar la transformación rígida óptima $\mathbf{q} = \left(\mathbf{q}_R, \mathbf{q}_T\right)$, formada por el cuaternio unitario $\mathbf{q}_R \in \mathbb{R}^4$ y el vector de traslación $\mathbf{q}_T \in \mathbb{R}^3$, que minimice la función objetivo cuadrática:

$$f\left(\mathbf{q}\right) = \frac{1}{n}\sum_{i=1}^{n}\left|\mathbf{x}_i - \mathbf{R}\left(\mathbf{q}_R\right)\mathbf{p}_i - \mathbf{q}_T\right|^2 \tag{77}$$

En donde $\mathbf{R}\left(\mathbf{q}_R\right)$ es la matriz de rotación generada por el cuaternio unitario $\mathbf{q}_R = \begin{pmatrix} q_{\mathcal{W}} & q_{\mathcal{X}} & q_{\mathcal{Y}} & q_{\mathcal{Z}} \end{pmatrix}^{\mathrm{T}}$, con $q_{\mathcal{W}} \geq 0$ y $q_{\mathcal{W}}^2 + q_{\mathcal{X}}^2 + q_{\mathcal{Y}}^2 + q_{\mathcal{Z}}^2 = 1$, dada por:

$$\mathbf{R}(\mathbf{q}_R)=\begin{pmatrix} 1-2q_{\mathcal{Y}}^2-2q_{\mathcal{Z}}^2 & 2q_{\mathcal{X}}q_{\mathcal{Y}}-2q_{\mathcal{Z}}q_{\mathcal{W}} & 2q_{\mathcal{X}}q_{\mathcal{Z}}+2q_{\mathcal{Y}}q_{\mathcal{W}} \\ 2q_{\mathcal{X}}q_{\mathcal{Y}}+2q_{\mathcal{Z}}q_{\mathcal{W}} & 1-2q_{\mathcal{X}}^2-2q_{\mathcal{Z}}^2 & 2q_{\mathcal{Y}}q_{\mathcal{Z}}-2q_{\mathcal{X}}q_{\mathcal{W}} \\ 2q_{\mathcal{X}}q_{\mathcal{Z}}-2q_{\mathcal{Y}}q_{\mathcal{W}} & 2q_{\mathcal{Y}}q_{\mathcal{Z}}+2q_{\mathcal{X}}q_{\mathcal{W}} & 1-2q_{\mathcal{X}}^2-2q_{\mathcal{Y}}^2 \end{pmatrix} \tag{78}$$

Para encontrar la transformación $\mathbf{q}$ que minimice la función objetivo f, se emplea el método basado en la *matriz de covarianza cruzada* propuesto por (Besl and McKay, 1992), como se describe a continuación.

Sean $\boldsymbol{\mu}_{\mathbf{P}}$, $\boldsymbol{\mu}_{\mathbf{X}}$ los centros de masa correspondientes a los conjuntos de puntos que conforman las matrices $\mathbf{P}$, $\mathbf{X}$ respectivamente, dados por:

$$\boldsymbol{\mu}_{\mathbf{P}}=\frac{1}{n}\sum_{1}^{n}\mathbf{p}_i \tag{79}$$

$$\boldsymbol{\mu}_{\mathbf{X}}=\frac{1}{n}\sum_{1}^{n}\mathbf{x}_i \tag{80}$$

La matriz de covarianza cruzada $\boldsymbol{\Sigma}_{\mathbf{PX}}\in\mathcal{M}_3(\mathbb{R})$ para los conjuntos de puntos representados por las matrices $\mathbf{P}$, $\mathbf{X}$ se calcula como:

$$\boldsymbol{\Sigma}_{\mathbf{PX}}=\frac{1}{n-1}\left(\mathbf{P}-\mathbf{M}_{\mathbf{P}}\right)^{\mathrm{T}}\left(\mathbf{X}-\mathbf{M}_{\mathbf{X}}\right) \tag{81}$$

En donde $\mathbf{M}_{\mathbf{P}}\in\mathcal{M}_{n\times 3}(\mathbb{R})$, $\mathbf{M}_{\mathbf{X}}\in\mathcal{M}_{n\times 3}(\mathbb{R})$ son matrices cuyas filas son $\boldsymbol{\mu}_{\mathbf{P}}$, $\boldsymbol{\mu}_{\mathbf{X}}$ correspondientemente.

Los componentes cíclicos de la matriz anti-simétrica $\mathbf{A}=\boldsymbol{\Sigma}_{\mathbf{PX}}-\boldsymbol{\Sigma}_{\mathbf{PX}}^{\mathrm{T}}$ son usados para formar el vector $\boldsymbol{\Delta}=\begin{pmatrix}a_{23} & a_{31} & a_{12}\end{pmatrix}^{\mathrm{T}}$. Este vector es usado luego para formar la matriz simétrica $\mathbf{Q}(\boldsymbol{\Sigma}_{\mathbf{PX}})\in\mathcal{M}_4(\mathbb{R})$ definida como:

$$\mathbf{Q}(\boldsymbol{\Sigma}_{\mathbf{PX}})=\begin{pmatrix}\mathrm{tr}(\boldsymbol{\Sigma}_{\mathbf{PX}}) & \boldsymbol{\Delta}^{\mathrm{T}} \\ \boldsymbol{\Delta} & \boldsymbol{\Sigma}_{\mathbf{PX}}+\boldsymbol{\Sigma}_{\mathbf{PX}}^{\mathrm{T}}-\mathrm{tr}(\boldsymbol{\Sigma}_{\mathbf{PX}})\mathbf{I}_3\end{pmatrix} \tag{82}$$

En donde $\mathbf{I}_3\in\mathcal{M}_3(\mathbb{R})$ es la matriz identidad.

El vector propio $\mathbf{q}_R=\begin{pmatrix}q_{\mathcal{W}} & q_{\mathcal{X}} & q_{\mathcal{Y}} & q_{\mathcal{Z}}\end{pmatrix}^{\mathrm{T}}$ correspondiente al máximo valor propio de la matriz $\mathbf{Q}(\boldsymbol{\Sigma}_{\mathbf{PX}})$, es seleccionado como la rotación óptima. El vector de traslación óptimo está dado por:

$$\mathbf{q}_T=\boldsymbol{\mu}_{\mathbf{X}}-\mathbf{R}(\mathbf{q}_R)\boldsymbol{\mu}_{\mathbf{P}} \tag{83}$$

3.6.2 Registro Afín

Una transformación afín $\mathbf{T}=(\mathbf{A},\ \mathbf{b})$ consiste de una transformación lineal seguida de una traslación, representadas por la matriz $\mathbf{A}\in\mathcal{M}_3(\mathbb{R})$ y el vector $\mathbf{b}\in\mathbb{R}^3$. Las transformaciones afines satisfacen las siguientes propiedades:

1. Las relaciones de linealidad entre puntos se conservan.
2. Las razones entre distancias a lo largo de una línea se conservan.

Para el caso del registro afín, se desea encontrar la transformación lineal $\mathbf{A}$ y el vector de traslación $\mathbf{b}$, tales que minimicen la función objetivo cuadrática:

$$f(\mathbf{T})=\frac{1}{n}\sum_{i=1}^{n}\left|\mathbf{x}_i-\mathbf{A}\mathbf{p}_i-\mathbf{b}\right|^2 \tag{84}$$

Sean $\mathbf{p}'_i$ los puntos resultantes de aplicar la transformación afín $\mathbf{T}=(\mathbf{A},\ \mathbf{b})$ a cada punto original $\mathbf{p}_i$, obtenidos a partir de:

$$\mathbf{p}'_i=\mathbf{A}\mathbf{p}_i+\mathbf{b}=\begin{pmatrix} a_{11}p_{i_{\mathcal{X}}}+a_{12}p_{i_{\mathcal{Y}}}+a_{13}p_{i_{\mathcal{Z}}}+b_{\mathcal{X}} \\ a_{21}p_{i_{\mathcal{X}}}+a_{22}p_{i_{\mathcal{Y}}}+a_{23}p_{i_{\mathcal{Z}}}+b_{\mathcal{Y}} \\ a_{31}p_{i_{\mathcal{X}}}+a_{32}p_{i_{\mathcal{Y}}}+a_{33}p_{i_{\mathcal{Z}}}+b_{\mathcal{Z}} \end{pmatrix} \tag{85}$$

Si hacemos cada punto $\mathbf{p}_i$ igual a su correspondiente punto objetivo $\mathbf{x}_i$, obtenemos los siguientes sistemas de ecuaciones lineales, por cada eje del sistema de coordenadas, expresados matricialmente como:

$$\begin{pmatrix} \vdots & & & \\ p_{i_{\mathcal{X}}} & p_{i_{\mathcal{Y}}} & p_{i_{\mathcal{Z}}} & 1 \\ \vdots & & & \end{pmatrix}\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ b_x \end{pmatrix}=\begin{pmatrix} \vdots \\ x_{i_{\mathcal{X}}} \\ \vdots \end{pmatrix} \tag{86}$$

$$\begin{pmatrix} \vdots & & & \\ p_{i_{\mathcal{X}}} & p_{i_{\mathcal{Y}}} & p_{i_{\mathcal{Z}}} & 1 \\ \vdots & & & \end{pmatrix}\begin{pmatrix} a_{21} \\ a_{22} \\ a_{23} \\ b_y \end{pmatrix}=\begin{pmatrix} \vdots \\ x_{i_{\mathcal{Y}}} \\ \vdots \end{pmatrix} \tag{87}$$

$$\begin{pmatrix} \vdots & & & \\ p_{i_{\mathcal{X}}} & p_{i_{\mathcal{Y}}} & p_{i_{\mathcal{Z}}} & 1 \\ \vdots & & & \end{pmatrix}\begin{pmatrix} a_{31} \\ a_{32} \\ a_{33} \\ b_z \end{pmatrix}=\begin{pmatrix} \vdots \\ x_{i_{\mathcal{Z}}} \\ \vdots \end{pmatrix} \tag{88}$$

Estos sistemas son de la forma $\mathbf{MX} = \mathbf{Y}$, si el número de puntos es distinto de 4, los sistemas son sub-determinados o sobre-determinados, y en general se resuelven como:

$$\mathbf{X} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{Y} \tag{89}$$

De este modo se obtienen los coeficientes de la matriz $\mathbf{A}$ y del vector $\mathbf{b}$, y con ello la transformación afín óptima.

3.6.3 Registro Deformable

Para el registro deformable de modelos de la cabeza humana utilizamos el método de registro iterativo basado en jaulas (*Iterative Cage-based Registration*), propuesto por (Savoye, 2012, Savoye, 2013), diseñado para la alineación de un modelo de plantilla del cuerpo humano con una nube de puntos para la captura de poses (Figura 34). En este método la malla de referencia o plantilla es acoplada a una malla de control envolvente o jaula, con vértices $\mathbf{C} = \left\{\mathbf{c}_j\right\}$, mediante el uso de pesos bi-armónicos (*Biharmonic Weights*) (Jacobson, 2011). Cada vértice $\mathbf{v}_i$ de la malla de referencia es representado como un conjunto de pesos normalizados w_{ij} correspondiente a cada uno de los vértices $\mathbf{c}_j$ de la malla de control. La posición de los vértices $\tilde{\mathbf{v}}_i$ del modelo deformado, dada una configuración deformada de los vértices de la jaula $\tilde{\mathbf{c}}_i$, se obtienen como:

$$\tilde{\mathbf{v}}_i = \sum_{j=1}^{m} w_{ij}\tilde{\mathbf{c}}_j \tag{90}$$

En cada iteración $t+$ se inicializan los vértices de la jaula a las posiciones $\mathbf{c}^t$ obtenidas de la iteración anterior t. Luego se actualiza el operador laplaciano $\boldsymbol{\mathcal{L}}$ y las correspondientes coordenadas laplacianas $\boldsymbol{\delta}_j$, y se determina, de manera automática, el conjunto de pares de correspondencias $\boldsymbol{\mathcal{S}} = \left\{s_k : \left(k, \mathbf{q}_k, \gamma_k\right)\right\}$, en donde $\mathbf{q}_k$ es la posición objetivo del vértice k de la plantilla ponderada por el peso γ_k. Finalmente, la nueva pose de la jaula $\mathbf{c}^{t+}$ es estimada mediante la resolución de la función objetivo:

$$\underset{\left\{\mathbf{c}_1^{t+},\cdots,\mathbf{c}_m^{t+}\right\}}{\arg\min}\left(\alpha\sum_{j=1}^{m}\left|\boldsymbol{\mathcal{L}}^t\mathbf{c}_j^{t+} - \boldsymbol{\delta}_j^t\right|^2 + \beta\sum_{s_k\in\mathcal{S}^t}\gamma_k\left|\mathbf{q}_k - \sum_{j=1}^{m}w_{kj}\mathbf{c}_j^{t+}\right|^2\right) \tag{91}$$

En donde α^t es el término de distorsión, que modula la preservación de la forma de la malla de control, y β^t es el término de ajuste, que penaliza el error de alineación. El término de ajuste, inicializado en $\beta^0 = 0.01$, es incrementado en cada iteración siguiendo un crecimiento exponencial para promover las restricciones de los pares de correspondencias, mientras que el término de distorsión, inicializado en $\alpha^0 = 1$, decae en cada iteración para favorecer la flexibilidad de la malla de control.

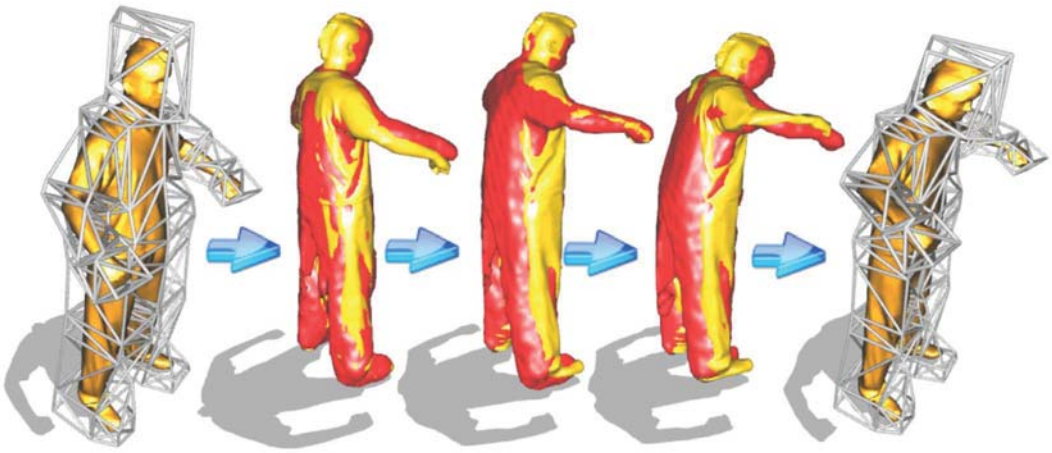


Figura 34. Una jaula humanoide es ajustada a una malla de referencia (amarillo) escaneada mediante laser, luego la malla de referencia es alineada con una malla con ruido (rojo) mediante la deformación de la jaula. Extraído de (Savoye, 2013).

Por simplicidad en nuestra implementación se realizaron las siguientes modificaciones al método original. Primero, en lugar de utilizar pesos bi-armónicos utilizamos coordenadas de valor medio, ya que son más fáciles de calcular al poseer forma cerrada discreta (Floater, 2003, Ju et al., 2005). Segundo, el conjunto de pares de correspondencias $\boldsymbol{\mathcal{S}}$ es el mismo en cada iteración y corresponde a pares de puntos de referencia correspondientes entre el modelo de referencia y el modelo de la adquisición, y se otorga el mismo peso $\gamma_k = 1$ a todos los pares s_k. Antes de aplicar el método de manera directa, primero se realiza un registro, ya sea rígido o afín, del conjunto de puntos de referencia del modelo de referencia hacia el conjunto de puntos de referencia sobre del modelo de la adquisición. Esto último debido a que las Coordenadas Laplacianas utilizadas para la deformación de la jaula no son invariantes a rotaciones. El método propuesto se ilustra en la (Figura 35).

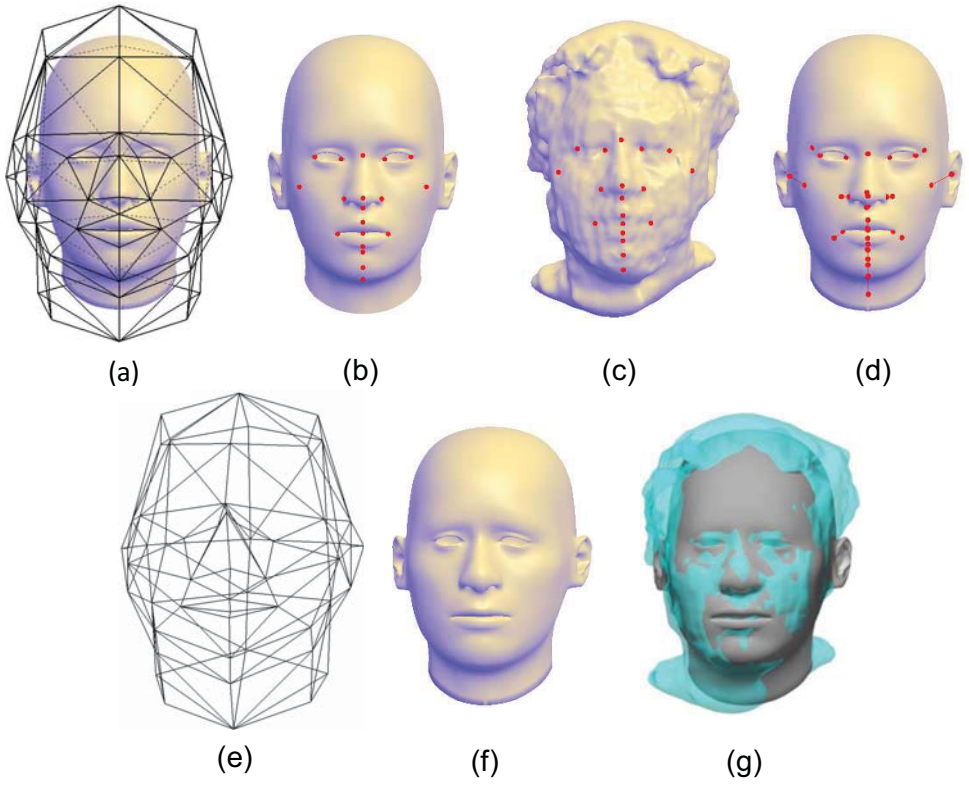


Figura 35. Malla de referencia de la cabeza humana con la jaula envolvente (a), el usuario selecciona puntos de referencia correspondientes sobre la malla de referencia (b) y la malla de la adquisición (c), se realiza un registro rígido o afín de los puntos de referencia del modelo de referencia hacia los puntos de referencia del modelo de la adquisición (d), finalmente se realiza el registro iterativo basado en jaulas para obtener la jaula deformada (e) y el modelo de referencia deformado (f). Modelos superpuestos tras el registro (g).

La solución de la función objetivo para cada iteración se obtiene siguiendo el esquema propuesto en (Su, 2010), mediante la resolución por mínimos cuadrados del sistema lineal disperso:

$$\begin{pmatrix} \alpha^t \boldsymbol{\mathcal{L}}^t \\ \beta^t \boldsymbol{\mathcal{W}} \end{pmatrix} \mathbf{C}^{t+} = \begin{pmatrix} \alpha^t \boldsymbol{\Delta}^t \\ \beta^t \mathbf{Q} \end{pmatrix} \tag{92}$$

En donde $\mathbf{C} \in \mathcal{M}_{m\times 3}(\mathbb{R})$ corresponde a la matriz variable cuyas filas son las posiciones de los m vértices de la malla de control, $\mathbf{Q} \in \mathcal{M}_{k\times 3}(\mathbb{R})$ es una matriz cuyas filas son las coordenadas de los k vértices objetivos, $\boldsymbol{\mathcal{L}} \in \mathcal{M}_{m}(\mathbb{R})$ es el operador laplaciano de la malla de control, $\boldsymbol{\mathcal{W}} \in \mathcal{M}_{k\times m}(\mathbb{R})$ es la matriz con las coordenadas de valor medio del conjunto de vértices de origen de los k pares de correspondencias asociadas a cada uno de los m vértices de la malla de control, y $\boldsymbol{\Delta} \in \mathcal{M}_{m\times 3}(\mathbb{R})$ es la matriz cuyas filas corresponden a las coordenadas laplacianas de los m vértices de la malla de control. Lo cual genera un sistema lineal sobre-determinado, por cada eje del sistema de coordenadas, de m variables y $m+k$ ecuaciones, de la forma:

$$\mathbf{A}^t \mathbf{C}_{\mathcal{X}}^{t+} = \mathbf{B}_{\mathcal{X}}^t \tag{93}$$

$$\mathbf{A}^t\mathbf{C}_{\mathcal{Y}}^{t+} = \mathbf{B}_{\mathcal{Y}}^t \tag{94}$$

$$\mathbf{A}^t\mathbf{C}_{\mathcal{Z}}^{t+} = \mathbf{B}_{\mathcal{Z}}^t \tag{95}$$

Para la solución de los sistemas de ecuaciones se utilizó la biblioteca QR_SOLVE[1] implementada en lenguaje C++.

Para determinar los valores correspondientes a los pesos de control α^t, β^t en cada iteración, se utilizan las reglas de actualización obtenidas experimentalmente en (Savoye, 2013):

$$\alpha^t = 0.99e^{-0.001t} \tag{96}$$

$$\beta^t = e^{0.01t} - 1 \tag{97}$$

El valor de α^t es ajustado entre 0.85 y 1.0, y el valor de β^t es ajustado entre 0.0 y 0.8, para mantener el término de distorsión por encima del término de ajuste con el objetivo de priorizar la preservación de la forma.

3.7 Análisis Estadístico de los Puntos de Referencia del Rostro

3.7.1 Análisis de Componentes Principales

Para determinar la varianza de los puntos de referencia del rostro humano en la población estudiada se realiza un análisis de componentes principales (PCA) basado en la matriz de covarianza (Smith, 2002). Este análisis se realiza una vez que los 19 puntos de referencia de cada una de las 35 cabezas de la población de estudio han sido alineados, mediante un registro rígido, con los 19 puntos correspondientes de la cabeza de referencia. Con lo cual se obtienen 19 nubes de puntos, cada una formada por puntos de referencia correspondientes en las 35 cabezas adquiridas.

Sea $\mathbf{X} \in \mathcal{M}_{n\times 3}(\mathbb{R})$ la matriz formada por las filas $\mathbf{x}_i = (x_i, y_i, z_i)$ correspondientes al mismo punto de referencia de la i-ésima cabeza de la población. Se desea encontrar las tres direcciones principales de variación de las n muestras y la varianza correspondiente a cada dirección.

Primero se obtiene la media de los datos $\boldsymbol{\mu}$ como:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{1}^{n}\mathbf{x}_i \tag{98}$$

La matriz de covarianza $\boldsymbol{\Sigma}$ está dada por:

$$\boldsymbol{\Sigma} = \frac{1}{n-1}(\mathbf{X}-\mathbf{M})^{\mathrm{T}}(\mathbf{X}-\mathbf{M}) \tag{99}$$

En donde $\mathbf{M} \in \mathcal{M}_{n\times 3}(\mathbb{R})$ es una matriz compuesta por las filas $\boldsymbol{\mu}$.

[1] https://people.sc.fsu.edu/~jburkardt/cpp_src/qr_solve/qr_solve.html

Finalmente se realiza la descomposición en vectores propios de la matriz de covarianza, de acuerdo la siguiente ecuación:

$$\boldsymbol{\Sigma}\hat{\mathbf{v}}_i = \lambda_i \quad \hat{\mathbf{v}}_i \; i \in \{1,2,3\} \tag{100}$$

Los vectores propios unitarios $\hat{\mathbf{v}}_i$ corresponden a las direcciones principales en que la nube de puntos varía espacialmente, y la magnitud respectiva representa la varianza en esa misma dirección. El análisis de componente principales se implementó utilizando el software MATLAB.

3.7.2 Error Técnico de Medición

Para evaluar el error en las medidas antropométricas se utiliza el error técnico de medición (TEM, por sus siglas en inglés), definido por:

$$\mathrm{TEM} = \sqrt{\frac{\sum_{k=1}^{n} d_k^2}{2n}} \tag{101}$$

En donde d_k es la diferencia entre la misma medición realizada por el mismo observador k veces (o la misma medición realizada una vez por k observadores diferentes), y n es el número de sujetos evaluados (Frisancho, 1990). El análisis del error técnico de medición se implementó utilizando el software MATLAB.

4 IMPLEMENTACIÓN

En este capítulo se expondrá en detalle la implementación del software destinado a la selección interactiva de puntos de referencia, haciendo énfasis en los requerimientos, las estructuras de datos y la interfaz gráfica de usuario. Para la visualización científica se implementó el modelo de iluminación de Gooch (Gooch et al., 1998) y la visualización mediante mapa de color divergente (Moreland, 2009) utilizando el lenguaje sombreador GLSL[1]. Para la selección interactiva se implementó el algoritmo estanco para la intersección entre un rayo y un triángulo propuesto por (Woop et al., 2013), mientras que para la intersección entre un rayo y una caja envolvente alineada con los ejes se utilizó el algoritmo propuesto en (Akenine-Möller, 2005), cuya implementación está disponible en (Akenine-Möller, 2001). Para el control del dispositivo háptico Phantom se utilizó la biblioteca OpenHaptics[2] siendo además necesario instalar los controladores proporcionados por el fabricante[3]. Para la deformación de mallas triangulares se implementó el método de coordenadas laplacianas, propuesto en (Alexa, 2003), y el método de deformación basado en jaulas utilizando coordenadas de valor medio como se detalla en (Ju et al., 2005, Floater et al., 2005). Para aproximar las curvaturas de las mallas se implementó el método propuesto en (Theisel et al., 2004). Para el registro rígido se implementó el método propuesto en (Besl and McKay, 1992), mientras que para el registro afín se implementó el método detallado en la sección 3.6.2. Para el registro deformable se implementó el método iterativo de registro deformable basado en jaulas propuesto por (Savoye, 2012), mientras que para la resolución del sistema lineal disperso resultante se empleó la biblioteca QR_SOLVE[4]. Finalmente el análisis de los puntos de referencia y del error técnico de medición (Frisancho, 1990) se realizó por separado utilizando el software MATLAB[5].

4.1 REQUERIMIENTOS

Uno de los objetivos principales de este trabajo fue desarrollar un software para la edición de los puntos de referencia sobre modelos 3D de rostros humanos. Este software nos permite, además de extraer los puntos de referencia correspondientes a los rasgos faciales, realizar mediciones lineales y angulares entre puntos, así como trazar caminos y contornos sobre las mallas 3D para obtener longitudes, perímetros y áreas. Esto es con el fin de emular, de manera virtual, los procedimientos de medición utilizados por la antropometría directa (Kolar and Salter, 1997).

Los requerimientos del software son:

- Importar modelos 3D desde varios formatos comunes.
- Visualizar el mallado utilizando varios estilos de renderizado.
- Realizar la selección interactiva de los puntos de referencia en tiempo real.
- Definir distancias lineales entre puntos de referencia.
- Definir ángulos formados por puntos de referencia.
- Trazar caminos y contornos sobre el mallado, para obtener longitudes, perímetros y área.

[1] https://www.opengl.org/
[2] http://www.geomagic.com/es/products/open-haptics/overview/
[3] http://www.dentsable.com/support-download-pdd.htm
[4] https://people.sc.fsu.edu/~jburkardt/cpp_src/qr_solve/qr_solve.html
[5] https://www.mathworks.com/products/matlab/

- Establecer índices como proporciones entre distancias.
- Asignar un nombre, nemónico (nombre corto o abreviatura) y una descripción a cada punto de referencia, distancia, ángulo, camino, contorno e índice.
- Visualizar utilizando mapas de color las curvaturas de la malla en forma local.
- Respaldar en un archivo los puntos y mediciones realizadas por el usuario para su posterior edición y análisis.

Además de los requerimientos anteriores, el software debe ser eficiente, es decir, debe funcionar en tiempo real con mallas de cualquier complejidad, debe ser robusto ante configuraciones de mallados con ruido y triángulos degenerados, debe ser flexible, con lo que puede aplicarse para el estudio de otros tipos de geometría y no sólo limitarse al análisis de rostros, y, por último, debe ser intuitivo para el usuario.

Considerando estos requerimientos, el software para la edición de puntos de referencia fue desarrollado en lenguaje C++, utilizando el entorno de desarrollo integrado Visual Studio 2013. Para la implementación de las herramientas computacionales se utilizó el paradigma de Programación Orientada a Objetos. En el presente trabajo las clases serán nombradas en el idioma inglés, utilizando el estilo de escritura *CamelCase*, y añadiendo el prefijo T para denotar que se trata de un tipo de dato. Los atributos de las clases serán nombrados de igual forma, pero utilizando el prefijo m para denotar que se trata de un miembro de la clase.

Para representar las clases y las dependencias entre clases, se emplearon diagramas de clases utilizando el Lenguaje Unificado de Modelado (UML) utilizando el software Visual Paradigm. El diagrama de clases del proyecto para la edición de puntos de referencia se muestra en la (Figura 36).

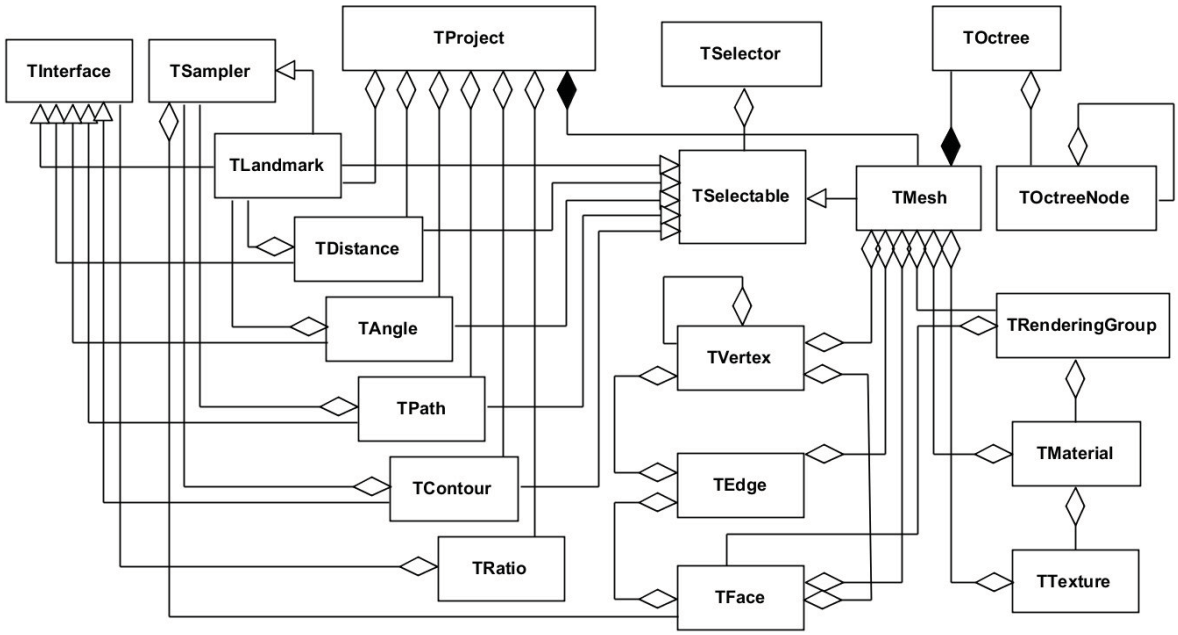


Figura 36. Diagrama de clases del proyecto para la edición de puntos de referencia.

4.2 ESTRUCTURAS DE DATOS

4.2.1 Clase TProject

La clase TProject, es la clase principal del software para la edición de puntos de referencia, ya contiene la información sobre el estado actual del proyecto, representado por la malla 3D y los puntos de referencia y mediciones realizadas por el usuario. Esta clase posee métodos para guardar la sesión de trabajo en un archivo, así como para cargar los datos de un proyecto realizado previamente. El diagrama de clase de la clase TProject se muestra en la (Figura 37). El diagrama de dependencias de clases de la clase TProject se muestra en la (Figura 38).

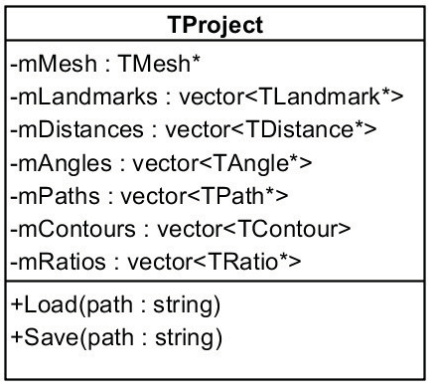


Figura 37. Diagrama de clase de la clase TProject

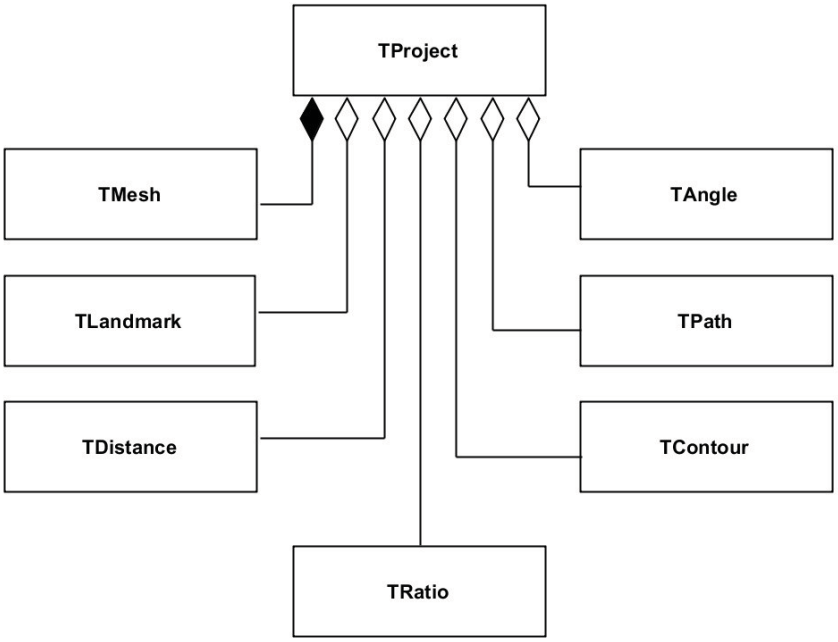


Figura 38. Diagrama de dependencias de clases de la clase TProject

4.2.2 Clase TSelector y TSelectable

La clase TSelectable representa un objeto seleccionable en la escena, es una clase abstracta, por lo tanto, no se puede instanciar. Los objetos derivados de la clase TSelectable deben implementar el método RayIntersection, el cual recibe como argumento el origen y dirección normalizada del rayo,

y devuelve verdadero en caso de que el rayo intersecte con el objeto, almacenando el punto de intersección en el atributo mIntersection. La clase selector mantienen una lista de todos los objetos seleccionables en la escena, y cuando el método RayIntersection es invocado, utilizando como argumento el origen y dirección de un rayo sobre la escena, se verifica cada objeto seleccionable contra el rayo, y el objeto con el punto de intersección más cercano al origen del rayo, es asignado a la variable mSelected. El diagrama de clase de la clase TSelector y TSelectable se muestra en la (Figura 39).

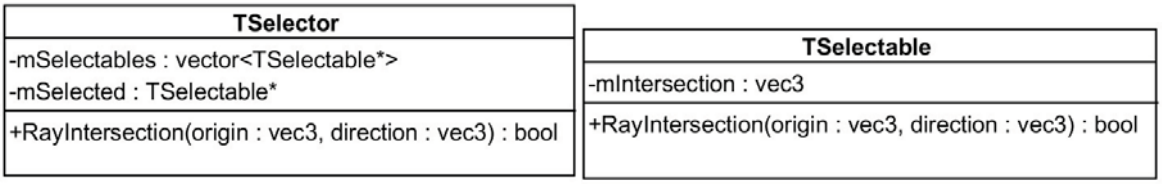


Figura 39. Diagrama de clase de la clase TSelector y de la clase TSelectable.

En cada evento de movimiento del puntero del ratón sobre el área de renderizado, se obtiene el origen y dirección normalizada del rayo correspondiente, proyectado dentro de la escena, los cuales son pasados como argumento al método RayIntersection de la clase TSelector, si el resutado es verdadero, significa que el puntero del ratón está sobre un objeto seleccionable de la escena. Dependiendo del objeto seleccionado actualmente y anteriormente, y del estado de los botones del ratón se desencadenan ciertos eventos, los cuales son resueltos por funciones de devolución de llamada (*Callback*) asociadas por cada evento a cada objeto seleccionable.

Los eventos son:

- OnEnter. Cuando el rayo proyectado a través del puntero del ratón, colisiona por primera vez con el objeto (Figura 40 a).
- OnMove. Cuando el puntero del ratón se mueve sin que se presione ningún botón del ratón, y el rayo correspondiente colisiona con el mismo objeto (Figura 40 b).
- OnExit. Cuando el rayo proyectado a través del puntero del ratón deja de colisionar con el objeto luego de ser desplazado por la pantalla (Figura 40 c).
- OnClick. Cuando el rayo proyectado a través del puntero del ratón está colisionando con el objeto y el usuario realiza un clic con alguno de los botones del mouse.
- OnDoubleClick. Cuando el rayo proyectado a través del puntero del ratón está colisionando con el objeto y el usuario realiza doble clic con alguno de los botones del mouse.
- OnEnterDrag. Cuando el usuario empieza a arrastrar el puntero del ratón y el rayo correspondiente colisiona con el objeto (Figura 40 d).
- OnDragMove. Cuando el usuario mueve el puntero del ratón mientras mantiene oprimido algún botón del ratón, y el rayo correspondiente se mantiene colisionando con el objeto (Figura 40 e).
- OnDrop. Cuando el usuario libera el botón del ratón tras haber arrastrado el puntero del ratón mientras el rayo correspondiente colisionaba con el objeto (Figura 40 f).

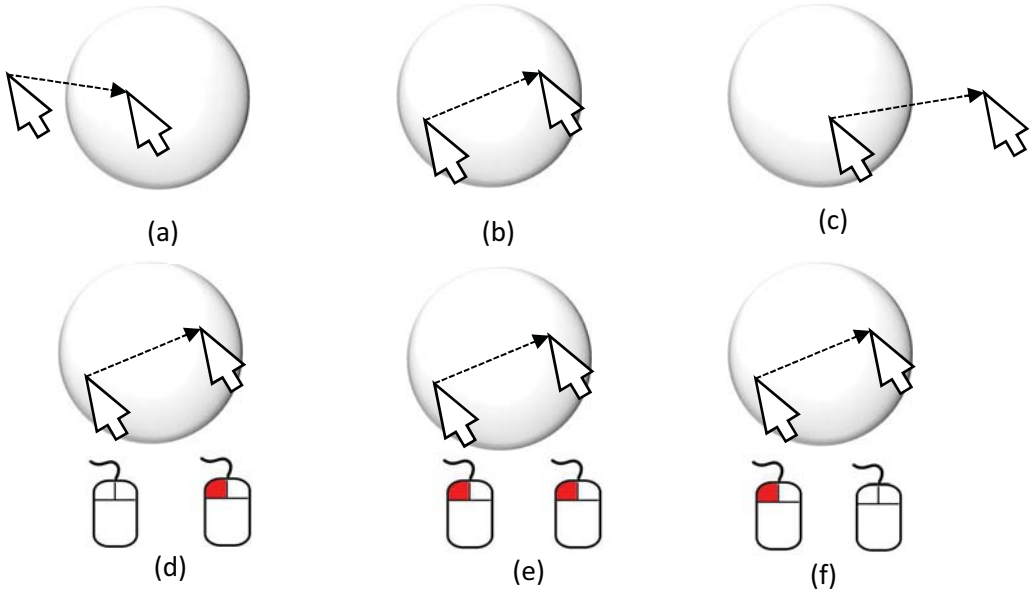


Figura 40. Algunos eventos desencadenados por la clase TSelector dependiendo del movimiento del puntero del ratón y del estado de los botones del ratón: OnEnter (a), OnMove (b), OnExit (c), OnEnterDrag (d), OnDragMove (e) y OnDrop (f).

Las clases que se derivan de la clase TSelectable son las clases TLandmark, TDistance, TAngle, TPath, TContour y TMesh, con lo cual se le permite al usuario interactuar con los puntos de referencia, distancias, ángulos, caminos, contornos y con la malla triangular misma. Cada una de estas clases implementa el método RayIntersection según la representación gráfica que los objetos de cada clase proyectan sobre la pantalla al ser renderizados. El diagrama de dependencias de clases para la clase TSelectable se muestra en la (Figura 41).

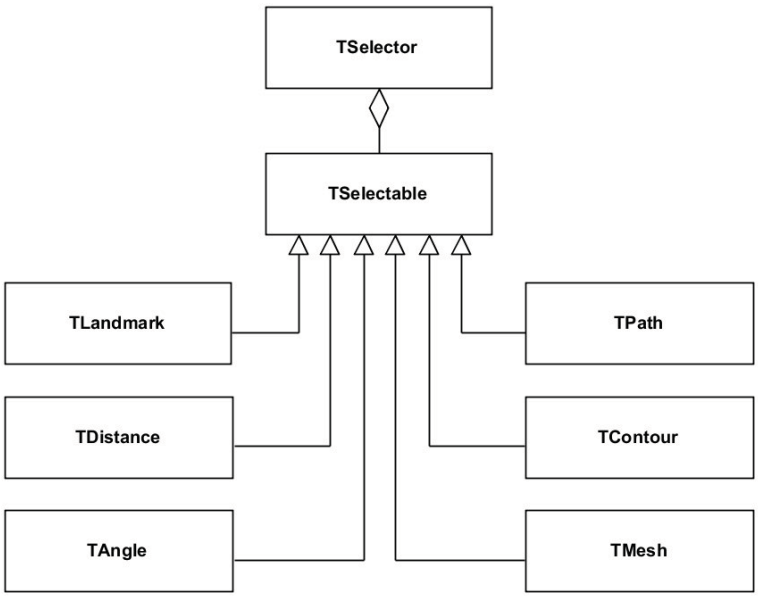


Figura 41. Diagrama de dependencias de clases de la clase TSelectable.

4.2.3 Clase TSampler y TLandmark

La clase TSampler se llama así porque se utiliza para muestrear la superficie de la malla en un punto, esto se logra almacenando un apuntador hacia alguno de los triángulos de la malla, y las coordenadas baricéntricas del punto correspondiente al interior del triángulo, permitiendo de esta forma interpolar los atributos de los vértices que forman el triángulo. Sean u, v, w las coordenadas baricéntricas de un punto al interior del triángulo, y sean f_1, f_2, f_3 los atributos correspondientes a los vértices del triángulo, el atributo interpolado se obtiene como:

$$\tilde{f} = uf_1 + vf_2 + wf_3 \tag{102}$$

Con la observación de que $u + v + w = 1$. Los atributos que se pueden interpolar son la posición, el vector normal, el color, la curvatura gaussiana, la curvatura media, el índice de curvatura, la curvidad y el índice de curvatura. La clase TSampler implementa métodos para interpolar cada uno de estos atributos.

La clase TLandmark corresponde a un punto de referencia sobre la malla triangular, se deriva de la clase TSampler con lo cual hereda todos sus métodos y atributos. De esta forma, en lugar de almacenar explícitamente la posición del punto de referencia, se toma un punto de muestra sobre la superficie de la malla, con lo cual, aunque la malla se deforme o se transforme, el punto de referencia permanecerá sobre la misma zona de la malla en que fue colocado, adicionalmente, se obtiene información adicional a la posición del punto de referencia, como lo es el vector normal, el color y las curvaturas. El diagrama de clases de la clase TSampler y de la clase TLandmark se muestra en la (Figura 42).

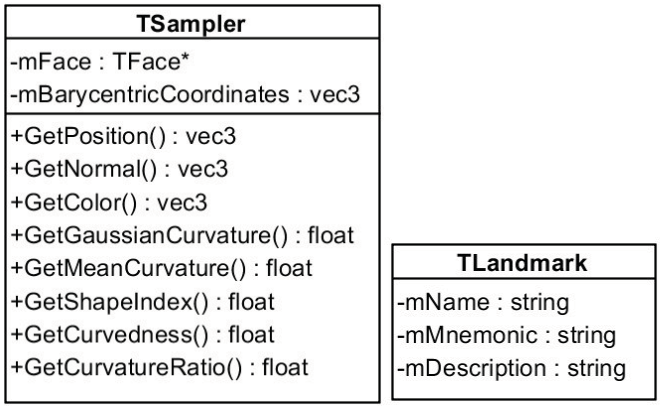


Figura 42. Diagramas de clase de la clase TSampler y de la clase TLandmark.

La clase TLandmark se deriva también de la clase TSelectable, lo cual permite que el usuario interactué con la representación gráfica del punto de referencia, y de la clase TInterface, que como se verá más adelante, permite a la clase TRatio obtener los valores de las propiedades de los objetos de la clase TLandmark para establecer proporciones. En diagrama de dependencia de clases para la clase TLandmark se muestra en la (Figura 43).

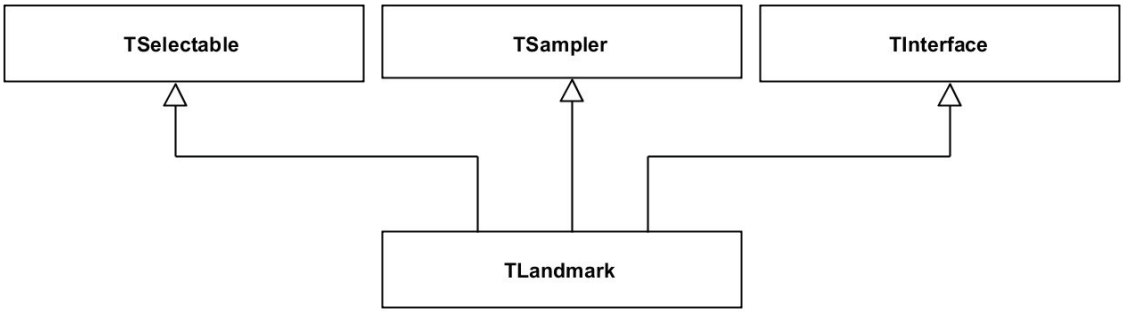


Figura 43. Diagrama de dependencias de clases de la clase TLandmark

El punto de referencia se representa en la escena como una esfera con centro en la posición del punto muestra y el radio se calcula, dependiendo del estado de la cámara virtual, para que siempre proyecte el mismo tamaño sobre la pantalla. Para ello existen dos casos, dependiendo de la proyección de la cámara:

Proyección en perspectiva. El radio de la esfera r se calcula como:

$$r = \frac{s}{e}\left|\mathbf{c} - \mathbf{e}\right| \tag{103}$$

En donde s es el radio en pixeles de la esfera proyectada en la pantalla, e es la distancia focal de la cámara virtual, $\mathbf{c}$ es la posición del centro de la esfera en coordenadas globales, y $\mathbf{e}$ es la posición de la cámara en coordenadas globales.

Proyección ortográfica: El radio de la esfera r está dado por:

$$r = \frac{sd}{\zeta e} \tag{104}$$

En donde ζ es el factor de escala o zoom de la cámara, y d es la distancia de la cámara al punto de interés o pivote.

El método RayIntersection de la clase TLandmark, realiza la prueba de intersección entre el rayo y la esfera que representa visualmente el punto de referencia en la pantalla. Considérese el rayo con origen en $\mathbf{p}$ y vector de dirección unitaria $\hat{\mathbf{d}}$, y la esfera con centro en $\mathbf{c}$ y radio r. Sea $\mathbf{x}$ un punto de la superficie de la esfera, la ecuación vectorial de la esfera es:

$$\left(\mathbf{x} - \mathbf{c}\right)^2 - r^2 = 0 \tag{105}$$

La intersección del rayo con la esfera se obtiene sustituyendo $\mathbf{x} = \mathbf{p} + t\hat{\mathbf{d}}$ y resolviendo para t, generándose la siguiente ecuación cuadrática:

$$
\begin{aligned}
&at^2 + bt + c = 0 \\
&a = \hat{\mathbf{d}}^2 = 1 \\
&b = 2\left(\mathbf{p} - \mathbf{c}\right) \cdot \hat{\mathbf{d}} \\
&c = \left(\mathbf{p} - \mathbf{c}\right)^2 - r^2
\end{aligned}
\tag{106}
$$

Si el discriminante es negativo, entonces el rayo no intersecta a la esfera. Si el discriminante es cero, el rayo es tangente a la esfera, y por convención, diremos que no la intersecta. En otro caso, tendremos dos valores de t, si ambos son negativos, entonces la esfera está detrás del rayo y no es intersectada. Si todas estas pruebas fallan, entonces el rayo intersecta a la esfera.

4.2.4 Clase TDistance y TAngle

Las clases TDistance y TAngle, son las que nos permiten obtener medidas lineales y angulares, respectivamente, sobre los puntos de referencia, ambas clases poseen atributos de nombre, nemónico y descripción, mismos que pueden ser editados por el usuario a través de la interface gráfica, el diagrama de clases para la clase TDistance y TLandmark se presenta en la (Figura 44). La clase TDistance mantiene dos apuntadores a dos objetos de la clase TLandmark distintos. Al invocar el método GetDistance, se consulta las posiciones actuales de los puntos de referencia respectivos y se obtiene la distancia lineal entre ellos. De esta forma, al no guardar de manera explícita la distancia entre los puntos, el valor de la distancia cambia conforme los puntos de referencia son trasladados. La clase TAngle mantiene apuntadores a tres objetos de la clase TLandmark distintos. Al invocar el método GetAngle, se consultan las posiciones actuales de los puntos de referencia correspondientes y se calcula en ángulo formado por los tres puntos, tomando al punto central como vértice del ángulo. Sean p_1, p_2, p_3 las posiciones actuales de los puntos de referencia que forman el ángulo $\measuredangle p_1 p_2 p_3$, la medición del ángulo θ está dada por:

$$
\theta = \cos^{-1} \frac{\left(p_1 - p_2\right) \cdot \left(p_3 - p_2\right)}{\left|p_1 - p_2\right| \left|p_3 - p_2\right|}
\tag{107}
$$

El valor del ángulo obtenido con este método está siempre en el intervalo $0 \le \theta \le \pi$, que corresponde al menor ángulo medido en $\measuredangle p_1 p_2 p_3$.

TDistance
-mName : string -mMnemonic : string -mDescription : string -mLandmarks : TLandmark*[2]
+GetDistance() : float

TAngle
-mName : string -mMnemonic : string -mDescription : string -mLandmarks : TLandmark*[3]
+GetAngle() : float

Figura 44. Diagramas de clase de la clase TDistance y de la clase TAngle.

Las clases TDistance y TAngle se derivan de la clase TSelectable, con lo cual el usuario puede interactuar con la representación gráfica de las distancias y ángulos en la pantalla, y de la clase TInterface, lo cual permite a los objetos de la clase TRatio acceder a las propiedades de las distancias

y ángulos para establecer índices. El diagrama de dependencias de clases para las clases TDistance y TSelectable se muestra en la (Figura 45).

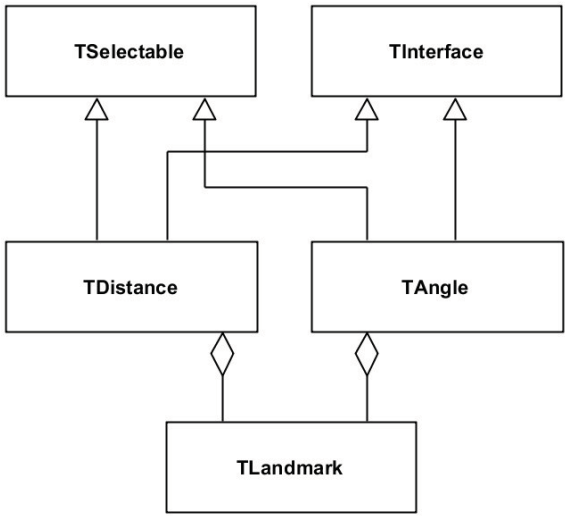


Figura 45. Diagrama de dependencias de clases de las clases TDistance y TAngle.

Las distancias son representadas en pantalla como una línea, con determinado grosor en pixeles, que une los dos puntos de referencia correspondientes. En el método RayIntersection implementado por la clase TDistance, se efectúa la prueba de intersección entre un rayo y un segmento, para determinar si el rayo correspondiente, a la posición actual del puntero del ratón en la pantalla, está colisionando con el segmento que representa la distancia. Considérese el rayo $\mathbf{p}_1 + t_1\hat{\mathbf{d}}_1$, y el segmento de recta formado por los puntos $\mathbf{a},\ \mathbf{b}$. El segmento puede verse como un rayo $\mathbf{p}_2 + t_2\hat{\mathbf{d}}_2,\ t_2 \in [0,\ l]$ con origen en $\mathbf{p}_2 = \mathbf{a}$ y dirección unitaria $\hat{\mathbf{d}}_2 = (\mathbf{b} - \mathbf{a})/l$, con $l = |\mathbf{b} - \mathbf{a}|$ la longitud del segmento. Se desea encontrar en cada rayo el punto más cercano al otro rayo, si los rayos realmente se intersectan, el punto en ambos rayos será el mismo, de lo contrario tendremos dos puntos distintos:

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{p}_1 + t_1\hat{\mathbf{d}}_1 \\ \mathbf{x}_2 &= \mathbf{p}_2 + t_2\hat{\mathbf{d}}_2 \end{aligned} \tag{108}$$

Como la distancia más cercana entre dos rectas se mide sobre la línea que es perpendicular a ambas, se tiene que:

$$\begin{aligned} (\mathbf{x}_1 - \mathbf{x}_2)\cdot\hat{\mathbf{d}}_1 &= 0 \\ (\mathbf{x}_1 - \mathbf{x}_2)\cdot\hat{\mathbf{d}}_2 &= 0 \end{aligned} \tag{109}$$

Sustituyendo (108) en (109) se obtiene el siguiente sistema de ecuaciones:

$$\begin{aligned} t_1 - t_2\hat{\mathbf{d}}_1\cdot\hat{\mathbf{d}}_2 &= (\mathbf{p}_2 - \mathbf{p}_1)\cdot\hat{\mathbf{d}}_1 \\ t_1\hat{\mathbf{d}}_1\cdot\hat{\mathbf{d}}_2 - t_2 &= (\mathbf{p}_2 - \mathbf{p}_1)\cdot\hat{\mathbf{d}}_2 \end{aligned} \tag{110}$$

Escribiendo (110) en forma matricial y resolviendo para $t_1,\ t_2$ obtenemos:

$$\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} 1 & -\hat{\mathbf{d}}_1 \cdot \hat{\mathbf{d}}_2 \\ \hat{\mathbf{d}}_1 \cdot \hat{\mathbf{d}}_2 & -1 \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{p}_2 - \mathbf{p}_1) \cdot \hat{\mathbf{d}}_1 \\ (\mathbf{p}_2 - \mathbf{p}_1) \cdot \hat{\mathbf{d}}_2 \end{pmatrix} \tag{111}$$

Cuando los dos rayos son paralelos el determinante de la matriz es cero, en ese caso se considera que no hay intersección. La intersección del rayo con el segmento ocurre sólo si $(t_1 > 0) \wedge (0 \le t_2 \le l)$ y si la distancia mínima entre los rayos es menor que un umbral.

Los ángulos son representados en pantalla como dos segmentos con un punto en común, que unen los respectivos puntos de referencia, y una curva indicando el ángulo en donde se realiza la medición. En el método RayIntersection de la clase TAngle, el área contenida dentro dela curva que representa el ángulo se aproxima mediante un abanico formado por diez triángulos con un vértice en común (Figura 46). Luego se realiza la prueba de colisión del rayo con cada uno de los triángulos, utilizando el algoritmo del (**¡Error! No se encuentra el origen de la referencia.**). La interacción del rayo proyectado a través de la posición actual del puntero del ratón en la pantalla, sólo ocurre sobre esta área.

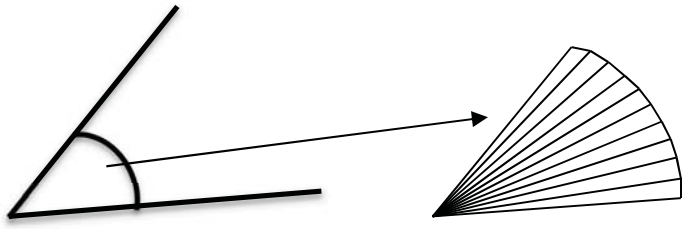


Figura 46. Representación de un ángulo como un abanico de triángulos para la detección de colisión con un rayo.

4.2.5 Clase TPath y TContour

Las clases TPath y TContour son dos clases muy similares, la primera permite al usuario trazar caminos sobre la malla 3D y medir su longitud, como una aproximación de la distancia geodésica, mientras que la clase TContour permite al usuario trazar contornos sobre la malla, para obtener perímetros o áreas. La diferencia entre un camino y un contorno, es que le primero es abierto, mientras que el segundo es cerrado y el último punto trazado se une con el primero para crear el contorno. Los caminos y contornos están compuestos por una lista de objetos de la clase TSampler, con lo cual se evita tener de manera explícita las posiciones de los puntos que forman los caminos o los contornos, permitiendo conservar la configuración de los caminos y contornos aun cuando la malla se transforme o sea deformada, y realizar mediciones de propiedades de la malla, como normales y curvatura, a lo largo de los caminos y de los contornos, accediendo a esta información a través de los puntos de muestra. Los diagramas de clase para la clase TPath y TContour se muestran en la (Figura 47).

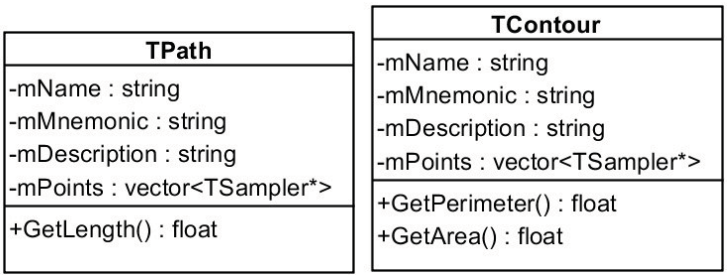


Figura 47. Diagramas de clase de la clase TPath y de la clase TContour.

Las clases TPath y TContour se derivan de la clase TSelectable, con lo cual se le permite al usuario interactuar con las representaciones gráficas de los caminos y contornos en la pantalla, así como de la clase TInterface, para que los objetos de la clase TRatio puedan definir índices entre propiedades de las clases TPath y TContour. El diagrama de dependencias de clases de las clases TPath y TContour se muestran en la (Figura 48). Los caminos y contornos son representados de manera gráfica como polilíneas, proyectadas en la pantalla con un determinado grosor en pixeles. En los métodos RayIntersection de ambas clases se realiza la intersección del rayo, correspondiente a la posición actual del puntero del ratón sobre la pantalla, con cada uno de los segmentos que forman los caminos y los contornos, de manera similar a como se realiza la intersección entre un rayo y una distancia.

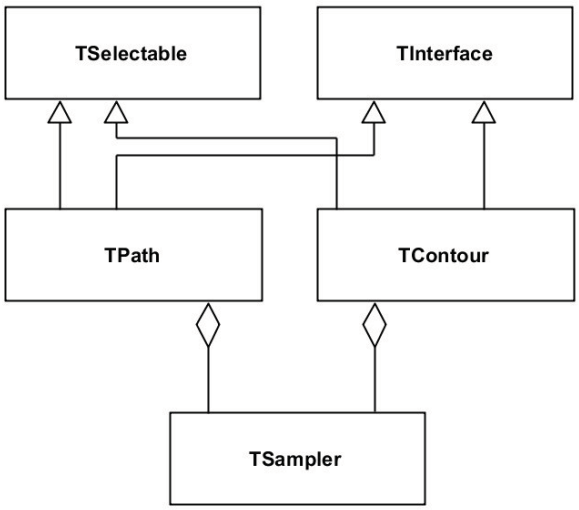


Figura 48. Diagrama de dependencias de clases de las clases TPath y TContour.

Para obtener la longitud de un camino o el perímetro de un contorno, trazados sobre la superficie de la malla, se suman las longitudes de cada uno de los segmentos formados por dos puntos consecutivos que conforman el camino o el contorno, en el caso del contorno, existe un segmento que une el primer punto con el último punto en la lista. Para obtener el área de la superficie de la malla contenida al interior del contorno, se utiliza un método aproximado. Primero se obtiene el punto correspondiente al centroide de los puntos que conforman el contorno, obtenido como el promedio de las coordenadas, luego el punto es trasladado al punto más cercano sobre la malla, finalmente el área se aproxima sumando el área de cada uno de los triángulos, formados por dos puntos consecutivos dentro del contorno y el centroide desplazado, el procedimiento se ilustra en

la (Figura 49). El área obtenida con este método es sólo una aproximación muy burda, pero fácil de computar, y es lo suficientemente buena cuando la curvatura de la malla dentro del contorno tiende a cero, y resulta ser exacto cuando se mide sobre un plano. Por simplicidad, no se consideran los casos cuando el contorno se auto-intersecta.

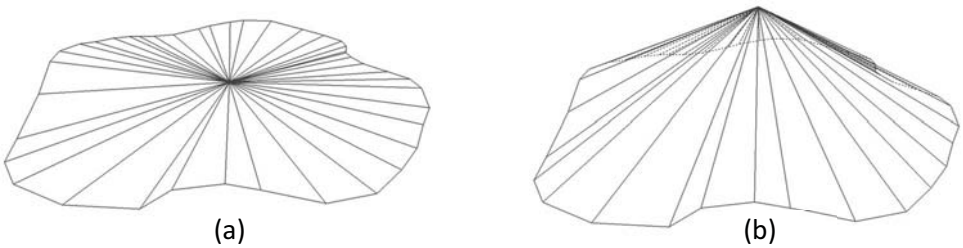


Figura 49. Para aproximar el área dentro de un contorno sobre la superficie de la malla, primero se obtiene el centroide y se triangula (a), luego se desplaza el centroide hacia el punto más cercano en la superficie de la malla (b), el área se obtiene sumando el área de los triángulos.

4.2.6 Clase TRatio y TInterface

La clase TRatio permite definir proporciones entre dos valores, provenientes de las propiedades medidas sobre los puntos de referencia, segmentos, ángulos, caminos y contornos. Por ello es que se define la clase TInterface, como tipo común, para acceder a las propiedades de los objetos sin importar el tipo de objeto de que se trate. Esto es posible a través del método Get de la clase TInterface. Este método recibe como argumento una cadena de texto, con el nombre de la propiedad a la cual se quiere consultar, y un apuntador genérico a un tipo de dato en donde se escribirá el valor de la propiedad consultada. Por ejemplo, si queremos obtener la posición de un objeto de la clase TLandmark, se invoca al método Get con el nombre de la propiedad "Position" y un apuntador a un objeto tipo vec3. La clase TRatio posee dos apuntadores a objetos de tipo TInterface, uno para el numerador del cociente, y otro para el denominador, así como dos cadenas de texto con los nombres de las propiedades del numerador y del denominador. Al igual que con los otros tipos de medición, la clase TRatio posee atributos para definir el nombre, nemónico o nombre corto y un texto descriptivo. Los diagramas de clase de la clase TRatio y de la clase TInterface se muestran en la (Figura 50). El diagrama de dependencias de clases de las clases TRatio y TInterface se muestra en la (Figura 51).

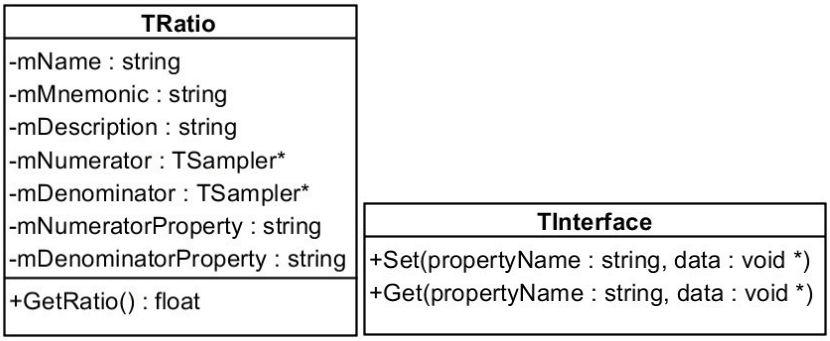


Figura 50. Diagramas de clases de la clase TRatio y de la clase TInterface.

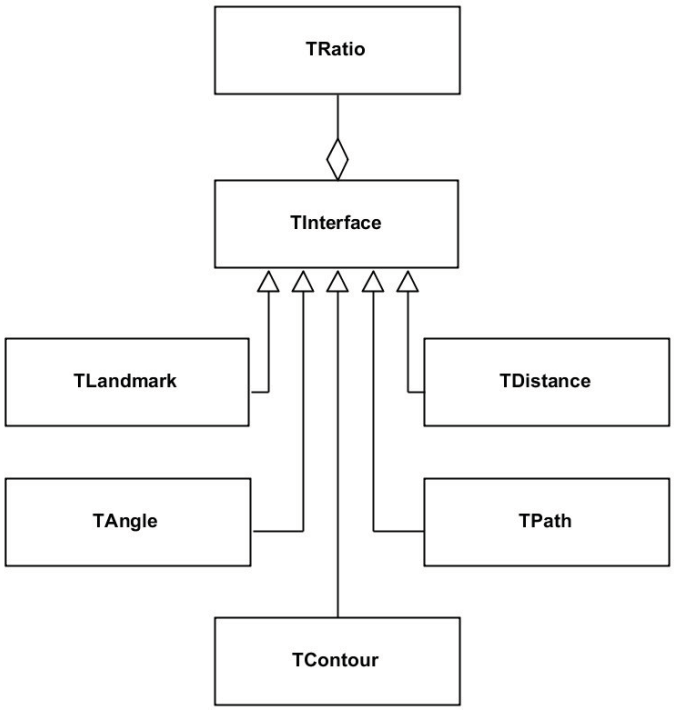


Figura 51. Diagrama de dependencias de clases de las clases TRatio y TInterface.

4.2.7 Clase TOctree y TOctreeNode

La clase TOctree y TOctreeNode, en conjunto, implementan la estructura de datos de árbol octal construido sobre las caras triangulares de la malla, una vez que el modelo ha sido importado. La clase TOctree posee un único apuntador a un objeto de la clase TOctreeNode, el cual representa la raíz del octree. Por su parte, cada objeto de la clase TOctreeNode posee atributos correspondientes a la profundidad dentro del árbol octal a la cual pertenece el nodo, el centro y extensiones medias en cada eje de la celda asociada al nodo, un apuntador a la celda padre del nodo, que en el caso del nodo raíz, es nulo, y un arreglo de ocho apuntadores a cada uno de los hijos posibles del nodo. Las celdas hojas tiene la característica de poseer un atributo no nulo, correspondiente a la lista de índice de los triángulos, contenidos parcial o totalmente, dentro de la celda asociada espacialmente al nodo. Los diagramas de clase de la clase TOctree y TOctreeNode se muestran en la (Figura 52).

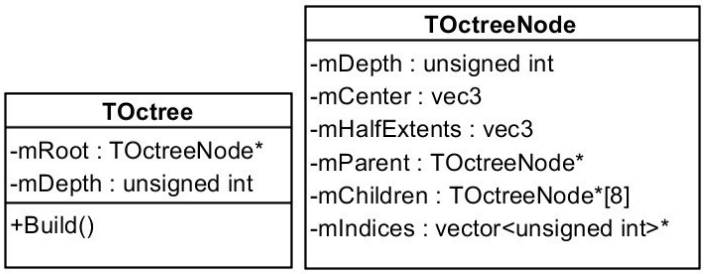


Figura 52. Diagramas de clase de la clase TOctree y de la clase TOctreeNode

4.2.8 Clase TMesh

La clase TMesh se encarga de representar en memoria una malla triangular en 3D sobre la cual se aplican los algoritmos geométricos. La malla se importa desde un archivo en disco utilizando la

biblioteca ASSIMP[1], típicamente desde un archivo en formato OBJ o PLY. Luego se realiza un post procesamiento para traducir la información de la malla importada a nuestro formato mientras se eliminan los vértices y caras duplicadas, y, por último, se calculan los vectores normales en cada vértice, así como las curvaturas.

En nuestro formato la malla consta de listas separadas de vértices, aristas y caras, en donde cada vértice tiene información sobre las aristas y caras de las cuales forma parte, y las aristas y las caras tienen información sobre los vértices que las forman. Haciendo explícita la información de las vecindades en cada elemento, el recorrido de la malla se hace más eficiente pero la memoria requerida se incrementa.

Además de contener la información sobre la geometría e la malla, la clase TMesh contiene listas separadas con los materiales, texturas y grupos de renderizado, mismos que están definidos en el archivo correspondiente desde el cual se importa el modelo. Para realizar una intersección eficiente de un rayo con la malla triangular, la clase TMesh contiene un objeto del tipo TOctree, con el cual se construye un árbol octal, a partir de la información de las caras triangulares del modelo, durante el proceso de importación de la malla. El diagrama de clase de la clase TMesh se muestra en la (Figura 53). El diagrama de dependencias de clases de la clase TMesh se muestra en la (Figura 54).

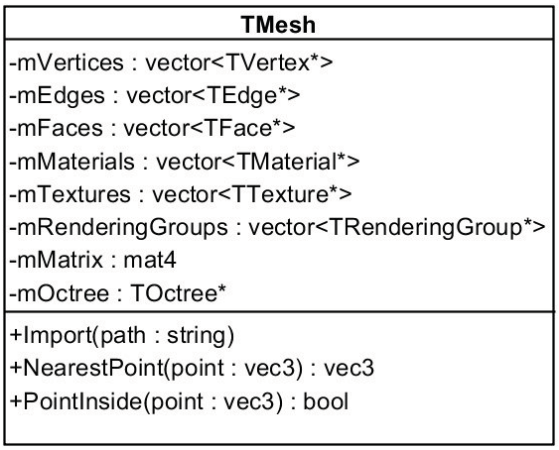


Figura 53. Diagrama de clase de la clase TMesh.

[1] www.assimp.org/

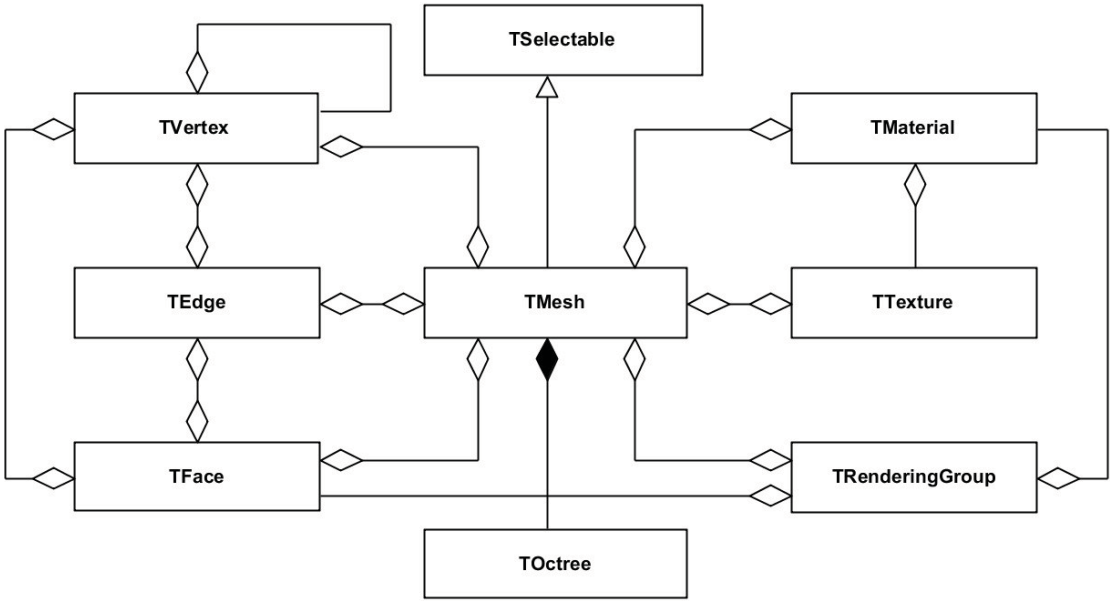


Figura 54. Diagrama de dependencias de clases de la clase TMesh.

4.2.9 Clase TVertex

La clase vértice contiene información sobre la posición espacial del vértice, el vector normal, curvaturas y listas con apuntadores a las aristas incidentes y a las caras que comparten ese vértice. Durante el proceso de la importación de la malla se eliminan los vértices duplicados (Figura 55). La eliminación de redundancias de vértices se realiza asociando a cada posición espacial, correspondiente a la posición del vértice, un apuntador a la estructura de datos del vértice correspondiente, para formar pares clave-valor dentro de una lista ordenada o mapa. Cada vez que un vértice es referenciado por una cara se realiza una búsqueda en la lista, si ya existe un vértice en esa posición espacial con cierta tolerancia, se adquiere el puntero al vértice asociado a esa posición, en caso contrario, se crea un nuevo vértice y se inserta ordenadamente en la lista. Para esto se utilizó la clase map[1] de la biblioteca de plantillas estándar de C++. En esta implementación la inserción se realiza en tiempo logarítmico en el número de elementos dentro del mapa. En total se procesan $3n$ vértices en donde n es el número de caras de la malla, por lo que la complejidad del algoritmo para la eliminación de vértices duplicados es $O(n \log n)$, y se ejecuta fuera de línea al cargar el modelo.

[1] http://www.cplusplus.com/reference/map/map/

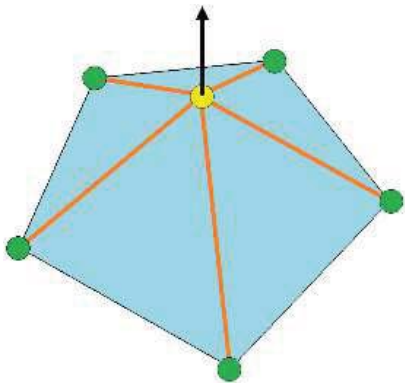


Figura 55. Un vértice (amarillo) y sus correspondientes vértices vecinos (verde), aristas adyacentes (naranja), caras adyacentes (azul) y vector normal (negro).

Los atributos de la clase TVertex se muestran en el diagrama de clase de la (Figura 56). La lista de aristas y caras adyacentes se construye durante la importación de la malla por lo que no existe ninguna relación entre el orden de los elementos en las listas y su configuración espacial, por otra parte, la lista de vértices vecinos se construye a partir de la lista de aristas adyacentes y se ordena en sentido anti horario en torno a la dirección del vector normal, obtenido del promedio de los vectores normales de las caras adyacentes.

TVertex
-mMesh : TMesh*
-mIndex : int
-mPosition : vec3
-mNormal : vec3
-mNeighbours : vector<TVertex*>
-mAdjacentEdges : vector<Edge*>
-mAdjacentFaces : vector<Face*>
-mGaussianCurvature : float
-mMeanCurvature : float
-mShapeIndex : float
-mCurvedness : float
-mCurvatureRatio : float

Figura 56. Diagrama de clase de la clase TVertex.

4.2.10 Clase TEdge

La clase arista mantiene información sobre la conectividad de los vértices, conteniendo apuntadores a los vértices que forman la arista y a las caras que son adyacentes a esa arista (Figura 57, Figura 58). Las aristas no están orientadas, por lo que la arista $\overline{AB}$ es la misma que la arista $\overline{BA}$. Para la eliminación de aristas duplicadas se emplea un algoritmo similar al utilizado con los vértices, con la diferencia de que la clave corresponde a los índices de los vértices que conforman la arista, reordenados de tal forma que el primer índice es el menor. En total se procesan $3n$ aristas en donde n es el número de caras de la malla, por lo que la complejidad del algoritmo para la eliminación de aristas redundantes es $O(n \log n)$.

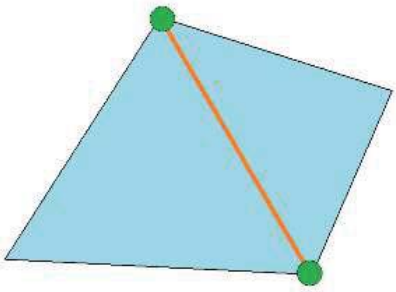


Figura 57. Arista (naranja) y sus respectivos vértices (verde) y caras adyacentes (azul).

TEdge
-mMesh : TMesh*
-mIndex : int
-mVertices : TVertex*[2]
-mAdjacentFaces : vector<TFace*>

Figura 58. Diagrama de clase de la clase TEdge.

4.2.11 Clase TFace

La clase cara agrupa vértices y aristas para formar triángulos, manteniendo punteros a los vértices que componen frontalmente el triángulo en sentido anti-horario. Adicionalmente la clase cara contiene información del vector normal de la misma cara y de cada uno de los vértices que la componen, así como los colores por vértices y las coordenadas de textura (Figura 59, Figura 60). Para la eliminación de caras redundantes, es decir, formadas por los mismos vértices, se aplica un algoritmo para la eliminación de redundancias similar al utilizado para las aristas, con la diferencia de que la clave corresponde a los índices de los vértices que forman la cara, reordenados de forma tal que el primer índice es el menor sin alterar el orden circular de los vértices. Dado que se procesan n caras, la complejidad del algoritmo para la eliminación de caras redundantes es $O(n \log n)$.

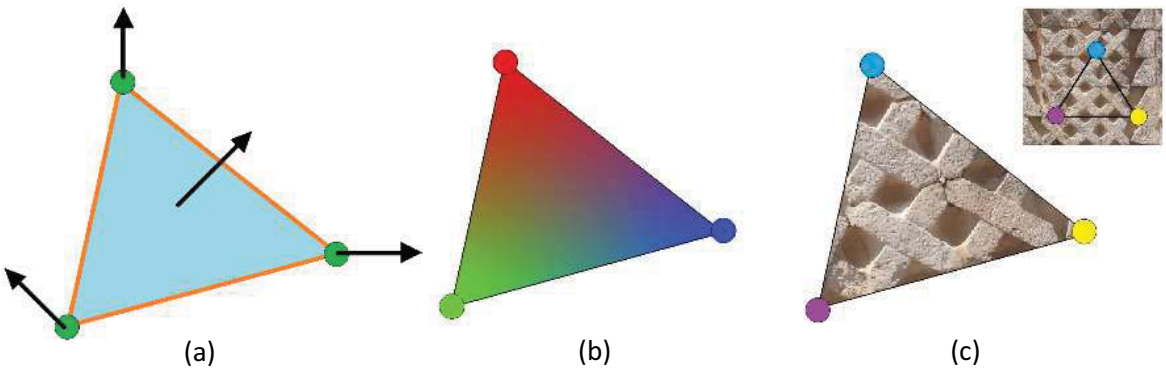


Figura 59. Una cara triangular (azul) con los vértices (verde) y aristas (naranja) que la forman, así como los vectores normales (negro) de la cara y de cada vértice (a). Cara triangular con colores por vértice (b). Cara triangular con coordenadas de textura por vértice, para mapeado de textura, representadas con colores correspondientes (c).

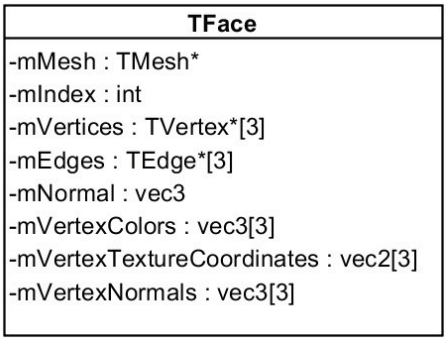


Figura 60. Diagrama de clase de la clase TFace.

4.2.12 Clase TMaterial

La clase TMaterial contiene información sobre la apariencia visual de los objetos en la escena. Los atributos de la clase TMaterial corresponden a las propiedades materiales, utilizadas para el cálculo del color final de los fragmentos, dentro del programa sombreador de fragmentos. Estas propiedades son: color ambiental, color difuso, color y exponente especular o brillantez (shininess), color emisivo, opacidad (transparencia) y, opcionalmente, mapa de textura. El diagrama de clase de la clase TMaterial se muestra en la (Figura 61).

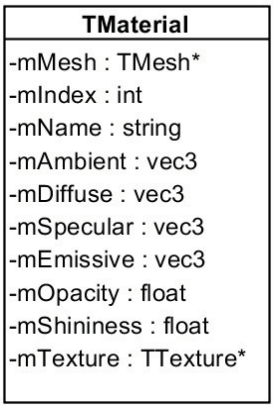


Figura 61. Diagrama de clase de la clase TMaterial.

Los materiales son obtenidos del archivo original del modelo importado, los cuales son definidos dentro del software de modelado 3D en que los modelos fueron generados. Dependiendo de los parámetros de cada material se pueden generar distintos efectos visuales, con el fin de imitar materiales reales como plástico o metal. En la (Figura 62) se muestran algunos ejemplos de materiales.

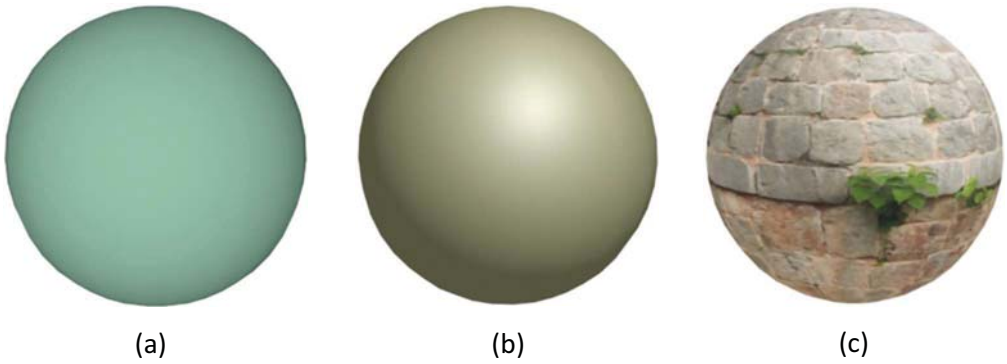


Figura 62. Ejemplos de distintos materiales: difuso (a), especular (b) y con mapeado de textura (c).

4.2.13 Clase TTexture

La clase TTexture tiene la función de almacenar en memoria la información de las imágenes necesarias para realizar el mapeado de textura sobre los objetos en la escena. Los atributos de la clase TTextura corresponden al nombre del archivo, desde el cual se carga la textura, las dimensiones en pixeles y número de canales de la imagen, y un arreglo de bytes con la información de color de cada uno de los pixeles que conforman la imagen de la textura. Los miembros de la clase TTexture se muestran en el diagrama de clase de la (Figura 63).

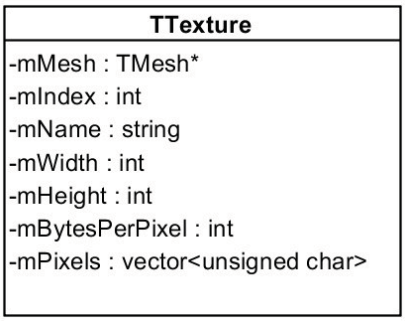


Figura 63. Diagrama de clase de la clase TTexture.

La imagen de la textura se almacena dentro del arreglo de bytes, correspondiente a los pixeles, comenzando por el pixel de la esquina inferior izquierda, alternando los canales de color de la imagen. Por ejemplo, si la imagen está en formato RGB, el primer elemento del arreglo es el valor de rojo del pixel de la esquina inferior de la imagen, el segundo elemento es el valor de verde del mismo pixel, y el tercer elemento es el correspondiente valor de azul del pixel. Los pixeles se almacenan por filas, por lo que los últimos valores en el arreglo corresponden a los canales de color del pixel en la esquina superior derecha de la imagen (Figura 64).

Figura 64. Los pixeles se almacenan en memoria por filas, comenzando por el pixel de la esquina inferior de la imagen, y finalizando con el pixel de la esquina superior derecha de la imagen.

Para mapear la textura sobre los triángulos del modelo, se utilizan las coordenadas de textura definidas por cada cara triangular. El sistema de coordenadas de textura se establece con el origen en la esquina inferior izquierda de la imagen, con el eje $\mathcal{X}$ positivo hacia la derecha y el eje $\mathcal{Y}$ positivo hacia arriba. La esquina superior derecha de la imagen, en el sistema de coordenadas de textura, corresponde a la coordenada $(1,\ 1)$, tal como se visualiza en la (Figura 65).

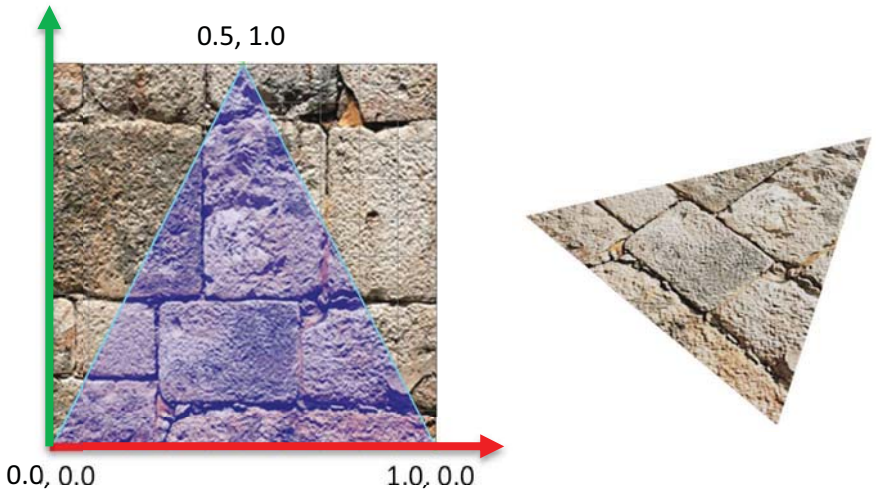


Figura 65. Sistema de coordenadas de textura con las coordenadas de textura de un triángulo simple.

4.2.14 Clase TRenderingGroup

La clase TRenderingGroup permite agrupar las caras triangulares de la malla que tienen asignado el mismo material, con el fin de hacer eficiente el renderizado de la malla. Cada objeto de la clase TRenderignGroup mantiene una lista con apuntadores a los triángulos que conforman el grupo de renderizado, así como un apuntador a un objeto del tipo TMaterial, correspondiente al material que será aplicado a las caras triangulares. El diagrama de la clase TRenderingGroup se muestra en la (Figura 66). Los grupos de renderizado son creados durante el proceso de importación de la malla, y corresponde a la forma en que los materiales fueron asignados a los objetos en la escena, desde el software de modelado 3D en que el modelo fue generado. En la (Figura 67) se muestra un ejemplo

de una escena con diferentes materiales, en donde cada grupo de renderizado corresponde a un objeto diferente de la escena.

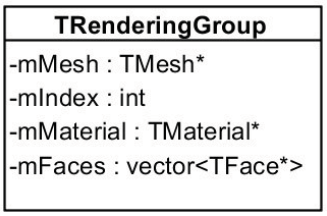


Figura 66. Diagrama de clase de la clase TRenderingGroup.

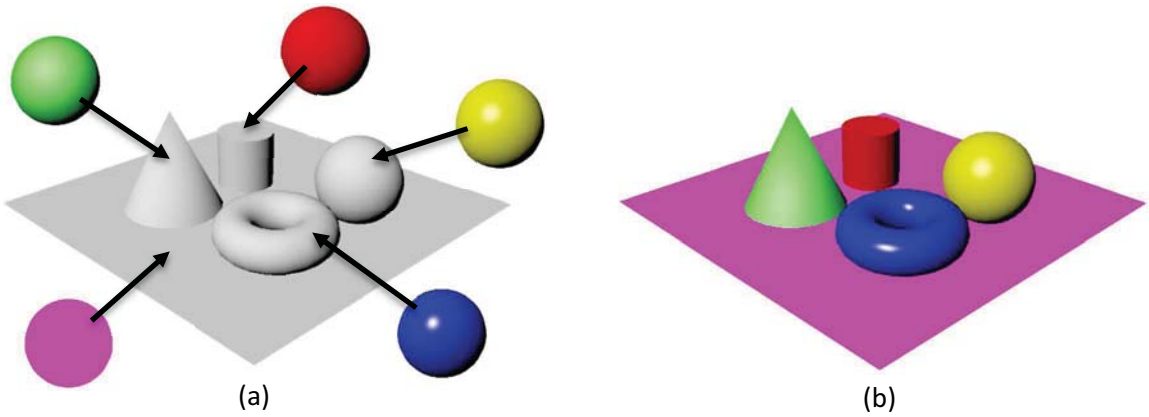


Figura 67. Escena sin materiales, indicando por cada objeto el material correspondiente (a). Escena renderizada aplicando los materiales respectivos a cada objeto (b).

4.3 INTERFAZ GRÁFICA DE USUARIO

Para la implementación de la interfaz gráfica de usuario se utilizó la biblioteca en lenguaje C++ Qt 5.5. La interfaz gráfica consiste de un conjunto de menús, botones y otros elementos visuales que permiten al usuario desde cargar el modelo y visualizar la escena desde distintos puntos de vista, hasta crear y editar puntos de referencia, distancias y ángulos entre puntos de referencia, así como trazar caminos y contornos sobre la superficie de la malla. Otra de las características del software es la capacidad para obtener información de los puntos de referencia, por ejemplo, las curvaturas y el vector normal, obtener las proyecciones de las distancias en cada eje, medir la distancia sobre un camino trazado sobre la malla y calcular cocientes entre los distintos parámetros medidos, permitiendo de esta forma, definir índices como cocientes entre las distancias lineales entre puntos seleccionados.

4.3.1 Visualización de curvaturas por mapas de color

La interfaz gráfica de usuario se diseñó para ser intuitiva al usuario, imitando la apariencia y características de softwares comerciales para visualización 3D como Maya, MeshLab, Amira[1], tal como lo es la capacidad para visualizar la escena hasta desde cuatro puntos de vista diferentes, utilizando ventanas de visualización por separado, cada una con sus propias herramientas de visualización y opciones configurables (Figura 68). Otras de las características del software

[1] http://www.fei.com/software/amira-3d-for-life-sciences/

desarrollado en este trabajo, es la implementación de distintos modelos de iluminación para visualizar el objeto virtual, así como el uso de mapas de color para visualizar las curvaturas de la malla (Figura 69). De esta forma el usuario puede seleccionar distintos estilos de renderizado y parámetros a visualizar, variando el tipo de mapa de color utilizado, a fin de tener una mejor percepción de la geometría del modelo con el que se está trabajando (Borland and Taylor, 2007).

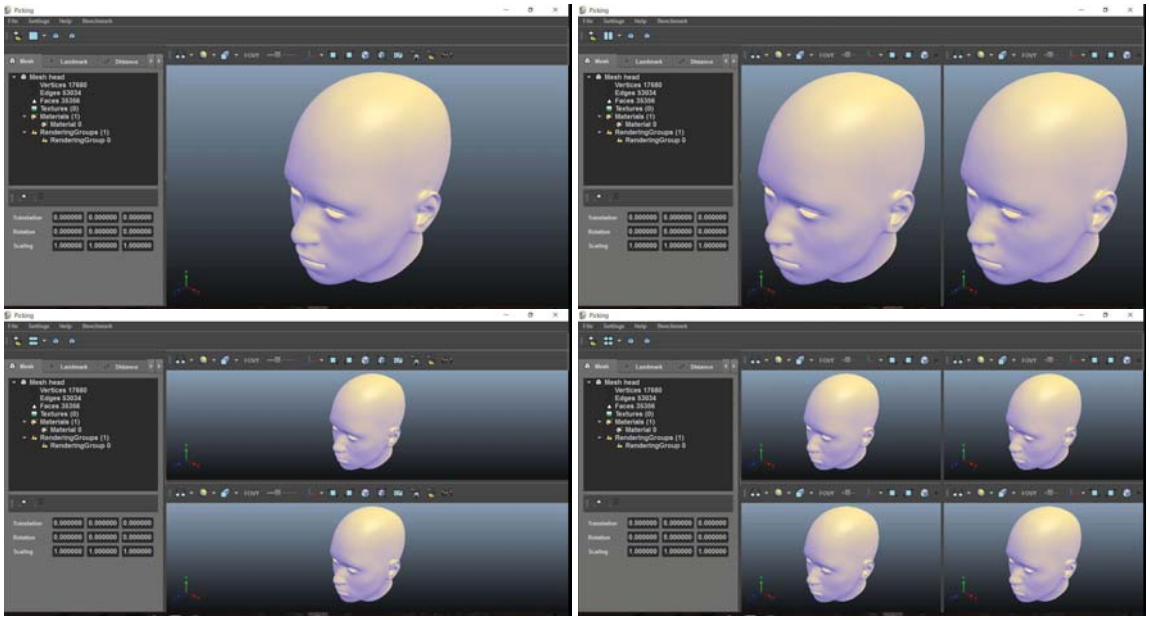


Figura 68. Software para la edición de puntos de referencia con hasta cuatro ventanas de visualización diferentes.

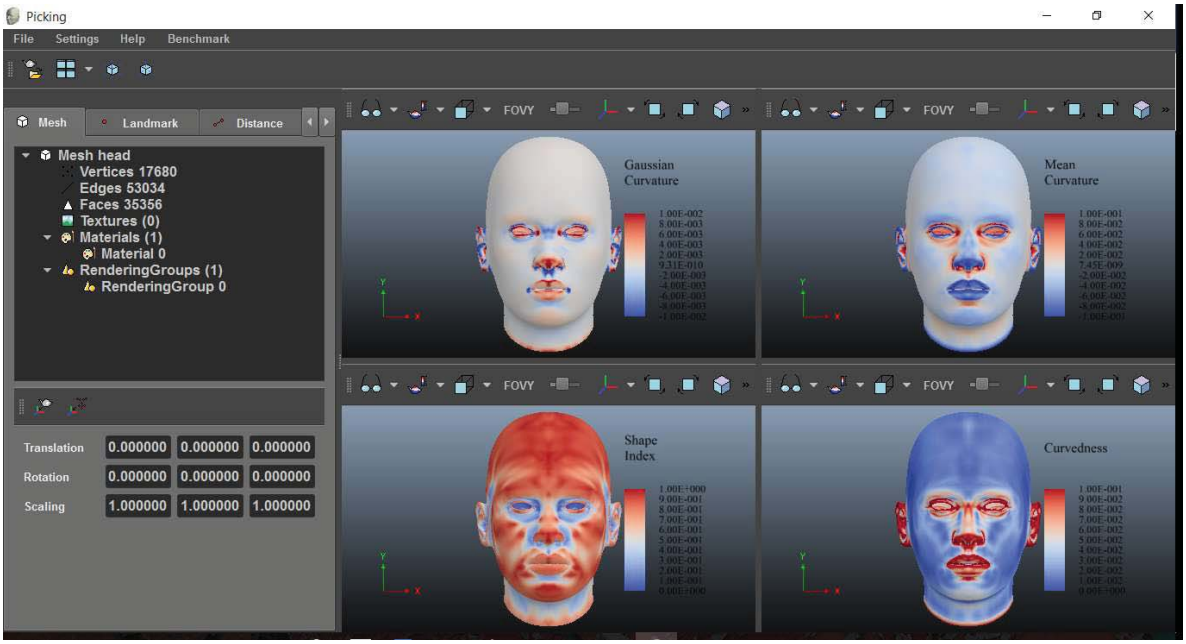


Figura 69. Distintas visualizaciones de curvaturas de la malla utilizando mapas de color.

4.3.2 Visualización de métricas sobre los puntos de referencia

El principal objetivo del desarrollo de este software, es brindar al usuario una herramienta computacional, mediante la visualización científica del modelo de la cabeza humana, sin importar el origen o la complejidad del modelo, para seleccionar los puntos de referencia de manera interactiva. Esto se logra a través de la intersección en tiempo real de un rayo con la malla 3D, que en todo momento corresponde con la posición del puntero del ratón sobre el área de visualización. De esta forma la selección de los puntos es precisa e intuitiva, ya que corresponde directamente a lo que el usuario ve en la pantalla. Para facilitar la identificación de los puntos de referencia, el software permite asignarle a cada punto un nombre específico, así como un nemónico que representa el nombre corto, abreviatura o símbolo del punto de referencia, cuyo texto se muestra en la visualización cerca del punto correspondiente. Adicionalmente el usuario puede escribir una descripción del punto de referencia, como por ejemplo la descripción o localización del punto, con lo cual esta herramienta puede ser utilizada para catalogar los puntos de referencia y servir de herramienta didáctica.

Cada punto de referencia contiene información sobre la posición espacial del punto y del vector normal de la superficie asociado a ese punto, información textual para el nombre, nemónico y descripción del punto de referencia, y adicionalmente, información sobre la curvatura gaussiana, curvatura media, índice de forma, curvidad e índice de curvatura correspondiente al punto sobre la superficie de la malla en que fue colocado dicho punto (Figura 70). Dicha información puede ser respaldada en forma de un archivo de texto, utilizando un formato de valores separados por comas, el cual puede ser fácilmente interpretado por otros softwares, por ejemplo, Excel[1] o MATLAB, para realizar el análisis estadístico de los puntos.

Los puntos de referencia (landmarks) pueden ser editados por el usuario, permitiéndole cambiarlos de posición de manera interactiva, arrastrando los puntos sobre la superficie de la malla utilizando el ratón, o bien, eliminar los puntos haciendo doble clic sobre el punto. Los puntos de referencia pueden ser seleccionados desde una lista, en la interfaz gráfica, o directamente y de manera interactiva, desde la ventana de visualización, posicionando el cursor del ratón sobre un punto de referencia, representado mediante una esfera de color rojo, la cual se muestra de un tamaño ligeramente más grande cuando el punto correspondiente es seleccionado.

[1] https://products.office.com/es-mx/excel

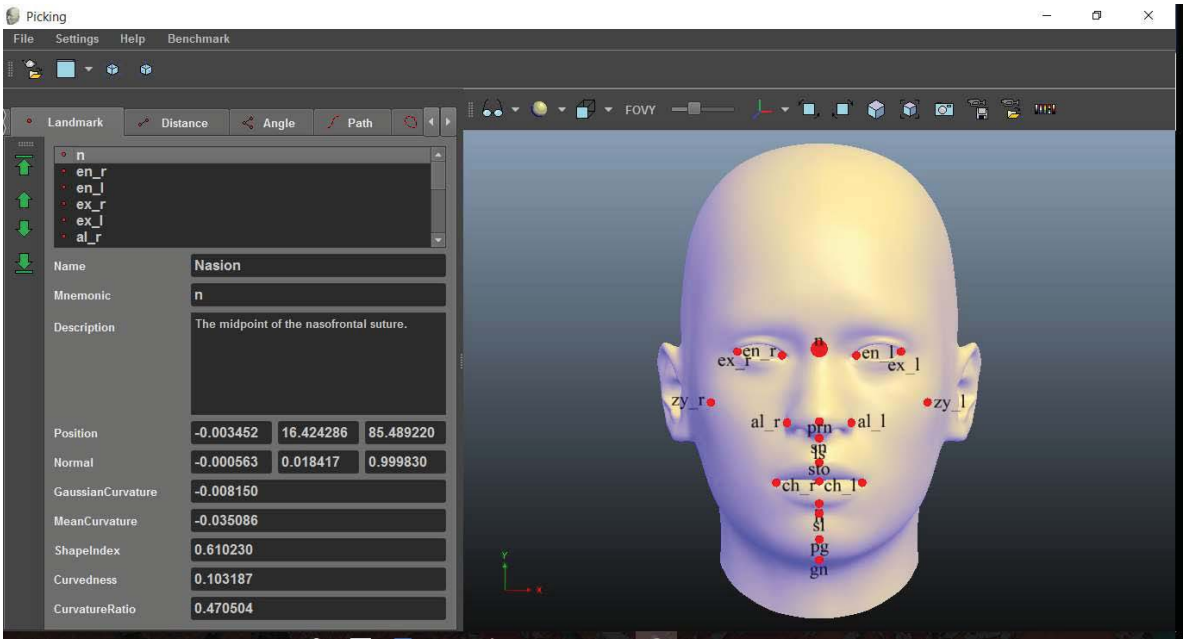


Figura 70. Visualización de los puntos de referencia y de las propiedades correspondientes a un punto.

Además de seleccionar y editar puntos de referencia sobre el modelo 3D, el usuario puede definir distancias lineales entre puntos, permitiéndole obtener la distancia euclidiana entre los puntos, así como la longitud de la línea proyectada en cada eje coordenado. A cada segmento entre puntos, al igual que con los puntos de referencia, se le asigna un nombre, un nemónico y un texto descriptivo (Figura 71). El usuario puede seleccionar los segmentos entre puntos desde el área de graficación, de manera intuitiva, posicionando el cursor del mouse sobre el segmento correspondiente. Cuando un segmento es seleccionado, ya sea desde la lista de distancias en la interfaz gráfica, o de manera interactiva en la visualización, el segmento correspondiente se muestra con un grosor más grande, como una forma de retroalimentación visual al usuario.

También se incluye en el software la capacidad de seleccionar ángulos formados por tres puntos de referencia distintos, y asignarles un nombre, un nemónico y un texto descriptivo (Figura 72). Tanto en las distancias, como en los ángulos, siempre que un punto de referencia, que forme parte de alguna distancia o ángulo, cambie de posición, los valores de las distancias y los ángulos son actualizados en tiempo real. De la misma forma, cuando un punto de referencia es eliminado, se eliminan también todas las distancias y ángulos que utilicen este mismo punto. Las distancias lineales y los ángulos pueden de igual forma ser respaldados en forma de un archivo con valores separados por comas, en adelante CSV, para su posterior análisis.

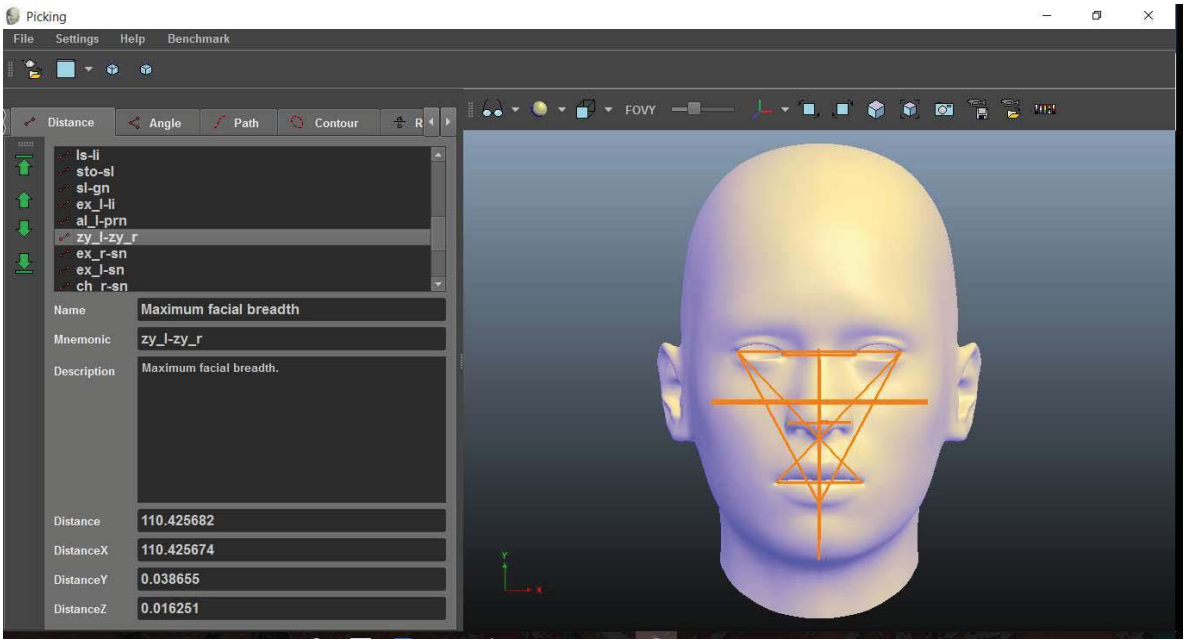


Figura 71. Distancias lineales entre puntos de referencia.

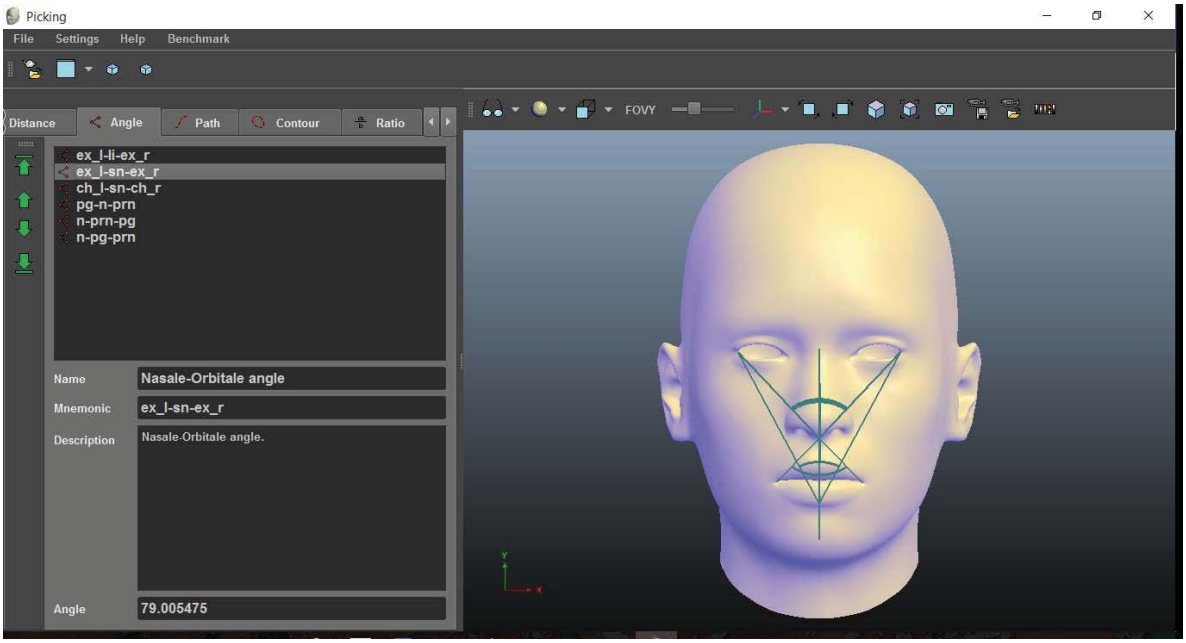


Figura 72. Medidas angulares entre puntos de referencia.

Adicionalmente el usuario puede definir cocientes entre distancias para obtener de manera automática índices o proporciones entre distancias, a las cuales se les puede asignar un nombre, nemónico y texto de descripción (Figura 73). Dichos valores pueden ser respaldas en un archivo para su posterior análisis. A diferencia de los puntos de referencia, o de las distancias y los ángulos, los índices no tienen representación visual en el área de renderizado.

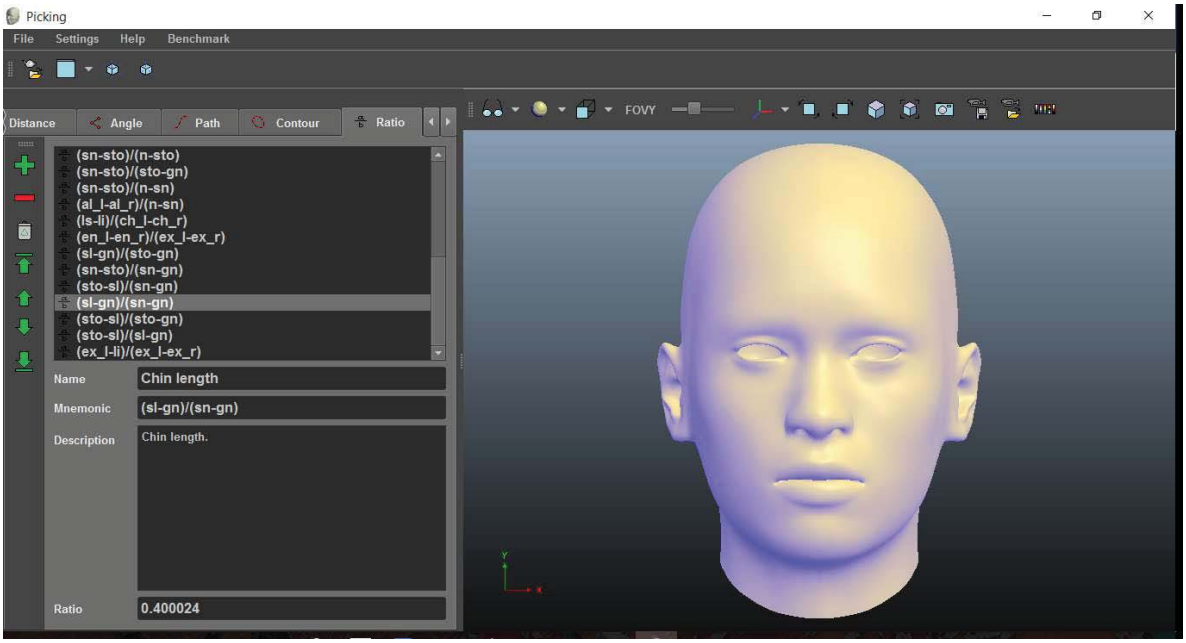


Figura 73. Proporciones o cocientes entre distancias para la obtención de índices.

4.3.3 Opciones de visualización de los modelos

Cada ventana de visualización tiene sus propias opciones de visualización, el usuario modifica la apariencia de la visualización en cada ventana mediante una barra de herramientas en la parte superior de cada ventana (Figura 74). Entre las opciones de visualización está el ocultar o mostrar los vértices, aristas y caras que conforman la malla, mostrar u ocultar los puntos de referencia, segmentos, ángulos, caminos o contornos, así como el texto asociado a cada uno de ellos. También se incluyen distintos modelos de iluminación, como el modelo de iluminación de Lambert, Phong, Gooch y sombreado estilo caricatura (*Cel Shading*). En estos modos la malla se muestra con un color predefinido dentro del programa del sombreador en lenguaje GLSL, para visualizar el modelo con los colores y los materiales originales, tal y como están definidos en el archivo desde el cual se importó la malla, se incluye un sombreador de colores por vértice y con mapeado de texturas. El modelo de iluminación por defecto es el modelo de iluminación no fotorrealista de Gooch.

Adicionalmente el usuario puede elegir visualizar las curvaturas del modelo utilizando mapas de color, los parámetros que se pueden visualizar son la curvatura gaussiana, curvatura media, índice de forma, curvidad e índice de curvatura. El mapa de color puede configurarse a través de una ventana de configuración (Figura 75), para modificar el intervalo y los colores de la escala. Por simplicidad para el usuario, las escalas de color están predefinidas de acuerdo a las escalas de color más comunes en softwares de visualización científica (Figura 76).

Otras opciones de visualización que el usuario puede seleccionar están el cambiar el tipo de proyección de la cámara y el campo de visión, en el caso de proyección en perspectiva, así como elegir la orientación de los ejes coordenados. Una opción muy importante es la de autoajustar la cámara sobre el modelo, para lo cual se mueve el pivote de la cámara y la distancia de la cámara al pivote, a fin de que la malla proyectada en la pantalla abarque toda el área de visualización. Finalmente se incluyen mandos para que el usuario tome una captura de pantalla del área de

visualización y la guarde en un archivo de imagen, y para guardar el estado de la cámara en un archivo y posteriormente restaurar la vista de la cámara virtual.

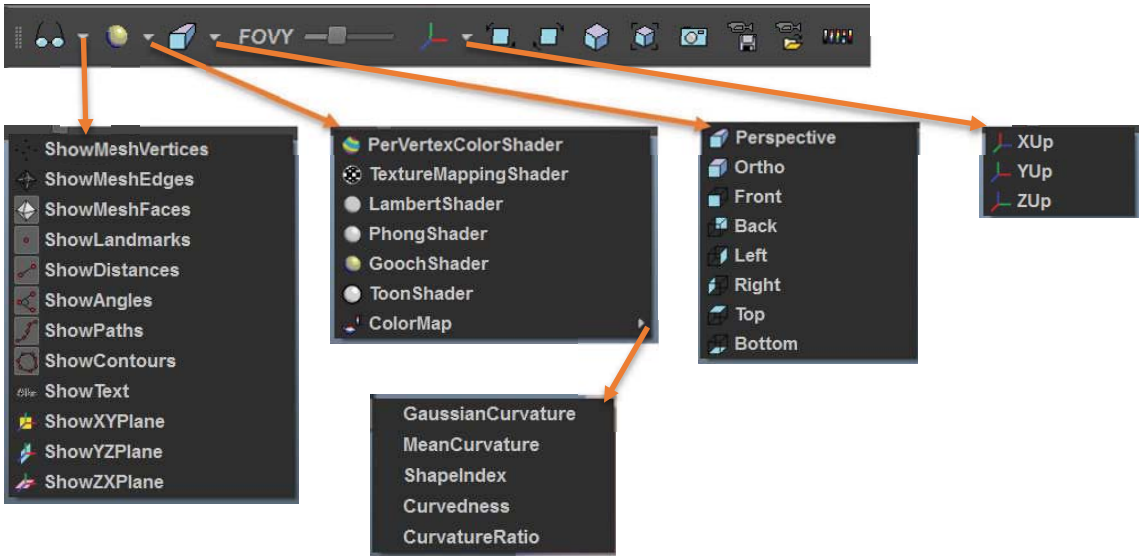


Figura 74. Barra de herramientas de visualización.

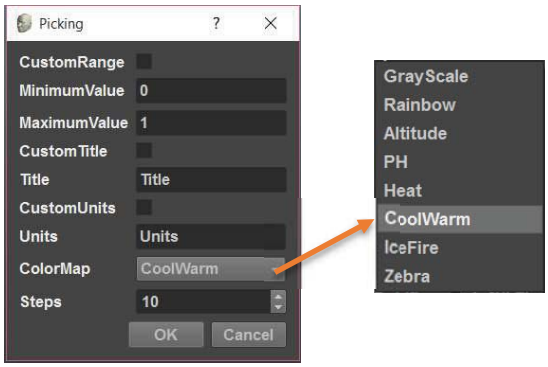


Figura 75. Ventana para configurar el mapa de color.

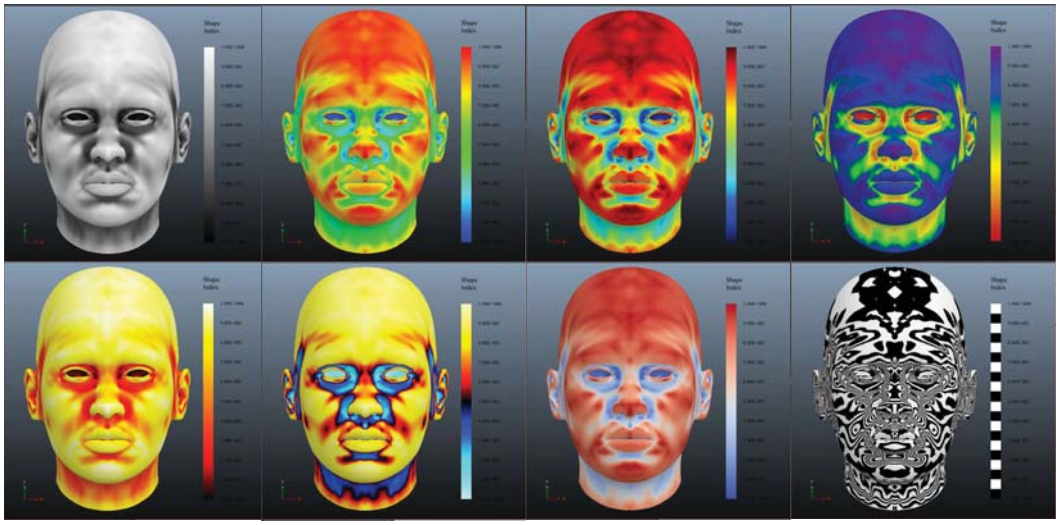


Figura 76. Índice de forma de una misma malla visualizado mediante diferentes mapas de color.

4.3.4 Visualización de perfiles de forma

Como herramienta adicional el usuario puede trazar caminos o trayectorias sobre el mallado, de manera precisa y en tiempo real, desplazando el puntero del ratón sobre el modelo 3D mientras mantiene presionado el botón izquierdo del ratón (Figura 77). Una de las características novedosas de este software es que, además de obtener la longitud del camino, se pueden obtener la variación de las propiedades de la malla, como lo es la posición, vector normal y curvaturas, a lo largo de la trayectoria definida por el usuario, y esa información puede visualizarse en forma de una gráfica, como se muestra en la (Figura 78). Para generar las gráficas se utilizó la biblioteca de programación en lenguaje C++ QCustomPlot[1]. Las propiedades de la malla, muestreadas en la trayectoria del camino, pueden ser respaldados en un archivo en formato CSV, para su posterior análisis.

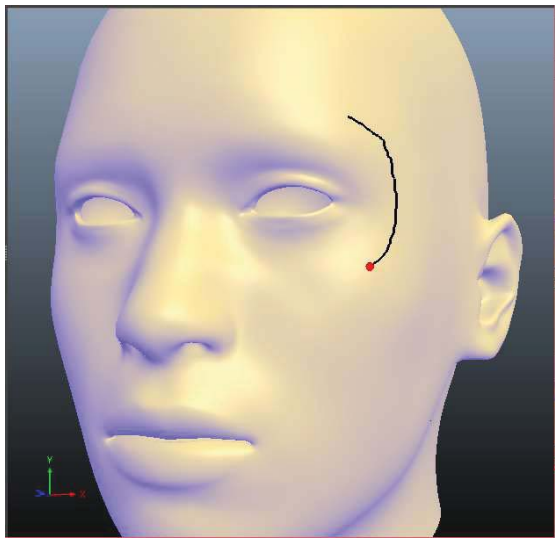


Figura 77. Trazado de caminos sobre la malla 3D.

[1] http://www.qcustomplot.com/

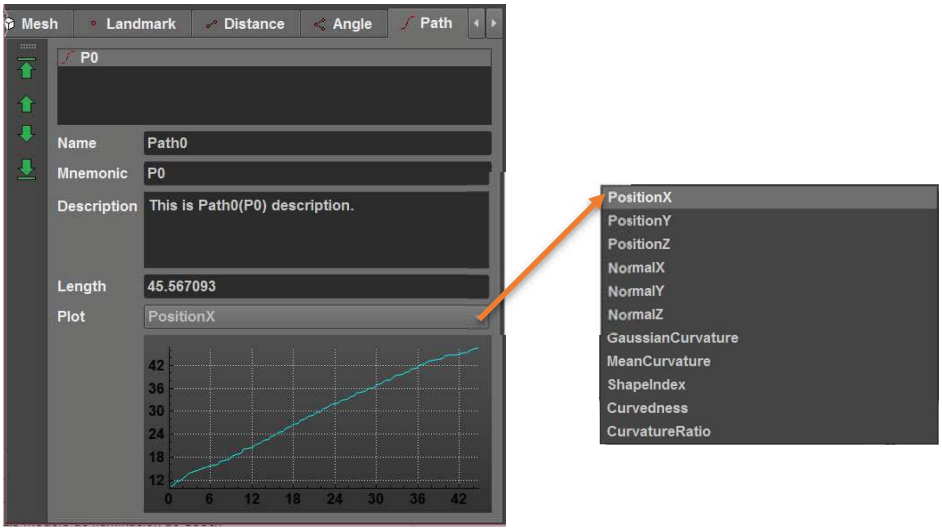


Figura 78. Propiedades de un camino seleccionado con gráfica de posición en x.

Los contornos se trazan de manera interactiva sobre la malla, de la misma en que se trazan los caminos. La información que el usuario puede obtener a partir de los contornos son el perímetro del contorno y el área (aproximada) de la región de la malla contenida dentro del contorno (Figura 79). Las propiedades de posición, vector normal y curvaturas, asociadas a cada punto que conforman el contorno, pueden ser respaldados en un archivo CSV para su posterior análisis.

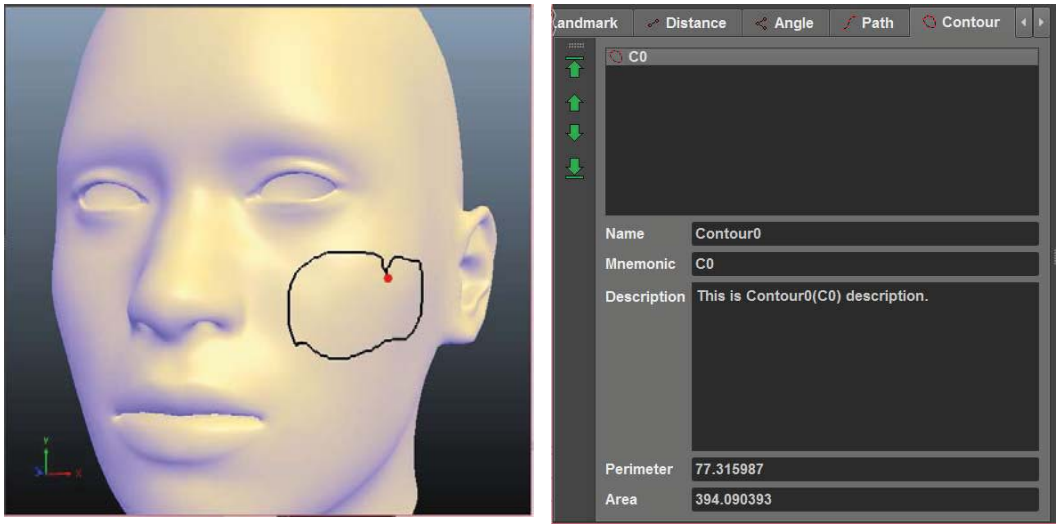


Figura 79. Contorno definido sobre la superficie de la malla junto con sus propiedades.

4.4 INTERFAZ HÁPTICA

Para hacer intuitivo al usuario la selección de puntos de referencia y trazado de caminos y contornos sobre las mallas 3D, se le implementó al software la capacidad para utilizar el dispositivo háptico PHANTOM OMNI, para inducir en el usuario la sensación de estar palpando el objeto 3D mediante la retroalimentación de fuerza. Para conseguir que los movimientos del usuario coincidan con el punto de vista de la cámara virtual, independientemente de la traslación, rotación o escala de la escena visualizada, se mapea el espacio de trabajo del dispositivo háptico dentro del volumen de

visualización, o *frustum*, de la cámara virtual. Considérese que el espacio de trabajo del dispositivo háptico corresponde a una caja, descrita por las coordenadas mínimas y máximas, medidas sobre cada eje del sistema de coordenadas del dispositivo con respecto al punto de cero (Figura 80).

Figura 80. Sistema de coordenadas y volumen de trabajo del dispositivo háptico PHANTOM OMNI.

La matriz que transforma de coordenadas del dispositivo háptico a coordenadas globales en la escena, de acuerdo a la configuración de la cámara virtual, es:

$$\mathbf{M}_{\text{haptic}} = \mathbf{T}\left(p_{\mathcal{X}}, p_{\mathcal{Y}}, p_{\mathcal{Z}}\right)\mathbf{R}_{\mathcal{Z}}\left(r_{\mathcal{Z}}\right)\mathbf{R}_{\mathcal{Y}}\left(r_{\mathcal{Y}}\right)\mathbf{R}_{\mathcal{X}}\left(r_{\mathcal{X}}\right)\mathbf{S}\left(s\right)\mathbf{T}\left(c_x, c_y, c_z\right)^{-1} \tag{112}$$

En donde $\left(p_{\mathcal{X}}, p_{\mathcal{Y}}, p_{\mathcal{Z}}\right)$ es la posición del pivote de la cámara en la escena, $\left(r_{\mathcal{X}}, r_{\mathcal{Y}}, r_{\mathcal{Z}}\right)$ son los ángulos de Euler correspondientes a la rotación de la cámara virtual con respecto al pivote, $\left(c_x, c_y, c_z\right)$ corresponden al punto central del espacio de trabajo del dispositivo, obtenido como:

$$\begin{pmatrix} c_x & c_y & c_z \end{pmatrix} = \begin{pmatrix} \dfrac{x_{\min} + x_{\max}}{2} & \dfrac{y_{\min} + y_{\max}}{2} & \dfrac{z_{\min} + z_{\max}}{2} \end{pmatrix} \tag{113}$$

Y s es un factor de escala que depende del modelo de proyección de la cámara virtual.

Proyección perspectiva. El factor de escala es:

$$s = \max\left(\frac{x_{\max} - x_{\min}}{w}, \frac{y_{\max} - y_{\min}}{h}\right)\frac{\left|\mathbf{p} - \mathbf{e}\right|}{e} \tag{114}$$

En donde w, h corresponden a las dimensiones físicas del visor en pixeles, $\mathbf{p}, \mathbf{e}$ son respectivamente el pivote y la posición de la cámara en coordenadas globales, y e es la distancia focal de la cámara virtual.

Proyección ortográfica. El factor de escala está dado por:

$$s = \max\left(\frac{x_{\max} - x_{\min}}{w}, \frac{y_{\max} - y_{\min}}{h}\right)\frac{d}{\zeta e} \tag{115}$$

Siendo d la distancia de la cámara la pivote y ζ el factor de escala, o zoom, de la cámara.

Sea $\mathbf{h} \in \mathbb{R}^3$ la posición del Punto de Interfaz Háptica (HIP) en coordenadas del dispositivo dentro del espacio de trabajo. Las coordenadas del HIP dentro del volumen de visualización $\mathbf{h}'$, se obtiene como:

$$\begin{pmatrix} \mathbf{h}' \\ 1 \end{pmatrix} = \mathbf{M}_{\text{haptic}} \begin{pmatrix} \mathbf{h} \\ 1 \end{pmatrix} \tag{116}$$

En cada cuadro de renderizado háptico se aplica el algoritmo de renderizado háptico descrito en el (Pseudocódigo 4). Primero se lee $\mathbf{h}$ directamente del dispositivo háptico, se calcula $\mathbf{M}_{\text{haptic}}$, y se obtiene $\mathbf{h}'$. Luego se determina el punto más cercano a $\mathbf{h}'$ sobre la superficie de la malla, $\mathbf{n}'$ en coordenadas globales, utilizando el algoritmo descrito en el (Pseudocódigo 3), y se mapea al espacio de trabajo del dispositivo háptico $\mathbf{n}$ como:

$$\begin{pmatrix} \mathbf{n} \\ 1 \end{pmatrix} = \mathbf{M}_{\text{haptic}}^{-1} \begin{pmatrix} \mathbf{n}' \\ 1 \end{pmatrix} \tag{117}$$

Se establece la posición de referencia del controlador de posición PID asociado al dispositivo háptico en $\mathbf{n}'$, y se actualiza el estado del controlador. Finalmente se verifica si el punto $\mathbf{h}'$, está colisionando con la malla, esto es, se encuentra en el interior. En caso afirmativo, se obtiene la fuerza correspondiente a la acción de control del controlador de posición PID, y se establece la salida de fuerza del dispositivo háptico a este valor, con el fin de mantener la posición del HIP sobre la superficie de la malla dentro de la escena. En caso contrario, se deshabilita la salida de fuerza del dispositivo háptico, para que el usuario pueda mover libremente el HIP dentro de la escena.

La herramienta acoplada al efector final del dispositivo háptico PHANTOM OMNI tiene la forma de una pluma con dos botones (Figura 81). El botón 1, dentro de la integración del dispositivo háptico con el software para la edición de puntos de referencia, está configurado para colocar puntos de referencia sobre la malla 3D al ser presionado, mientras la ventana de propiedades de puntos de referencia, dentro de la interfaz gráfica de usuario, está activa. Este mismo botón se utiliza para trazar los caminos o los contornos sobre la malla cuando la correspondiente ventana de propiedades está activa. El botón 2 se utiliza para cambiar la ventana de propiedades activa, en la interfaz gráfica de usuario.

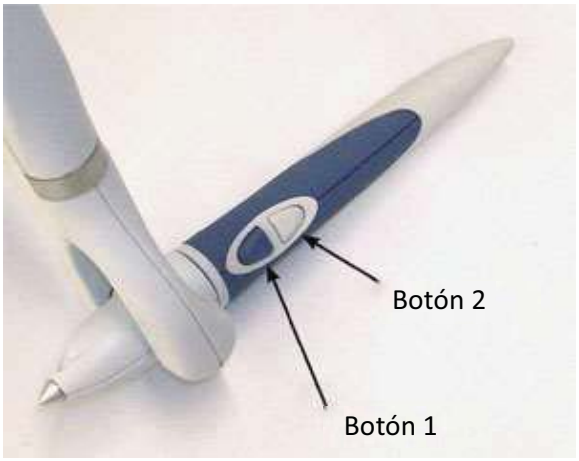


Figura 81. Botones en la pluma del dispositivo háptico PHANTOM OMNI

La integración del dispositivo háptico con el software para la edición de puntos de referencia desarrollado en este trabajo se muestra en la (Figura 82). Opcionalmente se puede utilizar un segundo dispositivo háptico PHANTOM OMNI para realizar el control de la cámara virtual. Cuando el segundo dispositivo háptico está activo, el botón 1 se utiliza para rotar la cámara virtual, el botón 2 controla la posición del pivote y, si ambos botones se mantienen presionando al mismo tiempo, se controla el factor de escala, o zoom, de la cámara virtual.

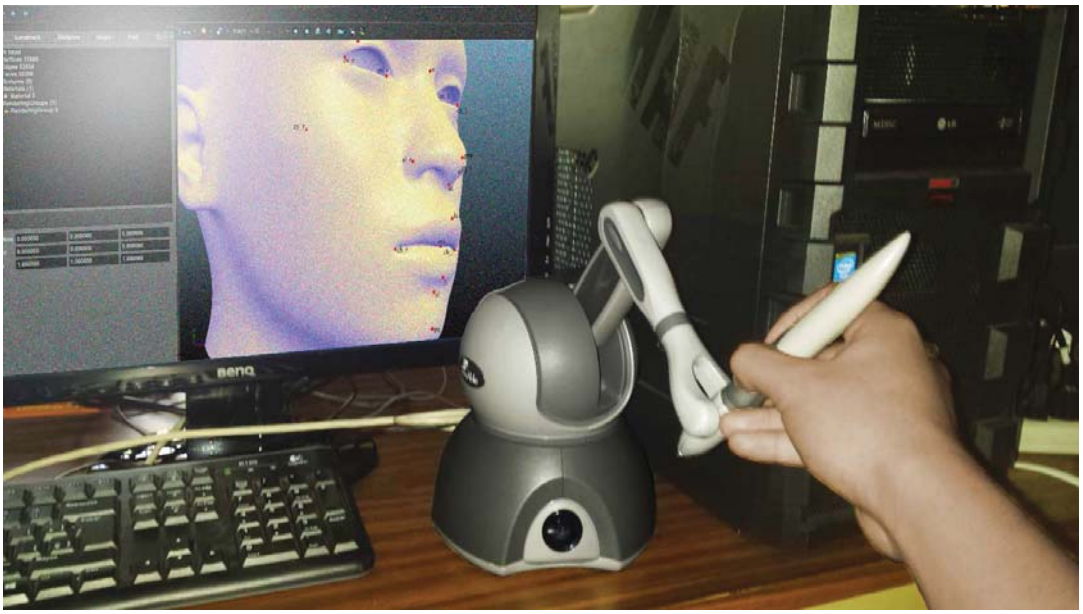


Figura 82. Integración de un dispositivo háptico PHANTOM OMNI al software para la selección interactiva de puntos de referencia desarrollado en este trabajo.

5 EXPERIMENTACIÓN

5.1 SELECCIÓN INTERACTIVA DE PUNTOS DE REFERENCIA

Utilizando el software para la edición de puntos de referencia desarrollado en este trabajo, se seleccionó de manera interactiva, el conjunto de 19 puntos de referencia escogidos para este estudio, sobre el modelo de la cabeza humana de referencia (Figura 83), y sobre cada una de los 35 modelos de cabezas (Figura 84, Figura 85), obtenidos de individuos sanos de nacionalidad mexicana, mediante la técnica de estereovisión con luz estructurada desarrollada en (López, 2015). Los puntos fueron posicionados sobre las mallas 3D, siguiendo la descripción de su localización en (Kolar and Salter, 1997, George, 2007), y se les asignó el nombre y etiquetas correspondientes. Para facilitar el procesamiento de los puntos de referencia, en forman de listas, los puntos fueron colocados en el mismo orden. Finalmente se generó un archivo CSV, por cada modelo, con los puntos de referencia respectivos, para su posterior análisis.

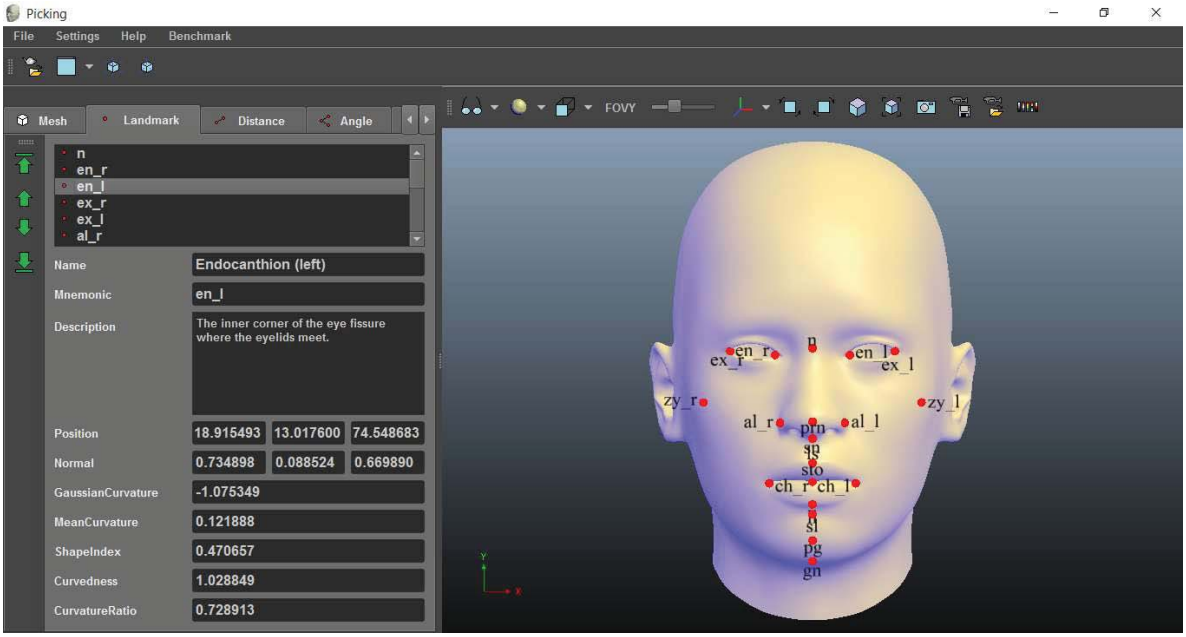


Figura 83. Selección interactiva de los puntos de referencia sobre el modelo 3D de la cabeza de referencia utilizada.

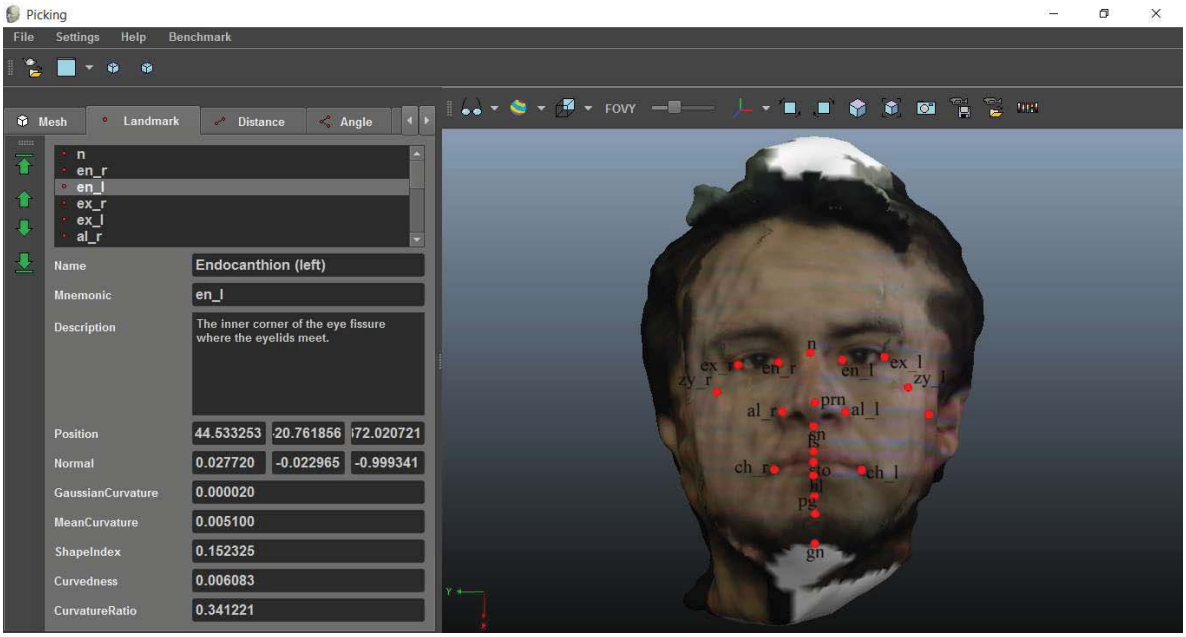


Figura 84. Selección interactiva de los puntos de referencia en una de las cabezas de la base de datos de sujetos mexicanos.

Figura 85. Algunos de los modelos de la base de datos de cabezas 3D de sujetos mexicanos, con sus correspondientes puntos de referencia señalados con puntos rojos.

5.2 REGISTRO DE MODELOS DE CABEZAS HUMANAS

Para el registro deformable de los modelos de las cabezas, utilizando el método de registro basado en jaulas, se construyó una malla de control de baja resolución, formada por 55 vértices y 106 triángulos, que envuelve a la cabeza de referencia (Figura 86). En el caso de la deformación de mallas basada en jaulas, las mallas de control son totalmente arbitrarias, con las condiciones de que sean cerradas y que contengan dentro de su volumen al objeto a deformar (Nieto and Susín, 2013). En los casos de prueba presentados en la literatura, las mallas de control son, por lo general, una versión de baja resolución de la malla original. En el método de deformación presentado en (Gao et al., 2009), los autores obtienen el modelo de baja resolución, de manera automática, mediante simplificación de la malla original, mientras que en (Xian et al., 2012) la jaula se genera automáticamente a partir del árbol de cajas delimitadoras orientadas (OBB).

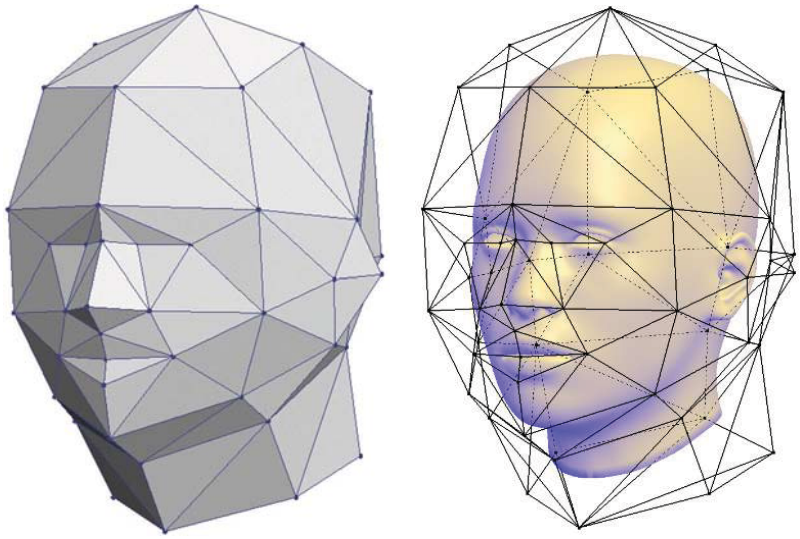


Figura 86. Malla de control o jaula para el modelo de la cabeza de referencia (a). La malla de control envuelve por completo al modelo de referencia (b).

En la (Figura 87) se ilustra el proceso de registro de los modelos de las cabezas realizado en este trabajo, en total se emplean cuatro métodos de registro diferente, con el fin de determinar con cuál de los métodos propuestos se obtiene un menor error de alineación. Primero se realiza por separado el registro rígido, del modelo de la adquisición con el modelo de referencia, basado en puntos de referencia correspondientes. Dado que los puntos de referencia superpuestos no coinciden totalmente, se procede a realizar un registro afín, de los puntos de referencia del modelo de referencia, hacia los puntos de referencia del modelo de la adquisición, transformados como consecuencia del registro rígido, con el objetivo de minimizar la distancia entre puntos de referencia correspondientes. En ambos casos existen diferencias entre la posición de los puntos de referencia correspondientes, por lo que se procede a realizar el registro deformable basado en jaulas, el cual tiene el efecto de desplazar los puntos de referencia sobre el modelo de referencia, conforme se deforma la malla de control, hasta hacerlos coincidir con los puntos de referencia del modelo de la adquisición, previamente transformados por el registro rígido. El número de iteraciones realizadas por el método de registro basado en jaulas se fijó en 100 iteraciones, para garantizar la convergencia, tal como se reporta experimentalmente en (Savoye, 2013).

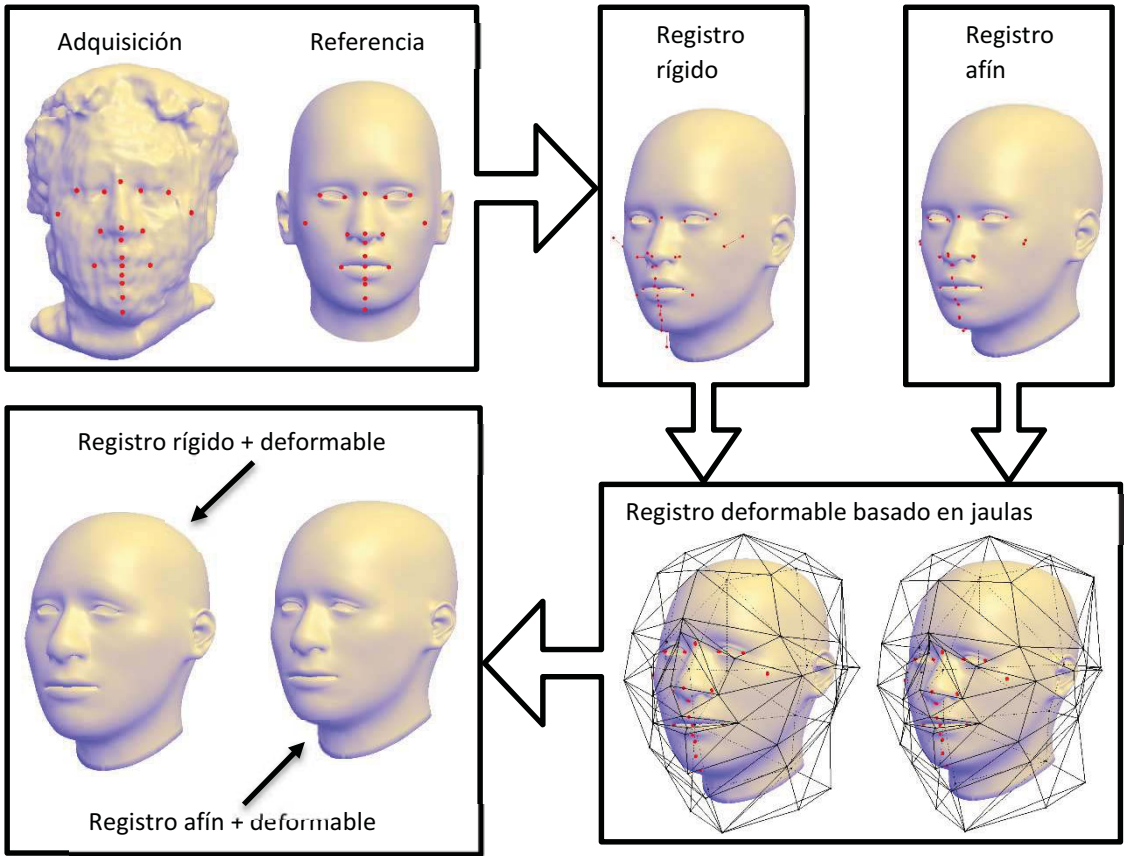


Figura 87. Proceso de registro de los modelos 3D de las cabezas.

5.3 ANÁLISIS DE COMPONENTES PRINCIPALES

Con el objetivo de determinar la variación de los puntos de referencia de la población de estudio, se realizó un análisis de componentes principales (PCA). Este análisis se realizó utilizando el software MATLAB, tomando como datos de entrada las listas en formato CSV de los puntos de referencia generados con el software de edición de puntos de referencia desarrollado. En la (Figura 88) se muestran los puntos de referencia extraídos directamente de las mallas correspondientes a las 35 cabezas de los sujetos en la población de estudio. El primer paso consiste en alinear datos de la población de estudio, con los puntos de referencia de la cabeza utilizada como referencia en este trabajo (Figura 89). Esto se realiza utilizando el método de registro rígido, determinando por cada conjunto de puntos correspondientes a cada una de las cabezas de la población, la transformación rígida que minimiza la distancia de puntos correspondientes con respecto a los puntos del modelo de referencia. Los puntos de referencia alineados se muestran en la (Figura 90).

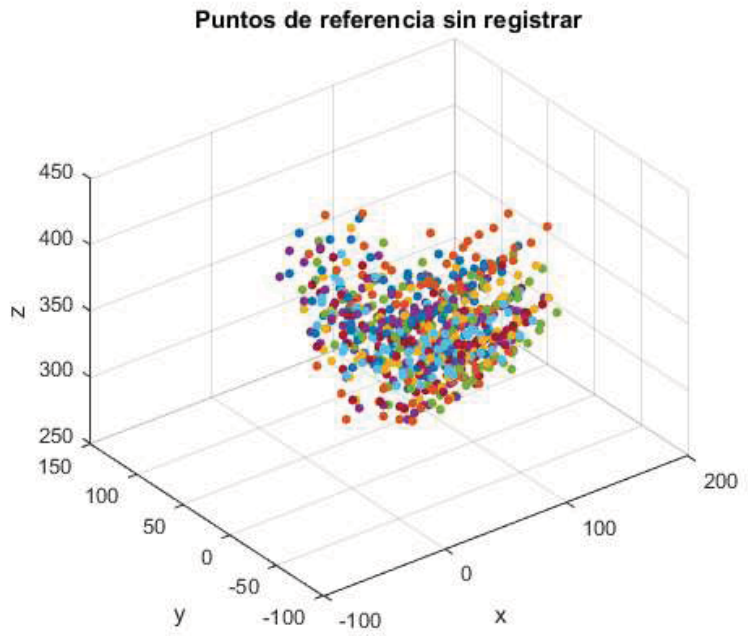


Figura 88. Puntos de referencia correspondientes a las 35 cabezas sin alinear, visualizados como una nube de puntos.

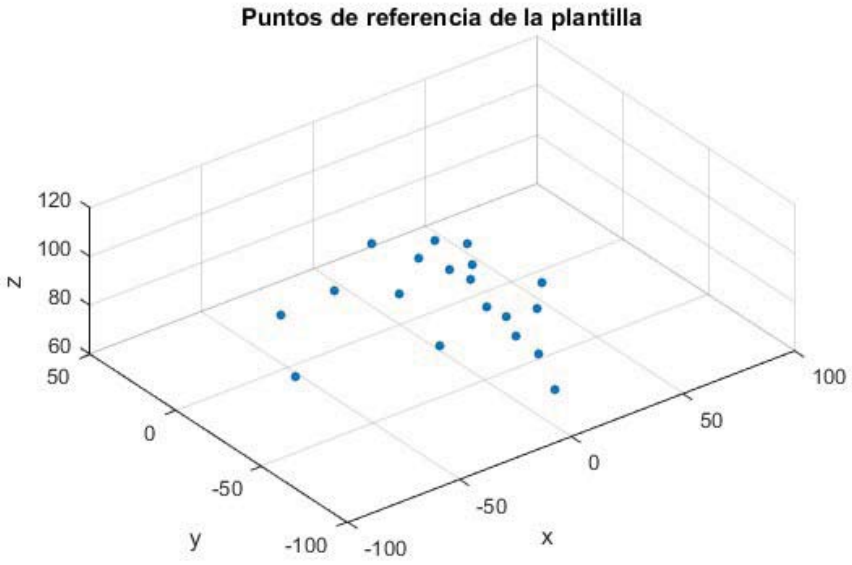


Figura 89. Puntos de referencia del modelo de referencia visualizados como una nube de puntos.

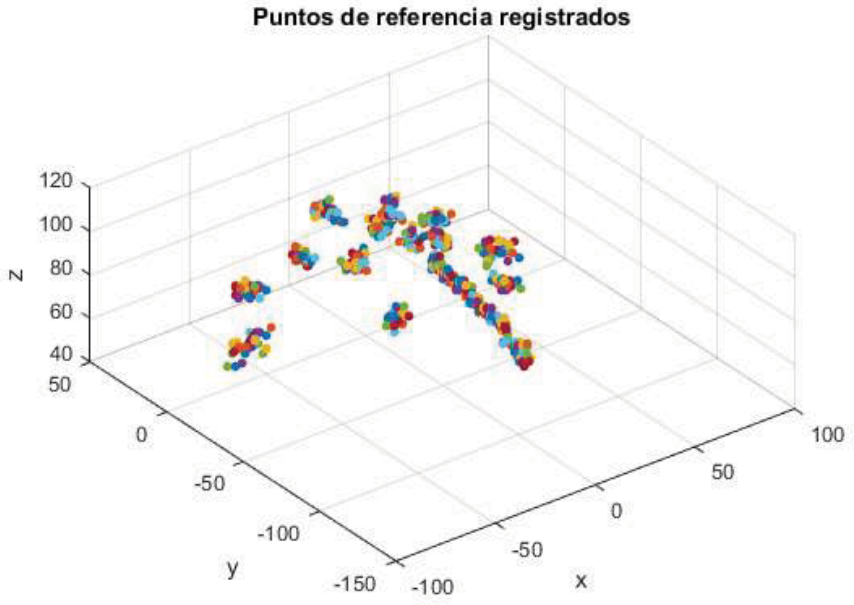


Figura 90. Puntos de referencia correspondientes a los 35 modelos de cabezas adquiridas, alineados con los puntos de referencia del modelo de referencia mediante registro rígido.

Una vez que los puntos de referencia de las 35 cabezas están alineados con los puntos de referencia de la cabeza de referencia, se procede a analizar por separado cada conjunto de puntos de referencia correspondientes. Primero se obtiene el promedio de cada conjunto de puntos, y se procede a calcular la matriz de covarianza. Finalmente se realiza la descomposición en vectores y valores propios de la matriz de covarianza, obteniéndose así las direcciones principales de variación de cada rasgo y la varianza en cada dirección.

5.4 ERROR TÉCNICO DE MEDICIÓN

Con el fin de determinar la variación en los puntos de referencia seleccionados sobre la misma cabeza, se realizó un análisis del error técnico de medición (TEM). Para ello un mismo observador seleccionó los mismos puntos de referencia, en 10 ocasiones distintas, sobre cada una de las mallas correspondientes a los 35 modelos de cabezas de la población de estudio. La diferencia en los puntos de referencia seleccionados en diferentes sesiones, se evidencia en la (Figura 91). El error técnico de medición se determinó con base a la distancia de cada punto al promedio del conjunto de puntos correspondientes, para cada una de las 35 cabezas. En la (Tabla 4) se muestra el error técnico de medición promedio de los puntos de referencia en las 35 cabezas.

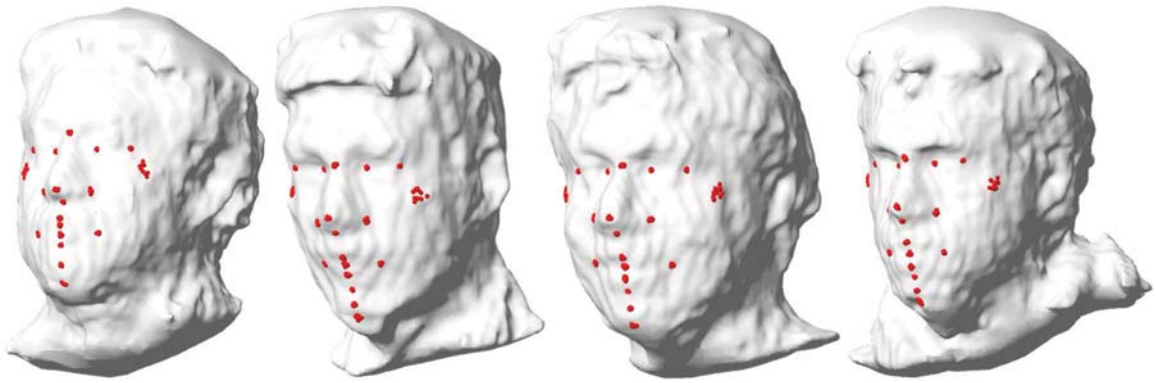


Figura 91. Algunos modelos de la base de datos de cabezas 3D de sujetos mexicanos, con puntos de referencia colocados en 10 sesiones diferentes.

Tabla 4. Error técnico de edición promedio para cada punto de referencia.

Punto de referencia	Error técnico de medición (mm)
Nasion	0.914829794092171
Endocanthion derecho	0.452682882036134
Endocanthion izquierdo	0.454545207035173
Exocanthion derecho	0.466100475985339
Endocanthion izquierdo	0.457194978879004
Alare derecho	0.929588963579049
Alare izquierdo	0.926440613378532
Subnasale	0.471082420757380
Labiale superius	0.899203802640971
Stomion	0.915350532786866
Labiale inferius	0.891914274706196
Gnathion	1.37979092217568
Cheilion derecho	0.679412154178660
Cheilion izquierdo	0.671982487667946
Sublabiale	0.441464942639540
Pogonion	0.658689531170999
Pronasale	0.896493887380740
Zygion derecho	3.47685705184875
Zygion izquierdo	3.33483981576932

5.5 CAMPO DE DISTANCIA EUCLIDIANA

El campo de distancia euclidiana (o a veces transformada de distancia euclidiana, EDT) (Marquez, 2008) se utilizó para evaluar el error en la alineación del modelo de referencia de la cabeza humana transformado, luego de aplicar cada uno de los métodos de registro propuestos en este trabajo, con cada una de las mallas originales en la base de datos de cabezas de sujetos mexicanos estudiada. Sea A un objeto en $\mathbb{R}^3$, con ∂A como frontera, en este caso, ∂A es la superficie de la malla

triangular cerrada que describe al objeto. El campo de distancia euclidiana con signo de A, se define como:

$$D_{\pm}(A) = \left\{ \left(\mathbf{p}, d(\mathbf{p}) \right) \mid d = \text{sgn} \cdot \min_{\mathbf{q} \in \partial A} |\mathbf{q} - \mathbf{p}| \right\}$$
$$\text{sgn} = \begin{cases} +1 & \mathbf{p} \in A^c \\ -1 & \mathbf{p} \in A \end{cases} \tag{118}$$

Esto es, el campo escalar formado por la distancia más cercana de cada punto $\mathbf{p}$ a la superficie de la malla ∂A, el signo se define positivo si $\mathbf{p}$ está en el exterior de A, y negativo en caso contrario. En nuestro caso, para determinar el error de alineación entre dos mallas triangulares A, B, se calcula el campo de distancia de ∂A restringido a ∂B, expresado como:

$$D_{\pm}(A)\big|_{\partial B} \tag{119}$$

Como una aproximación al campo de distancia restringido, se calcula la distancia más cercana de cada vértice de la malla B a la superficie de la malla triangular de A, utilizando el algoritmo del punto más cercano descrito por el (Pseudocódigo 3). Para determinar el signo se utiliza el algoritmo de inclusión de un punto en una malla poligonal cerrada, como se visualiza en la (Figura 22). La complejidad del algoritmo es $O(m \log n)$, en donde m es el número de vértices de la malla B y n es el número de triángulos de la malla A. En las (Figura 92, Figura 93) se muestra un corte 2D del campo de distancia euclidiana 3D de un conjunto de 5 esferas de igual radio, restringido a un plano que pasa por el centro de las esferas, visualizado usando dos mapas de color diferentes; el segundo visualiza conjuntos y curvas de isonivel.

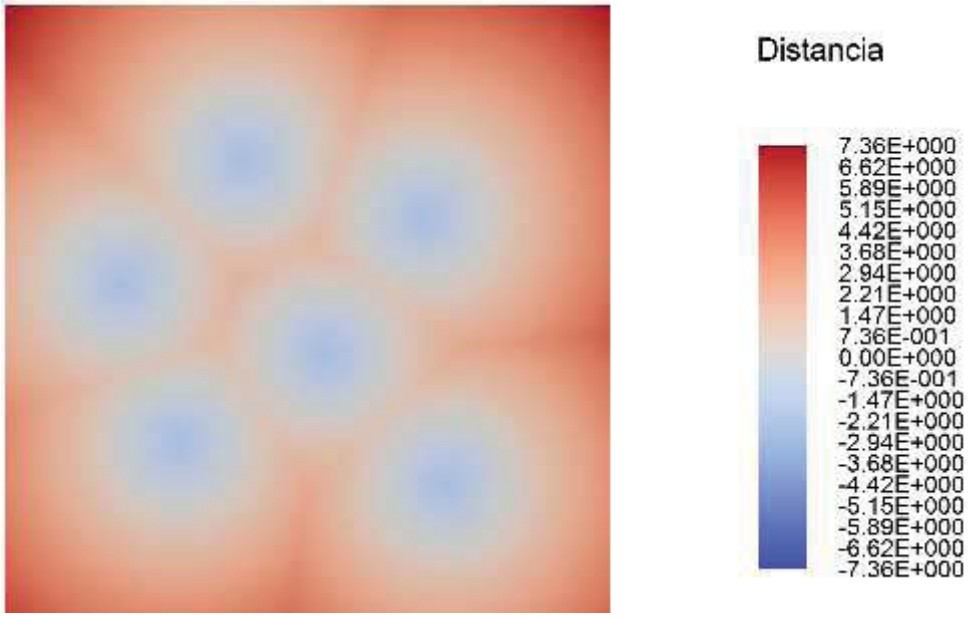


Figura 92. Campo de distancia muestreado sobre un plano de un conjunto de esferas del mismo tamaño, visualizado utilizando un mapa de color divergente.

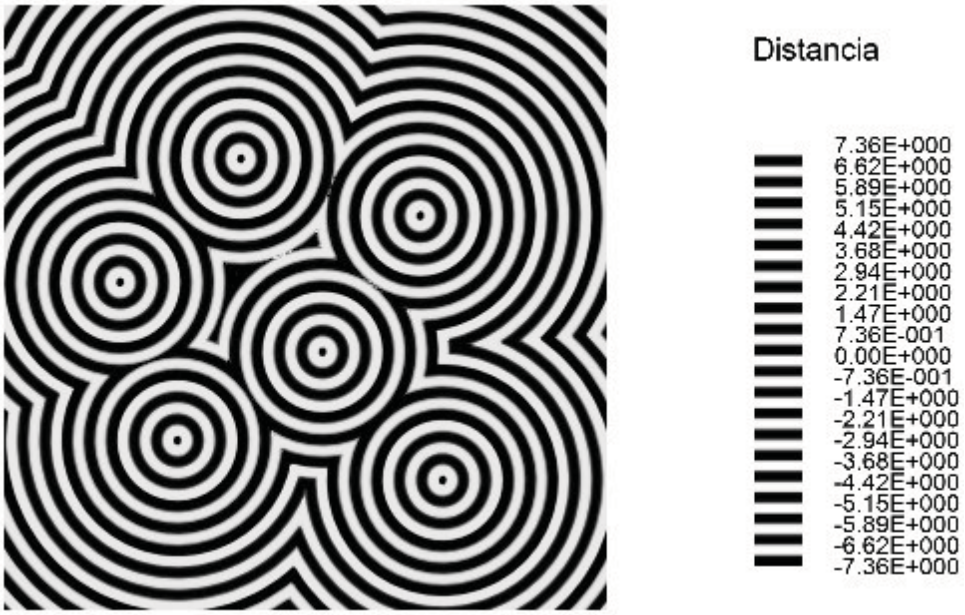


Figura 93. Campo de distancia de un conjunto de esferas visualizado con un mapa de color con colores alternados blanco-negro para observar curvas de equidistancia.

6 RESULTADOS

6.1 DESEMPEÑO DEL ALGORITMO DE INTERSECCIÓN ENTRE UN RAYO Y UNA MALLA TRIANGULAR

Para determinar la eficiencia de nuestra implementación del algoritmo de intersección entre un rayo y una malla triangular utilizando un octree, desarrollado en este trabajo para la selección interactiva de puntos sobre mallas tridimensionales (*Picking*), se comparó el tiempo de ejecución de nuestra versión contra las versiones implementadas por los motores de física PhysX[1] y Bullet[2]. La prueba consistió en medir el tiempo de ejecución de 100000 pruebas de colisión sobre diferentes modelos de alta resolución. Los modelos 3D seleccionados para esta prueba, mostrados en la (Figura 94), fueron elegidos debido a que son ampliamente utilizados en las publicaciones del área de gráficos por computadora, para realizar pruebas de desempeño de algoritmos gráficos y en simulación.

(a) (b) (c) (d) (e) (f)

Figura 94. Modelos utilizados para la prueba del algoritmo de colisión entre un rayo y una malla: Bunny[3] (69451 triángulos) (a), Igea[4] (268686 triángulos) (b), Armadillo[3] (345944 triángulos) (c), Dragon[3] (869928 triángulos) (d), Happy[3] (1087138 triángulos) (e), Ramesses[5] (1652528 triángulos) (f).

Para que las pruebas de colisión se realizaran en iguales condiciones, en cada prueba se calculó el origen del rayo alejado 100 unidades del centroide de cada triángulo en la dirección de la normal, y la dirección del rayo se estableció en dirección opuesta a la normal del respectivo triángulo, esto para garantizar que en cada prueba el rayo intersectara a la malla. Las pruebas se realizaron sobre cada modelo 10 veces, luego se promediaron los tiempos para obtener un tiempo representativo en milisegundos, el cual luego se divide entre 100000 para obtener el tiempo por prueba simple de intersección de un rayo contra la malla triangular. Los resultados de esta prueba comparativa se muestran en la gráfica de la (Figura 95). Para apreciar mejor las diferencias de tiempo, el eje vertical se muestra en escala logarítmica.

[1] http://www.geforce.com/hardware/technology/physx
[2] http://bulletphysics.org/wordpress/
[3] http://graphics.stanford.edu/data/3Dscanrep/
[4] http://gfx.cs.princeton.edu/proj/sugcon/models/
[5] https://www.rocq.inria.fr/gamma/gamma/download/download.php

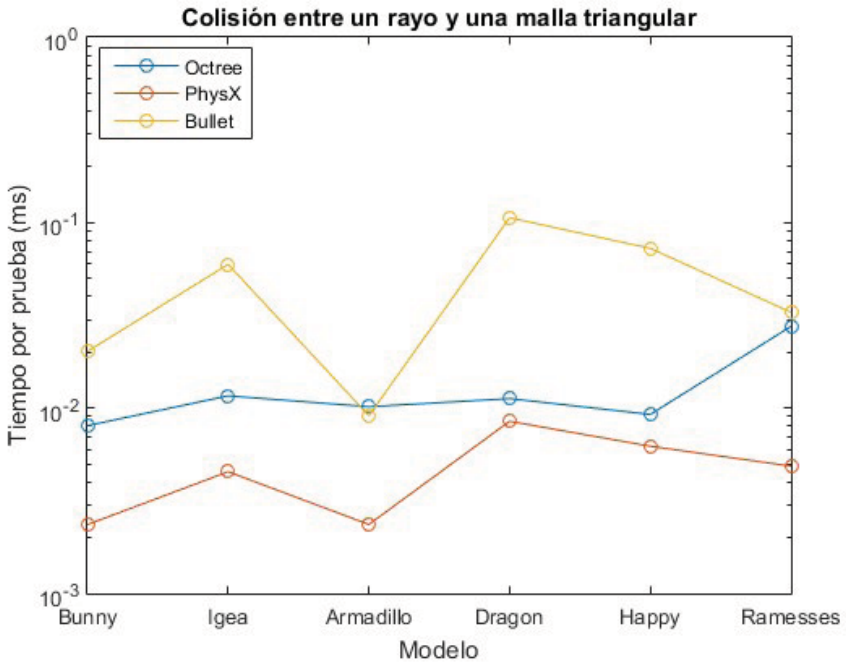


Figura 95. Gráfica comparativa con tiempos de ejecución de la prueba de colisión entre un rayo y una malla triangular.

Observando el comportamiento general de la gráfica, la implementación con mejor desempeño, en cuanto a tiempo de ejecución, corresponde al motor de física PhysX; nuestra implementación se posiciona en segundo lugar seguida del motor de física Bullet. Aún con el modelo más complejo utilizado en las pruebas, con poco más de un millón y medio de triángulos, todos corren en tiempo real, a una frecuencia mayor a 1 KHz. El contar con un algoritmo rápido es muy importante para la selección interactiva, ya que esto elimina posibles errores debido a la falta de sincronización entre lo que el usuario ve y lo que el usuario selecciona. La razón por la cual la implementación de PhysX es más rápida, es debido a que utiliza árboles de volúmenes delimitadores (VBH) optimizados para realizar las pruebas de colisión en paralelo utilizando la GPU; mientras que nuestra implementación y la de Bullet, corren completamente en la CPU. En el caso de la implementación de Bullet, ésta utiliza árboles de partición binaria del espacio (BSP-tree) para optimizar el algoritmo de colisión. Por otra parte, nuestra implementación utiliza árboles octales u octrees para acelerar las pruebas de colisión, lo cual se ve reflejado en los tiempos de ejecución.

Analizando los tiempos de inicialización de las estructuras de datos, requeridas por las distintas implementaciones analizadas, según se observa en la gráfica de la (Figura 96), la implementación de PhysX es la requiere mayor tiempo de inicialización, debido a la sobrecarga de la transferencia de datos entre la CPU y la GPU; por otra parte, la implementación que requiere menor tiempo de inicialización, es la implementación de Bullet, debido a que la estructura BSP-tree es más simple de construir que la estructura de árbol octal en nuestra implementación. La razón por la que en este trabajo se prefirió el uso de nuestra implementación por encima de los motores PhysX y Bullet, es debido a que la misma estructura de octree se utiliza para las pruebas de inclusión de un punto dentro de la malla, durante el renderizado háptico, y para el cálculo del campo de distancia euclidiana para evaluar el error de alineación durante el registro deformable de los modelos, características que no están implementadas en las bibliotecas antes mencionadas; además, como

ya se demostró con las pruebas de ejecución, el rendimiento de nuestra implementación es comparable a las utilizados por los motores de física comerciales contra los cuales se realizó el análisis comparativo.

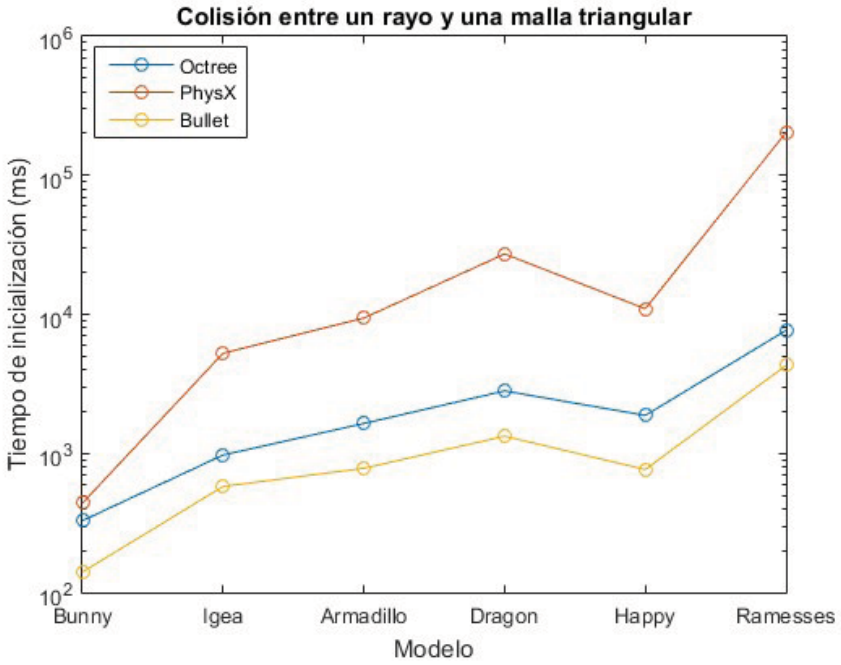


Figura 96. Gráfica comparativa con los tiempos de inicialización de las estructuras de datos para distintas implementaciones del algoritmo de colisión entre un rayo y una malla triangular.

6.2 ERROR DE ALINEACIÓN EN MODELOS DE CABEZAS HUMANAS

Para evaluar el error en la alineación en cada uno de los métodos de registro utilizados en este trabajo, se procedió a extraer la región del rostro del modelo de referencia utilizado, con el fin de eliminar los errores generados por la presencia de los detalles del cuello, orejas y parte superior y posterior del cráneo en el modelo de referencia, ya que en los modelos de las adquisiciones sobre sujetos reales, no se tiene esa información, y por otra parte, en el conjunto de puntos de referencia utilizados para este trabajo, no se tienen muestras en esa zonas.

La extracción de la zona de la cara se realizó tal como se propone en (Nair and Cavallaro, 2009), utilizando una esfera de recorte centrada en el borde de la nariz y con radio $r = 2.6l_n$, siendo l_n la longitud de la nariz, como se ilustra en la (Figura 97).

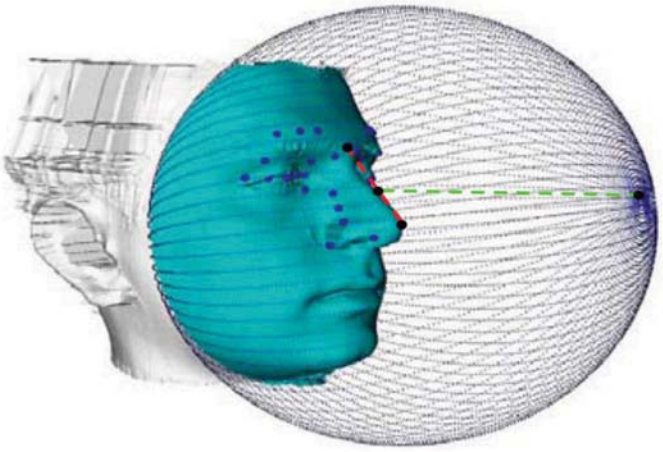


Figura 97. Segmentación de la cara, esfera de recorte centrado sobre el borde la nariz, con radio (verde) en función de la longitud de la nariz (rojo). Extraído de (Nair and Cavallaro, 2009).

En nuestro caso el centro de la esfera se define como el punto medio entre los puntos de referencia Nasion y Pronasale, y la longitud de la nariz como la distancia entre esos mismos dos puntos. La extracción se realizó utilizando el software de modelado 3D Autodesk Maya, para ello se generó una esfera poligonal altamente teselada con el centro y radio calculado a partir de los puntos de referencia, sobre el modelo 3D de la cabeza de referencia. Posteriormente se realizó la intersección booleana entre las dos mallas para obtener la malla de la zona del rostro (Figura 98).

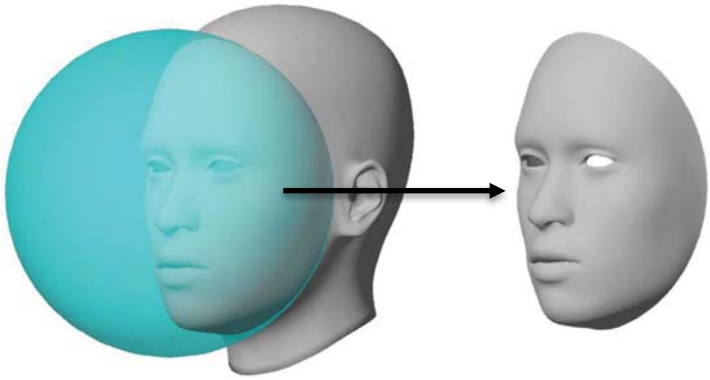


Figura 98. Proceso de extracción de la región del rostro del modelo de referencia de la cabeza, utilizando una esfera de recorte.

Luego de que la cara del modelo de referencia de la cabeza humana fue extraída, se procedió a aplicar los métodos de registro de la cara segmentada con cada uno de los modelos 3D en la base de datos de cabezas de sujetos mexicanos utilizada en este trabajo. En la (Figura 99) se muestra el modelo de la cara segmentada, tras aplicarle con los diferentes métodos de registro, superpuesto con una de las cabezas de la base de datos.

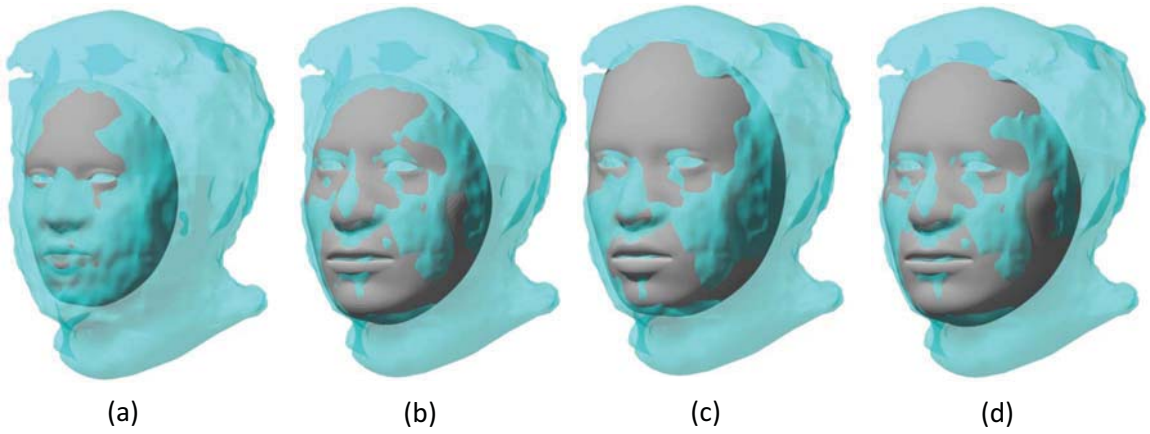


Figura 99. Alineación de modelos utilizando registro rígido (a), registro rígido seguido de registro deformable (b), registro afín (c), y registro afín seguido de registro deformable (e). El modelo de la adquisición se muestra en color azul translúcido, el modelo de referencia registrado se muestra en color gris opaco.

Después se procede a calcular el campo de distancia euclidiana, de la malla hacia la cual se está realizando el registro, es decir, del modelo adquirido por estereovisión, restringido a la superficie de la cara segmentada. En la (Figura 100) se muestra el campo de distancia euclidiana restringido, de una de las cabezas de la base de datos, para cada uno de los métodos de registro utilizados, utilizando un mapa de color divergente. Los colores azulados, asociados a los valores negativos, significan que la superficie de la malla se encuentra al interior de la malla de la cabeza que se obtuvo por estereovisión, mientras que los colores rojizos, asociados a los valores positivos, significan que la malla estuvo en el exterior del modelo adquirido por estereovisión tras la alineación. Los colores más claros, denotan que la distancia entre las superficies tiende a ser cero.

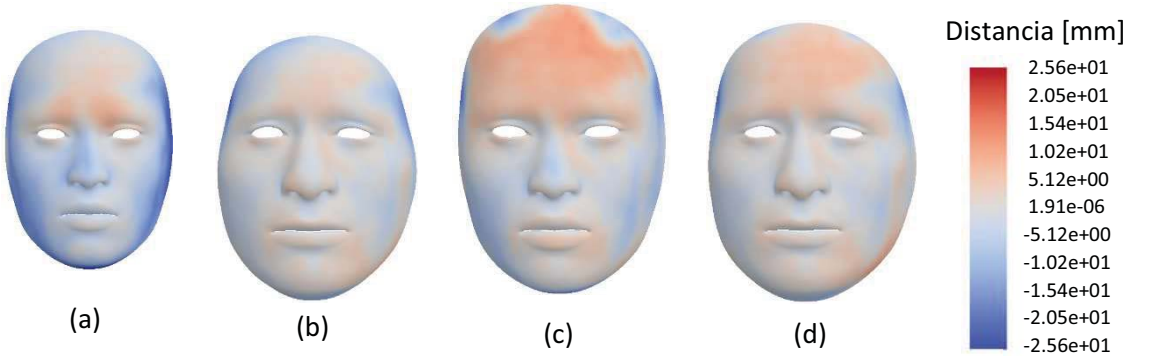


Figura 100. Campo de distancia euclidiana del modelo de la adquisición, muestreado desde la superficie del modelo alineado mediante registro rígido (a), registro rígido seguido de registro deformable (b), registro afín (c) y registro afín seguido de registro deformable (d).

Como una medida del error en la alineación, se calcula el error en la alineación de los puntos de referencia, del modelo trasformado debido al método de registro aplicado, y del modelo de la adquisición hacia el cual se realiza el registro, para ello objetemos el error cuadrático medio (RMS), como:

$$e_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \tag{120}$$

En donde d es la distancia euclidiana entre puntos de referencia correspondientes de los modelos alineados. El error cuadrático medio se calcula para cada método de registro y para cada una de las cabezas en la base de datos de cabezas humanas obtenidas por estereovisión. Para observar el comportamiento general de los datos, se utiliza un "diagrama de caja" (box plot), como se muestra en la gráfica de la (Figura 101), en donde se representa de manera gráfica la mediana, y el primer y el tercer cuartil.

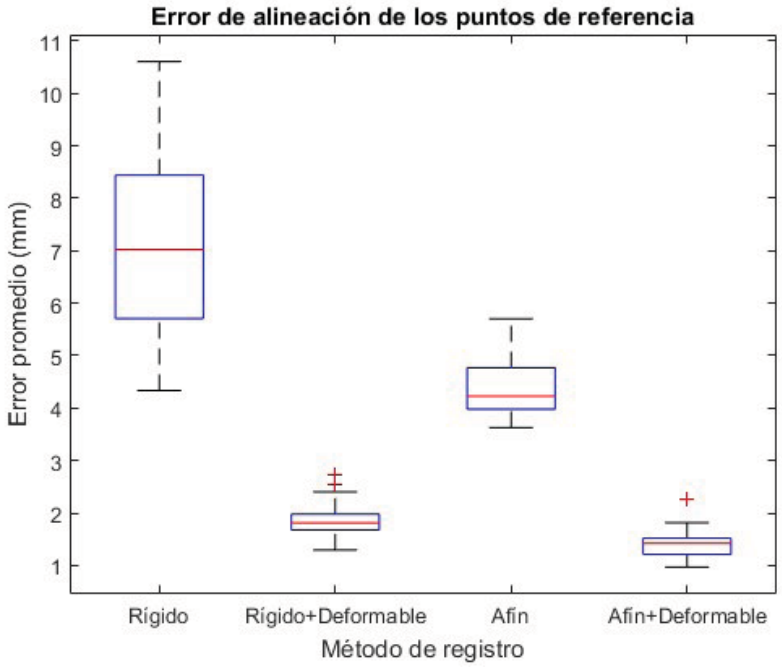


Figura 101. Diagrama de caja del error promedio de la alineación de los puntos de referencia de las 35 cabezas.

Observando la gráfica de la (Figura 101) se aprecia a simple vista una gran diferencia entre el error obtenido mediante el método de registro rígido y el método de registro afín, sin embargo, en el caso de los métodos de registro con deformación, no es posible determinar a simple vista si existe una diferencia significativa, por lo que se procederá a realizar un análisis de la varianza en el método rígido con deformación y afín más deformación, con el fin de verificar que la diferencia entre los resultados es estadísticamente significativa. Primero se realizó una prueba de Kolmogorov–Smirnov para saber si los datos siguen una distribución normal (Wilcox, 2005). Para esta prueba se utilizó el software MATLAB con la función *kstest*[1], cuyo resultado fue que los datos sí siguen una distribución normal, por lo tanto, al ser paramétricos, podemos aplicar un análisis de la varianza ANOVA utilizando la función de MATLAB anova1[2]. El resultado del análisis de la varianza ANOVA aplicado a los datos de errores promedio correspondientes al registro rígido seguido de deformación y registro afín seguido de deformación se muestra en la (Figura 102), en donde es posible apreciar que ambos grupos resultaron ser significativamente diferentes.

[1] https://www.mathworks.com/help/stats/kstest.html
[2] https://www.mathworks.com/help/stats/anova1.html

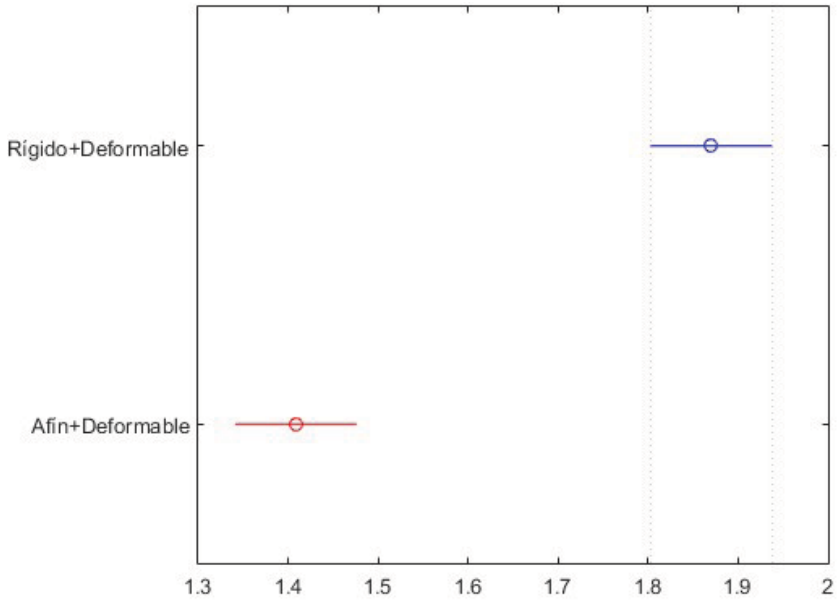


Figura 102. Resultado gráfico del análisis de la varianza ANOVA; en donde se aprecia que los datos son significativamente diferentes.

Para obtener un valor representativo del error, asociado al campo de distancia euclidiana restringido a la superficie de la malla registrada, se calculó el valor RMS de las distancias en cada campo de distancia, el cual se considerará para este trabajo, como el error de la alineación de las mallas. La distribución estadística de los resultados se visualiza en el diagrama de caja y bigotes de la (Figura 103).

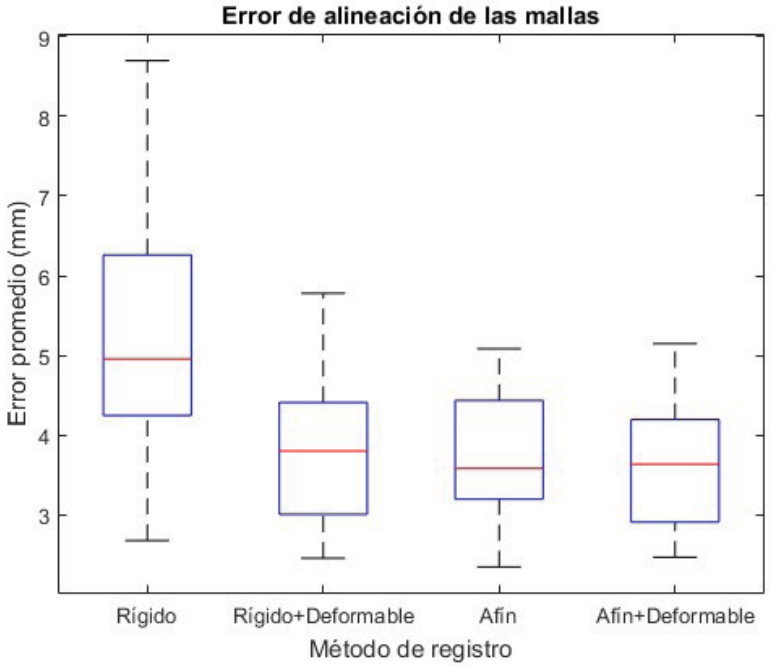


Figura 103. Diagrama de caja y bigotes del error cuadrático medio en la alineación de las mallas.

En general, se observa que el mayor error, en la alineación de los puntos de referencia, se presenta en el método de registro rígido, esto debido a que la malla no se deforma, sino que es trasladada y rotada mientras se minimiza la distancia entre puntos de referencia correspondientes, por lo tanto, estos valores son una buena referencia para evaluar los demás métodos, por lo que se espera que este error se reduzca, tal como sucede en el método de registro afín, en donde el error en la alineación se reduce en un 38%, con respecto al método de registro rígido, esto se debe que la malla, además de trasladarse y rotar, puede contener escalado anisotrópico y sesgo, con lo cual se reduce el error de alineación en los puntos. Por otra parte, tras aplicar el método de registro deformable basado en jaulas, al resultado obtenido mediante registro rígido y registro afín, se observa una reducción del error en un 70% en ambos casos. El método con menor error de alineación en los puntos de referencia, corresponde al método de registro afín seguido del registro deformable basado en jaulas. En el caso del error en la alineación de las mallas, el error en el método de registro rígido se reduce en un 26% tras aplicar el método de registro deformable basado en jaulas, en un 28% con el método de registro afín y en 30% con el método de registro afín seguido del método de registro deformable basado en jaulas. Se observa en la (Figura 104) que la reducción en el error en la alineación, entre los tres últimos métodos, no es tan significativo como se esperaría, debido a la malla original presenta ruido de la adquisición y detalles de alta frecuencia que no pueden aproximarse mediante la deformación del modelo de referencia. Aun así, el método de registro afín combinado el método de registro deformable basado en jaulas, es el que presenta el menor error de alineación en las mallas.

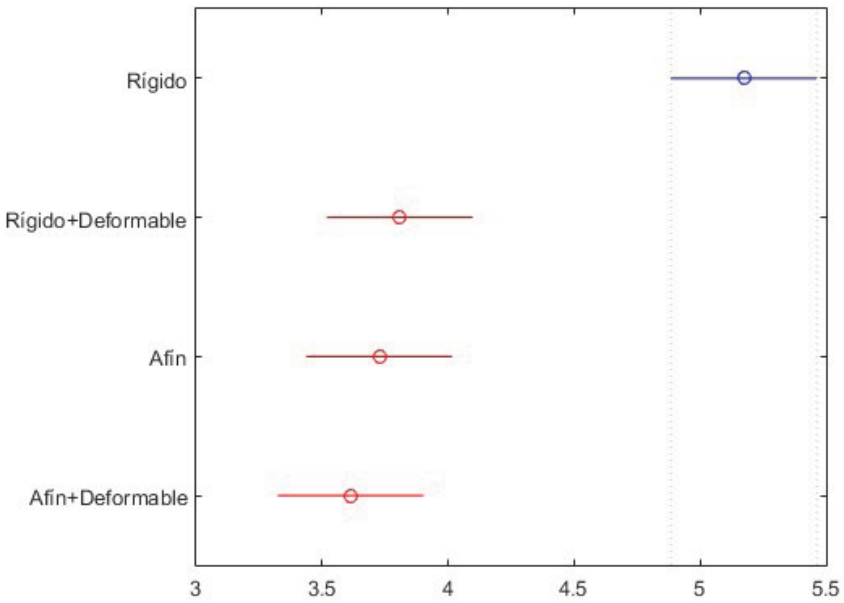


Figura 104. Resultado gráfico del análisis de la varianza ANOVA, en donde se aprecia que los métodos de registro rígido con deformación, afín y afín con formación no presentan diferencias estadísticamente significativas entre sí.

Posteriormente, se transfirió el color por vértice, presente en las mallas originales, de las cabezas en la base de datos, hacia los modelos aproximados, obtenidos mediante registro rígido combinado con registro deformable basado en jaulas. Esto se realizó visitando cada vértice de la malla de referencia registrada, y localizando en la otra malla el vértice más cercano, utilizando el algoritmo descrito en el (Pseudocódigo 3). Los resultados visuales se observan en la (Figura 105).

Figura 105. Algunos modelos de cabezas de la base de datos de sujetos mexicanos, junto con el modelo registrado con los colores por vértice trasferidos.

En algunos casos, las mallas registradas no se asemejan con las mallas originales en la zona de las mejillas, esto es debido a que en esas zonas no se tomaron puntos de referencia, por lo que no se tiene información suficiente para reconstruir esa parte. En otras mallas se observa deformación excesiva y asimetría, esto se debe a que los modelos adquiridos mediante estereovisión, presentan ruido y artefactos debidos al proceso de reconstrucción, y los sujetos presentan distintas posturas, dificultando la localización de algunos puntos. Por ejemplo, en algunos modelos la nariz está aplanada, así como en la zona de la boca y del mentón, por lo que los puntos vistos de perfil, no presentan mucha diferencia en las alturas. Debido a esto, al realizar el registro deformable de la malla de referencia, ésta tiende a aplanarse, a fin de minimizar el error de correspondencia con estos puntos, haciendo que los demás rasgos se distorsionen.

6.3 ANÁLISIS DE COMPONENTES PRINCIPALES DE LOS PUNTOS DE REFERENCIA

En la (Figura 106) se muestran los componentes principales, del conjunto de puntos de referencia de las 35 en la base de datos estudiada, alineados mediante registro rígido. Como resultado de este análisis, se obtuvo los puntos de referencia promedio, de la población de estudio, así como las principales direcciones de variación, de cada uno de los puntos de referencia. Estos resultados se muestran en forma de elipsoides en la (Figura 107). La varianza y desviación estándar de cada punto de referencia medidas sobre las direcciones principales correspondientes se enlistan en la (Tabla 5).

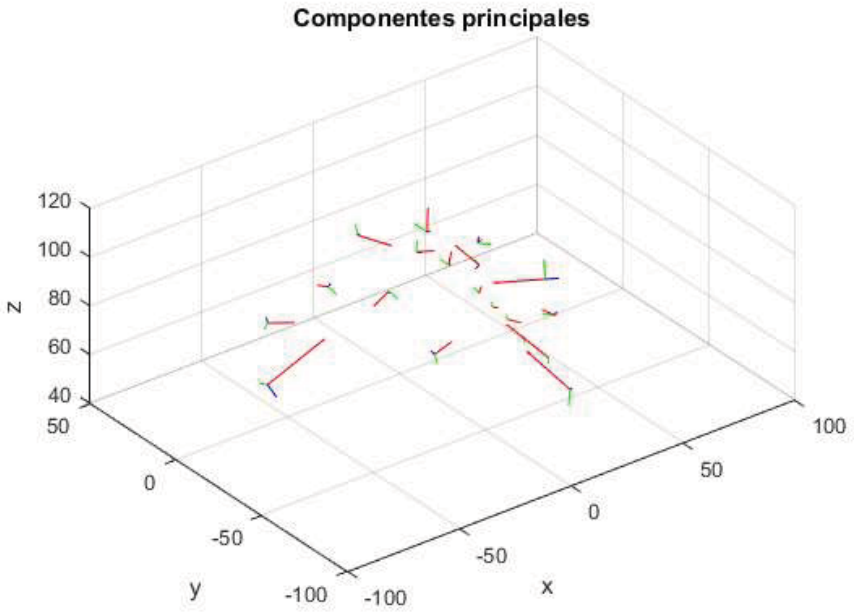


Figura 106. Componentes principales graficados como vectores, el origen de cada conjunto de tres vectores formando una base ortogonal, corresponde a la forma promedio, los vectores de color rojo corresponden a la dirección principal, la magnitud de los vectores es igual a la varianza en cada dirección.

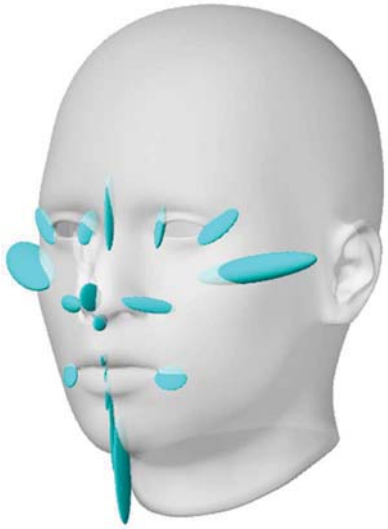


Figura 107. Componentes principales interpretados como elipsoides sobre la cabeza de referencia, el centro de los elipsoides corresponde a la forma promedio y los radios corresponden a las direcciones principales de variación con magnitud igual a la varianza.

Tabla 5. Varianza y desviación estándar en la dirección principal para cada punto de referencia en orden descendente.

Punto de referencia	Varianza (mm^2)	Desviación estándar (mm)
Zygion izquierdo	28.9066104	5.37648681
Zygion derecho	26.6506225	5.16242409

Gnathion	26.081464	5.10700147
Nasion	18.0827145	4.25237751
Pogonion	14.9568077	3.86740323
Alare izquierdo	14.0379912	3.74673074
Endocanthion izquierdo	12.0780977	3.47535577
Endocanthion izquierdo	10.6756476	3.26736096
Sublabiale	10.3682991	3.21998432
Exocanthion derecho	10.2021073	3.19407378
Pronasale	10.1428082	3.18477758
Endocanthion derecho	8.77819884	2.96280253
Cheilion izquierdo	7.80331156	2.79344081
Alare derecho	7.79602903	2.792137
Cheilion derecho	7.6493805	2.76575135
Labiale inferius	7.63971317	2.76400311
Subnasale	6.62332932	2.57358297
Labiale superius	3.7715417	1.94204575
Stomion	3.43836475	1.85428281

Los puntos de referencia (ver sección 2.3.1) con mayor varianza en la dirección principal, corresponden a Zygion (zy) derecho e izquierdo, con una varianza de 28.9 y 26.7 mm^2 correspondientemente, Gnathion (gn) con 26.1 mm^2, Nasion (n) con 18.1 mm^2 y Pogonion (pg) con 15 mm^2. La varianza alta en los puntos Zygion, ubicados a los lados del rostro, refleja que existe una alta variación en el ancho de la cabeza de los sujetos en la base de datos, esto se verifica al observar que los vectores propios correspondientes, son casi horizontales. El punto Gnathion localizado en la parte más baja de la barbilla, para el cual se observa varianza alta, refleja una alta variación en el largo de la barbilla en la población de estudio, lo cual se verifica al observar que la dirección del vector propio correspondiente es casi vertical.

Los puntos de referencia con menor varianza corresponden a Stomion (sto) con 3.4 mm^2, Labiale superius (ls) con 3.8 mm^2, Subnasale (sn) con 6.6 mm^2 y Labiale inferius con 7.6 mm^2. Todos estos puntos tienen en común el hecho de que se encuentran sobre el plano sagital, y las direcciones principales se encuentra sobre este mismo plano, además de localizarse cerca de la boca. La baja variación de esto puntos, refleja que, dentro de la población estudiada, existe muy poca variación en la altura de los labios, y de la región existente entre el borde del labio superior y el punto más bajo de la unión entre la nariz y el rostro.

6.3.1 Rostro promedio de la población de estudio

Como parte del análisis de componentes principales, se obtuvo el conjunto promedio de puntos de referencia, correspondiente a la población estudiada. Estos puntos fueron utilizados para aproximar la forma de la cara promedio de los sujetos que conforman la base de datos utilizada en este trabajo. Para ello se retomó el rostro segmentado del modelo de referencia, y se aplicó el método de registro afín, combinado con registro deformable basado en jaulas. De esta forma se obtuvo un rosto cuyos puntos de referencia corresponden al promedio de los puntos de referencia de la población de estudio. Como se ha demostrado que el registro afín, junto con el registro deformable basado en jaulas, es el que registra menor error de alineación entre las mallas, el rostro obtenido mediante

este método es una buena aproximación. Finalmente se obtuvieron los colores del rostro promedio, asignando a cada vértice del modelo deformado, el promedio de los colores correspondientes al punto más cercano, en cada uno de los modelos de las 35 cabezas en la base de datos. El resultado se observa en la (Figura 108). El valor promedio global del color RGB es $(114.0333, 95.9903, 80.9214)$, con una desviación estándar por canal de $(34.6567, 32.6444, 30.7037)$, el histograma de color se muestra en la (Figura 108).

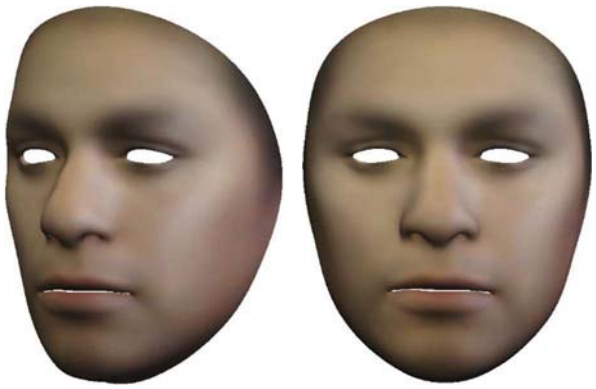


Figura 108. Rostro promedio de la población de estudio, obtenido mediante registro afín combinado con registro deformable basado en jaulas, de un modelo de referencia del rostro humano.

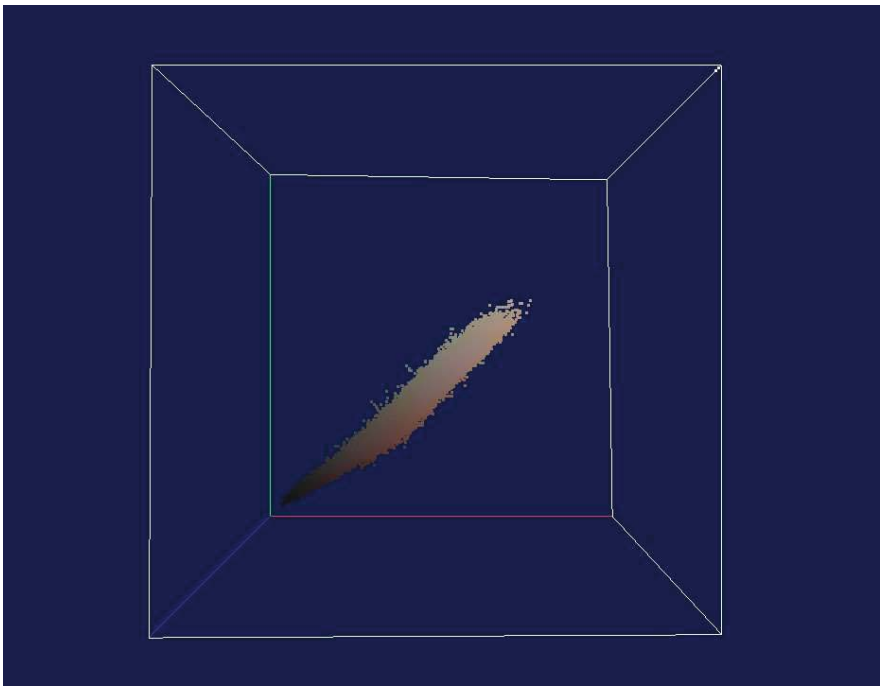


Figura 109. Histograma de color de las caras en el espacio RGB. La presencia de tonalidades verde y azul se debe al patrón de luz proyectado durante la adquisición de los modelos.

6.3.2 Generación de modelos de cabezas humanas aleatorias

Uno de los objetivos del análisis de componentes principales, fue obtener un modelo de distribución de puntos (PDM), con el fin de generar cabezas humanas aleatorias, cuyos puntos de referencia tienen la misma distribución estadística, que los puntos de referencia correspondientes a los

modelos en la base de datos estudiada. Utilizando un modelo de distribución de puntos, cualquier conjunto de puntos de referencia puede ser aproximado como:

$$\tilde{\mathbf{X}} = \overline{\mathbf{X}} + \mathbf{Pb} \tag{121}$$

En donde $\overline{\mathbf{X}}$ es la matriz correspondiente al conjunto promedio de los puntos de referencia, $\mathbf{P}$ es una matriz formada por los vectores propios unitarios $\mathbf{v}_i$, correspondientes al máximo valor propio λ_i para cada punto de referencia, y $\mathbf{b}$ es vector de factores de escala b_i asociados a cada uno de los puntos de referencia. Para asegurar que los conjuntos de puntos de referencia $\tilde{\mathbf{X}}$ generados utilizando este modelo, los valores de b_i se eligen en el intervalo $\pm 3\sqrt{\lambda_i}$, es decir, tres veces la desviación estándar de cada punto de referencia sobre a lo largo de la dirección principal. Los puntos de referencia aleatorios generados con este modelo se observan en la (Figura 110).

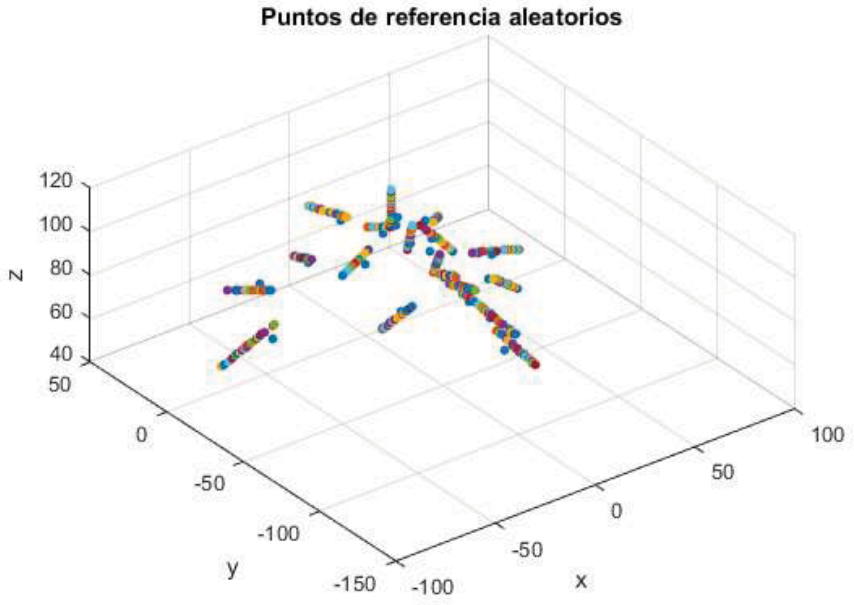


Figura 110. Puntos de referencia aleatorios, generados utilizando un modelo de distribución de puntos.

Utilizando el conjunto de puntos de referencia generados aleatoriamente, se procedió a realizar el registro afín, combinado con registro deformable basado en jaulas, de los puntos de referencia del modelo de referencia de la cabeza humana utilizado en este trabajo, hacia cada uno de los conjuntos de puntos de referencia aleatorios obtenidos, con el fin de aproximar los modelos aleatorios de cabezas humanas, cuyos puntos de referencia tienen la misma distribución estadística, que los puntos de referencia correspondientes a los modelos en la población de estudio. Algunos de los modelos obtenidos con este método se muestran en la (Figura 111).

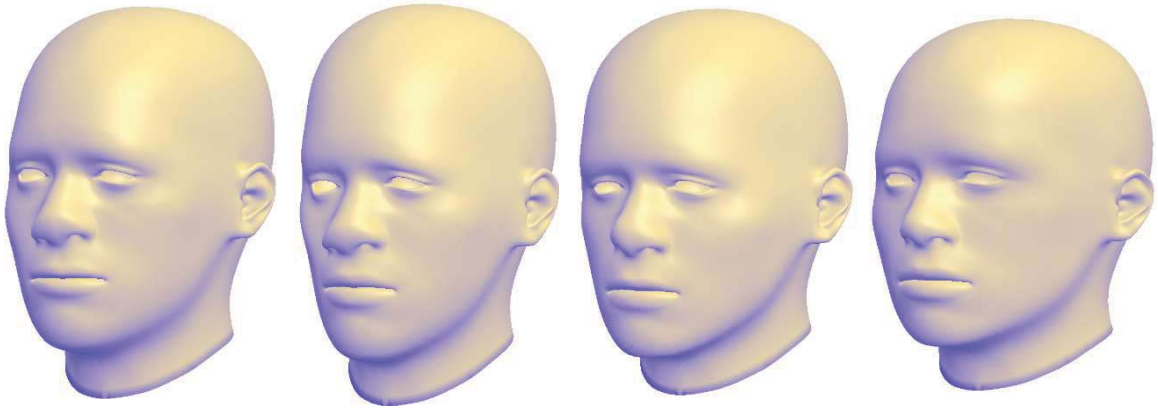


Figura 111. Algunos modelos de cabezas humanas generados de manera aleatoria, mediante registro afín combinado con registro deformable basado en jaulas.

6.4 ERROR TÉCNICO DE MEDICIÓN

En la (Figura 112) se muestra el error técnico de medición en cada punto de referencia. Analizando el comportamiento general del diagrama de caja en donde se representa de manera gráfica la media y el primer y tercer cuartil, se observa que, exceptuando a los puntos 18 y 19, que corresponden a los puntos Zygion izquierdo y derecho respectivamente, el error técnico de medición está por debajo de los 2 mm, entre los cuales sobresale el punto 12, correspondiente al punto Gnathion. Los puntos con menor error de medición, corresponden a los puntos 2, 3, 4, 5, con errores inferiores a medio milímetro. Estos puntos están localizados en las uniones de los párpados, por lo cual son fáciles de localizar visualizando la textura de color de los modelos.

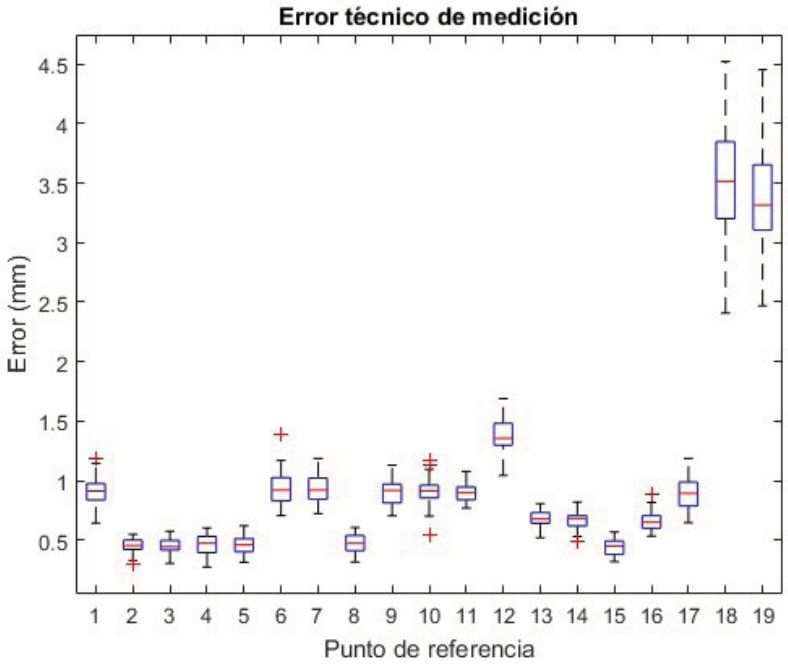


Figura 112. Diagrama de caja y bigotes mostrando la distribución del error técnico de medición en cada punto de referencia.

Por otra parte, los puntos con mayor error técnico de medición, corresponde a los puntos Zygion izquierdo y derecho, con un error promedio de 3.5 y 3.3 mm respectivamente. A diferencia de los otros puntos de referencia, cuya localización está bien defina, puesto que corresponden a puntos prominentes del rostro, los puntos Zygion son localizados por los antropólogos *in situ*, mediante palpación directa del rostro del sujeto, con el fin de percibir a través de la piel, el punto más lateral del hueso cigomático de la cara (Kolar and Salter, 1997). En el caso de los modelos 3D, la localización de estos puntos se decidió según el criterio del usuario, tras observar el modelo virtual desde diferentes puntos de vista. El ruido presente en los modelos adquiridos mediante estereovisión con luz estructurada, aunado a la variación en la postura de los sujetos durante la adquisición 3D, dificultó la selección unívoca de esos puntos.

En la (Figura 113) se presenta el error técnico de medición en cada punto de referencia de manera visual, utilizando esferas cuyo radio es proporcional al error. Se observa que, en la región de la boca, los puntos sobre el borde del labio superior e inferior, así como en la unión de los labios, presenta un error alto, debido a que visualmente, estos bordes no están bien definidos en la textura de los modelos. Lo mismo sucede con los puntos de la nariz, esto último, debido a que los modelos de la base de datos, presentan ruido y asimetrías en esas zonas, como consecuencias de los artefactos del algoritmo de reconstrucción 3D. Después de los puntos Zygion, el punto con mayor error corresponde al Gnathion, localizado en la parte inferior de la barbilla. En algunos modelos, este punto fue difícil de localizar, ya que no estaba bien definida la zona de la barbilla, debido a las limitaciones del sistema de adquisición.

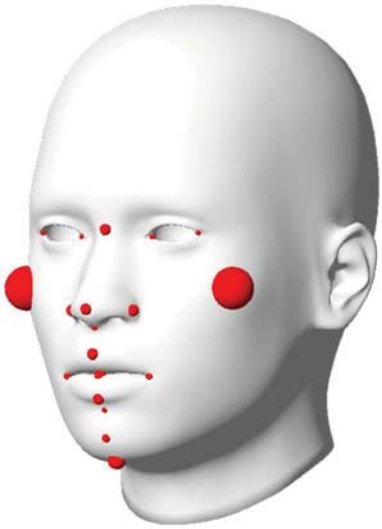


Figura 113. Error técnico de medición visualizado como esferas sobre el modelo de referencia, para fines ilustrativos, el radio de las esferas es 3 veces mayor que el error de medición.

6.5 METAMORFOSIS DE ROSTROS

Como tarea adicional a los objetivos de este trabajo, se realizó la interpolación de los rostros, utilizando los modelos resultantes del registro afín, combinado con registro deformable basado en jaulas, con los colores trasferidos desde las mallas originales. Debido a que, en todos los casos, fue el modelo de referencia el que se registró al modelo adquirido de los sujetos reales, la topología de estas mallas, es la misma, esto es, tienen el mismo número de vértices y caras, organizados

internamente en el mismo orden. Con lo cual la interpolación entre dos modelos diferentes, se logra simplemente interpolando la posición y los atributos de vértices correspondientes, manteniendo la misma conectividad de las aristas y de las caras. Sea a_i un atributo (color, posición, vector normal, etc.) de la malla A, y sea b_i el mismo atributo de la malla B muestreado sobre el vértice con el mismo índice. La función de interpolación lineal aplicada para obtener los nuevos atributos c_i, de cada vértice de la nueva malla C es:

$$c_i(\alpha) = (1-\alpha)a_i + \alpha b_i \tag{122}$$

En donde $\alpha \in [0,\ 1]$, es un factor de mezcla. Aplicando esta función de interpolación sobre cada atributo que describe la malla, se obtiene una nueva malla C, cuyas propiedades son intermedias a las mallas A y B, de manera más general:

$$C(\alpha) = (1-\alpha)A + \alpha B \tag{123}$$

Con la observación de que $C(0) = A$ y $C(1) = B$. Esto puede extenderse a más de dos modelos como:

$$C(\mathbf{W}) = \sum_{i=1}^{n} w_i X_i \tag{124}$$

Siendo $\mathbf{W} = \{w_1, \cdots, w_i, \cdots, w_n\}$ un conjunto de pesos, que pondera la contribución de cada malla X_i a la malla resultante C, tal que:

$$\sum_{i=1}^{n} w_i = 1 \tag{125}$$

Para el caso en que $w_i = 1/n$, el resultado obtenido es el promedio de las n mallas. En la (Figura 114) se muestra el resultado visual de realizar la interpolación entre tres rostros diferentes. Se observa que, entre dos rostros cercanos, la diferencia es casi imperceptible, pero si se observan los modelos en los extremos, estos son muy diferentes, tanto en la forma como en el color.

Figura 114. Metamorfosis realizada entre tres rostros diferentes.

7 CONCLUSIONES

En este trabajo se desarrolló un software para la selección interactiva de puntos de referencia sobre modelos 3D de rostros humanos, correspondientes a rasgos faciales, como una alternativa a la antropometría directa. El software desarrollado implementa métodos de visualización científica, mediante el uso de mapas de color divergentes, para visualizar las curvaturas sobre la superficie de la malla. Adicionalmente, se integró al software un dispositivo háptico, permitiendo una interacción más intuitiva con la geometría del modelo, mediante retroalimentación de fuerzas. Para esto último fue necesario implementar un controlador de posición PID basado en fuerza.

Utilizando este software se realizó un estudio sobre una población reducida, formada por un conjunto de 35 modelos de cabezas, obtenidas mediante estereovisión con luz estructurada, pertenecientes a sujetos de nacionalidad mexicana. Como resultado del análisis de componentes principales, se obtuvo la forma promedio del rostro de la población de estudio, así como los modos de variación, obteniéndose un modelo de distribución de puntos, que describe a los puntos de referencia de la población de estudio. Con estos datos fue posible generar rostros aleatorios con la misma distribución estadística que el conjunto analizado.

Para realizar la selección interactiva de los puntos de referencia, sobre mallas 3D complejas, se implementó un algoritmo basado en una estructura de árbol octal, para la detección de colisiones. Dicha implementación resultó tener un rendimiento similar, a la de los motores de física comerciales, con los cuales se realizó el análisis comparativo. Esta misma estructura sirvió para acelerar la prueba de inclusión de un punto dentro de una malla triangular cerrada, así como para encontrar el punto más cercano sobre la superficie. Con ello fue posible obtener el campo de distancia euclidiana, el cual fue utilizado como herramienta para la validación del método de registro deformable propuesto en este trabajo.

En este trabajo, se propuso un método de registro deformable basado en jaulas partiendo de un registro afín. El error en la alineación producido por este método, es inferior al error obtenido mediante registro rígido. Aplicando este método de registro a un modelo de referencia de la cabeza humana, fue posible aproximar la forma original de los rostros adquiridos mediante estereovisión, usando únicamente como referencia, el conjunto de puntos de referencia. Con este mismo método se logró obtener una forma aproximada del rostro promedio, así como realizar interpolación entre más de dos rostros distintos.

Aunque la principal aplicación del software es en análisis antropométrico, es posible utilizar las mismas herramientas para analizar otro tipo de estructuras, debido a la robustez del método de selección interactiva desarrollado, y a la flexibilidad del método de registro basado en jaulas.

7.1 TRABAJO A FUTURO

En este trabajo se abordó el problema de la selección de puntos de referencia craneofaciales, enfocándonos en dar una solución interactiva, y limitándonos al análisis de mallas triangulares estáticas. Un posible trabajo a futuro es el de modificar el software de selección (Picking interactivo) con el fin de ser capaces de trabajar directamente sobre volúmenes sin necesidad de tener la reconstrucción. Una posible forma de hacerlo es aprovechar la capacidad de las interfaces hápticas

de poder explorar el espacio 3D, los problemas a resolver serían en cuanto a la visualización y a la retroalimentación de fuerza en tiempo real. Otro posible trabajo a futuro es el de incorporar la deformación de la malla triangular que representa la piel del rostro, con el fin de ser capaces de reproducir la sensación de palpación de los huesos craneales debajo de la piel. Para ello sería necesario contar con el modelo 3D del cráneo y considerar las propiedades mecánicas de la piel y los tejidos subcutáneos. En esta última propuesta el principal reto es la deformación en tiempo real de los tejidos, así como la retroalimentación de fuerza.

8 ANEXOS

8.1 PUNTOS DE REFERENCIA

8.1.1 Clasificación según Kolar y Salter

Según (Kolar and Salter, 1997) los puntos de referencia de la cabeza humana pueden clasificarse como sigue.

8.1.1.1 Cabeza

Nombre	Abreviatura	Descripción
Euryon	eu	El punto más lateral de la cabeza.
Frontotemporale	ft	El punto más profundo de la cresta temporal en el hueso frontal sobre la línea de las cejas.
Frontozygomaticus	fz	El punto más lateral del cigomático.
Glabella	g	Punto más prominente en el plano sagital medio entre las cejas.
Ophyron	on	El punto en el plano medio, de una línea tangente a los límites superiores de las cejas.
Opisthocranion	op	El punto más prominente en la parte posterior del occipital.
Porion	po	El punto superior del meato auditivo.
Tragion	t	Localizado en la parte superior del cartílago proyectado al exterior del canal auditivo.
Trichion	tr	Punto medio de la línea del pelo.
Vertex	v	Punto más alto de la cabeza.

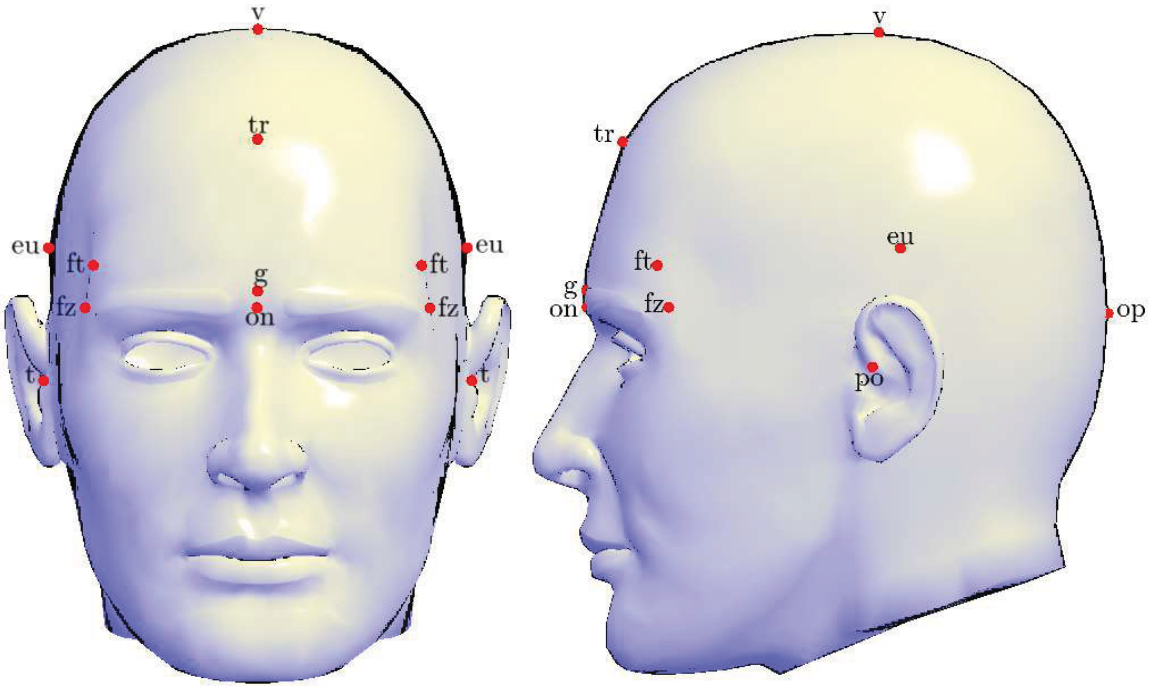


Figura 115. Puntos de referencia de la cabeza.

8.1.1.2 Rostro

Nombre	Abreviatura	Descripción
Condylion laterale	cdl	El punto más lateral del cóndilo mandibular.
Gnathion	gn	El punto más bajo de la mandíbula.
Gonion	go	El punto más lateral en el ángulo de la mandíbula.
Nasion	n	El punto medio de la sutura nasal-frontal
Pogonion	pg	El punto más prominente de la barbilla.
Sublabiale	sl	El punto más profundo entre el labio inferior y el mentón visto de perfil.
Subnasale	sn	La unión entre la división de los orificios nasales y el labio superior.
Stomion	sto	El punto medio de la fisura labial cuando los labios están cerrados.
Zygion	zy	El punto más lateral sobre el arco cigomático.

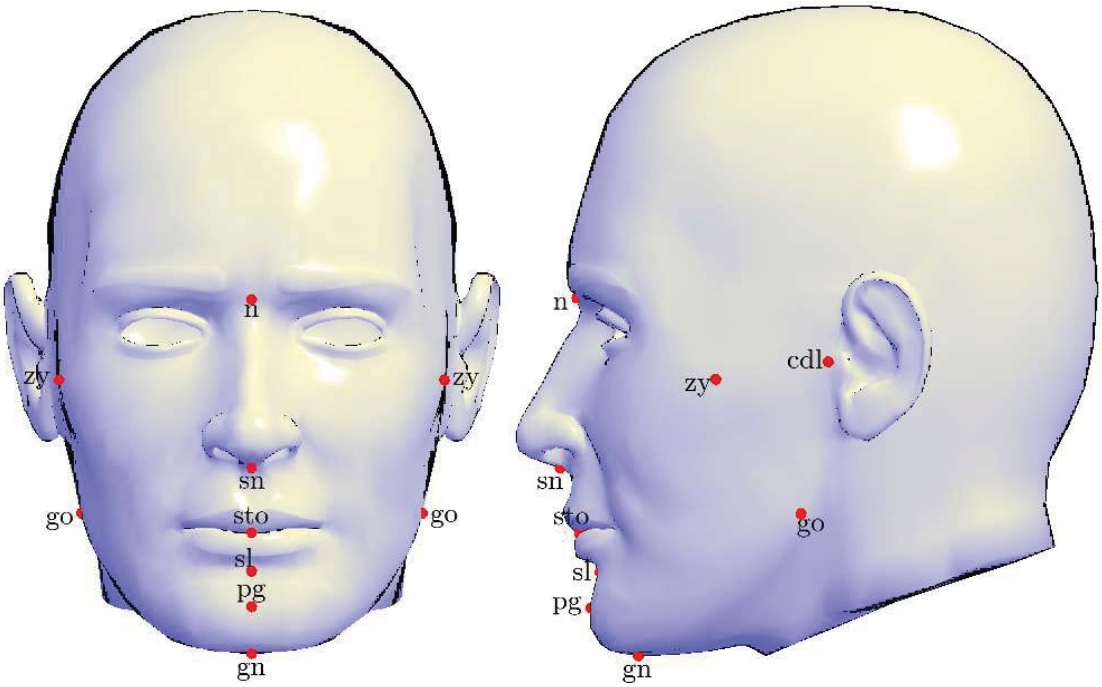


Figura 116. Puntos de referencia del rostro.

8.1.1.3 Órbitas

Nombre	Abreviatura	Descripción
Endocanthion	en	Las esquinas internas de las fisuras de los ojos.
Exocanthion	ex	Las esquinas externas de las fisuras de los ojos.
Orbitale	or	El punto más bajo en el margen de la órbita.
Orbitale superius	os	El punto más alto en el margen de la órbita.
Palpebrale inferius	pi	El punto más bajo del borde del párpado inferior.
Palpebrale superius	ps	El punto más alto del borde del párpado superior.
Superciliare	sci	El punto más alto del margen superior de las cejas.

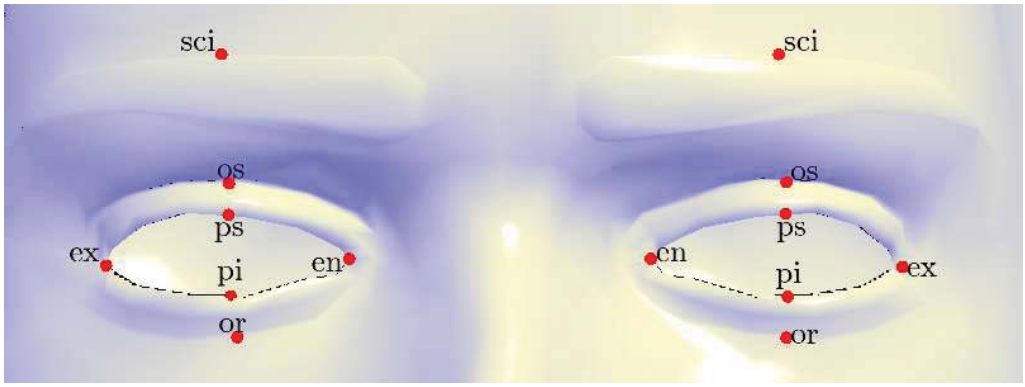


Figura 117. Puntos de referencia de las órbitas.

8.1.1.4 Nariz

Nombre	Abreviatura	Descripción
Punto de curvatura alar	ac	El punto más lateral en la curvatura de la base del ala nasal.
Alare	al	El punto más lateral del ala nasal.
Columella apex	c’	El punto más alto de los orificios nasales.
Maxillofrontale	mf	El punto más prominente de la sutura frontal-nasal.
Pronasale	prn	El punto más prominente de la punta de la nariz.
Sellion	s	El punto más profundo del ángulo nasal-frontal.
Subalare	sbal	El punto más bajo del margen de la base del ala nasal.

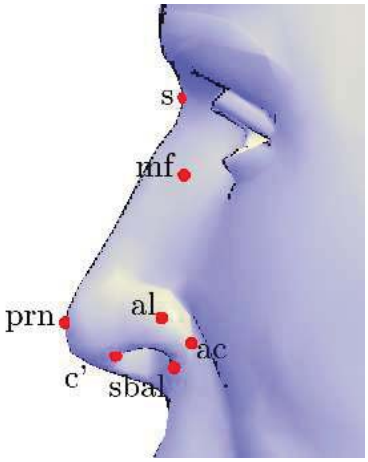


Figura 118. Puntos de referencia de la nariz.

8.1.1.5 Orales-Labiales

Nombre	Abreviatura	Descripción
Cheilion	ch	La esquina externa de la boca.
Crista philtri	cph	El punto en la cresta del surco nasal-labial.
Labiale inferius	li	El punto medio del borde bermellón del labio inferior.
Labiale superius	ls	El punto medio del borde bermellón del labio superior.
Labiale superious lateralis	ls'	El punto en el borde del bermellón del labio superior debajo del subalare (sbal).

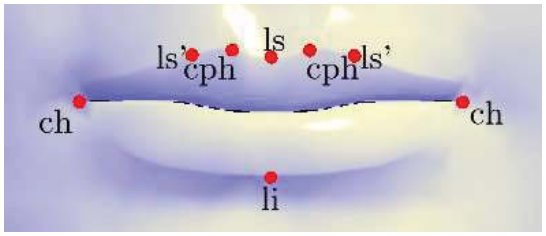


Figura 119. Puntos de referencia orales-labiales.

8.1.1.6 Orejas

Nombre	Abreviatura	Descripción
Otobasion inferius	obi	El punto más bajo de la unión de la cabeza y la oreja.
Otobasion superius	obs	El punto más alto de la unión de la cabeza y la oreja.
Postaurale	pa	El punto posterior en la hélice de la oreja.
Preaurale	pra	El punto sobre obi-obs opuesto al postaurale.
Superaurale	sa	El punto más alto de la hélice de la oreja.
Subaurale	sba	El punto más bajo del lóbulo de la oreja.

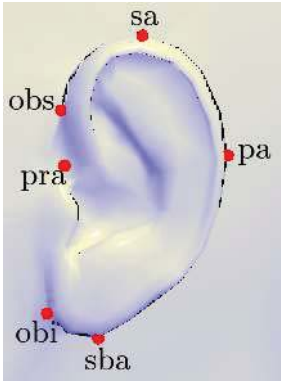


Figura 120. Puntos de referencia de las orejas.

8.1.2 Clasificación según Robert M. George

Según (George, 2007) los puntos de referencia de la cabeza humana pueden clasificarse como sigue.

8.1.2.1 Puntos craneales

Nombre	Abreviatura	Descripción	Observaciones
Vertex	v	El ápice en la línea media del neurocráneo.	Igual que en (Kolar and Salter, 1997).
Supraglabella	sg	Un punto arbitrario aproximadamente una pulgada sobre la glabela.	Término introducido por (George, 2007).
Glabella	g	El punto calvo entre las cejas sobre el plano sagital medio.	La definición es diferente a la de (Kolar and Salter, 1997), por lo que cambia de posición.
Euryon	eu	El punto más lateral del neurocráneo en el área parietal-temporal.	Igual que en (Kolar and Salter, 1997).
Auriculotemporale	at	La unión superior de la oreja con el lado de la cabeza.	Equivalente a Otobasion superius (obs) en (Kolar and Salter, 1997).

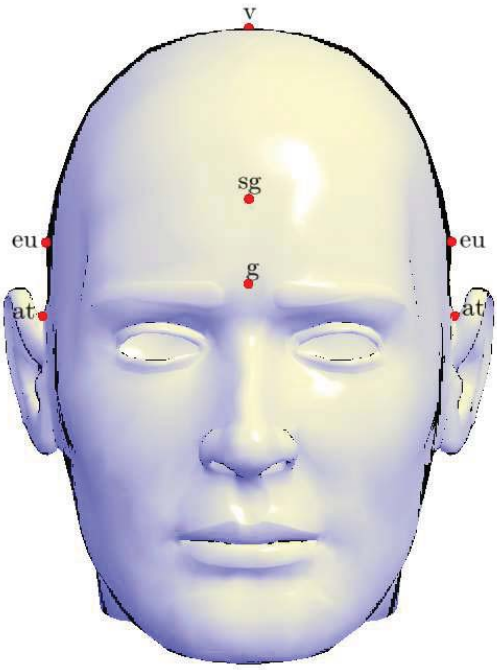


Figura 121. Puntos craneales.

8.1.2.2 Puntos laterales

Nombre	Abreviatura	Descripción	Observaciones
Zygion	zy	El punto más ancho de las mejillas visto frontalmente.	Igual que en (Kolar and Salter, 1997).
Gonion	go	El punto más ancho de la mandíbula inferior visto frontalmente.	Igual que en (Kolar and Salter, 1997).

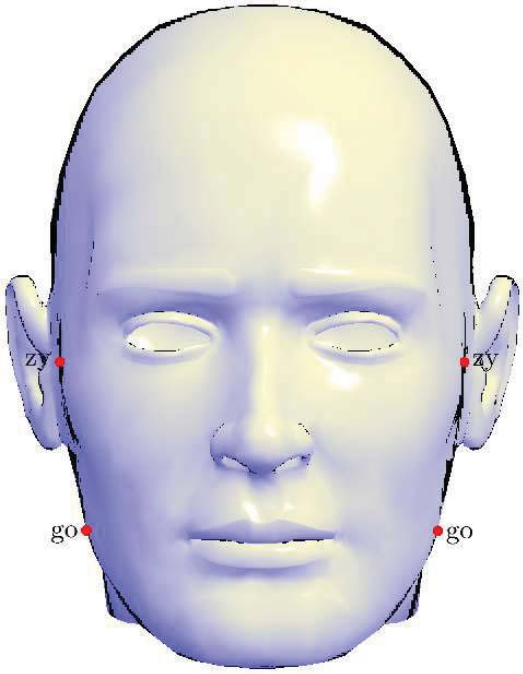


Figura 122. Puntos laterales.

8.1.2.3 Puntos orbitales

Nombre	Abreviatura	Descripción	Observaciones
Ectocanthion	ec	La esquina (ángulo) lateral del ojo.	En (Kolar and Salter, 1997) es llamado Exocanthion (ex).
Endocanthion	en	La esquina (ángulo) medial del ojo.	Igual que en (Kolar and Salter, 1997).
Iridion laterale	il	El punto más lateral de la circunferencia del iris.	Término introducido por (George, 2007).
Iridion mediale	im	El punto más medial en la circunferencia del iris.	Término introducido por (George, 2007).

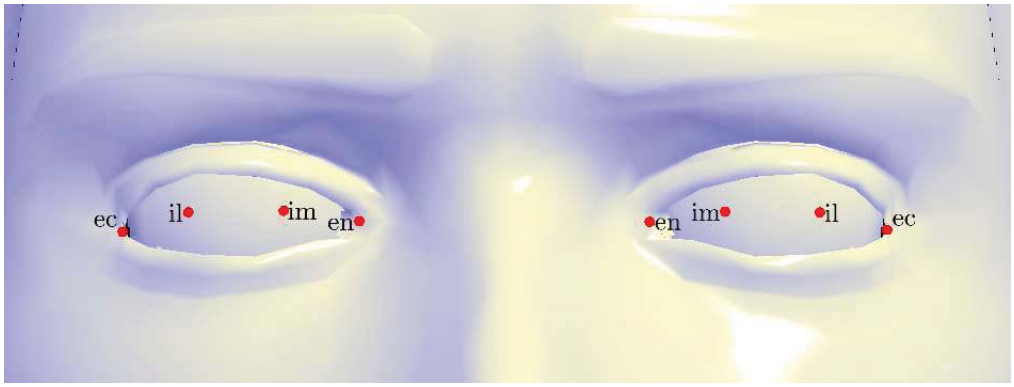


Figura 123. Puntos orbitales.

8.1.2.4 Puntos nasales

Nombre	Abreviatura	Descripción	Observaciones
Nasion	n	En la vista lateral, el ápice del ángulo frontal-nasal.	Difiere de la definición de (Kolar and Salter, 1997), aparece en el lugar del Sellion (s).
Nasale	na	El punto medio del puente nasal de hueso.	Término introducido por (George, 2007).
Pronasale	prn	La punta de la nariz vista lateralmente.	Igual que en (Kolar and Salter, 1997).
Subnasale	sn	El punto más bajo de la nariz en el plano sagital medio.	Igual que en (Kolar and Salter, 1997).
Alare	al	El punto más lateral del ala de la nariz.	Igual que en (Kolar and Salter, 1997).

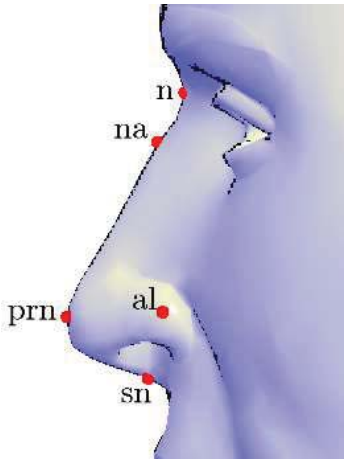


Figura 124. Puntos nasales.

8.1.2.5 Puntos labiales

Nombre	Abreviatura	Descripción	Observaciones
Superior labial sulcus	sls	El punto de máxima indentación del labio superior.	Término introducido por (George, 2007).
Labiale superius	ls	La intersección del plano sagital medio con el borde del bermellón del labio superior.	Igual que en (Kolar and Salter, 1997).
Stomion	sto	La intersección del plano sagital medio con la fisura labial.	Igual que en (Kolar and Salter, 1997).
Labiale inferius	li	La intersección del plano sagital medio con el borde del bermellón del labio inferior.	Igual que en (Kolar and Salter, 1997).
Chelion	ch	La esquina de la boca.	En (Kolar and Salter, 1997) es deletreado como Cheilion (ch).

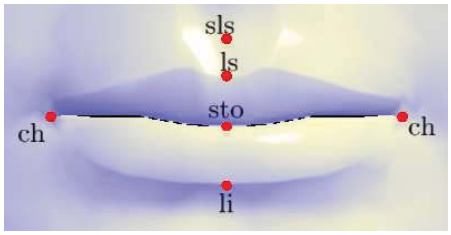


Figura 125. Puntos labiales.

8.1.2.6 Puntos del mentón

Nombre	Abreviatura	Descripción	Observaciones
Labiomentale	lm	La intersección del plano sagital medio con surco del labio inferior.	Corresponde a Sublabiale (sl) en (Kolar and Salter, 1997).
Pogonion	pog	El punto más anterior sobre la barbilla visto lateralmente.	Igual que en (Kolar and Salter, 1997).
Gnathion	gn	El punto más bajo de la barbilla sobre el plano sagital medio.	Igual que en (Kolar and Salter, 1997).

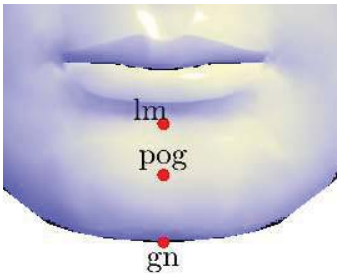


Figura 126. Puntos del mentón.

8.1.2.7 Puntos auriculares

Nombre	Abreviatura	Descripción	Observaciones
Superaurale	sa	El polo superior de la hélice.	Igual que en (Kolar and Salter, 1997).
Subaurale	sba	El polo inferior del lóbulo.	Igual que en (Kolar and Salter, 1997).
Preaurale	pra	El punto más anterior de la oreja en la base del tragus.	La definición es diferente a la de (Kolar and Salter, 1997), por lo que cambia de posición.
Postaurale	pa	El punto más posterior de la hélice.	Igual que en (Kolar and Salter, 1997).
Tragion	tr	El ápice del tragus.	La abreviatura y definición es diferente a la de (Kolar and Salter, 1997), por lo que cambia de posición.

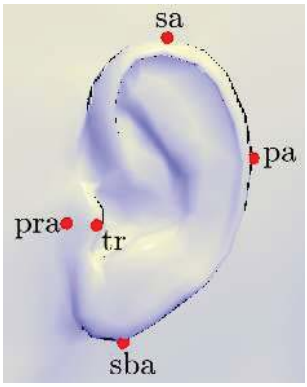


Figura 127. Puntos auriculares.

8.1.3 Puntos de referencia en otros trabajos

En (Enciso et al., 2003), los autores realizaron antropometría de la cabeza en 3D mediante luz estructurada sobre un maniquí de referencia utilizando los siguientes puntos de referencia:

- Cabeza: glabella (g), trichion (tr), frontotemporale (ft).
- Rostro: zygion (zy), gonion (go), sublabiale (sl), pogonion (pg), gnathion (gn), condylion laterale (cdl).
- Órbitas: endocanthion (en), exocanthion (ex).
- Nariz: nasion (n), pronasale (prn), subnasale (sn), subalare (sbal), punto de curvatura alar (ac).
- Labios y boca: crista philtri (cph), cheilion (ch), stomion (sto), labiale superius (ls), labiale inferius (li).
- Orejas: otobasion inferius (obi), otobasion superius (pobs), superaurale (sa), subaurale (sba), postaurale (pa), preaurale (pra).

En (Scheenstra, 2005) se realizó morfometría sobre rostros en 3D para reconocimiento facial utilizando adquisiciones 3D de cuerpo completo realizadas por el proyecto CAESAR (Civilian American and European Surface Anthropometry Resource Project) en donde los siguientes puntos de referencia de la cabeza fueron extraídos automáticamente:

- Sellion (se).
- Infraorbitale (io). Corresponde al punto Orbitale (or) en (Kolar and Salter, 1997).
- Supramenton (sp). Corresponde al punto Pogonion (pg) en (Kolar and Salter, 1997).
- Tragion (tr). Corresponde a la definición de Tragion (t) en (Kolar and Salter, 1997).
- Gonion (go).
- Nuchale (n). Punto más bajo del occipital palpado en los músculos de la nuca.

En (Calignano and Vezzetti, 2010) se analizaron características geométricas sobre escaneos 3D del rostro para realizar diagnósticos en cirugía maxilofacial utilizando el siguiente conjunto de puntos de referencia: Nasion (n), Pronasale (prn), Subnasale (sn), Labiale superius (ls), Stomion (sto), Labiale inferius (li), Sublabiale (sls), Pogonion (pog), Tragion (t), Cresta alar nasal (al), Cheilion (ch), Gonion (go), Vertex (v).

En el trabajo de (Perakis et al., 2010) sobre extracción automática de puntos de referencia y registro de rostros en 3D, se trabajó con un conjunto reducido de puntos (Figura 128): esquina exterior del ojo derecho (1), esquina interior del ojo derecho (2), esquina interior del ojo izquierdo (3), esquina exterior del ojo izquierdo (4), punta de la nariz (5), esquina derecha de la boca (6), esquina izquierda de la boca (7) y punta de la barbilla (8).

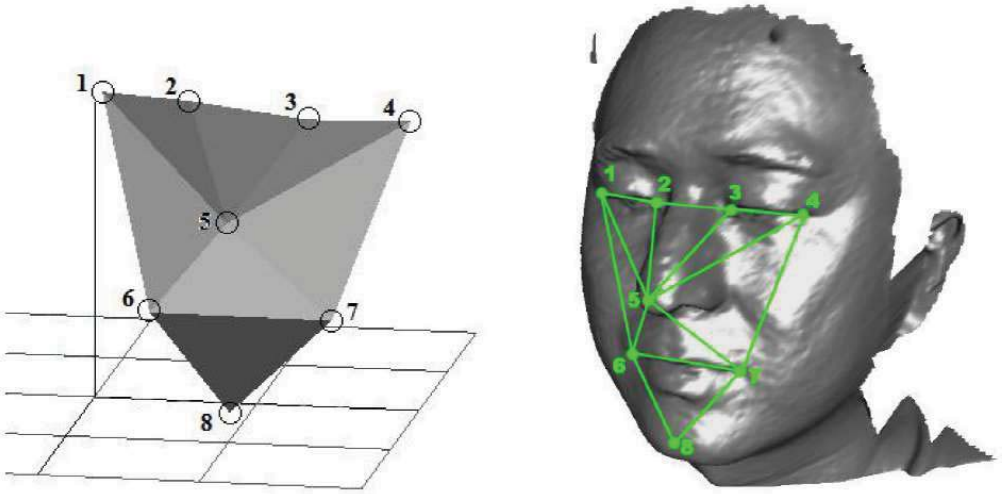


Figura 128. Puntos de referencia. Extraído de (Perakis et al., 2010).

8.2 CÓDIGOS DE GLSL

8.2.1 Modelo de iluminación de Lambert

```
layout(location = 0)in vec3 aPosition;
layout(location = 1)in vec3 aNormal;

out vec3 vNormal;

uniform mat4 uMVPMatrix;
uniform mat3 uNMatrix;

void main()
{
	vNormal = uNMatrix * aNormal;

	gl_Position = uMVPMatrix * vec4(aPosition, 1.0);
}
```

Código 1 Código en lenguaje GLSL del programa sombreador de vértices para el modelo de iluminación de Lambert.

```
in vec3 vNormal;

layout(location = 0) out vec4 fFragColor;

void main()
{
	vec3 light = normalize(vec3(1.0, 2.0, 2.0));
	vec3 normal = normalize(vNormal);
	float diffuse = max(0.0, dot(light, normal));
	fFragColor = vec4(vec3(diffuse), 1.0);
}
```

Código 2 Código en lenguaje GLSL del programa sombreador de fragmentos para el modelo de iluminación de Lambert.

8.2.2 Modelo de iluminación de Phong

```
layout(location = 0)in vec3 aPosition;
layout(location = 1)in vec3 aNormal;

out vec3 vPosition;
out vec3 vNormal;

uniform mat4 uMVPMatrix;
uniform mat4 uMVMatrix;
uniform mat3 uNMatrix;

void main()
{
	vPosition = (uMVMatrix * vec4(aPosition, 1.0)).xyz;
	vNormal = uNMatrix * aNormal;

	gl_Position = uMVPMatrix * vec4(aPosition, 1.0);
}
```

Código 3 Código en lenguaje GLSL del programa sombreador de vértices para el modelo de iluminación de Phong.

```
in vec3 vPosition;
in vec3 vNormal;

uniform bool uPerspective;

layout(location = 0) out vec4 fFragColor;

void main()
{
	vec3 light = normalize(vec3(1.0, 2.0, 2.0));
	vec3 normal = normalize(vNormal);
	float diffuse = max(0.0, dot(light, normal));
	vec3 eye = uPerspective ? normalize(-vPosition) : vec3(0.0f, 0.0f, 1.0f);
	vec3 reflection = reflect(-light, normal);
	float specular = pow(max(0.0, dot(reflection, eye)), 100.0);
	fFragColor = vec4(vec3(diffuse + specular), 1.0);
}
```

Código 4 Código en lenguaje GLSL del programa sombreador de fragmentos para el modelo de iluminación de Phong.

8.2.3 Modelo de iluminación de Gooch

```
layout(location = 0)in vec3 aPosition;
layout(location = 1)in vec3 aNormal;

out vec3 vPosition;
out vec3 vNormal;

uniform mat4 uMVPMatrix;
uniform mat4 uMVMatrix;
uniform mat3 uNMatrix;

void main()
{
	vPosition = (uMVMatrix * vec4(aPosition, 1.0)).xyz;
	vNormal = uNMatrix * aNormal;

	gl_Position = uMVPMatrix * vec4(aPosition, 1.0);
}
```

Código 5. Código en lenguaje GLSL del programa sombreador de vértices para el modelo de iluminación de Gooch.

```
in vec3 vPosition;
in vec3 vNormal;

uniform bool uPerspective;

layout(location = 0) out vec4 fFragColor;

void main()
{
	float b = 0.8;
	float y = 0.2;
	vec3 k_blue = vec3(0.0, 0.0, b);
	vec3 k_yellow = vec3(y, y, 0.0);
	float alpha = 0.2;
	float beta = 0.8;
	vec3 k_d = vec3(1.0, 0.9, 0.8);
	vec3 k_cool = k_blue + alpha * k_d;
	vec3 k_warm = k_yellow + beta * k_d;
	const vec3 l = normalize(vec3(1.0, 2.0, 2.0));
	vec3 n = normalize(vNormal);
	float temperature = 0.5 + 0.5 * dot(l, n);
	vec3 e = uPerspective ? normalize(-vPosition) : vec3(0.0f, 0.0f, 1.0f);
	vec3 r = reflect(-l, n);
	float specular = pow(max(0.0, dot(r, e)), 10.0);
	fFragColor = vec4(mix(k_cool, k_warm, temperature) + 0.1 * specular, 1.0);
}
```

Código 6. Código en lenguaje GLSL del programa sombreador de fragmentos para el modelo de iluminación de Gooch.

8.3 CÓDIGOS DE MATLAB

8.3.1 Registro rígido

```
function [R_q_R, q_T] = rigid_registration(P, X)

Sigma_px = ((P - repmat(mean(P), length(P), 1))' * (X - repmat(mean(X),
length(X), 1))) / (length(P) - 1);

A = Sigma_px - Sigma_px';

[V, ~] = eig([trace(Sigma_px), [A(2, 3), A(3, 1), A(1, 2)]; [A(2, 3);
A(3, 1); A(1, 2)], Sigma_px + Sigma_px' - trace(Sigma_px) * eye(3)]);

R_q_R = quat2rotm(V(:, 4)');
q_T = mean(X) - mean(P) * R_q_R';
```

Código 7. Script de MATLAB para realizar el registro rígido.

8.3.2 Registro afín

```
function [R, t] = non_rigid_registration(P, X)

A = [P'; ones(1, length(P))]';
T = [A \ X(:, 1), A \ X(:, 2), A \ X(:, 3)];

R = T(1 : 3, :)';
t = T(4, :);
```

Código 8. Script de MATLAB para realizar el registro afín.

9 BIBLIOGRAFÍA

AHN, S. H. 2008. *OpenGL Projection Matrix* [Online]. Available: http://www.songho.ca/opengl/gl_projectionmatrix.html [Accessed September 6 2015].

AKENINE-MÖLLER, T. 2001. *AABB-triangle overlap test code* [Online]. Available: http://fileadmin.cs.lth.se/cs/Personal/Tomas_Akenine-Moller/code/tribox2.txt [Accessed August 2016].

AKENINE-MÖLLER, T. Fast 3D triangle-box overlap testing. ACM SIGGRAPH 2005 Courses, 2005. ACM, 8.

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer,* 19**,** 105-114.

ALLEN, B., CURLESS, B. & POPOVI, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.,* 22**,** 587-594.

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., L, B., #233, VY & DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph.,* 22**,** 485-493.

ANG, K. H., CHONG, G. & LI, Y. 2005. PID control system analysis, design, and technology. *IEEE transactions on control systems technology,* 13**,** 559-576.

ANSARI, N., ABDEL-MOTTALEB, M. & MAHOOR, M. H. 3D Face Mesh Modeling from Range Images for 3D Face Recognition. Image Processing, 2007. ICIP 2007. IEEE International Conference on, Sept. 16 2007-Oct. 19 2007 2007. IV - 509-IV - 512.

BESL, P. J. & MCKAY, N. D. 1992. A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 14**,** 239-256.

BOOKSTEIN, F. L. 1997. *Morphometric tools for landmark data: geometry and biology*, Cambridge University Press.

BORLAND, D. & TAYLOR, R. M. 2007. Rainbow Color Map (Still) Considered Harmful. *Computer Graphics and Applications, IEEE,* 27**,** 14-17.

BREWER, C. 2004. *Designing Better Maps: A Guide for Gis Users*, Environmental Systems Research.

CALIGNANO, F. & VEZZETTI, E. 2010. Soft Tissue Diagnosis in Maxillofacial Surgery: A Preliminary Study on Three-Dimensional Face Geometrical Features-Based Analysis. *Aesthetic Plastic Surgery,* 34**,** 200-211.

CARNICKY, J. & JR., D. C. 2006. Three-dimensional measurement of human face with structured-light illumination. *Measurement Science Review,* 6**,** 1.

CAZALS, F. & POUGET, M. 2003. Estimating differential quantities using polynomial fitting of osculating jets. *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing.* Aachen, Germany: Eurographics Association.

COHEN-STEINER, D. & MORVAN, J.-M. 2003. Restricted delaunay triangulations and normal cycle. *Proceedings of the nineteenth annual symposium on Computational geometry.* San Diego, California, USA: ACM.

DORAI, C. & JAIN, A. K. 1997. COSMOS-A representation scheme for 3D free-form objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 19**,** 1115-1130.

DRYDEN, I. L. & MARDIA, K. V. 1998. *Statistical shape analysis*, John Wiley & Sons, Chichester.

EL-HUSSUNA, A. 2003. *Statistical variation of Three Dimensional face models.* Master Master Thesis project, IT-University of Copenhagen.

ENCISO, R., SHAWA, A., NEUMANN, U. & MAH, J. 2003. 3D head anthropometric analysis.

FLOATER, M. S. 2003. Mean value coordinates. *Comput. Aided Geom. Des.,* 20**,** 19-27.

FLOATER, M. S., KÓS, G. & REIMERS, M. 2005. Mean value coordinates in 3D. *Computer Aided Geometric Design,* 22**,** 623-631.

FRISANCHO, A. R. 1990. *Anthropometric standards for the assessment of growth and nutritional status*, University of Michigan Press.

GAO, Y., HAO, A., ZHAO, Q. & DOGSON, N. A. 2009. Skin-detached surface for interactive large mesh editing. *Technical Report* University of Cambridge.

GEORGE, R. M. 2007. *Facial Geometry: Graphic Facial Analysis for Forensic Artists*, Charles C. Thomas.

GOLDFEATHER, J. & INTERRANTE, V. 2004. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.,* 23**,** 45-63.

GOOCH, A., GOOCH, B., SHIRLEY, P. & COHEN, E. 1998. A non-photorealistic lighting model for automatic technical illustration. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* ACM.

GOURAUD, H. 1971. Continuous Shading of Curved Surfaces. *IEEE Trans. Comput.,* 20**,** 623-629.

GUPTA, S., MARKEY, M. & BOVIK, A. 2010. Anthropometric 3D Face Recognition. *International Journal of Computer Vision,* 90**,** 331-349.

HAMEIRI, E. & SHIMSHONI, I. 2003. Estimating the principal curvatures and the Darboux frame from real 3-D range data. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 33**,** 626-637.

HO, H. T. & GIBBINS, D. 2009. Curvature-based approach for multi-scale feature extraction from 3D meshes and unstructured point clouds. *Computer Vision,* 3**,** 201 - 212.

JIMÉNEZ, J. J., OGÁYAR, C. J., NOGUERA, J. M. & PAULANO, F. 2014. Performance Analysis for GPU-based Ray-triangle Algorithms. *International Conference on Computer Graphics Theory and Applications.*

JIMÉNEZ, J. J., SEGURA, R. J. & FEITO, F. R. 2010. A robust segment/triangle intersection algorithm for interference tests. Efficiency study. *Computational Geometry,* 43**,** 474-492.

JU, T., SCHAEFER, S. & WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.,* 24**,** 561-566.

KAUSHIK, V. D., PATHAK, V. K. & GUPTA, P. 2010. Geometric Modeling of 3D-Face Features and Its Applications. *JOURNAL OF COMPUTERS,* 5**,** 1-10.

KOENDERINK, J. J. & VAN DOORN, A. J. 1992. Surface shape and curvature scales. *Image and Vision Computing,* 10**,** 557-564.

KOLAR, J. C. & SALTER, E. M. 1997. *Craniofacial Anthropometry Practical Measuremet of the Head and Face for Clinical, Surgical and Research Use,* Springfield, Illinois, U.S.A., Charles C Thomas Publisher, LTD.

KOPPERMAN, R., MEYER, P. R. & WILSON, R. G. 1991. A Jordan surface theorem for three-dimensional digital spaces. *Discrete & Computational Geometry,* 6**,** 155-161.

LIANG, S., WU, J., WEINBERG, S. M. & SHAPIRO, L. G. Improved Detection of Landmarks on 3D Human Face Data. Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2013.

LOÏC, S. B. 2007. *Automatic and adaptative registration of a 3D database of human heads for anthropometric applications.* UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

LÓPEZ, L. 2015. *Morfometría facial en poblaciones sanas mediante un sistema de estereovisión.* Tesis de Maestría, Universidad Nacional Autónoma de México, Posgrado en Ciencias Físicas.

MAHMOOD, S. A., GHANI, R. F. & KERIM, A. A. 2013. Nose Tip Detection Using Shape index and Energy Effective for 3d Face Recognition. *International Journal of Modern Engineering Research (IJMER),* 2013; 3**,** 3086–3090.

MAJID, Z., CHONG, A. K., AHMAD, A., SETAN, H. & SAMSUDIN, A. R. 2005. Photogrammetry and 3D Laser scanning as spatial data capture techniques for a national craniofacial database. *The Photogrammetric Record,* 20**,** 48-68.

MARQUEZ, J. A. Enhancing watershed segmentation of touching and weakly-connected features in biomedical images. Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE, 20-25 Aug. 2008 2008. 3095-3098.

MASSIE, T. H. & SALISBURY, J. K. 1994. The phantom haptic interface: A device for probing virtual objects. *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems,* 55**,** 295-300.

MEYER, M., DESBRUN, M., SCHRÖDER, P. & BARR, A. 2003. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *In:* HEGE, H.-C. & POLTHIER, K. (eds.) *Visualization and Mathematics III.* Springer Berlin Heidelberg.

MÖLLER, T. & TRUMBORE, B. 1997. Fast, Minimum Storage Ray-Triangle Intersection. *journal of graphics tools,* 2**,** 21-28.

MORELAND, K. 2009. Diverging Color Maps for Scientific Visualization. *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part II.* Las Vegas, Nevada: Springer-Verlag.

NAIR, P. & CAVALLARO, A. 2009. 3-D face detection, landmark localization, and registration using a point distribution model. *Multimedia, IEEE Transactions on,* 11**,** 611-623.

NIETO, J. & SUSÍN, A. 2013. Cage Based Deformations: A Survey. *In:* GONZÁLEZ HIDALGO, M., MIR TORRES, A. & VARONA GÓMEZ, J. (eds.) *Deformation Models.* Springer Netherlands.

PALMERIUS, K. L. 2007. *Direct Volume Haptics for Visualization.* Master, Linköpings universitet.

PERAKIS, P., PASSALIS, G., THEOHARIS, T. & KAKADIARIS, I. A. 2010. 3D Facial Landmark Detection & Face Registration A 3D Facial Landmark Model & 3D Local Shape Descriptors Approach.

PHONG, B. T. 1975. Illumination for computer generated pictures. *Commun. ACM,* 18**,** 311-317.

POÓ, M., BUYSEE, E., CARMONA, R., COTO, E. & NAVARRO, H. 2013. GPU-accelerated Polyp Detection in Virtual Colonoscopy. *CLEI ELECTRONIC JOURNAL,* 16.

RUGIS, J. & KLETTE, R. 2006. A scale invariant surface curvature estimator. *Advances in Image and Video Technology.* Springer.

RUIZ, M. C. & ILLINGWORTH, J. Automatic landmarking of faces in 3D - ALF^{3D}. Visual Information Engineering, 2008. VIE 2008. 5th International Conference on, July 29 2008-Aug. 1 2008 2008. 41-46.

RUSINKIEWICZ, S. 2004. Estimating Curvatures and Their Derivatives on Triangle Meshes. *Symposium on 3D Data Processing, Visualization, and Transmission.*

RUSPINI, D. C., KOLAROV, K. & KHATIB, O. 1997. The haptic display of complex graphical environments. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co.

SAVOYE, Y. 2012. Iterative cage-based registration for dynamic shape capture. *ACM SIGGRAPH 2012 Posters.* Los Angeles, California: ACM.

SAVOYE, Y. 2013. Iterative cage-based registration from multi-view silhouettes. *Proceedings of the 10th European Conference on Visual Media Production.* London, United Kingdom: ACM.

SCHEENSTRA, A. 2005. *3D Facial Image Comparison Using Landmarks - A study to the discriminating value of the characteristics of 3D facial landmarks and their automated detection.* Master Thesis, Utrecht University, Netherlands Forensic Institute - Institute of Information and Computing Sciences.

SJÖBERG, H. & YLINENPÄÄ, O. 2009. *Collision Detection for Haptic Rendering.* Master Thesis, Umea University, Department Of Computing Science.

SMITH, L. I. 2002. A tutorial on principal components analysis. *Cornell University, USA,* 51**,** 52.

SZELISKI, R. 2010. *Computer Vision: Algorithms and Applications*, Springer London.
TAUBIN, G. 1995. Estimating the tensor of curvature of a surface from a polyhedral approximation. *Proceedings of the Fifth International Conference on Computer Vision.* IEEE Computer Society.
TAYLOR, A. E. F. 2000. *Illumination Fundamentals*, Rensselaer.
THEISEL, H., ROSSL, C., ZAYER, R. & SEIDEL, H.-P. Normal Based Estimation of the Curvature Tensor for Triangular Meshes. Computer Graphics and Applications, October 6-8 2004.
WHITMARSH, T., VELTKAMP, R. C., SPAGNUOLO, M., MARINI, S. & TER HAAR, F. Landmark detection on 3d face scans by facial model registration. 1st international symposium on shapes and semantics, 2006. Citeseer, 71-75.
WHITTED, T. An improved illumination model for shaded display. ACM Siggraph 2005 Courses, 2005. ACM, 4.
WILCOX, R. 2005. Kolmogorov–Smirnov Test. *Encyclopedia of Biostatistics.* John Wiley & Sons, Ltd.
WILLIAMS, A. 2005. *Ray-Box Intersection Algorithm* [Online]. Available: http://www.cs.utah.edu/~awilliam/box/box.tar.gz [Accessed August 16 2016].
WILLIAMS, A., BARRUS, S., MORLEY, R. K. & SHIRLEY, P. 2005. An efficient and robust ray-box intersection algorithm. *ACM SIGGRAPH 2005 Courses.* Los Angeles, California: ACM.
WOOP, S., BENTHIN, C. & WALD, I. 2013. Watertight Ray/Triangle Intersection. *Journal of Computer Graphics Techniques (JCGT),* 2**,** 65-82.
XIAN, C., LIN, H. & GAO, S. 2012. Automatic cage generation by improved OBBs for mesh deformation. *The Visual Computer,* 28**,** 21-33.
ZILLES, C. B. 1995. *Haptic Rendering with the Toolhandle Haptic Interface.* Master Thesis, Massachusetts Institute Of Technology, Mechanical Engineering.